PDF

Выпуск 1.0.0a29

oceanai

апр. 17, 2024

Содержание:

OCEAN-AI - библиотека с открытым исходным кодом, состоящая из набора алгоритмов интеллектуального анализа поведения человека на основе его мультимодальных данных для автоматического оценивания уровня отдельных персональных качеств личности человека (ПКЛЧ). Библиотека оценивает 5 ПКЛЧ: Открытость опыту (**O**penness), Добросовестность (**C**onscientiousness), Экстраверсия (**E**xtraversion), Доброжелательность (**A**greeableness), Эмоциональная стабильность (Non-**N**euroticism).

OCEAN-AI включает четыре основных алгоритма:

1. Алгоритм анализа аудиоинформации (ААИ).
2. Алгоритм анализа видеоинформации (АВИ).
3. Алгоритм анализа текстовой информации (АТИ).
4. Алгоритм мультимодального объединения информации (МОИ).

Алгоритмы ААИ, АВИ и АТИ реализуют функции сильного искусственного интеллекта (ИИ) в части комплексирования акустических, визуальных и текстовых признаков, построенных на различных принципах (экспертных и нейросетевых), т.е. данные алгоритмы реализуют подходы композитного (гибридного) ИИ. В алгоритмах осуществляется необходимая предобработка аудио-, видео- и текстовой информации, вычисление акустических, визуальных и текстовых признаков и выдача гипотез предсказаний по ним.

Алгоритм МОИ является связующим звеном трех алгоритмов анализа информации (ААИ, АВИ и АТИ). Данный алгоритм выполняет нейросетевое объединение признаков полученных с помощью алгоритмов ААИ, АВИ и АТИ.

OCEAN-AI предоставляет примеры решения прикладных задач на основе полученных гипотез предсказаний оценок ПКЛЧ:

1. **Ранжирование потенциальных кандидатов для выполнения профессиональных обязанностей:**
    1. по группам профессий;
    2. по профессиональным навыкам.
2. **Прогнозирование потребительских предпочтений по выбору промышленных потребительских товаров:**
    1. на примере характеристик автомобиля;
    2. на примере категорий применения мобильного устройства.
3. **Формирование эффективных рабочих коллективов:**
    1. поиск подходящего младшего коллеги;

2. поиск подходящего старшего коллеги.

Помимо основной задачи - мультимодального оценивания персональных качеств личности человека, реализованные в OCEAN-AI признаки позволят исследователям решать другие задачи анализа поведения человека, например распознавание его аффективных состояний.

OCEAN-AI использует самые актуальные библиотеки с открытым исходным кодом для обработки аудио-, видео- и текстовой информации: librosa, openSMILE, openCV, mediapipe, transformers.

OCEAN-AI написана на языке программирования python. Нейросетевые модели реализованы и обучены с использованием библиотеки с открытым исходным кодом TensorFlow.

ГЛАВА 1

Исследовательские данные

Библиотека OCEAN-AI была апробирована на двух корпусах:

1. Общедоступном и крупномаштабном корпусе First Impressions V2.
2. На первом общедоступном рускоязычном мультимодальном корпусе для оценки персональных качеств - Multimodal Personality Traits Assessment (MuPTA) Corpus.

ГЛАВА 2

Свидетельство о государственной регистрации программы для ЭВМ

Библиотека алгоритмов интеллектуального анализа поведения человека на основе его мультимодальных данных, обеспечивающих оценивание уровня отдельных персональных качеств личности человека для выполнения профессиональных обязанностей (OCEAN-AI)

ГЛАВА 3

Свидетельство о государственной регистрации базы данных

Корпус для мультимодального оценивания персональных качеств личности человека (MuPTA - Multimodal Personality Traits Assessment Corpus)

ГЛАВА 4

Публикации

4.1 Журналы

```
@article{ryumina22_neurocomputing,
    author = {Elena Ryumina and Denis Dresvyanskiy and Alexey Karpov},
    title = {In Search of a Robust Facial Expressions Recognition Model: A Large-Scale␣
↪Visual Cross-Corpus Study},
    journal = {Neurocomputing},
    volume = {514},
    pages = {435-450},
    year = {2022},
    doi = {https://doi.org/10.1016/j.neucom.2022.10.013},
}
```

```
@article{ryumina24_eswa,
    author = {Elena Ryumina and Maxim Markitantov and Dmitry Ryumin and Alexey Karpov},
    title = {OCEAN-AI Framework with EmoFormer Cross-Hemiface Attention Approach for␣
↪Personality Traits Assessment},
    journal = {Expert Systems with Applications},
    volume = {239},
    pages = {122441},
    year = {2024},
    doi = {https://doi.org/10.1016/j.eswa.2023.122441},
}
```

4.2 Конференции

```
@inproceedings{ryumina23_interspeech,
    author = {Elena Ryumina and Dmitry Ryumin and Maxim Markitantov and Heysem Kaya and
↪Alexey Karpov},
    title = {Multimodal Personality Traits Assessment (MuPTA) Corpus: The Impact of
↪Spontaneous and Read Speech},
    year = {2023},
    booktitle = {INTERSPEECH},
    pages = {4049--4053},
    doi = {https://doi.org/10.21437/Interspeech.2023-1686},
}
```

Разработка поддерживается исследовательским центром «Сильный искусственный интеллект в промышленности» Университета ИТМО.

4.2.1 Быстрый старт

Установка и обновление

Установка с помощью PyPi

```
pip install oceanai
```

Обновление с помощью PyPi

```
pip install --upgrade oceanai
```

Зависимости

Таблица 1: Устанавливаются автоматически

Библиотека	Рекомендуемая версия	Текущая версия
ipython	8.7.0	
jupyterlab	3.5.0	
tensorflow	2.11.0	
keras	2.11.0	
Keras-Applications	1.0.8	
numpy	1.23.5	
scipy	1.9.3	
pandas	1.5.2	
requests	2.28.1	
opensmile	2.4.1	
librosa	0.9.2	
audioread	3.0.0	
scikit-learn	1.1.3	
opencv-contrib-python	4.6.0.66	
pymediainfo	6.0.1	
mediapipe	0.9.0	
liwc	0.5.0	
transformers	4.36.0	
sentencepiece	0.1.99	
torch	2.0.1	
torchaudio	2.0.2	
sacremoses	0.0.1	

Примеры

Решение практических задач

Решение практической задачи 1

Задача: ранжирование потенциальных кандидатов по профессиональным обязанностям

Решение практической задачи выполняется в два этапа. На первом этапе необходимо использовать библиотеку OCEAN-AI для получения гипотез предсказаний (оценок персональных качеств личности человека). На втором этапе следует использовать методы _candidate_ranking и _priority_skill_calculation из библиотеки OCEAN-AI для решения представленной практической задачи. Примеры результатов работы и реализации представлены ниже.

Таким образом, библиотека OCEAN-AI предоставляет инструменты для анализа персональных качеств личности кандидатов и их соответствия требованиям должности, что может значительно улучшить процесс подбора персонала и помочь в принятии более объективных и систематизированных решений при ранжировании кандидатов.

FI V2

[2]:
```
# Импорт необходимых инструментов
import os
import pandas as pd

# Импорт модуля
from oceanai.modules.lab.build import Run

# Создание экземпляра класса
_b5 = Run()

# Настройка ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000      # Размер загрузки файла из сети за 1 шаг

corpus = 'fi'

# Формирование аудиомоделей
res_load_model_hc = _b5.load_audio_model_hc()
res_load_model_nn = _b5.load_audio_model_nn()

# Загрузка весов аудиомоделей
url = _b5.weights_for_big5_['audio'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_audio_model_weights_hc(url = url)

url = _b5.weights_for_big5_['audio'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_audio_model_weights_nn(url = url)

# Формирование видеомоделей
res_load_model_hc = _b5.load_video_model_hc(lang='en')
res_load_model_deep_fe = _b5.load_video_model_deep_fe()
res_load_model_nn = _b5.load_video_model_nn()

# Загрузка весов видеомоделей
url = _b5.weights_for_big5_['video'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_video_model_weights_hc(url = url)

url = _b5.weights_for_big5_['video'][corpus]['fe']['sberdisk']
res_load_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(url = url)

url = _b5.weights_for_big5_['video'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_video_model_weights_nn(url = url)

# Загрузка словаря с экспертными признаками (текстовая модальность)
res_load_text_features = _b5.load_text_features()

# Формирование текстовых моделей
res_setup_translation_model = _b5.setup_translation_model() # только для русского языка
res_setup_translation_model = _b5.setup_bert_encoder()
res_load_text_model_hc_fi = _b5.load_text_model_hc(corpus=corpus)
res_load_text_model_nn_fi = _b5.load_text_model_nn(corpus=corpus)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Загрузка весов текстовых моделей
url = _b5.weights_for_big5_['text'][corpus]['hc']['sberdisk']
res_load_text_model_weights_hc_fi = _b5.load_text_model_weights_hc(url = url)

url = _b5.weights_for_big5_['text'][corpus]['nn']['sberdisk']
res_load_text_model_weights_nn_fi = _b5.load_text_model_weights_nn(url = url)

# Формирование модели для мультимодального объединения информации
res_load_avt_model_b5 = _b5.load_avt_model_b5()

# Загрузка весов модели для мультимодального объединения информации
url = _b5.weights_for_big5_['avt'][corpus]['b5']['sberdisk']
res_load_avt_model_weights_b5 = _b5.load_avt_model_weights_b5(url = url)

PATH_TO_DIR = './video_FI/'
PATH_SAVE_VIDEO = './video_FI/test/'

_b5.path_to_save_ = PATH_SAVE_VIDEO

# Загрузка 10 тестовых аудиовидеозаписей из корпуса First Impression V2
# URL: https://chalearnlap.cvc.uab.cat/dataset/24/description/
domain = 'https://download.sberdisk.ru/download/file/'
tets_name_files = [
    '429713680?token=FqHdMLSSh7zYSZt&filename=_plk5k7PBEg.003.mp4',
    '429713681?token=Hz9b4lQkrLfic33&filename=be0DQawtVkE.002.mp4',
    '429713683?token=EgUXS9Xs8xHm5gz&filename=2d6btbaNdfo.000.mp4',
    '429713684?token=1U26753kmPYdIgt&filename=300gK3CnzW0.003.mp4',
    '429713685?token=LyigAWLTzDNwKJO&filename=300gK3CnzW0.001.mp4',
    '429713686?token=EpfRbCKHyuc4HPu&filename=cLaZxEf1nE4.004.mp4',
    '429713687?token=FNTkwqBr4jOS95l&filename=g24JGYuT74A.004.mp4',
    '429713688?token=qDT95nz7hfm2Nki&filename=JZNMxa3OKHY.000.mp4',
    '429713689?token=noLguEGXDpbcKhg&filename=nvlqJbHk_Lc.003.mp4',
    '429713679?token=9L7RQ0hgdJlcek6&filename=4vdJGgZpj4k.003.mp4'
]

for curr_files in tets_name_files:
    _b5.download_file_from_url(url = domain + curr_files, out = True)

# Получение прогнозов
_b5.path_to_dataset_ = PATH_TO_DIR # Директория набора данных
_b5.ext_ = ['.mp4'] # Расширения искомых файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_[corpus]['sberdisk']

_b5.get_avt_predictions(url_accuracy = url_accuracy, lang = 'en')
```

[2023-12-16 18:42:02] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-16 18:42:05] Получение прогнозов и вычисление точности (мультимодальное объединение) ...

(continues on next page)

(продолжение с предыдущей страницы)

10 из 10 (100.0%) ... GitHub:nbsphinx-math:*OCEANAI*_guide:nbsphinx-math:*notebooks*_FI:nbsphinx-math:*test*_plk5k7PBEg.003.mp4
...

Person ID	Path	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism
1	2d6btbaNdfo.000.mp4	0.581159	0.628822	0.466609	0.622129	0.553832
2	300gK3CnzW0.001.mp4	0.463991	0.418851	0.41301	0.493329	0.423093
3	300gK3CnzW0.003.mp4	0.454281	0.415049	0.39189	0.485114	0.420741
4	4vdJGgZpj4k.003.mp4	0.588461	0.643233	0.530789	0.603038	0.593398
5	be0DQawtVkE.002.mp4	0.633433	0.533295	0.523742	0.608591	0.588456
6	cLaZxEf1nE4.004.mp4	0.636944	0.542386	0.558461	0.570975	0.558983
7	g24JGYuT74A.004.mp4	0.531518	0.376987	0.393309	0.4904	0.447881
8	JZNMxa3OKHY.000.mp4	0.610342	0.541418	0.563163	0.595013	0.569461
9	nvlqJbHk_Lc.003.mp4	0.495809	0.458526	0.414436	0.469152	0.435461
10	_plk5k7PBEg.003.mp4	0.60707	0.591893	0.520662	0.603938	0.565726

[2023-12-16 18:42:05] Точность по отдельным персональным качествам личности человека ...

Metrics	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism	Mean
MAE	0.0589	0.0612	0.0864	0.0697	0.0582	0.0669
Accuracy	0.9411	0.9388	0.9136	0.9303	0.9418	0.9331

[2023-12-16 18:42:05] Средняя средних абсолютных ошибок: 0.0669, средняя точность: 0.9331 ...

Лог файлы успешно сохранены ...

— Время выполнения: 64.481 сек. —

[2]: True

Для выполнения ранжирования кандидатов необходимо знать весовые коэффициенты, определяющие приоритетность персональных качеств личности в зависимости от профессии.

Предлагаются весовые коэффициенты для 5 профессий, вычисленные на основе научных статей:

1) Sajjad H. et al. Personality and Career Choices // African Journal of Business Management. - 2012. –

Vol. 6 (6) – pp. 2255-2260.

2) Alkhelil A. H. The Relationship between Personality Traits and Career Choice: A Case Study of Secondary School Students // International Journal of Academic Research in Progressive Education and Development. – 2016. – Vol. 5(2). – pp. 2226-6348.

3) De Jong N. et al. Personality Traits and Career Role Enactment: Career Role Preferences as a Mediator // Frontiers in Psychology. – 2019. – Vol. 10. – pp. 1720.

Пользователь может установить свои весовые коэффициенты; сумма весов должна быть равна 100.

[3]:
```
# Загрузка датафрейма с весовыми коэффициентами
url = 'https://download.sberdisk.ru/download/file/478675798?token=fF5fNZVpthQlEV0&
↪filename=traits_priority_for_professions.csv'
traits_priority_for_professions = pd.read_csv(url)

traits_priority_for_professions.index.name = 'ID'
traits_priority_for_professions.index += 1
traits_priority_for_professions.index = traits_priority_for_professions.index.map(str)

traits_priority_for_professions
```

[3]:

ID	Profession	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism
1	Managers/executives	15	35	15	30	5
2	Entrepreneurship	30	30	5	5	30
3	Social/Non profit making professions	5	5	35	35	20
4	Public sector professions	15	50	15	15	5
5	Scientists/researchers, and engineers	50	15	5	15	15

Ранжирование кандидатов на должность инженера-проектировщика

[4]:
```
weights = traits_priority_for_professions.iloc[4].values[1:]
weights = list(map(int, weights))

_b5._candidate_ranking(
    weigths_openness = weights[0],
    weigths_conscientiousness = weights[1],
    weigths_extraversion = weights[2],
    weigths_agreeableness = weights[3],
    weigths_non_neuroticism = weights[4],
    out = False
)

_b5._save_logs(df = _b5.df_files_ranking_, name = 'engineer_candidate_ranking_fi_en',␣
↪out = True)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Опционно
df = _b5.df_files_ranking_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON',
↪ 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[4]:

	Path	OPE	CON	EXT	AGR	NNEU	\
Person ID							
5	be0DQawtVkE.002.mp4	0.633	0.533	0.524	0.609	0.588	
6	cLaZxEf1nE4.004.mp4	0.637	0.542	0.558	0.571	0.559	
4	4vdJGgZpj4k.003.mp4	0.588	0.643	0.531	0.603	0.593	
10	_plk5k7PBEg.003.mp4	0.607	0.592	0.521	0.604	0.566	
8	JZNMxa3OKHY.000.mp4	0.610	0.541	0.563	0.595	0.569	
1	2d6btbaNdfo.000.mp4	0.581	0.629	0.467	0.622	0.554	
7	g24JGYuT74A.004.mp4	0.532	0.377	0.393	0.490	0.448	
9	nvlqJbHk_Lc.003.mp4	0.496	0.459	0.414	0.469	0.435	
2	300gK3CnzW0.001.mp4	0.464	0.419	0.413	0.493	0.423	
3	300gK3CnzW0.003.mp4	0.454	0.415	0.392	0.485	0.421	

	Candidate score
Person ID	
5	60.246
6	59.725
4	59.672
10	59.380
8	58.921
1	58.463
7	48.271
9	47.310
2	45.294
3	44.487

Ранжирование кандидатов на должность менеджера

[5]:
```
weights = traits_priority_for_professions.iloc[0].values[1:]
weights = list(map(int, weights))

_b5._candidate_ranking(
    weigths_openness = weights[0],
    weigths_conscientiousness = weights[1],
    weigths_extraversion = weights[2],
    weigths_agreeableness = weights[3],
    weigths_non_neuroticism = weights[4],
    out = False
)

_b5._save_logs(df = _b5.df_files_ranking_, name = 'executive_candidate_ranking_fi_en',␣
↪out = True)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Опционно
df = _b5.df_files_ranking_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON',
↪ 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[5]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU	\
4	4vdJGgZpj4k.003.mp4	0.588	0.643	0.531	0.603	0.593	
1	2d6btbaNdfo.000.mp4	0.581	0.629	0.467	0.622	0.554	
10	_plk5k7PBEg.003.mp4	0.607	0.592	0.521	0.604	0.566	
8	JZNMxa3OKHY.000.mp4	0.610	0.541	0.563	0.595	0.569	
5	be0DQawtVkE.002.mp4	0.633	0.533	0.524	0.609	0.588	
6	cLaZxEf1nE4.004.mp4	0.637	0.542	0.558	0.571	0.559	
9	nvlqJbHk_Lc.003.mp4	0.496	0.459	0.414	0.469	0.435	
2	300gK3CnzW0.001.mp4	0.464	0.419	0.413	0.493	0.423	
7	g24JGYuT74A.004.mp4	0.532	0.377	0.393	0.490	0.448	
3	300gK3CnzW0.003.mp4	0.454	0.415	0.392	0.485	0.421	

Person ID	Candidate score
4	60.360
1	59.158
10	58.579
8	57.250
5	57.223
6	56.839
9	45.954
2	44.730
7	44.018
3	43.876

Для ранжирования кандидатов по профессиональным навыкам необходимо задать по два коэффициента корреляции для каждого персонального качества личности человека и навыка, а также порога полярности качеств. Эти коэффициенты должны показывать, как измениться оценка качества человека если она больше или меньше заданного порога полярности качеств.

В качестве примера предлагается использование коэффициентов корреляции между двумя людьми в четырьмя профессиональными навыками, представленных в статье:

1) Wehner C., de Grip A., Pfeifer H. Do recruiters select workers with different personality traits for different tasks? A discrete choice experiment // Labour Economics. - 2022. - vol. 78. - pp. 102186.

Представлены 4 профессиональных навыка:

1) Analytical (Аналитические навыки). Умение эффективно решать новые задачи, требующие глубокого анализа.

2) Interactive (Навыки межличностного общения). Умение убеждать и достигать компромиссов с заказчиками и коллегами.

3) Routine (Способность выполнять рутинную работу). Умение эффективно управлять рутинными задачами, соблюдая точность и внимание к деталям.

4) Non-Routine (Способность выполнять нестандартную работу). Умение реагировать и решать проблемы, не имеющие установленного порядка, проявляя адаптивность и креативные навыки в ре-

шении задач.

Пользователь может установить свои коэффициенты корреляции и ранжировать кандидатов по другим профессиональным навыкам.

Ранжирование кандидатов по профессиональным навыкам

[6]:
```
# Загрузка датафрейма с коэффициентами корреляции
url = 'https://download.sberdisk.ru/download/file/478678231?token=0qiZwliLtHWWYMv&
↪filename=professional_skills.csv'
df_professional_skills = pd.read_csv(url)

df_professional_skills.index.name = 'ID'
df_professional_skills.index += 1
df_professional_skills.index = df_professional_skills.index.map(str)

df_professional_skills
```

[6]:

ID	Trait	Score_level	Analytical	Interactive	Routine	Non-Routine
1	Openness	high	0.082	0.348	0.571	0.510
2	Openness	low	0.196	0.152	0.148	0.218
3	Conscientiousness	high	0.994	1.333	1.507	1.258
4	Conscientiousness	low	0.241	0.188	0.191	0.267
5	Extraversion	high	0.169	-0.060	0.258	0.017
6	Extraversion	low	0.181	0.135	0.130	0.194
7	Agreeableness	high	1.239	0.964	1.400	1.191
8	Agreeableness	low	0.226	0.180	0.189	0.259
9	Non-Neuroticism	high	0.636	0.777	0.876	0.729
10	Non-Neuroticism	low	0.207	0.159	0.166	0.238

[7]:
```
_b5._priority_skill_calculation(
    correlation_coefficients = df_professional_skills,
    threshold = 0.5,
    out = True
)

_b5._save_logs(df = _b5.df_files_priority_skill_, name = 'skill_candidate_ranking_fi_en',
↪ out = True)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Опционно
df = _b5.df_files_priority_skill_.rename(columns = {'Openness':'OPE', 'Conscientiousness
↪':'CON', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[7]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU	Analytical	Interactive	Routine	Non-Routine
4	4vdJGgZpj4k.003.mp4	0.588	0.643	0.531	0.603	0.593	0.380	0.415	0.561	0.454
1	2d6btbaNdfo.000.mp4	0.581	0.629	0.467	0.622	0.554	0.376	0.427	0.539	0.465
10	_plk5k7PBEg.003.mp4	0.607	0.592	0.521	0.604	0.566	0.367	0.398	0.543	0.439
5	be0DQawtVkE.002.mp4	0.633	0.533	0.524	0.609	0.588	0.360	0.389	0.534	0.431
8	JZNMxa3OKHY.000.mp4	0.610	0.541	0.563	0.595	0.569	0.357	0.383	0.528	0.425
6	cLaZxEf1nE4.004.mp4	0.637	0.542	0.558	0.571	0.559	0.350	0.379	0.523	0.421
9	nvlqJbHk_Lc.003.mp4	0.496	0.459	0.414	0.469	0.435	0.096	0.074	0.075	0.107
2	300gK3CnzW0.001.mp4	0.464	0.419	0.413	0.493	0.423	0.093	0.072	0.073	0.104
3	300gK3CnzW0.003.mp4	0.454	0.415	0.392	0.485	0.421	0.091	0.071	0.072	0.102
7	g24JGYuT74A.004.mp4	0.532	0.377	0.393	0.490	0.448	0.082	0.094	0.119	0.136

MuPTA **(ru)**

[9]:

```
import os
import pandas as pd

# Импорт модуля
from oceanai.modules.lab.build import Run

# Создание экземпляра класса
_b5 = Run()

corpus = 'mupta'
lang = 'ru'

# Настройка ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000      # Размер загрузки файла из сети за 1 шаг
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Формирование аудиомоделей
res_load_model_hc = _b5.load_audio_model_hc()
res_load_model_nn = _b5.load_audio_model_nn()

# Загрузка весов аудиомоделей
url = _b5.weights_for_big5_['audio'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_audio_model_weights_hc(url = url)

url = _b5.weights_for_big5_['audio'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_audio_model_weights_nn(url = url)

# Формирование видеомоделей
res_load_model_hc = _b5.load_video_model_hc(lang=lang)
res_load_model_deep_fe = _b5.load_video_model_deep_fe()
res_load_model_nn = _b5.load_video_model_nn()

# Загрузка весов видеомоделей
url = _b5.weights_for_big5_['video'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_video_model_weights_hc(url = url)

url = _b5.weights_for_big5_['video'][corpus]['fe']['sberdisk']
res_load_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(url = url)

url = _b5.weights_for_big5_['video'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_video_model_weights_nn(url = url)

# Загрузка словаря с экспертными признаками (текстовая модальность)
res_load_text_features = _b5.load_text_features()

# Формирование текстовых моделей
res_setup_translation_model = _b5.setup_translation_model() # только для русского языка
res_setup_translation_model = _b5.setup_bert_encoder()
res_load_text_model_hc_fi = _b5.load_text_model_hc(corpus=corpus)
res_load_text_model_nn_fi = _b5.load_text_model_nn(corpus=corpus)

# Загрузка весов текстовых моделей
url = _b5.weights_for_big5_['text'][corpus]['hc']['sberdisk']
res_load_text_model_weights_hc_fi = _b5.load_text_model_weights_hc(url = url)

url = _b5.weights_for_big5_['text'][corpus]['nn']['sberdisk']
res_load_text_model_weights_nn_fi = _b5.load_text_model_weights_nn(url = url)

# Формирование модели для мультимодального объединения информации
res_load_avt_model_b5 = _b5.load_avt_model_b5()

# Загрузка весов модели для мультимодального объединения информации
url = _b5.weights_for_big5_['avt'][corpus]['b5']['sberdisk']
res_load_avt_model_weights_b5 = _b5.load_avt_model_weights_b5(url = url)

PATH_TO_DIR = './video_MuPTA/'
PATH_SAVE_VIDEO = './video_MuPTA/test/'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
_b5.path_to_save_ = PATH_SAVE_VIDEO

# Загрузка 10 тестовых аудиовидеозаписей из корпуса MuPTA
# URL: https://hci.nw.ru/en/pages/mupta-corpus
domain = 'https://download.sberdisk.ru/download/file/'
tets_name_files = [
    '477995979?token=2cvyk7CS0mHx2MJ&filename=speaker_06_center_83.mov',
    '477995980?token=jGPtBPS69uzFU6Y&filename=speaker_01_center_83.mov',
    '477995967?token=zCaRbNB6ht5wMPq&filename=speaker_11_center_83.mov',
    '477995966?token=B1rbinDYRQKrI3T&filename=speaker_15_center_83.mov',
    '477995978?token=dEpVDtZg1EQiEQ9&filename=speaker_07_center_83.mov',
    '477995961?token=o1hVjw8G45q9L9Z&filename=speaker_19_center_83.mov',
    '477995964?token=5K22OAqf673VHPq&filename=speaker_23_center_83.mov',
    '477995965?token=v1LVD2KT1cU7Lpb&filename=speaker_24_center_83.mov',
    '477995962?token=tmaSGyyWLA6XCy9&filename=speaker_27_center_83.mov',
    '477995963?token=bTpo96qNDPcwGqb&filename=speaker_10_center_83.mov',
]

for curr_files in tets_name_files:
    _b5.download_file_from_url(url = domain + curr_files, out = True)

# Получение прогнозов
_b5.path_to_dataset_ = PATH_TO_DIR # Директория набора данных
_b5.ext_ = ['.mov'] # Расширения искомых файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_['mupta']['sberdisk']

_b5.get_avt_predictions(url_accuracy = url_accuracy, lang = lang)
```

[2023-12-16 18:51:57] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-16 18:52:01] Получение прогнозов и вычисление точности (мультимодальное объединение) ...

10 из 10 (100.0%) ... GitHub:nbsphinx-math:*OCEANAI*_guide:nbsphinx-math:*notebooks*_MuPTA:nbsphinx-math:*test*_27_center_83.mov
...

Person ID	Path	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism
1	speaker_01_center_83.mov	0.758137	0.693356	0.650108		
2	speaker_06_center_83.mov	0.681602	0.654339	0.607156		
3	speaker_07_center_83.mov	0.666104	0.656836	0.567863		
4	speaker_10_center_83.mov	0.694171	0.596195	0.571414		
5	speaker_11_center_83.mov	0.712885	0.594764	0.571709		
6	speaker_15_center_83.mov	0.664158	0.670411	0.60421		
7	speaker_19_center_83.mov	0.761213	0.652635	0.651028		
8	speaker_23_center_83.mov	0.692788	0.68324	0.616737		
9	speaker_24_center_83.mov	0.705923	0.658382	0.610645		
10	speaker_27_center_83.mov	0.753417	0.708372	0.654608		

(continues on next page)

(продолжение с предыдущей страницы)

```
1              0.744589        0.488671
2              0.731282        0.417908
3              0.685067        0.378102
4               0.66223        0.348639
5              0.716696         0.37802
6              0.696056        0.399842
7              0.788677        0.459676
8              0.795205        0.447242
9              0.697415        0.411988
10             0.816416        0.504743
```

[2023-12-16 18:52:01] Точность по отдельным персональным качествам личности человека ...

```
          Openness  Conscientiousness  Extraversion  Agreeableness  \
Metrics
MAE         0.0673             0.0789        0.1325          0.102
Accuracy    0.9327             0.9211        0.8675          0.898

          Non-Neuroticism    Mean
Metrics
MAE                0.1002  0.0962
Accuracy           0.8998  0.9038
```

[2023-12-16 18:52:01] Средняя средних абсолютных ошибок: 0.0962, средняя точность: 0.9038 ...

Лог файлы успешно сохранены ...

— Время выполнения: 415.41 сек. —

[9]:
```
True
```

Для выполнения ранжирования кандидатов необходимо знать весовые коэффициенты, определяющие приоритетность персональных качеств личности в зависимости от профессии.

Предлагаются весовые коэффициенты для 5 профессий, вычисленные на основе научных статей:

1) Sajjad H. et al. Personality and Career Choices // African Journal of Business Management. - 2012. – Vol. 6 (6) – pp. 2255-2260.
2) Alkhelil A. H. The Relationship between Personality Traits and Career Choice: A Case Study of Secondary School Students // International Journal of Academic Research in Progressive Education and Development. – 2016. – Vol. 5(2). – pp. 2226-6348.
3) De Jong N. et al. Personality Traits and Career Role Enactment: Career Role Preferences as a Mediator // Frontiers in Psychology. – 2019. – Vol. 10. – pp. 1720.

Пользователь может установить свои весовые коэффициенты; сумма весов должна быть равна 100.

[10]:
```
# Загрузка датафрейма с весовыми коэффициентами
url = 'https://download.sberdisk.ru/download/file/478675798?token=fF5fNZVpthQlEV0&
↪filename=traits_priority_for_professions.csv'
traits_priority_for_professions = pd.read_csv(url)

traits_priority_for_professions.index.name = 'ID'
traits_priority_for_professions.index += 1
traits_priority_for_professions.index = traits_priority_for_professions.index.map(str)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
traits_priority_for_professions
```

[10]:

ID	Profession	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism
1	Managers/executives	15	35	15	30	5
2	Entrepreneurship	30	30	5	5	30
3	Social/Non profit making professions	5	5	35	35	20
4	Public sector professions	15	50	15	15	5
5	Scientists/researchers, and engineers	50	15	5	15	15

Ранжирование кандидатов на должность инженера-проектировщика

[11]:
```
weights = traits_priority_for_professions.iloc[4].values[1:]
weights = list(map(int, weights))

_b5._candidate_ranking(
    weigths_openness = weights[0],
    weigths_conscientiousness = weights[1],
    weigths_extraversion = weights[2],
    weigths_agreeableness = weights[3],
    weigths_non_neuroticism = weights[4],
    out = False
)

_b5._save_logs(df = _b5.df_files_ranking_, name = 'engineer_candidate_ranking_mupta_ru',␣
↪out = True)

# Опционно
df = _b5.df_files_ranking_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON',
↪ 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[11]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU
10	speaker_27_center_83.mov	0.753	0.708	0.655	0.816	0.505
1	speaker_01_center_83.mov	0.758	0.693	0.650	0.745	0.489
7	speaker_19_center_83.mov	0.761	0.653	0.651	0.789	0.460
8	speaker_23_center_83.mov	0.693	0.683	0.617	0.795	0.447
9	speaker_24_center_83.mov	0.706	0.658	0.611	0.697	0.412
2	speaker_06_center_83.mov	0.682	0.654	0.607	0.731	0.418

(continues on next page)

(продолжение с предыдущей страницы)

```
5          speaker_11_center_83.mov  0.713  0.595  0.572  0.717  0.378
6          speaker_15_center_83.mov  0.664  0.670  0.604  0.696  0.400
3          speaker_07_center_83.mov  0.666  0.657  0.568  0.685  0.378
4          speaker_10_center_83.mov  0.694  0.596  0.571  0.662  0.349

           Candidate score
Person ID
10                 71.387
1                  70.057
7                  69.831
8                  66.608
9                  64.866
2                  64.169
5                  63.845
6                  62.724
3                  61.945
4                  61.672
```

Ранжирование кандидатов на должность менеджера

[12]:
```
weights = traits_priority_for_professions.iloc[0].values[1:]
weights = list(map(int, weights))

_b5._candidate_ranking(
    weigths_openness = weights[0],
    weigths_conscientiousness = weights[1],
    weigths_extraversion = weights[2],
    weigths_agreeableness = weights[3],
    weigths_non_neuroticism = weights[4],
    out = False
)

_b5._save_logs(df = _b5.df_files_ranking_, name = 'executive_candidate_ranking_mupta_ru',
↪ out = True)

# Опционно
df = _b5.df_files_ranking_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON',
↪ 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[12]:
```
                             Path    OPE    CON    EXT    AGR   NNEU  \
Person ID
10         speaker_27_center_83.mov  0.753  0.708  0.655  0.816  0.505
1          speaker_01_center_83.mov  0.758  0.693  0.650  0.745  0.489
7          speaker_19_center_83.mov  0.761  0.653  0.651  0.789  0.460
8          speaker_23_center_83.mov  0.693  0.683  0.617  0.795  0.447
2          speaker_06_center_83.mov  0.682  0.654  0.607  0.731  0.418
9          speaker_24_center_83.mov  0.706  0.658  0.611  0.697  0.412
6          speaker_15_center_83.mov  0.664  0.670  0.604  0.696  0.400
```

(continues on next page)

(продолжение с предыдущей страницы)

```
3          speaker_07_center_83.mov  0.666  0.657  0.568  0.685  0.378
5          speaker_11_center_83.mov  0.713  0.595  0.572  0.717  0.378
4          speaker_10_center_83.mov  0.694  0.596  0.571  0.662  0.349

           Candidate score
Person ID
10                 72.930
1                  70.172
7                  69.985
8                  69.649
2                  66.261
9                  65.774
6                  65.371
3                  63.941
5                  63.477
4                  61.461
```

Для ранжирования кандидатов по профессиональным навыкам необходимо задать по два коэффициента корреляции для каждого персонального качества личности человека и навыка, а также порога полярности качеств. Эти коэффициенты должны показывать, как измениться оценка качества человека если она больше или меньше заданного порога полярности качеств.

В качестве примера предлагается использование коэффициентов корреляции между двумя людьми в четырьмя профессиональными навыками, представленных в статье:

1) Wehner C., de Grip A., Pfeifer H. Do recruiters select workers with different personality traits for different tasks? A discrete choice experiment // Labour Economics. - 2022. - vol. 78. - pp. 102186.

Представлены 4 профессиональных навыка:

1) Analytical (Аналитические навыки). Умение эффективно решать новые задачи, требующие глубокого анализа.

2) Interactive (Навыки межличностного общения). Умение убеждать и достигать компромиссов с заказчиками и коллегами.

3) Routine (Способность выполнять рутинную работу). Умение эффективно управлять рутинными задачами, соблюдая точность и внимание к деталям.

4) Non-Routine (Способность выполнять нестандартную работу). Умение реагировать и решать проблемы, не имеющие установленного порядка, проявляя адаптивность и креативные навыки в решении задач.

Пользователь может установить свои коэффициенты корреляции и ранжировать кандидатов по другим профессиональным навыкам.

Ранжирование кандидатов по профессиональным навыкам

[13]:
```
# Загрузка датафрейма с коэффициентами корреляции
url = 'https://download.sberdisk.ru/download/file/478678231?token=0qiZwliLtHWWYMv&
↪filename=professional_skills.csv'
df_professional_skills = pd.read_csv(url)

df_professional_skills.index.name = 'ID'
df_professional_skills.index += 1
df_professional_skills.index = df_professional_skills.index.map(str)

df_professional_skills
```

[13]:

ID	Trait	Score_level	Analytical	Interactive	Routine	Non-Routine
1	Openness	high	0.082	0.348	0.571	0.510
2	Openness	low	0.196	0.152	0.148	0.218
3	Conscientiousness	high	0.994	1.333	1.507	1.258
4	Conscientiousness	low	0.241	0.188	0.191	0.267
5	Extraversion	high	0.169	-0.060	0.258	0.017
6	Extraversion	low	0.181	0.135	0.130	0.194
7	Agreeableness	high	1.239	0.964	1.400	1.191
8	Agreeableness	low	0.226	0.180	0.189	0.259
9	Non-Neuroticism	high	0.636	0.777	0.876	0.729
10	Non-Neuroticism	low	0.207	0.159	0.166	0.238

[14]:
```
_b5._priority_skill_calculation(
    correlation_coefficients = df_professional_skills,
    threshold = 0.5,
    out = True
)

_b5._save_logs(df = _b5.df_files_priority_skill_, name = 'skill_candidate_ranking_mupta_
↪ru', out = True)

# Опционно
df = _b5.df_files_priority_skill_.rename(columns = {'Openness':'OPE', 'Conscientiousness
↪':'CON', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
```

(continues on next page)

(продолжение с предыдущей страницы)

```
df
```

[14]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU	\
10	speaker_27_center_83.mov	0.753	0.708	0.655	0.816	0.505	
8	speaker_23_center_83.mov	0.693	0.683	0.617	0.795	0.447	
7	speaker_19_center_83.mov	0.761	0.653	0.651	0.789	0.460	
1	speaker_01_center_83.mov	0.758	0.693	0.650	0.745	0.489	
2	speaker_06_center_83.mov	0.682	0.654	0.607	0.731	0.418	
6	speaker_15_center_83.mov	0.664	0.670	0.604	0.696	0.400	
9	speaker_24_center_83.mov	0.706	0.658	0.611	0.697	0.412	
3	speaker_07_center_83.mov	0.666	0.657	0.568	0.685	0.378	
5	speaker_11_center_83.mov	0.713	0.595	0.572	0.717	0.378	
4	speaker_10_center_83.mov	0.694	0.596	0.571	0.662	0.349	

Person ID	Analytical	Interactive	Routine	Non-Routine
10	0.442	0.469	0.650	0.525
8	0.384	0.391	0.554	0.455
7	0.379	0.386	0.553	0.454
1	0.377	0.389	0.554	0.455
2	0.360	0.369	0.525	0.430
6	0.354	0.365	0.517	0.423
9	0.353	0.365	0.520	0.425
3	0.346	0.359	0.508	0.416
5	0.343	0.352	0.503	0.413
4	0.328	0.339	0.485	0.397

MuPTA **(en)**

[15]:
```
import os
import pandas as pd

# Импорт модуля
from oceanai.modules.lab.build import Run

# Создание экземпляра класса
_b5 = Run()

corpus = 'fi'
lang = 'en'

# Настройка ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000      # Размер загрузки файла из сети за 1 шаг

# Формирование аудиомоделей
res_load_model_hc = _b5.load_audio_model_hc()
res_load_model_nn = _b5.load_audio_model_nn()

# Загрузка весов аудиомоделей
```

(continues on next page)

(продолжение с предыдущей страницы)

```
url = _b5.weights_for_big5_['audio'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_audio_model_weights_hc(url = url)

url = _b5.weights_for_big5_['audio'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_audio_model_weights_nn(url = url)

# Формирование видеомоделей
res_load_model_hc = _b5.load_video_model_hc(lang=lang)
res_load_model_deep_fe = _b5.load_video_model_deep_fe()
res_load_model_nn = _b5.load_video_model_nn()

# Загрузка весов видеомоделей
url = _b5.weights_for_big5_['video'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_video_model_weights_hc(url = url)

url = _b5.weights_for_big5_['video'][corpus]['fe']['sberdisk']
res_load_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(url = url)

url = _b5.weights_for_big5_['video'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_video_model_weights_nn(url = url)

# Загрузка словаря с экспертными признаками (текстовая модальность)
res_load_text_features = _b5.load_text_features()

# Формирование текстовых моделей
res_setup_translation_model = _b5.setup_translation_model() # только для русского языка
res_setup_translation_model = _b5.setup_bert_encoder()
res_load_text_model_hc_fi = _b5.load_text_model_hc(corpus=corpus)
res_load_text_model_nn_fi = _b5.load_text_model_nn(corpus=corpus)

# Загрузка весов текстовых моделей
url = _b5.weights_for_big5_['text'][corpus]['hc']['sberdisk']
res_load_text_model_weights_hc_fi = _b5.load_text_model_weights_hc(url = url)

url = _b5.weights_for_big5_['text'][corpus]['nn']['sberdisk']
res_load_text_model_weights_nn_fi = _b5.load_text_model_weights_nn(url = url)

# Формирование модели для мультимодального объединения информации
res_load_avt_model_b5 = _b5.load_avt_model_b5()

# Загрузка весов модели для мультимодального объединения информации
url = _b5.weights_for_big5_['avt'][corpus]['b5']['sberdisk']
res_load_avt_model_weights_b5 = _b5.load_avt_model_weights_b5(url = url)

PATH_TO_DIR = './video_MuPTA/'
PATH_SAVE_VIDEO = './video_MuPTA/test/'

_b5.path_to_save_ = PATH_SAVE_VIDEO

# Загрузка 10 тестовых аудиовидеозаписей из корпуса MuPTA
# URL: https://hci.nw.ru/en/pages/mupta-corpus
domain = 'https://download.sberdisk.ru/download/file/'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
tets_name_files = [
    '477995979?token=2cvyk7CS0mHx2MJ&filename=speaker_06_center_83.mov',
    '477995980?token=jGPtBPS69uzFU6Y&filename=speaker_01_center_83.mov',
    '477995967?token=zCaRbNB6ht5wMPq&filename=speaker_11_center_83.mov',
    '477995966?token=B1rbinDYRQKrI3T&filename=speaker_15_center_83.mov',
    '477995978?token=dEpVDtZg1EQiEQ9&filename=speaker_07_center_83.mov',
    '477995961?token=o1hVjw8G45q9L9Z&filename=speaker_19_center_83.mov',
    '477995964?token=5K220Aqf673VHPq&filename=speaker_23_center_83.mov',
    '477995965?token=v1LVD2KT1cU7Lpb&filename=speaker_24_center_83.mov',
    '477995962?token=tmaSGyyWLA6XCy9&filename=speaker_27_center_83.mov',
    '477995963?token=bTpo96qNDPcwGqb&filename=speaker_10_center_83.mov',
]

for curr_files in tets_name_files:
    _b5.download_file_from_url(url = domain + curr_files, out = True)

# Получение прогнозов
_b5.path_to_dataset_ = PATH_TO_DIR # Директория набора данных
_b5.ext_ = ['.mov'] # Расширения искомых файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_['mupta']['sberdisk']

_b5.get_avt_predictions(url_accuracy = url_accuracy, lang = lang)
```

[2023-12-16 19:00:49] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-16 19:00:52] Получение прогнозов и вычисление точности (мультимодальное объединение) ...

10 из 10 (100.0%) ... GitHub:nbsphinx-math:*OCEANAI*_guide:nbsphinx-math:*notebooks*_MuPTA:nbsphinx-math:*test*_27_center_83.mov
...

Person ID	Path	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism
1	speaker_01_center_83.mov	0.564985	0.539052	0.440615	0.59251	0.488763
2	speaker_06_center_83.mov	0.650774	0.663849	0.607308	0.643847	0.620627
3	speaker_07_center_83.mov	0.435976	0.486683	0.313828	0.415446	0.396618
4	speaker_10_center_83.mov	0.498542	0.511243	0.412592	0.468947	0.44399
5	speaker_11_center_83.mov	0.394776	0.341608	0.327082	0.427304	0.354936
6	speaker_15_center_83.mov	0.566107	0.543811	0.492766		
7	speaker_19_center_83.mov	0.506271	0.438215	0.430894		
8	speaker_23_center_83.mov	0.486463	0.521755	0.309894		
9	speaker_24_center_83.mov	0.417404	0.473339	0.320714		
10	speaker_27_center_83.mov	0.526112	0.661107	0.443167		

(continues on next page)

(продолжение с предыдущей страницы)

```
6              0.587411        0.499433
7              0.456177         0.44075
8              0.432291        0.433601
9              0.445086        0.414649
10             0.558965        0.554224
```

[2023-12-16 19:00:52] Точность по отдельным персональным качествам личности человека ...

```
          Openness  Conscientiousness  Extraversion  Agreeableness  \
Metrics
MAE         0.1727             0.1672        0.1661         0.2579
Accuracy    0.8273             0.8328        0.8339         0.7421

          Non-Neuroticism    Mean
Metrics
MAE                 0.107  0.1742
Accuracy            0.893  0.8258
```

[2023-12-16 19:00:52] Средняя средних абсолютных ошибок: 0.1742, средняя точность: 0.8258 ...

Лог файлы успешно сохранены ...

— Время выполнения: 372.823 сек. —

[15]:
```
True
```

Для выполнения ранжирования кандидатов необходимо знать весовые коэффициенты, определяющие приоритетность персональных качеств личности в зависимости от профессии.

Предлагаются весовые коэффициенты для 5 профессий, вычисленные на основе научных статей:

1) Sajjad H. et al. Personality and Career Choices // African Journal of Business Management. - 2012. – Vol. 6 (6) – pp. 2255-2260.
2) Alkhelil A. H. The Relationship between Personality Traits and Career Choice: A Case Study of Secondary School Students // International Journal of Academic Research in Progressive Education and Development. – 2016. – Vol. 5(2). – pp. 2226-6348.
3) De Jong N. et al. Personality Traits and Career Role Enactment: Career Role Preferences as a Mediator // Frontiers in Psychology. – 2019. – Vol. 10. – pp. 1720.

Пользователь может установить свои весовые коэффициенты; сумма весов должна быть равна 100.

[16]:
```
# Загрузка датафрейма с весовыми коэффициентами
url = 'https://download.sberdisk.ru/download/file/478675798?token=fF5fNZVpthQlEV0&
↪filename=traits_priority_for_professions.csv'
traits_priority_for_professions = pd.read_csv(url)

traits_priority_for_professions.index.name = 'ID'
traits_priority_for_professions.index += 1
traits_priority_for_professions.index = traits_priority_for_professions.index.map(str)

traits_priority_for_professions
```

[16]:
```
                            Profession  Openness  Conscientiousness  \
ID
1                   Managers/executives        15                 35
```

(continues on next page)

(продолжение с предыдущей страницы)

```
2                        Entrepreneurship        30                30
3    Social/Non profit making professions         5                 5
4                Public sector professions       15                50
5   Scientists/researchers, and engineers        50                15

    Extraversion  Agreeableness  Non-Neuroticism
ID
1             15             30                5
2              5              5               30
3             35             35               20
4             15             15                5
5              5             15               15
```

Ранжирование кандидатов на должность инженера-проектировщика

[17]:
```
weights = traits_priority_for_professions.iloc[4].values[1:]
weights = list(map(int, weights))

_b5._candidate_ranking(
    weigths_openness = weights[0],
    weigths_conscientiousness = weights[1],
    weigths_extraversion = weights[2],
    weigths_agreeableness = weights[3],
    weigths_non_neuroticism = weights[4],
    out = False
)

_b5._save_logs(df = _b5.df_files_ranking_, name = 'engineer_candidate_ranking_mupta_en',␣
↪out = True)

# Опционно
df = _b5.df_files_ranking_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON',
↪ 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[17]:
```
                              Path    OPE    CON    EXT    AGR   NNEU  \
Person ID
2          speaker_06_center_83.mov  0.651  0.664  0.607  0.644  0.621
6          speaker_15_center_83.mov  0.566  0.544  0.493  0.587  0.499
10         speaker_27_center_83.mov  0.526  0.661  0.443  0.559  0.554
1          speaker_01_center_83.mov  0.565  0.539  0.441  0.593  0.489
4          speaker_10_center_83.mov  0.499  0.511  0.413  0.469  0.444
7          speaker_19_center_83.mov  0.506  0.438  0.431  0.456  0.441
8          speaker_23_center_83.mov  0.486  0.522  0.310  0.432  0.434
3          speaker_07_center_83.mov  0.436  0.487  0.314  0.415  0.397
9          speaker_24_center_83.mov  0.417  0.473  0.321  0.445  0.415
5          speaker_11_center_83.mov  0.395  0.342  0.327  0.427  0.355

           Candidate score
```

(continues on next page)

(продолжение с предыдущей страницы)

```
Person ID
2                  64.500
6                  55.229
10                 55.136
1                  54.757
4                  48.353
7                  47.495
8                  46.687
3                  42.849
9                  42.470
5                  38.232
```

Ранжирование кандидатов на должность менеджера

[18]:
```
weights = traits_priority_for_professions.iloc[0].values[1:]
weights = list(map(int, weights))

_b5._candidate_ranking(
    weigths_openness = weights[0],
    weigths_conscientiousness = weights[1],
    weigths_extraversion = weights[2],
    weigths_agreeableness = weights[3],
    weigths_non_neuroticism = weights[4],
    out = False
)

_b5._save_logs(df = _b5.df_files_ranking_, name = 'executive_candidate_ranking_mupta_en',
↪ out = True)

# Опционно
df = _b5.df_files_ranking_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON',
↪ 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[18]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU	\
2	speaker_06_center_83.mov	0.651	0.664	0.607	0.644	0.621	
10	speaker_27_center_83.mov	0.526	0.661	0.443	0.559	0.554	
6	speaker_15_center_83.mov	0.566	0.544	0.493	0.587	0.499	
1	speaker_01_center_83.mov	0.565	0.539	0.441	0.593	0.489	
4	speaker_10_center_83.mov	0.499	0.511	0.413	0.469	0.444	
8	speaker_23_center_83.mov	0.486	0.522	0.310	0.432	0.434	
7	speaker_19_center_83.mov	0.506	0.438	0.431	0.456	0.441	
9	speaker_24_center_83.mov	0.417	0.473	0.321	0.445	0.415	
3	speaker_07_center_83.mov	0.436	0.487	0.314	0.415	0.397	
5	speaker_11_center_83.mov	0.395	0.342	0.327	0.427	0.355	

Person ID	Candidate score

(continues on next page)

(продолжение с предыдущей страницы)

```
2                  64.524
10                 57.218
6                  55.036
1                  54.170
4                  47.849
8                  45.344
7                  45.284
9                  43.064
3                  42.727
5                  37.378
```

Для ранжирования кандидатов по профессиональным навыкам необходимо задать по два коэффициента корреляции для каждого персонального качества личности человека и навыка, а также порога полярности качеств. Эти коэффициенты должны показывать, как измениться оценка качества человека если она больше или меньше заданного порога полярности качеств.

В качестве примера предлагается использование коэффициентов корреляции между двумя людьми в четырьмя профессиональными навыками, представленных в статье:

1) Wehner C., de Grip A., Pfeifer H. Do recruiters select workers with different personality traits for different tasks? A discrete choice experiment // Labour Economics. - 2022. - vol. 78. - pp. 102186.

Представлены 4 профессиональных навыка:

1) Analytical (Аналитические навыки). Умение эффективно решать новые задачи, требующие глубокого анализа.
2) Interactive (Навыки межличностного общения). Умение убеждать и достигать компромиссов с заказчиками и коллегами.
3) Routine (Способность выполнять рутинную работу). Умение эффективно управлять рутинными задачами, соблюдая точность и внимание к деталям.
4) Non-Routine (Способность выполнять нестандартную работу). Умение реагировать и решать проблемы, не имеющие установленного порядка, проявляя адаптивность и креативные навыки в решении задач.

Пользователь может установить свои коэффициенты корреляции и ранжировать кандидатов по другим профессиональным навыкам.

Ранжирование кандидатов по профессиональным навыкам

[19]:
```
# Загрузка датафрейма с коэффициентами корреляции
url = 'https://download.sberdisk.ru/download/file/478678231?token=0qiZwliLtHWWYMv&
↪filename=professional_skills.csv'
df_professional_skills = pd.read_csv(url)

df_professional_skills.index.name = 'ID'
df_professional_skills.index += 1
df_professional_skills.index = df_professional_skills.index.map(str)

df_professional_skills
```

[19]:
```
                Trait Score_level  Analytical  Interactive  Routine  \
ID
```

(continues on next page)

(продолжение с предыдущей страницы)

```
1            Openness        high       0.082        0.348    0.571
2            Openness         low       0.196        0.152    0.148
3   Conscientiousness        high       0.994        1.333    1.507
4   Conscientiousness         low       0.241        0.188    0.191
5        Extraversion        high       0.169       -0.060    0.258
6        Extraversion         low       0.181        0.135    0.130
7       Agreeableness        high       1.239        0.964    1.400
8       Agreeableness         low       0.226        0.180    0.189
9     Non-Neuroticism        high       0.636        0.777    0.876
10    Non-Neuroticism         low       0.207        0.159    0.166

    Non-Routine
ID
1         0.510
2         0.218
3         1.258
4         0.267
5         0.017
6         0.194
7         1.191
8         0.259
9         0.729
10        0.238
```

[20]:
```
_b5._priority_skill_calculation(
    correlation_coefficients = df_professional_skills,
    threshold = 0.5,
    out = True
)

_b5._save_logs(df = _b5.df_files_priority_skill_, name = 'skill_candidate_ranking_mupta_
↪en', out = True)

# Опционно
df = _b5.df_files_priority_skill_.rename(columns = {'Openness':'OPE', 'Conscientiousness
↪':'CON', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[20]:
```
                              Path    OPE    CON    EXT    AGR   NNEU  \
Person ID
2          speaker_06_center_83.mov  0.651  0.664  0.607  0.644  0.621
10         speaker_27_center_83.mov  0.526  0.661  0.443  0.559  0.554
6          speaker_15_center_83.mov  0.566  0.544  0.493  0.587  0.499
1          speaker_01_center_83.mov  0.565  0.539  0.441  0.593  0.489
4          speaker_10_center_83.mov  0.499  0.511  0.413  0.469  0.444
8          speaker_23_center_83.mov  0.486  0.522  0.310  0.432  0.434
9          speaker_24_center_83.mov  0.417  0.473  0.321  0.445  0.415
3          speaker_07_center_83.mov  0.436  0.487  0.314  0.415  0.397
7          speaker_19_center_83.mov  0.506  0.438  0.431  0.456  0.441
5          speaker_11_center_83.mov  0.395  0.342  0.327  0.427  0.355
```

(continues on next page)

(продолжение с предыдущей страницы)

```
           Analytical  Interactive  Routine  Non-Routine
Person ID
2               0.402        0.436    0.595        0.479
10              0.365        0.419    0.524        0.451
6               0.301        0.327    0.422        0.377
1               0.299        0.325    0.421        0.375
4               0.176        0.194    0.212        0.212
8               0.172        0.192    0.210        0.208
9               0.088        0.068    0.069        0.099
3               0.087        0.068    0.069        0.098
7               0.084        0.094    0.118        0.136
5               0.078        0.060    0.061        0.087
```

Решение практической задачи 2

Задача: прогнозирование потребительских предпочтений на промышленные товары

Решение практической задачи выполняется в два этапа. На первом этапе необходимо использовать библиотеку OCEAN-AI для получения гипотез предсказаний (оценок персональных качеств личности человека). На втором этапе следует использовать метод _priority_calculation из библиотеки OCEAN-AI для решения представленной практической задачи. Примеры результатов работы и реализации представлены ниже.

Таким образом, библиотека OCEAN-AI предоставляет инструмент для анализа персональных качеств личности потребителей, что полезно для предсказания того, что их заинтересует. Это позволит компаниям более точно адаптировать свои товары и услуги к предпочтениям потребителей, делая их более уникальными и персонализированными.

FI V2

[2]:
```
# Импорт необходимых инструментов
import os
import pandas as pd

# Импорт модуля
from oceanai.modules.lab.build import Run

# Создание экземпляра класса
_b5 = Run()

# Настройка ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000      # Размер загрузки файла из сети за 1 шаг

corpus = 'fi'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Формирование аудиомоделей
res_load_model_hc = _b5.load_audio_model_hc()
res_load_model_nn = _b5.load_audio_model_nn()

# Загрузка весов аудиомоделей
url = _b5.weights_for_big5_['audio'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_audio_model_weights_hc(url = url)

url = _b5.weights_for_big5_['audio'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_audio_model_weights_nn(url = url)

# Формирование видеомоделей
res_load_model_hc = _b5.load_video_model_hc(lang='en')
res_load_model_deep_fe = _b5.load_video_model_deep_fe()
res_load_model_nn = _b5.load_video_model_nn()

# Загрузка весов видеомоделей
url = _b5.weights_for_big5_['video'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_video_model_weights_hc(url = url)

url = _b5.weights_for_big5_['video'][corpus]['fe']['sberdisk']
res_load_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(url = url)

url = _b5.weights_for_big5_['video'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_video_model_weights_nn(url = url)

# Загрузка словаря с экспертными признаками (текстовая модальность)
res_load_text_features = _b5.load_text_features()

# Формирование текстовых моделей
res_setup_translation_model = _b5.setup_translation_model() # только для русского языка
res_setup_translation_model = _b5.setup_bert_encoder(force_reload = False)
res_load_text_model_hc_fi = _b5.load_text_model_hc(corpus=corpus)
res_load_text_model_nn_fi = _b5.load_text_model_nn(corpus=corpus)

# Загрузка весов текстовых моделей
url = _b5.weights_for_big5_['text'][corpus]['hc']['sberdisk']
res_load_text_model_weights_hc_fi = _b5.load_text_model_weights_hc(url = url)

url = _b5.weights_for_big5_['text'][corpus]['nn']['sberdisk']
res_load_text_model_weights_nn_fi = _b5.load_text_model_weights_nn(url = url)

# Формирование модели для мультимодального объединения информации
res_load_avt_model_b5 = _b5.load_avt_model_b5()

# Загрузка весов модели для мультимодального объединения информации
url = _b5.weights_for_big5_['avt'][corpus]['b5']['sberdisk']
res_load_avt_model_weights_b5 = _b5.load_avt_model_weights_b5(url = url)

PATH_TO_DIR = './video_FI/'
PATH_SAVE_VIDEO = './video_FI/test/'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
_b5.path_to_save_ = PATH_SAVE_VIDEO

# Загрузка 10 тестовых аудиовидеозаписей из корпуса First Impression V2
# URL: https://chalearnlap.cvc.uab.cat/dataset/24/description/
domain = 'https://download.sberdisk.ru/download/file/'
tets_name_files = [
    '429713680?token=FqHdMLSSh7zYSZt&filename=_plk5k7PBEg.003.mp4',
    '429713681?token=Hz9b4lQkrLfic33&filename=be0DQawtVkE.002.mp4',
    '429713683?token=EgUXS9Xs8xHm5gz&filename=2d6btbaNdfo.000.mp4',
    '429713684?token=1U26753kmPYdIgt&filename=300gK3CnzW0.003.mp4',
    '429713685?token=LyigAWLTzDNwKJO&filename=300gK3CnzW0.001.mp4',
    '429713686?token=EpfRbCKHyuc4HPu&filename=cLaZxEf1nE4.004.mp4',
    '429713687?token=FNTkwqBr4jOS95l&filename=g24JGYuT74A.004.mp4',
    '429713688?token=qDT95nz7hfm2Nki&filename=JZNMxa3OKHY.000.mp4',
    '429713689?token=noLguEGXDpbcKhg&filename=nvlqJbHk_Lc.003.mp4',
    '429713679?token=9L7RQ0hgdJlcek6&filename=4vdJGgZpj4k.003.mp4'
]

for curr_files in tets_name_files:
    _b5.download_file_from_url(url = domain + curr_files, out = True)

# Получение прогнозов
_b5.path_to_dataset_ = PATH_TO_DIR # Директория набора данных
_b5.ext_ = ['.mp4'] # Расширения искомых файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_[corpus]['sberdisk']

_b5.get_avt_predictions(url_accuracy = url_accuracy, lang = 'en')
```

[2023-12-16 19:05:15] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-16 19:05:17] Получение прогнозов и вычисление точности (мультимодальное объединение) ...

10 из 10 (100.0%) ... GitHub:nbsphinx-math:*OCEANAI*_guide:nbsphinx-math:*notebooks*_FI:nbsphinx-math:*test*_plk5k7PBEg.003.mp4
...

Person ID	Path	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism
1	2d6btbaNdfo.000.mp4	0.581159	0.628822	0.466609		
2	300gK3CnzW0.001.mp4	0.463991	0.418851	0.41301		
3	300gK3CnzW0.003.mp4	0.454281	0.415049	0.39189		
4	4vdJGgZpj4k.003.mp4	0.588461	0.643233	0.530789		
5	be0DQawtVkE.002.mp4	0.633433	0.533295	0.523742		
6	cLaZxEf1nE4.004.mp4	0.636944	0.542386	0.558461		
7	g24JGYuT74A.004.mp4	0.531518	0.376987	0.393309		
8	JZNMxa3OKHY.000.mp4	0.610342	0.541418	0.563163		
9	nvlqJbHk_Lc.003.mp4	0.495809	0.458526	0.414436		
10	_plk5k7PBEg.003.mp4	0.60707	0.591893	0.520662		

(continues on next page)

(продолжение с предыдущей страницы)

```
Person ID
1              0.622129        0.553832
2              0.493329        0.423093
3              0.485114        0.420741
4              0.603038        0.593398
5              0.608591        0.588456
6              0.570975        0.558983
7                0.4904        0.447881
8              0.595013        0.569461
9              0.469152        0.435461
10             0.603938        0.565726
```

[2023-12-16 19:05:17] Точность по отдельным персональным качествам личности человека ...

```
          Openness  Conscientiousness  Extraversion  Agreeableness  \
Metrics
MAE         0.0589             0.0612        0.0864         0.0697
Accuracy    0.9411             0.9388        0.9136         0.9303

          Non-Neuroticism    Mean
Metrics
MAE                0.0582  0.0669
Accuracy           0.9418  0.9331
```

[2023-12-16 19:05:17] Средняя средних абсолютных ошибок: 0.0669, средняя точность: 0.9331 ...

Лог файлы успешно сохранены ...

— Время выполнения: 64.147 сек. —

[2]:
```
True
```

Для прогнозирования потребительских предпочтений в промышленных товарах необходимо знать коэффициенты корреляции, определяющие взаимосвязь между персональными качествами личности человека и предпочтениями в товарах или услугах.

В качестве примера предлагается использование коэффициентов корреляции между персональными качествами человека и характеристиками автомобилей, представленными в статье:

1) O'Connor P. J. et al. What Drives Consumer Automobile Choice? Investigating Personality Trait Predictors of Vehicle Preference Factors // Personality and Individual Differences. – 2022. – Vol. 184. – pp. 111220.

Пользователь может установить свои коэффициенты корреляции.

Прогнозирование потребительских предпочтений на характеристики атомобиля

[3]:
```
# Загрузка датафрейма с коэффициентами корреляции
url = 'https://download.sberdisk.ru/download/file/478675818?token=EjfLMqOeK8cfnOu&
↪filename=auto_characteristics.csv'
df_correlation_coefficients = pd.read_csv(url)
df_correlation_coefficients = pd.DataFrame(
    df_correlation_coefficients.drop(['Style and performance', 'Safety and practicality
↪'], axis = 1)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
)
df_correlation_coefficients.index.name = 'ID'
df_correlation_coefficients.index += 1
df_correlation_coefficients.index = df_correlation_coefficients.index.map(str)

df_correlation_coefficients
```

[3]:

ID	Trait	Performance	Classic car features	Luxury additions	Fashion and attention	Recreation	Technology	Family friendly	Safe and reliable	Practical and easy to use	Economical/low cost	Basic features
1	Openness	0.020000	-0.033333	-0.030000	-0.050000	0.033333	0.013333	-0.030000	0.136667	0.106667	0.093333	0.006667
2	Conscientiousness	0.013333	-0.193333	-0.063333	-0.096667	-0.096667	0.086667	-0.063333	0.280000	0.180000	0.130000	0.143333
3	Extraversion	0.133333	0.060000	0.106667	0.123333	0.126667	0.120000	0.090000	0.136667	0.043333	0.073333	0.050000
4	Agreeableness	-0.036667	-0.193333	-0.133333	-0.133333	-0.090000	0.046667	-0.016667	0.240000	0.160000	0.120000	0.083333
5	Non-Neuroticism	0.016667	-0.006667	-0.010000	-0.006667	-0.033333	0.046667	-0.023333	0.093333	0.046667	0.046667	-0.040000

[4]:
```
_b5._priority_calculation(
    correlation_coefficients = df_correlation_coefficients,
    col_name_ocean = 'Trait',
    threshold = 0.55,
    number_priority = 3,
    number_importance_traits = 3,
    out = False
)

_b5._save_logs(df = _b5.df_files_priority_, name = 'auto_characteristics_priorities_fi_en
↪', out = True)

# Опционно
```

(continues on next page)

(продолжение с предыдущей страницы)

```
df = _b5.df_files_priority_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON
↪', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = ['OPE', 'CON', 'EXT', 'AGR', 'NNEU']
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[4]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU	Priority 1	Priority 2	Priority 3	Trait importance 1	Trait importance 2	Trait importance 3
1	2d6btbaNdfo.000.mp4	0.581	0.629	0.467	0.622	0.554	Safe and reliable	Practical and easy to use	Economical/low cost	Conscientiousness	Agreeableness	Openness
2	300gK3CnzW0.001.mp4	0.464	0.419	0.413	0.493	0.423	Classic car features	Fashion and attention	Luxury additions	Agreeableness	Conscientiousness	Openness
3	300gK3CnzW0.003.mp4	0.454	0.415	0.392	0.485	0.421	Classic car features	Fashion and attention	Luxury additions	Agreeableness	Conscientiousness	Openness
4	4vdJGgZpj4k.003.mp4	0.588	0.643	0.531	0.603	0.593	Safe and reliable	Practical and easy to use	Economical/low cost	Conscientiousness	Agreeableness	Openness
5	be0DQawtVkE.002.mp4	0.633	0.533	0.524	0.609	0.588	Practical and easy to use	Safe and reliable	Economical/low cost	Agreeableness	Openness	Non-Neuroticism
6	cLaZxEf1nE4.004.mp4	0.637	0.542	0.558	0.571	0.559	Safe and reliable	Economical/low cost	Practical and easy to use	Agreeableness	Openness	Non-Neuroticism
7	g24JGYuT74A.004.mp4	0.532	0.377	0.393	0.490	0.448	Classic car features	Fashion and attention	Luxury additions	Agreeableness	Conscientiousness	Openness
8	JZNMxa3OKHY.000.mp4	0.610	0.541	0.563	0.595	0.569	Safe and reliable	Economical/low cost	Practical and easy to use	Agreeableness	Openness	Non-Neuroticism
9	nvlqJbHk_Lc.003.mp4	0.496	0.459	0.414	0.469	0.435	Classic car features	Fashion and attention	Luxury additions	Agreeableness	Conscientiousness	Openness
10	_plk5k7PBEg.003.mp4	0.607	0.592	0.521	0.604	0.566	Safe and reliable	Practical and easy to use	Economical/low cost	Conscientiousness	Agreeableness	Openness

(continues on next page)

(продолжение с предыдущей страницы)

```
6               Extraversion
7                   Openness
8               Extraversion
9                   Openness
10                  Openness
```

Прогнозирование потребительских предпочтений на характеристики мобильного устройства

В качестве примера предлагается использование коэффициентов корреляции между персональными качествами человека и характеристиками мобильного устройства, представленными в статье:

1) Peltonen E., Sharmila P., Asare K. O., Visuri A., Lagerspetz E., Ferreira D. (2020). When phones get personal: Predicting Big Five personality traits from application usage // Pervasive and Mobile Computing. – 2020. – Vol. 69. – 101269.

[5]:
```
# Загрузка датафрейма с коэффициентами корреляции
url = 'https://download.sberdisk.ru/download/file/478676690?token=7KcAxPqMpWiYQnx&
↪filename=divice_characteristics.csv'
df_divice_characteristics = pd.read_csv(url)

df_divice_characteristics.index.name = 'ID'
df_divice_characteristics.index += 1
df_divice_characteristics.index = df_divice_characteristics.index.map(str)

df_divice_characteristics
```

[5]:
```
                Trait  Communication  Game Action  Game Board  Game Casino  \
ID
1            Openness          0.118        0.056       0.079        0.342
2   Conscientiousness          0.119        0.043       0.107        0.448
3        Extraversion          0.246        0.182       0.211        0.311
4       Agreeableness          0.218        0.104       0.164        0.284
5     Non-Neuroticism          0.046        0.047       0.125        0.515

    Game Educational  Game Simulation  Game Trivia  Entertainment  Finance  \
ID
1              0.027            0.104        0.026          0.000    0.006
2              0.039            0.012        0.119          0.000    0.005
3              0.102            0.165        0.223          0.001    0.003
4              0.165            0.122        0.162          0.000    0.003
5              0.272            0.179        0.214          0.002    0.030

    Health and Fitness  Media and Video  Music and Audio  News and Magazines  \
ID
1                0.002            0.000            0.000               0.001
2                0.001            0.000            0.002               0.002
3                0.000            0.001            0.001               0.001
4                0.001            0.000            0.002               0.002
5                0.001            0.000            0.005               0.003

    Personalisation  Travel and Local  Weather
```

(continues on next page)

(продолжение с предыдущей страницы)

```
ID
1             0.004             0.002    0.004
2             0.001             0.001    0.003
3             0.004             0.009    0.003
4             0.001             0.004    0.003
5             0.008             0.004    0.007
```

[6]:
```
_b5._priority_calculation(
    correlation_coefficients = df_divice_characteristics,
    col_name_ocean = 'Trait',
    threshold = 0.55,
    number_priority = 3,
    number_importance_traits = 3,
    out = True
)

_b5._save_logs(df = _b5.df_files_priority_, name = 'divice_characteristics_priorities_fi_
↪en', out = True)

# Опционно
df = _b5.df_files_priority_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON
↪', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = ['OPE', 'CON', 'EXT', 'AGR', 'NNEU']
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[6]:
```
                          Path    OPE    CON    EXT    AGR   NNEU  \
Person ID
1          2d6btbaNdfo.000.mp4  0.581  0.629  0.467  0.622  0.554
2          300gK3CnzW0.001.mp4  0.464  0.419  0.413  0.493  0.423
3          300gK3CnzW0.003.mp4  0.454  0.415  0.392  0.485  0.421
4          4vdJGgZpj4k.003.mp4  0.588  0.643  0.531  0.603  0.593
5          be0DQawtVkE.002.mp4  0.633  0.533  0.524  0.609  0.588
6          cLaZxEf1nE4.004.mp4  0.637  0.542  0.558  0.571  0.559
7          g24JGYuT74A.004.mp4  0.532  0.377  0.393  0.490  0.448
8          JZNMxa3OKHY.000.mp4  0.610  0.541  0.563  0.595  0.569
9          nvlqJbHk_Lc.003.mp4  0.496  0.459  0.414  0.469  0.435
10         _plk5k7PBEg.003.mp4  0.607  0.592  0.521  0.604  0.566

                Priority 1        Priority 2          Priority 3  \
Person ID
1              Game Casino  Game Educational         Game Trivia
2          Media and Video     Entertainment  Health and Fitness
3          Media and Video     Entertainment  Health and Fitness
4              Game Casino  Game Educational         Game Trivia
5              Game Casino  Game Educational     Game Simulation
6              Game Casino   Game Simulation    Game Educational
7          Media and Video     Entertainment  Health and Fitness
8              Game Casino   Game Simulation    Game Educational
9          Media and Video     Entertainment  Health and Fitness
10             Game Casino  Game Educational         Game Trivia
```

(continues on next page)

(продолжение с предыдущей страницы)

```
          Trait importance 1 Trait importance 2 Trait importance 3
Person ID
1            Non-Neuroticism  Conscientiousness      Agreeableness
2          Conscientiousness      Agreeableness       Extraversion
3          Conscientiousness      Agreeableness       Extraversion
4            Non-Neuroticism  Conscientiousness      Agreeableness
5            Non-Neuroticism      Agreeableness           Openness
6            Non-Neuroticism      Agreeableness       Extraversion
7          Conscientiousness      Agreeableness       Extraversion
8            Non-Neuroticism      Agreeableness       Extraversion
9          Conscientiousness      Agreeableness       Extraversion
10           Non-Neuroticism      Agreeableness  Conscientiousness
```

MuPTA **(ru)**

[7]:
```
import os
import pandas as pd

# Импорт модуля
from oceanai.modules.lab.build import Run

# Создание экземпляра класса
_b5 = Run()

corpus = 'mupta'
lang = 'ru'

# Настройка ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000      # Размер загрузки файла из сети за 1 шаг

# Формирование аудиомоделей
res_load_model_hc = _b5.load_audio_model_hc()
res_load_model_nn = _b5.load_audio_model_nn()

# Загрузка весов аудиомоделей
url = _b5.weights_for_big5_['audio'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_audio_model_weights_hc(url = url)

url = _b5.weights_for_big5_['audio'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_audio_model_weights_nn(url = url)

# Формирование видеомоделей
res_load_model_hc = _b5.load_video_model_hc(lang=lang)
res_load_model_deep_fe = _b5.load_video_model_deep_fe()
res_load_model_nn = _b5.load_video_model_nn()

# Загрузка весов видеомоделей
url = _b5.weights_for_big5_['video'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_video_model_weights_hc(url = url)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
url = _b5.weights_for_big5_['video'][corpus]['fe']['sberdisk']
res_load_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(url = url)

url = _b5.weights_for_big5_['video'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_video_model_weights_nn(url = url)

# Загрузка словаря с экспертными признаками (текстовая модальность)
res_load_text_features = _b5.load_text_features()

# Формирование текстовых моделей
res_setup_translation_model = _b5.setup_translation_model() # только для русского языка
res_setup_translation_model = _b5.setup_bert_encoder(force_reload = False)
res_load_text_model_hc_fi = _b5.load_text_model_hc(corpus=corpus)
res_load_text_model_nn_fi = _b5.load_text_model_nn(corpus=corpus)

# Загрузка весов текстовых моделей
url = _b5.weights_for_big5_['text'][corpus]['hc']['sberdisk']
res_load_text_model_weights_hc_fi = _b5.load_text_model_weights_hc(url = url)

url = _b5.weights_for_big5_['text'][corpus]['nn']['sberdisk']
res_load_text_model_weights_nn_fi = _b5.load_text_model_weights_nn(url = url)

# Формирование модели для мультимодального объединения информации
res_load_avt_model_b5 = _b5.load_avt_model_b5()

# Загрузка весов модели для мультимодального объединения информации
url = _b5.weights_for_big5_['avt'][corpus]['b5']['sberdisk']
res_load_avt_model_weights_b5 = _b5.load_avt_model_weights_b5(url = url)

PATH_TO_DIR = './video_MuPTA/'
PATH_SAVE_VIDEO = './video_MuPTA/test/'

_b5.path_to_save_ = PATH_SAVE_VIDEO

# Загрузка 10 тестовых аудиовидеозаписей из корпуса MuPTA
# URL: https://hci.nw.ru/en/pages/mupta-corpus
domain = 'https://download.sberdisk.ru/download/file/'
tets_name_files = [
    '477995979?token=2cvyk7CS0mHx2MJ&filename=speaker_06_center_83.mov',
    '477995980?token=jGPtBPS69uzFU6Y&filename=speaker_01_center_83.mov',
    '477995967?token=zCaRbNB6ht5wMPq&filename=speaker_11_center_83.mov',
    '477995966?token=B1rbinDYRQKrI3T&filename=speaker_15_center_83.mov',
    '477995978?token=dEpVDtZg1EQiEQ9&filename=speaker_07_center_83.mov',
    '477995961?token=o1hVjw8G45q9L9Z&filename=speaker_19_center_83.mov',
    '477995964?token=5K22OAqf673VHPq&filename=speaker_23_center_83.mov',
    '477995965?token=v1LVD2KT1cU7Lpb&filename=speaker_24_center_83.mov',
    '477995962?token=tmaSGyyWLA6XCy9&filename=speaker_27_center_83.mov',
    '477995963?token=bTpo96qNDPcwGqb&filename=speaker_10_center_83.mov',
]

for curr_files in tets_name_files:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    _b5.download_file_from_url(url = domain + curr_files, out = True)

# Получение прогнозов
_b5.path_to_dataset_ = PATH_TO_DIR # Директория набора данных
_b5.ext_ = ['.mov'] # Расширения искомых файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_['mupta']['sberdisk']

_b5.get_avt_predictions(url_accuracy = url_accuracy, lang = lang)
```

[2023-12-16 19:13:25] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-16 19:13:30] Получение прогнозов и вычисление точности (мультимодальное объединение) ...

10 из 10 (100.0%) ... GitHub:nbsphinx-math:*OCEANAI*_guide:nbsphinx-math:*notebooks*_MuPTA:nbsphinx-math:*test*_27_center_83.mov
...

Person ID	Path	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism
1	speaker_01_center_83.mov	0.758137	0.693356	0.650108	0.744589	0.488671
2	speaker_06_center_83.mov	0.681602	0.654339	0.607156	0.731282	0.417908
3	speaker_07_center_83.mov	0.666104	0.656836	0.567863	0.685067	0.378102
4	speaker_10_center_83.mov	0.694171	0.596195	0.571414	0.66223	0.348639
5	speaker_11_center_83.mov	0.712885	0.594764	0.571709	0.716696	0.37802
6	speaker_15_center_83.mov	0.664158	0.670411	0.60421	0.696056	0.399842
7	speaker_19_center_83.mov	0.761213	0.652635	0.651028	0.788677	0.459676
8	speaker_23_center_83.mov	0.692788	0.68324	0.616737	0.795205	0.447242
9	speaker_24_center_83.mov	0.705923	0.658382	0.610645	0.697415	0.411988
10	speaker_27_center_83.mov	0.753417	0.708372	0.654608	0.816416	0.504743

[2023-12-16 19:13:30] Точность по отдельным персональным качествам личности человека ...

Metrics	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism	Mean
MAE	0.0673	0.0789	0.1325	0.102		
Accuracy	0.9327	0.9211	0.8675	0.898		

(continues on next page)

(продолжение с предыдущей страницы)

```
Metrics
MAE                 0.1002  0.0962
Accuracy            0.8998  0.9038
```

[2023-12-16 19:13:30] Средняя средних абсолютных ошибок: 0.0962, средняя точность: 0.9038 ...

Лог файлы успешно сохранены ...

— Время выполнения: 416.453 сек. —

[7]:
```
True
```

Для прогнозирования потребительских предпочтений в промышленных товарах необходимо знать коэффициенты корреляции, определяющие взаимосвязь между персональными качествами личности человека и предпочтениями в товарах или услугах.

В качестве примера предлагается использование коэффициентов корреляции между персональными качествами человека и характеристиками автомобилей, представленными в статье:

1) O'Connor P. J. et al. What Drives Consumer Automobile Choice? Investigating Personality Trait Predictors of Vehicle Preference Factors // Personality and Individual Differences. – 2022. – Vol. 184. – pp. 111220.

Пользователь может установить свои коэффициенты корреляции.

Прогнозирование потребительских предпочтений на характеристики атомобиля

[8]:
```
# Загрузка датафрейма с коэффициентами корреляции
url = 'https://download.sberdisk.ru/download/file/478675818?token=EjfLMqOeK8cfnOu&
↪filename=auto_characteristics.csv'
df_correlation_coefficients = pd.read_csv(url)
df_correlation_coefficients = pd.DataFrame(
    df_correlation_coefficients.drop(['Style and performance', 'Safety and practicality
↪'], axis = 1)
)
df_correlation_coefficients.index.name = 'ID'
df_correlation_coefficients.index += 1
df_correlation_coefficients.index = df_correlation_coefficients.index.map(str)

df_correlation_coefficients
```

[8]:

ID	Trait	Performance	Classic car features	Luxury additions	Fashion and attention	Recreation	Technology	Family friendly
1	Openness	0.020000	-0.033333	-0.030000	-0.050000	0.033333	0.013333	-0.030000
2	Conscientiousness	0.013333	-0.193333	-0.063333	-0.096667	-0.096667	0.086667	-0.063333
3	Extraversion	0.133333	0.060000	0.106667	0.123333	0.126667	0.120000	0.090000

(continues on next page)

(продолжение с предыдущей страницы)

```
4               -0.133333   -0.090000    0.046667        -0.016667
5               -0.006667   -0.033333    0.046667        -0.023333

    Safe and reliable  Practical and easy to use  Economical/low cost  \
ID
1            0.136667                   0.106667             0.093333
2            0.280000                   0.180000             0.130000
3            0.136667                   0.043333             0.073333
4            0.240000                   0.160000             0.120000
5            0.093333                   0.046667             0.046667

    Basic features
ID
1         0.006667
2         0.143333
3         0.050000
4         0.083333
5        -0.040000
```

[9]:
```
_b5._priority_calculation(
    correlation_coefficients = df_correlation_coefficients,
    col_name_ocean = 'Trait',
    threshold = 0.55,
    number_priority = 3,
    number_importance_traits = 3,
    out = False
)

_b5._save_logs(df = _b5.df_files_priority_, name = 'auto_characteristics_priorities_
↪mupta_ru', out = True)

# Опционно
df = _b5.df_files_priority_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON
↪', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = ['OPE', 'CON', 'EXT', 'AGR', 'NNEU']
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[9]:
```
                              Path    OPE    CON    EXT    AGR   NNEU  \
Person ID
1          speaker_01_center_83.mov  0.758  0.693  0.650  0.745  0.489
2          speaker_06_center_83.mov  0.682  0.654  0.607  0.731  0.418
3          speaker_07_center_83.mov  0.666  0.657  0.568  0.685  0.378
4          speaker_10_center_83.mov  0.694  0.596  0.571  0.662  0.349
5          speaker_11_center_83.mov  0.713  0.595  0.572  0.717  0.378
6          speaker_15_center_83.mov  0.664  0.670  0.604  0.696  0.400
7          speaker_19_center_83.mov  0.761  0.653  0.651  0.789  0.460
8          speaker_23_center_83.mov  0.693  0.683  0.617  0.795  0.447
9          speaker_24_center_83.mov  0.706  0.658  0.611  0.697  0.412
10         speaker_27_center_83.mov  0.753  0.708  0.655  0.816  0.505

                 Priority 1                Priority 2           Priority 3  \
```

(continues on next page)

(продолжение с предыдущей страницы)

```
Person ID
1          Safe and reliable  Practical and easy to use  Economical/low cost
2          Safe and reliable  Practical and easy to use  Economical/low cost
3          Safe and reliable  Practical and easy to use  Economical/low cost
4          Safe and reliable  Practical and easy to use  Economical/low cost
5          Safe and reliable  Practical and easy to use  Economical/low cost
6          Safe and reliable  Practical and easy to use  Economical/low cost
7          Safe and reliable  Practical and easy to use  Economical/low cost
8          Safe and reliable  Practical and easy to use  Economical/low cost
9          Safe and reliable  Practical and easy to use  Economical/low cost
10         Safe and reliable  Practical and easy to use  Economical/low cost

          Trait importance 1 Trait importance 2 Trait importance 3
Person ID
1          Conscientiousness      Agreeableness           Openness
2          Conscientiousness      Agreeableness           Openness
3          Conscientiousness      Agreeableness           Openness
4          Conscientiousness      Agreeableness           Openness
5              Agreeableness  Conscientiousness           Openness
6          Conscientiousness      Agreeableness           Openness
7              Agreeableness  Conscientiousness           Openness
8              Agreeableness  Conscientiousness           Openness
9          Conscientiousness      Agreeableness           Openness
10             Agreeableness  Conscientiousness           Openness
```

Прогнозирование потребительских предпочтений на характеристики мобильного устройства

В качестве примера предлагается использование коэффициентов корреляции между персональными качествами человека и характеристиками мобильного устройства, представленными в статье:

1) Peltonen E., Sharmila P., Asare K. O., Visuri A., Lagerspetz E., Ferreira D. (2020). When phones get personal: Predicting Big Five personality traits from application usage // Pervasive and Mobile Computing. – 2020. – Vol. 69. – 101269.

[10]:
```
# Загрузка датафрейма с коэффициентами корреляции
url = 'https://download.sberdisk.ru/download/file/478676690?token=7KcAxPqMpWiYQnx&
↪filename=divice_characteristics.csv'
df_divice_characteristics = pd.read_csv(url)

df_divice_characteristics.index.name = 'ID'
df_divice_characteristics.index += 1
df_divice_characteristics.index = df_divice_characteristics.index.map(str)

df_divice_characteristics
```

[10]:
```
                Trait  Communication  Game Action  Game Board  Game Casino  \
ID
1            Openness          0.118        0.056       0.079        0.342
2   Conscientiousness          0.119        0.043       0.107        0.448
3        Extraversion          0.246        0.182       0.211        0.311
4       Agreeableness          0.218        0.104       0.164        0.284
```

(continues on next page)

(продолжение с предыдущей страницы)

```
5     Non-Neuroticism          0.046        0.047       0.125        0.515

    Game Educational  Game Simulation  Game Trivia  Entertainment  Finance  \
ID
1              0.027            0.104        0.026          0.000    0.006
2              0.039            0.012        0.119          0.000    0.005
3              0.102            0.165        0.223          0.001    0.003
4              0.165            0.122        0.162          0.000    0.003
5              0.272            0.179        0.214          0.002    0.030

    Health and Fitness  Media and Video  Music and Audio  News and Magazines  \
ID
1                0.002            0.000            0.000               0.001
2                0.001            0.000            0.002               0.002
3                0.000            0.001            0.001               0.001
4                0.001            0.000            0.002               0.002
5                0.001            0.000            0.005               0.003

    Personalisation  Travel and Local  Weather
ID
1             0.004             0.002    0.004
2             0.001             0.001    0.003
3             0.004             0.009    0.003
4             0.001             0.004    0.003
5             0.008             0.004    0.007
```

[11]:
```
_b5._priority_calculation(
    correlation_coefficients = df_divice_characteristics,
    col_name_ocean = 'Trait',
    threshold = 0.55,
    number_priority = 3,
    number_importance_traits = 3,
    out = True
)

_b5._save_logs(df = _b5.df_files_priority_, name = 'divice_characteristics_priorities_
↪mupta_ru', out = True)

# Опционно
df = _b5.df_files_priority_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON
↪', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = ['OPE', 'CON', 'EXT', 'AGR', 'NNEU']
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[11]:
```
                              Path    OPE    CON    EXT    AGR   NNEU  \
Person ID
1          speaker_01_center_83.mov  0.758  0.693  0.650  0.745  0.489
2          speaker_06_center_83.mov  0.682  0.654  0.607  0.731  0.418
3          speaker_07_center_83.mov  0.666  0.657  0.568  0.685  0.378
4          speaker_10_center_83.mov  0.694  0.596  0.571  0.662  0.349
5          speaker_11_center_83.mov  0.713  0.595  0.572  0.717  0.378
```

(continues on next page)

(продолжение с предыдущей страницы)

```
6          speaker_15_center_83.mov  0.664  0.670  0.604  0.696  0.400
7          speaker_19_center_83.mov  0.761  0.653  0.651  0.789  0.460
8          speaker_23_center_83.mov  0.693  0.683  0.617  0.795  0.447
9          speaker_24_center_83.mov  0.706  0.658  0.611  0.697  0.412
10         speaker_27_center_83.mov  0.753  0.708  0.655  0.816  0.505

            Priority 1     Priority 2  Priority 3 Trait importance 1  \
Person ID
1          Game Casino  Communication  Game Board       Extraversion
2          Game Casino  Communication  Game Board      Agreeableness
3          Game Casino  Communication  Game Board      Agreeableness
4          Game Casino  Communication  Game Board      Agreeableness
5          Game Casino  Communication  Game Board      Agreeableness
6          Game Casino  Communication  Game Board       Extraversion
7          Game Casino  Communication  Game Board      Agreeableness
8          Game Casino  Communication  Game Board      Agreeableness
9          Game Casino  Communication  Game Board       Extraversion
10         Game Casino  Communication  Game Board      Agreeableness

          Trait importance 2 Trait importance 3
Person ID
1              Agreeableness  Conscientiousness
2               Extraversion  Conscientiousness
3          Conscientiousness       Extraversion
4               Extraversion  Conscientiousness
5               Extraversion  Conscientiousness
6              Agreeableness  Conscientiousness
7               Extraversion  Conscientiousness
8               Extraversion  Conscientiousness
9              Agreeableness  Conscientiousness
10              Extraversion  Conscientiousness
```

MuPTA **(en)**

[12]:
```
import os
import pandas as pd

# Импорт модуля
from oceanai.modules.lab.build import Run

# Создание экземпляра класса
_b5 = Run()

corpus = 'fi'
lang = 'en'

# Настройка ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000      # Размер загрузки файла из сети за 1 шаг
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Формирование аудиомоделей
res_load_model_hc = _b5.load_audio_model_hc()
res_load_model_nn = _b5.load_audio_model_nn()

# Загрузка весов аудиомоделей
url = _b5.weights_for_big5_['audio'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_audio_model_weights_hc(url = url)

url = _b5.weights_for_big5_['audio'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_audio_model_weights_nn(url = url)

# Формирование видеомоделей
res_load_model_hc = _b5.load_video_model_hc(lang=lang)
res_load_model_deep_fe = _b5.load_video_model_deep_fe()
res_load_model_nn = _b5.load_video_model_nn()

# Загрузка весов видеомоделей
url = _b5.weights_for_big5_['video'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_video_model_weights_hc(url = url)

url = _b5.weights_for_big5_['video'][corpus]['fe']['sberdisk']
res_load_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(url = url)

url = _b5.weights_for_big5_['video'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_video_model_weights_nn(url = url)

# Загрузка словаря с экспертными признаками (текстовая модальность)
res_load_text_features = _b5.load_text_features()

# Формирование текстовых моделей
res_setup_translation_model = _b5.setup_translation_model() # только для русского языка
res_setup_translation_model = _b5.setup_bert_encoder(force_reload = False)
res_load_text_model_hc_fi = _b5.load_text_model_hc(corpus=corpus)
res_load_text_model_nn_fi = _b5.load_text_model_nn(corpus=corpus)

# Загрузка весов текстовых моделей
url = _b5.weights_for_big5_['text'][corpus]['hc']['sberdisk']
res_load_text_model_weights_hc_fi = _b5.load_text_model_weights_hc(url = url)

url = _b5.weights_for_big5_['text'][corpus]['nn']['sberdisk']
res_load_text_model_weights_nn_fi = _b5.load_text_model_weights_nn(url = url)

# Формирование модели для мультимодального объединения информации
res_load_avt_model_b5 = _b5.load_avt_model_b5()

# Загрузка весов модели для мультимодального объединения информации
url = _b5.weights_for_big5_['avt'][corpus]['b5']['sberdisk']
res_load_avt_model_weights_b5 = _b5.load_avt_model_weights_b5(url = url)

PATH_TO_DIR = './video_MuPTA/'
PATH_SAVE_VIDEO = './video_MuPTA/test/'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
_b5.path_to_save_ = PATH_SAVE_VIDEO

# Загрузка 10 тестовых аудиовидеозаписей из корпуса MuPTA
# URL: https://hci.nw.ru/en/pages/mupta-corpus
domain = 'https://download.sberdisk.ru/download/file/'
tets_name_files = [
    '477995979?token=2cvyk7CS0mHx2MJ&filename=speaker_06_center_83.mov',
    '477995980?token=jGPtBPS69uzFU6Y&filename=speaker_01_center_83.mov',
    '477995967?token=zCaRbNB6ht5wMPq&filename=speaker_11_center_83.mov',
    '477995966?token=B1rbinDYRQKrI3T&filename=speaker_15_center_83.mov',
    '477995978?token=dEpVDtZg1EQiEQ9&filename=speaker_07_center_83.mov',
    '477995961?token=o1hVjw8G45q9L9Z&filename=speaker_19_center_83.mov',
    '477995964?token=5K22OAqf673VHPq&filename=speaker_23_center_83.mov',
    '477995965?token=v1LVD2KT1cU7Lpb&filename=speaker_24_center_83.mov',
    '477995962?token=tmaSGyyWLA6XCy9&filename=speaker_27_center_83.mov',
    '477995963?token=bTpo96qNDPcwGqb&filename=speaker_10_center_83.mov',
]

for curr_files in tets_name_files:
    _b5.download_file_from_url(url = domain + curr_files, out = True)

# Получение прогнозов
_b5.path_to_dataset_ = PATH_TO_DIR # Директория набора данных
_b5.ext_ = ['.mov'] # Расширения искомых файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_['mupta']['sberdisk']

_b5.get_avt_predictions(url_accuracy = url_accuracy, lang = lang)
```

[2023-12-16 19:20:55] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-16 19:20:57] Получение прогнозов и вычисление точности (мультимодальное объединение) ...

10 из 10 (100.0%) ... GitHub:nbsphinx-math:*OCEANAI*_guide:nbsphinx-math:*notebooks*_MuPTA:nbsphinx-math:*test*_27_center_83.mov
...

Person ID	Path	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism
1	speaker_01_center_83.mov	0.564985	0.539052	0.440615		
2	speaker_06_center_83.mov	0.650774	0.663849	0.607308		
3	speaker_07_center_83.mov	0.435976	0.486683	0.313828		
4	speaker_10_center_83.mov	0.498542	0.511243	0.412592		
5	speaker_11_center_83.mov	0.394776	0.341608	0.327082		
6	speaker_15_center_83.mov	0.566107	0.543811	0.492766		
7	speaker_19_center_83.mov	0.506271	0.438215	0.430894		
8	speaker_23_center_83.mov	0.486463	0.521755	0.309894		
9	speaker_24_center_83.mov	0.417404	0.473339	0.320714		
10	speaker_27_center_83.mov	0.526112	0.661107	0.443167		

(continues on next page)

(продолжение с предыдущей страницы)

```
1              0.59251        0.488763
2             0.643847        0.620627
3             0.415446        0.396618
4             0.468947         0.44399
5             0.427304        0.354936
6             0.587411        0.499433
7             0.456177         0.44075
8             0.432291        0.433601
9             0.445086        0.414649
10            0.558965        0.554224
```

[2023-12-16 19:20:57] Точность по отдельным персональным качествам личности человека ...

```
          Openness  Conscientiousness  Extraversion  Agreeableness  \
Metrics
MAE         0.1727             0.1672        0.1661         0.2579
Accuracy    0.8273             0.8328        0.8339         0.7421

          Non-Neuroticism    Mean
Metrics
MAE                 0.107  0.1742
Accuracy            0.893  0.8258
```

[2023-12-16 19:20:57] Средняя средних абсолютных ошибок: 0.1742, средняя точность: 0.8258 ...

Лог файлы успешно сохранены ...

— Время выполнения: 379.936 сек. —

[12]:
```
True
```

Для прогнозирования потребительских предпочтений в промышленных товарах необходимо знать коэффициенты корреляции, определяющие взаимосвязь между персональными качествами личности человека и предпочтениями в товарах или услугах.

В качестве примера предлагается использование коэффициентов корреляции между персональными качествами человека и характеристиками автомобилей, представленными в статье:

1) O'Connor P. J. et al. What Drives Consumer Automobile Choice? Investigating Personality Trait Predictors of Vehicle Preference Factors // Personality and Individual Differences. – 2022. – Vol. 184. – pp. 111220.

Пользователь может установить свои коэффициенты корреляции.

Прогнозирование потребительских предпочтений на характеристики атомобиля

[13]:
```
# Загрузка датафрейма с коэффициентами корреляции
url = 'https://download.sberdisk.ru/download/file/478675818?token=EjfLMqOeK8cfnOu&
↪filename=auto_characteristics.csv'
df_correlation_coefficients = pd.read_csv(url)
df_correlation_coefficients = pd.DataFrame(
    df_correlation_coefficients.drop(['Style and performance', 'Safety and practicality
↪'], axis = 1)
)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
df_correlation_coefficients.index.name = 'ID'
df_correlation_coefficients.index += 1
df_correlation_coefficients.index = df_correlation_coefficients.index.map(str)

df_correlation_coefficients
```

[13]:

ID	Trait	Performance	Classic car features	Luxury additions	Fashion and attention	Recreation	Technology	Family friendly	Safe and reliable	Practical and easy to use	Economical/low cost	Basic features
1	Openness	0.020000	-0.033333	-0.030000	-0.050000	0.033333	0.013333	-0.030000	0.136667	0.106667	0.093333	0.006667
2	Conscientiousness	0.013333	-0.193333	-0.063333	-0.096667	-0.096667	0.086667	-0.063333	0.280000	0.180000	0.130000	0.143333
3	Extraversion	0.133333	0.060000	0.106667	0.123333	0.126667	0.120000	0.090000	0.136667	0.043333	0.073333	0.050000
4	Agreeableness	-0.036667	-0.193333	-0.133333	-0.133333	-0.090000	0.046667	-0.016667	0.240000	0.160000	0.120000	0.083333
5	Non-Neuroticism	0.016667	-0.006667	-0.010000	-0.006667	-0.033333	0.046667	-0.023333	0.093333	0.046667	0.046667	-0.040000

[14]:

```
_b5._priority_calculation(
    correlation_coefficients = df_correlation_coefficients,
    col_name_ocean = 'Trait',
    threshold = 0.55,
    number_priority = 3,
    number_importance_traits = 3,
    out = False
)

_b5._save_logs(df = _b5.df_files_priority_, name = 'auto_characteristics_priorities_
↪mupta_en', out = True)

# Опционно
df = _b5.df_files_priority_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON
```

(continues on next page)

(продолжение с предыдущей страницы)

```
↪', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = ['OPE', 'CON', 'EXT', 'AGR', 'NNEU']
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[14]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU	Priority 1	Priority 2	Priority 3	Trait importance 1	Trait importance 2	Trait importance 3
1	speaker_01_center_83.mov	0.565	0.539	0.441	0.593	0.489	Practical and easy to use	Economical/low cost	Family friendly	Agreeableness	Openness	Non-Neuroticism
2	speaker_06_center_83.mov	0.651	0.664	0.607	0.644	0.621	Safe and reliable	Practical and easy to use	Economical/low cost	Conscientiousness	Agreeableness	Openness
3	speaker_07_center_83.mov	0.436	0.487	0.314	0.415	0.397	Classic car features	Fashion and attention	Luxury additions	Agreeableness	Conscientiousness	Openness
4	speaker_10_center_83.mov	0.499	0.511	0.413	0.469	0.444	Classic car features	Fashion and attention	Luxury additions	Agreeableness	Conscientiousness	Openness
5	speaker_11_center_83.mov	0.395	0.342	0.327	0.427	0.355	Classic car features	Fashion and attention	Luxury additions	Agreeableness	Conscientiousness	Openness
6	speaker_15_center_83.mov	0.566	0.544	0.493	0.587	0.499	Practical and easy to use	Economical/low cost	Family friendly	Agreeableness	Openness	Non-Neuroticism
7	speaker_19_center_83.mov	0.506	0.438	0.431	0.456	0.441	Classic car features	Fashion and attention	Luxury additions	Agreeableness	Conscientiousness	Openness
8	speaker_23_center_83.mov	0.486	0.522	0.310	0.432	0.434	Classic car features	Fashion and attention	Luxury additions	Agreeableness	Conscientiousness	Openness
9	speaker_24_center_83.mov	0.417	0.473	0.321	0.445	0.415	Classic car features	Fashion and attention	Luxury additions	Agreeableness	Conscientiousness	Openness
10	speaker_27_center_83.mov	0.526	0.661	0.443	0.559	0.554	Safe and reliable	Practical and easy to use	Economical/low cost	Conscientiousness	Agreeableness	Non-Neuroticism

(continues on next page)

(продолжение с предыдущей страницы)

```
7                   Openness
8                   Openness
9                   Openness
10          Non-Neuroticism
```

Прогнозирование потребительских предпочтений на характеристики мобильного устройства

В качестве примера предлагается использование коэффициентов корреляции между персональными качествами человека и характеристиками мобильного устройства, представленными в статье:

1) Peltonen E., Sharmila P., Asare K. O., Visuri A., Lagerspetz E., Ferreira D. (2020). When phones get personal: Predicting Big Five personality traits from application usage // Pervasive and Mobile Computing. – 2020. – Vol. 69. – 101269.

[15]:
```
# Загрузка датафрейма с коэффициентами корреляции
url = 'https://download.sberdisk.ru/download/file/478676690?token=7KcAxPqMpWiYQnx&
↪filename=divice_characteristics.csv'
df_divice_characteristics = pd.read_csv(url)

df_divice_characteristics.index.name = 'ID'
df_divice_characteristics.index += 1
df_divice_characteristics.index = df_divice_characteristics.index.map(str)

df_divice_characteristics
```

[15]:
```
                Trait  Communication  Game Action  Game Board  Game Casino  \
ID
1            Openness          0.118        0.056       0.079        0.342
2   Conscientiousness          0.119        0.043       0.107        0.448
3        Extraversion          0.246        0.182       0.211        0.311
4       Agreeableness          0.218        0.104       0.164        0.284
5     Non-Neuroticism          0.046        0.047       0.125        0.515

    Game Educational  Game Simulation  Game Trivia  Entertainment  Finance  \
ID
1              0.027            0.104        0.026          0.000    0.006
2              0.039            0.012        0.119          0.000    0.005
3              0.102            0.165        0.223          0.001    0.003
4              0.165            0.122        0.162          0.000    0.003
5              0.272            0.179        0.214          0.002    0.030

    Health and Fitness  Media and Video  Music and Audio  News and Magazines  \
ID
1                0.002            0.000            0.000               0.001
2                0.001            0.000            0.002               0.002
3                0.000            0.001            0.001               0.001
4                0.001            0.000            0.002               0.002
5                0.001            0.000            0.005               0.003

    Personalisation  Travel and Local  Weather
ID
```

(continues on next page)

(продолжение с предыдущей страницы)

```
1              0.004              0.002    0.004
2              0.001              0.001    0.003
3              0.004              0.009    0.003
4              0.001              0.004    0.003
5              0.008              0.004    0.007
```

[16]:
```
_b5._priority_calculation(
    correlation_coefficients = df_divice_characteristics,
    col_name_ocean = 'Trait',
    threshold = 0.55,
    number_priority = 3,
    number_importance_traits = 3,
    out = True
)

_b5._save_logs(df = _b5.df_files_priority_, name = 'divice_characteristics_priorities_
↪mupta_en', out = True)

# Опционно
df = _b5.df_files_priority_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON
↪', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = ['OPE', 'CON', 'EXT', 'AGR', 'NNEU']
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[16]:
```
                              Path    OPE    CON    EXT    AGR   NNEU  \
Person ID
1          speaker_01_center_83.mov  0.565  0.539  0.441  0.593  0.489
2          speaker_06_center_83.mov  0.651  0.664  0.607  0.644  0.621
3          speaker_07_center_83.mov  0.436  0.487  0.314  0.415  0.397
4          speaker_10_center_83.mov  0.499  0.511  0.413  0.469  0.444
5          speaker_11_center_83.mov  0.395  0.342  0.327  0.427  0.355
6          speaker_15_center_83.mov  0.566  0.544  0.493  0.587  0.499
7          speaker_19_center_83.mov  0.506  0.438  0.431  0.456  0.441
8          speaker_23_center_83.mov  0.486  0.522  0.310  0.432  0.434
9          speaker_24_center_83.mov  0.417  0.473  0.321  0.445  0.415
10         speaker_27_center_83.mov  0.526  0.661  0.443  0.559  0.554

                    Priority 1          Priority 2          Priority 3  \
Person ID
1                Communication  Health and Fitness     Media and Video
2                  Game Casino       Communication         Game Trivia
3              Media and Video       Entertainment  Health and Fitness
4              Media and Video       Entertainment  Health and Fitness
5              Media and Video       Entertainment  Health and Fitness
6           Health and Fitness     Media and Video  News and Magazines
7              Media and Video       Entertainment  Health and Fitness
8              Media and Video       Entertainment  Health and Fitness
9              Media and Video       Entertainment  Health and Fitness
10                 Game Casino    Game Educational         Game Trivia

           Trait importance 1 Trait importance 2 Trait importance 3
```

(continues on next page)

(продолжение с предыдущей страницы)

```
Person ID
1               Agreeableness           Openness    Non-Neuroticism
2            Non-Neuroticism       Extraversion  Conscientiousness
3               Agreeableness  Conscientiousness       Extraversion
4               Agreeableness  Conscientiousness       Extraversion
5           Conscientiousness      Agreeableness       Extraversion
6               Agreeableness           Openness       Extraversion
7           Conscientiousness      Agreeableness       Extraversion
8               Agreeableness  Conscientiousness       Extraversion
9               Agreeableness  Conscientiousness       Extraversion
10            Non-Neuroticism  Conscientiousness      Agreeableness
```

Решение практической задачи 3

Задача: Формирование эффективных рабочих коллективов

Решение практической задачи выполняется в два этапа. На первом этапе необходимо использовать библиотеку OCEAN-AI для получения гипотез предсказаний (оценок персональных качеств личности человека). На втором этапе следует использовать метод _colleague_ranking из библиотеки OCEAN-AI для решения представленной практической задачи на примере поиска подходящих коллег для целевого коллеги. Примеры результатов работы и реализации представлены ниже.

Таким образом, библиотека OCEAN-AI предоставляет инструменты для анализа персональных качеств личности коллег и может помочь в формировании эффективных рабочих групп, улучшении коммуникации и сокращении конфликтов в коллективе.

FI V2

[2]:
```python
# Импорт необходимых инструментов
import os
import pandas as pd

# Импорт модуля
from oceanai.modules.lab.build import Run

# Создание экземпляра класса
_b5 = Run()

# Настройка ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000      # Размер загрузки файла из сети за 1 шаг

corpus = 'fi'

# Формирование аудиомоделей
```

(continues on next page)

(продолжение с предыдущей страницы)

```
res_load_model_hc = _b5.load_audio_model_hc()
res_load_model_nn = _b5.load_audio_model_nn()

# Загрузка весов аудиомоделей
url = _b5.weights_for_big5_['audio'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_audio_model_weights_hc(url = url)

url = _b5.weights_for_big5_['audio'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_audio_model_weights_nn(url = url)

# Формирование видеомоделей
res_load_model_hc = _b5.load_video_model_hc(lang='en')
res_load_model_deep_fe = _b5.load_video_model_deep_fe()
res_load_model_nn = _b5.load_video_model_nn()

# Загрузка весов видеомоделей
url = _b5.weights_for_big5_['video'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_video_model_weights_hc(url = url)

url = _b5.weights_for_big5_['video'][corpus]['fe']['sberdisk']
res_load_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(url = url)

url = _b5.weights_for_big5_['video'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_video_model_weights_nn(url = url)

# Загрузка словаря с экспертными признаками (текстовая модальность)
res_load_text_features = _b5.load_text_features()

# Формирование текстовых моделей
res_setup_translation_model = _b5.setup_translation_model() # только для русского языка
res_setup_translation_model = _b5.setup_bert_encoder(force_reload = False)
res_load_text_model_hc_fi = _b5.load_text_model_hc(corpus=corpus)
res_load_text_model_nn_fi = _b5.load_text_model_nn(corpus=corpus)

# Загрузка весов текстовых моделей
url = _b5.weights_for_big5_['text'][corpus]['hc']['sberdisk']
res_load_text_model_weights_hc_fi = _b5.load_text_model_weights_hc(url = url)

url = _b5.weights_for_big5_['text'][corpus]['nn']['sberdisk']
res_load_text_model_weights_nn_fi = _b5.load_text_model_weights_nn(url = url)

# Формирование модели для мультимодального объединения информации
res_load_avt_model_b5 = _b5.load_avt_model_b5()

# Загрузка весов модели для мультимодального объединения информации
url = _b5.weights_for_big5_['avt'][corpus]['b5']['sberdisk']
res_load_avt_model_weights_b5 = _b5.load_avt_model_weights_b5(url = url)

PATH_TO_DIR = './video_FI/'
PATH_SAVE_VIDEO = './video_FI/test/'

_b5.path_to_save_ = PATH_SAVE_VIDEO
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Загрузка 10 тестовых аудиовидеозаписей из корпуса First Impression V2
# URL: https://chalearnlap.cvc.uab.cat/dataset/24/description/
domain = 'https://download.sberdisk.ru/download/file/'
tets_name_files = [
    '429713680?token=FqHdMLSSh7zYSZt&filename=_plk5k7PBEg.003.mp4',
    '429713681?token=Hz9b4lQkrLfic33&filename=be0DQawtVkE.002.mp4',
    '429713683?token=EgUXS9Xs8xHm5gz&filename=2d6btbaNdfo.000.mp4',
    '429713684?token=1U26753kmPYdIgt&filename=300gK3CnzW0.003.mp4',
    '429713685?token=LyigAWLTzDNwKJO&filename=300gK3CnzW0.001.mp4',
    '429713686?token=EpfRbCKHyuc4HPu&filename=cLaZxEf1nE4.004.mp4',
    '429713687?token=FNTkwqBr4jOS95l&filename=g24JGYuT74A.004.mp4',
    '429713688?token=qDT95nz7hfm2Nki&filename=JZNMxa3OKHY.000.mp4',
    '429713689?token=noLguEGXDpbcKhg&filename=nvlqJbHk_Lc.003.mp4',
    '429713679?token=9L7RQ0hgdJlcek6&filename=4vdJGgZpj4k.003.mp4'
]

for curr_files in tets_name_files:
    _b5.download_file_from_url(url = domain + curr_files, out = True)

# Получение прогнозов
_b5.path_to_dataset_ = PATH_TO_DIR # Директория набора данных
_b5.ext_ = ['.mp4'] # Расширения искомых файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_[corpus]['sberdisk']

_b5.get_avt_predictions(url_accuracy = url_accuracy, lang = 'en')
```

[2023-12-16 19:24:17] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-16 19:24:19] Получение прогнозов и вычисление точности (мультимодальное объединение) ...

10 из 10 (100.0%) ... GitHub:nbsphinx-math:*OCEANAI*_guide:nbsphinx-math:*notebooks*_FI:nbsphinx-math:*test*_plk5k7PBEg.003.mp4
...

Person ID	Path	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism
1	2d6btbaNdfo.000.mp4	0.581159	0.628822	0.466609	0.622129	0.553832
2	300gK3CnzW0.001.mp4	0.463991	0.418851	0.41301		
3	300gK3CnzW0.003.mp4	0.454281	0.415049	0.39189		
4	4vdJGgZpj4k.003.mp4	0.588461	0.643233	0.530789		
5	be0DQawtVkE.002.mp4	0.633433	0.533295	0.523742		
6	cLaZxEf1nE4.004.mp4	0.636944	0.542386	0.558461		
7	g24JGYuT74A.004.mp4	0.531518	0.376987	0.393309		
8	JZNMxa3OKHY.000.mp4	0.610342	0.541418	0.563163		
9	nvlqJbHk_Lc.003.mp4	0.495809	0.458526	0.414436		
10	_plk5k7PBEg.003.mp4	0.60707	0.591893	0.520662		

(continues on next page)

(продолжение с предыдущей страницы)

```
2              0.493329        0.423093
3              0.485114        0.420741
4              0.603038        0.593398
5              0.608591        0.588456
6              0.570975        0.558983
7                0.4904        0.447881
8              0.595013        0.569461
9              0.469152        0.435461
10             0.603938        0.565726
```

[2023-12-16 19:24:19] Точность по отдельным персональным качествам личности человека ...

```
          Openness  Conscientiousness  Extraversion  Agreeableness  \
Metrics
MAE         0.0589             0.0612        0.0864         0.0697
Accuracy    0.9411             0.9388        0.9136         0.9303

          Non-Neuroticism    Mean
Metrics
MAE                0.0582  0.0669
Accuracy           0.9418  0.9331
```

[2023-12-16 19:24:19] Средняя средних абсолютных ошибок: 0.0669, средняя точность: 0.9331 ...

Лог файлы успешно сохранены ...

— Время выполнения: 67.109 сек. —

[2]:
```
True
```

Для поиска подходящего коллеги по работе необходимо знать по два коэффициента корреляции для каждого персонального качества личности человека. Эти коэффициенты должны показывать, как изменится оценка качества одного человека, если она будет больше или меньше оценки качества другого человека.

В качестве примера предлагается использование коэффициентов корреляции между двумя людьми в контексте отношений “начальник-подчиненный”, представленных в статье:

1) Kuroda S., Yamamoto I. Good boss, bad boss, workers' mental health and productivity: Evidence from Japan // Japan & The World Economy. – 2018. – vol. 48. – pp. 106-118.

Пользователь может установить свои коэффициенты корреляции

[3]:
```
# Загрузка датафрейма с коэффициентами корреляции
url = 'https://download.sberdisk.ru/download/file/478675819?token=LuB7L1QsEY0UuSs&
↪filename=colleague_ranking.csv'
df_correlation_coefficients = pd.read_csv(url)
df_correlation_coefficients = pd.DataFrame(
    df_correlation_coefficients.drop(['ID'], axis = 1)
)

df_correlation_coefficients.index.name = 'ID'
df_correlation_coefficients.index += 1
df_correlation_coefficients.index = df_correlation_coefficients.index.map(str)

df_correlation_coefficients
```

[3]:
```
   Score_comparison  Openness  Conscientiousness  Extraversion  Agreeableness  \
ID
1            higher   -0.0602             0.0471       -0.1070        -0.0832
2             lower   -0.1720            -0.1050        0.0772         0.0703

    Non-Neuroticism
ID
1             0.190
2            -0.229
```

Поиск старшего коллеги

[4]:
```
# Список оценок персональных качеств личности целевого человека
target_scores = [0.527886, 0.522337, 0.458468, 0.51761, 0.444649]

_b5._colleague_ranking(
    correlation_coefficients = df_correlation_coefficients,
    target_scores = target_scores,
    colleague = 'major',
    equal_coefficients = 0.5,
    out = False
)

_b5._save_logs(df = _b5.df_files_colleague_, name = 'major_colleague_ranking_fi_en', out␣
↪= True)

# Опционно
df = _b5.df_files_colleague_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON
↪', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[4]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU	Match
7	g24JGYuT74A.004.mp4	0.532	0.377	0.393	0.490	0.448	0.078
4	4vdJGgZpj4k.003.mp4	0.588	0.643	0.531	0.603	0.593	0.001
1	2d6btbaNdfo.000.mp4	0.581	0.629	0.467	0.622	0.554	-0.002
10	_plk5k7PBEg.003.mp4	0.607	0.592	0.521	0.604	0.566	-0.007
5	be0DQawtVkE.002.mp4	0.633	0.533	0.524	0.609	0.588	-0.008
8	JZNMxa3OKHY.000.mp4	0.610	0.541	0.563	0.595	0.569	-0.013
6	cLaZxEf1nE4.004.mp4	0.637	0.542	0.558	0.571	0.559	-0.014
3	300gK3CnzW0.003.mp4	0.454	0.415	0.392	0.485	0.421	-0.154
2	300gK3CnzW0.001.mp4	0.464	0.419	0.413	0.493	0.423	-0.154
9	nvlqJbHk_Lc.003.mp4	0.496	0.459	0.414	0.469	0.435	-0.168

Поиск младшего коллеги

[5]:
```
# Список оценок персональных качеств личности целевого человека
target_scores = [0.527886, 0.522337, 0.458468, 0.51761, 0.444649]

_b5._colleague_ranking(
    correlation_coefficients = df_correlation_coefficients,
    target_scores = target_scores,
    colleague = 'minor',
    equal_coefficients = 0.5,
    out = False
)

_b5._save_logs(df = _b5.df_files_colleague_, name = 'minor_colleague_ranking_fi_en', out␣
↪= True)

# Опционно
df = _b5.df_files_colleague_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON
↪', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[5]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU	Match
9	nvlqJbHk_Lc.003.mp4	0.496	0.459	0.414	0.469	0.435	-0.009
3	300gK3CnzW0.003.mp4	0.454	0.415	0.392	0.485	0.421	-0.010
2	300gK3CnzW0.001.mp4	0.464	0.419	0.413	0.493	0.423	-0.013
8	JZNMxa3OKHY.000.mp4	0.610	0.541	0.563	0.595	0.569	-0.207
6	cLaZxEf1nE4.004.mp4	0.637	0.542	0.558	0.571	0.559	-0.211
1	2d6btbaNdfo.000.mp4	0.581	0.629	0.467	0.622	0.554	-0.213
10	_plk5k7PBEg.003.mp4	0.607	0.592	0.521	0.604	0.566	-0.213
5	be0DQawtVkE.002.mp4	0.633	0.533	0.524	0.609	0.588	-0.216
4	4vdJGgZpj4k.003.mp4	0.588	0.643	0.531	0.603	0.593	-0.221
7	g24JGYuT74A.004.mp4	0.532	0.377	0.393	0.490	0.448	-0.259

MuPTA (ru)

[6]:
```
import os
import pandas as pd

# Импорт модуля
from oceanai.modules.lab.build import Run

# Создание экземпляра класса
_b5 = Run()

corpus = 'mupta'
lang = 'ru'

# Настройка ядра
```

(continues on next page)

(продолжение с предыдущей страницы)

```
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000      # Размер загрузки файла из сети за 1 шаг

# Формирование аудиомоделей
res_load_model_hc = _b5.load_audio_model_hc()
res_load_model_nn = _b5.load_audio_model_nn()

# Загрузка весов аудиомоделей
url = _b5.weights_for_big5_['audio'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_audio_model_weights_hc(url = url)

url = _b5.weights_for_big5_['audio'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_audio_model_weights_nn(url = url)

# Формирование видеомоделей
res_load_model_hc = _b5.load_video_model_hc(lang=lang)
res_load_model_deep_fe = _b5.load_video_model_deep_fe()
res_load_model_nn = _b5.load_video_model_nn()

# Загрузка весов видеомоделей
url = _b5.weights_for_big5_['video'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_video_model_weights_hc(url = url)

url = _b5.weights_for_big5_['video'][corpus]['fe']['sberdisk']
res_load_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(url = url)

url = _b5.weights_for_big5_['video'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_video_model_weights_nn(url = url)

# Загрузка словаря с экспертными признаками (текстовая модальность)
res_load_text_features = _b5.load_text_features()

# Формирование текстовых моделей
res_setup_translation_model = _b5.setup_translation_model() # только для русского языка
res_setup_translation_model = _b5.setup_bert_encoder(force_reload = False)
res_load_text_model_hc_fi = _b5.load_text_model_hc(corpus=corpus)
res_load_text_model_nn_fi = _b5.load_text_model_nn(corpus=corpus)

# Загрузка весов текстовых моделей
url = _b5.weights_for_big5_['text'][corpus]['hc']['sberdisk']
res_load_text_model_weights_hc_fi = _b5.load_text_model_weights_hc(url = url)

url = _b5.weights_for_big5_['text'][corpus]['nn']['sberdisk']
res_load_text_model_weights_nn_fi = _b5.load_text_model_weights_nn(url = url)

# Формирование модели для мультимодального объединения информации
res_load_avt_model_b5 = _b5.load_avt_model_b5()

# Загрузка весов модели для мультимодального объединения информации
url = _b5.weights_for_big5_['avt'][corpus]['b5']['sberdisk']
res_load_avt_model_weights_b5 = _b5.load_avt_model_weights_b5(url = url)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
PATH_TO_DIR = './video_MuPTA/'
PATH_SAVE_VIDEO = './video_MuPTA/test/'

_b5.path_to_save_ = PATH_SAVE_VIDEO

# Загрузка 10 тестовых аудиовидеозаписей из корпуса MuPTA
# URL: https://hci.nw.ru/en/pages/mupta-corpus
domain = 'https://download.sberdisk.ru/download/file/'
tets_name_files = [
    '477995979?token=2cvyk7CS0mHx2MJ&filename=speaker_06_center_83.mov',
    '477995980?token=jGPtBPS69uzFU6Y&filename=speaker_01_center_83.mov',
    '477995967?token=zCaRbNB6ht5wMPq&filename=speaker_11_center_83.mov',
    '477995966?token=B1rbinDYRQKrI3T&filename=speaker_15_center_83.mov',
    '477995978?token=dEpVDtZg1EQiEQ9&filename=speaker_07_center_83.mov',
    '477995961?token=o1hVjw8G45q9L9Z&filename=speaker_19_center_83.mov',
    '477995964?token=5K220Aqf673VHPq&filename=speaker_23_center_83.mov',
    '477995965?token=v1LVD2KT1cU7Lpb&filename=speaker_24_center_83.mov',
    '477995962?token=tmaSGyyWLA6XCy9&filename=speaker_27_center_83.mov',
    '477995963?token=bTpo96qNDPcwGqb&filename=speaker_10_center_83.mov',
]

for curr_files in tets_name_files:
    _b5.download_file_from_url(url = domain + curr_files, out = True)

# Получение прогнозов
_b5.path_to_dataset_ = PATH_TO_DIR # Директория набора данных
_b5.ext_ = ['.mov'] # Расширения искомых файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_['mupta']['sberdisk']

_b5.get_avt_predictions(url_accuracy = url_accuracy, lang = lang)
```

[2023-12-16 19:32:56] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-16 19:33:00] Получение прогнозов и вычисление точности (мультимодальное объединение) ...

10 из 10 (100.0%) ... GitHub:nbsphinx-math:*OCEANAI*_guide:nbsphinx-math:*notebooks*_MuPTA:nbsphinx-math:*test*_27_center_83.mov
...

Person ID	Path	Openness	Conscientiousness	Extraversion	\
1	speaker_01_center_83.mov	0.758137	0.693356	0.650108	
2	speaker_06_center_83.mov	0.681602	0.654339	0.607156	
3	speaker_07_center_83.mov	0.666104	0.656836	0.567863	
4	speaker_10_center_83.mov	0.694171	0.596195	0.571414	
5	speaker_11_center_83.mov	0.712885	0.594764	0.571709	
6	speaker_15_center_83.mov	0.664158	0.670411	0.60421	
7	speaker_19_center_83.mov	0.761213	0.652635	0.651028	
8	speaker_23_center_83.mov	0.692788	0.68324	0.616737	
9	speaker_24_center_83.mov	0.705923	0.658382	0.610645	
10	speaker_27_center_83.mov	0.753417	0.708372	0.654608	

(continues on next page)

(продолжение с предыдущей страницы)

```
          Agreeableness Non-Neuroticism
Person ID
1              0.744589        0.488671
2              0.731282        0.417908
3              0.685067        0.378102
4               0.66223        0.348639
5              0.716696         0.37802
6              0.696056        0.399842
7              0.788677        0.459676
8              0.795205        0.447242
9              0.697415        0.411988
10             0.816416        0.504743
```

[2023-12-16 19:33:00] Точность по отдельным персональным качествам личности человека ...

```
          Openness  Conscientiousness  Extraversion  Agreeableness  \
Metrics
MAE         0.0673             0.0789        0.1325          0.102
Accuracy    0.9327             0.9211        0.8675          0.898

          Non-Neuroticism    Mean
Metrics
MAE                0.1002  0.0962
Accuracy           0.8998  0.9038
```

[2023-12-16 19:33:00] Средняя средних абсолютных ошибок: 0.0962, средняя точность: 0.9038 ...

Лог файлы успешно сохранены ...

— Время выполнения: 444.191 сек. —

[6]:
```
True
```

Для поиска подходящего коллеги по работе необходимо знать по два коэффициента корреляции для каждого персонального качества личности человека. Эти коэффициенты должны показывать, как изменится оценка качества одного человека, если она будет больше или меньше оценки качества другого человека.

В качестве примера предлагается использование коэффициентов корреляции между двумя людьми в контексте отношений “начальник-подчиненный”, представленных в статье:

1) Kuroda S., Yamamoto I. Good boss, bad boss, workers' mental health and productivity: Evidence from Japan // Japan & The World Economy. – 2018. – vol. 48. – pp. 106-118.

Пользователь может установить свои коэффициенты корреляции

[7]:
```
# Загрузка датафрейма с коэффициентами корреляции
url = 'https://download.sberdisk.ru/download/file/478675819?token=LuB7L1QsEYOUuSs&
↪filename=colleague_ranking.csv'
df_correlation_coefficients = pd.read_csv(url)
df_correlation_coefficients = pd.DataFrame(
    df_correlation_coefficients.drop(['ID'], axis = 1)
)

df_correlation_coefficients.index.name = 'ID'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
df_correlation_coefficients.index += 1
df_correlation_coefficients.index = df_correlation_coefficients.index.map(str)

df_correlation_coefficients
```

[7]:

ID	Score_comparison	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism
1	higher	-0.0602	0.0471	-0.1070	-0.0832	0.190
2	lower	-0.1720	-0.1050	0.0772	0.0703	-0.229

Поиск старшего коллеги

[8]:
```
# Список оценок персональных качеств личности целевого человека
target_scores = [0.527886, 0.522337, 0.458468, 0.51761, 0.444649]

_b5._colleague_ranking(
    correlation_coefficients = df_correlation_coefficients,
    target_scores = target_scores,
    colleague = 'major',
    equal_coefficients = 0.5,
    out = False
)

_b5._save_logs(df = _b5.df_files_colleague_, name = 'major_colleague_ranking_mupta_ru',␣
↪out = True)

# Опционно
df = _b5.df_files_colleague_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON
↪', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[8]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU	Match
1	speaker_01_center_83.mov	0.758	0.693	0.650	0.745	0.489	-0.052
10	speaker_27_center_83.mov	0.753	0.708	0.655	0.816	0.505	-0.054
8	speaker_23_center_83.mov	0.693	0.683	0.617	0.795	0.447	-0.057
7	speaker_19_center_83.mov	0.761	0.653	0.651	0.789	0.460	-0.063
4	speaker_10_center_83.mov	0.694	0.596	0.571	0.662	0.349	-0.210
3	speaker_07_center_83.mov	0.666	0.657	0.568	0.685	0.378	-0.214
5	speaker_11_center_83.mov	0.713	0.595	0.572	0.717	0.378	-0.222
6	speaker_15_center_83.mov	0.664	0.670	0.604	0.696	0.400	-0.223
9	speaker_24_center_83.mov	0.706	0.658	0.611	0.697	0.412	-0.229
2	speaker_06_center_83.mov	0.682	0.654	0.607	0.731	0.418	-0.232

Поиск младшего коллеги

[9]:
```
# Список оценок персональных качеств личности целевого человека
target_scores = [0.527886, 0.522337, 0.458468, 0.51761, 0.444649]

_b5._colleague_ranking(
    correlation_coefficients = df_correlation_coefficients,
    target_scores = target_scores,
    colleague = 'minor',
    equal_coefficients = 0.5,
    out = False
)

_b5._save_logs(df = _b5.df_files_colleague_, name = 'minor_colleague_ranking_mupta_ru',␣
↪out = True)

# Опционно
df = _b5.df_files_colleague_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON
↪', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[9]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU	Match
2	speaker_06_center_83.mov	0.682	0.654	0.607	0.731	0.418	-0.008
6	speaker_15_center_83.mov	0.664	0.670	0.604	0.696	0.400	-0.013
9	speaker_24_center_83.mov	0.706	0.658	0.611	0.697	0.412	-0.016
5	speaker_11_center_83.mov	0.713	0.595	0.572	0.717	0.378	-0.019
3	speaker_07_center_83.mov	0.666	0.657	0.568	0.685	0.378	-0.020
4	speaker_10_center_83.mov	0.694	0.596	0.571	0.662	0.349	-0.025
8	speaker_23_center_83.mov	0.693	0.683	0.617	0.795	0.447	-0.190
7	speaker_19_center_83.mov	0.761	0.653	0.651	0.789	0.460	-0.199
10	speaker_27_center_83.mov	0.753	0.708	0.655	0.816	0.505	-0.212
1	speaker_01_center_83.mov	0.758	0.693	0.650	0.745	0.489	-0.213

MuPTA (en)

[10]:
```
import os
import pandas as pd

# Импорт модуля
from oceanai.modules.lab.build import Run

# Создание экземпляра класса
_b5 = Run()

corpus = 'fi'
lang = 'en'

# Настройка ядра
```

(continues on next page)

(продолжение с предыдущей страницы)

```
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000      # Размер загрузки файла из сети за 1 шаг

# Формирование аудиомоделей
res_load_model_hc = _b5.load_audio_model_hc()
res_load_model_nn = _b5.load_audio_model_nn()

# Загрузка весов аудиомоделей
url = _b5.weights_for_big5_['audio'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_audio_model_weights_hc(url = url)

url = _b5.weights_for_big5_['audio'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_audio_model_weights_nn(url = url)

# Формирование видеомоделей
res_load_model_hc = _b5.load_video_model_hc(lang=lang)
res_load_model_deep_fe = _b5.load_video_model_deep_fe()
res_load_model_nn = _b5.load_video_model_nn()

# Загрузка весов видеомоделей
url = _b5.weights_for_big5_['video'][corpus]['hc']['sberdisk']
res_load_model_weights_hc = _b5.load_video_model_weights_hc(url = url)

url = _b5.weights_for_big5_['video'][corpus]['fe']['sberdisk']
res_load_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(url = url)

url = _b5.weights_for_big5_['video'][corpus]['nn']['sberdisk']
res_load_model_weights_nn = _b5.load_video_model_weights_nn(url = url)

# Загрузка словаря с экспертными признаками (текстовая модальность)
res_load_text_features = _b5.load_text_features()

# Формирование текстовых моделей
res_setup_translation_model = _b5.setup_translation_model() # только для русского языка
res_setup_translation_model = _b5.setup_bert_encoder(force_reload = False)
res_load_text_model_hc_fi = _b5.load_text_model_hc(corpus=corpus)
res_load_text_model_nn_fi = _b5.load_text_model_nn(corpus=corpus)

# Загрузка весов текстовых моделей
url = _b5.weights_for_big5_['text'][corpus]['hc']['sberdisk']
res_load_text_model_weights_hc_fi = _b5.load_text_model_weights_hc(url = url)

url = _b5.weights_for_big5_['text'][corpus]['nn']['sberdisk']
res_load_text_model_weights_nn_fi = _b5.load_text_model_weights_nn(url = url)

# Формирование модели для мультимодального объединения информации
res_load_avt_model_b5 = _b5.load_avt_model_b5()

# Загрузка весов модели для мультимодального объединения информации
url = _b5.weights_for_big5_['avt'][corpus]['b5']['sberdisk']
res_load_avt_model_weights_b5 = _b5.load_avt_model_weights_b5(url = url)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
PATH_TO_DIR = './video_MuPTA/'
PATH_SAVE_VIDEO = './video_MuPTA/test/'

_b5.path_to_save_ = PATH_SAVE_VIDEO

# Загрузка 10 тестовых аудиовидеозаписей из корпуса MuPTA
# URL: https://hci.nw.ru/en/pages/mupta-corpus
domain = 'https://download.sberdisk.ru/download/file/'
tets_name_files = [
    '477995979?token=2cvyk7CS0mHx2MJ&filename=speaker_06_center_83.mov',
    '477995980?token=jGPtBPS69uzFU6Y&filename=speaker_01_center_83.mov',
    '477995967?token=zCaRbNB6ht5wMPq&filename=speaker_11_center_83.mov',
    '477995966?token=B1rbinDYRQKrI3T&filename=speaker_15_center_83.mov',
    '477995978?token=dEpVDtZg1EQiEQ9&filename=speaker_07_center_83.mov',
    '477995961?token=o1hVjw8G45q9L9Z&filename=speaker_19_center_83.mov',
    '477995964?token=5K220Aqf673VHPq&filename=speaker_23_center_83.mov',
    '477995965?token=v1LVD2KT1cU7Lpb&filename=speaker_24_center_83.mov',
    '477995962?token=tmaSGyyWLA6XCy9&filename=speaker_27_center_83.mov',
    '477995963?token=bTpo96qNDPcwGqb&filename=speaker_10_center_83.mov',
]

for curr_files in tets_name_files:
    _b5.download_file_from_url(url = domain + curr_files, out = True)

# Получение прогнозов
_b5.path_to_dataset_ = PATH_TO_DIR # Директория набора данных
_b5.ext_ = ['.mov'] # Расширения искомых файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_['mupta']['sberdisk']

_b5.get_avt_predictions(url_accuracy = url_accuracy, lang = lang)
```

[2023-12-16 19:40:25] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-16 19:40:28] Получение прогнозов и вычисление точности (мультимодальное объединение) ...

10 из 10 (100.0%) ... GitHub:nbsphinx-math:*OCEANAI*_guide:nbsphinx-math:*notebooks*_MuPTA:nbsphinx-math:*test*_27_center_83.mov
...

Person ID	Path	Openness	Conscientiousness	Extraversion
1	speaker_01_center_83.mov	0.564985	0.539052	0.440615
2	speaker_06_center_83.mov	0.650774	0.663849	0.607308
3	speaker_07_center_83.mov	0.435976	0.486683	0.313828
4	speaker_10_center_83.mov	0.498542	0.511243	0.412592
5	speaker_11_center_83.mov	0.394776	0.341608	0.327082
6	speaker_15_center_83.mov	0.566107	0.543811	0.492766
7	speaker_19_center_83.mov	0.506271	0.438215	0.430894
8	speaker_23_center_83.mov	0.486463	0.521755	0.309894
9	speaker_24_center_83.mov	0.417404	0.473339	0.320714
10	speaker_27_center_83.mov	0.526112	0.661107	0.443167

(continues on next page)

(продолжение с предыдущей страницы)

```
          Agreeableness Non-Neuroticism
Person ID
1               0.59251        0.488763
2              0.643847        0.620627
3              0.415446        0.396618
4              0.468947         0.44399
5              0.427304        0.354936
6              0.587411        0.499433
7              0.456177         0.44075
8              0.432291        0.433601
9              0.445086        0.414649
10             0.558965        0.554224
```

[2023-12-16 19:40:28] Точность по отдельным персональным качествам личности человека ...

```
          Openness  Conscientiousness  Extraversion  Agreeableness  \
Metrics
MAE         0.1727             0.1672        0.1661         0.2579
Accuracy    0.8273             0.8328        0.8339         0.7421

          Non-Neuroticism    Mean
Metrics
MAE                 0.107  0.1742
Accuracy            0.893  0.8258
```

[2023-12-16 19:40:28] Средняя средних абсолютных ошибок: 0.1742, средняя точность: 0.8258 ...

Лог файлы успешно сохранены ...

— Время выполнения: 377.119 сек. —

[10]:
```
True
```

Для поиска подходящего коллеги по работе необходимо знать по два коэффициента корреляции для каждого персонального качества личности человека. Эти коэффициенты должны показывать, как изменится оценка качества одного человека, если она будет больше или меньше оценки качества другого человека.

В качестве примера предлагается использование коэффициентов корреляции между двумя людьми в контексте отношений “начальник-подчиненный”, представленных в статье:

1) Kuroda S., Yamamoto I. Good boss, bad boss, workers' mental health and productivity: Evidence from Japan // Japan & The World Economy. – 2018. – vol. 48. – pp. 106-118.

Пользователь может установить свои коэффициенты корреляции

[11]:
```
# Загрузка датафрейма с коэффициентами корреляции
url = 'https://download.sberdisk.ru/download/file/478675819?token=LuB7L1QsEYOUuSs&
↪filename=colleague_ranking.csv'
df_correlation_coefficients = pd.read_csv(url)
df_correlation_coefficients = pd.DataFrame(
    df_correlation_coefficients.drop(['ID'], axis = 1)
)

df_correlation_coefficients.index.name = 'ID'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
df_correlation_coefficients.index += 1
df_correlation_coefficients.index = df_correlation_coefficients.index.map(str)

df_correlation_coefficients
```

[11]:

ID	Score_comparison	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism
1	higher	-0.0602	0.0471	-0.1070	-0.0832	0.190
2	lower	-0.1720	-0.1050	0.0772	0.0703	-0.229

Поиск старшего коллеги

[12]:

```
# Список оценок персональных качеств личности целевого человека
target_scores = [0.527886, 0.522337, 0.458468, 0.51761, 0.444649]

_b5._colleague_ranking(
    correlation_coefficients = df_correlation_coefficients,
    target_scores = target_scores,
    colleague = 'major',
    equal_coefficients = 0.5,
    out = False
)

_b5._save_logs(df = _b5.df_files_colleague_, name = 'major_colleague_ranking_mupta_en',␣
↪out = True)

# Опционно
df = _b5.df_files_colleague_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON
↪', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[12]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU	Match
1	speaker_01_center_83.mov	0.565	0.539	0.441	0.593	0.489	0.069
10	speaker_27_center_83.mov	0.526	0.661	0.443	0.559	0.554	0.034
2	speaker_06_center_83.mov	0.651	0.664	0.607	0.644	0.621	-0.009
6	speaker_15_center_83.mov	0.566	0.544	0.493	0.587	0.499	-0.015
5	speaker_11_center_83.mov	0.395	0.342	0.327	0.427	0.355	-0.130
9	speaker_24_center_83.mov	0.417	0.473	0.321	0.445	0.415	-0.160
3	speaker_07_center_83.mov	0.436	0.487	0.314	0.415	0.397	-0.163
7	speaker_19_center_83.mov	0.506	0.438	0.431	0.456	0.441	-0.169
4	speaker_10_center_83.mov	0.499	0.511	0.413	0.469	0.444	-0.176
8	speaker_23_center_83.mov	0.486	0.522	0.310	0.432	0.434	-0.183

Поиск младшего коллеги

[13]:
```
# Список оценок персональных качеств личности целевого человека
target_scores = [0.527886, 0.522337, 0.458468, 0.51761, 0.444649]

_b5._colleague_ranking(
    correlation_coefficients = df_correlation_coefficients,
    target_scores = target_scores,
    colleague = 'minor',
    equal_coefficients = 0.5,
    out = False
)

_b5._save_logs(df = _b5.df_files_colleague_, name = 'minor_colleague_ranking_mupta_en',␣
↪out = True)

# Опционно
df = _b5.df_files_colleague_.rename(columns = {'Openness':'OPE', 'Conscientiousness':'CON
↪', 'Extraversion': 'EXT', 'Agreeableness': 'AGR', 'Non-Neuroticism': 'NNEU'})
columns_to_round = df.columns[1:]
df[columns_to_round] = df[columns_to_round].apply(lambda x: [round(i, 3) for i in x])
df
```

[13]:

Person ID	Path	OPE	CON	EXT	AGR	NNEU	Match
8	speaker_23_center_83.mov	0.486	0.522	0.310	0.432	0.434	0.009
9	speaker_24_center_83.mov	0.417	0.473	0.321	0.445	0.415	0.005
3	speaker_07_center_83.mov	0.436	0.487	0.314	0.415	0.397	0.004
4	speaker_10_center_83.mov	0.499	0.511	0.413	0.469	0.444	-0.005
7	speaker_19_center_83.mov	0.506	0.438	0.431	0.456	0.441	-0.010
5	speaker_11_center_83.mov	0.395	0.342	0.327	0.427	0.355	-0.011
6	speaker_15_center_83.mov	0.566	0.544	0.493	0.587	0.499	-0.189
2	speaker_06_center_83.mov	0.651	0.664	0.607	0.644	0.621	-0.232
10	speaker_27_center_83.mov	0.526	0.661	0.443	0.559	0.554	-0.236
1	speaker_01_center_83.mov	0.565	0.539	0.441	0.593	0.489	-0.271

Аудиообработка информации

Формирование нейросетевой архитектуры модели и загрузка ее весов для получения признаков / оценок на базе экспертных признаков (аудио модальность)

- `_b5.audio_model_hc_` - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе экспертных признаков

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-10 16:37:47] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a5 Лицензия: BSD License

Формирование нейросетевой архитектуры модели

[4]:
```
res_load_audio_model_hc = _b5.load_audio_model_hc(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:37:50] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (аудио модальность) ...

— Время выполнения: 3.03 сек. —

Загрузка весов нейросетевой модели

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['audio']['fi']['hc']['sberdisk']

res_load_audio_model_weights_hc = _b5.load_audio_model_weights_hc(
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:38:05] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (аудио модальность) ...

[2023-12-10 16:38:05] Загрузка файла "weights_2022-05-05_11-27-55.h5" 100.0% ...

— Время выполнения: 0.458 сек. —

Отображение сформированной нейросетевой архитектуры модели

[6]:
```
_b5.audio_model_hc_.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 196, 25)]         0

 lstm (LSTM)                 (None, 196, 64)           23040

 dropout (Dropout)           (None, 196, 64)           0

 lstm_128_a_hc (LSTM)        (None, 128)               98816

 dropout_1 (Dropout)         (None, 128)               0

 dense (Dense)               (None, 5)                 645

=================================================================
Total params: 122,501
Trainable params: 122,501
Non-trainable params: 0
_________________________________________________________________
```

Формирование нейросетевой архитектуры модели и загрузка ее весов для получения признаков / оценок на базе нейросетевых признаков (аудио модальность)

- `_b5.audio_model_nn_` - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе нейросетевых признаков

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-10 16:45:19] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a5 Лицензия: BSD License

Формирование нейросетевой архитектуры модели

[4]:
```
res_load_audio_model_nn = _b5.load_audio_model_nn(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:45:19] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (аудио модальность) ...

— Время выполнения: 1.221 сек. —

Загрузка весов нейросетевой модели

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['audio']['fi']['nn']['sberdisk']

res_load_audio_model_weights_nn = _b5.load_audio_model_weights_nn(
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:45:23] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (аудио модальность) ...

[2023-12-10 16:45:27] Загрузка файла "weights_2022-05-03_07-46-14.h5" 100.0% ...

— Время выполнения: 4.175 сек. —

Отображение сформированной нейросетевой архитектуры модели

[6]:
```
_b5.audio_model_nn_.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 512)               12845568

 dropout (Dropout)           (None, 512)               0

 dense_256 (Dense)           (None, 256)               131328

 dense_1 (Dense)             (None, 5)                 1285

=================================================================
Total params: 27,692,869
Trainable params: 27,692,869
Non-trainable params: 0
_________________________________________________________________
```

Формирование нейросетевых архитектур моделей и загрузка их весов для получения оценок персональных качеств (аудио модальность)

- `_b5.audio_models_b5_` - Нейросетевые модели **tf.keras.Model** для получения оценок персональных качеств

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-14 11:10:51] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a16 Лицензия: BSD License

Формирование нейросетевых архитектур моделей

[4]:
```
res_load_audio_models_b5 = _b5.load_audio_models_b5(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 11:10:51] Формирование нейросетевых архитектур моделей для получения результатов оценки персональных качеств (аудио модальность) ...

— Время выполнения: 0.157 сек. —

Загрузка весов нейросетевых моделей

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url_openness = _b5.weights_for_big5_['audio']['fi']['b5']['openness']['sberdisk']
url_conscientiousness = _b5.weights_for_big5_['audio']['fi']['b5']['conscientiousness'][
↪'sberdisk']
url_extraversion = _b5.weights_for_big5_['audio']['fi']['b5']['extraversion']['sberdisk']
url_agreeableness = _b5.weights_for_big5_['audio']['fi']['b5']['agreeableness']['sberdisk
↪']
url_non_neuroticism = _b5.weights_for_big5_['audio']['fi']['b5']['non_neuroticism'][
↪'sberdisk']

res_load_audio_models_weights_b5 = _b5.load_audio_models_weights_b5(
    url_openness = url_openness, # Открытость опыту
    url_conscientiousness = url_conscientiousness, # Добросовестность
    url_extraversion = url_extraversion, # Экстраверсия
    url_agreeableness = url_agreeableness, # Доброжелательность
    url_non_neuroticism = url_non_neuroticism, # Эмоциональная стабильность
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 11:11:23] Загрузка весов нейросетевых моделей для получения результатов оценки персональных качеств (аудио модальность) ...
[2023-12-14 11:11:23] Загрузка файла "weights_2022-06-15_16-16-20.h5" 100.0% ... Открытость опыту
[2023-12-14 11:11:23] Загрузка файла "weights_2022-06-15_16-21-57.h5" 100.0% ... Добросовестность
[2023-12-14 11:11:23] Загрузка файла "weights_2022-06-15_16-26-41.h5" 100.0% ... Экстраверсия
[2023-12-14 11:11:23] Загрузка файла "weights_2022-06-15_16-32-51.h5" 100.0% ... Доброжелательность
[2023-12-14 11:11:24] Загрузка файла "weights_2022-06-15_16-37-46.h5" 100.0% ... Эмоциональная стабильность
— Время выполнения: 0.907 сек. —

Отображение сформированной нейросетевой архитектуры модели

- `openness` - Открытость опыту
- `conscientiousness` - Добросовестность
- `extraversion` - Экстраверсия
- `agreeableness` - Доброжелательность
- `non_neuroticism` - Эмоциональная стабильность

[6]:
```
_b5.audio_models_b5_['openness'].summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32)]              0

 dense_1 (Dense)             (None, 1)                 33

 activ_1 (Activation)        (None, 1)                 0

=================================================================
Total params: 33 (132.00 Byte)
Trainable params: 33 (132.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Извлечение признаков из акустического сигнала

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-10 16:35:36] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a5 Лицензия: BSD License

Процесс извлечения акустических признаков

[5]:
```
# Настройки ядра
sr = 44100 # Частота дискретизации
# Путь к аудио или видеофайлу
path = 'video_FI/test/_plk5k7PBEg.003.mp4'

hc_features, melspectrogram_features = _b5.get_acoustic_features(
    path = path, # Путь к аудио или видеофайлу
    sr = sr, # Частота дискретизации
    window = 2, # Размер окна сегмента сигнала (в секундах)
    step = 1, # Шаг сдвига окна сегмента сигнала (в секундах)
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:36:06] Извлечение признаков (экспертных и лог мел-спектрограмм) из акустического сигнала ...

[2023-12-10 16:36:11] Статистика извлеченных признаков из акустического сигнала: Общее количество сегментов с: 1. экспертными признаками: 16 2. лог

(continues on next page)

(продолжение с предыдущей страницы)

мел-спектрограммами: 16 Размерность матрицы экспертных признаков одного сегмента: 196 × 25 Размерность тензора с лог мел-спектрограммами одного сегмента: 224 × 224 × 3

— Время выполнения: 5.292 сек. —

Получение прогнозов по аудио

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-14 16:54:20] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a16 Лицензия: BSD License

Получение и отображение версий установленных библиотек

- `_b5.df_pkgs_` - DataFrame с версиями установленных библиотек

[4]:
```
_b5.libs_vers(runtime = True, run = True)
```

```
          Package    Version
1      TensorFlow     2.15.0
2           Keras     2.15.0
3          OpenCV      4.8.1
4       MediaPipe      0.9.0
5           NumPy     1.26.2
6           SciPy     1.11.4
```

(continues on next page)

(продолжение с предыдущей страницы)

```
7          Pandas      2.1.3
8    Scikit-learn      1.3.2
9       OpenSmile      2.5.0
10        Librosa     0.10.1
11      AudioRead      3.0.1
12        IPython     8.18.1
13    PyMediaInfo      6.1.0
14       Requests     2.31.0
15     JupyterLab      4.0.9
16           LIWC      0.5.0
17   Transformers     4.36.0
18  Sentencepiece     0.1.99
19          Torch  2.0.1+cpu
20     Torchaudio  2.0.2+cpu
```

— Время выполнения: 0.005 сек. —

Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам

- _b5.audio_model_hc_ - Нейросетевая модель **tf.keras.Model** для получения оценок по экспертным признакам

[5]:
```
res_load_audio_model_hc = _b5.load_audio_model_hc(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 16:54:20] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (аудио модальность) ...

— Время выполнения: 0.335 сек. —

Загрузка весов нейросетевой модели для получения оценок по экспертным признакам

- _b5.audio_model_hc_ - Нейросетевая модель **tf.keras.Model** для получения оценок по экспертным признакам

[6]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['audio']['fi']['hc']['sberdisk']

res_load_audio_model_weights_hc = _b5.load_audio_model_weights_hc(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 16:54:21] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (аудио модальность) ...

[2023-12-14 16:54:21] Загрузка файла "weights_2022-05-05_11-27-55.h5" 100.0% ...

— Время выполнения: 0.323 сек. —

Формирование нейросетевой архитектуры модели для получения оценок по нейросетевым признакам

- _b5.audio_model_nn_ - Нейросетевая модель **tf.keras.Model** для получения оценок по нейросетевым признакам

[7]:
```
res_load_audio_model_nn = _b5.load_audio_model_nn(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 16:54:21] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (аудио модальность) ...

— Время выполнения: 0.212 сек. —

Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам

- _b5.audio_model_nn_ - Нейросетевая модель **tf.keras.Model** для получения оценок по нейросетевым признакам

[8]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['audio']['fi']['nn']['sberdisk']

res_load_audio_model_weights_nn = _b5.load_audio_model_weights_nn(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели␣
↪из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 16:54:21] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (аудио модальность) ...

[2023-12-14 16:54:22] Загрузка файла "weights_2022-05-03_07-46-14.h5"

— Время выполнения: 0.416 сек. —

Формирование нейросетевых архитектур моделей для получения результатов оценки персональных качеств

- _b5.audio_models_b5_ - Нейросетевые модели **tf.keras.Model** для получения результатов оценки персональных качеств

[9]:
```
res_load_audio_models_b5 = _b5.load_audio_models_b5(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 16:54:22] Формирование нейросетевых архитектур моделей для получения результатов оценки персональных качеств (аудио модальность) ...

— Время выполнения: 0.067 сек. —

Загрузка весов нейросетевых моделей для получения результатов оценки персональных качеств

- _b5.audio_models_b5_ - Нейросетевые модели **tf.keras.Model** для получения результатов оценки персональных качеств

[10]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url_openness = _b5.weights_for_big5_['audio']['fi']['b5']['openness']['sberdisk']
url_conscientiousness = _b5.weights_for_big5_['audio']['fi']['b5']['conscientiousness']['sberdisk']
url_extraversion = _b5.weights_for_big5_['audio']['fi']['b5']['extraversion']['sberdisk']
url_agreeableness = _b5.weights_for_big5_['audio']['fi']['b5']['agreeableness']['sberdisk']
url_non_neuroticism = _b5.weights_for_big5_['audio']['fi']['b5']['non_neuroticism']['sberdisk']

res_load_audio_models_weights_b5 = _b5.load_audio_models_weights_b5(
    url_openness = url_openness, # Открытость опыту
    url_conscientiousness = url_conscientiousness, # Добросовестность
    url_extraversion = url_extraversion, # Экстраверсия
    url_agreeableness = url_agreeableness, # Доброжелательность
    url_non_neuroticism = url_non_neuroticism, # Эмоциональная стабильность
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 16:54:22] Загрузка весов нейросетевых моделей для получения результатов оценки персональных качеств (аудио модальность) ...
[2023-12-14 16:54:22] Загрузка файла "weights_2022-06-15_16-16-20.h5" Открытость опыту
[2023-12-14 16:54:22] Загрузка файла "weights_2022-06-15_16-21-57.h5" Добросовестность
[2023-12-14 16:54:22] Загрузка файла "weights_2022-06-15_16-26-41.h5" Экстраверсия
[2023-12-14 16:54:22] Загрузка файла "weights_2022-06-15_16-32-51.h5" Доброжелательность
[2023-12-14 16:54:22] Загрузка файла "weights_2022-06-15_16-37-46.h5" Эмоциональная стабильность
— Время выполнения: 0.807 сек. —

Получение прогнозов (аудио модальность)

- `_b5.df_files_` - DataFrame с данными
- `_b5.df_accuracy_` - DataFrame с результатами вычисления точности

[11]:
```
# Настройки ядра
_b5.path_to_dataset_ = 'E:/Databases/FirstImpressionsV2/test' # Директория набора данных
# Директории не входящие в выборку
_b5.ignore_dirs_ = []
# Названия ключей для DataFrame набора данных
_b5.keys_dataset_ = ['Path', 'Openness', 'Conscientiousness', 'Extraversion',
↪'Agreeableness', 'Non-Neuroticism']
_b5.ext_ = ['.mp4'] # Расширения искомых файлов
_b5.path_to_logs_ = './logs' # Директория для сохранения LOG файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_['fi']['sberdisk']

res_get_audio_union_predictions = _b5.get_audio_union_predictions(
    depth = 1,          # Глубина иерархии для получения аудио и видеоданных
    recursive = False, # Рекурсивный поиск данных
    sr = 44100,         # Частота дискретизации
    window = 2,         # Размер окна сегмента сигнала (в секундах)
    step = 1,           # Шаг сдвига окна сегмента сигнала (в секундах)
    accuracy = True,    # Вычисление точности
    url_accuracy = url_accuracy,
    logs = True,       # При необходимости формировать LOG файл
    out = True,         # Отображение
    runtime = True,     # Подсчет времени выполнения
    run = True          # Блокировка выполнения
)
```

[2023-12-14 17:59:22] Получение прогнозов и вычисление точности (аудио модальность) ...

2000 из 2000 (100.0%) ... test80_25_Q4wOgixh7E.004.mp4 ...

```
                                                Path  Openness  \
ID
1   E:\Databases\FirstImpressionsV2\test\test80_01...  0.603529
```

(continues on next page)

(продолжение с предыдущей страницы)

```
2   E:\Databases\FirstImpressionsV2\test\test80_01...  0.568246
3   E:\Databases\FirstImpressionsV2\test\test80_01...  0.546209
4   E:\Databases\FirstImpressionsV2\test\test80_01...  0.691056
5   E:\Databases\FirstImpressionsV2\test\test80_01...  0.690808
6   E:\Databases\FirstImpressionsV2\test\test80_01...   0.65728
7   E:\Databases\FirstImpressionsV2\test\test80_01...  0.453781
8   E:\Databases\FirstImpressionsV2\test\test80_01...  0.558594
9   E:\Databases\FirstImpressionsV2\test\test80_01...  0.529081
10  E:\Databases\FirstImpressionsV2\test\test80_01...  0.537279
11  E:\Databases\FirstImpressionsV2\test\test80_01...  0.512779
12  E:\Databases\FirstImpressionsV2\test\test80_01...  0.447102
13  E:\Databases\FirstImpressionsV2\test\test80_01...  0.368372
14  E:\Databases\FirstImpressionsV2\test\test80_01...  0.582539
15  E:\Databases\FirstImpressionsV2\test\test80_01...  0.627705
16  E:\Databases\FirstImpressionsV2\test\test80_01...  0.708798
17  E:\Databases\FirstImpressionsV2\test\test80_01...  0.583968
18  E:\Databases\FirstImpressionsV2\test\test80_01...  0.550836
19  E:\Databases\FirstImpressionsV2\test\test80_01...  0.626745
20  E:\Databases\FirstImpressionsV2\test\test80_01...  0.593014
21  E:\Databases\FirstImpressionsV2\test\test80_01...  0.545921
22  E:\Databases\FirstImpressionsV2\test\test80_01...  0.548432
23  E:\Databases\FirstImpressionsV2\test\test80_01...  0.486083
24  E:\Databases\FirstImpressionsV2\test\test80_01...  0.558323
25  E:\Databases\FirstImpressionsV2\test\test80_01...  0.473017
26  E:\Databases\FirstImpressionsV2\test\test80_01...  0.530967
27  E:\Databases\FirstImpressionsV2\test\test80_01...   0.61807
28  E:\Databases\FirstImpressionsV2\test\test80_01...   0.64703
29  E:\Databases\FirstImpressionsV2\test\test80_01...  0.571473
30  E:\Databases\FirstImpressionsV2\test\test80_01...  0.655007

   Conscientiousness Extraversion Agreeableness Non-Neuroticism
ID
1           0.556223     0.526545      0.579621        0.547629
2           0.465263     0.460744      0.541769        0.511338
3           0.603946     0.469445      0.589493        0.545716
4           0.623856     0.628851      0.614669        0.645813
5           0.589734     0.636104      0.606598         0.63479
6           0.681336     0.571412      0.596052        0.623451
7           0.438842     0.376464      0.520368        0.438252
8           0.598366     0.452183      0.618858        0.571653
9           0.502482     0.426603      0.488263        0.443719
10          0.508283     0.438888      0.579794        0.512117
11          0.447352     0.422968      0.559107        0.491406
12          0.451113     0.364429      0.513031        0.414412
13          0.391985     0.274865       0.42951        0.307666
14          0.432871     0.412363      0.441974        0.462192
15          0.801831     0.528622      0.692623        0.691908
16          0.654007     0.640547      0.632052        0.669044
17          0.644164      0.50463      0.633507         0.59208
18          0.539624     0.468092      0.594872        0.544016
19          0.563271     0.556561      0.561901        0.549236
20          0.421482     0.504798      0.534224        0.532807
```

(continues on next page)

(продолжение с предыдущей страницы)

21	0.479671	0.465769	0.571302	0.518793
22	0.480831	0.453319	0.52774	0.47759
23	0.467779	0.396113	0.444633	0.399402
24	0.537912	0.474172	0.563599	0.52937
25	0.542138	0.370228	0.550093	0.467068
26	0.460241	0.410618	0.507322	0.450027
27	0.506396	0.572248	0.574811	0.563796
28	0.577771	0.565869	0.575279	0.60631
29	0.529536	0.48662	0.535691	0.529022
30	0.606712	0.592804	0.570543	0.600349

[2023-12-14 17:59:22] Точность по отдельным персональным качествам личности человека ...

Metrics	Openness	Conscientiousness	Extraversion	Agreeableness	\
MAE	0.0916	0.0925	0.0932	0.0918	
Accuracy	0.9084	0.9075	0.9068	0.9082	

Metrics	Non-Neuroticism	Mean
MAE	0.094	0.0926
Accuracy	0.906	0.9074

[2023-12-14 17:59:22] Средняя средних абсолютных ошибок: 0.0926, средняя точность: 0.9074 ...

Лог файлы успешно сохранены ...

— Время выполнения: 3899.26 сек. —

Видеообработка информации

Формирование нейросетевой архитектуры модели и загрузка ее весов для получения признаков / оценок на базе экспертных признаков (видео модальность)

- `_b5.video_model_hc_` - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе экспертных признаков

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-10 17:11:13] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a5 Лицензия: BSD License

Формирование нейросетевой архитектуры модели (FI V2)

[4]:
```
res_load_video_model_hc = _b5.load_video_model_hc(
    lang = 'en',
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 17:11:13] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (видео модальность) ...

— Время выполнения: 0.789 сек. —

Загрузка весов нейросетевой модели

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video']['fi']['hc']['sberdisk']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
res_load_video_model_weights_hc = _b5.load_video_model_weights_hc(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 17:11:14] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (видео модальность) ...

[2023-12-10 17:11:14] Загрузка файла "weights_2022-08-27_18-53-35.h5" 100.0% ...

— Время выполнения: 0.226 сек. —

Отображение сформированной нейросетевой архитектуры модели

[6]:
```
_b5.video_model_hc_.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 10, 115)]         0

 lstm (LSTM)                 (None, 10, 64)            46080

 dropout (Dropout)           (None, 10, 64)            0

 lstm_128_v_hc (LSTM)        (None, 128)               98816

 dropout_1 (Dropout)         (None, 128)               0

 dense (Dense)               (None, 5)                 645

=================================================================
Total params: 145,541
Trainable params: 145,541
Non-trainable params: 0
_________________________________________________________________
```

Формирование нейросетевой архитектуры модели (MuPTA)

[7]:
```
res_load_video_model_hc = _b5.load_video_model_hc(
    lang = 'ru',
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 17:11:14] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (видео модальность) ...

— Время выполнения: 0.25 сек. —

Загрузка весов нейросетевой модели

[8]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video']['mupta']['hc']['sberdisk']

res_load_video_model_weights_hc = _b5.load_video_model_weights_hc(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 17:11:14] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (видео модальность) ...

[2023-12-10 17:11:15] Загрузка файла “vhc_mupta_2022-07-22_10-02-37.h5” 100.0% ...

— Время выполнения: 0.307 сек. —

Отображение сформированной нейросетевой архитектуры модели

[9]:
```
_b5.video_model_hc_.summary()
```

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 10, 109)]         0

 lstm_1 (LSTM)               (None, 10, 64)            44544

 dropout_2 (Dropout)         (None, 10, 64)            0

 lstm_128_v_hc (LSTM)        (None, 128)               98816

 dropout_3 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 5)                 645

=================================================================
Total params: 144,005
```

(continues on next page)

(продолжение с предыдущей страницы)

```
Trainable params: 144,005
Non-trainable params: 0
_________________________________________________________________
```

Формирование нейросетевой архитектуры модели и загрузка ее весов для получения нейросетевых признаков (видео модальность)

- _b5.video_model_deep_fe_ - Нейросетевая модель **tf.keras.Model** для получения нейросетевых признаков

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

```
[2023-12-10 17:08:31] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a5 Лицензия: BSD License
```

Формирование нейросетевой архитектуры модели (FI V2)

[4]:
```
res_load_video_model_deep_fe = _b5.load_video_model_deep_fe(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

```
[2023-12-10 17:08:31] Формирование нейросетевой архитектуры для получения нейросетевых признаков (видео модальность) ...
```

— Время выполнения: 1.118 сек. —

Загрузка весов нейросетевой модели

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video']['fi']['fe']['sberdisk']

res_load_video_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 17:08:32] Загрузка весов нейросетевой модели для получения нейросетевых признаков (видео модальность) ...

[2023-12-10 17:08:36] Загрузка файла “weights_2022-11-01_12-27-07.h5” 100.0% ...

— Время выполнения: 4.042 сек. —

Отображение сформированной нейросетевой архитектуры модели

[6]:
```
_b5.video_model_deep_fe_.summary()
```

```
Model: "model_1"
__________________________________________________________________________________________
↪_________
 Layer (type)                   Output Shape         Param #     Connected to
==========================================================================================
↪=========
 input_1 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1/7x7_s2 (Conv2D)          (None, 112, 112, 64  9408        ['input_1[0][0]']
                                )

 conv1/7x7_s2/bn (BatchNormaliz  (None, 112, 112, 64  256        ['conv1/7x7_s2[0][0]']
 ation)                         )

 activation (Activation)        (None, 112, 112, 64  0           ['conv1/7x7_s2/bn[0][0]
↪']
                                )

 max_pooling2d (MaxPooling2D)   (None, 55, 55, 64)   0           ['activation[0][0]']

 conv2_1_1x1_reduce (Conv2D)    (None, 55, 55, 64)   4096        ['max_pooling2d[0][0]']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
conv2_1_1x1_reduce/bn (BatchNo  (None, 55, 55, 64)  256         ['conv2_1_1x1_
↪reduce[0][0]']
rmalization)

activation_1 (Activation)      (None, 55, 55, 64)   0           ['conv2_1_1x1_reduce/
↪bn[0][0]']

conv2_1_3x3 (Conv2D)           (None, 55, 55, 64)   36864       ['activation_1[0][0]']

conv2_1_3x3/bn (BatchNormaliza  (None, 55, 55, 64)  256         ['conv2_1_3x3[0][0]']
tion)

activation_2 (Activation)      (None, 55, 55, 64)   0           ['conv2_1_3x3/bn[0][0]']

conv2_1_1x1_increase (Conv2D)  (None, 55, 55, 256)  16384       ['activation_2[0][0]']

conv2_1_1x1_proj (Conv2D)      (None, 55, 55, 256)  16384       ['max_pooling2d[0][0]']

conv2_1_1x1_increase/bn (Batch  (None, 55, 55, 256)  1024       ['conv2_1_1x1_
↪increase[0][0]']
Normalization)

conv2_1_1x1_proj/bn (BatchNorm  (None, 55, 55, 256)  1024       ['conv2_1_1x1_proj[0][0]
↪']
alization)

add (Add)                      (None, 55, 55, 256)  0           ['conv2_1_1x1_increase/
↪bn[0][0]',
                                                                 'conv2_1_1x1_proj/
↪bn[0][0]']

activation_3 (Activation)      (None, 55, 55, 256)  0           ['add[0][0]']

conv2_2_1x1_reduce (Conv2D)    (None, 55, 55, 64)   16384       ['activation_3[0][0]']

conv2_2_1x1_reduce/bn (BatchNo  (None, 55, 55, 64)  256         ['conv2_2_1x1_
↪reduce[0][0]']
rmalization)

activation_4 (Activation)      (None, 55, 55, 64)   0           ['conv2_2_1x1_reduce/
↪bn[0][0]']

conv2_2_3x3 (Conv2D)           (None, 55, 55, 64)   36864       ['activation_4[0][0]']

conv2_2_3x3/bn (BatchNormaliza  (None, 55, 55, 64)  256         ['conv2_2_3x3[0][0]']
tion)

activation_5 (Activation)      (None, 55, 55, 64)   0           ['conv2_2_3x3/bn[0][0]']

conv2_2_1x1_increase (Conv2D)  (None, 55, 55, 256)  16384       ['activation_5[0][0]']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
conv2_2_1x1_increase/bn (Batch  (None, 55, 55, 256)  1024       ['conv2_2_1x1_
↪increase[0][0]']
Normalization)

add_1 (Add)                     (None, 55, 55, 256)  0          ['conv2_2_1x1_increase/
↪bn[0][0]',
                                                                 'activation_3[0][0]']

activation_6 (Activation)       (None, 55, 55, 256)  0          ['add_1[0][0]']

conv2_3_1x1_reduce (Conv2D)     (None, 55, 55, 64)   16384      ['activation_6[0][0]']

conv2_3_1x1_reduce/bn (BatchNo  (None, 55, 55, 64)  256        ['conv2_3_1x1_
↪reduce[0][0]']
rmalization)

activation_7 (Activation)       (None, 55, 55, 64)   0          ['conv2_3_1x1_reduce/
↪bn[0][0]']

conv2_3_3x3 (Conv2D)            (None, 55, 55, 64)   36864      ['activation_7[0][0]']

conv2_3_3x3/bn (BatchNormaliza  (None, 55, 55, 64)  256        ['conv2_3_3x3[0][0]']
tion)

activation_8 (Activation)       (None, 55, 55, 64)   0          ['conv2_3_3x3/bn[0][0]']

conv2_3_1x1_increase (Conv2D)   (None, 55, 55, 256)  16384      ['activation_8[0][0]']

conv2_3_1x1_increase/bn (Batch  (None, 55, 55, 256)  1024       ['conv2_3_1x1_
↪increase[0][0]']
Normalization)

add_2 (Add)                     (None, 55, 55, 256)  0          ['conv2_3_1x1_increase/
↪bn[0][0]',
                                                                 'activation_6[0][0]']

activation_9 (Activation)       (None, 55, 55, 256)  0          ['add_2[0][0]']

conv3_1_1x1_reduce (Conv2D)     (None, 28, 28, 128)  32768      ['activation_9[0][0]']

conv3_1_1x1_reduce/bn (BatchNo  (None, 28, 28, 128)  512        ['conv3_1_1x1_
↪reduce[0][0]']
rmalization)

activation_10 (Activation)      (None, 28, 28, 128)  0          ['conv3_1_1x1_reduce/
↪bn[0][0]']

conv3_1_3x3 (Conv2D)            (None, 28, 28, 128)  147456     ['activation_10[0][0]']

conv3_1_3x3/bn (BatchNormaliza  (None, 28, 28, 128)  512        ['conv3_1_3x3[0][0]']
tion)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
activation_11 (Activation)     (None, 28, 28, 128)  0           ['conv3_1_3x3/bn[0][0]']

conv3_1_1x1_increase (Conv2D)  (None, 28, 28, 512)  65536       ['activation_11[0][0]']

conv3_1_1x1_proj (Conv2D)      (None, 28, 28, 512)  131072      ['activation_9[0][0]']

conv3_1_1x1_increase/bn (Batch  (None, 28, 28, 512)  2048       ['conv3_1_1x1_
↪increase[0][0]']
Normalization)

conv3_1_1x1_proj/bn (BatchNorm  (None, 28, 28, 512)  2048       ['conv3_1_1x1_proj[0][0]
↪']
alization)

add_3 (Add)                    (None, 28, 28, 512)  0           ['conv3_1_1x1_increase/
↪bn[0][0]',
                                                                 'conv3_1_1x1_proj/
↪bn[0][0]']

activation_12 (Activation)     (None, 28, 28, 512)  0           ['add_3[0][0]']

conv3_2_1x1_reduce (Conv2D)    (None, 28, 28, 128)  65536       ['activation_12[0][0]']

conv3_2_1x1_reduce/bn (BatchNo  (None, 28, 28, 128)  512       ['conv3_2_1x1_
↪reduce[0][0]']
rmalization)

activation_13 (Activation)     (None, 28, 28, 128)  0           ['conv3_2_1x1_reduce/
↪bn[0][0]']

conv3_2_3x3 (Conv2D)           (None, 28, 28, 128)  147456      ['activation_13[0][0]']

conv3_2_3x3/bn (BatchNormaliza  (None, 28, 28, 128)  512       ['conv3_2_3x3[0][0]']
tion)

activation_14 (Activation)     (None, 28, 28, 128)  0           ['conv3_2_3x3/bn[0][0]']

conv3_2_1x1_increase (Conv2D)  (None, 28, 28, 512)  65536       ['activation_14[0][0]']

conv3_2_1x1_increase/bn (Batch  (None, 28, 28, 512)  2048       ['conv3_2_1x1_
↪increase[0][0]']
Normalization)

add_4 (Add)                    (None, 28, 28, 512)  0           ['conv3_2_1x1_increase/
↪bn[0][0]',
                                                                 'activation_12[0][0]']

activation_15 (Activation)     (None, 28, 28, 512)  0           ['add_4[0][0]']

conv3_3_1x1_reduce (Conv2D)    (None, 28, 28, 128)  65536       ['activation_15[0][0]']

conv3_3_1x1_reduce/bn (BatchNo  (None, 28, 28, 128)  512       ['conv3_3_1x1_
```

(continues on next page)

(продолжение с предыдущей страницы)

```
↪reduce[0][0]']
rmalization)

activation_16 (Activation)     (None, 28, 28, 128)  0           ['conv3_3_1x1_reduce/
↪bn[0][0]']

conv3_3_3x3 (Conv2D)           (None, 28, 28, 128)  147456      ['activation_16[0][0]']

conv3_3_3x3/bn (BatchNormaliza  (None, 28, 28, 128)  512        ['conv3_3_3x3[0][0]']
tion)

activation_17 (Activation)     (None, 28, 28, 128)  0           ['conv3_3_3x3/bn[0][0]']

conv3_3_1x1_increase (Conv2D)  (None, 28, 28, 512)  65536       ['activation_17[0][0]']

conv3_3_1x1_increase/bn (Batch  (None, 28, 28, 512)  2048       ['conv3_3_1x1_
↪increase[0][0]']
Normalization)

add_5 (Add)                    (None, 28, 28, 512)  0           ['conv3_3_1x1_increase/
↪bn[0][0]',
                                                                 'activation_15[0][0]']

activation_18 (Activation)     (None, 28, 28, 512)  0           ['add_5[0][0]']

conv3_4_1x1_reduce (Conv2D)    (None, 28, 28, 128)  65536       ['activation_18[0][0]']

conv3_4_1x1_reduce/bn (BatchNo  (None, 28, 28, 128)  512        ['conv3_4_1x1_
↪reduce[0][0]']
rmalization)

activation_19 (Activation)     (None, 28, 28, 128)  0           ['conv3_4_1x1_reduce/
↪bn[0][0]']

conv3_4_3x3 (Conv2D)           (None, 28, 28, 128)  147456      ['activation_19[0][0]']

conv3_4_3x3/bn (BatchNormaliza  (None, 28, 28, 128)  512        ['conv3_4_3x3[0][0]']
tion)

activation_20 (Activation)     (None, 28, 28, 128)  0           ['conv3_4_3x3/bn[0][0]']

conv3_4_1x1_increase (Conv2D)  (None, 28, 28, 512)  65536       ['activation_20[0][0]']

conv3_4_1x1_increase/bn (Batch  (None, 28, 28, 512)  2048       ['conv3_4_1x1_
↪increase[0][0]']
Normalization)

add_6 (Add)                    (None, 28, 28, 512)  0           ['conv3_4_1x1_increase/
↪bn[0][0]',
                                                                 'activation_18[0][0]']

activation_21 (Activation)     (None, 28, 28, 512)  0           ['add_6[0][0]']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
conv4_1_1x1_reduce (Conv2D)    (None, 14, 14, 256)  131072      ['activation_21[0][0]']

conv4_1_1x1_reduce/bn (BatchNo  (None, 14, 14, 256)  1024       ['conv4_1_1x1_
↪reduce[0][0]']
rmalization)

activation_22 (Activation)     (None, 14, 14, 256)  0           ['conv4_1_1x1_reduce/
↪bn[0][0]']

conv4_1_3x3 (Conv2D)           (None, 14, 14, 256)  589824      ['activation_22[0][0]']

conv4_1_3x3/bn (BatchNormaliza  (None, 14, 14, 256)  1024       ['conv4_1_3x3[0][0]']
tion)

activation_23 (Activation)     (None, 14, 14, 256)  0           ['conv4_1_3x3/bn[0][0]']

conv4_1_1x1_increase (Conv2D)  (None, 14, 14, 1024  262144      ['activation_23[0][0]']
                               )

conv4_1_1x1_proj (Conv2D)      (None, 14, 14, 1024  524288      ['activation_21[0][0]']
                               )

conv4_1_1x1_increase/bn (Batch  (None, 14, 14, 1024  4096       ['conv4_1_1x1_
↪increase[0][0]']
Normalization)                 )

conv4_1_1x1_proj/bn (BatchNorm  (None, 14, 14, 1024  4096       ['conv4_1_1x1_proj[0][0]
↪']
alization)                     )

add_7 (Add)                    (None, 14, 14, 1024  0           ['conv4_1_1x1_increase/
↪bn[0][0]',
                               )                                 'conv4_1_1x1_proj/
↪bn[0][0]']

activation_24 (Activation)     (None, 14, 14, 1024  0           ['add_7[0][0]']
                               )

conv4_2_1x1_reduce (Conv2D)    (None, 14, 14, 256)  262144      ['activation_24[0][0]']

conv4_2_1x1_reduce/bn (BatchNo  (None, 14, 14, 256)  1024       ['conv4_2_1x1_
↪reduce[0][0]']
rmalization)

activation_25 (Activation)     (None, 14, 14, 256)  0           ['conv4_2_1x1_reduce/
↪bn[0][0]']

conv4_2_3x3 (Conv2D)           (None, 14, 14, 256)  589824      ['activation_25[0][0]']

conv4_2_3x3/bn (BatchNormaliza  (None, 14, 14, 256)  1024       ['conv4_2_3x3[0][0]']
tion)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
activation_26 (Activation)     (None, 14, 14, 256)  0           ['conv4_2_3x3/bn[0][0]']

conv4_2_1x1_increase (Conv2D)  (None, 14, 14, 1024  262144      ['activation_26[0][0]']
                               )

conv4_2_1x1_increase/bn (Batch  (None, 14, 14, 1024  4096       ['conv4_2_1x1_
↪increase[0][0]']
Normalization)                 )

add_8 (Add)                    (None, 14, 14, 1024  0           ['conv4_2_1x1_increase/
↪bn[0][0]',
                               )                                 'activation_24[0][0]']

activation_27 (Activation)     (None, 14, 14, 1024  0           ['add_8[0][0]']
                               )

conv4_3_1x1_reduce (Conv2D)    (None, 14, 14, 256)  262144      ['activation_27[0][0]']

conv4_3_1x1_reduce/bn (BatchNo  (None, 14, 14, 256)  1024       ['conv4_3_1x1_
↪reduce[0][0]']
rmalization)

activation_28 (Activation)     (None, 14, 14, 256)  0           ['conv4_3_1x1_reduce/
↪bn[0][0]']

conv4_3_3x3 (Conv2D)           (None, 14, 14, 256)  589824      ['activation_28[0][0]']

conv4_3_3x3/bn (BatchNormaliza  (None, 14, 14, 256)  1024       ['conv4_3_3x3[0][0]']
tion)

activation_29 (Activation)     (None, 14, 14, 256)  0           ['conv4_3_3x3/bn[0][0]']

conv4_3_1x1_increase (Conv2D)  (None, 14, 14, 1024  262144      ['activation_29[0][0]']
                               )

conv4_3_1x1_increase/bn (Batch  (None, 14, 14, 1024  4096       ['conv4_3_1x1_
↪increase[0][0]']
Normalization)                 )

add_9 (Add)                    (None, 14, 14, 1024  0           ['conv4_3_1x1_increase/
↪bn[0][0]',
                               )                                 'activation_27[0][0]']

activation_30 (Activation)     (None, 14, 14, 1024  0           ['add_9[0][0]']
                               )

conv4_4_1x1_reduce (Conv2D)    (None, 14, 14, 256)  262144      ['activation_30[0][0]']

conv4_4_1x1_reduce/bn (BatchNo  (None, 14, 14, 256)  1024       ['conv4_4_1x1_
↪reduce[0][0]']
rmalization)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
activation_31 (Activation)     (None, 14, 14, 256)  0           ['conv4_4_1x1_reduce/
↪bn[0][0]']

conv4_4_3x3 (Conv2D)           (None, 14, 14, 256)  589824      ['activation_31[0][0]']

conv4_4_3x3/bn (BatchNormaliza  (None, 14, 14, 256)  1024       ['conv4_4_3x3[0][0]']
tion)

activation_32 (Activation)     (None, 14, 14, 256)  0           ['conv4_4_3x3/bn[0][0]']

conv4_4_1x1_increase (Conv2D)  (None, 14, 14, 1024  262144      ['activation_32[0][0]']
                               )

conv4_4_1x1_increase/bn (Batch  (None, 14, 14, 1024  4096       ['conv4_4_1x1_
↪increase[0][0]']
Normalization)                 )

add_10 (Add)                   (None, 14, 14, 1024  0           ['conv4_4_1x1_increase/
↪bn[0][0]',
                               )                                 'activation_30[0][0]']

activation_33 (Activation)     (None, 14, 14, 1024  0           ['add_10[0][0]']
                               )

conv4_5_1x1_reduce (Conv2D)    (None, 14, 14, 256)  262144      ['activation_33[0][0]']

conv4_5_1x1_reduce/bn (BatchNo  (None, 14, 14, 256)  1024       ['conv4_5_1x1_
↪reduce[0][0]']
rmalization)

activation_34 (Activation)     (None, 14, 14, 256)  0           ['conv4_5_1x1_reduce/
↪bn[0][0]']

conv4_5_3x3 (Conv2D)           (None, 14, 14, 256)  589824      ['activation_34[0][0]']

conv4_5_3x3/bn (BatchNormaliza  (None, 14, 14, 256)  1024       ['conv4_5_3x3[0][0]']
tion)

activation_35 (Activation)     (None, 14, 14, 256)  0           ['conv4_5_3x3/bn[0][0]']

conv4_5_1x1_increase (Conv2D)  (None, 14, 14, 1024  262144      ['activation_35[0][0]']
                               )

conv4_5_1x1_increase/bn (Batch  (None, 14, 14, 1024  4096       ['conv4_5_1x1_
↪increase[0][0]']
Normalization)                 )

add_11 (Add)                   (None, 14, 14, 1024  0           ['conv4_5_1x1_increase/
↪bn[0][0]',
                               )                                 'activation_33[0][0]']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
activation_36 (Activation)     (None, 14, 14, 1024  0           ['add_11[0][0]']
                               )

conv4_6_1x1_reduce (Conv2D)    (None, 14, 14, 256)  262144      ['activation_36[0][0]']

conv4_6_1x1_reduce/bn (BatchNo  (None, 14, 14, 256)  1024       ['conv4_6_1x1_
↪reduce[0][0]']
rmalization)

activation_37 (Activation)     (None, 14, 14, 256)  0           ['conv4_6_1x1_reduce/
↪bn[0][0]']

conv4_6_3x3 (Conv2D)           (None, 14, 14, 256)  589824      ['activation_37[0][0]']

conv4_6_3x3/bn (BatchNormaliza  (None, 14, 14, 256)  1024       ['conv4_6_3x3[0][0]']
tion)

activation_38 (Activation)     (None, 14, 14, 256)  0           ['conv4_6_3x3/bn[0][0]']

conv4_6_1x1_increase (Conv2D)  (None, 14, 14, 1024  262144      ['activation_38[0][0]']
                               )

conv4_6_1x1_increase/bn (Batch  (None, 14, 14, 1024  4096       ['conv4_6_1x1_
↪increase[0][0]']
Normalization)                 )

add_12 (Add)                   (None, 14, 14, 1024  0           ['conv4_6_1x1_increase/
↪bn[0][0]',
                               )                                 'activation_36[0][0]']

activation_39 (Activation)     (None, 14, 14, 1024  0           ['add_12[0][0]']
                               )

conv5_1_1x1_reduce (Conv2D)    (None, 7, 7, 512)    524288      ['activation_39[0][0]']

conv5_1_1x1_reduce/bn (BatchNo  (None, 7, 7, 512)   2048        ['conv5_1_1x1_
↪reduce[0][0]']
rmalization)

activation_40 (Activation)     (None, 7, 7, 512)    0           ['conv5_1_1x1_reduce/
↪bn[0][0]']

conv5_1_3x3 (Conv2D)           (None, 7, 7, 512)    2359296     ['activation_40[0][0]']

conv5_1_3x3/bn (BatchNormaliza  (None, 7, 7, 512)   2048        ['conv5_1_3x3[0][0]']
tion)

activation_41 (Activation)     (None, 7, 7, 512)    0           ['conv5_1_3x3/bn[0][0]']

conv5_1_1x1_increase (Conv2D)  (None, 7, 7, 2048)   1048576     ['activation_41[0][0]']

conv5_1_1x1_proj (Conv2D)      (None, 7, 7, 2048)   2097152     ['activation_39[0][0]']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
conv5_1_1x1_increase/bn (Batch  (None, 7, 7, 2048)  8192        ['conv5_1_1x1_
↪increase[0][0]']
Normalization)

conv5_1_1x1_proj/bn (BatchNorm  (None, 7, 7, 2048)  8192        ['conv5_1_1x1_proj[0][0]
↪']
alization)

add_13 (Add)                    (None, 7, 7, 2048)  0           ['conv5_1_1x1_increase/
↪bn[0][0]',
                                                                 'conv5_1_1x1_proj/
↪bn[0][0]']

activation_42 (Activation)      (None, 7, 7, 2048)  0           ['add_13[0][0]']

conv5_2_1x1_reduce (Conv2D)     (None, 7, 7, 512)   1048576     ['activation_42[0][0]']

conv5_2_1x1_reduce/bn (BatchNo  (None, 7, 7, 512)   2048        ['conv5_2_1x1_
↪reduce[0][0]']
rmalization)

activation_43 (Activation)      (None, 7, 7, 512)   0           ['conv5_2_1x1_reduce/
↪bn[0][0]']

conv5_2_3x3 (Conv2D)            (None, 7, 7, 512)   2359296     ['activation_43[0][0]']

conv5_2_3x3/bn (BatchNormaliza  (None, 7, 7, 512)   2048        ['conv5_2_3x3[0][0]']
tion)

activation_44 (Activation)      (None, 7, 7, 512)   0           ['conv5_2_3x3/bn[0][0]']

conv5_2_1x1_increase (Conv2D)   (None, 7, 7, 2048)  1048576     ['activation_44[0][0]']

conv5_2_1x1_increase/bn (Batch  (None, 7, 7, 2048)  8192        ['conv5_2_1x1_
↪increase[0][0]']
Normalization)

add_14 (Add)                    (None, 7, 7, 2048)  0           ['conv5_2_1x1_increase/
↪bn[0][0]',
                                                                 'activation_42[0][0]']

activation_45 (Activation)      (None, 7, 7, 2048)  0           ['add_14[0][0]']

conv5_3_1x1_reduce (Conv2D)     (None, 7, 7, 512)   1048576     ['activation_45[0][0]']

conv5_3_1x1_reduce/bn (BatchNo  (None, 7, 7, 512)   2048        ['conv5_3_1x1_
↪reduce[0][0]']
rmalization)

activation_46 (Activation)      (None, 7, 7, 512)   0           ['conv5_3_1x1_reduce/
↪bn[0][0]']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 conv5_3_3x3 (Conv2D)           (None, 7, 7, 512)    2359296     ['activation_46[0][0]']

 conv5_3_3x3/bn (BatchNormaliza  (None, 7, 7, 512)   2048        ['conv5_3_3x3[0][0]']
 tion)

 activation_47 (Activation)     (None, 7, 7, 512)    0           ['conv5_3_3x3/bn[0][0]']

 conv5_3_1x1_increase (Conv2D)  (None, 7, 7, 2048)   1048576     ['activation_47[0][0]']

 conv5_3_1x1_increase/bn (Batch  (None, 7, 7, 2048)  8192        ['conv5_3_1x1_
↪increase[0][0]']
 Normalization)

 add_15 (Add)                   (None, 7, 7, 2048)   0           ['conv5_3_1x1_increase/
↪bn[0][0]',
                                                                  'activation_45[0][0]']

 activation_48 (Activation)     (None, 7, 7, 2048)   0           ['add_15[0][0]']

 avg_pool (AveragePooling2D)    (None, 1, 1, 2048)   0           ['activation_48[0][0]']

 global_average_pooling2d (Glob  (None, 2048)        0           ['avg_pool[0][0]']
 alAveragePooling2D)

 gaussian_noise (GaussianNoise)  (None, 2048)        0           ['global_average_
↪pooling2d[0][0]'
                                                                 ]

 dense_x (Dense)                (None, 512)          1049088     ['gaussian_noise[0][0]']

==================================================================================================
Total params: 24,610,240
Trainable params: 24,557,120
Non-trainable params: 53,120
--------------------------------------------------------------------------------------------------
↪---------
```

Формирование нейросетевой архитектуры модели и загрузка ее весов для получения оценок по нейросетевым признакам (видео модальность)

- `_b5.video_model_nn_` - Нейросетевая модель **tf.keras.Model** для получения оценок по нейросетевым признакам

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-10 17:12:11] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a5 Лицензия: BSD License

Формирование нейросетевой архитектуры модели (FI V2)

[4]:
```
res_load_video_model_nn = _b5.load_video_model_nn(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 17:12:11] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (видео модальность) ...

— Время выполнения: 1.559 сек. —

Загрузка весов нейросетевой модели

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video']['fi']['nn']['sberdisk']

res_load_video_model_weights_nn = _b5.load_video_model_weights_nn(
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 17:12:13] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (видео модальность) ...

[2023-12-10 17:12:14] Загрузка файла "weights_2022-03-22_16-31-48.h5" 100.0% ...

— Время выполнения: 1.053 сек. —

Отображение сформированной нейросетевой архитектуры модели

[6]:
```
_b5.video_model_nn_.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 10, 512)]         0

 lstm_1024_v_nn (LSTM)       (None, 1024)              6295552

 dropout (Dropout)           (None, 1024)              0

 dense (Dense)               (None, 5)                 5125

 activation (Activation)     (None, 5)                 0

=================================================================
Total params: 6,300,677
Trainable params: 6,300,677
Non-trainable params: 0
_________________________________________________________________
```

Формирование нейросетевых архитектур моделей и загрузка их весов для получения результатов оценки персональных качеств (видео модальность)

- `_b5.video_models_b5_` - Нейросетевые модели **tf.keras.Model** для получения результатов оценки персональных качеств

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-14 21:04:19] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a16 Лицензия: BSD License

Формирование нейросетевых архитектур моделей (FI V2)

[4]:
```
res_load_video_models_b5 = _b5.load_video_models_b5(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 21:04:19] Формирование нейросетевых архитектур моделей для получения результатов оценки персональных качеств (видео модальность) ...

— Время выполнения: 0.094 сек. —

Загрузка весов нейросетевых моделей

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url_openness = _b5.weights_for_big5_['video']['fi']['b5']['openness']['sberdisk']
url_conscientiousness = _b5.weights_for_big5_['video']['fi']['b5']['conscientiousness'][
↪'sberdisk']
url_extraversion = _b5.weights_for_big5_['video']['fi']['b5']['extraversion']['sberdisk']
url_agreeableness = _b5.weights_for_big5_['video']['fi']['b5']['agreeableness']['sberdisk
↪']
url_non_neuroticism = _b5.weights_for_big5_['video']['fi']['b5']['non_neuroticism'][
↪'sberdisk']

res_load_video_models_weights_b5 = _b5.load_video_models_weights_b5(
    url_openness = url_openness, # Открытость опыту
    url_conscientiousness = url_conscientiousness, # Добросовестность
    url_extraversion = url_extraversion, # Экстраверсия
    url_agreeableness = url_agreeableness, # Доброжелательность
    url_non_neuroticism = url_non_neuroticism, # Нейротизм
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 21:04:19] Загрузка весов нейросетевых моделей для получения результатов оценки персональных качеств (видео модальность) ...

[2023-12-14 21:04:19] Загрузка файла "weights_2022-06-15_16-46-30.h5" 100.0% ...
Открытость опыту

[2023-12-14 21:04:20] Загрузка файла "weights_2022-06-15_16-48-50.h5" 100.0% ...
Добросовестность

[2023-12-14 21:04:20] Загрузка файла "weights_2022-06-15_16-54-06.h5" 100.0% ...
Экстраверсия

[2023-12-14 21:04:20] Загрузка файла "weights_2022-06-15_17-02-03.h5" 100.0% ...
Доброжелательность

[2023-12-14 21:04:20] Загрузка файла "weights_2022-06-15_17-06-15.h5" 100.0% ...
Эмоциональная стабильность

— Время выполнения: 0.998 сек. —

Отображение сформированной нейросетевой архитектуры модели

- `openness` - Открытость опыту
- `conscientiousness` - Добросовестность
- `extraversion` - Экстраверсия
- `agreeableness` - Доброжелательность
- `non_neuroticism` - Эмоциональная стабильность

[6]:
```
_b5.video_models_b5_['openness'].summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32)]              0

 dense_1 (Dense)             (None, 1)                 33

 activ_1 (Activation)        (None, 1)                 0

=================================================================
Total params: 33 (132.00 Byte)
Trainable params: 33 (132.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Извлечение признаков из визуального сигнала

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

```
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2024-03-28 21:50:44] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a22 Лицензия: BSD License

Формирование нейросетевой архитектуры модели

[4]:
```
res_load_video_model_deep_fe = _b5.load_video_model_deep_fe(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2024-03-28 21:50:46] Формирование нейросетевой архитектуры для получения нейросетевых признаков (видео модальность) ...

— Время выполнения: 1.001 сек. —

Загрузка весов нейросетевой модели

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video']['fi']['fe']['sberdisk']

res_load_video_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2024-03-28 21:50:50] Загрузка весов нейросетевой модели для получения нейросетевых признаков (видео модальность) ...

[2024-03-28 21:50:56] Загрузка файла “weights_2022-11-01_12-27-07.h5” 100.0% ...

— Время выполнения: 6.461 сек. —

Процесс извлечения визуальных признаков (FI V2)

[6]:
```
# Настройки ядра
# Путь к видеофайлу
path = 'video_FI/test/_plk5k7PBEg.003.mp4'

hc_features, nn_features = _b5.get_visual_features(
    path = path, # Путь к видеофайлу
    reduction_fps = 5, # Понижение кадровой частоты
    window = 10, # Размер окна сегмента сигнала (в кадрах)
    step = 5, # Шаг сдвига окна сегмента сигнала (в кадрах)
    lang = 'en',
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2024-03-28 21:50:58] Извлечение признаков (экспертных и нейросетевых) из визуального сигнала ...

[2024-03-28 21:51:22] Статистика извлеченных признаков из визуального сигнала: Общее количество сегментов с: 1. экспертными признаками: 16 2. нейросетевыми признаками: 16 Размерность матрицы экспертных признаков одного сегмента: 10 × 115 Размерность матрицы с нейросетевыми признаками одного сегмента: 10 × 512 Понижение кадровой частоты: с 30 до 5

— Время выполнения: 23.465 сек. —

Процесс извлечения визуальных признаков (MuPTA)

[7]:
```
# Настройки ядра
# Путь к видеофайлу
path = 'video_FI/test/_plk5k7PBEg.003.mp4'

hc_features, nn_features = _b5.get_visual_features(
    path = path, # Путь к видеофайлу
    reduction_fps = 5, # Понижение кадровой частоты
    window = 10, # Размер окна сегмента сигнала (в кадрах)
    step = 5, # Шаг сдвига окна сегмента сигнала (в кадрах)
    lang = 'ru',
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2024-03-28 21:51:25] Извлечение признаков (экспертных и нейросетевых) из визуального сигнала ...

[2024-03-28 21:51:43] Статистика извлеченных признаков из визуального сигнала: Общее количество сегментов с: 1. экспертными признаками: 16 2. нейросетевыми признаками: 16 Размерность матрицы экспертных признаков одного сегмента: 10 × 109 Размерность матрицы с нейросетевыми признаками одного сегмента: 10 × 512 Понижение кадровой частоты: с 30 до 5

— Время выполнения: 18.659 сек. —

[ ]:

Получение прогнозов по видео

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-14 21:05:26] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a16 Лицензия: BSD License

Получение и отображение версий установленных библиотек

- `_b5.df_pkgs_` - DataFrame с версиями установленных библиотек

[4]:
```
_b5.libs_vers(runtime = True, run = True)
```

```
          Package    Version
1      TensorFlow     2.15.0
2           Keras     2.15.0
3          OpenCV      4.8.1
4       MediaPipe      0.9.0
5           NumPy     1.26.2
6           SciPy     1.11.4
7          Pandas      2.1.3
8    Scikit-learn      1.3.2
9       OpenSmile      2.5.0
```

(continues on next page)

(продолжение с предыдущей страницы)

```
10        Librosa     0.10.1
11      AudioRead      3.0.1
12        IPython     8.18.1
13    PyMediaInfo      6.1.0
14       Requests     2.31.0
15     JupyterLab      4.0.9
16           LIWC      0.5.0
17   Transformers     4.36.0
18  Sentencepiece     0.1.99
19          Torch  2.0.1+cpu
20     Torchaudio  2.0.2+cpu
```

— Время выполнения: 0.005 сек. —

Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам

- _b5.video_model_hc_ - Нейросетевая модель **tf.keras.Model** для получения оценок по экспертным признакам

[5]:
```
res_load_video_model_hc = _b5.load_video_model_hc(
    lang = 'en',
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 21:05:26] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (видео модальность) ...

— Время выполнения: 0.321 сек. —

Загрузка весов нейросетевой модели для получения оценок по экспертным признакам

- _b5.video_model_hc_ - Нейросетевая модель **tf.keras.Model** для получения оценок по экспертным признакам

[6]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video']['fi']['hc']['sberdisk']

res_load_video_model_weights_hc = _b5.load_video_model_weights_hc(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 21:05:27] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (видео модальность) ...
[2023-12-14 21:05:27] Загрузка файла "weights_2022-08-27_18-53-35.h5" 100.0% ...
— Время выполнения: 0.249 сек. —

Формирование нейросетевой архитектуры для получения нейросетевых признаков

- `_b5.video_model_deep_fe_` - Нейросетевая модель **tf.keras.Model** для получения нейросетевых признаков

[7]:
```
res_load_video_model_deep_fe = _b5.load_video_model_deep_fe(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 21:05:27] Формирование нейросетевой архитектуры для получения нейросетевых признаков (видео модальность) ...
— Время выполнения: 0.823 сек. —

Загрузка весов нейросетевой модели для получения нейросетевых признаков

- `_b5.video_model_deep_fe_` - Нейросетевая модель **tf.keras.Model** для получения нейросетевых признаков

[8]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video']['fi']['fe']['sberdisk']

res_load_video_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 21:05:28] Загрузка весов нейросетевой модели для получения нейросетевых признаков (видео модальность) ...
[2023-12-14 21:05:31] Загрузка файла "weights_2022-11-01_12-27-07.h5" 100.0% ...
— Время выполнения: 3.342 сек. —

Формирование нейросетевой архитектуры модели для получения оценок по нейросетевым признакам

- _b5.video_model_nn_ - Нейросетевая модель **tf.keras.Model** для получения оценок по нейросетевым признакам

[9]:
```
res_load_video_model_nn = _b5.load_video_model_nn(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 21:05:31] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (видео модальность) ...

— Время выполнения: 0.708 сек. —

Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам

- _b5.video_model_nn_ - Нейросетевая модель **tf.keras.Model** для получения оценок по нейросетевым признакам

[10]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video']['fi']['nn']['sberdisk']

res_load_video_model_weights_nn = _b5.load_video_model_weights_nn(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 21:05:32] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (видео модальность) ...

[2023-12-14 21:05:32] Загрузка файла “weights_2022-03-22_16-31-48.h5”

— Время выполнения: 0.209 сек. —

Формирование нейросетевых архитектур моделей для получения результатов оценки персональных качеств

- _b5.video_models_b5_ - Нейросетевые модели **tf.keras.Model** для получения результатов оценки персональных качеств

[11]:
```
res_load_video_models_b5 = _b5.load_video_models_b5(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 21:05:32] Формирование нейросетевых архитектур моделей для получения результатов оценки персональных качеств (видео модальность) ...

— Время выполнения: 0.046 сек. —

Загрузка весов нейросетевых моделей для получения результатов оценки персональных качеств

- _b5.video_models_b5_ - Нейросетевые модели **tf.keras.Model** для получения результатов оценки персональных качеств

[12]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url_openness = _b5.weights_for_big5_['video']['fi']['b5']['openness']['sberdisk']
url_conscientiousness = _b5.weights_for_big5_['video']['fi']['b5']['conscientiousness']['sberdisk']
url_extraversion = _b5.weights_for_big5_['video']['fi']['b5']['extraversion']['sberdisk']
url_agreeableness = _b5.weights_for_big5_['video']['fi']['b5']['agreeableness']['sberdisk']
url_non_neuroticism = _b5.weights_for_big5_['video']['fi']['b5']['non_neuroticism']['sberdisk']

res_load_video_models_weights_b5 = _b5.load_video_models_weights_b5(
    url_openness = url_openness, # Открытость опыту
    url_conscientiousness = url_conscientiousness, # Добросовестность
    url_extraversion = url_extraversion, # Экстраверсия
    url_agreeableness = url_agreeableness, # Доброжелательность
    url_non_neuroticism = url_non_neuroticism, # Нейротизм
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 21:05:32] Загрузка весов нейросетевых моделей для получения результатов оценки персональных качеств (видео модальность) ...

[2023-12-14 21:05:32] Загрузка файла "weights_2022-06-15_16-46-30.h5" Открытость опыту

[2023-12-14 21:05:32] Загрузка файла “weights_2022-06-15_16-48-50.h5” Добросовестность
[2023-12-14 21:05:33] Загрузка файла “weights_2022-06-15_16-54-06.h5” Экстраверсия
[2023-12-14 21:05:33] Загрузка файла “weights_2022-06-15_17-02-03.h5” Доброжелательность
[2023-12-14 21:05:33] Загрузка файла “weights_2022-06-15_17-06-15.h5” Эмоциональная стабильность
— Время выполнения: 1.013 сек. —

Получение прогнозов (видео модальность)

- `_b5.df_files_` - DataFrame с данными
- `_b5.df_accuracy_` - DataFrame с результатами вычисления точности

[13]:
```
# Настройки ядра
_b5.path_to_dataset_ = 'E:/Databases/FirstImpressionsV2/test' # Директория набора данных
# Директории не входящие в выборку
_b5.ignore_dirs_ = []
# Названия ключей для DataFrame набора данных
_b5.keys_dataset_ = ['Path', 'Openness', 'Conscientiousness', 'Extraversion',
↪'Agreeableness', 'Non-Neuroticism']
_b5.ext_ = ['.mp4'] # Расширения искомых файлов
_b5.path_to_logs_ = './logs' # Директория для сохранения LOG файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_['fi']['sberdisk']

res_get_video_union_predictions = _b5.get_video_union_predictions(
    depth = 1,          # Глубина иерархии для получения аудио и видеоданных
    recursive = False, # Рекурсивный поиск данных
    reduction_fps = 5, # Понижение кадровой частоты
    window = 10,        # Размер окна сегмента сигнала (в секундах)
    step = 5,           # Шаг сдвига окна сегмента сигнала (в секундах)
    lang = 'en',
    accuracy = True,    # Вычисление точности
    url_accuracy = url_accuracy,
    logs = True,        # При необходимости формировать LOG файл
    out = True,         # Отображение
    runtime = True,     # Подсчет времени выполнения
    run = True          # Блокировка выполнения
)
```

[2023-12-14 22:24:55] Получение прогнозов и вычисление точности (видео модальность)

```
...
```

2000 из 2000 (100.0%) ... test80_25_Q4wOgixh7E.004.mp4 ...

```
                                                Path  Openness  \
ID
1   E:\Databases\FirstImpressionsV2\test\test80_01...  0.526971
2   E:\Databases\FirstImpressionsV2\test\test80_01...  0.559385
3   E:\Databases\FirstImpressionsV2\test\test80_01...  0.466969
4   E:\Databases\FirstImpressionsV2\test\test80_01...  0.626113
```

(continues on next page)

(продолжение с предыдущей страницы)

```
5   E:\Databases\FirstImpressionsV2\test\test80_01...    0.5925
6   E:\Databases\FirstImpressionsV2\test\test80_01...  0.671855
7   E:\Databases\FirstImpressionsV2\test\test80_01...  0.411555
8   E:\Databases\FirstImpressionsV2\test\test80_01...  0.583696
9   E:\Databases\FirstImpressionsV2\test\test80_01...  0.551353
10  E:\Databases\FirstImpressionsV2\test\test80_01...  0.575084
11  E:\Databases\FirstImpressionsV2\test\test80_01...  0.559182
12  E:\Databases\FirstImpressionsV2\test\test80_01...   0.50948
13  E:\Databases\FirstImpressionsV2\test\test80_01...  0.330026
14  E:\Databases\FirstImpressionsV2\test\test80_01...  0.649351
15  E:\Databases\FirstImpressionsV2\test\test80_01...  0.651914
16  E:\Databases\FirstImpressionsV2\test\test80_01...  0.523986
17  E:\Databases\FirstImpressionsV2\test\test80_01...  0.575113
18  E:\Databases\FirstImpressionsV2\test\test80_01...  0.566349
19  E:\Databases\FirstImpressionsV2\test\test80_01...  0.672282
20  E:\Databases\FirstImpressionsV2\test\test80_01...  0.684442
21  E:\Databases\FirstImpressionsV2\test\test80_01...  0.550788
22  E:\Databases\FirstImpressionsV2\test\test80_01...  0.525446
23  E:\Databases\FirstImpressionsV2\test\test80_01...  0.473489
24  E:\Databases\FirstImpressionsV2\test\test80_01...  0.667829
25  E:\Databases\FirstImpressionsV2\test\test80_01...  0.469207
26  E:\Databases\FirstImpressionsV2\test\test80_01...  0.625514
27  E:\Databases\FirstImpressionsV2\test\test80_01...  0.568821
28  E:\Databases\FirstImpressionsV2\test\test80_01...  0.696397
29  E:\Databases\FirstImpressionsV2\test\test80_01...  0.578405
30  E:\Databases\FirstImpressionsV2\test\test80_01...  0.637576

   Conscientiousness Extraversion Agreeableness Non-Neuroticism
ID
1           0.460063     0.422793      0.502726        0.450519
2           0.432843     0.504231      0.578673        0.513424
3            0.51701     0.331863      0.451395        0.406188
4           0.597363     0.564068      0.574056        0.589245
5           0.507246     0.505394      0.585405        0.493066
6           0.642559     0.614689      0.613508        0.619511
7           0.394029     0.329323      0.488684         0.39105
8           0.568682     0.505574      0.625314        0.587337
9           0.450333     0.449763      0.495501        0.438009
10          0.517972      0.46315      0.582468        0.537961
11          0.398618     0.433806      0.480592        0.492383
12          0.432549       0.3319      0.495221        0.486891
13          0.322635     0.235595      0.369766         0.25056
14          0.550074     0.502858      0.526621        0.566755
15           0.83048     0.535514      0.695223        0.734383
16          0.435594     0.382946       0.41001        0.466265
17          0.678301     0.468646      0.602139        0.626021
18          0.558975     0.462116      0.606252        0.569516
19            0.6552     0.656699      0.627328        0.663199
20          0.602593     0.680469      0.635343        0.652304
21          0.492015     0.404885      0.562745        0.478233
22          0.469039     0.428517      0.491442         0.45359
23          0.442729     0.353017      0.447929        0.358706
```

(continues on next page)

(продолжение с предыдущей страницы)

```
24          0.655159     0.603695      0.630121        0.614812
25          0.594029     0.364701      0.522734        0.481228
26          0.641622     0.514204      0.547718         0.54766
27          0.524382     0.475687      0.520644        0.531275
28          0.665074      0.70902      0.655993        0.689747
29          0.577321     0.487293      0.557221         0.52153
30          0.587702     0.614512      0.637398        0.613861
```

[2023-12-14 22:24:55] Точность по отдельным персональным качествам личности человека ...

```
           Openness  Conscientiousness  Extraversion  Agreeableness  \
Metrics
MAE          0.0873              0.082        0.0805          0.087
Accuracy     0.9127              0.918        0.9195          0.913

           Non-Neuroticism    Mean
Metrics
MAE                 0.0872  0.0848
Accuracy            0.9128  0.9152
```

[2023-12-14 22:24:55] Средняя средних абсолютных ошибок: 0.0848, средняя точность: 0.9152 ...

Лог файлы успешно сохранены ...

— Время выполнения: 4762.254 сек. —

Обработка текстовой информации

Формирование нейросетевой архитектуры модели и загрузка ее весов для получения признаков / оценок на базе экспертных признаков (текстовая модальность)

- `_b5.text_model_hc_` - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе экспертных признаков

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-10 16:53:54] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a5 Лицензия: BSD License

Формирование нейросетевой архитектуры модели (FI V2)

[4]:
```
res_load_text_model_hc_fi = _b5.load_text_model_hc(
    corpus = "fi", # Корпус для тестирования нейросетевой модели
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:53:55] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (текстовая модальность) ...

— Время выполнения: 1.886 сек. —

Загрузка весов нейросетевой модели

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['text']['fi']['hc']['sberdisk']

res_load_text_model_weights_hc_fi = _b5.load_text_model_weights_hc(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:54:00] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (текстовая модальность) ...

[2023-12-10 16:54:01] Загрузка файла “weights_2023-07-15_10-52-15.h5” 100.0% ...

— Время выполнения: 0.311 сек. —

Отображение сформированной нейросетевой архитектуры модели

[6]:
```
_b5.text_model_hc_.summary()
```

```
Model: "model"
_________________________________________________________________________________________
 Layer (type)                    Output Shape         Param #     Connected to
=========================================================================================
 model_hc/input (InputLayer)     [(None, 89, 64)]     0           []

 model_hc/bilstm_1 (Bidirection  (None, 89, 64)       24832       ['model_hc/input[0][0]']
 al)

 model_hc/dence_2 (Dense)        (None, 89, 64)       4160        ['model_hc/input[0][0]']

 model_hc/attention (Attention)  (None, 89, 64)       0           ['model_hc/bilstm_
↪1[0][0]',
                                                                  'model_hc/bilstm_
↪1[0][0]']

 model_hc/bilstm_2 (Bidirection  (None, 89, 64)       24832       ['model_hc/dence_2[0][0]
↪']
 al)

 add (Add)                       (None, 89, 64)       0           ['model_hc/bilstm_
↪1[0][0]',
                                                                  'model_hc/
↪attention[0][0]',
                                                                  'model_hc/bilstm_
↪2[0][0]']

 model_hc/add (Addition)         (None, 128)          0           ['add[0][0]']

 dense (Dense)                   (None, 5)            645         ['model_hc/add[0][0]']

=========================================================================================
Total params: 54,469
Trainable params: 54,469
Non-trainable params: 0
_________________________________________________________________________________________
```

Формирование нейросетевой архитектуры модели (MuPTA)

[7]:
```
res_load_text_model_hc_mupta = _b5.load_text_model_hc(
    corpus = "mupta", # Корпус для тестирования нейросетевой модели
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:54:06] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (текстовая модальность) ...

— Время выполнения: 0.577 сек. —

Загрузка весов нейросетевой модели

[8]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['text']['mupta']['hc']['sberdisk']

res_load_text_model_weights_hc_mupta = _b5.load_text_model_weights_hc(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:54:19] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (текстовая модальность) ...

[2023-12-10 16:54:19] Загрузка файла “weights_2023-07-15_10-53-38.h5” 100.0% ...

— Время выполнения: 0.264 сек. —

Отображение сформированной нейросетевой архитектуры модели

[9]:
```
_b5.text_model_hc_.summary()
```

```
Model: "model_1"
_________________________________________________________________________________________
↪_________
 Layer (type)                   Output Shape         Param #     Connected to
=========================================================================================================
 model_hc/input (InputLayer)    [(None, 365, 64)]    0           []

 model_hc/bilstm_1 (Bidirection  (None, 365, 64)     24832       ['model_hc/input[0][0]']
 al)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 model_hc/dence_2 (Dense)       (None, 365, 64)      4160        ['model_hc/input[0][0]']

 model_hc/attention (Attention)  (None, 365, 64)     0           ['model_hc/bilstm_
↪1[0][0]',
                                                                  'model_hc/bilstm_
↪1[0][0]']

 model_hc/bilstm_2 (Bidirection  (None, 365, 64)     24832       ['model_hc/dence_2[0][0]
↪']
 al)

 add_1 (Add)                    (None, 365, 64)      0           ['model_hc/bilstm_
↪1[0][0]',
                                                                  'model_hc/
↪attention[0][0]',
                                                                  'model_hc/bilstm_
↪2[0][0]']

 model_hc/add (Addition)        (None, 128)          0           ['add_1[0][0]']

 dense_1 (Dense)                (None, 5)            645         ['model_hc/add[0][0]']

==================================================================================================
Total params: 54,469
Trainable params: 54,469
Non-trainable params: 0
--------------------------------------------------------------------------------------------------
↪---------
```

Формирование нейросетевой архитектуры модели и загрузка ее весов для получения признаков / оценок на базе нейросетевых признаков (текстовая модальность)

- `_b5.text_model_nn_` - Нейросетевая модель **tf.keras.Model** для признаков / оценок на базе нейросетевых признаков

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-10 16:55:37] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a5 Лицензия: BSD License

Формирование нейросетевой архитектуры модели (FI V2)

[4]:
```
res_load_text_model_nn_fi = _b5.load_text_model_nn(
    corpus = "fi", # Корпус для тестирования нейросетевой модели
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:55:40] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (текстовая модальность) ...

— Время выполнения: 1.03 сек. —

Загрузка весов нейросетевой модели

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['text']['fi']['nn']['sberdisk']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
res_load_text_model_weights_nn_fi = _b5.load_text_model_weights_nn(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:55:45] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (текстовая модальность) ...

[2023-12-10 16:55:45] Загрузка файла "weights_2023-07-03_15-01-08.h5" 100.0% ...

— Время выполнения: 0.393 сек. —

Отображение сформированной нейросетевой архитектуры модели

[6]:
```
_b5.text_model_nn_.summary()
```

```
Model: "model"
_________________________________________________________________________________________
↪_________
 Layer (type)                   Output Shape         Param #     Connected to
=========================================================================================
↪========
 model_nn/input (InputLayer)    [(None, 104, 768)]   0           []

 model_nn/bilstm_1 (Bidirection  (None, 104, 64)     205056      ['model_nn/input[0][0]']
 al)

 model_nn/attention (Attention)  (None, 104, 64)     0           ['model_nn/bilstm_
↪1[0][0]',
                                                                  'model_nn/bilstm_
↪1[0][0]']

 model_nn/dence_2 (Dense)       (None, 104, 128)     8320        ['model_nn/
↪attention[0][0]']

 model_nn/add (Addition)        (None, 256)          0           ['model_nn/dence_2[0][0]
↪']

 model_nn/dence_3 (Dense)       (None, 128)          32896       ['model_nn/add[0][0]']

 dense (Dense)                  (None, 5)            645         ['model_nn/dence_3[0][0]
↪']

=========================================================================================
↪========
Total params: 246,917
Trainable params: 246,917
Non-trainable params: 0
_________________________________________________________________________________________
↪_________
```

Формирование нейросетевой архитектуры модели (MuPTA)

[7]:
```
res_load_text_model_nn_mupta = _b5.load_text_model_nn(
    corpus = "mupta", # Корпус для тестирования нейросетевой модели
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:55:49] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (текстовая модальность) ...

— Время выполнения: 0.264 сек. —

Загрузка весов нейросетевой модели

[8]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['text']['mupta']['nn']['sberdisk']

res_load_text_model_weights_nn_mupta = _b5.load_text_model_weights_nn(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 16:55:51] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (текстовая модальность) ...

[2023-12-10 16:55:52] Загрузка файла “weights_2023-07-16_18-12-01.h5” 100.0% ...

— Время выполнения: 0.373 сек. —

Отображение сформированной нейросетевой архитектуры модели

[9]:
```
_b5.text_model_nn_.summary()
```

```
Model: "model_1"
_________________________________________________________________________________________
↪_________
 Layer (type)                Output Shape              Param #     Connected to
=========================================================================================================
 model_nn/input (InputLayer)    [(None, 414, 768)]   0           []

 model_nn/bilstm_1 (Bidirection  (None, 414, 64)     205056      ['model_nn/input[0][0]']
 al)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 model_nn/attention (Attention)  (None, 414, 64)     0           ['model_nn/bilstm_
↪1[0][0]',
                                                                  'model_nn/bilstm_
↪1[0][0]']

 model_nn/dence_2 (Dense)        (None, 414, 128)    8320        ['model_nn/
↪attention[0][0]']

 model_nn/add (Addition)         (None, 256)         0           ['model_nn/dence_2[0][0]
↪']

 model_nn/dence_3 (Dense)        (None, 128)         32896       ['model_nn/add[0][0]']

 dense_1 (Dense)                 (None, 5)           645         ['model_nn/dence_3[0][0]
↪']

==================================================================================================
Total params: 246,917
Trainable params: 246,917
Non-trainable params: 0
--------------------------------------------------------------------------------------------------
↪---------
```

Формирование нейросетевых архитектур моделей и загрузка их весов для получения оценок персональных качеств (текстовая модальность)

- `_b5.text_model_b5_` - Нейросетевая модель **tf.keras.Model** для получения оценок персональных качеств

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-10 17:03:46] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a5 Лицензия: BSD License

Формирование нейросетевых архитектур моделей

[4]:
```
res_load_text_model_b5 = _b5.load_text_model_b5(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 17:03:46] Формирование нейросетевой архитектуры модели для получения оценок персональных качеств (текстовая модальность) ...

— Время выполнения: 0.539 сек. —

Загрузка весов нейросетевых моделей

```
FI V2
```

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['text']['fi']['b5']['sberdisk']

res_load_text_model_weights_b5 = _b5.load_text_model_weights_b5(
    url = url,
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели␣
↪из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 17:03:46] Загрузка весов нейросетевой модели для получения оценок персональных качеств (текстовая модальность) ...

[2023-12-10 17:03:47] Загрузка файла “ft_fi_2023-12-09_14-25-13.h5”

— Время выполнения: 0.144 сек. —

MuPTA

[6]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['text']['mupta']['b5']['sberdisk']

res_load_text_model_weights_b5 = _b5.load_text_model_weights_b5(
    url = url,
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели␣
↪из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-10 17:03:47] Загрузка весов нейросетевой модели для получения оценок персональных качеств (текстовая модальность) ...

[2023-12-10 17:03:47] Загрузка файла "ft_mupta_2023-12-09_14-25-13.h5"

— Время выполнения: 0.137 сек. —

Отображение сформированной нейросетевой архитектуры модели

[7]:
```
_b5.text_model_b5_.summary()
```

```
Model: "model"
__________________________________________________________________________________________
↪_________
 Layer (type)                   Output Shape         Param #     Connected to
==========================================================================================
↪=========
 input_1 (InputLayer)           [(None, 5)]          0           []

 input_2 (InputLayer)           [(None, 5)]          0           []

 tf.concat (TFOpLambda)         (None, 10)           0           ['input_1[0][0]',
                                                                  'input_2[0][0]']

 dense (Dense)                  (None, 5)            55          ['tf.concat[0][0]']

==========================================================================================
↪=========
Total params: 55
Trainable params: 55
Non-trainable params: 0
__________________________________________________________________________________________
↪_________
```

Извлечение признаков из текстового сигнала

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

```
2023-12-03 00:29:47.655916: I tensorflow/core/platform/cpu_feature_guard.cc:193] This␣
↪TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use␣
↪the following CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler␣
↪flags.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#FFF', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-03 00:29:57] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a5 Лицензия: BSD License

Загрузка словаря с экспертными признаками

[4]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

res_load_text_features = _b5.load_text_features(
    force_reload = True, # Принудительная загрузка файла
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-03 00:29:57] Загрузка словаря с экспертными признаками ...

[2023-12-03 00:30:00] Загрузка файла "LIWC2007.txt" 100.0% ...

— Время выполнения: 3.073 сек. —

Формирование токенизатора и нейросетевой модели машинного перевода (RU -> EN)

[5]:
```
res_setup_translation_model = _b5.setup_translation_model(
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-03 00:30:00] Формирование токенизатора и нейросетевой модели машинного перевода ...

— Время выполнения: 3.098 сек. —

Формирование токенизатора и нейросетевой модели BERT (для кодирования слов)

[6]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

res_setup_translation_model = _b5.setup_bert_encoder(
    force_reload = False, # Принудительная загрузка файла
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-03 00:30:04] Формирование токенизатора и нейросетевой модели BERT ...

[2023-12-03 00:30:07] Загрузка файла “bert-base-multilingual-cased.zip”

[2023-12-03 00:30:04] Формирование токенизатора и нейросетевой модели BERT ...

[2023-12-03 00:30:07] Загрузка файла “bert-base-multilingual-cased.zip”

[2023-12-03 00:30:07] Разархивирование архива “bert-base-multilingual-cased.zip” ...

— Время выполнения: 14.752 сек. —

Процесс извлечения признаков из текста

Пример 1 (Анализ видефайла (EN) с учетом ручной транскрипции)

[7]:
```
# Путь к видеофайлу
path = '/Users/dl/GitHub/OCEANAI/docs/source/user_guide/notebooks/glgfB3vFewc.004.mp4'

hc_features, nn_features = _b5.get_text_features(
    path = path, # Путь к видеофайлу
    asr = False, # Распознавание речи
    lang = 'en', # Выбор языка
    show_text = True, # Отображение текста
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-03 00:30:18] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-03 00:30:19] Статистика извлеченных признаков из текста: Размерность матрицы экспертных признаков: 89 × 64 Размерность матрицы с нейросетевыми признаками: 104 × 768 Текст: during those times i feel sad i feel confused and

— Время выполнения: 0.343 сек. —

Пример 2 (Анализ видефайла (EN) без учета ручной транскрипции)

[8]:
```
# Путь к видеофайлу
path = '/Users/dl/GitHub/OCEANAI/docs/source/user_guide/notebooks/glgfB3vFewc.004.mp4'

hc_features, nn_features = _b5.get_text_features(
    path = path, # Путь к видеофайлу
    asr = True, # Распознавание речи
    lang = 'en', # Выбор языка
    show_text = True, # Отображение текста
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-03 00:30:19] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-03 00:30:25] Статистика извлеченных признаков из текста: Размерность матрицы экспертных признаков: 89 × 64 Размерность матрицы с нейросетевыми признаками: 104 × 768 Текст: during those times i feel sad i feel confused and- the school and introduce them to our administrators and the different faculty that work throughout the school and the library and the gym and so on and then they can get comfortable if theyre in a new school as well

— Время выполнения: 6.398 сек. —

Пример 3 (Анализ видефайла (RU) без учета ручной транскрипции)

[9]:
```
# Путь к текстовому файлу
path = '/Users/dl/GitHub/OCEANAI/docs/source/user_guide/notebooks/center_42.mov'

hc_features, nn_features = _b5.get_text_features(
    path = path, # Путь к видеофайлу
    asr = False, # Распознавание речи
    lang = 'ru', # Выбор языка
    show_text = True, # Отображение текста
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-03 00:30:25] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-03 00:30:43] Статистика извлеченных признаков из текста: Размерность матрицы экспертных признаков: 365 × 64 Размерность матрицы с нейросетевыми признаками: 414 × 768
Текст: на картинке изображены скорее всего друзья которые играют в груз мечом это скорее всего происходит где-то в америке возможно в калифорнии на пляже девушка в топе и в шортах пытается словить мяч также двое парней смотрят одинаково думает как перехватить следующую подачу меча на заднем фоне видны высокие пальмы стоят дома неба голубое песок чистой чётко написки отображаются силой этой людей у парня в дали одеты солнце защитные очки он также в шортах и в майке в близи не видно головы человека он одет в темные шорты и в серую фортболку

— Время выполнения: 18.045 сек. —

Пример 4 (Анализ текста - RU)

[10]:
```
# Текст
path = '''
На картинке изображены скорее всего друзья, которые играют в игру с мячом.
Это скорее всего происходит где-то в Америке, возможно, в Калифорнии на пляже.
Девушка в топе и в шортах пытается словить мяч. Также двое парней смотрят, один активно␣
↪думает,
как перехватить следующую подачу мяча. На заднем фоне видны высокие пальмы. Стоят дома.
Небо голубое. Песок чистый. Чётко на песке отображаются силуэты людей. У парня вдали␣
↪одеты солнцезащитные очки,
он также в шортах и в майке. Вблизи не видно головы человека. Он одет в тёмные шорты и в␣
↪серую футболку.
'''

hc_features, nn_features = _b5.get_text_features(
    path = path, # Текст
    asr = False, # Распознавание речи
    lang = 'ru', # Выбор языка
    show_text = True, # Отображение текста
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-03 00:30:43] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-03 00:30:52] Статистика извлеченных признаков из текста: Размерность матрицы экспертных признаков: 365 × 64 Размерность матрицы с нейросетевыми признаками: 414 × 768
Текст: на картинке изображены скорее всего друзья которые играют в игру с мячом это скорее всего происходит где-то в америке возможно в калифорнии на пляже девушка в топе и в шортах пытается словить мяч также двое парней смотрят один активно думает как перехватить следующую подачу мяча на заднем фоне видны высокие пальмы стоят дома небо голубое песок чистый чётко на песке отображаются силуэты людей у парня вдали одеты солнцезащитные очки он также в шортах и в майке вблизи не видно головы человека он одет в тёмные шорты и в серую футболку

— Время выполнения: 9.227 сек. —

Пример 5 (Анализ текста - EN)

[11]:
```
# Текст
path = '''
today says they to for that but right now i am just watching super girl a new images be␣
↪catching up
and some shows a good say you guys
'''

hc_features, nn_features = _b5.get_text_features(
    path = path, # Текст
    asr = False, # Распознавание речи
    lang = 'en', # Выбор языка
    show_text = True, # Отображение текста
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-03 00:30:52] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-03 00:30:53] Статистика извлеченных признаков из текста: Размерность матрицы экспертных признаков: 89 × 64 Размерность матрицы с нейросетевыми признаками: 104 × 768
Текст: today says they to for that but right now i am just watching super girl a new images be catching up and some shows a good say you guys

— Время выполнения: 0.247 сек. —

Пример 6 (Анализ текстового файла - EN)

[12]:
```
# Текст
path = '/Users/dl/GitHub/OCEANAI/docs/source/user_guide/notebooks/glgfB3vFewc.004.txt'

hc_features, nn_features = _b5.get_text_features(
    path = path, # Текст
    asr = False, # Распознавание речи
    lang = 'en', # Выбор языка
    show_text = True, # Отображение текста
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-03 00:30:53] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-03 00:30:53] Статистика извлеченных признаков из текста: Размерность матрицы экспертных признаков: 89 × 64 Размерность матрицы с нейросетевыми признаками: 104 × 768
Текст: during those times i feel sad i feel confused and

— Время выполнения: 0.204 сек. —

Получение прогнозов по тексту

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-14 18:07:43] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a16 Лицензия: BSD License

Получение и отображение версий установленных библиотек

- `_b5.df_pkgs_` - DataFrame с версиями установленных библиотек

[4]:
```
_b5.libs_vers(runtime = True, run = True)
```

	Package	Version
1	TensorFlow	2.15.0
2	Keras	2.15.0
3	OpenCV	4.8.1
4	MediaPipe	0.9.0
5	NumPy	1.26.2
6	SciPy	1.11.4
7	Pandas	2.1.3
8	Scikit-learn	1.3.2
9	OpenSmile	2.5.0
10	Librosa	0.10.1
11	AudioRead	3.0.1
12	IPython	8.18.1
13	PyMediaInfo	6.1.0

(continues on next page)

(продолжение с предыдущей страницы)

```
14       Requests     2.31.0
15     JupyterLab      4.0.9
16           LIWC      0.5.0
17   Transformers     4.36.0
18  Sentencepiece     0.1.99
19          Torch  2.0.1+cpu
20     Torchaudio  2.0.2+cpu
```

— Время выполнения: 0.006 сек. —

Загрузка словаря с экспертными признаками

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

res_load_text_features = _b5.load_text_features(
    force_reload = True, # Принудительная загрузка файла
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 18:07:43] Загрузка словаря с экспертными признаками ...

[2023-12-14 18:07:43] Загрузка файла “LIWC2007.txt” 100.0% ...

— Время выполнения: 0.232 сек. —

Формирование токенизатора и нейросетевой модели машинного перевода (RU -> EN)

[6]:
```
res_setup_translation_model = _b5.setup_translation_model(
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 18:07:43] Формирование токенизатора и нейросетевой модели машинного перевода ...

— Время выполнения: 1.71 сек. —

Формирование токенизатора и нейросетевой модели BERT (для кодирования слов)

[7]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

res_setup_translation_model = _b5.setup_bert_encoder(
    force_reload = False, # Принудительная загрузка файла
    out = True, # Отображение
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 18:07:45] Формирование токенизатора и нейросетевой модели BERT ...
[2023-12-14 18:07:47] Загрузка файла "bert-base-multilingual-cased.zip"
[2023-12-14 18:07:45] Формирование токенизатора и нейросетевой модели BERT ...
[2023-12-14 18:07:47] Загрузка файла "bert-base-multilingual-cased.zip"
[2023-12-14 18:07:47] Разархивирование архива "bert-base-multilingual-cased.zip" ...
— Время выполнения: 4.188 сек. —

FI V2

Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам

- `_b5.text_model_hc_` - Нейросетевая модель **tf.keras.Model** для получения оценок по экспертным признакам

[8]:
```
res_load_text_model_hc_fi = _b5.load_text_model_hc(
    corpus = "fi", # Корпус для тестирования нейросетевой модели
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 18:07:49] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (текстовая модальность) ...
— Время выполнения: 0.647 сек. —

Загрузка весов нейросетевой модели для получения оценок по экспертным признакам

- `_b5.text_model_hc_` - Нейросетевая модель **tf.keras.Model** для получения оценок по экспертным признакам

[9]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['text']['fi']['hc']['sberdisk']

res_load_text_model_weights_hc_fi = _b5.load_text_model_weights_hc(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    run = True # Блокировка выполнения
)
```

[2023-12-14 18:07:50] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (текстовая модальность) ...

[2023-12-14 18:07:50] Загрузка файла "weights_2023-07-15_10-52-15.h5" 100.0% ...

— Время выполнения: 0.289 сек. —

Формирование нейросетевой архитектуры модели для получения оценок по нейросетевым признакам

- _b5.text_model_nn_ - Нейросетевая модель **tf.keras.Model** для получения оценок по нейросетевым признакам

[10]:
```
res_load_text_model_nn_fi = _b5.load_text_model_nn(
    corpus = "fi", # Корпус для тестирования нейросетевой модели
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 18:07:50] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (текстовая модальность) ...

— Время выполнения: 0.279 сек. —

Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам

- _b5.text_model_nn_ - Нейросетевая модель **tf.keras.Model** для получения оценок по нейросетевым признакам

[11]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['text']['fi']['nn']['sberdisk']

res_load_text_model_weights_nn_fi = _b5.load_text_model_weights_nn(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 18:07:50] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (текстовая модальность) ...

[2023-12-14 18:07:51] Загрузка файла "weights_2023-07-03_15-01-08.h5" 100.0% ...

— Время выполнения: 0.337 сек. —

Формирование нейросетевой архитектуры модели для получения оценок персональных качеств

- _b5.text_model_b5_ - Нейросетевая модель **tf.keras.Model** для получения оценок персональных качеств

[12]:
```
res_load_text_model_b5 = _b5.load_text_model_b5(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 18:07:51] Формирование нейросетевой архитектуры модели для получения оценок персональных качеств (текстовая модальность) ...

— Время выполнения: 0.015 сек. —

Загрузка весов нейросетевой модели для получения оценок персональных качеств

- _b5.text_model_b5_ - Нейросетевая модель **tf.keras.Model** для получения оценок персональных качеств

[13]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['text']['fi']['b5']['sberdisk']

res_load_text_model_weights_b5 = _b5.load_text_model_weights_b5(
    url = url,
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели
↪из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 18:07:51] Загрузка весов нейросетевой модели для получения оценок персональных качеств (текстовая модальность) ...

[2023-12-14 18:07:51] Загрузка файла “ft_fi_2023-12-09_14-25-13.h5”

— Время выполнения: 0.163 сек. —

Получение прогнозов (текстовая модальность)

- _b5.df_files_ - DataFrame с данными
- _b5.df_accuracy_ - DataFrame с результатами вычисления точности

[14]:
```
# Настройки ядра
_b5.path_to_dataset_ = 'E:/Databases/FirstImpressionsV2/test' # Директория набора данных
# Директории не входящие в выборку
_b5.ignore_dirs_ = []
# Названия ключей для DataFrame набора данных
_b5.keys_dataset_ = ['Path', 'Openness', 'Conscientiousness', 'Extraversion',
↪'Agreeableness', 'Non-Neuroticism']
_b5.ext_ = ['.mp4'] # Расширения искомых файлов
_b5.path_to_logs_ = './logs' # Директория для сохранения LOG файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_['fi']['sberdisk']

res_get_text_union_predictions = _b5.get_text_union_predictions(
    depth = 1,          # Глубина иерархии для получения видеоданных
    recursive = False, # Рекурсивный поиск данных
    asr = True,         # Распознавание речи
    lang = 'en',        # Выбор языка
    accuracy = True,    # Вычисление точности
    url_accuracy = url_accuracy,
    logs = True,        # При необходимости формировать LOG файл
    out = True,         # Отображение
    runtime = True,     # Подсчет времени выполнения
    run = True          # Блокировка выполнения
)
```

[2023-12-14 19:00:14] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-14 19:00:15] Получение прогнозов и вычисление точности (текстовая модальность) ...

2000 из 2000 (100.0%) ... test80_25_Q4wOgixh7E.004.mp4 ...

ID	Path	Openness
1	E:\Databases\FirstImpressionsV2\test\test80_01...	0.624434
2	E:\Databases\FirstImpressionsV2\test\test80_01...	0.518305
3	E:\Databases\FirstImpressionsV2\test\test80_01...	0.516165
4	E:\Databases\FirstImpressionsV2\test\test80_01...	0.653522
5	E:\Databases\FirstImpressionsV2\test\test80_01...	0.672823
6	E:\Databases\FirstImpressionsV2\test\test80_01...	0.571563
7	E:\Databases\FirstImpressionsV2\test\test80_01...	0.579048
8	E:\Databases\FirstImpressionsV2\test\test80_01...	0.547369
9	E:\Databases\FirstImpressionsV2\test\test80_01...	0.630611
10	E:\Databases\FirstImpressionsV2\test\test80_01...	0.643665
11	E:\Databases\FirstImpressionsV2\test\test80_01...	0.610431
12	E:\Databases\FirstImpressionsV2\test\test80_01...	0.501841
13	E:\Databases\FirstImpressionsV2\test\test80_01...	0.516751
14	E:\Databases\FirstImpressionsV2\test\test80_01...	0.625826

(continues on next page)

(продолжение с предыдущей страницы)

```
15  E:\Databases\FirstImpressionsV2\test\test80_01...  0.506065
16  E:\Databases\FirstImpressionsV2\test\test80_01...  0.638552
17  E:\Databases\FirstImpressionsV2\test\test80_01...   0.51764
18  E:\Databases\FirstImpressionsV2\test\test80_01...  0.581101
19  E:\Databases\FirstImpressionsV2\test\test80_01...  0.545621
20  E:\Databases\FirstImpressionsV2\test\test80_01...  0.619155
21  E:\Databases\FirstImpressionsV2\test\test80_01...   0.58491
22  E:\Databases\FirstImpressionsV2\test\test80_01...  0.504319
23  E:\Databases\FirstImpressionsV2\test\test80_01...  0.587255
24  E:\Databases\FirstImpressionsV2\test\test80_01...    0.6448
25  E:\Databases\FirstImpressionsV2\test\test80_01...  0.575514
26  E:\Databases\FirstImpressionsV2\test\test80_01...  0.561977
27  E:\Databases\FirstImpressionsV2\test\test80_01...  0.522762
28  E:\Databases\FirstImpressionsV2\test\test80_01...  0.642535
29  E:\Databases\FirstImpressionsV2\test\test80_01...  0.615789
30  E:\Databases\FirstImpressionsV2\test\test80_01...  0.620333

   Conscientiousness Extraversion Agreeableness Non-Neuroticism
ID
1           0.588915      0.53729      0.601771        0.587032
2           0.405696     0.440837      0.486431         0.42919
3           0.482939     0.419187      0.520959         0.46346
4           0.645953       0.5613       0.63864        0.635908
5           0.563164     0.597474      0.618239        0.627377
6            0.49441     0.477624      0.548336        0.509708
7           0.590844     0.470888      0.580203        0.545247
8           0.540064     0.441378       0.55407         0.52564
9           0.546466     0.548925      0.592785        0.576801
10          0.650126     0.561841       0.63202        0.636658
11          0.509742     0.532337      0.563182        0.548405
12          0.438787     0.408134      0.493867        0.433236
13          0.521908     0.412392      0.535759        0.475492
14          0.595756     0.545166      0.608196        0.601571
15          0.466968     0.428299      0.497129        0.451425
16          0.564402     0.561068      0.599493        0.594701
17          0.588128     0.392461      0.569938        0.512308
18          0.516556     0.489761      0.557651        0.521073
19          0.467661      0.46827      0.518607        0.478676
20          0.529129     0.535892       0.58141        0.571938
21          0.489063     0.500084      0.538159        0.525135
22          0.449576     0.427531      0.488319        0.441239
23          0.591969      0.50329      0.578679        0.566444
24           0.58204     0.558367       0.61345         0.60149
25          0.517498     0.481397      0.548056        0.514953
26          0.594428     0.456222      0.562595        0.536081
27          0.468697     0.426084      0.510566        0.451157
28          0.538425     0.564254      0.602641        0.595872
29           0.54139     0.522493      0.585496        0.570682
30          0.522955     0.543902      0.569043        0.559107
```

[2023-12-14 19:00:16] Точность по отдельным персональным качествам личности человека ...

```
          Openness  Conscientiousness  Extraversion  Agreeableness  \
Metrics
MAE         0.1097              0.114         0.115         0.1019
Accuracy    0.8903              0.886         0.885         0.8981

          Non-Neuroticism    Mean
Metrics
MAE                0.1154  0.1112
Accuracy           0.8846  0.8888
```

[2023-12-14 19:00:16] Средняя средних абсолютных ошибок: 0.1112, средняя точность: 0.8888 ...

Лог файлы успешно сохранены ...

— Время выполнения: 3131.846 сек. —

Мультимодальная обработка информации

Формирование нейросетевых архитектур моделей и загрузка их весов для получения результатов оценки персональных качеств (аудио и видео объединение)

- `_b5.av_models_b5_` - Нейросетевые модели **tf.keras.Model** для получения результатов оценки персональных качеств

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-14 22:44:38] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a16 Лицензия: BSD License

Формирование нейросетевых архитектур моделей

[4]:
```
res_load_av_models_b5 = _b5.load_av_models_b5(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:44:38] Формирование нейросетевых архитектур моделей для получения результатов оценки персональных качеств (мультимодальное объединение) ...

— Время выполнения: 0.095 сек. —

Загрузка весов нейросетевых моделей

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url_openness = _b5.weights_for_big5_['av']['fi']['b5']['openness']['sberdisk']
url_conscientiousness = _b5.weights_for_big5_['av']['fi']['b5']['conscientiousness'][
↪'sberdisk']
url_extraversion = _b5.weights_for_big5_['av']['fi']['b5']['extraversion']['sberdisk']
url_agreeableness = _b5.weights_for_big5_['av']['fi']['b5']['agreeableness']['sberdisk']
url_non_neuroticism = _b5.weights_for_big5_['av']['fi']['b5']['non_neuroticism'][
↪'sberdisk']

res_load_av_models_weights_b5 = _b5.load_av_models_weights_b5(
    url_openness = url_openness, # Открытость опыту
    url_conscientiousness = url_conscientiousness, # Добросовестность
    url_extraversion = url_extraversion, # Экстраверсия
    url_agreeableness = url_agreeableness, # Доброжелательность
    url_non_neuroticism = url_non_neuroticism, # Эмоциональная стабильность
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:44:53] Загрузка весов нейросетевых моделей для получения результатов оценки персональных качеств (мультимодальное объединение) ...

[2023-12-14 22:44:53] Загрузка файла “weights_2022-08-28_11-14-35.h5” 100.0% ...
Открытость опыту

[2023-12-14 22:44:54] Загрузка файла “weights_2022-08-28_11-08-10.h5” 100.0% ...
Добросовестность

[2023-12-14 22:44:54] Загрузка файла “weights_2022-08-28_11-17-57.h5” 100.0% ...
Экстраверсия

[2023-12-14 22:44:54] Загрузка файла “weights_2022-08-28_11-25-11.h5” 100.0% ...
Доброжелательность

[2023-12-14 22:44:54] Загрузка файла “weights_2022-06-14_21-44-09.h5” 100.0% ...
Эмоциональная стабильность
— Время выполнения: 0.914 сек. —

Отображение сформированной нейросетевой архитектуры модели

- `openness` - Открытость опыту
- `conscientiousness` - Добросовестность
- `extraversion` - Экстраверсия
- `agreeableness` - Доброжелательность
- `non_neuroticism` - Эмоциональная стабильность

[6]:
```
_b5.av_models_b5_['openness'].summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 64)]              0

 dense_1 (Dense)             (None, 1)                 65

 activ_1 (Activation)        (None, 1)                 0

=================================================================
Total params: 65 (260.00 Byte)
Trainable params: 65 (260.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Формирование нейросетевой архитектуры модели и загрузка ее весов для получения результатов оценки персональных качеств (аудио, видео и текст объединение)

- `_b5.avt_model_b5_` - Нейросетевая модель **tf.keras.Model** для получения результатов оценки персональных качеств

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-11 09:46:45] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a5 Лицензия: BSD License

Формирование нейросетевой архитектуры модели

[4]:
```
res_load_avt_model_b5 = _b5.load_avt_model_b5(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-11 09:46:45] Формирование нейросетевой архитектуры модели для получения оценок персональных качеств (мультимодальное объединение) ...

— Время выполнения: 0.814 сек. —

Загрузка весов нейросетевой модели

[5]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['avt']['fi']['b5']['sberdisk']

res_load_avt_model_weights_b5 = _b5.load_avt_model_weights_b5(
    url = url,
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели␣
↪из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-11 09:46:46] Загрузка весов нейросетевой модели для получения оценок персональных качеств (мультимодальное объединение) ...
[2023-12-11 09:46:46] Загрузка файла “avt_fi_2023-12-03_11-36-51.h5”
— Время выполнения: 0.218 сек. —

Отображение сформированной нейросетевой архитектуры модели

[6]:
```
_b5.avt_model_b5_.summary()
```

```
Model: "model"
__________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==========================================================================================
 hc_t (InputLayer)              [(None, 128)]        0           []

 hc_a (InputLayer)              [(None, 256)]        0           []

 nn_t (InputLayer)              [(None, 128)]        0           []

 nn_a (InputLayer)              [(None, 512)]        0           []

 hc_v (InputLayer)              [(None, 256)]        0           []

 nn_v (InputLayer)              [(None, 2048)]       0           []

 ln_hc_t (LayerNormalization)   (None, 128)          256         ['hc_t[0][0]']

 ln_hc_a (LayerNormalization)   (None, 256)          512         ['hc_a[0][0]']

 ln_nn_t (LayerNormalization)   (None, 128)          256         ['nn_t[0][0]']

 ln_nn_a (LayerNormalization)   (None, 512)          1024        ['nn_a[0][0]']

 ln_hc_v (LayerNormalization)   (None, 256)          512         ['hc_v[0][0]']

 ln_nn_v (LayerNormalization)   (None, 2048)         4096        ['nn_v[0][0]']

 gata (GFL)                     (None, 64)           131072      ['ln_hc_t[0][0]',
                                                                  'ln_hc_a[0][0]',
                                                                  'ln_nn_t[0][0]',
                                                                  'ln_nn_a[0][0]']

 gatv (GFL)                     (None, 64)           327680      ['ln_hc_t[0][0]',
                                                                  'ln_hc_v[0][0]',
                                                                  'ln_nn_t[0][0]',
                                                                  'ln_nn_v[0][0]']

 gaav (GFL)                     (None, 64)           393216      ['ln_hc_a[0][0]',
                                                                  'ln_hc_v[0][0]',
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                                                          'ln_nn_a[0][0]',
                                                          'ln_nn_v[0][0]']

 tf.concat (TFOpLambda)         (None, 192)          0           ['gata[0][0]',
                                                                  'gatv[0][0]',
                                                                  'gaav[0][0]']

 dense (Dense)                  (None, 50)           9650        ['tf.concat[0][0]']

 dence_cl (Dense)               (None, 5)            255         ['dense[0][0]']

==================================================================================================
Total params: 868,529
Trainable params: 868,529
Non-trainable params: 0
--------------------------------------------------------------------------------------------------
↪---------
```

Мультимодальное объединение для получения прогнозов по аудио и видео FI V2

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

```
[2023-12-14 22:46:31] OCEANAI - персональные качества личности человека:  Авторы:  Рюмина Елена [ryumina_ev@mail.ru]  Рюмин Дмитрий [dl_03.03.1991@mail.ru]  Карпов Алексей [karpov@iias.spb.su]  Сопровождающие:  Рюмина Елена [ryumina_ev@mail.ru]  Рюмин Дмитрий [dl_03.03.1991@mail.ru]  Версия:  1.0.0a16  Лицензия: BSD License
```

Получение и отображение версий установленных библиотек

- `_b5.df_pkgs_` - DataFrame с версиями установленных библиотек

[4]:
```
_b5.libs_vers(runtime = True, run = True)
```

	Package	Version
1	TensorFlow	2.15.0
2	Keras	2.15.0
3	OpenCV	4.8.1
4	MediaPipe	0.9.0
5	NumPy	1.26.2
6	SciPy	1.11.4
7	Pandas	2.1.3
8	Scikit-learn	1.3.2
9	OpenSmile	2.5.0
10	Librosa	0.10.1
11	AudioRead	3.0.1
12	IPython	8.18.1
13	PyMediaInfo	6.1.0
14	Requests	2.31.0
15	JupyterLab	4.0.9
16	LIWC	0.5.0
17	Transformers	4.36.0
18	Sentencepiece	0.1.99
19	Torch	2.0.1+cpu
20	Torchaudio	2.0.2+cpu

— Время выполнения: 0.006 сек. —

Анализ аудио информации (формирование и загрузка весов моделей)

Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (аудио модальность)

- `_b5.audio_model_hc_` - Нейросетевая модель **tf.keras.Model** для получения оценок по экспертным признакам

[5]:
```
res_load_audio_model_hc = _b5.load_audio_model_hc(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:46:31] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (аудио модальность) ...

— Время выполнения: 0.322 сек. —

Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (аудио модальность)

- _b5.audio_model_hc_ - Нейросетевая модель **tf.keras.Model** для получения оценок по экспертным признакам

[6]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['audio']['fi']['hc']['sberdisk']

res_load_audio_model_weights_hc = _b5.load_audio_model_weights_hc(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:46:31] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (аудио модальность) ...

[2023-12-14 22:46:32] Загрузка файла "weights_2022-05-05_11-27-55.h5" 100.0% ...

— Время выполнения: 0.277 сек. —

Формирование нейросетевой архитектуры модели для получения оценок по нейросетевым признакам (аудио модальность)

- _b5.audio_model_nn_ - Нейросетевая модель **tf.keras.Model** для получения оценок по нейросетевым признакам

[7]:
```
res_load_audio_model_nn = _b5.load_audio_model_nn(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:46:32] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (аудио модальность) ...

— Время выполнения: 0.244 сек. —

Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (аудио модальность)

- `_b5.audio_model_nn_` - Нейросетевая модель **tf.keras.Model** для получения оценок по нейросетевым признакам

[8]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['audio']['fi']['nn']['sberdisk']

res_load_audio_model_weights_nn = _b5.load_audio_model_weights_nn(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели␣
↪из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:46:32] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (аудио модальность) ...

[2023-12-14 22:46:32] Загрузка файла "weights_2022-05-03_07-46-14.h5"

— Время выполнения: 0.389 сек. —

Анализ видео информации (формирование и загрузка весов моделей)

Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (видео модальность)

- `_b5.video_model_hc_` - Нейросетевая модель **tf.keras.Model** для получения оценок по экспертным признакам

[9]:
```
res_load_video_model_hc = _b5.load_video_model_hc(
    lang = 'en',
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:46:32] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (видео модальность) ...

— Время выполнения: 0.257 сек. —

Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (видео модальность)

- `_b5.video_model_hc_` - Нейросетевая модель **tf.keras.Model** для получения оценок по экспертным признакам

[10]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video']['fi']['hc']['sberdisk']

res_load_video_model_weights_hc = _b5.load_video_model_weights_hc(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:46:32] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (видео модальность) ...

[2023-12-14 22:46:33] Загрузка файла "weights_2022-08-27_18-53-35.h5" 100.0% ...

— Время выполнения: 0.226 сек. —

Формирование нейросетевой архитектуры для получения нейросетевых признаков (видео модальность)

- `_b5.video_model_deep_fe_` - Нейросетевая модель **tf.keras.Model** для получения нейросетевых признаков

[11]:
```
res_load_video_model_deep_fe = _b5.load_video_model_deep_fe(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:46:34] Формирование нейросетевой архитектуры для получения нейросетевых признаков (видео модальность) ...

— Время выполнения: 0.783 сек. —

Загрузка весов нейросетевой модели для получения нейросетевых признаков (видео модальность)

- `_b5.video_model_deep_fe_` - Нейросетевая модель **tf.keras.Model** для получения нейросетевых признаков

[12]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video']['fi']['fe']['sberdisk']

res_load_video_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:46:35] Загрузка весов нейросетевой модели для получения нейросетевых признаков (видео модальность) ...

[2023-12-14 22:46:40] Загрузка файла "weights_2022-11-01_12-27-07.h5" 100.0% ...

— Время выполнения: 4.311 сек. —

Формирование нейросетевой архитектуры модели для получения оценок по нейросетевым признакам (видео модальность)

- `_b5.video_model_nn_` - Нейросетевая модель **tf.keras.Model** для получения оценок по нейросетевым признакам

[13]:
```
res_load_video_model_nn = _b5.load_video_model_nn(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:46:40] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (видео модальность) ...

— Время выполнения: 0.724 сек. —

Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (видео модальность)

- _b5.video_model_nn_ - Нейросетевая модель **tf.keras.Model** для получения оценок по нейросетевым признакам

[14]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video']['fi']['nn']['sberdisk']

res_load_video_model_weights_nn = _b5.load_video_model_weights_nn(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:46:40] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (видео модальность) ...

[2023-12-14 22:46:42] Загрузка файла "weights_2022-03-22_16-31-48.h5"

— Время выполнения: 1.355 сек. —

Анализ мультимодальной информации (формирование, загрузка весов моделей, получение оценок персональных качеств)

Формирование нейросетевых архитектур моделей для получения результатов оценки персональных качеств (мультимодальное объединение)

- _b5.av_models_b5_ - Нейросетевые модели **tf.keras.Model** для получения результатов оценки персональных качеств

[15]:
```
res_load_av_models_b5 = _b5.load_av_models_b5(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:46:42] Формирование нейросетевых архитектур моделей для получения результатов оценки персональных качеств (мультимодальное объединение) ...

— Время выполнения: 0.048 сек. —

Загрузка весов нейросетевых моделей для получения результатов оценки персональных качеств (мультимодальное объединение)

- `_b5.av_models_b5_` - Нейросетевые модели **tf.keras.Model** для получения результатов оценки персональных качеств

[16]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url_openness = _b5.weights_for_big5_['av']['fi']['b5']['openness']['sberdisk']
url_conscientiousness = _b5.weights_for_big5_['av']['fi']['b5']['conscientiousness']['sberdisk']
url_extraversion = _b5.weights_for_big5_['av']['fi']['b5']['extraversion']['sberdisk']
url_agreeableness = _b5.weights_for_big5_['av']['fi']['b5']['agreeableness']['sberdisk']
url_non_neuroticism = _b5.weights_for_big5_['av']['fi']['b5']['non_neuroticism']['sberdisk']

res_load_av_models_weights_b5 = _b5.load_av_models_weights_b5(
    url_openness = url_openness, # Открытость опыту
    url_conscientiousness = url_conscientiousness, # Добросовестность
    url_extraversion = url_extraversion, # Экстраверсия
    url_agreeableness = url_agreeableness, # Доброжелательность
    url_non_neuroticism = url_non_neuroticism, # "Эмоциональная стабильность
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-14 22:46:47] Загрузка весов нейросетевых моделей для получения результатов оценки персональных качеств (мультимодальное объединение) ...

[2023-12-14 22:46:47] Загрузка файла "weights_2022-08-28_11-14-35.h5" 100.0% ...
Открытость опыту

[2023-12-14 22:46:47] Загрузка файла "weights_2022-08-28_11-08-10.h5" 100.0% ...
Добросовестность

[2023-12-14 22:46:47] Загрузка файла "weights_2022-08-28_11-17-57.h5" 100.0% ...
Экстраверсия

[2023-12-14 22:46:47] Загрузка файла "weights_2022-08-28_11-25-11.h5" 100.0% ...
Доброжелательность

[2023-12-14 22:46:47] Загрузка файла "weights_2022-06-14_21-44-09.h5" 100.0% ...
Эмоциональная стабильность

— Время выполнения: 0.785 сек. —

Получение прогнозов (мультимодальное объединение)

- `_b5.df_files_` - DataFrame с данными
- `_b5.df_accuracy_` - DataFrame с результатами вычисления точности

[17]:
```
# Настройки ядра
_b5.path_to_dataset_ = 'E:/Databases/FirstImpressionsV2/test' # Директория набора данных
# Директории не входящие в выборку
_b5.ignore_dirs_ = []
# Названия ключей для DataFrame набора данных
_b5.keys_dataset_ = ['Path', 'Openness', 'Conscientiousness', 'Extraversion',
↪'Agreeableness', 'Non-Neuroticism']
_b5.ext_ = ['.mp4'] # Расширения искомых файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_['fi']['sberdisk']

_b5.get_av_union_predictions(
    depth = 1,          # Глубина иерархии для получения аудио и видеоданных
    recursive = False, # Рекурсивный поиск данных
    sr = 44100,         # Частота дискретизации
    window_audio = 2,  # Размер окна сегмента аудио сигнала (в секундах)
    step_audio = 1,    # Шаг сдвига окна сегмента аудио сигнала (в секундах)
    reduction_fps = 5, # Понижение кадровой частоты
    window_video = 10, # Размер окна сегмента видео сигнала (в секундах)
    step_video = 5,    # Шаг сдвига окна сегмента видео сигнала (в секундах)
    lang = 'en',
    accuracy = True,   # Вычисление точности
    url_accuracy = url_accuracy,
    logs = True,        # При необходимости формировать LOG файл
    out = True,         # Отображение
    runtime = True,    # Подсчет времени выполнения
    run = True          # Блокировка выполнения
)
```

[2023-12-15 01:11:04] Получение прогнозов и вычисление точности (мультимодальное объединение) ...

2000 из 2000 (100.0%) ... test80_25_Q4wOgixh7E.004.mp4 ...

ID	Path	Openness	\
1	E:\Databases\FirstImpressionsV2\test\test80_01...	0.554249	
2	E:\Databases\FirstImpressionsV2\test\test80_01...	0.558823	
3	E:\Databases\FirstImpressionsV2\test\test80_01...	0.477549	
4	E:\Databases\FirstImpressionsV2\test\test80_01...	0.662656	
5	E:\Databases\FirstImpressionsV2\test\test80_01...	0.645876	
6	E:\Databases\FirstImpressionsV2\test\test80_01...	0.67497	
7	E:\Databases\FirstImpressionsV2\test\test80_01...	0.39908	
8	E:\Databases\FirstImpressionsV2\test\test80_01...	0.577705	
9	E:\Databases\FirstImpressionsV2\test\test80_01...	0.543675	
10	E:\Databases\FirstImpressionsV2\test\test80_01...	0.54876	
11	E:\Databases\FirstImpressionsV2\test\test80_01...	0.546634	
12	E:\Databases\FirstImpressionsV2\test\test80_01...	0.459302	

(continues on next page)

(продолжение с предыдущей страницы)

```
13  E:\Databases\FirstImpressionsV2\test\test80_01...  0.309097
14  E:\Databases\FirstImpressionsV2\test\test80_01...  0.643403
15  E:\Databases\FirstImpressionsV2\test\test80_01...   0.65016
16  E:\Databases\FirstImpressionsV2\test\test80_01...  0.598313
17  E:\Databases\FirstImpressionsV2\test\test80_01...  0.571537
18  E:\Databases\FirstImpressionsV2\test\test80_01...  0.552433
19  E:\Databases\FirstImpressionsV2\test\test80_01...  0.658695
20  E:\Databases\FirstImpressionsV2\test\test80_01...  0.660076
21  E:\Databases\FirstImpressionsV2\test\test80_01...  0.543881
22  E:\Databases\FirstImpressionsV2\test\test80_01...  0.537325
23  E:\Databases\FirstImpressionsV2\test\test80_01...  0.464761
24  E:\Databases\FirstImpressionsV2\test\test80_01...  0.633951
25  E:\Databases\FirstImpressionsV2\test\test80_01...    0.4517
26  E:\Databases\FirstImpressionsV2\test\test80_01...  0.602848
27  E:\Databases\FirstImpressionsV2\test\test80_01...  0.586638
28  E:\Databases\FirstImpressionsV2\test\test80_01...  0.689552
29  E:\Databases\FirstImpressionsV2\test\test80_01...  0.583505
30  E:\Databases\FirstImpressionsV2\test\test80_01...  0.642695

   Conscientiousness Extraversion Agreeableness Non-Neuroticism
ID
1           0.506548     0.440194      0.540235         0.48605
2           0.442357      0.50397      0.558767        0.521587
3           0.568616     0.333939      0.491873        0.458966
4           0.621852      0.58996      0.599038        0.636035
5           0.532378     0.551939      0.589174        0.552269
6           0.666972     0.617604      0.610567        0.641452
7           0.397298     0.335823      0.497966         0.39729
8           0.597157     0.498064      0.640584        0.600152
9           0.451197     0.449555      0.482371        0.415256
10           0.51097     0.433856      0.579709        0.536171
11          0.398485     0.443701      0.518107        0.492343
12          0.427114     0.315686      0.495817        0.457954
13          0.317028     0.218514      0.372315        0.241697
14          0.509414     0.483608      0.503154        0.550979
15          0.840148     0.535299      0.710939        0.743357
16          0.520505     0.450767      0.486345        0.561532
17          0.673989     0.472203      0.615608        0.621064
18          0.568787     0.457108      0.613188        0.570902
19          0.625194     0.634877      0.612277        0.626052
20          0.544358      0.64178      0.604572        0.628259
21          0.477881     0.407731      0.555772        0.499664
22           0.46375     0.419255      0.499785        0.455146
23          0.434816     0.346836      0.428429        0.358087
24           0.63333     0.584644      0.615227        0.608006
25          0.574346     0.350136      0.526873        0.468283
26          0.592382     0.494679      0.539232        0.505865
27          0.521421     0.485391      0.530296        0.535499
28          0.643902     0.695799      0.646209        0.686243
29          0.564313     0.502263      0.554502        0.539899
30          0.588222     0.617706      0.615312        0.626649
```

[2023-12-15 01:11:04] Точность по отдельным персональным качествам личности человека ...

Metrics	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism	Mean
MAE	0.0845	0.0802	0.0793	0.0858	0.0847	0.0829
Accuracy	0.9155	0.9198	0.9207	0.9142	0.9153	0.9171

[2023-12-15 01:11:04] Средняя средних абсолютных ошибок: 0.0829, средняя точность: 0.9171 ...

Лог файлы успешно сохранены ...

— Время выполнения: 8654.754 сек. —

[17]:
```
True
```

Мультимодальное объединение для получения прогнозов по аудио, видео и тексту FI V2

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-15 07:01:44] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a16 Лицензия: BSD License

Получение и отображение версий установленных библиотек

- `_b5.df_pkgs_` - DataFrame с версиями установленных библиотек

[4]:
```
_b5.libs_vers(runtime = True, run = True)
```

	Package	Version
1	TensorFlow	2.15.0
2	Keras	2.15.0
3	OpenCV	4.8.1
4	MediaPipe	0.9.0
5	NumPy	1.26.2
6	SciPy	1.11.4
7	Pandas	2.1.3
8	Scikit-learn	1.3.2
9	OpenSmile	2.5.0
10	Librosa	0.10.1
11	AudioRead	3.0.1
12	IPython	8.18.1
13	PyMediaInfo	6.1.0
14	Requests	2.31.0
15	JupyterLab	4.0.9
16	LIWC	0.5.0
17	Transformers	4.36.0
18	Sentencepiece	0.1.99
19	Torch	2.0.1+cpu
20	Torchaudio	2.0.2+cpu

— Время выполнения: 0.004 сек. —

Анализ аудио информации (формированияе и загрузка весов моделей)

Формирование нейросетевой архитектуры модели для получения признаков / оценок на базе экспертных признаков

- `_b5.audio_model_hc_` - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе экспертных признаков

[5]:
```
res_load_audio_model_hc = _b5.load_audio_model_hc(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:44] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (аудио модальность) ...

— Время выполнения: 0.326 сек. —

Загрузка весов нейросетевой модели для получения признаков / оценок на базе экспертных признаков

- _b5.audio_model_hc_ - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе экспертных признаков

[6]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

corpus = 'fi'
lang = 'en'

url = _b5.weights_for_big5_['audio'][corpus]['hc']['sberdisk']

res_load_audio_model_weights_hc = _b5.load_audio_model_weights_hc(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:45] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (аудио модальность) ...

[2023-12-15 07:01:45] Загрузка файла "weights_2022-05-05_11-27-55.h5" 100.0% ...

— Время выполнения: 0.226 сек. —

Формирование нейросетевой архитектуры модели для получения признаков / оценок на базе нейросетевых признаков

- _b5.audio_model_nn_ - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе нейросетевых признаков

[7]:
```
res_load_audio_model_nn = _b5.load_audio_model_nn(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:45] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (аудио модальность) ...

— Время выполнения: 0.219 сек. —

Загрузка весов нейросетевой модели для получения признаков / оценок на базе нейросетевых признаков

- _b5.audio_model_nn_ - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе нейросетевых признаков

[8]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['audio'][corpus]['nn']['sberdisk']

res_load_audio_model_weights_nn = _b5.load_audio_model_weights_nn(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели␣
↪из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:45] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (аудио модальность) ...

[2023-12-15 07:01:45] Загрузка файла "weights_2022-05-03_07-46-14.h5"

— Время выполнения: 0.328 сек. —

Анализ видео информации (формирование и загрузка весов моделей)

Формирование нейросетевой архитектуры модели для получения признаков / оценок на базе экспертных признаков

- _b5.video_model_hc_ - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе экспертных признаков

[9]:
```
res_load_video_model_hc = _b5.load_video_model_hc(
    lang = lang,
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:45] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (видео модальность) ...

— Время выполнения: 0.252 сек. —

Загрузка весов нейросетевой модели для получения признаков / оценок на базе экспертных признаков

- `_b5.video_model_hc_` - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе экспертных признаков

[10]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video'][corpus]['hc']['sberdisk']

res_load_video_model_weights_hc = _b5.load_video_model_weights_hc(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:46] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (видео модальность) ...

[2023-12-15 07:01:46] Загрузка файла "weights_2022-08-27_18-53-35.h5" 100.0% ...

— Время выполнения: 0.24 сек. —

Формирование нейросетевой архитектуры для получения нейросетевых признаков

- `_b5.video_model_deep_fe_` - Нейросетевая модель **tf.keras.Model** для получения нейросетевых признаков

[11]:
```
res_load_video_model_deep_fe = _b5.load_video_model_deep_fe(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:46] Формирование нейросетевой архитектуры для получения нейросетевых признаков (видео модальность) ...

— Время выполнения: 0.794 сек. —

Загрузка весов нейросетевой модели для получения нейросетевых признаков (видео модальность)

- `_b5.video_model_deep_fe_` - Нейросетевая модель **tf.keras.Model** для получения нейросетевых признаков

[12]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video'][corpus]['fe']['sberdisk']

res_load_video_model_weights_deep_fe = _b5.load_video_model_weights_deep_fe(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:47] Загрузка весов нейросетевой модели для получения нейросетевых признаков (видео модальность) ...

[2023-12-15 07:01:50] Загрузка файла "weights_2022-11-01_12-27-07.h5" 100.0% ...

— Время выполнения: 3.937 сек. —

Формирование нейросетевой архитектуры модели для получения признаков / оценок на базе нейросетевых признаков

- `_b5.video_model_nn_` - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе нейросетевых признаков

[13]:
```
res_load_video_model_nn = _b5.load_video_model_nn(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:51] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (видео модальность) ...

— Время выполнения: 0.707 сек. —

Загрузка весов нейросетевой модели для получения признаков / оценок на базе нейросетевых признаков

- _b5.video_model_nn_ - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе нейросетевых признаков

[14]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['video'][corpus]['nn']['sberdisk']

res_load_video_model_weights_nn = _b5.load_video_model_weights_nn(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:51] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (видео модальность) ...

[2023-12-15 07:01:51] Загрузка файла "weights_2022-03-22_16-31-48.h5"

— Время выполнения: 0.166 сек. —

Анализ текстовой информации (формирование и загрузка весов моделей)

Загрузка словаря с экспертными признаками

[15]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

res_load_text_features = _b5.load_text_features(
    force_reload = True, # Принудительная загрузка файла
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:51] Загрузка словаря с экспертными признаками ...

[2023-12-15 07:01:52] Загрузка файла "LIWC2007.txt" 100.0% ...

— Время выполнения: 0.166 сек. —

Формирование токенизатора и нейросетевой модели машинного перевода (RU -> EN)

[16]:
```
res_setup_translation_model = _b5.setup_translation_model(
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:52] Формирование токенизатора и нейросетевой модели машинного перевода ...

— Время выполнения: 1.763 сек. —

Формирование токенизатора и нейросетевой модели BERT (для кодирования слов)

[17]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

res_setup_translation_model = _b5.setup_bert_encoder(
    force_reload = False, # Принудительная загрузка файла
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:53] Формирование токенизатора и нейросетевой модели BERT ...

[2023-12-15 07:01:55] Загрузка файла “bert-base-multilingual-cased.zip”

[2023-12-15 07:01:53] Формирование токенизатора и нейросетевой модели BERT ...

[2023-12-15 07:01:55] Загрузка файла “bert-base-multilingual-cased.zip”

[2023-12-15 07:01:55] Разархивирование архива “bert-base-multilingual-cased.zip” ...

— Время выполнения: 5.269 сек. —

Формирование нейросетевой архитектуры модели для получения признаков / оценок на базе экспертных признаков

- `_b5.text_model_hc_` - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе экспертных признаков

[18]:
```
res_load_text_model_hc_fi = _b5.load_text_model_hc(
    corpus = corpus, # Корпус для тестирования нейросетевой модели
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:59] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (текстовая модальность) ...

— Время выполнения: 0.701 сек. —

Загрузка весов нейросетевой модели для получения признаков / оценок на базе экспертных признаков

- _b5.text_model_hc_ - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе экспертных признаков

[19]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['text'][corpus]['hc']['sberdisk']

res_load_text_model_weights_hc_fi = _b5.load_text_model_weights_hc(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:01:59] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (текстовая модальность) ...

[2023-12-15 07:02:00] Загрузка файла "weights_2023-07-15_10-52-15.h5" 100.0% ...

— Время выполнения: 0.278 сек. —

Формирование нейросетевой архитектуры модели для получения признаков / оценок на базе нейросетевых признаков

- _b5.text_model_nn_ - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе нейросетевых признаков

[20]:
```
res_load_text_model_nn_fi = _b5.load_text_model_nn(
    corpus = corpus, # Корпус для тестирования нейросетевой модели
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:02:00] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (текстовая модальность) ...

— Время выполнения: 0.286 сек. —

Загрузка весов нейросетевой модели для получения признаков / оценок на базе нейросетевых признаков

- `_b5.text_model_nn_` - Нейросетевая модель **tf.keras.Model** для получения признаков / оценок на базе нейросетевых признаков

[21]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['text'][corpus]['nn']['sberdisk']

res_load_text_model_weights_nn_fi = _b5.load_text_model_weights_nn(
    url = url, # Полный путь к файлу с весами нейросетевой модели
    force_reload = True, # Принудительная загрузка файла с весами нейросетевой модели из␣
↪сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:02:00] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (текстовая модальность) ...

[2023-12-15 07:02:00] Загрузка файла "weights_2023-07-03_15-01-08.h5" 100.0% ...

— Время выполнения: 0.42 сек. —

Анализ мультимодальной информации (формирование, загрузка весов моделей, получение оценок персональных качеств)

Формирование нейросетевой архитектуры модели для получения оценок персональных качеств

- `_b5.avt_model_b5_` - Нейросетевая модель **tf.keras.Model** для получения оценок персональных качеств

[22]:
```
res_load_avt_model_b5 = _b5.load_avt_model_b5(
    show_summary = False, # Отображение сформированной нейросетевой архитектуры модели
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:02:00] Формирование нейросетевой архитектуры модели для получения оценок персональных качеств (мультимодальное объединение) ...

— Время выполнения: 0.212 сек. —

Загрузка весов нейросетевой модели для получения оценок персональных качеств

- `_b5.avt_model_b5_` - Нейросетевая модель **tf.keras.Model** для получения оценок персональных качеств

[23]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

url = _b5.weights_for_big5_['avt'][corpus]['b5']['sberdisk']

res_load_avt_model_weights_b5 = _b5.load_avt_model_weights_b5(
    url = url,
    force_reload = False, # Принудительная загрузка файла с весами нейросетевой модели␣
↪из сети
    out = True, # Отображение
    runtime = True, # Подсчет времени выполнения
    run = True # Блокировка выполнения
)
```

[2023-12-15 07:02:01] Загрузка весов нейросетевой модели для получения оценок персональных качеств (мультимодальное объединение) ...

[2023-12-15 07:02:01] Загрузка файла "avt_fi_2023-12-03_11-36-51.h5"

— Время выполнения: 0.295 сек. —

Получение прогнозов (мультимодальное объединение)

- `_b5.df_files_` - DataFrame с данными
- `_b5.df_accuracy_` - DataFrame с результатами вычисления точности

[24]:
```
# Настройки ядра
_b5.path_to_dataset_ = 'E:/Databases/FirstImpressionsV2/test' # Директория набора данных
# Директории не входящие в выборку
_b5.ignore_dirs_ = []
# Названия ключей для DataFrame набора данных
_b5.keys_dataset_ = ['Path', 'Openness', 'Conscientiousness', 'Extraversion',
↪'Agreeableness', 'Non-Neuroticism']
_b5.ext_ = ['.mp4'] # Расширения искомых файлов

# Полный путь к файлу с верными предсказаниями для подсчета точности
url_accuracy = _b5.true_traits_[corpus]['sberdisk']

_b5.get_avt_predictions(
    depth = 1,          # Глубина иерархии для получения аудио и видеоданных
    recursive = False, # Рекурсивный поиск данных
    sr = 44100,         # Частота дискретизации
    window_audio = 2,  # Размер окна сегмента аудио сигнала (в секундах)
    step_audio = 1,    # Шаг сдвига окна сегмента аудио сигнала (в секундах)
    reduction_fps = 5, # Понижение кадровой частоты
    window_video = 10, # Размер окна сегмента видео сигнала (в секундах)
    step_video = 5,    # Шаг сдвига окна сегмента видео сигнала (в секундах)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    asr = False,         # Распознавание речи
    lang = lang,        # Выбор языка
    accuracy = True,    # Вычисление точности
    url_accuracy = url_accuracy,
    logs = True,        # При необходимости формировать LOG файл
    out = True,         # Отображение
    runtime = True,     # Подсчет времени выполнения
    run = True          # Блокировка выполнения
)
```

[2023-12-15 10:22:11] Извлечение признаков (экспертных и нейросетевых) из текста ...

[2023-12-15 10:22:14] Получение прогнозов и вычисление точности (мультимодальное объединение) ...

2000 из 2000 (100.0%) ... test80_25_Q4wOgixh7E.004.mp4 ...

ID	Path	Openness	Conscientiousness	Extraversion	Agreeableness	Non-Neuroticism
1	E:\Databases\FirstImpressionsV2\test\test80_01...	0.545377	0.523155	0.456685	0.533811	0.516093
2	E:\Databases\FirstImpressionsV2\test\test80_01...	0.520572	0.396216	0.478419	0.528622	0.459169
3	E:\Databases\FirstImpressionsV2\test\test80_01...	0.450715				
4	E:\Databases\FirstImpressionsV2\test\test80_01...	0.665193				
5	E:\Databases\FirstImpressionsV2\test\test80_01...	0.669463				
6	E:\Databases\FirstImpressionsV2\test\test80_01...	0.632529				
7	E:\Databases\FirstImpressionsV2\test\test80_01...	0.489579				
8	E:\Databases\FirstImpressionsV2\test\test80_01...	0.59544				
9	E:\Databases\FirstImpressionsV2\test\test80_01...	0.559325				
10	E:\Databases\FirstImpressionsV2\test\test80_01...	0.509495				
11	E:\Databases\FirstImpressionsV2\test\test80_01...	0.599391				
12	E:\Databases\FirstImpressionsV2\test\test80_01...	0.458006				
13	E:\Databases\FirstImpressionsV2\test\test80_01...	0.377578				
14	E:\Databases\FirstImpressionsV2\test\test80_01...	0.563649				
15	E:\Databases\FirstImpressionsV2\test\test80_01...	0.7302				
16	E:\Databases\FirstImpressionsV2\test\test80_01...	0.620163				
17	E:\Databases\FirstImpressionsV2\test\test80_01...	0.603495				
18	E:\Databases\FirstImpressionsV2\test\test80_01...	0.543104				
19	E:\Databases\FirstImpressionsV2\test\test80_01...	0.624445				
20	E:\Databases\FirstImpressionsV2\test\test80_01...	0.658763				
21	E:\Databases\FirstImpressionsV2\test\test80_01...	0.562814				
22	E:\Databases\FirstImpressionsV2\test\test80_01...	0.472688				
23	E:\Databases\FirstImpressionsV2\test\test80_01...	0.43985				
24	E:\Databases\FirstImpressionsV2\test\test80_01...	0.638308				
25	E:\Databases\FirstImpressionsV2\test\test80_01...	0.506815				
26	E:\Databases\FirstImpressionsV2\test\test80_01...	0.517949				
27	E:\Databases\FirstImpressionsV2\test\test80_01...	0.570406				
28	E:\Databases\FirstImpressionsV2\test\test80_01...	0.637813				
29	E:\Databases\FirstImpressionsV2\test\test80_01...	0.572268				
30	E:\Databases\FirstImpressionsV2\test\test80_01...	0.658128				

(continues on next page)

(продолжение с предыдущей страницы)

```
3          0.491121      0.36674      0.510387        0.414304
4          0.648017     0.640581      0.580625        0.596675
5          0.606313     0.619956      0.653291        0.618665
6          0.722035     0.583922       0.63653        0.603358
7          0.453927     0.373339      0.486156        0.421787
8          0.615519     0.514064      0.627394        0.601345
9           0.50692     0.442211      0.537979        0.499341
10         0.526581     0.406979      0.565923         0.54616
11         0.516418     0.516382      0.589003        0.558064
12         0.496319     0.345605       0.48779        0.448027
13         0.410694     0.283698      0.384478        0.313993
14         0.499573     0.445833      0.454925        0.463903
15         0.784698      0.51636      0.698729        0.713016
16         0.564576     0.556421      0.563072        0.543618
17         0.644997     0.440616      0.603712        0.578639
18         0.489751     0.452691      0.566111        0.520961
19         0.574276     0.609165      0.582815        0.560111
20         0.545697     0.627865       0.61989        0.609391
21         0.493076     0.430422      0.539134        0.502142
22         0.417943     0.423233      0.472491        0.392815
23         0.429655     0.319237      0.420569        0.414306
24         0.632067     0.580016      0.642938        0.603159
25          0.57838     0.367448      0.523856        0.481819
26         0.562723     0.383299      0.483178        0.467141
27         0.441804     0.454944      0.530368        0.512669
28         0.611132     0.607629      0.636313        0.620745
29         0.532781     0.504937      0.575169        0.518609
30         0.598394      0.59656      0.621783        0.612908
```

[2023-12-15 10:22:14] Точность по отдельным персональным качествам личности человека ...

```
          Openness  Conscientiousness  Extraversion  Agreeableness  \
Metrics
MAE         0.0758             0.0716        0.0688         0.0752
Accuracy    0.9242             0.9284        0.9312         0.9248

          Non-Neuroticism    Mean
Metrics
MAE                0.0731  0.0729
Accuracy           0.9269  0.9271
```

[2023-12-15 10:22:14] Средняя средних абсолютных ошибок: 0.0729, средняя точность: 0.9271 ...

Лог файлы успешно сохранены ...

— Время выполнения: 12013.03 сек. —

[24]:
```
True
```

Дополнительные возможности

Загрузка файла из URL

Импорт необходимых инструментов

[2]:
```
from oceanai.modules.lab.build import Run
```

Сборка

[3]:
```
_b5 = Run(
    lang = 'ru', # Язык
    color_simple = '#333', # Цвет обычного текста (шестнадцатеричный код)
    color_info = '#1776D2', # Цвет текста содержащего информацию (шестнадцатеричный код)
    color_err = '#FF0000', # Цвет текста содержащего ошибку (шестнадцатеричный код)
    color_true = '#008001', # Цвет текста содержащего положительную информацию␣
↪(шестнадцатеричный код)
    bold_text = True, # Жирное начертание текста
    num_to_df_display = 30, # Количество строк для отображения в таблицах
    text_runtime = 'Время выполнения', # Текст времени выполнения
    metadata = True # Отображение информации о библиотеке
)
```

[2023-12-10 16:49:03] OCEANAI - персональные качества личности человека: Авторы: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Карпов Алексей [karpov@iias.spb.su] Сопровождающие: Рюмина Елена [ryumina_ev@mail.ru] Рюмин Дмитрий [dl_03.03.1991@mail.ru] Версия: 1.0.0a5 Лицензия: BSD License

Процесс загрузки

[4]:
```
# Настройки ядра
_b5.path_to_save_ = './models' # Директория для сохранения файла
_b5.chunk_size_ = 2000000 # Размер загрузки файла из сети за 1 шаг

res_download_file_from_url = _b5.download_file_from_url(
    url = 'https://download.sberdisk.ru/download/file/400635799?token=MMRrak8fMsyzxLE&
↪filename=weights_2022-05-05_11-27-55.h5',
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[2023-12-10 16:49:04] Загрузка файла “weights_2022-05-05_11-27-55.h5” 100.0% ...

— Время выполнения: 0.214 сек. —

[5]:
```
res_download_file_from_url
```

[5]:
```
200
```

4.2.2 Модули

Пользовательские исключения

exception oceanai.modules.core.exceptions.CustomException

Базовые классы: Exception

Класс для всех пользовательских исключений

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.exceptions import CustomException

message = 'Пользовательское исключение'

try: raise CustomException(message)
except CustomException as ex: print(ex)
```

[1]:
```
Пользовательское исключение
```

exception oceanai.modules.core.exceptions.InvalidContentLength

Базовые классы: *CustomException*

Не определен размер файла для загрузки

Пример

верно – 1 –

In [1]:
```
from oceanai.modules.core.exceptions import InvalidContentLength

message = 'Не определен размер файла для загрузки'

try: raise InvalidContentLength(message)
except InvalidContentLength as ex: print(ex)
```

[1]:
```
Не определен размер файла для загрузки
```

exception oceanai.modules.core.exceptions.IsSmallWindowSizeError

Базовые классы: *CustomException*

Указан слишком маленький размер окна сегмента сигнала

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.exceptions import IsSmallWindowSizeError

message = 'Указан слишком маленький размер окна сегмента сигнала'

try: raise IsSmallWindowSizeError(message)
except IsSmallWindowSizeError as ex: print(ex)
```

[1]:
```
Указан слишком маленький размер окна сегмента сигнала
```

Определение языка

class oceanai.modules.core.language.Language(*lang: str = 'ru'*)

Базовые классы: object

Класс для интернационализации (I18N) и локализации (L10N)

Параметры
lang (*str*) – Язык

__get_languages() → List[Optional[str]]

Получение поддерживаемых языков

Примечание: private (приватный метод)

Результат
Список поддерживаемых языков

Тип результата
List[Optional[str]]

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 'ru')
language._Language__get_languages()
```

[1]:
```
['ru', 'en']
```

__get_locales() → Dict[str, method]

Получение языковых пакетов

Примечание: private (приватный метод)

Результат
Словарь с языковыми пакетами

Тип результата
Dict[str, MethodType]

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 'ru')
language._Language__get_locales()
```

[1]:
```
{
    'ru': <bound method GNUTranslations.gettext of <gettext.GNUTranslations␣
↪object at 0x14680ce50>>,
    'en': <bound method GNUTranslations.gettext of <gettext.GNUTranslations␣
↪object at 0x1460ddbb0>>
}
```

`__set_locale`(*lang: str = ''*) → method
Установка языка

Примечание: private (приватный метод)

Параметры
`lang` (*str*) – Язык

Результат
MethodType перевода строк на один из поддерживаемых языков если метод запущен через конструктор

Тип результата
MethodType

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 'ru')
print(language.lang_)
```

[1]:
```
ru
```

– 2 –

In [2]:

```
from oceanai.modules.core.language import Language

language = Language(lang = 'ru')
language._Language__set_locale('en')
print(language.lang_)
```

[2]:
```
en
```

lang: str = 'ru'

Язык, доступные варианты:

- "ru" - Русский язык (по умолчанию)
- "en" - Английский язык

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.language import Language

language = Language()
print(language.lang, language.lang_)
```

[1]:
```
ru ru
```

– 2 –

In [2]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 'ru')
print(language.lang, language.lang_)
```

[2]:
```
ru ru
```

– 3 –

In [3]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 'en')
print(language.lang, language.lang_)
```

[3]:
```
en en
```

Лучше так не делать – 1 –

In [4]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 'es')
print(language.lang, language.lang_)
```

[4]:
```
es ru
```

– 2 –

In [5]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 1)
print(language.lang, language.lang_)
```

[5]:
```
1 ru
```

Type
str

property lang_: str

Получение текущего языка

Результат
Язык

Тип результата
str

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.language import Language

language = Language()
print(language.lang_)
```

[1]:
```
ru
```

– 2 –

In [2]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 'ru')
print(language.lang_)
```

[2]:
```
ru
```

– 3 –

In [3]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 'en')
print(language.lang_)
```

[3]:

```
en
```

Лучше так не делать – 1 –

In [4]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 'es')
print(language.lang_)
```

[4]:
```
ru
```

– 2 –

In [5]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 1)
print(language.lang_)
```

[5]:
```
ru
```

property locales_: List[str]

Получение поддерживаемых языков

Результат
Список поддерживаемых языков

Тип результата
List[str]

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 'en')
print(language.locales_)
```

[1]:
```
['ru', 'en']
```

property path_to_locales_: str

Получение директории с языковыми пакетами

Результат
Директория с языковыми пакетами

Тип результата
str

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.language import Language

language = Language(lang = 'en')
# У каждого пользователя свой путь
print(language.path_to_locales_)
```

[1]:
```
/Users/dl/GitHub/OCEANAI/oceanai/modules/locales
```

Сообщения

class oceanai.modules.core.messages.Messages(*lang: str = 'ru'*)

Базовые классы: *Language*

Класс для сообщений

Параметры
lang (*str*) – Смотреть *lang*

Настройки

class oceanai.modules.core.settings.Settings(*lang: str = 'ru'*, *color_simple: str = '#666'*, *color_info: str = '#1776D2'*, *color_err: str = '#FF0000'*, *color_true: str = '#008001'*, *bold_text: bool = True*, *text_runtime: str = ''*, *num_to_df_display: int = 30*)

Базовые классы: *Messages*

Класс для настроек

Параметры

- lang (*str*) – Смотреть *lang*
- color_simple (*str*) – Цвет обычного текста (шестнадцатеричный код)
- color_info (*str*) – Цвет текста содержащего информацию (шестнадцатеричный код)
- color_err (*str*) – Цвет текста содержащего ошибку (шестнадцатеричный код)
- color_true (*str*) – Цвет текста содержащего положительную информацию (шестнадцатеричный код)
- bold_text (*bool*) – Жирное начертание текста
- num_to_df_display (*int*) – Количество строк для отображения в таблицах
- text_runtime (*str*) – Текст времени выполнения

bold_text: bool = True

Жирное начертание текста

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(lang = 'ru')
print(settings.bold_text, settings.bold_text_)
```

[1]:
```
True True
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(lang = 'ru', bold_text = True)
print(settings.bold_text, settings.bold_text_)
```

[2]:
```
True True
```

– 3 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(lang = 'ru', bold_text = False)
print(settings.bold_text, settings.bold_text_)
```

[3]:
```
False False
```

Лучше так не делать – 1 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(lang = 'ru', bold_text = 1)
print(settings.bold_text, settings.bold_text_)
```

[4]:
```
True True
```

– 2 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(lang = 'ru', bold_text = 'какой-то_текст')
print(settings.bold_text, settings.bold_text_)
```

[5]:
```
True True
```

Type
bool

property bold_text_: bool

Получение/установка жирного начертания текста

Параметры
(bool) – **True** или **False**

Результат
True или **False**

Тип результата
bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(lang = 'ru', bold_text = True)
print(settings.bold_text_)
```

[1]:
```
True
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(lang = 'ru', bold_text = True)
settings.bold_text_ = False
print(settings.bold_text_)
```

[2]:
```
False
```

Лучше так не делать – 1 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(lang = 'en', bold_text = False)
settings.bold_text_ = 1
print(settings.bold_text_)
```

[3]:
```
False
```

– 2 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(lang = 'ru', bold_text = True)
settings.bold_text_ = 'какой-то_текст'
print(settings.bold_text_)
```

[4]:

```
True
```

property chunk_size_: int

Получение/установка размера загрузки файла из сети за 1 шаг

Параметры
(int) – Размер загрузки файла из сети за 1 шаг

Результат
Размер загрузки файла из сети за 1 шаг

Тип результата
int

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.chunk_size_)
```

[1]:
```
1000000
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.chunk_size_ = 2000000
print(settings.chunk_size_)
```

[2]:
```
2000000
```

Лучше так не делать – 1 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.chunk_size_ = -1
print(settings.chunk_size_)
```

[3]:
```
1000000
```

– 2 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.chunk_size_ = False
print(settings.chunk_size_)
```

[4]:
```
1000000
```

– 3 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.chunk_size_ = 'какой-то_текст'
print(settings.chunk_size_)
```

[5]:
```
1000000
```

color_err: str = '#FF0000'

Цвет текста содержащего ошибку (шестнадцатеричный код)

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.color_err, settings.color_err_)
```

[1]:
```
#FF0000 #FF0000
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_err = 'FF0000')
print(settings.color_err, settings.color_err_)
```

[2]:
```
#FF0000 #FF0000
```

– 3 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_err = '#FF0')
print(settings.color_err, settings.color_err_)
```

[3]:
```
#FF0 #FF0
```

Лучше так не делать – 1 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_err = 1)
print(settings.color_err, settings.color_err_)
```

[4]:
```
#FF0000 #FF0000
```

– 2 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_err = [])
print(settings.color_err, settings.color_err_)
```

[5]:
```
#FF0000 #FF0000
```

Type
str

property color_err_: str

Получение/установка цвета текста содержащего ошибку

Параметры
(str) – Шестнадцатеричный код

Результат
Шестнадцатеричный код

Тип результата
str

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_err = '#C22931')
print(settings.color_err_)
```

[1]:
```
#C22931
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.color_err_ = '#FF0'
print(settings.color_err_)
```

[2]:
```
#FF0
```

Лучше так не делать – 1 –

In [3]:

```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.color_err_ = 1
print(settings.color_err_)
```

[3]:
```
#FF0000
```

– 2 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.color_err_ = {}
print(settings.color_err_)
```

[4]:
```
#FF0000
```

color_info: str = '#1776D2'

Цвет текста содержащего информацию (шестнадцатеричный код)

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.color_info, settings.color_info_)
```

[1]:
```
#1776D2 #1776D2
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_info = '#1776D2')
print(settings.color_info, settings.color_info_)
```

[2]:
```
#1776D2 #1776D2
```

– 3 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_info = '#42F')
print(settings.color_info, settings.color_info_)
```

[3]:

```
#42F #42F
```

Лучше так не делать – 1 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_info = 1)
print(settings.color_info, settings.color_info_)
```

[4]:
```
#1776D2 #1776D2
```

– 2 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_info = [])
print(settings.color_info, settings.color_info_)
```

[5]:
```
#1776D2 #1776D2
```

Type
str

property color_info_: str

Получение/установка цвета текста содержащего информацию

Параметры
(str) – Шестнадцатеричный код

Результат
Шестнадцатеричный код

Тип результата
str

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_info = '#1776D2')
print(settings.color_info_)
```

[1]:
```
#1776D2
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

```

(continues on next page)

(продолжение с предыдущей страницы)

```
settings = Settings()
settings.color_info_ = '#42F'
print(settings.color_info_)
```

[2]:
```
#42F
```

Лучше так не делать – 1 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.color_info_ = 1
print(settings.color_info_)
```

[3]:
```
#1776D2
```

– 2 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.color_info_ = ()
print(settings.color_info_)
```

[4]:
```
#1776D2
```

`color_simple: str = '#666'`

Цвет обычного текста (шестнадцатеричный код)

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.color_simple, settings.color_simple_)
```

[1]:
```
#666 #666
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_simple = '#666')
print(settings.color_simple, settings.color_simple_)
```

[2]:

```
#666 #666
```

– 3 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_simple = '#222')
print(settings.color_simple, settings.color_simple_)
```

[3]:
```
#222 #222
```

Лучше так не делать – 1 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_simple = 1)
print(settings.color_simple, settings.color_simple_)
```

[4]:
```
#666 #666
```

– 2 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_simple = {1, 2, 3})
print(settings.color_simple, settings.color_simple_)
```

[5]:
```
#666 #666
```

Type
str

property color_simple_: str

Получение/установка цвета обычного текста

Параметры
(str) – Шестнадцатеричный код

Результат
Шестнадцатеричный код

Тип результата
str

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_simple = '#111')
print(settings.color_simple_)
```

[1]:
```
#111
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.color_simple_ = '#444'
print(settings.color_simple_)
```

[2]:
```
#444
```

Лучше так не делать – 1 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.color_simple_ = 1
print(settings.color_simple_)
```

[3]:
```
#666
```

– 2 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.color_simple_ = ()
print(settings.color_simple_)
```

[4]:
```
#666
```

`color_true: str = '#008001'`

Цвет текста содержащего положительную информацию (шестнадцатеричный код)

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.color_true, settings.color_true_)
```

[1]:
```
#008001 #008001
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_true = '#003332')
print(settings.color_true, settings.color_true_)
```

[2]:
```
#003332 #003332
```

– 3 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_true = '#42F')
print(settings.color_true, settings.color_true_)
```

[3]:
```
#42F #42F
```

Лучше так не делать – 1 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_true = 1)
print(settings.color_true, settings.color_true_)
```

[4]:
```
#008001 #008001
```

– 2 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_true = [])
print(settings.color_true, settings.color_true_)
```

[5]:
```
#008001 #008001
```

Type
str

property color_true_: str

Получение/установка цвета текста содержащего положительную информацию

Параметры
(str) – Шестнадцатеричный код

Результат
Шестнадцатеричный код

Тип результата
str

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(color_true = '#008001')
print(settings.color_true_)
```

[1]:
```
#008001
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.color_true_ = '#42F'
print(settings.color_true_)
```

[2]:
```
#42F
```

Лучше так не делать – 1 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.color_true = 1
print(settings.color_true)
```

[3]:
```
#008001
```

– 2 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.color_true_ = ()
print(settings.color_true_)
```

[4]:

```
#008001
```

property ext_: List[str]

Получение/установка расширений искомых файлов

Параметры
(List[str]) – Список с расширениями искомых файлов

Результат
Список с расширениями искомых файлов

Тип результата
List[str]

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.ext_)
```

[1]:
```
[]
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.ext_ = ['.mp4']
print(settings.ext_)
```

[2]:
```
['.mp4']
```

– 3 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.ext_ = ['.mp3', '.wav']
print(settings.ext_)
```

[3]:
```
['.mp3', '.wav']
```

– 4 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.ext_ = []
print(settings.ext_)
```

[4]:
```
[]
```

Лучше так не делать – 1 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.ext_ = [2, []]
print(settings.ext_)
```

[5]:
```
[]
```

property ignore_dirs_: List[str]

Получение/установка списка с директориями не входящими в выборку

Параметры
(List[str]) – Список с директориями

Результат
Список с директориями

Тип результата
List[str]

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.ignore_dirs_)
```

[1]:
```
[]
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.ignore_dirs_ = ['test', 'test_2']
print(settings.ignore_dirs_)
```

[2]:
```
['test', 'test_2']
```

– 3 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.ignore_dirs_ = []
print(settings.ignore_dirs_)
```

[3]:
```
[]
```

– 4 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.ext_ = ['1_a', '2_b']
print(settings.ext_)
```

[4]:
```
['1_a', '2_b']
```

Лучше так не делать – 1 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.ignore_dirs_ = [2, []]
print(settings.ignore_dirs_)
```

[5]:
```
[]
```

property keys_dataset_

Получение/установка названий ключей набора данных

Параметры
(List[str]) – Список с названиями ключей набора данных

Результат
Список с названиями ключей набора данных

Тип результата
List[str]

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.keys_dataset_)
```

[1]:
```
[
    'Path',
    'Openness',
    'Conscientiousness',
    'Extraversion',
    'Agreeableness',
    'Non-Neuroticism'
]
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.keys_dataset_ = ['P', 'O', 'C', 'E', 'A', 'N']
print(settings.keys_dataset_)
```

[2]:
```
['P', 'O', 'C', 'E', 'A', 'N']
```

Лучше так не делать – 1 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.keys_dataset_ = [{}, [], 1]
print(settings.keys_dataset_)
```

[3]:
```
[
    'Path',
    'Openness',
    'Conscientiousness',
    'Extraversion',
    'Agreeableness',
    'Non-Neuroticism'
]
```

– 2 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.keys_dataset_ = ['P', 'O']
print(settings.keys_dataset_)
```

[4]:
```
[
    'Path',
    'Openness',
    'Conscientiousness',
    'Extraversion',
    'Agreeableness',
    'Non-Neuroticism'
]
```

– 3 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.keys_dataset_ = []
print(settings.keys_dataset_)
```

[5]:
```
[
    'Path',
    'Openness',
    'Conscientiousness',
    'Extraversion',
    'Agreeableness',
    'Non-Neuroticism'
]
```

num_to_df_display: int = 30

Количество строк для отображения в таблицах

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.num_to_df_display, settings.num_to_df_display_)
```

[1]:
```
30 30
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(num_to_df_display = 30)
print(settings.num_to_df_display, settings.num_to_df_display_)
```

[2]:
```
30 30
```

– 3 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(num_to_df_display = 50)
print(settings.num_to_df_display, settings.num_to_df_display_)
```

[3]:
```
50 50
```

Лучше так не делать – 1 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(num_to_df_display = 0)
print(settings.num_to_df_display, settings.num_to_df_display_)
```

[4]:
```
30 30
```

– 2 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(num_to_df_display = 'какой-то_текст')
print(settings.num_to_df_display, settings.num_to_df_display_)
```

[5]:
```
30 30
```

Type
int

property num_to_df_display_: int

Получение/установка количества строк для отображения в таблицах

Параметры
(int) – Количество строк

Результат
Количество строк

Тип результата
int

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(num_to_df_display = 30)
print(settings.num_to_df_display_)
```

[1]:
```
30
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.num_to_df_display_ = 50
print(settings.num_to_df_display_)
```

[2]:
```
50
```

Лучше так не делать – 1 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.num_to_df_display_ = 0
print(settings.num_to_df_display_)
```

[3]:
```
30
```

– 2 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.num_to_df_display_ = ()
print(settings.num_to_df_display_)
```

[4]:
```
30
```

property path_to_dataset_: str

Получение/установка директории набора данных

Параметры
(str) – Директория набора данных

Результат
Директория набора данных

Тип результата
str

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.path_to_dataset_)
```

[1]:
```
.
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.path_to_dataset_ = './dataset'
print(settings.path_to_dataset_)
```

[2]:
```
dataset
```

– 3 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.path_to_dataset_ = ''
print(settings.path_to_dataset_)
```

[3]:
```
.
```

Лучше так не делать – 1 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.path_to_dataset_ = [2, []]
print(settings.path_to_dataset_)
```

[4]:
```
.
```

– 2 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.path_to_dataset_ = 1
print(settings.path_to_dataset_)
```

[5]:
```
.
```

property path_to_logs_: str

Получение/установка директории для сохранения LOG файлов

Параметры
(str) – Директория для сохранения LOG файлов

Результат
Директория для сохранения LOG файлов

Тип результата
str

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.path_to_logs_)
```

[1]:
```
logs
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.path_to_logs_ = './logs/DF'
print(settings.path_to_logs_)
```

[2]:
```
logs/DF
```

– 3 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.path_to_logs_ = ''
print(settings.path_to_logs_)
```

[3]:
```
.
```

Лучше так не делать – 1 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.path_to_logs_ = [2, []]
print(settings.path_to_logs_)
```

[4]:
```
logs
```

– 2 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.path_to_logs_ = {'a': 1, 'b': 2}
print(settings.path_to_logs_)
```

[5]:
```
logs
```

property path_to_save_: str

Получение/установка директории для сохранения данных

Параметры
(str) – Директория для сохранения данных

Результат
Директория для сохранения данных

Тип результата
str

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.path_to_save_)
```

[1]:
```
models
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.path_to_save_ = './models/Audio'
print(settings.path_to_save_)
```

[2]:
```
models/Audio
```

– 3 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.path_to_save_ = ''
print(settings.path_to_save_)
```

[3]:
```
.
```

Лучше так не делать – 1 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.path_to_save_ = [2, []]
print(settings.path_to_save_)
```

[4]:
```
models
```

– 2 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.path_to_save_ = {'a': 1, 'b': 2}
print(settings.path_to_save_)
```

[5]:

```
models
```

text_runtime: str = ''

Текст времени выполнения

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
print(settings.text_runtime, settings.text_runtime_)
```

[1]:
```
Время выполнения Время выполнения
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(text_runtime = 'Код выполнился за')
print(settings.text_runtime, settings.text_runtime_)
```

[2]:
```
Код выполнился за Код выполнился за
```

– 3 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(text_runtime = 'Время выполнения')
print(settings.text_runtime, settings.text_runtime_)
```

[3]:
```
Время выполнения Время выполнения
```

Лучше так не делать – 1 –

In [4]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(text_runtime = 1)
print(settings.text_runtime, settings.text_runtime_)
```

[4]:
```
Время выполнения Время выполнения
```

– 2 –

In [5]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(text_runtime = {1, 2, 3})
print(settings.text_runtime, settings.text_runtime_)
```

[5]:
```
Время выполнения Время выполнения
```

Type
str

property text_runtime_: str

Получение/установка текста времени выполнения

Параметры
(str) – Текст

Результат
Текст

Тип результата
str

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.settings import Settings

settings = Settings(text_runtime = 'Время выполнения')
print(settings.text_runtime_)
```

[1]:
```
Время выполнения
```

– 2 –

In [2]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.text_runtime_ = 'Код выполнился за'
print(settings.text_runtime_)
```

[2]:
```
Код выполнился за
```

Лучше так не делать – 1 –

In [3]:
```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.text_runtime_ = 1
print(settings.text_runtime_)
```

[3]:
```
Время выполнения
```

– 2 –

In [4]:

```
from oceanai.modules.core.settings import Settings

settings = Settings()
settings.text_runtime_ = ()
print(settings.text_runtime_)
```

[4]:
```
Время выполнения
```

Ядро

class oceanai.modules.core.core.CoreMessages(*lang: str = 'ru', color_simple: str = '#666', color_info: str = '#1776D2', color_err: str = '#FF0000', color_true: str = '#008001', bold_text: bool = True, text_runtime: str = '', num_to_df_display: int = 30*)

Базовые классы: *Settings*

Класс для сообщений

Параметры

- lang (*str*) – Смотреть *lang*
- color_simple (*str*) – Смотреть *color_simple*
- color_info (*str*) – Смотреть *color_info*
- color_err (*str*) – Смотреть *color_err*
- color_true (*str*) – Смотреть *color_true*
- bold_text (*bool*) – Смотреть *bold_text*
- num_to_df_display (*int*) – Смотреть *num_to_df_display*
- text_runtime (*str*) – Смотреть *text_runtime*

class oceanai.modules.core.core.Core(*lang: str = 'ru', color_simple: str = '#666', color_info: str = '#1776D2', color_err: str = '#FF0000', color_true: str = '#008001', bold_text: bool = True, text_runtime: str = '', num_to_df_display: int = 30*)

Базовые классы: *CoreMessages*

Класс-ядро модулей

Параметры

- lang (*str*) – Смотреть *lang*
- color_simple (*str*) – Смотреть *color_simple*
- color_info (*str*) – Смотреть *color_info*
- color_err (*str*) – Смотреть *color_err*
- color_true (*str*) – Смотреть *color_true*
- bold_text (*bool*) – Смотреть *bold_text*
- num_to_df_display (*int*) – Смотреть *num_to_df_display*

- `text_runtime` (*str*) – Смотреть *text_runtime*

`__is_notebook()` → bool

Определение запуска библиотеки в Jupyter или аналогах

Примечание: private (приватный метод)

Результат
True если библиотека запущена в Jupyter или аналогах, в обратном случае **False**

Тип результата
bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
core._Core__is_notebook()
```

[1]:
```
True
```

– 2 –

In [2]:
```
from oceanai.modules.core.core import Core

Core._Core__is_notebook()
```

[2]:
```
True
```

`_add_last_el_notebook_history_output(`*message: str*`)` → None

Добавление текста к последнему сообщению из истории вывода сообщений в ячейке Jupyter

Примечание: protected (защищенный метод)

Параметры
`message` (*str*) – Сообщение

Результат
None

Тип результата
None

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core._add_last_el_notebook_history_output(message = '...')

core._add_notebook_history_output(
    message = 'Сообщение 1', last = False
)
core._add_last_el_notebook_history_output(message = '...')

core.show_notebook_history_output()
```

[1]:
```
...
Сообщение 1 ...
```

`_add_notebook_history_output`(*message: str*, *last: bool = False*) → None

Добавление истории вывода сообщений в ячейке Jupyter

Примечание: protected (защищенный метод)

Параметры

- `message` (*str*) – Сообщение
- `last` (*bool*) – Замена последнего сообщения

Результат

None

Тип результата

None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core._add_notebook_history_output(
    message = 'Сообщение 1', last = False
)
core._add_notebook_history_output(
    message = 'Сообщение 2', last = False
)
core._add_notebook_history_output(
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    message = 'Замена последнего сообщения', last = True
)

core.show_notebook_history_output()
```

[1]:

```
Сообщение 1
Замена последнего сообщения
```

– 2 –

In [2]:

```
from oceanai.modules.core.core import Core

core = Core()

for message, last in zip(
    [
        'Сообщение 1',
        'Сообщение 2',
        'Замена последнего сообщения'
    ],
    [False, False, True]
):
    core._add_notebook_history_output(
        message = message, last = last
    )

core.show_notebook_history_output()
```

[2]:

```
Сообщение 1
Замена последнего сообщения
```

_append_to_list_of_accuracy(*preds: List[Optional[float]]*, *out: bool = True*) → bool

Добавление значений в словарь для DataFrame с результатами вычисления точности

Примечание: protected (защищенный метод)

Параметры

- preds (*List[Optional[float]]*) – Предсказания персональных качеств
- out (*bool*) – Отображение

Результат

True если значения в словарь для DataFrame были добавлены, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core.keys_dataset_ = ['O', 'C', 'E', 'A', 'N']

core._append_to_list_of_accuracy(
    preds = [0.5, 0.6, 0.2, 0.1, 0.8],
    out = True
)

core._append_to_list_of_accuracy(
    preds = [0.4, 0.5, 0.1, 0, 0.7],
    out = True
)

core.dict_of_accuracy_
```

[1]:
```
{
    'O': [0.5, 0.4],
    'C': [0.6, 0.5],
    'E': [0.2, 0.1],
    'A': [0.1, 0],
    'N': [0.8, 0.7]
}
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()

core.keys_dataset_ = ['O', 'C', 'E', 'A', 'N']

core._append_to_list_of_accuracy(
    preds = [0.5, 0.6, 0.2, 0.1, 0.8],
    out = True
)

core.keys_dataset_ = ['O2', 'C2', 'E2', 'A2', 'N2']

core._append_to_list_of_accuracy(
    preds = [0.4, 0.5, 0.1, 0, 0.7],
    out = True
)

core.dict_of_accuracy_
```

– 2 –

[2]:
```
[2022-12-03 23:08:15] Ой! Что-то пошло не так ... смотрите настройки ядра и цепочку выполнения действий ...

    Файл: /Users/dl/GitHub/OCEANAI.private/oceanai/modules/core/core.py
    Линия: 2669
    Метод: _append_to_list_of_accuracy
    Тип ошибки: KeyError

{
    'O': [0.5, 0.4],
    'C': [0.6, 0.5],
    'E': [0.2, 0.1],
    'A': [0.1, 0],
    'N': [0.8, 0.7]
}
```

_append_to_list_of_files(*path: str*, *preds: List[Optional[float]]*, *out: bool = True*) → bool

Добавление значений в словарь для DataFrame с данными

Примечание: protected (защищенный метод)

Параметры

- path (*str*) – Путь к файлу
- preds (*List[Optional[float]]*) – Предсказания персональных качеств
- out (*bool*) – Отображение

Результат

True если значения в словарь для DataFrame были добавлены, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core.keys_dataset_ = ['P', 'O', 'C', 'E', 'A', 'N']

core._append_to_list_of_files(
    path = './6V807Mf_gHM.003.mp4',
    preds = [0.5, 0.6, 0.2, 0.1, 0.8],
    out = True
)

core._append_to_list_of_files(
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    path = './6V807Mf_gHM.004.mp4',
    preds = [0.4, 0.5, 0.1, 0, 0.7],
    out = True
)

core.dict_of_files_
```

[1]:

```
{
    'P': ['./6V807Mf_gHM.003.mp4', './6V807Mf_gHM.004.mp4'],
    'O': [0.5, 0.4],
    'C': [0.6, 0.5],
    'E': [0.2, 0.1],
    'A': [0.1, 0],
    'N': [0.8, 0.7]
}
```

Ошибка – 1 –

In [2]:

```
from oceanai.modules.core.core import Core

core = Core()

core.keys_dataset_ = ['P', 'O', 'C', 'E', 'A', 'N']

core._append_to_list_of_files(
    path = './6V807Mf_gHM.003.mp4',
    preds = [0.5, 0.6, 0.2, 0.1, 0.8],
    out = True
)

core.keys_dataset_ = ['P2', 'O2', 'C2', 'E2', 'A2', 'N2']

core._append_to_list_of_files(
    path = './6V807Mf_gHM.004.mp4',
    preds = [0.4, 0.5, 0.1, 0, 0.7],
    out = True
)

core.dict_of_files_
```

– 2 –

[2]:

```
[2022-10-13 18:22:40] Ой! Что-то пошло не так ... смотрите настройки ядра и␣
↪цепочку выполнения действий ...

    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/core/core.py
    Линия: 1105
    Метод: _append_to_list_of_files
    Тип ошибки: KeyError

{
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    'P': ['./6V807Mf_gHM.003.mp4'],
    'O': [0.5],
    'C': [0.6],
    'E': [0.2],
    'A': [0.1],
    'N': [0.8]
}
```

_bold_wrapper(*message: str*) → str

Обернутое сообщение с жирным начертанием

Примечание: protected (защищенный метод)

Параметры
message (*str*) – Сообщение

Результат
Обернутое сообщение с жирным начертанием

Тип результата
str

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core(bold_text = False)
print(core._bold_wrapper(
    'Обернутое сообщение без жирного начертания'
))

core.bold_text = True
print(core._bold_wrapper(
    'Обернутое сообщение с жирным начертанием'
))
```

[1]:
```
<span style="color:#666">Обернутое сообщение без жирного начертания</span>
<span style="color:#666">**Обернутое сообщение с жирным начертанием**</span>
```

_candidate_ranking(*df_files: Optional[DataFrame] = None*, *weigths_openness: int = 0*, *weigths_conscientiousness: int = 0*, *weigths_extraversion: int = 0*, *weigths_agreeableness: int = 0*, *weigths_non_neuroticism: int = 0*, *out: bool = True*) → DataFrame

Ранжирование кандидатов по профессиональным обязанностям

Примечание: protected (защищенный метод)

Параметры

- `df_files` (*pd.DataFrame*) – **DataFrame** с данными
- `weigths_openness` (*int*) – Вес для ранжирования персонального качества (открытость опыту)
- `weigths_conscientiousness` (*int*) – Вес для ранжирования персонального качества (добросовестность)
- `weigths_extraversion` (*int*) – Вес для ранжирования персонального качества (экстраверсия)
- `weigths_agreeableness` (*int*) – Вес для ранжирования персонального качества (доброжелательность)
- `weigths_non_neuroticism` (*int*) – Вес для ранжирования персонального качества (эмоциональная стабильность)
- `out` (*bool*) – Отображение

Результат
DataFrame с ранжированными данными

Тип результата
pd.DataFrame

`_clear_notebook_history_output()` → None

Очистка истории вывода сообщений в ячейке Jupyter

Примечание: protected (защищенный метод)

Результат
None

Тип результата
None

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core._add_notebook_history_output(
    message = 'Сообщение 1', last = False
)
core._add_notebook_history_output(
    message = 'Сообщение 2', last = False
)

core._clear_notebook_history_output()

core.show_notebook_history_output()
```

[1]:
```
1
```

_colleague_personality_desorders(*df_files: Optional[DataFrame] = None*, *correlation_coefficients_mbti: Optional[DataFrame] = None*, *correlation_coefficients_disorders: Optional[DataFrame] = None*, *personality_desorder_number: int = 3*, *col_name_ocean: str = 'Trait'*, *threshold: float = 0.55*, *out: bool = True*) → DataFrame

Определение приоритетных профессиональных растройств по версии MBTI

Примечание: protected (защищенный метод)

Параметры

- `df_files` (*pd.DataFrame*) – **DataFrame** с данными
- `correlation_coefficients_mbti` (*pd.DataFrame*) – **DataFrame** с коэффициентами корреляции для MBTI
- `correlation_coefficients_disorders` (*pd.DataFrame*) – **DataFrame** с коэффициентами корреляции для расстройств
- `target_scores` (*List[float]*) – Список оценок персональных качеств личности целевого человека
- `personality_desorder_number` (*int*) – Количество приоритетных расстройств
- `threshold` (*float*) – Порог для оценок полярности качеств (например, интроверт < 0.55, экстраверт > 0.55)
- `out` (*bool*) – Отображение
- `col_name_ocean` (*str*) –

Результат
DataFrame с приоритетными расстройствами

Тип результата
pd.DataFrame

_colleague_personality_type_match(*df_files: Optional[DataFrame] = None*, *correlation_coefficients: Optional[DataFrame] = None*, *target_scores: List[float] = [0.47, 0.63, 0.35, 0.58, 0.51]*, *col_name_ocean: str = 'Trait'*, *threshold: float = 0.55*, *out: bool = True*) → DataFrame

Поиск коллег по совместимости персональных типов по версии MBTI

Примечание: protected (защищенный метод)

Параметры

- `df_files` (*pd.DataFrame*) – **DataFrame** с данными
- `correlation_coefficients` (*pd.DataFrame*) – **DataFrame** с коэффициентами корреляции

- `target_scores` (*List[float]*) – Список оценок персональных качеств личности целевого человека
- `threshold` (*float*) – Порог для оценок полярности качеств (например, интроверт < 0.55, экстраверт > 0.55)
- `out` (*bool*) – Отображение
- `col_name_ocean` (*str*) –

Результат
DataFrame с совместимостью коллег по персональным типам по версии MBTI

Тип результата
pd.DataFrame

`_colleague_ranking`(*df_files: Optional[DataFrame] = None*, *correlation_coefficients: Optional[DataFrame] = None*, *target_scores: List[float] = [0.47, 0.63, 0.35, 0.58, 0.51]*, *colleague: str = 'major'*, *equal_coefficients: float = 0.5*, *out: bool = True*) → DataFrame

Поиск подходящего коллеги

Примечание: protected (защищенный метод)

Параметры

- `df_files` (*pd.DataFrame*) – **DataFrame** с данными
- `correlation_coefficients` (*pd.DataFrame*) – **DataFrame** с коэффициентами корреляции
- `target_scores` (*List[float]*) – Список оценок персональных качеств личности целевого человека
- `colleague` (*str*) – Ранг коллеги по совместимости
- `equal_coefficients` (*float*) – Коэффициент применяемый к оценкам в случае равенства оценок двух человек
- `out` (*bool*) – Отображение

Результат
DataFrame с ранжированными коллегами

Тип результата
pd.DataFrame

`_compatibility_percentage`(*type1*, *type2*)

`_create_folder_for_logs`(*out: bool = True*)

Создание директории для сохранения LOG файлов

Примечание: protected (защищенный метод)

Параметры
`out` (*bool*) – Отображение

Результат
True если директория создана или существует, в обратном случае **False**

Тип результата
bool

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core.path_to_logs_ = './logs'

core._create_folder_for_logs(out = True)
```

[1]:
```
true
```

_del_last_el_notebook_history_output() → None
Удаление последнего сообщения из истории вывода сообщений в ячейке Jupyter

Примечание: protected (защищенный метод)

Результат
None

Тип результата
None

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core._add_notebook_history_output(
    message = 'Сообщение 1', last = False
)
core._add_notebook_history_output(
    message = 'Сообщение 2', last = False
)

core._del_last_el_notebook_history_output()

core.show_notebook_history_output()
```

[1]:

```
Сообщение 1
```

_error(*message: str*, *last: bool = False*, *out: bool = True*) → None

Сообщение об ошибке

Примечание: protected (защищенный метод)

Параметры

- `message` (*str*) – Сообщение
- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение

Результат

None

Тип результата

None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core._error(
    message = 'Сообщение об ошибке 1',
    last = False, out = True
)

core.color_simple_ = '#FFF'
core.color_err_ = 'FF0000'
core.bold_text_ = False

core._error(
    message = 'Сообщение об ошибке 2',
    last = True, out = True
)
```

[1]:
```
[2022-10-12 15:21:00] Сообщение об ошибке 1
[2022-10-12 15:21:00] Сообщение об ошибке 2
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()

```

(continues on next page)

(продолжение с предыдущей страницы)

```
core._error(
    message = '',
    last = False, out = True
)
```

[2]:
```
[2022-10-12 17:06:04] Неверные типы или значения аргументов в "Core._error" ...
```

_error_wrapper(*message: str*) → str

Обернутое сообщение об ошибке

Примечание: protected (защищенный метод)

Параметры

message (*str*) – Сообщение

Результат

Обернутое сообщение об ошибке

Тип результата

str

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
print(core._error_wrapper(
    'Обернутое сообщение об ошибке 1'
))

core.color_err_ = '#FF4545'
print(core._error_wrapper(
    'Обернутое сообщение об ошибке 2'
))
```

[1]:
```
<span style="color:#FF0000">Обернутое сообщение об ошибке 1</span>
<span style="color:#FF4545">Обернутое сообщение об ошибке 2</span>
```

_get_paths(*path: Iterable*, *depth: int = 1*, *out: bool = True*) → Union[List[str], bool]

Получение директорий где хранятся данные

Примечание: protected (защищенный метод)

Параметры

- path (*Iterable*) – Директория набора данных

- `depth` (*int*) – Глубина иерархии для извлечения классов
- `out` (*bool*) – Отображение

Результат
False если проверка аргументов не удалась или список с директориями

Тип результата
Union[List[str], bool]

Примеры

Верно – 1 –

In [1]:
```
core = Core()
core._get_paths(
    path = '/Users/dl/GitHub/oceanai/oceanai/dataset',
    depth = 1, out = True
)
```

[1]:
```
[
    '/Users/dl/GitHub/oceanai/oceanai/dataset/test80_01',
    '/Users/dl/GitHub/oceanai/oceanai/dataset/1',
    '/Users/dl/GitHub/oceanai/oceanai/dataset/test80_17'
]
```

Ошибки – 1 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()
core._get_paths(
    path = '',
    depth = 1, out = True
)
```

[2]:
```
[2022-10-12 16:36:16] Неверные типы или значения аргументов в "Core._get_paths"...
False
```

– 2 –

In [3]:
```
from oceanai.modules.core.core import Core

core = Core()
core._get_paths(
    path = '/Users/dl/GitHub/oceanai/oceanai/folder',
    depth = 1, out = True
)
```

[3]:
```
[2022-10-13 18:37:11] Ой! Что-то пошло не так ... директория "/Users/dl/GitHub/oceanai/oceanai/folder" не найдена ...
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/core/core.py
    Линия: 1023
    Метод: _get_paths
    Тип ошибки: FileNotFoundError

False
```

_info(*message: str*, *last: bool = False*, *out: bool = True*) → None

Информационное сообщение

Примечание: protected (защищенный метод)

Параметры

- message (*str*) – Сообщение
- last (*bool*) – Замена последнего сообщения
- out (*bool*) – Отображение

Результат

None

Тип результата

None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core._info(
    message = 'Информационное сообщение 1',
    last = False, out = True
)

core.color_simple_ = '#FFF'
core.color_info_ = '#0B45B9'
core.bold_text_ = False

core._info(
    message = 'Информационное сообщение 2',
    last = True, out = True
)
```

[1]:
```
[2022-10-14 11:35:00] Информационное сообщение 1
[2022-10-14 11:35:00] Информационное сообщение 2
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()

core._info(
    message = '',
    last = False, out = True
)
```

[2]:
```
[2022-10-14 11:43:00] Неверные типы или значения аргументов в "Core._info" ...
```

_info_true(*message: str*, *last: bool = False*, *out: bool = True*) → None

Положительная информация

Примечание: protected (защищенный метод)

Параметры

- message (*str*) – Сообщение
- last (*bool*) – Замена последнего сообщения
- out (*bool*) – Отображение

Результат
None

Тип результата
None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core._info_true(
    message = 'Информационное положительное сообщение 1',
    last = False, out = True
)

core.color_true_ = '#008001'
core.bold_text_ = False

core._info_true(
    message = 'Информационное положительное сообщение 2',
    last = True, out = True
)
```

[1]:
```
Информационное положительное сообщение 1

Информационное положительное сообщение 2
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()

core._info_true(
    message = '',
    last = False, out = True
)
```

[2]:
```
[2022-10-22 16:46:56] Неверные типы или значения аргументов в "Core._info_true"␣
↪...
```

_info_wrapper(*message: str*) → str

Обернутое информационное сообщение

Примечание: protected (защищенный метод)

Параметры
message (*str*) – Сообщение

Результат
Обернутое информационное сообщение

Тип результата
str

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
print(core._info_wrapper('Обернутое информационное сообщение 1'))

core.color_info_ = '#0B45B9'
print(core._info_wrapper('Обернутое информационное сообщение 2'))
```

[1]:
```
<span style="color:#1776D2">Обернутое информационное сообщение 1</span>
<span style="color:#0B45B9">Обернутое информационное сообщение 2</span>
```

_inv_args(*class_name: str*, *build_name: str*, *last: bool = False*, *out: bool = True*) → None

Сообщение об указании неверных типов аргументов

Примечание: protected (защищенный метод)

Параметры

- `class_name` (*str*) – Имя класса
- `build_name` (*str*) – Имя метода/функции
- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение

Результат
None

Тип результата
None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
core._inv_args(
    Core.__name__, core._info.__name__,
    last = False, out = True
)
```

[1]:
```
[2022-10-14 11:58:04] Неверные типы или значения аргументов в "Core._info" ...
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()
core._inv_args(1, '', last = False, out = True)
```

[2]:
```
[2022-10-14 11:58:04] Неверные типы или значения аргументов в "Core._inv_args" .
↪..
```

`_metadata_info`(*last: bool = False*, *out: bool = True*) → None

Информация об библиотеке

Примечание: protected (защищенный метод)

Параметры

- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение

Результат
None

Тип результата
None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
core._metadata_info(last = False, out = True)
```

[1]:
```
[2022-10-14 13:05:54] oceanai - персональные качества личности человека:
    Авторы:
        Рюмина Елена [ryumina_ev@mail.ru]
        Рюмин Дмитрий [dl_03.03.1991@mail.ru]
        Карпов Алексей [karpov@iias.spb.su]
    Сопровождающие:
        Рюмина Елена [ryumina_ev@mail.ru]
        Рюмин Дмитрий [dl_03.03.1991@mail.ru]
    Версия: 1.0.0-a7
    Лицензия: GPLv3
```

Лучше так не делать – 1 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()
core._metadata_info(last = 1, out = [])
```

[2]:
```
[2022-10-14 13:05:54] oceanai - персональные качества личности человека:
    Авторы:
        Рюмина Елена [ryumina_ev@mail.ru]
        Рюмин Дмитрий [dl_03.03.1991@mail.ru]
        Карпов Алексей [karpov@iias.spb.su]
    Сопровождающие:
        Рюмина Елена [ryumina_ev@mail.ru]
        Рюмин Дмитрий [dl_03.03.1991@mail.ru]
    Версия: 1.0.0-a7
    Лицензия: GPLv3
```

`_notebook_display_markdown`(*message: str*, *last: bool = False*, *out: bool = True*) → None

Отображение сообщения

Примечание: protected (защищенный метод)

Параметры

- `message` (*str*) – Сообщение
- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение

Результат
None

Тип результата
None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
core._notebook_display_markdown('Сообщение')
```

[1]:
```
Сообщение
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()
core._notebook_display_markdown(1)
```

[2]:
```
[2022-10-14 15:52:03] Неверные типы или значения аргументов в "Core._notebook_
↪display_markdown" ...
```

`_other_error`(*message: str*, *last: bool = False*, *out: bool = True*) → None
Сообщение об прочей ошибке

Примечание: protected (защищенный метод)

Параметры

- `message` (*str*) – Сообщение
- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение

Результат
None

Тип результата
None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

try: raise Exception
except:
    core._other_error(
        message = 'Сообщение об ошибке 1',
        last = False, out = True
    )

core.color_simple_ = '#FFF'
core.color_err_ = 'FF0000'
core.bold_text_ = False

try: raise Exception
except:
    core._other_error(
        message = 'Сообщение об ошибке 2',
        last = True, out = True
    )
```

[1]:
```
[2022-10-14 16:25:11] Сообщение об ошибке 1

    Файл: /var/folders/gw/w3k5kxtx0s3_nqdqw94zr8yh0000gn/T/ipykernel_20011/
↪333478077.py
    Линия: 5
    Метод: <cell line: 5>
    Тип ошибки: Exception

[2022-10-14 16:25:11] Сообщение об ошибке 2

    Файл: /var/folders/gw/w3k5kxtx0s3_nqdqw94zr8yh0000gn/T/ipykernel_20011/
↪333478077.py
    Линия: 16
    Метод: <cell line: 16>
    Тип ошибки: Exception
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()

try: raise Exception
except:
    core._other_error(
        message = '',
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        last = False, out = True
    )
```

[2]:
```
[2022-10-14 16:25:11] Неверные типы или значения аргументов в "Core._other_error
↪" ...
```

_priority_calculation(*df_files: Optional[DataFrame] = None*, *correlation_coefficients: Optional[DataFrame] = None*, *col_name_ocean: str = 'Trait'*, *threshold: float = 0.55*, *number_priority: int = 1*, *number_importance_traits: int = 1*, *out: bool = True*) → DataFrame

Ранжирование предпочтений

Примечание: protected (защищенный метод)

Параметры

- `df_files` (*pd.DataFrame*) – **DataFrame** с данными
- `correlation_coefficients` (*pd.DataFrame*) – **DataFrame** с коэффициентами корреляции
- `col_name_ocean` (*str*) – Столбец с названиями персональных качеств личности человека
- `threshold` (*float*) – Порог для оценок полярности качеств (например, интроверт < 0.55, экстраверт > 0.55)
- `number_priority` (*int*) – Количество приоритетных предпочтений
- `number_importance_traits` (*int*) – Количество наиболее важных персональных качеств личности человека
- `out` (*bool*) – Отображение

Результат
DataFrame с ранжированными предпочтениями

Тип результата
pd.DataFrame

_priority_skill_calculation(*df_files: Optional[DataFrame] = None*, *correlation_coefficients: Optional[DataFrame] = None*, *threshold: float = 0.55*, *out: bool = True*) → DataFrame

Ранжирование кандидатов по профессиональным навыкам

Примечание: protected (защищенный метод)

Параметры

- `df_files` (*pd.DataFrame*) – **DataFrame** с данными
- `correlation_coefficients` (*pd.DataFrame*) – **DataFrame** с коэффициентами корреляции

- `threshold` (*float*) – Порог для оценок полярности качеств (например, интроверт < 0.55, экстраверт > 0.55)
- `out` (*bool*) – Отображение

Результат
DataFrame с ранжированными кандидатами

Тип результата
pd.DataFrame

`_professional_match`(*df_files: Optional[DataFrame] = None*, *correlation_coefficients: Optional[DataFrame] = None*, *personality_type: Optional[str] = None*, *col_name_ocean: str = 'Trait'*, *threshold: float = 0.55*, *out: bool = True*) → DataFrame

Ранжирование кандидатов по одному из шестнадцати персональных типов по версии MBTI

Примечание: protected (защищенный метод)

Параметры

- `df_files` (*pd.DataFrame*) – **DataFrame** с данными
- `correlation_coefficients` (*pd.DataFrame*) – **DataFrame** с коэффициентами корреляции
- `personality_type` (*str*) – Персональный тип по версии MBTI
- `threshold` (*float*) – Порог для оценок полярности качеств (например, интроверт < 0.55, экстраверт > 0.55)
- `out` (*bool*) – Отображение
- `col_name_ocean` (*str*) –

Результат
DataFrame с ранжированными кандидатами

Тип результата
pd.DataFrame

`_progressbar`(*message: str*, *progress: str*, *clear_out: bool = True*, *last: bool = False*, *out: bool = True*) → None

Индикатор выполнения

Примечание: protected (защищенный метод)

Параметры

- `message` (*str*) – Сообщение
- `progress` (*str*) – Индикатор выполнения
- `clear_out` (*bool*) – Очистка области вывода
- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение

Результат
None

Тип результата
None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

for cnt in range(1, 4):
    core._progressbar(
        message = 'Цикл действий',
        progress = 'Итерация ' + str(cnt),
        clear_out = False,
        last = False, out = True
    )
```

[1]:
```
[2022-10-14 16:52:20] Цикл действий

    Итерация 1

[2022-10-14 16:52:20] Цикл действий

    Итерация 2

[2022-10-14 16:52:20] Цикл действий

    Итерация 3
```

– 2 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()

for cnt in range(1, 4):
    core._progressbar(
        message = 'Цикл действий',
        progress = 'Итерация ' + str(cnt),
        clear_out = True,
        last = True, out = True
    )
```

[2]:
```
[2022-10-14 16:52:20] Цикл действий

    Итерация 3
```

Ошибка – 1 –

In [3]:
```
from oceanai.modules.core.core import Core

core = Core()

for cnt in range(1, 4):
    core._progressbar(
        message = 1,
        progress = 2,
        clear_out = True,
        last = False, out = True
    )
```

[3]:
```
[2022-10-14 16:52:38] Неверные типы или значения аргументов в "Core._progressbar
↪" ...
```

_progressbar_union_predictions(*message: str*, *item: int*, *info: str*, *len_paths: int*, *clear_out: bool = True*, *last: bool = False*, *out: bool = True*) → None

Индикатор выполнения получения прогнозов по аудио

Примечание: private (приватный метод)

Параметры

- `message` (*str*) – Сообщение
- `item` (*int*) – Номер видеофайла
- `info` (*str*) – Локальный путь
- `len_paths` (*int*) – Количество видеофайлов
- `clear_out` (*bool*) – Очистка области вывода
- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение

Результат
None

Тип результата
None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

l = range(1, 4, 1)
```

(continues on next page)

(продолжение с предыдущей страницы)

```

for progress in l:
    core._progressbar_union_predictions(
        message = 'Цикл действий',
        item = progress,
        info = 'Путь к файлу',
        len_paths = len(l),
        clear_out = False,
        last = False, out = True
    )
```

[1]:
```
[2022-10-20 16:51:49] Цикл действий

    1 из 3 (33.33%) ... Путь к файлу ...

[2022-10-20 16:51:49] Цикл действий

    2 из 3 (66.67%) ... Путь к файлу ...

[2022-10-20 16:51:49] Цикл действий

    3 из 3 (100.0%) ... Путь к файлу ...
```

– 2 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()

l = range(1, 4, 1)

for progress in l:
    core._progressbar_union_predictions(
        message = 'Цикл действий',
        item = progress,
        info = 'Путь к файлу',
        len_paths = len(l),
        clear_out = True,
        last = True, out = True
    )
```

[2]:
```
[2022-10-20 16:51:55] Цикл действий

    3 из 3 (100.0%) ... Путь к файлу ...
```

Ошибка – 1 –

In [3]:
```
from oceanai.modules.core.core import Core

core = Core()

```

(continues on next page)

(продолжение с предыдущей страницы)

```
l = range(1, 4, 1)

for progress in l:
    core._progressbar_union_predictions(
        message = 1,
        item = progress,
        info = 'Путь к файлу',
        len_paths = len(l),
        clear_out = True,
        last = False, out = True
    )
```

[3]:

```
[2022-10-20 16:55:15] Неверные типы или значения аргументов в "Audio._
↪progressbar_union_predictions" ...
```

_r_end(*last: bool = False*, *out: bool = True*) → None

Конец отсчета времени выполнения

Примечание: protected (защищенный метод)

Подсказка: Работает в связке с `_r_start()`

Параметры

- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение

Результат
None

Тип результата
None

Примеры

Верно – 1 –

In [1]:

```
from oceanai.modules.core.core import Core

core = Core()

core._r_start()
for cnt in range(0, 10000000): res = cnt * 2
core._r_end()
```

[1]:

```
--- Время выполнения: 0.819 сек. ---
```

Ошибка – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

for cnt in range(0, 10000000): res = cnt * 2
core._r_end()
```

[1]:
```
--- Время выполнения: 1665756222.704 сек. ---
```

`_r_start()` → None

Начало отсчета времени выполнения

Примечание: protected (защищенный метод)

Подсказка: Работает в связке с `_r_end()`

Результат
None

Тип результата
None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core._r_start()
for cnt in range(0, 10000000): res = cnt * 2
core._r_end()
```

[1]:
```
--- Время выполнения: 0.819 сек. ---
```

Ошибка – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

for cnt in range(0, 10000000): res = cnt * 2
core._r_end()
```

[1]:
```
--- Время выполнения: 1665756222.704 сек. ---
```

_round_math(*val: Union[int, float]*, *out: bool = True*) → Union[int, bool]

Округление чисел по математическому закону

Примечание: protected (защищенный метод)

Параметры

- `val` (*Union[int, float]*) – Число для округления
- `out` (*bool*) – Отображение

Результат

Округленное число если ошибок не выявлено, в обратном случае **False**

Тип результата

Union[int, bool]

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core._round_math(4.5)
```

[1]:
```
5
```

– 2 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core._round_math(-2.5)
```

[1]:
```
-3
```

Ошибка – 1 –

In [3]:
```
from oceanai.modules.core.core import Core

core = Core()

core._round_math('')
```

[3]:
```
[2022-11-03 15:52:30] Неверные типы или значения аргументов в "Core._round_math
↪" ...

False
```

_save_logs(*df: DataFrame*, *name: str*, *out: bool = True*) → bool

Сохранение LOG файла

Примечание: protected (защищенный метод)

Параметры

- `df` (*pd.DataFrame*) – DataFrame который будет сохранен в LOG файл
- `name` (*str*) – Имя LOG файла
- `out` (*bool*) – Отображение

Результат

True если LOG файл сохранен, в обратном случае **False**

Тип результата

bool

Пример

Верно – 1 –

In [1]:
```
import pandas as pd
from oceanai.modules.core.core import Core

df = pd.DataFrame.from_dict(
    data = {'Test': [1, 2, 3]}
)

core = Core()

core.path_to_logs_ = './logs'

core._save_logs(
    df = df, name = 'test', out = True
)
```

[1]:
```
True
```

_search_file(*path_to_file: str*, *ext: str*, *create: bool = False*, *out: bool = True*) → bool

Поиск файла

Примечание: protected (защищенный метод)

Параметры

- `path_to_file` (*str*) – Путь к файлу
- `ext` (*str*) – Расширение файла
- `create` (*bool*) – Создание файла в случае его отсутствия
- `out` (*bool*) – Печатать процесс выполнения

Результат
True если файл найден, в обратном случае **False**

Тип результата
bool

_stat_acoustic_features(*last: bool = False*, *out: bool = True*, ***kwargs: Union[int, Tuple[int], TensorShape]*) → None

Сообщение со статистикой извлеченных признаков из акустического сигнала

Примечание: protected (защищенный метод)

Параметры

- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение
- `**kwargs` (*Union[int, Tuple[int], tf.TensorShape]*) – Дополнительные именованные аргументы

Результат
None

Тип результата
None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core(
    color_simple = '#FFF',
    color_info = '#1776D2',
    bold_text = True,
)

core._stat_acoustic_features(
    last = False, out = True,
    len_hc_features = 12,
    len_melspectrogram_features = 12,
    shape_hc_features = [196, 25],
    shape_melspectrogram_features = [224, 224, 3],
)
```

[1]:
```
[2022-10-14 17:59:20] Статистика извлеченных признаков из акустического сигнала:
    Общее количество сегментов с:
        1. экспертными признаками: 12
        2. лог мел-спектрограммами: 12
    Размерность матрицы экспертных признаков одного сегмента: 196 × 25
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    Размерность тензора с лог мел-спектрограммами одного сегмента: 224 × 224 ×␣
↪3
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core(
    color_simple = '#FFF',
    color_info = '#1776D2',
    bold_text = True,
)

core._stat_acoustic_features(
    last = False, out = True
)
```

[2]:
```
[2022-10-14 17:59:21] Неверные типы или значения аргументов в "Core._stat_
↪acoustic_features" ...
```

`_stat_text_features`(*last: bool = False*, *out: bool = True*, ***kwargs: Union[int, Tuple[int], TensorShape]*) → None

Сообщение с статистикой извлеченных признаков из текста

Примечание: protected (защищенный метод)

Параметры

- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение
- `**kwargs` (*Union[int, Tuple[int], tf.TensorShape]*) – Дополнительные именованные аргументы

Результат
None

Тип результата
None

`_stat_visual_features`(*last: bool = False*, *out: bool = True*, ***kwargs: Union[int, Tuple[int], TensorShape]*) → None

Сообщение с статистикой извлеченных признаков из визуального сигнала

Примечание: protected (защищенный метод)

Параметры

- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение

- **kwargs (*Union[int, Tuple[int], tf.TensorShape]*) – Дополнительные именованные аргументы

Результат
None

Тип результата
None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core(
    color_simple = '#FFF',
    color_info = '#1776D2',
    bold_text = True,
)

core._stat_visual_features(
    last = False, out = True,
    len_hc_features = 23,
    len_nn_features = 23,
    shape_hc_features = [10, 115],
    shape_nn_features = [10, 512],
    fps_before = 30,
    fps_after = 10
)
```

[1]:
```
[2022-11-03 16:18:40] Статистика извлеченных признаков из визуального сигнала:
    Общее количество сегментов с:
        1. экспертными признаками: 23
        2. нейросетевыми признаками: 23
    Размерность матрицы экспертных признаков одного сегмента: 10 × 115
    Размерность тензора с нейросетевыми признаками одного сегмента: 10 × 512
    Понижение кадровой частоты: с 30 до 10
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core(
    color_simple = '#FFF',
    color_info = '#1776D2',
    bold_text = True,
)

core._stat_visual_features(
    last = False, out = True
)
```

[2]:
```
[2022-11-03 16:19:35] Неверные типы или значения аргументов в "Core._stat_
↪visual_features" ...
```

static _traceback() → Dict

Трассировка исключений

Примечание: protected (защищенный метод)

Результат
Словарь с описанием исключения

Тип результата
Dict

Пример

Верно – 1 –

In [1]:
```
import pprint
from oceanai.modules.core.core import Core

core = Core()

try: raise Exception
except:
    pp = pprint.PrettyPrinter(compact = True)
    pp.pprint(core._traceback())
```

[1]:
```
{
    'filename': '/var/folders/gw/w3k5kxtx0s3_nqdqw94zr8yh0000gn/T/ipykernel_
↪22253/4179594971.py',
    'lineno': 6,
    'name': '<cell line: 6>',
    'type': 'Exception'
}
```

property df_accuracy_: DataFrame

Получение DataFrame с результатами вычисления точности

Результат
DataFrame с результатами вычисления точности

Тип результата
pd.DataFrame

Пример

Верно – 1 –

```
In [1]:
from oceanai.modules.core.core import Core

core = Core()
len(core.df_accuracy_)
```

```
[1]:
0
```

property df_files_: DataFrame

Получение DataFrame с данными

Результат
DataFrame с данными

Тип результата
pd.DataFrame

Пример

Верно – 1 –

```
In [1]:
from oceanai.modules.core.core import Core

core = Core()
len(core.df_files_)
```

```
[1]:
0
```

property df_files_MBTI_colleague_match_: DataFrame

Получение DataFrame с ранжированными коллегами на основе MBTI

Результат
DataFrame с данными

Тип результата
pd.DataFrame

property df_files_MBTI_disorders_: DataFrame

Получение DataFrame с ранжированными профессиональными расстройствами на основе MBTI

Результат
DataFrame с данными

Тип результата
pd.DataFrame

property df_files_MBTI_job_match_: DataFrame

Получение DataFrame с ранжированными кандидатами на основе MBTI

Результат
DataFrame с данными

Тип результата
pd.DataFrame

property df_files_colleague_: DataFrame

Получение DataFrame с ранжированными коллегами на основе данных

Результат
DataFrame с данными

Тип результата
pd.DataFrame

property df_files_priority_: DataFrame

Получение DataFrame с ранжированными предпочтениями на основе данных

Результат
DataFrame с данными

Тип результата
pd.DataFrame

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
len(core.df_files_priority_)
```

[1]:
```
0
```

property df_files_priority_skill_: DataFrame

Получение DataFrame с ранжированными коллегами на основе данных

Результат
DataFrame с данными

Тип результата
pd.DataFrame

property df_files_ranking_: DataFrame

Получение DataFrame с ранжированными данными

Результат
DataFrame с данными

Тип результата
pd.DataFrame

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
len(core.df_files_ranking_)
```

[1]:
```
0
```

property df_pkgs_: DataFrame

Получение DataFrame с версиями установленных библиотек

Результат
DataFrame с версиями установленных библиотек

Тип результата
pd.DataFrame

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
core.libs_vers(out = False, runtime = True, run = True)
core.df_pkgs_
```

[1]:
```
|----|--------------|---------|
|    | Package      | Version |
|----|--------------|---------|
| 1  | TensorFlow   | 2.11.0  |
| 2  | Keras        | 2.11.0  |
| 3  | OpenCV       | 4.6.0   |
| 4  | MediaPipe    | 0.9.0   |
| 5  | NumPy        | 1.23.5  |
| 6  | SciPy        | 1.9.3   |
| 7  | Pandas       | 1.5.2   |
| 8  | Scikit-learn | 1.1.3   |
| 9  | OpenSmile    | 2.4.1   |
| 10 | Librosa      | 0.9.2   |
| 11 | AudioRead    | 3.0.0   |
| 12 | IPython      | 8.7.0   |
| 14 | Requests     | 2.28.1  |
| 15 | JupyterLab   | 3.5.0   |
|----|--------------|---------|
```

property dict_of_accuracy_: Dict[str, List[Union[int, float]]]

Получение словаря для DataFrame с результатами вычисления точности

Подсказка: На основе данного словаря формируется DataFrame с данными `df_accuracy_`

Результат
Словарь для DataFrame с результатами вычисления точности

Тип результата
Dict[str, List[Union[int, float]]]

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
len(core.dict_of_accuracy_)
```

[1]:
```
0
```

`property dict_of_files_: Dict[str, List[Union[int, str, float]]]`

Получение словаря для DataFrame с данными

Подсказка: На основе данного словаря формируется DataFrame с данными `df_files_`

Результат
Словарь для DataFrame с данными

Тип результата
Dict[str, List[Union[int, str, float]]]

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
len(core.dict_of_files_)
```

[1]:
```
0
```

`property is_notebook_: bool`

Получение результата определения запуска библиотеки в Jupyter или аналогах

Результат
True если библиотека запущена в Jupyter или аналогах, в обратном случае **False**

Тип результата
bool

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
print(core.is_notebook_)
```

[1]:
```
True
```

libs_vers(*out: bool = True*, *runtime: bool = True*, *run: bool = True*) → None

Получение и отображение версий установленных библиотек

Параметры

- out (*bool*) – Отображение
- runtime (*bool*) – Подсчет времени выполнения
- run (*bool*) – Блокировка выполнения

Результат

None

Тип результата

None

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
core.libs_vers(out = True, runtime = True, run = True)
```

[1]:
```
|----|---------------|---------|
|    | Package       | Version |
|----|---------------|---------|
| 1  | TensorFlow    | 2.11.0  |
| 2  | Keras         | 2.11.0  |
| 3  | OpenCV        | 4.6.0   |
| 4  | MediaPipe     | 0.9.0   |
| 5  | NumPy         | 1.23.5  |
| 6  | SciPy         | 1.9.3   |
| 7  | Pandas        | 1.5.2   |
| 8  | Scikit-learn  | 1.1.3   |
| 9  | OpenSmile     | 2.4.1   |
| 10 | Librosa       | 0.9.2   |
| 11 | AudioRead     | 3.0.0   |
| 12 | IPython       | 8.7.0   |
| 14 | Requests      | 2.28.1  |
| 15 | JupyterLab    | 3.5.0   |
```

(continues on next page)

(продолжение с предыдущей страницы)

```
| 16 | LIWC          | 0.5.0   |
| 17 | Transformers  | 4.24.0  |
| 18 | Sentencepiece | 0.1.99  |
| 19 | Torch         | 1.12.1  |
| 20 | Torchaudio    | 0.12.1  |
|----|---------------|---------|
--- Время выполнения: 0.005 сек. ---
```

– 2 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()
core.libs_vers(out = True, runtime = True, run = False)
```

[2]:
```
[2022-10-15 18:17:27] Выполнение заблокировано пользователем ...
```

Ошибка – 1 –

In [3]:
```
from oceanai.modules.core.core import Core

core = Core()
core.libs_vers(out = True, runtime = True, run = 1)
```

[3]:
```
[2022-10-15 18:18:51] Неверные типы или значения аргументов в "Core.libs_vers" ...
```

property runtime_

Получение времени выполнения

Результат

Время выполнения

Тип результата

Union[int, float]

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()

core._r_start()
for cnt in range(0, 10000000): res = cnt * 2
core._r_end(out = False)

print(core.runtime_)
```

[1]:

```
0.838
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.core.core import Core

core = Core()

print(core.runtime_)
```

[2]:
```
-1
```

– 2 –

In [3]:
```
from oceanai.modules.core.core import Core

core = Core()

core._r_start()
for cnt in range(0, 10000000): res = cnt * 2

print(core.runtime_)
```

[3]:
```
-1
```

show_notebook_history_output() → None

Отображение истории вывода сообщений в ячейке Jupyter

Результат
None

Тип результата
None

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
core._info(
    message = 'Информационное сообщение',
    last = False, out = False
)

core.show_notebook_history_output()
```

[1]:
```
[2022-10-15 18:27:46] Информационное сообщение
```

property true_traits_: Dict[str, str]

Получение путей к верным предсказаниям для подсчета точности

Результат

Словарь с путями к верным предсказаниям для подсчета точности

Тип результата

Dict

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
core.true_traits_
```

[1]:
```
{
    'sberdisk': 'https://download.sberdisk.ru/download/file/410305241?token=TFePK6w5CW6ADnq&filename=data_true_traits.csv'
}
```

property weights_for_big5_: Dict[str, Dict]

Получение весов для нейросетевых архитектур

Результат

Словарь с весами для нейросетевых архитектур

Тип результата

Dict

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.core.core import Core

core = Core()
core.weights_for_big5_
```

[1]:
```
{
    'audio': {
            'hc': {
                'sberdisk': 'https://download.sberdisk.ru/download/file/400635799?token=MMRrak8fMsyzxLE&filename=weights_2022-05-05_11-27-55.h5',
            },
            'nn': {
                'sberdisk': 'https://download.sberdisk.ru/download/file/400635678?token=W6LCtD33FQHnYEz&filename=weights_2022-05-03_07-46-14.h5',
            },
            'b5': {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                'openness': {
                        'sberdisk': 'https://download.sberdisk.ru/download/file/
↪405035301?token=443WRA9MFWqWBAE&filename=weights_2022-06-15_16-16-20.h5',
                },
                'conscientiousness': {
                        'sberdisk': 'https://download.sberdisk.ru/download/file/
↪405034601?token=eDG28m3H6c8bWoE&filename=weights_2022-06-15_16-21-57.h5',
                },
                'extraversion': {
                        'sberdisk': 'https://download.sberdisk.ru/download/file/
↪405034830?token=3daBSTYnmZaesee&filename=weights_2022-06-15_16-26-41.h5',
                },
                'agreeableness': {
                        'sberdisk': 'https://download.sberdisk.ru/download/file/
↪405034397?token=52ZPHMjb4CFmdYa&filename=weights_2022-06-15_16-32-51.h5',
                },
                'non_neuroticism': {
                        'sberdisk': 'https://download.sberdisk.ru/download/file/
↪405035156?token=q8CZJ99rZqcNxkM&filename=weights_2022-06-15_16-37-46.h5',
                },
            },
    },
    'video': {
            'hc': {
                'sberdisk': 'https://download.sberdisk.ru/download/file/
↪412059444?token=JXerCfAjJZg6crD&filename=weights_2022-08-27_18-53-35.h5',
            },
            'nn': {
                'sberdisk': 'https://download.sberdisk.ru/download/file/
↪412059478?token=85KeW6q4QKy6kP8&filename=weights_2022-03-22_16-31-48.h5',
            },
            'fe': {
                'sberdisk': 'https://download.sberdisk.ru/download/file/
↪414207833?token=ygzxWEkndjSMnEL&filename=weights_2022-11-01_12-27-07.h5'
            },
            'b5': {
                'openness': {
                        'sberdisk': 'https://download.sberdisk.ru/download/file/
↪415127050?token=rfpy9TLdbeXtiN7&filename=weights_2022-06-15_16-46-30.h5',
                },
                'conscientiousness': {
                        'sberdisk': 'https://download.sberdisk.ru/download/file/
↪415126986?token=PnjzaHaR3wPg2RT&filename=weights_2022-06-15_16-48-50.h5',
                },
                'extraversion': {
                        'sberdisk': 'https://download.sberdisk.ru/download/file/
↪415127012?token=s5aTwbt8DBkt7G4&filename=weights_2022-06-15_16-54-06.h5',
                },
                'agreeableness': {
                        'sberdisk': 'https://download.sberdisk.ru/download/file/
↪415126845?token=joN7TMHk59Gffsf&filename=weights_2022-06-15_17-02-03.h5',
                },
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                'non_neuroticism': {
                        'sberdisk': 'https://download.sberdisk.ru/download/file/
↪415127032?token=NEBSsE7mjyjen3o&filename=weights_2022-06-15_17-06-15.h5',
                }
            }
    }
}
```

Обработка архивов

class oceanai.modules.lab.unzip.UnzipMessages(*lang: str = 'ru'*, *color_simple: str = '#666'*, *color_info: str = '#1776D2'*, *color_err: str = '#FF0000'*, *color_true: str = '#008001'*, *bold_text: bool = True*, *text_runtime: str = ''*, *num_to_df_display: int = 30*)

Базовые классы: *Core*

Класс для сообщений

Параметры

- lang (*str*) – Смотреть *lang*
- color_simple (*str*) – Смотреть *color_simple*
- color_info (*str*) – Смотреть *color_info*
- color_err (*str*) – Смотреть *color_err*
- color_true (*str*) – Смотреть *color_true*
- bold_text (*bool*) – Смотреть *bold_text*
- num_to_df_display (*int*) – Смотреть *num_to_df_display*
- text_runtime (*str*) – Смотреть *text_runtime*

class oceanai.modules.lab.unzip.Unzip(*lang: str = 'ru'*, *color_simple: str = '#666'*, *color_info: str = '#1776D2'*, *color_err: str = '#FF0000'*, *color_true: str = '#008001'*, *bold_text: bool = True*, *text_runtime: str = ''*, *num_to_df_display: int = 30*)

Базовые классы: *UnzipMessages*

Класс для обработки архивов

Параметры

- lang (*str*) – Смотреть *lang*
- color_simple (*str*) – Смотреть *color_simple*
- color_info (*str*) – Смотреть *color_info*
- color_err (*str*) – Смотреть *color_err*
- color_true (*str*) – Смотреть *color_true*
- bold_text (*bool*) – Смотреть *bold_text*
- num_to_df_display (*int*) – Смотреть *num_to_df_display*

- `text_runtime` (*str*) – Смотреть *text_runtime*

`__progressbar_unzip`(*path_to_zipfile: str*, *progress: float*, *clear_out: bool = True*, *last: bool = False*, *out: bool = True*) → None

Индикатор выполнения

Примечание: private (приватный метод)

Параметры

- `path_to_zipfile` (*str*) – Путь до архива
- `progress` (*float*) – Процент выполнения (от **0.0** до **100.0**)
- `clear_out` (*bool*) – Очистка области вывода
- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение

Тип результата
None

`_unzip`(*path_to_zipfile: str*, *new_name: Optional[str] = None*, *force_reload: bool = True*, *out: bool = True*) → bool

Разархивирование архива (без очистки истории вывода сообщений в ячейке Jupyter)

Примечание: protected (защищенный метод)

Параметры

- `path_to_zipfile` (*str*) – Полный путь до архива
- `new_name` (*str*) – Имя директории для разархивирования
- `force_reload` (*bool*) – Принудительное разархивирование
- `out` (*bool*) – Отображение

Результат
True если разархивирование прошло успешно, в обратном случае **False**

Тип результата
bool

`property path_to_unzip: str`

Получение директории для разархивирования

Результат
Директория для разархивирования

Тип результата
str

`unzip`(*path_to_zipfile: str*, *new_name: Optional[str] = None*, *force_reload: bool = True*, *out: bool = True*) → bool

Разархивирование архива

Параметры

- `path_to_zipfile` (*str*) – Полный путь до архива
- `new_name` (*str*) – Имя директории для разархивирования
- `force_reload` (*bool*) – Принудительное разархивирование
- `out` (*bool*) – Отображение

Результат
True если разархивирование прошло успешно, в обратном случае **False**

Тип результата
bool

Загрузка файлов

`class oceanai.modules.lab.download.DownloadMessages(`*lang: str = 'ru', color_simple: str = '#666', color_info: str = '#1776D2', color_err: str = '#FF0000', color_true: str = '#008001', bold_text: bool = True, text_runtime: str = '', num_to_df_display: int = 30*`)`

Базовые классы: *Unzip*

Класс для сообщений

Параметры

- `lang` (*str*) – Смотреть *lang*
- `color_simple` (*str*) – Смотреть *color_simple*
- `color_info` (*str*) – Смотреть *color_info*
- `color_err` (*str*) – Смотреть *color_err*
- `color_true` (*str*) – Смотреть *color_true*
- `bold_text` (*bool*) – Смотреть *bold_text*
- `num_to_df_display` (*int*) – Смотреть *num_to_df_display*
- `text_runtime` (*str*) – Смотреть *text_runtime*

`class oceanai.modules.lab.download.Download(`*lang: str = 'ru', color_simple: str = '#666', color_info: str = '#1776D2', color_err: str = '#FF0000', color_true: str = '#008001', bold_text: bool = True, text_runtime: str = '', num_to_df_display: int = 30*`)`

Базовые классы: *DownloadMessages*

Класс для загрузки файлов

Параметры

- `lang` (*str*) – Смотреть *lang*
- `color_simple` (*str*) – Смотреть *color_simple*
- `color_info` (*str*) – Смотреть *color_info*
- `color_err` (*str*) – Смотреть *color_err*
- `color_true` (*str*) – Смотреть *color_true*

- bold_text (*bool*) – Смотреть *bold_text*
- num_to_df_display (*int*) – Смотреть *num_to_df_display*
- text_runtime (*str*) – Смотреть *text_runtime*

__progressbar_download_file_from_url(*url_filename: str*, *progress: float*, *clear_ out: bool = True*, *last: bool = False*, *out: bool = True*) → None

Индикатор выполнения загрузки файла из URL

Примечание: private (приватный метод)

Параметры

- url_filename (*str*) – Путь до файла
- progress (*float*) – Процент выполнения (от **0.0** до **100.0**)
- clear_out (*bool*) – Очистка области вывода
- last (*bool*) – Замена последнего сообщения
- out (*bool*) – Отображение

Результат
None

Тип результата
None

Примеры

Верно – 1 –

In [1]:
```
import numpy as np
from oceanai.modules.lab.download import Download

download = Download()

for progress in np.arange(0., 101, 25):
    download._Download__progressbar_download_file_from_url(
        url_filename = 'https://clck.ru/32Nwdk',
        progress = float(progress),
        clear_out = False,
        last = False, out = True
    )
```

[1]:
```
[2022-10-16 16:58:51] Загрузка файла "https://clck.ru/32Nwdk" (0.0%) ...

[2022-10-16 16:58:51] Загрузка файла "https://clck.ru/32Nwdk" (25.0%) ...

[2022-10-16 16:58:51] Загрузка файла "https://clck.ru/32Nwdk" (50.0%) ...

[2022-10-16 16:58:51] Загрузка файла "https://clck.ru/32Nwdk" (75.0%) ...
```

(continues on next page)

(продолжение с предыдущей страницы)

```
[2022-10-16 16:58:51] Загрузка файла "https://clck.ru/32Nwdk" (100.0%) ...
```

– 2 –

In [2]:
```
import numpy as np
from oceanai.modules.lab.download import Download

download = Download()

for progress in np.arange(0., 101, 25):
    download._Download__progressbar_download_file_from_url(
        url_filename = 'https://clck.ru/32Nwdk',
        progress = float(progress),
        clear_out = True,
        last = True, out = True
    )
```

[2]:
```
[2022-10-16 16:59:41] Загрузка файла "https://clck.ru/32Nwdk" (100.0%) ...
```

Ошибка – 1 –

In [3]:
```
import numpy as np
from oceanai.modules.lab.download import Download

download = Download()

for progress in np.arange(0., 101, 25):
    download._Download__progressbar_download_file_from_url(
        url_filename = 'https://clck.ru/32Nwdk',
        progress = 101,
        clear_out = True,
        last = False, out = True
    )
```

[3]:
```
[2022-10-16 17:00:11] Неверные типы или значения аргументов в "Download.__
↪progressbar_download_file_from_url" ...
```

_download_file_from_url(*url: str*, *force_ reload: bool = True*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → int

Загрузка файла из URL (без очистки истории вывода сообщений в ячейке Jupyter)

Примечание: protected (защищенный метод)

Параметры

- `url` (*str*) – Полный путь к файлу
- `force_reload` (*bool*) – Принудительная загрузка файла из сети
- `out` (*bool*) – Отображение

- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

Код статуса ответа:

- 200 - Файл загружен
- 400 - Ошибка при проверке аргументов
- 403 - Выполнение заблокировано пользователем
- 404 - Не удалось скачать файл

Тип результата
int

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.download import Download

download = Download()

download.path_to_save_ = './models'
download.chunk_size_ = 2000000

res_download_file_from_url = download._download_file_from_url(
    url = 'https://download.sberdisk.ru/download/file/400635799?
↪token=MMRrak8fMsyzxLE&filename=weights_2022-05-05_11-27-55.h5',
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[1]:
```
[2022-10-16 20:23:25] Загрузка файла "weights_2022-05-05_11-27-55.h5" (100.0%) .
↪..

--- Время выполнения: 0.373 сек. ---

200
```

– 2 –

In [2]:
```
from oceanai.modules.lab.download import Download

download = Download()

download.path_to_save_ = './models'
download.chunk_size_ = 2000000

res_download_file_from_url = download._download_file_from_url(
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    url = 'https://clck.ru/32Nwdk',
    force_reload = True,
    out = True,
    runtime = True,
    run = False
)
res_download_file_from_url
```

[2]:

```
[2022-10-16 19:33:05] Выполнение заблокировано пользователем ...

403
```

Ошибки – 1 –

In [3]:

```
from oceanai.modules.lab.download import Download

download = Download()

download.path_to_save_ = './models'
download.chunk_size_ = 2000000

res_download_file_from_url = download._download_file_from_url(
    url = 1,
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
res_download_file_from_url
```

[3]:

```
[2022-10-16 19:33:01] Неверные типы или значения аргументов в "Download._
↪download_file_from_url" ...

400
```

– 2 –

In [4]:

```
from oceanai.modules.lab.download import Download

download = Download()

download.path_to_save_ = './models'
download.chunk_size_ = 2000000

res_download_file_from_url = download._download_file_from_url(
    url = 'https://',
    force_reload = True,
    out = True,
    runtime = True,
    run = True
```

(continues on next page)

(продолжение с предыдущей страницы)

```
)
res_download_file_from_url
```

[4]:

```
[2022-10-16 19:33:10] Что-то пошло не так ... не удалось обработать указанный URL ...

    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/download.py
    Линия: 257
    Метод: _download_file_from_url
    Тип ошибки: InvalidURL

--- Время выполнения: 0.061 сек. ---

404
```


In [5]:

```
from oceanai.modules.lab.download import Download

download = Download()

download.path_to_save_ = './models'
download.chunk_size_ = 2000000

res_download_file_from_url = download._download_file_from_url(
    url = 'https://www.iconfinder.com/icons/4375050/download/svg/4096',
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
res_download_file_from_url
```

[5]:

```
[2022-10-16 19:33:15] Загрузка файла "4375050_logo_python_icon.svg"

[2022-10-16 19:33:15] Что-то пошло не так ... Не определен размер файла для загрузки ...

    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/download.py
    Линия: 324
    Метод: _download_file_from_url
    Тип ошибки: InvalidContentLength

--- Время выполнения: 0.386 сек. ---

404
```

download_file_from_url(*url: str*, *force_reload: bool = True*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → int

Загрузка файла из URL

Параметры

- `url` (*str*) – Полный путь к файлу
- `force_reload` (*bool*) – Принудительная загрузка файла из сети
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

Код статуса ответа:

- `200` - Файл загружен
- `400` - Ошибка при проверке аргументов
- `403` - Выполнение заблокировано пользователем
- `404` - Не удалось скачать файл

Тип результата
int

Пример

Аудио

`class oceanai.modules.lab.audio.AudioMessages(`*lang: str = 'ru', color_simple: str = '#666', color_info: str = '#1776D2', color_err: str = '#FF0000', color_true: str = '#008001', bold_text: bool = True, text_runtime: str = '', num_to_df_display: int = 30*`)`

Базовые классы: *Download*

Класс для сообщений

Параметры

- `lang` (*str*) – Смотреть *lang*
- `color_simple` (*str*) – Смотреть *color_simple*
- `color_info` (*str*) – Смотреть *color_info*
- `color_err` (*str*) – Смотреть *color_err*
- `color_true` (*str*) – Смотреть *color_true*
- `bold_text` (*bool*) – Смотреть *bold_text*
- `num_to_df_display` (*int*) – Смотреть *num_to_df_display*
- `text_runtime` (*str*) – Смотреть *text_runtime*

`class oceanai.modules.lab.audio.Audio(`*lang: str = 'ru', color_simple: str = '#666', color_info: str = '#1776D2', color_err: str = '#FF0000', color_true: str = '#008001', bold_text: bool = True, text_runtime: str = '', num_to_df_display: int = 30*`)`

Базовые классы: *AudioMessages*

Класс для обработки аудио

Параметры

- lang (*str*) – Смотреть *lang*
- color_simple (*str*) – Смотреть *color_simple*
- color_info (*str*) – Смотреть *color_info*
- color_err (*str*) – Смотреть *color_err*
- color_true (*str*) – Смотреть *color_true*
- bold_text (*bool*) – Смотреть *bold_text*
- num_to_df_display (*int*) – Смотреть *num_to_df_display*
- text_runtime (*str*) – Смотреть *text_runtime*

__concat_pred(*pred_hc: ndarray*, *pred_melspectrogram: ndarray*, *out: bool = True*) → List[Optional[ndarray]]

Конкатенация оценок по экспертным и нейросетевым признакам

Примечание: private (приватный метод)

Параметры

- pred_hc (*np.ndarray*) – Оценки по экспертным признакам
- pred_melspectrogram (*np.ndarray*) – Оценки по нейросетевым признакам
- out (*bool*) – Отображение

Результат
Конкатенированные оценки по экспертным и нейросетевым признакам

Тип результата
List[Optional[np.ndarray]]

Примеры

Верно – 1 –

In [1]:
```
import numpy as np
from oceanai.modules.lab.audio import Audio

audio = Audio()

arr_hc = np.array([
    [0.64113516, 0.6217892, 0.54451424, 0.6144415, 0.59334993],
    [0.6652424, 0.63606125, 0.572305, 0.63169795, 0.612515]
])

arr_melspectrogram = np.array([
    [0.56030345, 0.7488746, 0.44648764, 0.59893465, 0.5701077],
    [0.5900006, 0.7652722, 0.4795154, 0.6409055, 0.6088242]
])

audio._Audio__concat_pred(
    pred_hc = arr_hc,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    pred_melspectrogram = arr_melspectrogram,
    out = True
)
```

[1]:

```
[
    array([
            0.64113516, 0.6652424, 0.65318878, 0.65318878, 0.65318878,
            0.65318878, 0.65318878, 0.65318878, 0.65318878, 0.65318878,
            0.65318878, 0.65318878, 0.65318878, 0.65318878, 0.65318878,
            0.65318878, 0.56030345, 0.5900006, 0.57515202, 0.57515202,
            0.57515202, 0.57515202, 0.57515202, 0.57515202, 0.57515202,
            0.57515202, 0.57515202, 0.57515202, 0.57515202, 0.57515202,
            0.57515202, 0.57515202
    ]),
    array([
            0.6217892, 0.63606125, 0.62892523, 0.62892523, 0.62892523,
            0.62892523, 0.62892523, 0.62892523, 0.62892523, 0.62892523,
            0.62892523, 0.62892523, 0.62892523, 0.62892523, 0.62892523,
            0.62892523, 0.7488746, 0.7652722, 0.7570734, 0.7570734,
            0.7570734, 0.7570734, 0.7570734, 0.7570734, 0.7570734,
            0.7570734, 0.7570734, 0.7570734, 0.7570734, 0.7570734,
            0.7570734, 0.7570734
    ]),
    array([
            0.54451424, 0.572305, 0.55840962, 0.55840962, 0.55840962,
            0.55840962, 0.55840962, 0.55840962, 0.55840962, 0.55840962,
            0.55840962, 0.55840962, 0.55840962, 0.55840962, 0.55840962,
            0.55840962, 0.44648764, 0.4795154, 0.46300152, 0.46300152,
            0.46300152, 0.46300152, 0.46300152, 0.46300152, 0.46300152,
            0.46300152, 0.46300152, 0.46300152, 0.46300152, 0.46300152,
            0.46300152, 0.46300152
    ]),
    array([
            0.6144415, 0.63169795, 0.62306972, 0.62306972, 0.62306972,
            0.62306972, 0.62306972, 0.62306972, 0.62306972, 0.62306972,
            0.62306972, 0.62306972, 0.62306972, 0.62306972, 0.62306972,
            0.62306972, 0.59893465, 0.6409055, 0.61992008, 0.61992008,
            0.61992008, 0.61992008, 0.61992008, 0.61992008, 0.61992008,
            0.61992008, 0.61992008, 0.61992008, 0.61992008, 0.61992008,
            0.61992008, 0.61992008
    ]),
    array([
            0.59334993, 0.612515, 0.60293247, 0.60293247, 0.60293247,
            0.60293247, 0.60293247, 0.60293247, 0.60293247, 0.60293247,
            0.60293247, 0.60293247, 0.60293247, 0.60293247, 0.60293247,
            0.60293247, 0.5701077, 0.6088242, 0.58946595, 0.58946595,
            0.58946595, 0.58946595, 0.58946595, 0.58946595, 0.58946595,
            0.58946595, 0.58946595, 0.58946595, 0.58946595, 0.58946595,
            0.58946595, 0.58946595
    ])
]
```

Ошибка – 1 –

In [2]:
```
import numpy as np
from oceanai.modules.lab.audio import Audio

audio = Audio()

arr_hc = np.array([
    [0.64113516, 0.6217892, 0.54451424, 0.6144415],
    [0.6652424, 0.63606125, 0.572305, 0.63169795, 0.612515]
])

arr_melspectrogram = np.array([
    [0.56030345, 0.7488746, 0.44648764, 0.59893465, 0.5701077],
    [0.5900006, 0.7652722, 0.4795154, 0.6409055, 0.6088242]
])

audio._Audio__concat_pred(
    pred_hc = arr_hc,
    pred_melspectrogram = arr_melspectrogram,
    out = True
)
```

[3]:
```
[2022-10-20 22:33:31] Что-то пошло не так ... конкатенация оценок по экспертным␣
↪и нейросетевым
признакам не произведена (аудио модальность) ...

[]
```

__load_audio_model_b5(*show_summary: bool = False*, *out: bool = True*) → Optional[Model]

Формирование нейросетевой архитектуры модели для получения результата оценки персонального качества

Примечание: private (приватный метод)

Параметры

- `show_summary` (*bool*) – Отображение сформированной нейросетевой архитектуры модели
- `out` (*bool*) – Отображение

Результат

None если неверные типы или значения аргументов, в обратном случае нейросетевая модель **tf.keras.Model** для получения результата оценки персонального качества

Тип результата

Optional[tf.keras.Model]

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio._Audio__load_audio_model_b5(
    show_summary = True, out = True
)
```

[1]:
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32)]              0

 dense_1 (Dense)             (None, 1)                 33

 activ_1 (Activation)        (None, 1)                 0

=================================================================
Total params: 33
Trainable params: 33
Non-trainable params: 0
_________________________________________________________________
<tf.keras.Model at 0x13d442940>
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio._Audio__load_audio_model_b5(
    show_summary = True, out = []
)
```

[3]:
```
[2022-10-17 10:53:03] Неверные типы или значения аргументов в "Audio.__load_
↪audio_model_b5" ...
```

__load_model_weights(*url: str*, *force_ reload: bool = True*, *info_ text: str = ''*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Загрузка весов нейросетевой модели

Примечание: private (приватный метод)

Параметры

- `url` (*str*) – Полный путь к файлу с весами нейросетевой модели

- `force_reload` (*bool*) – Принудительная загрузка файла с весами нейросетевой модели из сети
- `info_text` (*str*) – Текст для информационного сообщения
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат
True если веса нейросетевой модели загружены, в обратном случае **False**

Тип результата
bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.path_to_save_ = './models'
audio.chunk_size_ = 2000000

audio._Audio__load_model_weights(
    url = 'https://download.sberdisk.ru/download/file/400635799?token=MMRrak8fMsyzxLE&filename=weights_2022-05-05_11-27-55.h5',
    force_reload = True,
    info_text = 'Загрузка весов нейросетевой модели',
    out = True, runtime = True, run = True
)
```

[1]:
```
[2022-10-17 12:21:48] Загрузка весов нейросетевой модели

[2022-10-17 12:21:48] Загрузка файла "weights_2022-05-05_11-27-55.h5" (100.0%) ...

--- Время выполнения: 0.439 сек. ---

True
```

– 2 –

In [2]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.path_to_save_ = './models'
audio.chunk_size_ = 2000000

audio._Audio__load_model_weights(
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    url = './models/weights_2022-05-05_11-27-55.h5',
    force_reload = True,
    info_text = 'Загрузка весов нейросетевой модели',
    out = True, runtime = True, run = True
)
```

[2]:

```
[2022-10-17 12:21:50] Загрузка весов нейросетевой модели

--- Время выполнения: 0.002 сек. ---

True
```

Ошибка – 1 –

In [3]:

```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.path_to_save_ = './models'
audio.chunk_size_ = 2000000

audio._Audio__load_model_weights(
    url = 'https://download.sberdisk.ru/download/file/400635799?token=MMRrak8fMsyzxLE&filename=weights_2022-05-05_11-27-55.h5',
    force_reload = True, info_text = '',
    out = True, runtime = True, run = True
)
```

[3]:

```
[2022-10-17 12:21:57] Неверные типы или значения аргументов в "Audio.__load_model_weights" ...

False
```

__norm_pred(*pred_data: ndarray*, *len_spec: int = 16*, *out: bool = True*) → ndarray

Нормализация оценок по экспертным и нейросетевым признакам

Примечание: private (приватный метод)

Параметры

- pred_data (*np.ndarray*) – Оценки
- len_spec (*int*) – Максимальный размер вектора оценок
- out (*bool*) – Отображение

Результат

Нормализованные оценки по экспертным и нейросетевым признакам

Тип результата

np.ndarray

Примеры

Верно – 1 –

In [1]:
```
import numpy as np
from oceanai.modules.lab.audio import Audio

audio = Audio()

arr = np.array([
    [0.64113516, 0.6217892, 0.54451424, 0.6144415, 0.59334993],
    [0.6652424, 0.63606125, 0.572305, 0.63169795, 0.612515]
])

audio._Audio__norm_pred(
    pred_data = arr,
    len_spec = 4,
    out = True
)
```

[1]:
```
array([
    [0.64113516, 0.6217892 , 0.54451424, 0.6144415 , 0.59334993],
    [0.6652424 , 0.63606125, 0.572305  , 0.63169795, 0.612515],
    [0.65318878, 0.62892523, 0.55840962, 0.62306972, 0.60293247],
    [0.65318878, 0.62892523, 0.55840962, 0.62306972, 0.60293247]
])
```

Ошибка – 1 –

In [2]:
```
import numpy as np
from oceanai.modules.lab.audio import Audio

audio = Audio()

arr = np.array([])

audio._Audio__norm_pred(
    pred_data = arr,
    len_spec = 4,
    out = True
)
```

[3]:
```
[2022-10-20 22:03:17] Неверные типы или значения аргументов в "Audio.__norm_pred
↪" ...

array([], dtype=float64)
```

`__smile()` → Smile

Извлечение функций OpenSmile

Примечание: private (приватный метод)

Результат
Извлеченные функции OpenSmile

Тип результата
opensmile.core.smile.Smile

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()
audio._Audio__smile()
```

[1]:
```
{
    '$opensmile.core.smile.Smile': {
        'feature_set': 'eGeMAPSv02',
        'feature_level': 'LowLevelDescriptors',
        'options': {},
        'sampling_rate': None,
        'channels': [0],
        'mixdown': False,
        'resample': False
    }
}
```

_get_acoustic_features(*path: str*, *sr: int = 44100*, *window: Union[int, float] = 2.0*, *step: Union[int, float] = 1.0*, *last: bool = False*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → Tuple[List[Optional[ndarray]], List[Optional[ndarray]]]

Извлечение признаков из акустического сигнала (без очистки истории вывода сообщений в ячейке Jupyter)

Примечание: protected (защищенный метод)

Параметры

- `path` (*str*) – Путь к аудио или видеофайлу
- `sr` (*int*) – Частота дискретизации
- `window` (*Union[int, float]*) – Размер окна сегмента сигнала (в секундах)
- `step` (*Union[int, float]*) – Шаг сдвига окна сегмента сигнала (в секундах)
- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

Кортеж с двумя списками:

1. Список с экспертными признаками
2. Список с лог мел-спектрограммами

Тип результата

Tuple[List[Optional[np.ndarray]], List[Optional[np.ndarray]]]

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

sr = 44100
path = '/Users/dl/GitHub/oceanai/oceanai/dataset/test80_01/glgfB3vFewc.004.mp4'

hc_features, melspectrogram_features = audio._get_acoustic_features(
    path = path, sr = sr,
    window = 2, step = 1,
    last = False, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-10-19 14:58:19] Извлечение признаков (экспертных и лог мел-спектрограмм)␣
↪из акустического сигнала ...

[2022-10-19 14:58:20] Статистика извлеченных признаков из акустического сигнала:
    Общее количество сегментов с:
    1. экспертными признаками: 12
    2. лог мел-спектрограммами: 12
    Размерность матрицы экспертных признаков одного сегмента: 196 × 25
    Размерность тензора с лог мел-спектрограммами одного сегмента: 224 × 224 ×␣
↪3

--- Время выполнения: 1.273 сек. ---
```

Ошибки – 1 –

In [2]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

sr = 44100
path = '/Users/dl/GitHub/oceanai/oceanai/dataset/test80_01/glgfB3vFewc.004.mp4'

hc_features, melspectrogram_features = audio._get_acoustic_features(
    path = 1, sr = sr,
    window = 2, step = 1,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    last = False, out = True,
    runtime = True, run = True
)
```

[2]:

```
[2022-10-19 15:33:04] Неверные типы или значения аргументов в "Audio._get_
↪acoustic_features" ...
```


In [2]:

```
from oceanai.modules.lab.audio import Audio

audio = Audio()

sr = 44100
path = '/Users/dl/GitHub/oceanai/oceanai/dataset/test80_01/glgfB3vFewc.004.mp4'

hc_features, melspectrogram_features = audio._get_acoustic_features(
    path = path, sr = sr,
    window = 0.04, step = 1,
    last = False, out = True,
    runtime = True, run = True
)
```

[2]:

```
[2022-10-19 15:34:38] Извлечение признаков (экспертных и лог мел-спектрограмм)␣
↪из акустического сигнала ...

[2022-10-19 15:34:38] Что-то пошло не так ... указан слишком маленький размер␣
↪(0.04) окна сегмента сигнала ...

    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/audio.py
    Линия: 863
    Метод: _get_acoustic_features
    Тип ошибки: IsSmallWindowSizeError

--- Время выполнения: 0.049 сек. ---
```

property audio_model_hc_: Optional[Model]

Получение нейросетевой модели **tf.keras.Model** для получения оценок по экспертным признакам

Результат

Нейросетевая модель **tf.keras.Model** или None

Тип результата

Optional[tf.keras.Model]

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.load_audio_model_hc(
    show_summary = False, out = True,
    runtime = True, run = True
)

audio.audio_model_hc_
```

[1]:
```
[2022-10-17 13:54:35] Формирование нейросетевой архитектуры модели для␣
↪получения оценок по экспертным признакам (аудио модальность) ...

--- Время выполнения: 0.509 сек. ---

<tf.keras.Model at 0x13dd600a0>
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.audio_model_hc_
```

[2]:
```

```

property audio_model_nn_: Optional[Model]

Получение нейросетевой модели **tf.keras.Model** для получения оценок по нейросетевым признакам

Результат
Нейросетевая модель **tf.keras.Model** или None

Тип результата
Optional[tf.keras.Model]

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.load_audio_model_nn(
    show_summary = False, out = True,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    runtime = True, run = True
)

audio.audio_model_nn_
```

```
[1]: [2022-10-17 13:58:29] Формирование нейросетевой архитектуры для получения
     ↪оценок по нейросетевым признакам ...

--- Время выполнения: 0.444 сек. ---

<tf.keras.Model at 0x13db97760>
```

Ошибка – 1 –

```
In [2]: from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.audio_model_nn_
```

```
[2]:
```

property audio_models_b5_: Dict[str, Optional[Model]]

Получение нейросетевых моделей **tf.keras.Model** для получения результатов оценки персональных качеств

Результат

Словарь с нейросетевыми моделями **tf.keras.Model**

Тип результата

Dict

Примеры

Верно – 1 –

```
In [1]: from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.load_audio_models_b5(
    show_summary = False, out = True,
    runtime = True, run = True
)

audio.audio_models_b5_
```

```
[1]: [2022-10-19 15:45:35] Формирование нейросетевых архитектур моделей для
     ↪получения результатов оценки
     персональных качеств (аудио модальность) ...
```

(continues on next page)

(продолжение с предыдущей страницы)

```
3
4  --- Время выполнения: 0.07 сек. ---
5
6  {
7      'openness': <tf.keras.Model at 0x1481e03a0>,
8      'conscientiousness': <tf.keras.Model at 0x147d13520>,
9      'extraversion': <tf.keras.Model at 0x1481edfa0>,
10     'agreeableness': <tf.keras.Model at 0x1481cfc40>,
11     'non_neuroticism': <tf.keras.Model at 0x1481cffd0>
12 }
```

Ошибка – 1 –

In [2]:
```
1  from oceanai.modules.lab.audio import Audio
2
3  audio = Audio()
4
5  audio.audio_models_b5_
```

[2]:
```
1  {
2      'openness': None,
3      'conscientiousness': None,
4      'extraversion': None,
5      'agreeableness': None,
6      'non_neuroticism': None
7  }
```

get_acoustic_features(*path: str*, *sr: int = 44100*, *window: Union[int, float] = 2.0*, *step: Union[int, float] = 1.0*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → Tuple[List[Optional[ndarray]], List[Optional[ndarray]]]

Извлечение признаков из акустического сигнала

Параметры

- `path` (*str*) – Путь к аудио или видеофайлу
- `sr` (*int*) – Частота дискретизации
- `window` (*Union[int, float]*) – Размер окна сегмента сигнала (в секундах)
- `step` (*Union[int, float]*) – Шаг сдвига окна сегмента сигнала (в секундах)
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

Кортеж с двумя списками:

1. Список с экспертными признаками
2. Список с лог мел-спектрограммами

Тип результата

Tuple[List[Optional[np.ndarray]], List[Optional[np.ndarray]]]

Пример

get_audio_union_predictions(*depth: int = 1*, *recursive: bool = False*, *sr: int = 44100*, *window: Union[int, float] = 2.0*, *step: Union[int, float] = 1.0*, *accuracy=True*, *url_ accuracy: str = ''*, *logs: bool = True*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Получения прогнозов по аудио

Параметры

- `depth` (*int*) – Глубина иерархии для получения данных
- `recursive` (*bool*) – Рекурсивный поиск данных
- `sr` (*int*) – Частота дискретизации
- `window` (*Union[int, float]*) – Размер окна сегмента сигнала (в секундах)
- `step` (*Union[int, float]*) – Шаг сдвига окна сегмента сигнала (в секундах)
- `accuracy` (*bool*) – Вычисление точности
- `url_accuracy` (*str*) – Полный путь к файлу с верными предсказаниями для подсчета точности
- `logs` (*bool*) – При необходимости формировать LOG файл
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат
True если прогнозы успешно получены, в обратном случае **False**

Тип результата
bool

Пример

load_audio_model_hc(*show_ summary: bool = False*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам

Параметры

- `show_summary` (*bool*) – Отображение сформированной нейросетевой архитектуры модели
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат
True если нейросетевая архитектура модели сформирована, в обратном случае **False**

Тип результата
bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()
audio.load_audio_model_hc(
    show_summary = False, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-10-17 13:16:23] Формирование нейросетевой архитектуры модели для␣
↪получения оценок по экспертным признакам (аудио модальность) ...

--- Время выполнения: 0.364 сек. ---

True
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()
audio.load_audio_model_hc(
    show_summary = 1, out = True,
    runtime = True, run = True
)
```

[2]:
```
[2022-10-17 13:20:04] Неверные типы или значения аргументов в "Audio.load_audio_
↪model_hc" ...

False
```

load_audio_model_nn(*show_summary: bool = False*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам

Параметры

- `show_summary` (*bool*) – Отображение сформированной нейросетевой архитектуры модели
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

True если нейросетевая архитектура модели сформирована, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()
audio.load_audio_model_nn(
    show_summary = True, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-10-17 13:25:34] Формирование нейросетевой архитектуры для получения␣
↪оценок по нейросетевым признакам (аудио модальность) ...

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 512)               12845568

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 5)                 1285

=================================================================
Total params: 27,692,869
Trainable params: 27,692,869
Non-trainable params: 0
_________________________________________________________________
--- Время выполнения: 0.407 сек. ---

True
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()
audio.load_audio_model_nn(
    show_summary = 1, out = True,
    runtime = True, run = True
)
```

[2]:
```
[2022-10-17 13:25:40] Неверные типы или значения аргументов в "Audio.load_audio_
↪model_nn" ...

False
```

load_audio_model_weights_hc(*url: str*, *force_ reload: bool = True*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Загрузка весов нейросетевой модели для получения оценок по экспертным признакам

Параметры

- `url` (*str*) – Полный путь к файлу с весами нейросетевой модели
- `force_reload` (*bool*) – Принудительная загрузка файла с весами нейросетевой модели из сети
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат
True если веса нейросетевой модели загружены, в обратном случае **False**

Тип результата
bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.load_audio_model_hc(
    show_summary = False, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-10-17 14:24:28] Формирование нейросетевой архитектуры модели для␣
↪получения оценок по экспертным признакам (аудио модальность) ...

--- Время выполнения: 0.398 сек. ---

True
```

In [2]:
```
audio.path_to_save_ = './models'
audio.chunk_size_ = 2000000

url = audio.weights_for_big5_['audio']['hc']['sberdisk']

audio.load_audio_model_weights_hc(
    url = url,
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[2]:
```
[2022-10-17 14:24:30] Загрузка весов нейросетевой модели для получения оценок␣
↪по экспертным признакам (аудио модальность) ...

[2022-10-17 14:24:30] Загрузка файла "weights_2022-05-05_11-27-55.h5" (100.0%) .
↪..

--- Время выполнения: 0.414 сек. ---

True
```

Ошибка – 1 –

In [3]:

```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.path_to_save_ = './models'
audio.chunk_size_ = 2000000

url = audio.weights_for_big5_['audio']['hc']['sberdisk']

audio.load_audio_model_weights_hc(
    url = url,
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[3]:

```
[2022-10-17 15:21:13] Загрузка весов нейросетевой модели для получения оценок по экспертным признакам (аудио модальность) ...

[2022-10-17 15:21:14] Загрузка файла "weights_2022-05-05_11-27-55.h5" (100.0%) ...

[2022-10-17 15:21:14] Что-то пошло не так ... нейросетевая архитектура модели для получения оценок по экспертным признакам не сформирована (аудио модальность) ...

--- Время выполнения: 0.364 сек. ---

False
```

load_audio_model_weights_nn(*url: str*, *force_reload: bool = True*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам

Параметры

- `url` (*str*) – Полный путь к файлу с весами нейросетевой модели
- `force_reload` (*bool*) – Принудительная загрузка файла с весами нейросетевой модели из сети
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

True если веса нейросетевой модели загружены, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.load_audio_model_nn(
    show_summary = False, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-10-17 15:47:20] Формирование нейросетевой архитектуры для получения␣
↪оценок по нейросетевым
признакам (аудио модальность) ...

--- Время выполнения: 0.419 сек. ---

True
```

In [2]:
```
audio.path_to_save_ = './models'
audio.chunk_size_ = 2000000

url = audio.weights_for_big5_['audio']['nn']['sberdisk']

audio.load_audio_model_weights_nn(
    url = url,
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[2]:
```
[2022-10-17 15:47:22] Загрузка весов нейросетевой модели для получения оценок␣
↪по нейросетевым
признакам (аудио модальность) ...

[2022-10-17 15:47:26] Загрузка файла "weights_2022-05-03_07-46-14.h5" (100.0%) .
↪..

--- Время выполнения: 3.884 сек. ---

True
```

Ошибка – 1 –

In [3]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

```

(continues on next page)

(продолжение с предыдущей страницы)

```
audio.path_to_save_ = './models'
audio.chunk_size_ = 2000000

url = audio.weights_for_big5_['audio']['nn']['sberdisk']

audio.load_audio_model_weights_nn(
    url = url,
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[3]:

```
[2022-10-17 15:49:57] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (аудио модальность) ...

[2022-10-17 15:50:04] Загрузка файла "weights_2022-05-03_07-46-14.h5" (100.0%) ...

[2022-10-17 15:50:04] Что-то пошло не так ... нейросетевая архитектура модели для получения оценок по нейросетевым признакам не сформирована (аудио модальность) ...

--- Время выполнения: 6.786 сек. ---

False
```

load_audio_models_b5(*show_summary: bool = False*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Формирование нейросетевых архитектур моделей для получения результатов оценки персональных качеств

Параметры

- **show_summary** (*bool*) – Отображение последней сформированной нейросетевой архитектуры моделей
- **out** (*bool*) – Отображение
- **runtime** (*bool*) – Подсчет времени выполнения
- **run** (*bool*) – Блокировка выполнения

Результат

True если нейросетевые архитектуры модели сформированы, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()
audio.load_audio_models_b5(
    show_summary = True, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-10-18 11:39:22] Формирование нейросетевых архитектур моделей для␣
↪получения результатов оценки
персональных качеств (аудио модальность) ...

Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32)]              0

 dense_1 (Dense)             (None, 1)                 33

 activ_1 (Activation)        (None, 1)                 0

=================================================================
Total params: 33
Trainable params: 33
Non-trainable params: 0
_________________________________________________________________
--- Время выполнения: 0.163 сек. ---

True
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()
audio.load_audio_models_b5(
    show_summary = 1, out = True,
    runtime = True, run = True
)
```

[2]:
```
[2022-10-18 13:47:36] Неверные типы или значения аргументов в "Audio.load_audio_
↪models_b5" ...

False
```

load_audio_models_weights_b5(*url_openness: str*, *url_conscientiousness: str*, *url_extraversion: str*, *url_agreeableness: str*, *url_non_neuroticism: str*, *force_reload: bool = True*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Загрузка весов нейросетевых моделей для получения результатов оценки персональных качеств

Параметры

- `url_openness` (*str*) – Полный путь к файлу с весами нейросетевой модели (открытость опыту)
- `url_conscientiousness` (*str*) – Полный путь к файлу с весами нейросетевой модели (добросовестность)
- `url_extraversion` (*str*) – Полный путь к файлу с весами нейросетевой модели (экстраверсия)
- `url_agreeableness` (*str*) – Полный путь к файлу с весами нейросетевой модели (доброжелательность)
- `url_non_neuroticism` (*str*) – Полный путь к файлу с весами нейросетевой модели (эмоциональная стабильность)
- `force_reload` (*bool*) – Принудительная загрузка файлов с весами нейросетевых моделей из сети
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

True если веса нейросетевых моделей загружены, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.load_audio_models_b5(
    show_summary = False, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-10-18 22:40:05] Формирование нейросетевых архитектур моделей для
↪получения результатов оценки
персональных качеств (аудио модальность) ...

--- Время выполнения: 0.163 сек. ---
```

(continues on next page)

(продолжение с предыдущей страницы)

```

True
```

In [2]:

```
audio.path_to_save_ = './models'
audio.chunk_size_ = 2000000

url_openness = audio.weights_for_big5_['audio']['b5']['openness']['sberdisk']
url_conscientiousness = audio.weights_for_big5_['audio']['b5']['conscientiousness']['sberdisk']
url_extraversion = audio.weights_for_big5_['audio']['b5']['extraversion']['sberdisk']
url_agreeableness = audio.weights_for_big5_['audio']['b5']['agreeableness']['sberdisk']
url_non_neuroticism = audio.weights_for_big5_['audio']['b5']['non_neuroticism']['sberdisk']

audio.load_audio_models_weights_b5(
    url_openness = url_openness,
    url_conscientiousness = url_conscientiousness,
    url_extraversion = url_extraversion,
    url_agreeableness = url_agreeableness,
    url_non_neuroticism = url_non_neuroticism,
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[2]:

```
[2022-10-18 23:08:37] Загрузка весов нейросетевых моделей для получения␣
↪результатов оценки
персональных качеств (аудио модальность) ...

[2022-10-18 23:08:37] Загрузка файла "weights_2022-06-15_16-16-20.h5" (100.0%) .
↪.. Открытость опыту

[2022-10-18 23:08:38] Загрузка файла "weights_2022-06-15_16-21-57.h5" (100.0%) .
↪.. Добросовестность

[2022-10-18 23:08:38] Загрузка файла "weights_2022-06-15_16-26-41.h5" (100.0%) .
↪.. Экстраверсия

[2022-10-18 23:08:38] Загрузка файла "weights_2022-06-15_16-32-51.h5" (100.0%) .
↪.. Доброжелательность

[2022-10-18 23:08:39] Загрузка файла "weights_2022-06-15_16-37-46.h5" (100.0%) .
↪.. Эмоциональная стабильность

--- Время выполнения: 1.611 сек. ---

True
```

Ошибка – 1 –

In [3]:

```
from oceanai.modules.lab.audio import Audio

audio = Audio()

audio.path_to_save_ = './models'
audio.chunk_size_ = 2000000

url_openness = audio.weights_for_big5_['audio']['b5']['openness']['sberdisk']
url_conscientiousness = audio.weights_for_big5_['audio']['b5']['conscientiousness']['sberdisk']
url_extraversion = audio.weights_for_big5_['audio']['b5']['extraversion']['sberdisk']
url_agreeableness = audio.weights_for_big5_['audio']['b5']['agreeableness']['sberdisk']
url_non_neuroticism = audio.weights_for_big5_['audio']['b5']['non_neuroticism']['sberdisk']

audio.load_audio_models_weights_b5(
    url_openness = url_openness,
    url_conscientiousness = url_conscientiousness,
    url_extraversion = url_extraversion,
    url_agreeableness = url_agreeableness,
    url_non_neuroticism = url_non_neuroticism,
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[3]:

```
[2022-10-18 23:09:40] Загрузка весов нейросетевых моделей для получения результатов оценки
персональных качеств (аудио модальность) ...

[2022-10-18 23:09:41] Загрузка файла "weights_2022-06-15_16-16-20.h5" (100.0%) ...

[2022-10-18 23:09:41] Что-то пошло не так ... не удалось загрузить веса нейросетевой модели ...
Открытость опыту

    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/audio.py
    Линия: 1764
    Метод: load_audio_models_weights_b5
    Тип ошибки: AttributeError

[2022-10-18 23:09:41] Загрузка файла "weights_2022-06-15_16-21-57.h5" (100.0%) ...

[2022-10-18 23:09:41] Что-то пошло не так ... не удалось загрузить веса нейросетевой модели ...
Добросовестность

```

(continues on next page)

(продолжение с предыдущей страницы)

```
    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/audio.py
    Линия: 1764
    Метод: load_audio_models_weights_b5
    Тип ошибки: AttributeError

[2022-10-18 23:09:41] Загрузка файла "weights_2022-06-15_16-26-41.h5" (100.0%) ...

[2022-10-18 23:09:41] Что-то пошло не так ... не удалось загрузить веса нейросетевой модели ...
Экстраверсия

    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/audio.py
    Линия: 1764
    Метод: load_audio_models_weights_b5
    Тип ошибки: AttributeError

[2022-10-18 23:09:42] Загрузка файла "weights_2022-06-15_16-32-51.h5" (100.0%) ...

[2022-10-18 23:09:42] Что-то пошло не так ... не удалось загрузить веса нейросетевой модели ...
Доброжелательность

    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/audio.py
    Линия: 1764
    Метод: load_audio_models_weights_b5
    Тип ошибки: AttributeError

[2022-10-18 23:09:42] Загрузка файла "weights_2022-06-15_16-37-46.h5" (100.0%) ...

[2022-10-18 23:09:42] Что-то пошло не так ... не удалось загрузить веса нейросетевой модели ...
Эмоциональная стабильность

    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/audio.py
    Линия: 1764
    Метод: load_audio_models_weights_b5
    Тип ошибки: AttributeError

--- Время выполнения: 1.573 сек. ---

False
```

property smile_: Smile

Получение функций OpenSmile

Результат

Извлеченные функции OpenSmile

Тип результата

opensmile.core.smile.Smile

Пример

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.audio import Audio

audio = Audio()
audio.smile_
```

[1]:
```
{
    '$opensmile.core.smile.Smile': {
        'feature_set': 'eGeMAPSv02',
        'feature_level': 'LowLevelDescriptors',
        'options': {},
        'sampling_rate': None,
        'channels': [0],
        'mixdown': False,
        'resample': False
    }
}
```

Видео

class oceanai.modules.lab.video.VideoMessages(*lang: str = 'ru', color_simple: str = '#666', color_info: str = '#1776D2', color_err: str = '#FF0000', color_true: str = '#008001', bold_text: bool = True, text_runtime: str = '', num_to_df_display: int = 30*)

Базовые классы: *Download*

Класс для сообщений

Параметры

- lang (*str*) – Смотреть *lang*
- color_simple (*str*) – Смотреть *color_simple*
- color_info (*str*) – Смотреть *color_info*
- color_err (*str*) – Смотреть *color_err*
- color_true (*str*) – Смотреть *color_true*
- bold_text (*bool*) – Смотреть *bold_text*
- num_to_df_display (*int*) – Смотреть *num_to_df_display*
- text_runtime (*str*) – Смотреть *text_runtime*

class oceanai.modules.lab.video.Video(*lang: str = 'ru', color_simple: str = '#666', color_info: str = '#1776D2', color_err: str = '#FF0000', color_true: str = '#008001', bold_text: bool = True, text_runtime: str = '', num_to_df_display: int = 30*)

Базовые классы: *VideoMessages*

Класс для обработки видео

Параметры

- `lang` (*str*) – Смотреть *lang*
- `color_simple` (*str*) – Смотреть *color_simple*
- `color_info` (*str*) – Смотреть *color_info*
- `color_err` (*str*) – Смотреть *color_err*
- `color_true` (*str*) – Смотреть *color_true*
- `bold_text` (*bool*) – Смотреть *bold_text*
- `num_to_df_display` (*int*) – Смотреть *num_to_df_display*
- `text_runtime` (*str*) – Смотреть *text_runtime*

`__calc_reshape_img_coef`(*shape: Union[Tuple[int], List[int]]*, *new_shape: Union[int, Tuple[int], List[int]]*, *out: bool = True*) → float

Вычисление коэффициента изменения размера изображения

Примечание: private (приватный метод)

Параметры

- `shape` (*Union[Tuple[int], List[int]]*) – Текущий размер изображения (ширина, высота)
- `new_shape` (*Union[int, Tuple[int], List[int]]*) – Желаемый размер изображения
- `out` (*bool*) – Отображение

Результат
Коэффициент изменения размера изображения

Тип результата
float

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()

video._Video__calc_reshape_img_coef(
    shape = (1280, 720),
    new_shape = 224,
    out = True
)
```

[1]:
```
0.175
```

Верно – 2 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()

video._Video__calc_reshape_img_coef(
    shape = (1280, 720),
    new_shape = (1920, 1080),
    out = True
)
```

[1]:
```
1.5
```

Ошибка – 1 –

In [3]:
```
from oceanai.modules.lab.video import Video

video = Video()

video._Video__calc_reshape_img_coef(
    shape = (1280, 720),
    new_shape = '',
    out = True
)
```

[4]:
```
[2022-10-29 13:24:27] Неверные типы или значения аргументов в "Video.__calc_
↪reshape_img_coef" ...

-1.0
```

__concat_pred(*pred_hc: ndarray*, *pred_nn: ndarray*, *out: bool = True*) → List[Optional[ndarray]]

Конкатенация оценок по экспертным и нейросетевым признакам

Примечание: private (приватный метод)

Параметры

- pred_hc (*np.ndarray*) – Оценки по экспертным признакам
- pred_nn (*np.ndarray*) – Оценки по нейросетевым признакам
- out (*bool*) – Отображение

Результат

Конкатенированные оценки по экспертным и нейросетевым признакам

Тип результата

List[Optional[np.ndarray]]

Примеры

Верно – 1 –

In [1]:
```
import numpy as np
from oceanai.modules.lab.video import Video

video = Video()

arr_hc = np.array([
    [0.64113516, 0.6217892, 0.54451424, 0.6144415, 0.59334993],
    [0.6652424, 0.63606125, 0.572305, 0.63169795, 0.612515]
])

arr_nn = np.array([
    [0.56030345, 0.7488746, 0.44648764, 0.59893465, 0.5701077],
    [0.5900006, 0.7652722, 0.4795154, 0.6409055, 0.6088242]
])

video._Video__concat_pred(
    pred_hc = arr_hc,
    pred_nn = arr_nn,
    out = True
)
```

[1]:
```
[
    array([
            0.64113516, 0.6652424, 0.65318878, 0.65318878, 0.65318878,
            0.65318878, 0.65318878, 0.65318878, 0.65318878, 0.65318878,
            0.65318878, 0.65318878, 0.65318878, 0.65318878, 0.65318878,
            0.65318878, 0.56030345, 0.5900006, 0.57515202, 0.57515202,
            0.57515202, 0.57515202, 0.57515202, 0.57515202, 0.57515202,
            0.57515202, 0.57515202, 0.57515202, 0.57515202, 0.57515202,
            0.57515202, 0.57515202
    ]),
    array([
            0.6217892, 0.63606125, 0.62892523, 0.62892523, 0.62892523,
            0.62892523, 0.62892523, 0.62892523, 0.62892523, 0.62892523,
            0.62892523, 0.62892523, 0.62892523, 0.62892523, 0.62892523,
            0.62892523, 0.7488746, 0.7652722, 0.7570734, 0.7570734,
            0.7570734, 0.7570734, 0.7570734, 0.7570734, 0.7570734,
            0.7570734, 0.7570734, 0.7570734, 0.7570734, 0.7570734,
            0.7570734, 0.7570734
    ]),
    array([
            0.54451424, 0.572305, 0.55840962, 0.55840962, 0.55840962,
            0.55840962, 0.55840962, 0.55840962, 0.55840962, 0.55840962,
            0.55840962, 0.55840962, 0.55840962, 0.55840962, 0.55840962,
            0.55840962, 0.44648764, 0.4795154, 0.46300152, 0.46300152,
            0.46300152, 0.46300152, 0.46300152, 0.46300152, 0.46300152,
            0.46300152, 0.46300152, 0.46300152, 0.46300152, 0.46300152,
            0.46300152, 0.46300152
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    ]),
    array([
            0.6144415, 0.63169795, 0.62306972, 0.62306972, 0.62306972,
            0.62306972, 0.62306972, 0.62306972, 0.62306972, 0.62306972,
            0.62306972, 0.62306972, 0.62306972, 0.62306972, 0.62306972,
            0.62306972, 0.59893465, 0.6409055, 0.61992008, 0.61992008,
            0.61992008, 0.61992008, 0.61992008, 0.61992008, 0.61992008,
            0.61992008, 0.61992008, 0.61992008, 0.61992008, 0.61992008,
            0.61992008, 0.61992008
    ]),
    array([
            0.59334993, 0.612515, 0.60293247, 0.60293247, 0.60293247,
            0.60293247, 0.60293247, 0.60293247, 0.60293247, 0.60293247,
            0.60293247, 0.60293247, 0.60293247, 0.60293247, 0.60293247,
            0.60293247, 0.5701077, 0.6088242, 0.58946595, 0.58946595,
            0.58946595, 0.58946595, 0.58946595, 0.58946595, 0.58946595,
            0.58946595, 0.58946595, 0.58946595, 0.58946595, 0.58946595,
            0.58946595, 0.58946595
    ])
]
```

Ошибка – 1 –

In [2]:

```
import numpy as np
from oceanai.modules.lab.video import Video

video = Video()

arr_hc = np.array([
    [0.64113516, 0.6217892, 0.54451424, 0.6144415],
    [0.6652424, 0.63606125, 0.572305, 0.63169795, 0.612515]
])

arr_nn = np.array([
    [0.56030345, 0.7488746, 0.44648764, 0.59893465, 0.5701077],
    [0.5900006, 0.7652722, 0.4795154, 0.6409055, 0.6088242]
])

video._Video__concat_pred(
    pred_hc = arr_hc,
    pred_nn = arr_nn,
    out = True
)
```

[3]:

```
[2022-10-20 22:33:31] Ой! Что-то пошло не так ... конкатенация оценок по␣
↪экспертным и нейросетевым
признакам не произведена (видео модальность) ...

[]
```

__load_model_weights(*url: str*, *force_reload: bool = True*, *info_text: str = ''*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Загрузка весов нейросетевой модели

Примечание: private (приватный метод)

Параметры

- `url` (*str*) – Полный путь к файлу с весами нейросетевой модели
- `force_reload` (*bool*) – Принудительная загрузка файла с весами нейросетевой модели из сети
- `info_text` (*str*) – Текст для информационного сообщения
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

True если веса нейросетевой модели загружены, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.path_to_save_ = './models'
video.chunk_size_ = 2000000

video._Video__load_model_weights(
    url = 'https://download.sberdisk.ru/download/file/412059444?token=JXerCfAjJZg6crD&filename=weights_2022-08-27_18-53-35.h5',
    force_reload = True,
    info_text = 'Загрузка весов нейросетевой модели',
    out = True, runtime = True, run = True
)
```

[1]:
```
[2022-10-27 12:46:55] Загрузка весов нейросетевой модели

[2022-10-27 12:46:55] Загрузка файла "weights_2022-08-27_18-53-35.h5" (100.0%) ...

--- Время выполнения: 0.626 сек. ---

True
```

– 2 –

In [2]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.path_to_save_ = './models'
video.chunk_size_ = 2000000

video._Video__load_model_weights(
    url = './models/weights_2022-08-27_18-53-35.h5',
    force_reload = True,
    info_text = 'Загрузка весов нейросетевой модели',
    out = True, runtime = True, run = True
)
```

[2]:
```
[2022-10-27 12:47:52] Загрузка весов нейросетевой модели

--- Время выполнения: 0.002 сек. ---

True
```

Ошибка – 1 –

In [3]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.path_to_save_ = './models'
video.chunk_size_ = 2000000

video._Video__load_model_weights(
    url = 'https://download.sberdisk.ru/download/file/412059444?
↪token=JXerCfAjJZg6crD&filename=weights_2022-08-27_18-53-35.h5',
    force_reload = True, info_text = '',
    out = True, runtime = True, run = True
)
```

[3]:
```
[2022-10-27 12:48:24] Неверные типы или значения аргументов в "Video.__load_
↪model_weights" ...

False
```

__load_video_model_b5(*show_summary: bool = False*, *out: bool = True*) → Optional[Model]

Формирование нейросетевой архитектуры модели для получения результата оценки персонального качества

Примечание: private (приватный метод)

Параметры

- `show_summary` (*bool*) – Отображение сформированной нейросетевой архитектуры модели

- out (*bool*) – Отображение

Результат

None если неверные типы или значения аргументов, в обратном случае нейросетевая модель **tf.keras.Model** для получения результата оценки персонального качества

Тип результата

Optional[tf.keras.Model]

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()

video._Video__load_video_model_b5(
    show_summary = True, out = True
)
```

[1]:
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32)]              0

 dense_1 (Dense)             (None, 1)                 33

 activ_1 (Activation)        (None, 1)                 0

=================================================================
Total params: 33
Trainable params: 33
Non-trainable params: 0
_________________________________________________________________
<tf.keras.Model at 0x13d442940>
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.lab.video import Video

video = Video()

video._Video__load_video_model_b5(
    show_summary = True, out = []
)
```

[3]:
```
[2022-10-17 10:53:03] Неверные типы или значения аргументов в "Video.__load_
↪video_model_b5" ...
```

__norm_pred(*pred_ data: ndarray*, *len_ nn: int = 16*, *out: bool = True*) → ndarray

Нормализация оценок по экспертным и нейросетевым признакам

Примечание: private (приватный метод)

Параметры

- pred_data (*np.ndarray*) – Оценки
- len_nn (*int*) – Максимальный размер вектора оценок
- out (*bool*) – Отображение

Результат

Нормализованные оценки по экспертным и нейросетевым признакам

Тип результата

np.ndarray

Примеры

Верно – 1 –

In [1]:
```
import numpy as np
from oceanai.modules.lab.video import Video

video = Video()

arr = np.array([
    [0.64113516, 0.6217892, 0.54451424, 0.6144415, 0.59334993],
    [0.6652424, 0.63606125, 0.572305, 0.63169795, 0.612515]
])

video._Video__norm_pred(
    pred_data = arr,
    len_nn = 4,
    out = True
)
```

[1]:
```
array([
    [0.64113516, 0.6217892 , 0.54451424, 0.6144415 , 0.59334993],
    [0.6652424 , 0.63606125, 0.572305  , 0.63169795, 0.612515],
    [0.65318878, 0.62892523, 0.55840962, 0.62306972, 0.60293247],
    [0.65318878, 0.62892523, 0.55840962, 0.62306972, 0.60293247]
])
```

Ошибка – 1 –

In [2]:
```
import numpy as np
from oceanai.modules.lab.video import Video

video = Video()
```

(continues on next page)

(продолжение с предыдущей страницы)

```
5
6  arr = np.array([])
7
8  video._Video__norm_pred(
9      pred_data = arr,
10     len_nn = 4,
11     out = True
12 )
```

[3]:

```
1  [2022-10-20 22:03:17] Неверные типы или значения аргументов в "Video.__norm_pred
   ↪" ...
2
3  array([], dtype=float64)
```

_get_visual_features(*path: str*, *reduction_fps: int = 5*, *window: int = 10*, *step: int = 5*, *lang: str = 'ru'*, *last: bool = False*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → Tuple[ndarray, ndarray]

Извлечение признаков из визуального сигнала (без очистки истории вывода сообщений в ячейке Jupyter)

Примечание: protected (защищенный метод)

Параметры

- `path` (*str*) – Путь к видеофайлу
- `reduction_fps` (*int*) – Понижение кадровой частоты
- `window` (*int*) – Размер окна сегмента сигнала (в кадрах)
- `step` (*int*) – Шаг сдвига окна сегмента сигнала (в кадрах)
- `lang` (*str*) – Язык
- `last` (*bool*) – Замена последнего сообщения
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

Кортеж с двумя np.ndarray:

1. np.ndarray с экспертными признаками
2. np.ndarray с нейросетевыми признаками

Тип результата

Tuple[np.ndarray, np.ndarray]

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()

res_load_model_deep_fe = video.load_video_model_deep_fe(
    show_summary = False,
    out = True,
    runtime = True,
    run = True
)
```

[1]:
```
[2022-11-03 16:37:12] Формирование нейросетевой архитектуры для получения␣
↪нейросетевых признаков (видео модальность) ...

--- Время выполнения: 1.564 сек. ---
```

In [2]:
```
video.path_to_save_ = './models'
video.chunk_size_ = 2000000

url = video.weights_for_big5_['video']['fe']['sberdisk']

res_load_video_model_weights_deep_fe = video.load_video_model_weights_deep_fe(
    url = url,
    force_reload = True, out = True,
    runtime = True, run = True
)
```

[2]:
```
[2022-11-03 16:39:10] Загрузка весов нейросетевой модели для получения␣
↪нейросетевых признаков (видео модальность) ...

[2022-11-03 16:39:14] Загрузка файла "weights_2022-11-01_12-27-07.h5" (100.0%) .
↪..

--- Время выполнения: 4.874 сек. ---
```

In [3]:
```
path = '/Users/dl/GitHub/oceanai/oceanai/dataset/test80_01/glgfB3vFewc.004.mp4'

hc_features, nn_features = video.get_visual_features(
    path = path, reduction_fps = 5,
    window = 10, step = 5,
    out = True, runtime = True, run = True
)
```

[3]:
```
[2022-11-03 16:56:52] Извлечение признаков (экспертных и нейросетевых) из␣
↪визуального сигнала ...

```

(continues on next page)

(продолжение с предыдущей страницы)

```
[2022-11-03 16:56:58] Статистика извлеченных признаков из визуального сигнала:
    Общее количество сегментов с:
    1. экспертными признаками: 12
    2. нейросетевыми признаками: 12
    Размерность матрицы экспертных признаков одного сегмента: 10 × 115
    Размерность тензора с нейросетевыми признаками одного сегмента: 10 × 512
    Понижение кадровой частоты: с 30 до 5

--- Время выполнения: 6.109 сек. ---
```

Ошибка – 1 –

In [4]:
```
from oceanai.modules.lab.video import Video

video = Video()

path = '/Users/dl/GitHub/oceanai/oceanai/dataset/test80_01/glgfB3vFewc.004.mp4'

hc_features, nn_features = video.get_visual_features(
    path = path, reduction_fps = 5,
    window = 10, step = 5,
    out = True, runtime = True, run = True
)
```

[4]:
```
[2022-11-03 16:59:45] Извлечение признаков (экспертных и нейросетевых) из␣
↪визуального сигнала ...

[2022-11-03 16:59:46] Ой! Что-то пошло не так ... нейросетевая архитектура␣
↪модели для получения нейросетевых признаков не сформирована (видео␣
↪модальность) ...

--- Время выполнения: 1.358 сек. ---
```

`get_video_union_predictions`(*depth: int = 1*, *recursive: bool = False*, *reduction_fps: int = 5*, *window: int = 10*, *step: int = 5*, *lang: str = 'ru'*, *accuracy=True*, *url_accuracy: str = ''*, *logs: bool = True*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Получения прогнозов по видео

Параметры

- `depth` (*int*) – Глубина иерархии для получения данных
- `recursive` (*bool*) – Рекурсивный поиск данных
- `reduction_fps` (*int*) – Понижение кадровой частоты
- `window` (*int*) – Размер окна сегмента сигнала (в кадрах)
- `step` (*int*) – Шаг сдвига окна сегмента сигнала (в кадрах)
- `lang` (*str*) – Язык
- `accuracy` (*bool*) – Вычисление точности

- `url_accuracy` (*str*) – Полный путь к файлу с верными предсказаниями для подсчета точности
- `logs` (*bool*) – При необходимости формировать LOG файл
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат
True если прогнозы успешно получены, в обратном случае **False**

Тип результата
bool

Пример

`get_visual_features`(*path: str*, *reduction_fps: int = 5*, *window: int = 10*, *step: int = 5*, *lang: str = 'ru'*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → Tuple[ndarray, ndarray]

Извлечение признаков из визуального сигнала

Параметры

- `path` (*str*) – Путь к видеофайлу
- `reduction_fps` (*int*) – Понижение кадровой частоты
- `window` (*int*) – Размер окна сегмента сигнала (в кадрах)
- `step` (*int*) – Шаг сдвига окна сегмента сигнала (в кадрах)
- `lang` (*str*) – Язык
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

Кортеж с двумя np.ndarray:

1. np.ndarray с экспертными признаками
2. np.ndarray с нейросетевыми признаками

Тип результата
Tuple[np.ndarray, np.ndarray]

Пример

`load_video_model_deep_fe`(*show_summary: bool = False*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Формирование нейросетевой архитектуры для получения нейросетевых признаков

Параметры

- `show_summary` (*bool*) – Отображение сформированной нейросетевой архитектуры модели
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения

- run (*bool*) – Блокировка выполнения

Результат
True если нейросетевая архитектура модели сформирована, в обратном случае **False**

Тип результата
bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()
video.load_video_model_deep_fe(
    show_summary = True, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-11-01 12:18:14] Формирование нейросетевой архитектуры для получения нейросетевых признаков (видео модальность) ...

Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_2 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1/7x7_s2 (Conv2D)          (None, 112, 112, 64  9408        ['input_2[0][0]']
                                )

 conv1/7x7_s2/bn (BatchNormaliz  (None, 112, 112, 64  256        ['conv1/7x7_s2[0][0]']
 ation)                         )

 activation_49 (Activation)     (None, 112, 112, 64  0           ['conv1/7x7_s2/bn[0][0]']
                                )

 max_pooling2d_1 (MaxPooling2D)  (None, 55, 55, 64)  0           ['activation_49[0][0]']

 conv2_1_1x1_reduce (Conv2D)    (None, 55, 55, 64)   4096        ['max_pooling2d_1[0][0]']

 conv2_1_1x1_reduce/bn (BatchNo  (None, 55, 55, 64)  256         ['conv2_1_1x1_reduce[0][0]']
 rmalization)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
activation_50 (Activation)     (None, 55, 55, 64)   0           ['conv2_1_1x1_
↪reduce/bn[0][0]']

conv2_1_3x3 (Conv2D)           (None, 55, 55, 64)   36864       ['activation_
↪50[0][0]']

conv2_1_3x3/bn (BatchNormaliza  (None, 55, 55, 64)  256         ['conv2_1_
↪3x3[0][0]']
tion)

activation_51 (Activation)     (None, 55, 55, 64)   0           ['conv2_1_3x3/
↪bn[0][0]']

conv2_1_1x1_increase (Conv2D)  (None, 55, 55, 256)  16384       ['activation_
↪51[0][0]']

conv2_1_1x1_proj (Conv2D)      (None, 55, 55, 256)  16384       ['max_
↪pooling2d_1[0][0]']

conv2_1_1x1_increase/bn (Batch  (None, 55, 55, 256)  1024       ['conv2_1_1x1_
↪increase[0][0]']
Normalization)

conv2_1_1x1_proj/bn (BatchNorm  (None, 55, 55, 256)  1024       ['conv2_1_1x1_
↪proj[0][0]']
alization)

add_16 (Add)                   (None, 55, 55, 256)  0           ['conv2_1_1x1_
↪increase/bn[0][0]',
                                'conv2_1_1x1_proj/bn[0][0]']

activation_52 (Activation)     (None, 55, 55, 256)  0           ['add_16[0][0]
↪']

conv2_2_1x1_reduce (Conv2D)    (None, 55, 55, 64)   16384       ['activation_
↪52[0][0]']

conv2_2_1x1_reduce/bn (BatchNo  (None, 55, 55, 64)  256         ['conv2_2_1x1_
↪reduce[0][0]']
rmalization)

activation_53 (Activation)     (None, 55, 55, 64)   0           ['conv2_2_1x1_
↪reduce/bn[0][0]']

conv2_2_3x3 (Conv2D)           (None, 55, 55, 64)   36864       ['activation_
↪53[0][0]']

conv2_2_3x3/bn (BatchNormaliza  (None, 55, 55, 64)  256         ['conv2_2_
↪3x3[0][0]']
tion)

```

(continues on next page)

(продолжение с предыдущей страницы)

```
activation_54 (Activation)     (None, 55, 55, 64)   0           ['conv2_2_3x3/
↪bn[0][0]']

conv2_2_1x1_increase (Conv2D)  (None, 55, 55, 256)  16384       ['activation_
↪54[0][0]']

conv2_2_1x1_increase/bn (Batch  (None, 55, 55, 256)  1024       ['conv2_2_1x1_
↪increase[0][0]']
Normalization)

add_17 (Add)                   (None, 55, 55, 256)  0           ['conv2_2_1x1_
↪increase/bn[0][0]',
                                 'activation_52[0][0]']

activation_55 (Activation)     (None, 55, 55, 256)  0           ['add_17[0][0]
↪']

conv2_3_1x1_reduce (Conv2D)    (None, 55, 55, 64)   16384       ['activation_
↪55[0][0]']

conv2_3_1x1_reduce/bn (BatchNo  (None, 55, 55, 64)  256         ['conv2_3_1x1_
↪reduce[0][0]']
rmalization)

activation_56 (Activation)     (None, 55, 55, 64)   0           ['conv2_3_1x1_
↪reduce/bn[0][0]']

conv2_3_3x3 (Conv2D)           (None, 55, 55, 64)   36864       ['activation_
↪56[0][0]']

conv2_3_3x3/bn (BatchNormaliza  (None, 55, 55, 64)  256         ['conv2_3_
↪3x3[0][0]']
tion)

activation_57 (Activation)     (None, 55, 55, 64)   0           ['conv2_3_3x3/
↪bn[0][0]']

conv2_3_1x1_increase (Conv2D)  (None, 55, 55, 256)  16384       ['activation_
↪57[0][0]']

conv2_3_1x1_increase/bn (Batch  (None, 55, 55, 256)  1024       ['conv2_3_1x1_
↪increase[0][0]']
Normalization)

add_18 (Add)                   (None, 55, 55, 256)  0           ['conv2_3_1x1_
↪increase/bn[0][0]',
                                 'activation_55[0][0]']

activation_58 (Activation)     (None, 55, 55, 256)  0           ['add_18[0][0]
↪']

conv3_1_1x1_reduce (Conv2D)    (None, 28, 28, 128)  32768       ['activation_
```

(continues on next page)

(продолжение с предыдущей страницы)

```
↪58[0][0]']

conv3_1_1x1_reduce/bn (BatchNo  (None, 28, 28, 128)  512        ['conv3_1_1x1_
↪reduce[0][0]']
rmalization)

activation_59 (Activation)     (None, 28, 28, 128)  0          ['conv3_1_1x1_
↪reduce/bn[0][0]']

conv3_1_3x3 (Conv2D)           (None, 28, 28, 128)  147456     ['activation_
↪59[0][0]']

conv3_1_3x3/bn (BatchNormaliza  (None, 28, 28, 128)  512        ['conv3_1_
↪3x3[0][0]']
tion)

activation_60 (Activation)     (None, 28, 28, 128)  0          ['conv3_1_3x3/
↪bn[0][0]']

conv3_1_1x1_increase (Conv2D)  (None, 28, 28, 512)  65536      ['activation_
↪60[0][0]']

conv3_1_1x1_proj (Conv2D)      (None, 28, 28, 512)  131072     ['activation_
↪58[0][0]']

conv3_1_1x1_increase/bn (Batch  (None, 28, 28, 512)  2048       ['conv3_1_1x1_
↪increase[0][0]']
Normalization)

conv3_1_1x1_proj/bn (BatchNorm  (None, 28, 28, 512)  2048       ['conv3_1_1x1_
↪proj[0][0]']
alization)

add_19 (Add)                   (None, 28, 28, 512)  0          ['conv3_1_1x1_
↪increase/bn[0][0]',
                                'conv3_1_1x1_proj/bn[0][0]']

activation_61 (Activation)     (None, 28, 28, 512)  0          ['add_19[0][0]
↪']

conv3_2_1x1_reduce (Conv2D)    (None, 28, 28, 128)  65536      ['activation_
↪61[0][0]']

conv3_2_1x1_reduce/bn (BatchNo  (None, 28, 28, 128)  512        ['conv3_2_1x1_
↪reduce[0][0]']
rmalization)

activation_62 (Activation)     (None, 28, 28, 128)  0          ['conv3_2_1x1_
↪reduce/bn[0][0]']

conv3_2_3x3 (Conv2D)           (None, 28, 28, 128)  147456     ['activation_
↪62[0][0]']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
135
136 conv3_2_3x3/bn (BatchNormaliza  (None, 28, 28, 128)  512        ['conv3_2_
    ↪3x3[0][0]']
137 tion)
138
139 activation_63 (Activation)     (None, 28, 28, 128)  0          ['conv3_2_3x3/
    ↪bn[0][0]']
140
141 conv3_2_1x1_increase (Conv2D)  (None, 28, 28, 512)  65536      ['activation_
    ↪63[0][0]']
142
143 conv3_2_1x1_increase/bn (Batch  (None, 28, 28, 512)  2048       ['conv3_2_1x1_
    ↪increase[0][0]']
144 Normalization)
145
146 add_20 (Add)                   (None, 28, 28, 512)  0          ['conv3_2_1x1_
    ↪increase/bn[0][0]',
147                                  'activation_61[0][0]']
148
149 activation_64 (Activation)     (None, 28, 28, 512)  0          ['add_20[0][0]
    ↪']
150
151 conv3_3_1x1_reduce (Conv2D)    (None, 28, 28, 128)  65536      ['activation_
    ↪64[0][0]']
152
153 conv3_3_1x1_reduce/bn (BatchNo  (None, 28, 28, 128)  512        ['conv3_3_1x1_
    ↪reduce[0][0]']
154 rmalization)
155
156 activation_65 (Activation)     (None, 28, 28, 128)  0          ['conv3_3_1x1_
    ↪reduce/bn[0][0]']
157
158 conv3_3_3x3 (Conv2D)           (None, 28, 28, 128)  147456     ['activation_
    ↪65[0][0]']
159
160 conv3_3_3x3/bn (BatchNormaliza  (None, 28, 28, 128)  512        ['conv3_3_
    ↪3x3[0][0]']
161 tion)
162
163 activation_66 (Activation)     (None, 28, 28, 128)  0          ['conv3_3_3x3/
    ↪bn[0][0]']
164
165 conv3_3_1x1_increase (Conv2D)  (None, 28, 28, 512)  65536      ['activation_
    ↪66[0][0]']
166
167 conv3_3_1x1_increase/bn (Batch  (None, 28, 28, 512)  2048       ['conv3_3_1x1_
    ↪increase[0][0]']
168 Normalization)
169
170 add_21 (Add)                   (None, 28, 28, 512)  0          ['conv3_3_1x1_
    ↪increase/bn[0][0]',
171                                  'activation_64[0][0]']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
172
173 activation_67 (Activation)     (None, 28, 28, 512)  0           ['add_21[0][0]
    ↪']
174
175 conv3_4_1x1_reduce (Conv2D)    (None, 28, 28, 128)  65536       ['activation_
    ↪67[0][0]']
176
177 conv3_4_1x1_reduce/bn (BatchNo  (None, 28, 28, 128)  512         ['conv3_4_1x1_
    ↪reduce[0][0]']
178 rmalization)
179
180 activation_68 (Activation)     (None, 28, 28, 128)  0           ['conv3_4_1x1_
    ↪reduce/bn[0][0]']
181
182 conv3_4_3x3 (Conv2D)           (None, 28, 28, 128)  147456      ['activation_
    ↪68[0][0]']
183
184 conv3_4_3x3/bn (BatchNormaliza  (None, 28, 28, 128)  512         ['conv3_4_
    ↪3x3[0][0]']
185 tion)
186
187 activation_69 (Activation)     (None, 28, 28, 128)  0           ['conv3_4_3x3/
    ↪bn[0][0]']
188
189 conv3_4_1x1_increase (Conv2D)  (None, 28, 28, 512)  65536       ['activation_
    ↪69[0][0]']
190
191 conv3_4_1x1_increase/bn (Batch  (None, 28, 28, 512)  2048        ['conv3_4_1x1_
    ↪increase[0][0]']
192 Normalization)
193
194 add_22 (Add)                   (None, 28, 28, 512)  0           ['conv3_4_1x1_
    ↪increase/bn[0][0]',
195                                  'activation_67[0][0]']
196
197 activation_70 (Activation)     (None, 28, 28, 512)  0           ['add_22[0][0]
    ↪']
198
199 conv4_1_1x1_reduce (Conv2D)    (None, 14, 14, 256)  131072      ['activation_
    ↪70[0][0]']
200
201 conv4_1_1x1_reduce/bn (BatchNo  (None, 14, 14, 256)  1024        ['conv4_1_1x1_
    ↪reduce[0][0]']
202 rmalization)
203
204 activation_71 (Activation)     (None, 14, 14, 256)  0           ['conv4_1_1x1_
    ↪reduce/bn[0][0]']
205
206 conv4_1_3x3 (Conv2D)           (None, 14, 14, 256)  589824      ['activation_
    ↪71[0][0]']
207
208 conv4_1_3x3/bn (BatchNormaliza  (None, 14, 14, 256)  1024        ['conv4_1_
```

(continues on next page)

(продолжение с предыдущей страницы)

```
↪3x3[0][0]']
tion)

activation_72 (Activation)     (None, 14, 14, 256)  0           ['conv4_1_3x3/
↪bn[0][0]']

conv4_1_1x1_increase (Conv2D)  (None, 14, 14, 1024  262144      ['activation_
↪72[0][0]']
               )

conv4_1_1x1_proj (Conv2D)      (None, 14, 14, 1024  524288      ['activation_
↪70[0][0]']
               )

conv4_1_1x1_increase/bn (Batch  (None, 14, 14, 1024  4096       ['conv4_1_1x1_
↪increase[0][0]']
Normalization)                 )

conv4_1_1x1_proj/bn (BatchNorm  (None, 14, 14, 1024  4096       ['conv4_1_1x1_
↪proj[0][0]']
alization)                     )

add_23 (Add)                   (None, 14, 14, 1024  0           ['conv4_1_1x1_
↪increase/bn[0][0]',
               )                                  'conv4_1_1x1_proj/bn[0][0]']

activation_73 (Activation)     (None, 14, 14, 1024  0           ['add_23[0][0]
↪']
               )

conv4_2_1x1_reduce (Conv2D)    (None, 14, 14, 256)  262144      ['activation_
↪73[0][0]']

conv4_2_1x1_reduce/bn (BatchNo  (None, 14, 14, 256)  1024       ['conv4_2_1x1_
↪reduce[0][0]']
rmalization)

activation_74 (Activation)     (None, 14, 14, 256)  0           ['conv4_2_1x1_
↪reduce/bn[0][0]']

conv4_2_3x3 (Conv2D)           (None, 14, 14, 256)  589824      ['activation_
↪74[0][0]']

conv4_2_3x3/bn (BatchNormaliza  (None, 14, 14, 256)  1024       ['conv4_2_
↪3x3[0][0]']
tion)

activation_75 (Activation)     (None, 14, 14, 256)  0           ['conv4_2_3x3/
↪bn[0][0]']

conv4_2_1x1_increase (Conv2D)  (None, 14, 14, 1024  262144      ['activation_
↪75[0][0]']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                )

 conv4_2_1x1_increase/bn (Batch  (None, 14, 14, 1024  4096       ['conv4_2_1x1_
↪increase[0][0]']
 Normalization)                 )

 add_24 (Add)                   (None, 14, 14, 1024  0           ['conv4_2_1x1_
↪increase/bn[0][0]',
                                                                  'activation_73[0][0]']

 activation_76 (Activation)     (None, 14, 14, 1024  0           ['add_24[0][0]
↪']
                )

 conv4_3_1x1_reduce (Conv2D)    (None, 14, 14, 256)  262144      ['activation_
↪76[0][0]']

 conv4_3_1x1_reduce/bn (BatchNo  (None, 14, 14, 256)  1024       ['conv4_3_1x1_
↪reduce[0][0]']
 rmalization)

 activation_77 (Activation)     (None, 14, 14, 256)  0           ['conv4_3_1x1_
↪reduce/bn[0][0]']

 conv4_3_3x3 (Conv2D)           (None, 14, 14, 256)  589824      ['activation_
↪77[0][0]']

 conv4_3_3x3/bn (BatchNormaliza  (None, 14, 14, 256)  1024       ['conv4_3_
↪3x3[0][0]']
 tion)

 activation_78 (Activation)     (None, 14, 14, 256)  0           ['conv4_3_3x3/
↪bn[0][0]']

 conv4_3_1x1_increase (Conv2D)  (None, 14, 14, 1024  262144      ['activation_
↪78[0][0]']
                )

 conv4_3_1x1_increase/bn (Batch  (None, 14, 14, 1024  4096       ['conv4_3_1x1_
↪increase[0][0]']
 Normalization)                 )

 add_25 (Add)                   (None, 14, 14, 1024  0           ['conv4_3_1x1_
↪increase/bn[0][0]',
                                                                  'activation_76[0][0]']

 activation_79 (Activation)     (None, 14, 14, 1024  0           ['add_25[0][0]
↪']
                )

 conv4_4_1x1_reduce (Conv2D)    (None, 14, 14, 256)  262144      ['activation_
↪79[0][0]']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
conv4_4_1x1_reduce/bn (BatchNo  (None, 14, 14, 256)  1024       ['conv4_4_1x1_
↪reduce[0][0]']
rmalization)

activation_80 (Activation)     (None, 14, 14, 256)  0          ['conv4_4_1x1_
↪reduce/bn[0][0]']

conv4_4_3x3 (Conv2D)           (None, 14, 14, 256)  589824     ['activation_
↪80[0][0]']

conv4_4_3x3/bn (BatchNormaliza  (None, 14, 14, 256)  1024       ['conv4_4_
↪3x3[0][0]']
tion)

activation_81 (Activation)     (None, 14, 14, 256)  0          ['conv4_4_3x3/
↪bn[0][0]']

conv4_4_1x1_increase (Conv2D)  (None, 14, 14, 1024  262144     ['activation_
↪81[0][0]']
                               )

conv4_4_1x1_increase/bn (Batch  (None, 14, 14, 1024  4096       ['conv4_4_1x1_
↪increase[0][0]']
Normalization)                 )

add_26 (Add)                   (None, 14, 14, 1024  0          ['conv4_4_1x1_
↪increase/bn[0][0]',
                               )                                 'activation_79[0][0]']

activation_82 (Activation)     (None, 14, 14, 1024  0          ['add_26[0][0]
↪']
                               )

conv4_5_1x1_reduce (Conv2D)    (None, 14, 14, 256)  262144     ['activation_
↪82[0][0]']

conv4_5_1x1_reduce/bn (BatchNo  (None, 14, 14, 256)  1024       ['conv4_5_1x1_
↪reduce[0][0]']
rmalization)

activation_83 (Activation)     (None, 14, 14, 256)  0          ['conv4_5_1x1_
↪reduce/bn[0][0]']

conv4_5_3x3 (Conv2D)           (None, 14, 14, 256)  589824     ['activation_
↪83[0][0]']

conv4_5_3x3/bn (BatchNormaliza  (None, 14, 14, 256)  1024       ['conv4_5_
↪3x3[0][0]']
tion)

activation_84 (Activation)     (None, 14, 14, 256)  0          ['conv4_5_3x3/
```

(continues on next page)

(продолжение с предыдущей страницы)

```
↪bn[0][0]']

conv4_5_1x1_increase (Conv2D)  (None, 14, 14, 1024  262144      ['activation_
↪84[0][0]']
                )

conv4_5_1x1_increase/bn (Batch  (None, 14, 14, 1024  4096       ['conv4_5_1x1_
↪increase[0][0]']
Normalization)                 )

add_27 (Add)                   (None, 14, 14, 1024  0           ['conv4_5_1x1_
↪increase/bn[0][0]',
                )                                   'activation_82[0][0]']

activation_85 (Activation)     (None, 14, 14, 1024  0           ['add_27[0][0]
↪']
                )

conv4_6_1x1_reduce (Conv2D)    (None, 14, 14, 256)  262144      ['activation_
↪85[0][0]']

conv4_6_1x1_reduce/bn (BatchNo  (None, 14, 14, 256)  1024       ['conv4_6_1x1_
↪reduce[0][0]']
rmalization)

activation_86 (Activation)     (None, 14, 14, 256)  0           ['conv4_6_1x1_
↪reduce/bn[0][0]']

conv4_6_3x3 (Conv2D)           (None, 14, 14, 256)  589824      ['activation_
↪86[0][0]']

conv4_6_3x3/bn (BatchNormaliza  (None, 14, 14, 256)  1024       ['conv4_6_
↪3x3[0][0]']
tion)

activation_87 (Activation)     (None, 14, 14, 256)  0           ['conv4_6_3x3/
↪bn[0][0]']

conv4_6_1x1_increase (Conv2D)  (None, 14, 14, 1024  262144      ['activation_
↪87[0][0]']
                )

conv4_6_1x1_increase/bn (Batch  (None, 14, 14, 1024  4096       ['conv4_6_1x1_
↪increase[0][0]']
Normalization)                 )

add_28 (Add)                   (None, 14, 14, 1024  0           ['conv4_6_1x1_
↪increase/bn[0][0]',
                )                                   'activation_85[0][0]']

activation_88 (Activation)     (None, 14, 14, 1024  0           ['add_28[0][0]
↪']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                )

 conv5_1_1x1_reduce (Conv2D)    (None, 7, 7, 512)    524288      ['activation_
↪88[0][0]']

 conv5_1_1x1_reduce/bn (BatchNo  (None, 7, 7, 512)   2048        ['conv5_1_1x1_
↪reduce[0][0]']
 rmalization)

 activation_89 (Activation)     (None, 7, 7, 512)    0           ['conv5_1_1x1_
↪reduce/bn[0][0]']

 conv5_1_3x3 (Conv2D)           (None, 7, 7, 512)    2359296     ['activation_
↪89[0][0]']

 conv5_1_3x3/bn (BatchNormaliza  (None, 7, 7, 512)   2048        ['conv5_1_
↪3x3[0][0]']
 tion)

 activation_90 (Activation)     (None, 7, 7, 512)    0           ['conv5_1_3x3/
↪bn[0][0]']

 conv5_1_1x1_increase (Conv2D)  (None, 7, 7, 2048)   1048576     ['activation_
↪90[0][0]']

 conv5_1_1x1_proj (Conv2D)      (None, 7, 7, 2048)   2097152     ['activation_
↪88[0][0]']

 conv5_1_1x1_increase/bn (Batch  (None, 7, 7, 2048)  8192        ['conv5_1_1x1_
↪increase[0][0]']
 Normalization)

 conv5_1_1x1_proj/bn (BatchNorm  (None, 7, 7, 2048)  8192        ['conv5_1_1x1_
↪proj[0][0]']
 alization)

 add_29 (Add)                   (None, 7, 7, 2048)   0           ['conv5_1_1x1_
↪increase/bn[0][0]',
                                 'conv5_1_1x1_proj/bn[0][0]']

 activation_91 (Activation)     (None, 7, 7, 2048)   0           ['add_29[0][0]
↪']

 conv5_2_1x1_reduce (Conv2D)    (None, 7, 7, 512)    1048576     ['activation_
↪91[0][0]']

 conv5_2_1x1_reduce/bn (BatchNo  (None, 7, 7, 512)   2048        ['conv5_2_1x1_
↪reduce[0][0]']
 rmalization)

 activation_92 (Activation)     (None, 7, 7, 512)    0           ['conv5_2_1x1_
↪reduce/bn[0][0]']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
conv5_2_3x3 (Conv2D)           (None, 7, 7, 512)    2359296     ['activation_
↪92[0][0]']

conv5_2_3x3/bn (BatchNormaliza  (None, 7, 7, 512)   2048        ['conv5_2_
↪3x3[0][0]']
tion)

activation_93 (Activation)     (None, 7, 7, 512)    0           ['conv5_2_3x3/
↪bn[0][0]']

conv5_2_1x1_increase (Conv2D)  (None, 7, 7, 2048)   1048576     ['activation_
↪93[0][0]']

conv5_2_1x1_increase/bn (Batch  (None, 7, 7, 2048)  8192        ['conv5_2_1x1_
↪increase[0][0]']
Normalization)

add_30 (Add)                   (None, 7, 7, 2048)   0           ['conv5_2_1x1_
↪increase/bn[0][0]',
                                'activation_91[0][0]']

activation_94 (Activation)     (None, 7, 7, 2048)   0           ['add_30[0][0]
↪']

conv5_3_1x1_reduce (Conv2D)    (None, 7, 7, 512)    1048576     ['activation_
↪94[0][0]']

conv5_3_1x1_reduce/bn (BatchNo  (None, 7, 7, 512)   2048        ['conv5_3_1x1_
↪reduce[0][0]']
rmalization)

activation_95 (Activation)     (None, 7, 7, 512)    0           ['conv5_3_1x1_
↪reduce/bn[0][0]']

conv5_3_3x3 (Conv2D)           (None, 7, 7, 512)    2359296     ['activation_
↪95[0][0]']

conv5_3_3x3/bn (BatchNormaliza  (None, 7, 7, 512)   2048        ['conv5_3_
↪3x3[0][0]']
tion)

activation_96 (Activation)     (None, 7, 7, 512)    0           ['conv5_3_3x3/
↪bn[0][0]']

conv5_3_1x1_increase (Conv2D)  (None, 7, 7, 2048)   1048576     ['activation_
↪96[0][0]']

conv5_3_1x1_increase/bn (Batch  (None, 7, 7, 2048)  8192        ['conv5_3_1x1_
↪increase[0][0]']
Normalization)

```

(continues on next page)

(продолжение с предыдущей страницы)

```
 add_31 (Add)                    (None, 7, 7, 2048)   0           ['conv5_3_1x1_
↪increase/bn[0][0]',
                                  'activation_94[0][0]']

 activation_97 (Activation)      (None, 7, 7, 2048)   0           ['add_31[0][0]
↪']

 avg_pool (AveragePooling2D)     (None, 1, 1, 2048)   0           ['activation_
↪97[0][0]']

 global_average_pooling2d_1 (Gl  (None, 2048)         0           ['avg_
↪pool[0][0]']
 obalAveragePooling2D)

 gaussian_noise_1 (GaussianNois  (None, 2048)         0           ['global_
↪average_pooling2d_1[0][0
 e)                                                              ]']

 dense_x (Dense)                 (None, 512)          1049088     ['gaussian_
↪noise_1[0][0]']

 dropout_1 (Dropout)             (None, 512)          0           ['dense_x[0][0]
↪']

 dense_1 (Dense)                 (None, 7)            3591        ['dropout_
↪1[0][0]']

==================================================================================================
Total params: 24,613,831
Trainable params: 24,560,711
Non-trainable params: 53,120
__________________________________________________________________________________
↪________________
--- Время выполнения: 2.222 сек. ---

True
```

Ошибка – 1 –

In [2]:

```
from oceanai.modules.lab.video import Video

video = Video()
video.load_video_model_deep_fe(
    show_summary = 1, out = True,
    runtime = True, run = True
)
```

[2]:

```
[2022-11-01 12:21:23] Неверные типы или значения аргументов в "Video.load_video_
↪model_deep_fe" ...

False
```

load_video_model_hc(*lang: str*, *show_summary: bool = False*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам

Параметры

- lang (*str*) – Язык
- show_summary (*bool*) – Отображение сформированной нейросетевой архитектуры модели
- out (*bool*) – Отображение
- runtime (*bool*) – Подсчет времени выполнения
- run (*bool*) – Блокировка выполнения

Результат

True если нейросетевая архитектура модели сформирована, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()
video.load_video_model_hc(
    show_summary = False, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-10-25 16:37:43] Формирование нейросетевой архитектуры модели для
↪получения оценок по экспертным признакам (видео модальность) ...

--- Время выполнения: 0.659 сек. ---

True
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.lab.video import Video

video = Video()
video.load_video_model_hc(
    show_summary = 1, out = True,
    runtime = True, run = True
)
```

[2]:

```
[2022-10-26 12:27:41] Неверные типы или значения аргументов в "Video.load_video_
↪model_hc" ...

False
```

load_video_model_nn(*show_summary: bool = False*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам

Параметры

- **show_summary** (*bool*) – Отображение сформированной нейросетевой архитектуры модели
- **out** (*bool*) – Отображение
- **runtime** (*bool*) – Подсчет времени выполнения
- **run** (*bool*) – Блокировка выполнения

Результат

True если нейросетевая архитектура модели сформирована, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()
video.load_video_model_nn(
    show_summary = True, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-10-27 14:46:11] Формирование нейросетевой архитектуры для получения␣
↪оценок по нейросетевым признакам (видео модальность) ...

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 10, 512)]         0

 lstm (LSTM)                 (None, 1024)              6295552

 dropout (Dropout)           (None, 1024)              0

 dense (Dense)               (None, 5)                 5125

```

(continues on next page)

(продолжение с предыдущей страницы)

```
 activation (Activation)     (None, 5)                 0

=================================================================
Total params: 6,300,677
Trainable params: 6,300,677
Non-trainable params: 0
_________________________________________________________________
--- Время выполнения: 2.018 сек. ---

True
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.lab.video import Video

video = Video()
video.load_video_model_nn(
    show_summary = 1, out = True,
    runtime = True, run = True
)
```

[2]:
```
[2022-10-27 14:47:22] Неверные типы или значения аргументов в "Video.load_video_
↪model_nn" ...

False
```

load_video_model_weights_deep_fe(*url: str*, *force_ reload: bool = True*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Загрузка весов нейросетевой модели для получения нейросетевых признаков

Параметры

- `url` (*str*) – Полный путь к файлу с весами нейросетевой модели
- `force_reload` (*bool*) – Принудительная загрузка файла с весами нейросетевой модели из сети
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

True если веса нейросетевой модели загружены, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.load_video_model_deep_fe(
    show_summary = False, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-11-01 12:41:59] Формирование нейросетевой архитектуры для получения␣
↪нейросетевых признаков (видео модальность) ...

--- Время выполнения: 1.306 сек. ---

True
```

In [2]:
```
video.path_to_save_ = './models'
video.chunk_size_ = 2000000

url = video.weights_for_big5_['video']['fe']['sberdisk']

video.load_video_model_weights_deep_fe(
    url = url,
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[2]:
```
[2022-11-01 12:42:51] Загрузка весов нейросетевой модели для получения␣
↪нейросетевых признаков (видео модальность) ...

[2022-11-01 12:43:06] Загрузка файла "weights_2022-11-01_12-27-07.h5" (100.0%) .
↪..

--- Время выполнения: 14.781 сек. ---

True
```

Ошибка – 1 –

In [3]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.path_to_save_ = './models'
video.chunk_size_ = 2000000
```

(continues on next page)

(продолжение с предыдущей страницы)

```
7
8  url = video.weights_for_big5_['video']['fe']['sberdisk']
9
10 video.load_video_model_weights_deep_fe(
11     url = url,
12     force_reload = True,
13     out = True,
14     runtime = True,
15     run = True
16 )
```

[3]:

```
1  [2022-11-01 12:44:14] Загрузка весов нейросетевой модели для получения␣
   ↪нейросетевых признаков (видео модальность) ...
2
3  [2022-11-01 12:44:28] Загрузка файла "weights_2022-11-01_12-27-07.h5" (100.0%) .
   ↪..
4
5  [2022-11-01 12:44:28] Ой! Что-то пошло не так ... нейросетевая архитектура␣
   ↪модели для получения нейросетевых признаков не сформирована (видео␣
   ↪модальность) ...
6
7  --- Время выполнения: 13.926 сек. ---
8
9  False
```

load_video_model_weights_hc(*url: str*, *force_reload: bool = True*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Загрузка весов нейросетевой модели для получения оценок по экспертным признакам

Параметры

- `url` (*str*) – Полный путь к файлу с весами нейросетевой модели
- `force_reload` (*bool*) – Принудительная загрузка файла с весами нейросетевой модели из сети
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

True если веса нейросетевой модели загружены, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.load_video_model_hc(
    show_summary = False, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-10-27 12:55:31] Формирование нейросетевой архитектуры модели для␣
↪получения оценок по экспертным признакам (видео модальность) ...

--- Время выполнения: 0.606 сек. ---

True
```

In [2]:
```
video.path_to_save_ = './models'
video.chunk_size_ = 2000000

url = video.weights_for_big5_['video']['hc']['sberdisk']

video.load_video_model_weights_hc(
    url = url,
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[2]:
```
[2022-10-27 13:08:04] Загрузка весов нейросетевой модели для получения оценок␣
↪по экспертным признакам (видео модальность) ...

[2022-10-27 13:08:05] Загрузка файла "weights_2022-08-27_18-53-35.h5" (100.0%) .
↪..

--- Время выполнения: 0.493 сек. ---

True
```

Ошибка – 1 –

In [3]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.path_to_save_ = './models'
video.chunk_size_ = 2000000
```

(continues on next page)

(продолжение с предыдущей страницы)

```
7
8  url = video.weights_for_big5_['video']['hc']['sberdisk']
9
10 video.load_video_model_weights_hc(
11     url = url,
12     force_reload = True,
13     out = True,
14     runtime = True,
15     run = True
16 )
```

[3]:

```
1  [2022-10-27 13:09:54] Загрузка весов нейросетевой модели для получения оценок
   ↪по экспертным признакам (видео модальность) ...
2
3  [2022-10-27 13:09:54] Загрузка файла "weights_2022-08-27_18-53-35.h5" (100.0%) .
   ↪..
4
5  [2022-10-27 13:09:54] Ой! Что-то пошло не так ... нейросетевая архитектура
   ↪модели для получения оценок по экспертным признакам не сформирована (видео
   ↪модальность) ...
6
7  --- Время выполнения: 0.424 сек. ---
8
9  False
```

load_video_model_weights_nn(*url: str*, *force_reload: bool = True*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам

Параметры

- `url` (*str*) – Полный путь к файлу с весами нейросетевой модели
- `force_reload` (*bool*) – Принудительная загрузка файла с весами нейросетевой модели из сети
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

True если веса нейросетевой модели загружены, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.load_video_model_nn(
    show_summary = False, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-10-27 15:17:13] Формирование нейросетевой архитектуры для получения␣
↪оценок по нейросетевым признакам (видео модальность) ...

--- Время выполнения: 1.991 сек. ---

True
```

In [2]:
```
video.path_to_save_ = './models'
video.chunk_size_ = 2000000

url = video.weights_for_big5_['video']['nn']['sberdisk']

video.load_video_model_weights_nn(
    url = url,
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[2]:
```
[2022-10-27 15:19:08] Загрузка весов нейросетевой модели для получения оценок␣
↪по нейросетевым признакам (видео модальность) ...

[2022-10-27 15:19:11] Загрузка файла "weights_2022-03-22_16-31-48.h5" (100.0%) .
↪..

--- Время выполнения: 3.423 сек. ---

True
```

Ошибка – 1 –

In [3]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.path_to_save_ = './models'
video.chunk_size_ = 2000000
```

(continues on next page)

(продолжение с предыдущей страницы)

```
url = video.weights_for_big5_['video']['nn']['sberdisk']

video.load_video_model_weights_nn(
    url = url,
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[3]:

```
[2022-10-27 15:19:40] Загрузка весов нейросетевой модели для получения оценок по нейросетевым признакам (видео модальность) ...

[2022-10-27 15:19:43] Загрузка файла "weights_2022-03-22_16-31-48.h5" (100.0%) ...

[2022-10-27 15:19:43] Ой! Что-то пошло не так ... нейросетевая архитектура модели для получения оценок по нейросетевым признакам не сформирована (видео модальность) ...

--- Время выполнения: 3.469 сек. ---

False
```

load_video_models_b5(*show_summary: bool = False*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Формирование нейросетевых архитектур моделей для получения результатов оценки персональных качеств

Параметры

- `show_summary` (*bool*) – Отображение последней сформированной нейросетевой архитектуры моделей
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

True если нейросетевые архитектуры модели сформированы, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()
video.load_video_models_b5(
    show_summary = True, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-11-04 15:29:26] Формирование нейросетевых архитектур моделей для␣
↪получения результатов оценки персональных качеств (видео модальность) ...

Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32)]              0

 dense_1 (Dense)             (None, 1)                 33

 activ_1 (Activation)        (None, 1)                 0

=================================================================
Total params: 33
Trainable params: 33
Non-trainable params: 0
_________________________________________________________________
--- Время выполнения: 0.116 сек. ---

True
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.lab.video import Video

video = Video()
video.load_video_models_b5(
    show_summary = 1, out = True,
    runtime = True, run = True
)
```

[2]:
```
[2022-11-04 15:30:15] Неверные типы или значения аргументов в "Video.load_video_
↪models_b5" ...

False
```

load_video_models_weights_b5(*url_openness: str*, *url_conscientiousness: str*, *url_extraversion: str*, *url_agreeableness: str*, *url_non_neuroticism: str*, *force_reload: bool = True*, *out: bool = True*, *runtime: bool = True*, *run: bool = True*) → bool

Загрузка весов нейросетевых моделей для получения результатов оценки персональных качеств

Параметры

- `url_openness` (*str*) – Полный путь к файлу с весами нейросетевой модели (открытость опыту)
- `url_conscientiousness` (*str*) – Полный путь к файлу с весами нейросетевой модели (добросовестность)
- `url_extraversion` (*str*) – Полный путь к файлу с весами нейросетевой модели (экстраверсия)
- `url_agreeableness` (*str*) – Полный путь к файлу с весами нейросетевой модели (доброжелательность)
- `url_non_neuroticism` (*str*) – Полный путь к файлу с весами нейросетевой модели (эмоциональная стабильность)
- `force_reload` (*bool*) – Принудительная загрузка файлов с весами нейросетевых моделей из сети
- `out` (*bool*) – Отображение
- `runtime` (*bool*) – Подсчет времени выполнения
- `run` (*bool*) – Блокировка выполнения

Результат

True если веса нейросетевых моделей загружены, в обратном случае **False**

Тип результата

bool

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.load_video_models_b5(
    show_summary = False, out = True,
    runtime = True, run = True
)
```

[1]:
```
[2022-11-04 18:56:41] Формирование нейросетевых архитектур моделей для␣
↪получения результатов оценки персональных качеств (видео модальность) ...

--- Время выполнения: 0.117 сек. ---

True
```

In [2]:

```
video.path_to_save_ = './models'
video.chunk_size_ = 2000000

url_openness = video.weights_for_big5_['video']['b5']['openness']['sberdisk']
url_conscientiousness = video.weights_for_big5_['video']['b5']['conscientiousness']['sberdisk']
url_extraversion = video.weights_for_big5_['video']['b5']['extraversion']['sberdisk']
url_agreeableness = video.weights_for_big5_['video']['b5']['agreeableness']['sberdisk']
url_non_neuroticism = video.weights_for_big5_['video']['b5']['non_neuroticism']['sberdisk']

video.load_video_models_weights_b5(
    url_openness = url_openness,
    url_conscientiousness = url_conscientiousness,
    url_extraversion = url_extraversion,
    url_agreeableness = url_agreeableness,
    url_non_neuroticism = url_non_neuroticism,
    force_reload = True,
    out = True,
    runtime = True,
    run = True
)
```

[2]:

```
[2022-11-04 18:58:59] Загрузка весов нейросетевых моделей для получения результатов оценки персональных качеств (видео модальность) ...

[2022-11-04 18:59:00] Загрузка файла "weights_2022-06-15_16-46-30.h5" (100.0%) ... Открытость опыту

[2022-11-04 18:59:00] Загрузка файла "weights_2022-06-15_16-48-50.h5" (100.0%) ... Добросовестность

[2022-11-04 18:59:00] Загрузка файла "weights_2022-06-15_16-54-06.h5" (100.0%) ... Экстраверсия

[2022-11-04 18:59:01] Загрузка файла "weights_2022-06-15_17-02-03.h5" (100.0%) ... Доброжелательность

[2022-11-04 18:59:01] Загрузка файла "weights_2022-06-15_17-06-15.h5" (100.0%) ... Эмоциональная стабильность

--- Время выполнения: 1.827 сек. ---

True
```

Ошибка – 1 –

In [3]:

```
from oceanai.modules.lab.video import Video

video = Video()
```

(continues on next page)

(продолжение с предыдущей страницы)

```
4
5  video.path_to_save_ = './models'
6  video.chunk_size_ = 2000000
7
8  url_openness = video.weights_for_big5_['video']['b5']['openness']['sberdisk']
9  url_conscientiousness = video.weights_for_big5_['video']['b5']['
   ↪'conscientiousness']['sberdisk']
10 url_extraversion = video.weights_for_big5_['video']['b5']['extraversion']['
   ↪'sberdisk']
11 url_agreeableness = video.weights_for_big5_['video']['b5']['agreeableness']['
   ↪'sberdisk']
12 url_non_neuroticism = video.weights_for_big5_['video']['b5']['non_neuroticism']['
   ↪'sberdisk']
13
14 video.load_video_models_weights_b5(
15     url_openness = url_openness,
16     url_conscientiousness = url_conscientiousness,
17     url_extraversion = url_extraversion,
18     url_agreeableness = url_agreeableness,
19     url_non_neuroticism = url_non_neuroticism,
20     force_reload = True,
21     out = True,
22     runtime = True,
23     run = True
24 )
```

[3]:

```
1  [2022-11-04 19:02:32] Загрузка весов нейросетевых моделей для получения␣
   ↪результатов оценки персональных качеств (видео модальность) ...
2
3  [2022-11-04 19:02:32] Загрузка файла "weights_2022-06-15_16-46-30.h5" (100.0%) .
   ↪..
4
5  [2022-11-04 19:02:32] Ой! Что-то пошло не так ... не удалось загрузить веса␣
   ↪нейросетевой модели ... Открытость опыту
6
7      Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/video.py
8      Линия: 2833
9      Метод: load_video_models_weights_b5
10     Тип ошибки: AttributeError
11
12 [2022-11-04 19:02:32] Загрузка файла "weights_2022-06-15_16-48-50.h5" (100.0%) .
   ↪..
13
14 [2022-11-04 19:02:32] Ой! Что-то пошло не так ... не удалось загрузить веса␣
   ↪нейросетевой модели ... Добросовестность
15
16     Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/video.py
17     Линия: 2833
18     Метод: load_video_models_weights_b5
19     Тип ошибки: AttributeError
20
```

(continues on next page)

(продолжение с предыдущей страницы)

```
[2022-11-04 19:02:33] Загрузка файла "weights_2022-06-15_16-54-06.h5" (100.0%) ...

[2022-11-04 19:02:33] Ой! Что-то пошло не так ... не удалось загрузить веса нейросетевой модели ... Экстраверсия

    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/video.py
    Линия: 2833
    Метод: load_video_models_weights_b5
    Тип ошибки: AttributeError

[2022-11-04 19:02:33] Загрузка файла "weights_2022-06-15_17-02-03.h5" (100.0%) ...

[2022-11-04 19:02:33] Ой! Что-то пошло не так ... не удалось загрузить веса нейросетевой модели ... Доброжелательность

    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/video.py
    Линия: 2833
    Метод: load_video_models_weights_b5
    Тип ошибки: AttributeError

[2022-11-04 19:02:34] Загрузка файла "weights_2022-06-15_17-06-15.h5" (100.0%) ...

[2022-11-04 19:02:34] Ой! Что-то пошло не так ... не удалось загрузить веса нейросетевой модели ... Эмоциональная стабильность

    Файл: /Users/dl/GitHub/oceanai/oceanai/modules/lab/video.py
    Линия: 2833
    Метод: load_video_models_weights_b5
    Тип ошибки: AttributeError

--- Время выполнения: 1.831 сек. ---

False
```

property video_model_deep_fe_: Optional[Model]

Получение нейросетевой модели **tf.keras.Model** для получения нейросетевых признаков

Результат

Нейросетевая модель **tf.keras.Model** или None

Тип результата

Optional[tf.keras.Model]

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.load_video_model_deep_fe(
    show_summary = False, out = True,
    runtime = True, run = True
)

video.video_model_deep_fe_
```

[1]:
```
[2022-11-01 12:12:35] Формирование нейросетевой архитектуры для получения
↪нейросетевых признаков (видео модальность) ...

--- Время выполнения: 1.468 сек. ---

<tf.keras.Model at 0x14e138100>
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.video_model_deep_fe_
```

[2]:
```

```

property video_model_hc_: Optional[Model]

Получение нейросетевой модели **tf.keras.Model** для получения оценок по экспертным признакам

Результат
Нейросетевая модель **tf.keras.Model** или None

Тип результата
Optional[tf.keras.Model]

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.load_video_model_hc(
    show_summary = False, out = True,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    runtime = True, run = True
)

video.video_model_hc_
```

```
[1]:
[2022-10-26 12:37:42] Формирование нейросетевой архитектуры модели для получения оценок по экспертным признакам (видео модальность) ...

--- Время выполнения: 1.112 сек. ---

<tf.keras.Model at 0x1434eb1f0>
```

Ошибка – 1 –

```
In [2]:
from oceanai.modules.lab.video import Video

video = Video()

video.video_model_hc_
```

```
[2]:

```

property video_model_nn_: Optional[Model]

Получение нейросетевой модели **tf.keras.Model** для получения оценок по нейросетевым признакам

Результат

Нейросетевая модель **tf.keras.Model** или None

Тип результата

Optional[tf.keras.Model]

Примеры

Верно – 1 –

```
In [1]:
from oceanai.modules.lab.video import Video

video = Video()

video.load_video_model_nn(
    show_summary = False, out = True,
    runtime = True, run = True
)

video.video_model_nn_
```

```
[1]:
[2022-10-27 14:49:00] Формирование нейросетевой архитектуры для получения оценок по нейросетевым признакам (видео модальность) ...

```

(continues on next page)

(продолжение с предыдущей страницы)

```
--- Время выполнения: 1.986 сек. ---

<tf.keras.Model at 0x13d5295b0>
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.video_model_nn_
```

[2]:
```

```

property video_models_b5_: Dict[str, Optional[Model]]

Получение нейросетевых моделей **tf.keras.Model** для получения результатов оценки персональных качеств

Результат

Словарь с нейросетевыми моделями **tf.keras.Model**

Тип результата

Dict

Примеры

Верно – 1 –

In [1]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.load_video_models_b5(
    show_summary = False, out = True,
    runtime = True, run = True
)

video.video_models_b5_
```

[1]:
```
[2022-10-19 15:45:35] Формирование нейросетевых архитектур моделей для␣
↪получения результатов оценки персональных качеств ...

--- Время выполнения: 0.07 сек. ---

{
    'openness': <tf.keras.Model at 0x1481e03a0>,
    'conscientiousness': <tf.keras.Model at 0x147d13520>,
    'extraversion': <tf.keras.Model at 0x1481edfa0>,
    'agreeableness': <tf.keras.Model at 0x1481cfc40>,
    'non_neuroticism': <tf.keras.Model at 0x1481cffd0>
}
```

Ошибка – 1 –

In [2]:
```
from oceanai.modules.lab.video import Video

video = Video()

video.video_models_b5_
```

[2]:
```
{
    'openness': None,
    'conscientiousness': None,
    'extraversion': None,
    'agreeableness': None,
    'non_neuroticism': None
}
```

Текст

Мультимодальное объединение информации

Сборка

Пользовательские исключения
Документация для файла `modules/core/exceptions.py`

Определение языка
Документация для файла `modules/core/language.py`

Сообщения
Документация для файла `modules/core/messages.py`

Настройки
Документация для файла `modules/core/settings.py`

Ядро
Документация для файла `modules/core/core.py`

Обработка архивов
Документация для файла `modules/lab/unzip.py`

Загрузка файлов
Документация для файла `modules/lab/download.py`

Аудио
Документация для файла `modules/lab/audio.py`

Видео
Документация для файла `modules/lab/video.py`

Текст
Документация для файла `modules/lab/text.py`

Мультимодальное объединение информации
Документация для файла `modules/lab/prediction.py`

Сборка
Документация для файла `modules/lab/build.py`

4.2.3 Диаграмма классов

Рис. 1: Исключения

Рис. 2: Основные классы

4.2.4 Команда разработчиков

Библиотека `OCEAN-AI` разработана и поддерживается исследовательской группой из лаборатории речевых и многомодальных интерфейсов (ЛРМИ) Санкт-Петербургского Федерального исследовательского центра Российской академии наук (СПб ФИЦ РАН), входящей в состав НИЦ Сильный искусственный интеллект в промышленности (Университет ИТМО) а именно:

- Рюмина Елена - аспирант Университета ИТМО, младший научный сотрудник ЛРМИ СПб ФИЦ РАН.
- Рюмин Дмитрий - кандидат технических наук (PhD in Engineering, 2020), старший научный сотрудник ЛРМИ СПб ФИЦ РАН.
- Карпов Алексей - доктор технических наук (2013), профессор по специальности 05.13.11 (2022), главный научный сотрудник (руководитель) ЛРМИ СПб ФИЦ РАН.

4.2.5 Ответы на часто задаваемые вопросы

Это список часто задаваемых вопросов о *OCEAN-AI* и ответов на них.

OCEAN-AI это?

Библиотека с открытым исходным кодом, состоящая из набора алгоритмов интеллектуального анализа поведения человека на основе его мультимодальных данных для автоматического оценивания уровня отдельных персональных качеств личности человека (ПКЛЧ). Библиотека оценивает 5 ПКЛЧ: открытость опыту, добросовестность, экстраверсия, доброжелательность, эмоциональная стабильность.

Почему библиотека называется _OCEAN-AI_?

OCEAN - это аббревиатура 5 ПКЛЧ (Openness, Conscientiousness, Extraversion, Agreeableness, Non-Neuroticism), а AI - это аббревиатура искусственного интеллекта.

Можно ли установить _OCEAN-AI_ с помошью pip?

Да, для этого нужно перейти по ссылке.

ГЛАВА 5

Индексация

- genindex
- modindex
- search

Содержание модулей Python

Символы

A

B

C

D

R

S

T

U

V

W

модуль