Сколковский институт науки и технологий

Руководство системного программиста

Платформа DATASKAI

версия: 0.0.1

Москва
2020

Оглавление

Данное руководство содержит описания программных интерфейсов для сервисов и инструментариев, включенных в платформу DATASKAI.

Глава 1

Models Player

Models Player создан в виде веб-сервиса Flask. Он работает как контроллер моделей в боевой среде. Сервис предоставляет пользователям функции управления моделями: запуск, остановку моделей, выполнение предсказаний, получение статистики и т.п.

Внешний API Models Player работает по протоколу http. Внутренний, по которому Models Player запускается и управляет моделями - по grpc.

GET /api/get_models_settings; POST /api/get_models_settings

Возвращает параметры запущенных моделей.

Ответ включает в себя идентификатор модели, порт и конфигурацию (лог-файл по умолчанию, имя модели, имя задачи, идентификатор процесса (pid) и т.д.)

Можно получить параметры конкретной модели, если указать директорию, ID или имя класса (имя Python-класса, в котором описывается поведение модели в файле model.py). Если модель не уточняется, метод возвращает информацию обо всех зарегистрированных моделях.

результат параметры конкретной модели, если она идентифицирована, в противном случае - параметры всех зарегистрированных моделей.

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/get_models_settings?model_id=AllSumModel
```

Пример ответа:

```
{
"model_config": {
    "default_log_file": "./model_wrapper/logs/model.log",
    "description": "",
    "feature_name_aliases": {
    "feature_10": "feature_3",
    "feature_9": "feature_2"
    },
    "host": "localhost:10100",
    "logging_level": "info",
    "message": "SumVector",
    "model_class_name": "AllSumModel",
    "model_features": [
```

(continues on next page)

(продолжение с предыдущей страницы)


```
    "feature_1",
    "feature_2",
    "feature_3"
    ],
    "model_filename": "model.py",
    "model_name": "AllSumModel",
    "model_threshold": 0.0077,
    "old_model_features": [
    "feature_1",
    "feature_9",
    "feature_10"
    ],
    "pid": 10857,
    "tags": [
    "is_prod_model"
    ],
    "window": 50
},
"model_directory": "/home/j/Desktop/dev/python/models_player/model_player/src/../../models/
↪AllSumModel",
"model_id": "AllSumModel",
"model_port": 10100
}
```


GET /api/get_models_stats; POST /api/get_models_stats

Возвращает статистику (время запуска, время инициализации и т.д.) для всех запущенных моделей.

результат статистика по конкретной модели, если модель идентифицирована, в противном случае - статистика по всем зарегистрированным моделям в виде списка.

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/get_models_stats
```

Пример ответа:


```
[
{
    "model_config": {
        "init_done_at": "24.05.2019 12:45:33",
        "init_time(sec)": 0.000372,
        "started_at": "24.05.2019 12:45:33",
        "total_X_vectors_processed": 1,
        "uptime_total(sec)": 842.860546
    },
    "model_directory": "/home/j/Desktop/dev/python/models_player/model_player/src/../../models/
↪AllSumModel",
    "model_id": "AllSumModel",
    "model_port": 10100
}
]
```


GET /api/start_model; POST /api/start_model

Запускает модель с выбранным model_id.

Модель запускается в новом процессе. Запуск происходит только в том случае, если модель еще не была запущена. ID модели должен быть передан GET или POST методом.

Также метод возвращает следующие статусы:

- ignored (если по время запуска модели произошли некоторые ошибки);
- started (если модель была запущена успешно);
- working (если модель уже была запущена на момент попытки старта).

результат директория и статус модели.

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/start_model?model_id=AllSumModel
```

Пример ответа:

```
{"all_started_models": {
    "/home/models_player/model_player/src/../../models/AllSumModel": "started"
    }
}
```

GET /api/start_all_models; POST /api/start_all_models

Запускает все модели в директории моделей.

Target directory is described in players environment variable MODELS_PATH or players config. Each model starts in a new process, but only if it isn't already started.

Для каждой модели метод возвращает один из следующих статусов:

- ignored (если по время запуска модели произошли некоторые ошибки);
- started (если модель была запущена успешно);
- working (если модель уже была запущена на момент попытки старта).

результат расположение и статусы моделей.

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/start_all_models
```

Пример ответа:

```
{"all_started_models": {
        "/home/models_player/model_player/src/../../models/AllMultModel": "started",
        "/home/models_player/model_player/src/../../models/AllSumModel": "working"
    }
}
```

GET /api/stop_all_models; POST /api/stop_all_models

Останавливает все запущенные модели.

результат расположение моделей и сообщения о результатах их остановки.

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/stop_all_models
```

Пример ответа:

```
{"stopped_models": {
    "/home/j/Desktop/dev/python/models_player/model_player/src/../../models/AllMultModel":␣
↪        "shutdown after 5 seconds",
    "/home/j/Desktop/dev/python/models_player/model_player/src/../../models/AllSumModel":␣
↪       "shutdown after 5 seconds"
    }
}
```

GET /api/stop_model; POST /api/stop_model

Останавливает модель с выбранным ID.

ID модели может быть передан через методы GET и POST.

результат расположение и результат остановки модели.

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/stop_model?model_id=AllSumModel
```

Пример ответа. Остановка запущенной модели:

```
{
    "/home/j/Desktop/dev/python/models_player/model_player/src/../../models/AllSumModel":
↪"shutdown after 5 seconds"
}
```

Пример ответа. Попытка остановки модели, которая не была запущена:

```
{
    "error": "model_id is not in running models. Use /api/list_running_models to get all␣
↪running models at the moment"
}
```

GET /api/predict_one; POST /api/predict_one

Передает модели вектор признаков X и возвращает предсказание.

Для получения результата необходимо передать вектор X и ID модели, либо ее директорию или имя класса (Python-класс из model.py) с помощью метода GET или POST.

результат key-value {«y_pred»: prediction result}.

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/predict_one?model_id=AllSumModel&X=1,10,14
```

Пример ответа:

```
{
    "y_pred": 25.0
}
```

Пример неверного запроса:

```
http://localhost:9000/api/predict_one?model_id=AllSumModel
```

Пример ответа:

```
{
    "error": "no X vector provided"
}
```

POST /api/predict_batch

Принимает список веторов признаков X через POST-запрос и возвращает список предсказаний. Работает только через POST-запрос.

результат json в виде списка со словарями {«y_pred»: result} для каждого вектора X.

тип результата flask.wrappers.Response(json)

Пример запроса:

```
POST http://0.0.0.0:9066/api/predict_batch
Content-Type: application/json

{
    "model_id":"AllSumModel",
    "X":[
            [1,2,3],
            [1.1,2,3]
        ]
}
```

Пример ответа:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 58
Server: Werkzeug/0.14.1 Python/3.6.8
Date: Tue, 28 May 2019 13:18:59 GMT

[
    {"y_pred": 6.0},
    {"y_pred": 6.1}
]
```

GET /api/upload_model; POST /api/upload_model

Метод для загрузки в Models Player архива с моделью.

Saves archive file to corresponding directory (customs in environment variable ARCHIVED_MODELS_PATH or player configuration). The file must have .zip or .tar.gz format.

Если вы используете метод GET, сервер вернет веб-форму для загрузки модели. Возможно также использование POST-метода. Для корректной работы директория для сохранения архивов должна быть создана заранее.

Если загрузка была успешной, происходит перенаправление на /list_archives_dir

Пример запроса:

```
http://localhost:9000/api/upload_model
```

GET /api/unpack_model

Метод для распаковки модели из архива в директорию.

Идентифицирует архив по параметру filename, переданном в строке GET-запроса. Имя файла должно иметь расширение .zip или .tar.gz. Модель распаковывается в директорию, определенную в конфигурации Models Player.

Если загрузка была успешной, происходит перенаправление на /list_models_dir

результат список папок в директории моделей.

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/unpack_model?filename=new_model.tar.gz
```

Пример ответа:

```
[
    "AllSumModel",
    "AllMultModel",
    "new_model"
]
```

GET /api/download_model

Загружает архив с моделью на локальный диск пользователя.

Идентифицирует архив по параметру filename, переданном в строке GET-запроса. Имя файла должно содержать расширение .zip или .tar.gz.

результат модель в архиве .tar.gz

Пример запроса:

```
http://localhost:9000/api/download_model?filename=AllSumModel.tar.gz
```

GET /api/pack_model

Метод для упаковки модели в архив.

Identifies model by ID, which should be passed in GET request. Packs model to .tar.gz archive and puts it to models archives directory (it is described in environment variable ARCHIVED_MODELS_PATH or player configuration).

результат список файлов в директории с архивами моделей

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/pack_model?model_id=AllSumModel
```

Пример ответа:

```
[
    "AllSumModel.tar.gz",
]
```

GET /api/list_models_dir

Возвращает список моделей (директорий моделей).

Path to target directory is defined in environment variable MODELS_PATH or player configuration.

результат список моделей (директорий моделей).

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/list_models_dir
```

Пример ответа:

```
[
    "AllSumModel",
    "AllMultModel"
]
```

GET /api/list_running_models

Возвращает список запущенных моделей (директорий моделей).

Path to target directory is defined in environment variable MODELS_PATH or player configuration.

результат список запущенных моделей.

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/list_running_models
```

Пример ответа:

```
[
    "AllSumModel",
    "AllMultModel"
]
```

GET /api/list_archives_dir

Возвращает список архивов с моделями.

A directory with archives is defined in environment variable ARCHIVED_MODELS_PATH or player configuration.

результат список файлов в директории с архивами моделей.

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/list_archives_dir
```

Пример ответа:

```
[
    "AllSumModel.tar.gz",
    "AllMultModel.tar.gz"
]
```

GET /api/remove_archive

Метод для удаления архива модели из директории с архивами.

Идентифицирует архив по параметру filename, переданном в строке GET-запроса. Имя файла должно содержать расширение .zip или .tar.gz.

Если удаление прошло успешно, возвращает список файлов в директории с архивами.

результат список файлов в директории с архивами моделей

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/remove_archive?filename=new_model.tar.gz
```

Пример ответа:

```
[
    "AllSumModel.tar.gz",
    "AllMultModel.tar.gz"
]
```

GET /api/remove_model_dir

Удаляет директорию модели.

Идентифицирует модель по параметру model_id, переданному в строке GET-запроса.

Если удаление прошло успешно, возвращает список файлов в директории с моделями.

результат список папок моделей в директории с моделями.

тип результата flask.wrappers.Response(json)

Пример запроса:

```
http://localhost:9000/api/remove_model_dir?model_id=new_model
```

Пример ответа:

```
[
    "AllSumModel",
    "AllMultModel"
]
```

models_player.init_all_models_sequentially (*player_config*, *logger=None*)

Starts up all models in models directory, which is described in environment variable MODELS_PATH or player config file.

Результат директория и статус модели.

models_player.init_one_model (*player_config*, *model_id*, *logger=None*)

Инициализирует модель с требуемым model_id, если она еще не запущена.

Для каждой модели метод возвращает один из следующих статусов: игнорировать, запустить, уже запущено.

Результат словарь {директория: статус модели}

models_player.stop_all_models_sequentially (*player_config*)

Запускает все модели в ALL_MODELS_DICT.

Удаляет записи о моделях из ALL_MODELS_DICT, ALL_MODELS_PROCESSES, ALL_MODELS_ID_TO_DIR, ALL_MODELS_CLASSNAME_TO_DIR. Освобождает порты.

Тип результата dict

Глава 2

Metrics Service

GET /api/v1/system/status

—

Check that service is alive.

Request example:

```
curl 'http://0.0.0.0:17000/api/v1/system/status'
```

Response example:

```
{
    "service_name": "metrics_service",
    "status": "OK"
}
```

Status Codes

- 200 OK – Returns JSON with service name and status.
- 404 Not Found – Not Found.

GET /api/v1/system/version

—

Method that returns service version in MAJOR.MINOR.PATCH format.

Request example:

```
curl 'http://0.0.0.0:17000/api/v1/system/version'
```

Response example:

```
{
    "service_name": "metrics_service",
    "version": "1.0.0"
}
```

Status Codes

- 200 OK – Returns JSON with service name and version.

- 404 Not Found – Not Found.

Глава 3

Submits Web App

Ресурс	Операция	Описание
	GET /api/boards/(board_name)/submits/(submit_id)/download	
	GET /(board_name)/message/(message)	
API	*GET /api/v1/system/version*	Returns current service version.
	GET /api/v1/system/status	Returns service status.
	GET /api/boards	Returns settings for existing boards from configuration file.
	GET /api/boards/(board_name)/messages/(message)/submits	Возвращает все результаты моделирования для текущей задачи в формате json.
	GET /api/boards/(board_name)/frontend_config	Возвращает параметры визуального интерфейса для таблицы из файла конфигурации.
Home	*GET /(board_name)/message/(message)/submits*	Возвращает html-страницу с таблицей отправленных на сервер результатов моделирования.

GET /api/v1/system/version

Returns current service version.

результат json with service name and version.

Пример запроса:

```
http://0.0.0.0:5000/api/v1/system/version
```

Пример ответа:

```
{
    "service_name": "submits_web_app",
    "version": "0.1.0"
}
```

GET /api/v1/system/status

Returns service status.

результат json with service name and status.

Пример запроса:

```
http://0.0.0.0:5000/api/v1/system/status
```

Пример ответа:

```
{
    "service_name": "submits_web_app",
    "status": "OK"
}
```

GET /api/boards

Returns settings for existing boards.

Parameters are returned in json format, including 2 fields for each board:

- board name;
- messages whitelist.

результат Boards list in json format.

Пример запроса:

```
http://0.0.0.0:5000/api/boards
```

Пример ответа:

```
{
    "data":
    {
        "available_boards":
        [
            {
                "board_name":"population_split__non_cancers_self_reported_glaucoma_v1",
                "messages_whitelist":["non_cancers_self_reported_glaucoma"]
            }
        ]
    }
}
```

GET /api/boards/ (*board_name*)/messages/ *message*/submits Returns all submits for current problem (except «_id» field) in json format.

Ответ содержит следующие поля:

- «author» - имя автора;
- «creation_ts» - время отправки результата;
- «model_name» - имя модели;
- «model_version» - версия модели;
- «tags» - пользовательские теги для модели, например, список примененных признаков;
- «message» - имя текущей задачи;

- «result_gzipped_file» - dataframe с решением пользователя (целевой переменной) в бинарном формате;
- «notebook_gziped_file» - ноутбук пользователя в бинарном формате;
- «feature_mining_comments» - пользовательский комментарий о примененных признаках;
- «model» - объект модели;
- «model_feature_columns» - имена признаков модели;
- «model_feature_importances» - важность признаков;
- «model_fill_na_values» - значения для заполнения неопределенных (NaN) ячеек;
- «data_scaler» - название примененного data scaler;
- метрики (например, «mse», «rmse» и т.д.) - метрики для текущей задачи;
- «signals_hash» - хэш для пользовательского dataframe и других составляющих решения.

Параметры

- board_name (str) – Name of board.
- message (str) – Informative name of current problem.

результат Submits for current problem in json format.

Пример запроса:

```
http://0.0.0.0:5000/api/boards/some_board/messages/some_message/submits
```

Пример ответа:


```
{"data":
        [
            {
                "author": "some_user",
                "creation_ts": 1555581779,
                "model_name": "my_model",
                "model_version": "1.0",
                "tags": null,
                "message": "some_problem",
                "result_gzipped_file": {
                    "$binary": "H4sICFNLuFwC/zQ2NzIwZTQwLTIAdF3JrgS5cbz7WwYN7stfCJDxgAA",
                    "$type": "00"
                    },
                "notebook_gziped_file": {
                    "$binary": "H4sICFNLuFwC/zlhNjFkOThhLWIAzZ3vb/jw4Op7NFNZ3fmjeCAQA=",
                    "$type": "00"
                    },
                "feature_mining_comments": "user features v1",
                "model": null,
                "model_feature_columns": ["feature1", "feature2"],
                "model_feature_importances": [0.4, 0.6]
                "model_fill_na_values": null,
                "data_scaler": null,
                "metric1": 1.55,
                "metric2": 2.75,
```


(continues on next page)

(продолжение с предыдущей страницы)

```
                "metric3": 14.75,
                "signals_hash":
↪"71d269320308472c2f5ae83c8dcb427c1a88a5b6f4ece762cb6e39e42aaea4e6d"
            }
        ]
    }
```

GET /api/boards/ (*board_name*)/submits/

submit_id/download Возвращает архив для сабмита. Архив содержит файлы model.pkl, results_df.csv, notebook.ipynb, data_scaler.pkl.

Параметры

- board_name (str) – Name of board.
- submit_id (str) – ID сабмита.

GET /api/boards/ (*board_name*)/frontend_config

Возвращает параметры визуального интерфейса для таблицы из файла конфигурации.

Parameters are returned in json format which contains 5 elements:

- fields: все отображаемые поля;
- text_fields: поля в текстовом формате;
- sort columns: поля и способ сортировки (по возрастанию или нет);
- searchable_columns: столбцы, по которым производится поиск через окно поиска на странице;
- download_available: is submits downloading available or not.

Параметры

- board_name (str) – Name of board.

результат Frontend parameters in json format.

Пример запроса:

```
http://0.0.0.0:5000/api/boards/some_board/frontend_config
```

Пример ответа:

```
{
    "data": {
        "fields": [
            "_id",
            "creation_ts",
            "model_name",
            "metric1",
            "metric2",
            "metric3",
            "model_version",
            "tags",
            "author",
            "feature_mining_comments",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            "model_feature_columns"
        ],
        "text_fields": [
            "model_name",
            "author",
            "tags",
            "feature_mining_comments",
            "model_feature_columns"
        ],
        "sort_columns": [
            {"field": "metric1", "ascending": true}
        ],
        "searchable_columns": [
            "model_name",
            "tags",
            "author",
            "feature_mining_comments"
        ]
    }
}
```

GET / (*board_name*)/message/

message/submits Возвращает html-страницу с таблицей отправленных на сервер результатов моделирования.

Html-шаблон таблицы находится в директории «submits_web_app/static».

Пример запроса:

```
http://0.0.0.0:5000/some_board/message/some_message/submits
```

Параметры

- board_name (str) – Name of board.
- message (str) – Name of current problem.

результат Result html table.

тип результата flask.wrappers.Response

GET / (*board_name*)/message/

message Redirect user to **GET /(board_name)/message/(message)/submits**.

Глава 4

Prediction Builder

POST /api/{version}/prediction/build

Строит предсказания и возвращает статус построения.

Request:

```
curl -X POST 'http://localhost:8390/api/v1/prediction/build?featureset=1570026953257-
↪3c9422f3-fb82-4c98-a748-2fdf82f29cde&from=1451606400&to=1451650000&obj=VP-BTP&
↪model=RENAMED_ENG_2_REV_PRESSURIZED_QAR'
```

Response:

```
{
    "predictionsBuilt": 0,
    "predictionsFailed": 1,
    "timeTaken": 0,
    "objects": {
        "VP-BTP": {
            "objectId": "VP-BTP",
            "predictionsBuilt": 0,
            "predictionsFailed": 1,
            "models": {
                "RENAMED_ENG_2_REV_PRESSURIZED_QAR": {
                    "predictionsBuilt": 0,
                    "missingFeaturesByTime": {
                        "1451643900": [
                            "ngram_bi_w_20_perc_95_90_85_15_10_5_CS\ **tmp**\ rep_qar\␣
↪**component_40_phase**\ fourier\ **only_6_phase**\ normalize_2s\ **feature_1052",
                            "ngram_bi_w_20_perc_95_90_85_15_10_5_FS**\ tmp\ **rep_qar**\␣
↪component_53_abs\ **fourier**\ only_6_phase\ **normalize_2s**\ feature_1051",
                            "ngram_bi_w_20_perc_95_90_85_15_10_5_DH\ **tmp**\ rep_qar\␣
↪**component_39_phase**\ fourier\ **only_6_phase**\ normalize_2s\ **feature_1051",
                            "ngram_bi_w_20_perc_95_90_85_15_10_5_FH**\ tmp\ **rep_qar**\␣
↪component_25_phase\ **fourier**\ only_6_phase\ **normalize_2s**\ feature_1062",
                            "ngram_bi_w_20_perc_95_90_85_15_10_5_NL\ **tmp**\ rep_qar\␣
↪**component_20_abs**\ fourier\ **only_6_phase**\ normalize_2s\ **feature_1062",
                            "ngram_bi_w_20_perc_95_90_85_15_10_5_BS**\ tmp\ **rep_qar**\␣
↪component_57_phase\ **fourier**\ only_6_phase\ **normalize_2s**\ feature_1052",
                        ]
                    },
                    "missingFeaturesGlobally": [
                        "ngram_bi_w_20_perc_95_90_85_15_10_5_CS\ **tmp**\ rep_qar\␣
↪**component_40_phase**\ fourier\ **only_6_phase**\ normalize_2s\ **feature_1052",
                        "ngram_bi_w_20_perc_95_90_85_15_10_5_FS**\ tmp\ **rep_qar**\␣
↪component_53_abs\ **fourier**\ only_6_phase\ **normalize_2s**\ feature_1051",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                        "ngram_bi_w_20_perc_95_90_85_15_10_5_DH\ **tmp**\ rep_qar\␣
↪**component_39_phase**\ fourier\ **only_6_phase**\ normalize_2s\ **feature_1051",
                        "ngram_bi_w_20_perc_95_90_85_15_10_5_FH**\ tmp\ **rep_qar**\␣
↪component_25_phase\ **fourier**\ only_6_phase\ **normalize_2s**\ feature_1062",
                        "ngram_bi_w_20_perc_95_90_85_15_10_5_NL\ **tmp**\ rep_qar\␣
↪**component_20_abs**\ fourier\ **only_6_phase**\ normalize_2s\ **feature_1062",
                        "ngram_bi_w_20_perc_95_90_85_15_10_5_BS**\ tmp\ **rep_qar**\␣
↪component_57_phase\ **fourier**\ only_6_phase\ **normalize_2s**\ feature_1052",
                    ]
                }
            }
        }
    }
}
```

Parameters

- version (string) – Rest API version.

Параметры запроса

- features_build (string) – ${api.build.params.features_build}
- from (integer) – Начало периода. Поддерживается Unix Timestamp формат (например, 1514764800).
- to (integer) – Конец периода. Поддерживается Unix Timestamp формат (например, 1514764800).

Коды статуса

- 200 OK – Статус предсказания, который включает количество построенных и непостроенных предсказаний, затраченное время и словарь статусов объектов предсказаний.
- 201 Created – Создан
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- objects (object) – Словарь статусов предсказаний, разделенных по объектам.
- objects.*.models (object) – Map (dictionary) of prediction statuses divided by model.
- objects.*.models.*.missingFeaturesByTime (object) – Missing features by time.
- objects.*.models.*.missingFeaturesGlobally (array) – Missing features globally.
- objects.*.models.*.predictionsBuilt (integer) – Count prediction built.
- objects.*.objectId (string) – Object ID.
- objects.*.predictionsBuilt (integer) – Количество построенных предсказаний.
- objects.*.predictionsFailed (integer) – Количество неудачных предсказаний.
- predictionsBuilt (integer) – Количество построенных предсказаний.

- predictionsFailed (integer) – Количество неудачных предсказаний.
- timeTaken (integer) – Время, затраченное на построение предсказаний в секундах.

GET /api/{version}/prediction/export

Экспортирует предсказания в TSV формате.

Request:

```
curl 'http://localhost:8390/api/v1/prediction/export?featureset=1570026953257-3c9422f3-
↪fb82-4c98-a748-2fdf82f29cde&from=1451606400&to=1451650000&obj=VP-BTP&model=RENAMED_ENG_
↪2_REV_PRESSURIZED_QAR'
```

Response:

predictions.tsv

Parameters

- version (string) – Rest API version.

Параметры запроса

- from (integer) – Начало периода. Поддерживается Unix Timestamp формат (например, 1514764800).
- to (integer) – Конец периода. Поддерживается Unix Timestamp формат (например, 1514764800).

Коды статуса

- 200 OK – TSV файл с предсказаниями.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

GET /api/{version}/system/status

Returns service status information in JSON format.

Request:

```
curl 'http://127.0.0.1:8390/api/v1/system/status'
```

Response:

```
{
    "service": "prediction-builder-api",
    "status": "OK",
    "time": 1597155326
}
```

Parameters

- version (string) – Rest API version.

Коды статуса

- 200 OK – Service status information in JSON format.
- 401 Unauthorized – Авторизация не выполнена

- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- service (string) – Service name.
- status (string) – Service status.
- time (integer) – Service request time.

GET /api/{version}/system/version

Returns service version information in JSON format.

Request:

```
curl 'http://127.0.0.1:8390/api/v1/system/version'
```

Response:

```
{
    "service": "prediction-builder-api",
    "profile": "aero_staging",
    "version": "0.5.1",
    "time": 1597155326
}
```

Parameters

- version (string) – Rest API version.

Коды статуса

- 200 OK – Service version information in JSON format.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- profile (string) – Service configuration profile.
- service (string) – Service name.
- time (integer) – Service request time.
- version (string) – Service API version.

Глава 5

Feature Builder

POST /api/{version}/feature/build

Builds featureset and returns featureset status.

Пример запроса:

```
curl -X POST 'http://localhost:8280/api/v1/feature/build/?extractors=cycle_counter_feats&
↪from=1451606400&to=1454284800&buildOnly=true'
```

You can pass additional information about featureset in Json format as a body parameter. It can be received by user with requests /api/v1/feature/featuresets, /api/v1/feature/featuresets/{UID} and etc. as «metadata» parameter.

Пример ответа:

```
{
    "featuresetId": "1576577796277-7c978686-03fc-459c-b72c-2071b55cfd76",
    "buildOnly": true,
    "featuresetLink": "http://127.0.1.1:8280/api/v1/feature/getFeatureset?
↪uid=1576577796277-7c978686-03fc-459c-b72c-2071b55cfd76",
    "extractors": [
        {
            "completed": true,
            "config": "workdir/aero_s/extractors/cycle_counter_feats/config/default.json",
            "extractor": "cycle_counter_feats",
            "timeTaken": 1
        },
        {
            "completed": true,
            "config": "workdir/aero_s/extractors/cycle_counter_feats/config/config_feat_
↪ENG_1_REV_PRESSURIZED.json",
            "extractor": "cycle_counter_feats",
            "timeTaken": 1
        },
        {
            "completed": true,
            "config": "workdir/aero_s/extractors/cycle_counter_feats/config/config_feat_
↪ENG_2_REV_PRESSURIZED.json",
            "extractor": "cycle_counter_feats",
            "timeTaken": 1
        }
    ]
}
```

Параметры

- version (string) – Rest API version.

Параметры запроса

- extractors (array) – Строка с названиями экстракторов. Используется „,“ как разделитель.
- featuresetId (string) –
- from (integer) – Начало периода времени. Поддерживает Unix Timestamp формат (например 1514764800).
- to (integer) – Конец периода времени. Поддерживает Unix Timestamp формат (например 1514764800).
- persistent (boolean) – Флаг для удаления (boolean).
- buildOnly (boolean) – Build features without import.
- postProcess (boolean) – Using post processing for built features.
- metadata (string) – Metadata in JSON format.

Коды статуса

- 200 OK – Featureset build status which includes featuresetId and list of states of executed extractors.
- 201 Created – Создано
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- buildId (string) – Build UUID.
- buildLink (string) – Link get build.
- buildOnly (boolean) – Build featureset without import.
- extractors (array) – Список статусов работы построителей.
- extractors[].completed (boolean) – Работа экстрактора завершена.
- extractors[].config (string) – Путь до файла конфигурации.
- extractors[].extractor (string) – Название экстрактора.
- extractors[].timeTaken (integer) – Время, затраченное на работу построителя, в секундах.
- featuresetId (string) – UUID фичасета.
- featuresetLink (string) – Link get featureset.
- importDuration (integer) – Feature import duration.
- importFeatures (integer) – Count of imported features.
- importObjects (integer) – Count of imported objects.

- importSuccess (boolean) – Feature import status.

POST /api/{version}/feature/build/async

Строит признаки и возвращает ID фичасета.

Пример запроса:

```
curl -X POST 'http://localhost:8280/api/v1/feature/build/async/?extractors=cycle_counter_
↪feats&from=1451606400&to=1454284800'
```

You can pass additional information about featureset in Json format as a body parameter. It can be received by user with requests /api/v1/feature/featuresets, /api/v1/feature/featuresets/{UID} and etc. as «metadata» parameter.

Пример ответа:

1571986311509-1129c8f9-d80c-4557-a874-e0f8e935e0f1

Параметры

- version (string) – Rest API version.

Параметры запроса

- extractors (array) – Строка с названиями экстракторов. Используется „,“ как разделитель.
- featuresetId (string) –
- from (integer) – Начало периода времени. Поддерживает Unix Timestamp формат (например 1514764800).
- to (integer) – Конец периода времени. Поддерживает Unix Timestamp формат (например 1514764800).
- persistent (boolean) – Флаг для удаления (boolean).
- buildOnly (boolean) – Build features without import.
- postProcess (boolean) – Using post processing for built features.
- metadata (string) – Metadata in JSON format.

Коды статуса

- 200 OK – FeaturesetId.
- 201 Created – Создано
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

GET /api/{version}/feature/builds/{UID}/logs

Returns build logs for the specified featureset.

Пример запроса:

```
curl 'http://localhost:8280/api/v1/feature/featuresets/1570019673434-b62e7dc4-3e5d-4809-
↪935b-fbf95ab1fffb/logs'
```

Пример ответа:

```
{
    "featuresetId": "1570019673434-b62e7dc4-3e5d-4809-935b-fbf95ab1fffb",
    "buildLogs": {
        "workdir/aero_s/extractors/cycle_counter_feats/config/config_feat_ENG_2_REV_
↪PRESSURIZED.json": [
            "2019-10-02 15:34:34:INFO: Feature generation started",
            "2019-10-02 15:34:35:INFO: Feature generation done, generated 61494 records␣
↪with 7 features",
            "2019-10-02 15:34:35:INFO: Feature generation finished"
        ],
        "workdir/aero_s/extractors/cycle_counter_feats/config/default.json": [
            "2019-10-02 15:34:34:INFO: Feature generation started",
            "2019-10-02 15:34:35:INFO: Feature generation done, generated 61494 records␣
↪with 7 features",
            "2019-10-02 15:34:35:INFO: Feature generation finished"
        ],
        "workdir/aero_s/extractors/cycle_counter_feats/config/config_feat_ENG_1_REV_
↪PRESSURIZED.json": [
            "2019-10-02 15:34:34:INFO: Feature generation started",
            "2019-10-02 15:34:35:INFO: Feature generation done, generated 61494 records␣
↪with 7 features",
            "2019-10-02 15:34:35:INFO: Feature generation finished"
        ]
    }
}
```

Параметры

- version (string) – Rest API version.
- UID (string) – UUID фичасета (например, 1560938628840-6531d51f-64ba-47c8-b3d4-1847913cd6ed).

Коды статуса

- 200 OK – Featureset build log which includes featuresetId and map (dictionary) of logs divided by extractor.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- buildId (string) – Build UUID.
- buildLogs (object) – Словарь логов, разделенных по построителям.
- featuresetId (string) – UUID фичасета.

GET /api/{version}/feature/builds/{UID}/logs/extractor

Return build logs for the specified featureset and extractor.

Пример запроса:

```
curl 'http://localhost:8280/api/v1/feature/featuresets/1570019673434-b62e7dc4-3e5d-4809-
↪935b-fbf95ab1fffb/logs/extractor?config=workdir/aero_s/extractors/cycle_counter_feats/
↪config/config_feat_ENG_2_REV_PRESSURIZED.json'
```

Пример ответа:

2019-10-02 15:34:34:INFO: Feature generation startedn 2019-10-02 15:34:35:INFO: Feature generation done, generated 61494 records with 7 featuresn 2019-10-02 15:34:35:INFO: Feature generation finished

Параметры

- version (string) – Rest API version.
- UID (string) – UUID фичасета (например, 1560938628840-6531d51f-64ba-47c8-b3d4-1847913cd6ed).

Параметры запроса

- config (string) – Имя построителя.

Коды статуса

- 200 OK – Логи построителя.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

GET /api/{version}/feature/data/client/meta

** Запрос метаданных. **

Пример запроса:

```
curl 'http://localhost:8280/api/v1/feature/data/client/meta'
```

Пример ответа:

```
{
    "namespace": "aerophm",
    "datasets": [
        {
            "name": "flights",
            "fields": [
                "aircraft_id",
                "flight",
                "airport_from",
                "airport_to",
                "est_departure_time",
                "actual_departure_time",
                "est_arrival_time",
                "actual_arrival_time"
            ],
            "exclude": [],
            "special": {
                "from": "actual_arrival_time",
                "to": "actual_departure_time"
            },
            "alias": {},
            "order": "actual_departure_time",
            "limit": "",
            "format": "tsv",
            "header": false,
            "zip": false
        }
    ]
}
```

Параметры

- version (string) – Rest API version.

Коды статуса

- 200 OK – Метаданные запроса.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- datasets (array) – Список набора метаданных запросов.
- datasets[].alias (object) – Словарь, пары (ключ, значение) которого наименование таблицы и временное название таблицы соответственно.
- datasets[].exclude (array) – Список исключаемых полей таблицы.
- datasets[].fields (array) – Список полей таблицы.
- datasets[].format (string) – Result-set format (possible options: „tsv“, „tsv-view“, „json“, „json-view“). Default „tsv“.
- datasets[].header (boolean) – Добавить заголовки в файл. По умолчанию, true.
- datasets[].limit (integer) – Количество строк в результирующем наборе данных.
- datasets[].name (string) – Название таблицы.
- datasets[].order (string) – Сортировка выходных значений.
- datasets[].special (object) – Словарь условий, значения которого - наименования столбцов.
- datasets[].zip (boolean) – Флаг архивации zip. По умолчанию, „false“.
- namespace (string) – Название базы данных.

GET /api/{version}/feature/extractors

Список существующий экстракторов.

Request:

```
curl 'http://localhost:8280/api/v1/feature/extractors'
```

Пример ответа:

```
[
    "quantile_feats_90_94_97",
    "quantile_feats_75_85_95",
    "aircraft_indicator_feats",
    "quantile_feats_85_90_95",
    "past_errors_feats",
    "weather_feats",
    "months_feats",
    "unseasoned_quantile_feats_90_94_97",
    "cycle_counter_feats",
    "ngram_features_w_40_7_intervals_5",
    "exponential_feats",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "reduced_ngram_features_w_40_7_intervals_5"
]
```

Параметры

- version (string) – Rest API version.

Коды статуса

- 200 OK – List of extractors names.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

GET /api/{version}/feature/extractors/config

Return config for specified extractor.

Request:

```
curl 'localhost:8280//api/v1/feature/extractors/config/?extractor=cycle_counter_feats'
```

Пример ответа:

```
{
    "workdir/aero_s/extractors/cycle_counter_feats/config/config_feat_ENG_1_REV_
↪PRESSURIZED.json": {
        "feature_extractor_name": "cycle_counter_feats",
        "features_sources": [
            {
                "filename": "flights.tsv",
                "datatype": "flights",
                "request_all_time": true
            }
        ],
        "sources_minimum_time_lag_seconds": 604800,
        "input_folder": "./input",
        "output_folder": "./output",
        "for_model": "RENAMED_ENG_2_REV_PRESSURIZED_SINGLE_NO_QAR",
        "feature_extractor_settings": {
            "output\ *file": "feats*\ {FEATURE_EXTRACTOR_NAME}_ENG_1_REV_PRESSURIZED.csv",
            "messages": [
                "ENG 1 REV PRESSURIZED"
            ]
        },
        "base_class": {
            "messages_extractor": {
                "cfds_file": "events.tsv",
                "schedule_file": "flights.tsv"
            }
        }
    },
    "workdir/aero_s/extractors/cycle_counter_feats/config/default.json": {
        "feature_extractor_name": "cycle_counter_feats",
        "features_sources": [
            {
                "filename": "flights.tsv",
                "datatype": "flights",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                "request_all_time": true
            }
        ],
        "sources_minimum_time_lag_seconds": 604800,
        "input_folder": "./input",
        "output_folder": "./output",
        "feature_extractor_settings": {
            "output\ *file": "feats*\ {FEATURE_EXTRACTOR_NAME}.csv",
            "messages": [
                "ENG 1 FIRE LOOP A FAULT"
            ]
        },
        "base_class": {
            "messages_extractor": {
                "cfds_file": "events.tsv",
                "schedule_file": "flights.tsv"
            }
        }
    }
}
```

Параметры

- version (string) – Rest API version.

Параметры запроса

- extractor (string) – Name extractor.

Коды статуса

- 200 OK – Extractor import status which includes extractor name, extractor path and created time.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- extractor (string) – List of imported extractor names.
- extractorPath (string) – Path to extractor directory.
- time (integer) – Created time in seconds.

GET /api/{version}/feature/extractors/export

Exports extractor as a single ZIP file.

Request:

```
curl 'localhost:8280/api/v1/feature/extractors/export?extractor=aircraft_indicator_feats'
```

Пример ответа:

aircraft_indicator_feats.zip with aircraft_indicator_feats directory inside.

Параметры

- version (string) – Rest API version.

Параметры запроса

- extractor (string) – Name extractor.

Коды статуса

- 200 OK – Zip archive with extractor file.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

POST /api/{version}/feature/extractors/import

Import extractor from the provided ZIP file into the feature storage and creates a new extractor.

Request:

```
curl -X POST 'http://localhost:8280/api/v1/feature/extractors/import'\
  -F file=@./feature-builder-impl/src/test/resources/test/months_feats.zip
```

Пример ответа:

```
{
    "extractor": "months_feats",
    "extractorPath": "file:///home/user/IdeaProjects/platform-proto-java/workdir/aero_s/
↪extractors/months_feats/",
    "time": 1580130493
}
```

Параметры

- version (string) – Rest API version.

Параметры формы

- file (file) – File with extractor in zip format.

Коды статуса

- 200 OK – Extractor import status which includes extractor name, extractor path and created time.
- 201 Created – Создано
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- extractor (string) – List of imported extractor names.
- extractorPath (string) – Path to extractor directory.
- time (integer) – Created time in seconds.

PUT /api/{version}/feature/extractors/restore

Delete extractors and load extractors from dump.

Request:

```
curl -X PUT 'http://localhost:8280/api/v1/feature/extractors/restore'
```

Пример ответа:

true

Параметры

- version (string) – Rest API version.

Коды статуса

- 200 OK – OK
- 201 Created – Создано
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

DELETE /api/{version}/feature/extractors/{extractor}

Delete extractor.

Request:

```
curl -X DELETE 'http://localhost:8280/api/v1/feature/extractors/aircraft_indicator_feats'
```

Пример ответа:

true

Параметры

- version (string) – Rest API version.
- extractor (string) – Name extractor.

Коды статуса

- 200 OK – OK
- 204 No Content – Нет содержимого
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен

GET /api/{version}/feature/featuresets/{UID}/logs

Returns build logs for the specified featureset.

Пример запроса:

```
curl 'http://localhost:8280/api/v1/feature/featuresets/1570019673434-b62e7dc4-3e5d-4809-
↪935b-fbf95ab1fffb/logs'
```

Пример ответа:

```
{
    "featuresetId": "1570019673434-b62e7dc4-3e5d-4809-935b-fbf95ab1fffb",
    "buildLogs": {
        "workdir/aero_s/extractors/cycle_counter_feats/config/config_feat_ENG_2_REV_
↪PRESSURIZED.json": [
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            "2019-10-02 15:34:34:INFO: Feature generation started",
            "2019-10-02 15:34:35:INFO: Feature generation done, generated 61494 records␣
↪with 7 features",
            "2019-10-02 15:34:35:INFO: Feature generation finished"
        ],
        "workdir/aero_s/extractors/cycle_counter_feats/config/default.json": [
            "2019-10-02 15:34:34:INFO: Feature generation started",
            "2019-10-02 15:34:35:INFO: Feature generation done, generated 61494 records␣
↪with 7 features",
            "2019-10-02 15:34:35:INFO: Feature generation finished"
        ],
        "workdir/aero_s/extractors/cycle_counter_feats/config/config_feat_ENG_1_REV_
↪PRESSURIZED.json": [
            "2019-10-02 15:34:34:INFO: Feature generation started",
            "2019-10-02 15:34:35:INFO: Feature generation done, generated 61494 records␣
↪with 7 features",
            "2019-10-02 15:34:35:INFO: Feature generation finished"
        ]
    }
}
```

Параметры

- version (string) – Rest API version.
- UID (string) – UUID фичасета (например, 1560938628840-6531d51f-64ba-47c8-b3d4-1847913cd6ed).

Коды статуса

- 200 OK – Featureset build log which includes featuresetId and map (dictionary) of logs divided by extractor.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- buildId (string) – Build UUID.
- buildLogs (object) – Словарь логов, разделенных по построителям.
- featuresetId (string) – UUID фичасета.

GET /api/{version}/feature/featuresets/{UID}/logs/extractor

Return build logs for the specified featureset and extractor.

Пример запроса:

```
curl 'http://localhost:8280/api/v1/feature/featuresets/1570019673434-b62e7dc4-3e5d-4809-
↪935b-fbf95ab1fffb/logs/extractor?config=workdir/aero_s/extractors/cycle_counter_feats/
↪config/config_feat_ENG_2_REV_PRESSURIZED.json'
```

Пример ответа:

2019-10-02 15:34:34:INFO: Feature generation startedn 2019-10-02 15:34:35:INFO: Feature generation done, generated 61494 records with 7 featuresn 2019-10-02 15:34:35:INFO: Feature generation finished

Параметры

- version (string) – Rest API version.
- UID (string) – UUID фичасета (например, 1560938628840-6531d51f-64ba-47c8-b3d4-1847913cd6ed).

Параметры запроса

- config (string) – Имя построителя.

Коды статуса

- 200 OK – Логи построителя.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

GET /api/{version}/feature/getBuild

Get zip archive of built.

Пример запроса:

```
curl 'http://localhost:8280/api/v1/feature/getFeatureset?uid=1576163995972-5004a819-f176-
↪463b-a117-7a75f0a7796'
```

Пример ответа:

1576163995972-5004a819-f176-463b-a117-7a75f0a7796.zip with built features files in CSV format.

Параметры

- version (string) – Rest API version.

Параметры запроса

- UID (string) – UUID фичасета (например, 1560938628840-6531d51f-64ba-47c8-b3d4-1847913cd6ed).

Коды статуса

- 200 OK – Zip archive of featureset.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

GET /api/{version}/feature/getFeatureset

Get zip archive of built.

Пример запроса:

```
curl 'http://localhost:8280/api/v1/feature/getFeatureset?uid=1576163995972-5004a819-f176-
↪463b-a117-7a75f0a7796'
```

Пример ответа:

1576163995972-5004a819-f176-463b-a117-7a75f0a7796.zip with built features files in CSV format.

Параметры

- version (string) – Rest API version.

Параметры запроса

- UID (string) – UUID фичасета (например, 1560938628840-6531d51f-64ba-47c8-b3d4-1847913cd6ed).

Коды статуса

- 200 OK – Zip archive of featureset.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

GET /api/{version}/feature/status/{UID}

Возвращение логов заданного фичасета.

Пример запроса:

```
curl 'http://localhost:8280/api/v1/feature/status/1576587373685-764c7203-6e92-4242-a885-
↪e5dcd22521d8'
```

Пример ответа:

```
{
    "featuresetId": "1576587373685-764c7203-6e92-4242-a885-e5dcd22521d8",
    "buildOnly": true,
    "featuresetLink": "http://127.0.1.1:8280/api/v1/feature/getBuild?uid=1576587373685-
↪764c7203-6e92-4242-a885-e5dcd22521d8",
    "extractors": [
        {
            "completed": true,
            "config": "workdir/aero_s/extractors/cycle_counter_feats/config/default.json",
            "extractor": "cycle_counter_feats",
            "timeTaken": 1
        },
        {
            "completed": true,
            "config": "workdir/aero_s/extractors/cycle_counter_feats/config/config_feat_
↪ENG_1_REV_PRESSURIZED.json",
            "extractor": "cycle_counter_feats",
            "timeTaken": 1
        },
        {
            "completed": true,
            "config": "workdir/aero_s/extractors/cycle_counter_feats/config/config_feat_
↪ENG_2_REV_PRESSURIZED.json",
            "extractor": "cycle_counter_feats",
            "timeTaken": 1
        }
    ]
}
```

Параметры

- version (string) – Rest API version.
- UID (string) – UUID фичасета (например, 1560938628840-6531d51f-64ba-47c8-b3d4-1847913cd6ed).

Коды статуса

- 200 OK – Featureset build status which includes featuresetId and list of states of executed extractors.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- buildId (string) – Build UUID.
- buildLink (string) – Link get build.
- buildOnly (boolean) – Build featureset without import.
- extractors (array) – Список статусов работы построителей.
- extractors[].completed (boolean) – Работа экстрактора завершена.
- extractors[].config (string) – Путь до файла конфигурации.
- extractors[].extractor (string) – Название экстрактора.
- extractors[].timeTaken (integer) – Время, затраченное на работу построителя, в секундах.
- featuresetId (string) – UUID фичасета.
- featuresetLink (string) – Link get featureset.
- importDuration (integer) – Feature import duration.
- importFeatures (integer) – Count of imported features.
- importObjects (integer) – Count of imported objects.
- importSuccess (boolean) – Feature import status.

GET /api/{version}/system/status

Returns service status information in JSON format.

Пример запроса:

```
curl 'http://127.0.0.1:8280/api/v1/system/status'
```

Пример ответа:

```
{
    "service": "feature-builder-api",
    "status": "OK",
    "time": 1597155326
}
```

Параметры

- version (string) – Rest API version.

Коды статуса

- 200 OK – Service status information in JSON format.

- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- service (string) – Service name.
- status (string) – Service status.
- time (integer) – Service request time.

GET /api/{version}/system/version

Returns service version information in JSON format.

Пример запроса:

```
curl 'http://127.0.0.1:8280/api/v1/system/version'
```

Пример ответа:

```
{
    "service": "feature-builder-api",
    "profile": "aero_staging",
    "version": "0.5.1",
    "time": 1597155326
}
```

Параметры

- version (string) – Rest API version.

Коды статуса

- 200 OK – Service version information in JSON format.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- profile (string) – Service configuration profile.
- service (string) – Service name.
- time (integer) – Service request time.
- version (string) – Service API version.

Глава 6

Data Service

POST /api/{version}/data/batch

Query url which return result-sets in your format in a single ZIP file.

Пример запроса:

```
curl --location --request POST '127.0.0.1:8170/api/v1/data/batch' \
--header 'Content-Type: application/json' \
--data-raw '[
  {
    "namespace" : "aerophm",
    "dataset" : "aircrafts",
    "filename" : "AIRCRAFTS.csv",
    "fields" : "*",
    "alias" : "aircrafts.id:ida,flights.aircraft_id:idf",
    "filter" : "aircrafts.is_active==true",
    "format" : "csv-view"
  },
  {
    "namespace" : "aerophm",
    "dataset" : "flights",
    "filename" : "FLIGHTS_01012016_01022016.csv",
    "fields" : "*",
    "filter" : "actual_departure_time=gt=1451606400;actual_arrival_time=lt=1454284800",
    "order" : "actual_departure_time,actual_arrival_time",
    "format" : "csv-view"
  },
  {
    "namespace" : "aerophm",
    "dataset" : "airports",
    "filename" : "AIRPORTS.csv",
    "fields" : "id, name, iata as code",
    "order" : "code",
    "format" : "csv-view"
  },
  {
    "namespace" : "aerophm",
    "dataset" : "aircrafts",
    "filename" : "AIRCRAFTS_AIRPORTS_VISITS_01012016_01022016.csv",
    "fields" : "aircrafts.id,aircrafts.model,airports.iata,count(airports.iata) as␣
↪airport_visits_count",
    "alias": "aircrafts.id:aircraft_id,airports.iata:airport",
    "join": "inner:flights:flights.aircraft_id:aircrafts.id,left:airports:airports.iata:
↪flights.airport_to;flights.airport_from",
    "filter": "flights.actual_departure_time=gt=1451606400;flights.actual_arrival_
↪time=lt=1454284800",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "group": "aircrafts.id,aircrafts.model,airports.iata",
    "having": "count(airports.iata)=gt=5",
    "order": "aircraft_id, airport_visits_count desc",
    "format" : "csv-view"
  }
]'
```

Используйте rsql-parser, операторы сравнения которого имеют синтаксис (например для параметра filter):

- Равно : ==
- Не равно : !=
- Меньше : =lt= или <
- Меньше или равно : =le= или <=
- Больше : =gt= или >
- Больше или равно : =ge= или >=
- Оператор IN : =in=
- Оператор NOT IN : =out=
- Logical AND: ;
- Logical OR: ,

Response: batch_cfbb63340cfdb76ee4d4a26401b298a7.zip with 4 files inside.

Параметры

- version (string) – Rest API version.

JSON-объект запроса

- [].alias (string) – Имя таблицы и временное название таблицы, разделенные двоеточием (например „name:temp“).
- [].dataset (string) – Название таблицы.
- [].exclude (string) – Колонка, которая будет исключена.
- [].fields (string) – Строка с названиями полей. Использует „,“ как разделитель. По умолчанию выводит все поля.
- [].filename (string) – Name creating file.
- [].filter (string) – Выражение для фильтрации данных по параметру (например, парам>знач).
- [].format (string) – Result-set format (possible options: „csv“, „csv-view“, „tsv“, „tsv-view“, „json“, „json-view“). Default „csv“.
- [].group (string) – String with fields for to group the result-set by columns. Uses „,“ as a delimiter.
- [].having (string) – Expression to filter data by parameter, which can be used with aggregate functions. (e.g. param>val).
- [].header (string) – Добавляет заголовок в выходной файл. По умолчанию „true“.

- [].join (string) – List of datasets and columns, which are joined, where type join, dataset name and names columns (which compare) separated by colon (e.g. „inner:datasetA:datasetA.columnA:datasetB.columnB<;datasetB.columnC>“). Uses „,“ as a delimiter.
- [].limit (string) – Количество строк в результирующем наборе.
- [].namespace (string) – Название базы данных.
- [].offset (string) – Rows to skip before starting to count rows for result-set.
- [].order (string) – Сорти результат в порядке возрастания или убывания.
- [].zip (string) – Флаг архивации zip. По умолчанию, „false“.

Статус-коды

- 200 OK – OK
- 201 Created – Создано
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

POST /api/{version}/data/catalog/describe

Возвращает метаданные.

Пример запроса:

```
curl -X POST 'localhost:8170/api/v1/data/catalog/describe?namespace=aerophm&dataset=test&
↪forceVarchar=false&format=csv' \
--form 'file=@./data-service-impl/src/test/resources/catalog/describe/test_describe.csv'
```

Пример ответа:

```
{
    "namespace": "aerophm",
    "dataset": "test",
    "fields": [
        {
            "name": "t3_2",
            "type": "integer"
        },
        {
            "name": "ap_1",
            "type": "varchar(2)"
        },
        {
            "name": "pha_sc_1",
            "type": "real"
        },
        {
            "name": "id",
            "type": "integer"
        },
        {
            "name": "report_type",
            "type": "double"
        },
        {
            "name": "ssel_1",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            "type": "bigint"
        }
    ]
}
```

Параметры

- version (string) – Rest API version.

Параметры запроса

- namespace (string) – Название базы данных.
- dataset (string) – Название таблицы.
- file (file) – Data file in CSV or TSV format.
- forceVarchar (boolean) – Flag to set all columns to „varchar“ type. Default „false“.
- format (string) – Format inserted files (possible options: „tsv“, „csv“). Default „csv“.

Статус-коды

- 200 OK – OK
- 201 Created – Создано
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- dataset (string) – Dataset name.
- fields (array) – Список полей таблицы.
- fields[].name (string) – Название столбца.
- fields[].type (string) – Тип столбца.
- namespace (string) – Название базы данных.

GET /api/{version}/data/catalog/namespaces

Return a list of namespaces.

Пример запроса:

```
curl 'http://localhost:8170/api/v1/data/catalog/namespaces'
```

Пример ответа:

```
[
    "aerophm",
    "aerophm_import"
]
```

Параметры

- version (string) – Rest API version.

Статус-коды

- 200 OK – List of namespaces.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

GET /api/{version}/data/catalog/types

Return a list of types.

Пример запроса:

```
curl 'http://localhost:8170/api/v1/data/catalog/types'
```

Пример ответа:

```
[
    "BOOLEAN",
    "TINYINT",
    "SMALLINT",
    "INTEGER",
    "BIGINT",
    "REAL",
    "DOUBLE",
    "DECIMAL",
    "VARCHAR",
    "CHAR",
    "VARBINARY",
    "JSON",
    "DATE",
    "TIME",
    "TIME WITH TIME ZONE",
    "TIMESTAMP",
    "TIMESTAMP WITH TIME ZONE",
    "INTERVAL YEAR TO MONTH",
    "INTERVAL DAY TO SECOND",
    "ARRAY",
    "IPADDRESS",
    "IPPREFIX"
]
```

Параметры

- version (string) – Rest API version.

Статус-коды

- 200 OK – List of types.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

GET /api/{version}/data/catalog/{namespace}/meta

Возвращает список наборов данных из пространства имен.

Пример запроса:

```
curl 'http://127.0.0.1:8170/api/v1/data/catalog/aero_s/meta'
```

Пример ответа:

```
{
    "name": "aero_s",
    "datasets": [
        "aids_reports_01",
        "aids_reports_02",
        "aircrafts",
        "airports"
    ]
}
```

Параметры

- version (string) – Rest API version.
- namespace (string) – Название базы данных.

Статус-коды

- 200 OK – OK
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- datasets (array) – Названия таблиц
- name (string) – Название базы данных

GET /api/{version}/data/catalog/{namespace}/{dataset}/fields

Return a list of dataset fields.

Пример запроса:

```
curl 'http://localhost:8170/api/v1/data/catalog/aerophm/aircrafts/fields'
```

Пример ответа:

```
[
    "id",
    "model",
    "is_active"
]
```

Параметры

- version (string) – Rest API version.
- namespace (string) – Название базы данных.
- dataset (string) – Название таблицы.

Статус-коды

- 200 OK – OK
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

GET /api/{version}/data/catalog/{namespace}/{dataset}/meta

Возвращает метаданные.

Пример запроса:

```
curl 'http://127.0.0.1:8170/api/v1/data/catalog/aero_s/aids_reports_01/meta'
```

Пример ответа:

```
{
    "namespace": "aero_s",
    "name": "aids_reports_01",
    "fields": [
        {
            "name": "id",
            "type": "integer"
        },
        {
            "name": "report_type",
            "type": "integer"
        },
        {
            "name": "report_time_string",
            "type": "text"
        },
        {
            "name": "report_time",
            "type": "bigint"
        },
        {
            "name": "aircraft_id",
            "type": "text"
        }
    ]
}
```

Параметры

- version (string) – Rest API version.
- namespace (string) – Название базы данных.
- dataset (string) – Название таблицы.

Статус-коды

- 200 OK – OK
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- dataset (string) – Dataset name.
- fields (array) – Список полей таблицы.
- fields[].name (string) – Название столбца.
- fields[].type (string) – Тип столбца.
- namespace (string) – Название базы данных.

POST /api/{version}/data/import/{namespace}/{dataset}

Return data import status.

Пример запроса:

```
curl -X POST 'localhost:8170/api/v1/data/import/aerophm_import/test_import?header=false&
↪forceVarchar=false&format=csv' -H 'content-type: multipart/form-data' -F config=@./data-
↪service-impl/src/test/resources/import/datasets/test_import.json -F files=@./data-
↪service-impl/src/test/resources/import/dataCsv/file1.csv
```

Пример ответа:

```
{
    "namespace": "aerophm_import",
    "dataset": "test_import",
    "inserted": 2
}
```

Параметры

- version (string) – Rest API version.
- namespace (string) – Название базы данных.
- dataset (string) – Название таблицы.

Параметры запроса

- files (array) – List of data files in CSV format.
- config (file) – Metadata file in JSON format.
- forceVarchar (boolean) – Flag to set all columns to „varchar“ type. Default „false“.
- header (boolean) – There are headers in files. If header isn't passed compare first row in files with column names, if they match, then there are headers in files, else there aren't.
- format (string) – Format inserted files (possible options: „tsv“, „csv“). Default „csv“.

Статус-коды

- 200 OK – OK
- 201 Created – Создано
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- dataset (string) – Название таблицы.

- inserted (integer) – Row count were inserted.
- namespace (string) – Название базы данных.

GET /api/{version}/data/query/{namespace}/{dataset}

URL - запрос, который возвращает результат в выбранном формате.

Пример запроса:

```
curl --location --request GET '127.0.0.1:8170/api/v1/data/query/aerophm/cfds_events
  ?fields=count(message)%20as%20msg_count,%20aircraft_id,%20message%20as%20msg
  &group=aircraft_id,%20msg
  &filter=event_time=gt=1451606400;event_time=lt=1454284800
  &having=count(message)=ge=10
  &format=json-view
  &order=cfds_events.aircraft_id,%20msg_count%20desc'
```

Use rsql-parser, which comparison operators syntax (e.g. for parameter filter and having):

- Logical AND: ;
- Logical OR: ,
- Равно : ==
- Не равно : !=
- Меньше : =lt= или <
- Меньше или равно : =le= или <=
- Больше : =gt= или >
- Больше или равно : =ge= или >=
- Оператор IN : =in=
- Оператор NOT IN : =out=

Пример ответа:

Ключи JSON определяются параметрами запроса.

```
[
  {
      "msg_count": 80,
      "aircraft_id": "VP-BHF",
      "msg": "CHECK HF-1 ANTENNA       CIRCUIT"
  },
  {
      "msg_count": 28,
      "aircraft_id": "VP-BHF",
      "msg": "CHECK FDU APU LOOP A WARN CKT"
  },
  {
      "msg_count": 26,
      "aircraft_id": "VP-BHF",
      "msg": "AIR BLEED"
  },
  {
      "msg_count": 24,
      "aircraft_id": "VP-BHF",
      "msg": "L WING LOOP A INOP"
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  },
  {
      "msg_count": 12,
      "aircraft_id": "VP-BHF",
      "msg": "CHECK FDU APU LOOP B WARN CKT"
  },
  {
      "msg_count": 11,
      "aircraft_id": "VP-BHF",
      "msg": "PRINTER(4TW)/ATSU1(1TX1)"
  }
]
```

Параметры

- version (string) – Rest API version.
- namespace (string) – Название базы данных.
- dataset (string) – Название таблицы.

Параметры запроса

- fields (string) – Строка с названиями полей. Использует „,“ как разделитель. По умолчанию выводит все поля.
- alias (string) – Имя таблицы и временное название таблицы, разделенные двоеточием (например „name:temp“).
- exclude (string) – Колонка, которая будет исключена.
- join (string) – List of datasets and columns, which are joined, where type join, dataset name and names columns (which compare) separated by colon (e.g. „inner:datasetA:datasetA.columnA:datasetB.columnB<;datasetB.columnC>“). Uses „,“ as a delimiter.
- filter (string) – Выражение для фильтрации данных по параметру (например, парам>знач).
- order (string) – Сорти результат в порядке возрастания или убывания.
- group (string) – String with fields for to group the result-set by columns. Uses „,“ as a delimiter.
- having (string) – Expression to filter data by parameter, which can be used with aggregate functions. (e.g. param>val).
- limit (string) – Количество строк в результирующем наборе.
- offset (string) – Rows to skip before starting to count rows for result-set.
- header (string) – Добавляет заголовок в выходной файл. По умолчанию „true“.
- format (string) – Result-set format (possible options: „csv“, „csv-view“, „tsv“, „tsv-view“, „json“, „json-view“). Default „csv“.
- zip (string) – Флаг архивации zip. По умолчанию, „false“.

Статус-коды

- 200 OK – OK
- 401 Unauthorized – Авторизация не выполнена

- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

POST /api/{version}/data/query/{namespace}/{dataset}

URL - запрос, который возвращает результат в выбранном формате.

Пример запроса:

```
curl --location --request POST '127.0.0.1:8170/api/v1/data/query/aerophm/aircrafts' \
--header 'Content-Type: application/json' \
--data-raw '{
    "fields" : "aircrafts.id,aircrafts.model,airports.iata,count(airports.iata) as␣
↪airport_visits_count",
    "alias": "aircrafts.id:aircraft_id,airports.iata:airport",
    "join": "inner:flights:flights.aircraft_id:aircrafts.id,left:airports:airports.iata:
↪flights.airport_to;flights.airport_from",
    "filter": "flights.actual_departure_time=gt=1451606400;flights.actual_arrival_
↪time=lt=1454284800",
    "group": "aircrafts.id,aircrafts.model,airports.iata",
    "having": "count(airports.iata)=gt=5",
    "order": "aircraft_id, airport_visits_count desc",
    "format" : "json-view"
  }'
```

Используйте rsql-parser, операторы сравнения которого имеют синтаксис (например для параметра filter):

- Logical AND: ;
- Logical OR: ,
- Равно : ==
- Не равно : !=
- Меньше : =lt= или <
- Меньше или равно : =le= или <=
- Больше : =gt= или >
- Больше или равно : =ge= или >=
- Оператор IN : =in=
- Оператор NOT IN : =out=

Пример ответа:

Ключи JSON определяются параметрами запроса.

```
[
  {
      "aircraft_id": "VP-BHF",
      "model": "A319",
      "airport": "DME",
      "airport_visits_count": 159
  },
  {
      "aircraft_id": "VP-BHF",
      "model": "A319",
      "airport": "LED",
```

(continues on next page)

(продолжение с предыдущей страницы)


```
    "airport_visits_count": 17
},
{
    "aircraft_id": "VP-BHF",
    "model": "A319",
    "airport": "VOZ",
    "airport_visits_count": 12
},
{
    "aircraft_id": "VP-BHF",
    "model": "A319",
    "airport": "ROV",
    "airport_visits_count": 10
},
{
    "aircraft_id": "VP-BHF",
    "model": "A319",
    "airport": "NBC",
    "airport_visits_count": 10
},
{
    "aircraft_id": "VP-BHF",
    "model": "A319",
    "airport": "PEE",
    "airport_visits_count": 10
},
{
    "aircraft_id": "VP-BHF",
    "model": "A319",
    "airport": "SVX",
    "airport_visits_count": 10
},
{
    "aircraft_id": "VP-BHF",
    "model": "A319",
    "airport": "VOG",
    "airport_visits_count": 8
},
{
    "aircraft_id": "VP-BHF",
    "model": "A319",
    "airport": "KUF",
    "airport_visits_count": 8
},
{
    "aircraft_id": "VP-BHF",
    "model": "A319",
    "airport": "UFA",
    "airport_visits_count": 6
},
{
    "aircraft_id": "VP-BHF",
    "model": "A319",
    "airport": "KZN",
    "airport_visits_count": 6
},
{
    "aircraft_id": "VP-BHF",
    "model": "A319",
    "airport": "AAQ",
    "airport_visits_count": 6
```


(continues on next page)

(продолжение с предыдущей страницы)

```
  }
]
```

Параметры

- version (string) – Rest API version.
- namespace (string) – Название базы данных.
- dataset (string) – Название таблицы.

JSON-объект запроса

- alias (string) – Имя таблицы и временное название таблицы, разделенные двоеточием (например „name:temp“).
- exclude (string) – Колонка, которая будет исключена.
- fields (string) – Строка с названиями полей. Использует „,“ как разделитель. По умолчанию выводит все поля.
- filter (string) – Выражение для фильтрации данных по параметру (например, парам>знач).
- format (string) – Result-set format (possible options: „csv“, „csv-view“, „tsv“, „tsv-view“, „json“, „json-view“). Default „csv“.
- group (string) – String with fields for to group the result-set by columns. Uses „,“ as a delimiter.
- having (string) – Expression to filter data by parameter, which can be used with aggregate functions. (e.g. param>val).
- header (string) – Добавляет заголовок в выходной файл. По умолчанию „true“.
- join (string) – List of datasets and columns, which are joined, where type join, dataset name and names columns (which compare) separated by colon (e.g. „inner:datasetA:datasetA.columnA:datasetB.columnB<;datasetB.columnC>“). Uses „,“ as a delimiter.
- limit (string) – Количество строк в результирующем наборе.
- offset (string) – Rows to skip before starting to count rows for result-set.
- order (string) – Сорти результат в порядке возрастания или убывания.
- zip (string) – Флаг архивации zip. По умолчанию, „false“.

Статус-коды

- 200 OK – OK
- 201 Created – Создано
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

GET /api/{version}/data/query/{namespace}/{dataset}/trace

Url which return generated query.

Пример запроса:

```
curl --location --request GET '127.0.0.1:8170/api/v1/data/query/aerophm/cfds_events/trace
  ?fields=count(message)%20as%20msg_count,%20aircraft_id,%20message%20as%20msg
  &group=aircraft_id,%20msg
  &filter=event_time=gt=1451606400;event_time=lt=1454284800
  &having=count(message)=ge=10
  &format=tsv-view
  &order=cfds_events.aircraft_id,%20msg_count%20desc'
```

Use rsql-parser, which comparison operators syntax (e.g. for parameter filter and having):

- Logical AND: ;
- Logical OR: ,
- Равно : ==
- Не равно : !=
- Меньше : =lt= или <
- Меньше или равно : =le= или <=
- Больше : =gt= или >
- Больше или равно : =ge= или >=
- Оператор IN : =in=
- Оператор NOT IN : =out=

Пример ответа:

Ключи JSON определяются параметрами запроса.

```
{
  "version": "v1",
  "namespace": "aerophm",
  "dataset": "cfds_events",
  "request": {
    "fields": "count(message) as msg_count, aircraft_id, message as msg",
    "alias": "",
    "exclude": "",
    "join": "",
    "filter": "event_time=gt=1451606400;event_time=lt=1454284800",
    "order": "cfds_events.aircraft_id, msg_count desc",
    "group": "aircraft_id, msg",
    "having": "count(message)=ge=10",
    "limit": "",
    "offset": "",
    "header": "true",
    "format": "tsv-view",
    "zip": "false"
  },
  "query": "SELECT count(message) AS msg_count,aircraft_id,message AS msg FROM cfds_
↪events WHERE event_time>1451606400  AND event_time<1454284800  GROUP BY aircraft_id,msg␣
↪HAVING COUNT(message)>=10  ORDER BY cfds_events.aircraft_id, msg_count desc"
}
```

Параметры

- version (string) – Rest API version.

- namespace (string) – Название базы данных.
- dataset (string) – Название таблицы.

Параметры запроса

- fields (string) – Строка с названиями полей. Использует „,“ как разделитель. По умолчанию выводит все поля.
- alias (string) – Имя таблицы и временное название таблицы, разделенные двоеточием (например „name:temp“).
- exclude (string) – Колонка, которая будет исключена.
- join (string) – List of datasets and columns, which are joined, where type join, dataset name and names columns (which compare) separated by colon (e.g. „inner:datasetA:datasetA.columnA:datasetB.columnB<;datasetB.columnC>“). Uses „,“ as a delimiter.
- filter (string) – Выражение для фильтрации данных по параметру (например, парам>знач).
- order (string) – Сорти результат в порядке возрастания или убывания.
- group (string) – String with fields for to group the result-set by columns. Uses „,“ as a delimiter.
- having (string) – Expression to filter data by parameter, which can be used with aggregate functions. (e.g. param>val).
- limit (string) – Количество строк в результирующем наборе.
- offset (string) – Rows to skip before starting to count rows for result-set.
- header (string) – Добавляет заголовок в выходной файл. По умолчанию „true“.
- format (string) – Result-set format (possible options: „csv“, „csv-view“, „tsv“, „tsv-view“, „json“, „json-view“). Default „csv“.
- zip (string) – Флаг архивации zip. По умолчанию, „false“.

Статус-коды

- 200 OK – OK
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- dataset (string) – Название таблицы.
- namespace (string) – Название базы данных.
- query (string) – Generated query.
- request (object) –
- request.alias (string) – Имя таблицы и временное название таблицы, разделенные двоеточием (например „name:temp“).
- request.exclude (string) – Колонка, которая будет исключена.

- request.fields (string) – Строка с названиями полей. Использует „,“ как разделитель. По умолчанию выводит все поля.
- request.filter (string) – Выражение для фильтрации данных по параметру (например, парам>знач).
- request.format (string) – Result-set format (possible options: „csv“, „csv-view“, „tsv“, „tsv-view“, „json“, „json-view“). Default „csv“.
- request.group (string) – String with fields for to group the result-set by columns. Uses „,“ as a delimiter.
- request.having (string) – Expression to filter data by parameter, which can be used with aggregate functions. (e.g. param>val).
- request.header (string) – Добавляет заголовок в выходной файл. По умолчанию „true“.
- request.join (string) – List of datasets and columns, which are joined, where type join, dataset name and names columns (which compare) separated by colon (e.g. „inner:datasetA:datasetA.columnA:datasetB.columnB<;datasetB.columnC>“). Uses „,“ as a delimiter.
- request.limit (string) – Количество строк в результирующем наборе.
- request.offset (string) – Rows to skip before starting to count rows for result-set.
- request.order (string) – Сорти результат в порядке возрастания или убывания.
- request.zip (string) – Флаг архивации zip. По умолчанию, „false“.
- version (string) – Rest API version.

GET /api/{version}/system/status

Returns service status information in JSON format.

Пример запроса:

```
curl 'http://127.0.0.1:8170/api/v1/system/status'
```

Пример ответа:

```
{
    "service": "data-service-api",
    "status": "OK",
    "time": 1597155326
}
```

Параметры

- version (string) – Rest API version.

Статус-коды

- 200 OK – Service status information in JSON format.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- service (string) – Service name.
- status (string) – Service status.
- time (integer) – Service request time.

GET /api/{version}/system/version

Returns service version information in JSON format.

Пример запроса:

```
curl 'http://127.0.0.1:8170/api/v1/system/version'
```

Пример ответа:

```
{
    "service": "data-service-api",
    "profile": "aero_staging",
    "version": "0.5.1",
    "time": 1597155326
}
```

Параметры

- version (string) – Rest API version.

Статус-коды

- 200 OK – Service version information in JSON format.
- 401 Unauthorized – Авторизация не выполнена
- 403 Forbidden – Доступ запрещен
- 404 Not Found – Не найдено

Возвращаемый объект JSON

- profile (string) – Service configuration profile.
- service (string) – Service name.
- time (integer) – Service request time.
- version (string) – Service API version.

Глава 7

Model Wrapper

Этот класс используется как родительский когда выполняется упаковка модели для запуска в боевой среде через Models Player.

class model_abstract_wrapper.ModelAbstractWrapper (*config={}*)

Этот класс используется как родительский когда выполняется упаковка модели для запуска в боевой среде через Models Player.

См пример упаковки модели на странице установки и конфигурации model_wrapper.

model_init (*model_parameters={}*)

Метод вызывается при запуске модели с аргументом в виде словаря, содержащего параметры модели.

Параметры model_parameters – словарь с параметрами модели, по умолчанию словарь пуст.

predict (*X*)

Метод может быть вызван из внешней среды через rpc.

Параметры X –

Результат current_prediction: должен иметь тип float, результат работы модели на векторе X

Тип результата list[float]

shutdown ()

Метод должен быть вызван при остановке модели.

Глава 8

Evaluation Tools

8.1 Инструменты для поддержки процесса анализа данных

8.1.1 evaluation_tools.feature_loader

Содержит класс FeatureLoader, исключения и именованные кортежи для загрузки признаков из локальных файлов или по сети.

FeatureExtractorParameters(files, . . .)	Класс для создания именованных кортежей, содержащий параметры построителей признаков, работающих с локальными данными.
FeatureLoader([config_name, mongo_config, . . .])	Класс используется для загрузки признаков из хранилища признаков.
NetworkFeatureParameters	Класс для создания именованных кортежей, содержащий параметры построителей признаков, работающих с данными по сети.

evaluation_tools.feature_loader.FeatureExtractorParameters

class evaluation_tools.feature_loader.FeatureExtractorParameters (*files*, *index_columns*, *dtypes*, *default_dtype*, *sep*)

Класс для создания именованных кортежей, содержащий параметры построителей признаков, работающих с локальными данными.

Методы и атрибуты

default_dtype	Псевдоним для поля №3
dtypes	Псевдоним для поля №2
files	Псевдоним для поля №0
index_columns	Псевдоним для поля №1
sep	Псевдоним для поля №4

default_dtype

Псевдоним для поля №3

dtypes

Псевдоним для поля №2

files

Псевдоним для поля №0

index_columns
Псевдоним для поля №1

sep
Псевдоним для поля №4

evaluation_tools.feature_loader.FeatureLoader

class evaluation_tools.feature_loader.FeatureLoader(*config_name=None*, *mongo_config=None*, *tool_config=None*, *logger=None*, *verbose=False*, *config_not_found_error=True*, *use_config_from_mongo_directly=False*, ***kwargs*)

Класс используется для загрузки признаков из хранилища признаков.

Can get data from local files (.parquet.gz or with pd.read_csv from .csv, .tsv, etc.) and by the REST API using local and network workers. In case of loading features by network uses Feature Builder service.

Параметры

- config_name (str) – Name of tool configuration.
- mongo_config (dict, optional) – Configuration for database with parameters: host, port, db. In case of None, connection settings will be obtained from ProjectContextManager. (Default value = None)
- tool_config (dict, optional) – is being used as config of tool is specified instead of getting via config_db_connector.ConfigConnector. (Default value = None)
- logger (logging.Logger, optional) – Logger object from logging module. (Default value = None)
- verbose (bool, optional) – Controls the verbosity when requesting features. (Default value = False)
- config_not_found_error (bool, optional) – This flag blocks raising error when tool config not found. This is using for tests and tools without config. (Default value = True)
- external_service_config (dict, optional) – dict with configuration of the REST API of the Service which the data is received from (for ex. Feature Store). Configuration must be written in form: { "host" : "127.0.0.1", "port" : 8590, "ssh_tunnel" : {"ssh_host" : "some.host.ip", "ssh_port" : 8080 }}. This parameter is transmitted via kwargs. (Default value = None) tmp_dir(str, optional): The full path to the directory for storing temporary files. This parameter is transmitted via kwargs. (Default value = „/tmp/dataskai“)

Примеры

```
>>> fl = FeatureLoader(config_name="aero__fw_classification_v1_features")
>>> configs = fl.list_configs()
>>> fl.load_features(records_to_use=['last_val__aids_rep_01'])
```

Methods

get_config_storage_name()	Overrided parent method
list_configs()	Загружает из хранилища конфигурации объекта FeatureLoader и возвращает их в виде списка.
load_features([records_to_use, feature_selector])	Метод возвращает признаки, заданные для объекта FeatureLoader.

continues on next page

Таблица 8.3 – продолжение с предыдущей страницы

load_features_dtypes([records_to_use, . . .])	Метод возвращает типы данных признаков, заданных для объекта FeatureLoader.
remove_cache()	Removes probable cache files.

get_config_storage_name ()
Overrided parent method

Результат Name of the tool's config storage name. In this case - mongo collection name.

Тип результата str

list_configs ()
Загружает из хранилища конфигурации объекта FeatureLoader и возвращает их в виде списка.

Один объект может ссылаться на несколько наборов признаков, идентифицируемых по параметру config_name, который определяется при создании объекта FeatureLoader.

Результат Список конфигураций. Конфигурации могут использоваться для загрузки признаков.

Тип результата list[dict]

Raises

- ConfigNameNotFoundError – Feature configuration with required name isn't found in configuration storage.
- ConfigNameNotUniqueError – Feature configuration with required name is not unique in storage.

load_features (*records_to_use=None*, *feature_selector=<function FeatureLoader.<lambda>>*)
Метод возвращает признаки, заданные для объекта FeatureLoader.

Возвращает наборы признаков, соответствующие параметру records_to_use, который содержит имя набора (см. self.list_records, параметр name в списке feature_records) или их список. Если records_to_use имеет значение None, метод выполнит загрузку всех доступных наборов признаков. В случае если метод загружает несколько объектов pd.DataFrame, они объединяются в один по значениям индексов. Дублирующие столбцы удаляются.

Параметр feature_selector - функция, которая используется для определения релевантных признаков.

Параметры

- records_to_use (str or list[str], optional) – Имя набора признаков для загрузки или список таких имен. Загружаются только релевантные признаки. (Значение по умолчанию = None)
- feature_selector (function(str), optional) – Функция „column_name -> bool", которая используется для отбора релевантных признаков. Если возвращаемое значение feature_selector(feature_column) == True, это означает, что признак необходимо включить в выборку. Если возвращаемое значение feature_selector(feature_column) == False, признак в выборку не включается. (Значение по умолчанию = lambda c: True)

Результат Релевантные признаки.

Тип результата pandas.DataFrame

load_features_dtypes (*records_to_use=None*, *feature_selector=<function FeatureLoader.<lambda>>*)
Метод возвращает типы данных признаков, заданных для объекта FeatureLoader.

Возвращает типы данных соответственно параметру records_to_use, который содержит имя набора признаков (см self.list_records, параметр name списка feature_records) или список наборов. Если records_to_use имеет значение None, метод выполнит загрузку всех доступных наборов признаков. Повторения типов данных удаляются.

Параметр feature_selector - функция, которая используется для определения релевантных признаков.

Параметры

- records_to_use (str or list[str], optional) – Имя набора признаков для загрузки или список таких имен. Загружаются только релевантные признаки. (Значение по умолчанию = None)
- feature_selector (function(str), optional) – Функция „column_name -> bool“, которая используется для отбора релевантных признаков. Если возвращаемое значение feature_selector(feature_column) == True, это означает, что признак необходимо включить в выборку. Если возвращаемое значение feature_selector(feature_column) == False, признак в выборку не включается. (Значение по умолчанию = lambda c: True)

Результат Признаки с указанием типов данных.

Тип результата dict

remove_cache ()
Removes probable cache files.

Результат Filepaths of the removed files.

Тип результата list

evaluation_tools.feature_loader.NetworkFeatureParameters

evaluation_tools.feature_loader.NetworkFeatureParameters
Класс для создания именованных кортежей, содержащий параметры построителей признаков, работающих с данными по сети.

псевдоним класса evaluation_tools.feature_loader.FeatureExtractorParameters

—

Исключения (Exceptions):

ConfigNameNotFoundError	Вызывается если требуемое имя конфигурации (config_name) не найдено в хранилище конфигураций.
ConfigNameNotUniqueError	Вызывается когда в хранилище конфигураций обнаруживается несколько конфигураций с запрашиваемым именем.
FeatureLoaderError	Класс, который используется в качестве базового для всех классов исключений модуля feature_loader.
FeaturesNotFoundError	Вызывается, когда для текущей конфигурации load_features и load_features_dtypes признаков не найдено.
RecordsNotFoundError	Вызывается, когда для текущих значений load_features и load_features_dtypes наборов признаков не найдено.

evaluation_tools.feature_loader.ConfigNameNotFoundError

class evaluation_tools.feature_loader.ConfigNameNotFoundError
Raised when the required config_name is not found in the configuration repository.

evaluation_tools.feature_loader.ConfigNameNotUniqueError

class evaluation_tools.feature_loader.ConfigNameNotUniqueError
Raised when a FeatureLoader founds more than one configuration with the same name in a configuration storage.

evaluation_tools.feature_loader.FeatureLoaderError

class evaluation_tools.feature_loader.FeatureLoaderError
This is a base class for all exceptions within the feature_loader module.

evaluation_tools.feature_loader.FeaturesNotFoundError

class evaluation_tools.feature_loader.FeaturesNotFoundError
Raised when there is no features found for load_features or load_features_dtypes methods configuration.

evaluation_tools.feature_loader.RecordsNotFoundError

class evaluation_tools.feature_loader.RecordsNotFoundError
Raised when there is no records found for load_features or load_features_dtypes methods configuration.

8.1.2 evaluation_tools.model_selection

Содержит классы для разбиения данных на подмножества для кросс-валидации.

SignalAggregationKFold(h[, n_splits])	Класс для создания KFold-сплитов на основе групп последовательных событий с настраиваемой шириной.
TimeRangeBase(n_splits)	Базовый класс для создания разбиений с сортировкой данных по времени.
TimeRangeBaseExtraFold(n_splits)	Базовый класс для создания разбиений с сортировкой данных по времени.
TimeRangeByEntityBase(n_splits)	Базовый класс для создания разбиений с распределением данных каждой сущности равномерно по подмножествам в пределах одного разбиения.
TimeRangeByEntityBaseExtraFold(n_splits)	Базовый класс для создания разбиений с распределением данных каждой сущности равномерно по подмножествам в пределах одного разбиения, создает одно дополнительное подмножество.
TimeRangeByEntityKFold(n_splits)	Создает сплиты с группировкой по имени сущности; 3 сплита будут реализованы как (2+3, 1), (1+3, 2), (1+2, 3), где 1, 2, 3 - входные фолды.
TimeRangeByTimeKFold(n_splits)	Создает сплиты с сортировкой по времени; 3 сплита будут реализованы как (2+3, 1), (1+3, 2), (1+2, 3), где 1, 2, 3 - входные фолды.
TimeRangeConsecutiveByEntityKFold(n_splits)	Создает сплиты с группировкой данных по имени сущности; 3 сплита будут реализованы как (1, 2), (1+2, 3), (1+2+3, 4), где 1, 2, 3, 4 - входные фолды.
TimeRangeConsecutiveByTimeKFold(n_splits)	Создает сплиты с сортировкой по времени; 3 сплита будут реализованы как (1, 2), (1+2, 3), (1+2+3, 4), где 1, 2, 3, 4 - входные фолды.
TimeRangeConsecutiveOnesBase	Базовый класс для создания разбиений на основе групп последовательных событий.
TimeRangeConsecutiveOnesByTimeKFold	Создает сплиты на основе групп последовательных событий. Из 4 групп создаются сплиты (1, 2), (1+2, 3), (1+2+3, 4).
TimeRangeConsecutivePairsByEntityCycledKFold(...)	Создает сплиты с группировкой данных по имени сущности; 3 сплита будут реализованы как (3, 1), (1, 2), (2, 3), где 1, 2, 3 - входные фолды.

continues on next page

Таблица 8.6 – продолжение с предыдущей страницы

TimeRangeConsecutivePairsByEntityKFold(n_splits)	Создает сплиты с группировкой по имени сущности; 3 сплита будут реализованы как (1, 2), (2, 3), (3, 4), где 1, 2, 3, 4 - входные фолды.
TimeRangeConsecutivePairsByTimeCycledKFold(. . .)	Создает сплиты с сортировкой по времени. 3 сплита будут реализованы как (3, 1), (1, 2), (2, 3), где 1, 2, 3 - входные фолды.
TimeRangeConsecutivePairsByTimeKFold(n_splits)	Создает сплиты с сортировкой по времени. 3 сплита будут реализованы как (1, 2), (2, 3), (3, 4), где 1, 2, 3, 4 - входные фолды.

evaluation_tools.model_selection.SignalAggregationKFold

class evaluation_tools.model_selection.SignalAggregationKFold (*h*, *n_splits=None*)

Класс для создания KFold-сплитов на основе групп последовательных событий с настраиваемой шириной.

Создает n сплитов для валидации. Фолды в каждом сплите создаются на основе группы единиц, с добавлением n последующих и n предыдущих для группы значений, и расширением фолдов нулями, не вошедшими ни в один из них ранее. Если значение с одним и тем же индексом попадает в разные фолды одного сплита, дубликат удаляется.

Если параметр n_splits меньше, чем количество групп событий, фолды объединяются по правилу number_of_group % 2.

See more more in Data Scientist User Guide

Параметры

- h (int) – Добавляет к фолду h предыдущих и h последующих для группы значений.
- n_splits (int) – Количество сплитов, если None - равно количеству отдельных групп событий (единиц).

Примеры

```
from model_selection import SignalAggregationKFold

y = [0, 1, 0, 0, 1, 1, 0, 1, 1, 0]

splitter = SignalAggregationKFold(0, 3)
for f in splitter.split(y):
    print(f, end=" ")
# ([2, 3, 4, 5, 6, 7, 8], [1, 0]) ([0, 1, 3, 6, 7, 8], [4, 5, 2]) ([0, 1, 2, 4, 5], [7, 8,
↪ 3, 6])

splitter = SignalAggregationKFold(1, 3)
for f in splitter.split(y):
    print(f, end=" ")
# ([3, 4, 5, 6, 7, 8, 9], [0, 1, 2]) ([0, 1, 2, 6, 7, 8, 9], [3, 4, 5]) ([0, 1, 2, 3, 4,␣
↪5], [6, 7, 8, 9])

splitter = SignalAggregationKFold(0, 2)
for f in splitter.split(y):
    print(f, end=" ")
# ([3, 4, 5, 6], [1, 7, 8, 0, 2]) ([0, 1, 2, 7, 8], [4, 5, 3, 6])
```

Методы

split(y)	Генератор сплитов.

split (*y*)
Генератор сплитов.

Результирующие сплиты содержат фолды не с исходными данными, а с номерами соответствующих строк. Количество сплитов и добавляемых в фолд предшествующих и следующих за группой единиц значений определяется при создании объекта класса.

Параметры y (subscriptable) – Перечислимый объект (pandas.Series, numpy.ndarray, list, tuple, etc.) содержащий информацию о наступлении событий (значения 0 и 1).

Возвращает генератор *tuple* – Сплиты.

evaluation_tools.model_selection.TimeRangeBase

class evaluation_tools.model_selection.TimeRangeBase (*n_splits*)
Базовый класс для создания сплитов с сортировкой данных по времени.

Сортирует значения во входящем столбце и разбивает его на n_splits равных фолдов.

See more in Data Scientist User Guide

Наследуется классами:

- TimeRangeByTimeKFold
- TimeRangeConsecutivePairsByTimeCycledKFold

Параметры n_splits (int) – Количество сплитов.

Raises NotEnoughSplitsError – Cross-validation required at least 2 train/test splits, n_splits parameter should be 2 or more.

Примеры

```
from model_selection import TimeRangeBase


splitter = TimeRangeBase(2)
print(splitter.get_base_folds([1, 2, 3, 4]),
        splitter.get_base_folds([1, 2, 3, 4, 5]))
# [[0, 1], [2, 3]]
# [[0, 1], [2, 3, 4]]

splitter = TimeRangeBase(3)
print(splitter.get_base_folds([1, 2, 3, 4]),
        splitter.get_base_folds([1, 2, 3, 4, 5]))
# [[0], [1, 2], [3]]
# [[0, 1], [2], [3, 4]]
```

Методы

get_base_folds(time_col)	Разбивает данные на фолды.

get_base_folds (*time_col*)

Разбивает данные на фолды.

Результирующие фолды содержат не исходные данные, но номера соответствующих строк. Количество сплитов выбирается при создании экземпляра класса.

Параметры time_col (subscriptable) – Перечислимый объект с данными для разбиения (pandas.Series, numpy.ndarray, list, tuple, etc.)

Результат Список с номерами строк фолдов.

Тип результата list[list[int]]

Raises NotEnoughElementsError – The length of time_col is too small for splitting.

evaluation_tools.model_selection.TimeRangeBaseExtraFold

class evaluation_tools.model_selection.TimeRangeBaseExtraFold (*n_splits*)

Базовый класс для создания сплитов с сортировкой данных по времени, добавляет один дополнительный фолд.

Сортирует данные во входном столбце и создает n_splits+1 равных фолдов. Добавление дополнительного фолда позволяет использовать этот класс в качестве родительского для TimeRangeConsecutiveByTimeKFold и TimeRangeConsecutivePairsByTimeKFold и корректно создавать сплиты.

See more in Data Scientist User Guide

Параметры n_splits (int) – Количество сплитов.

Наследуется классами:

- TimeRangeConsecutiveByTimeKFold
- TimeRangeConsecutivePairsByTimeKFold

Примеры

```
from model_selection import TimeRangeBaseExtraFold


splitter = TimeRangeBaseExtraFold(2)
print(splitter.get_base_folds([1, 2, 3, 4]),
        splitter.get_base_folds([1, 2, 3, 4, 5]))
# [[0], [1, 2], [3]]
# [[0, 1], [2], [3, 4]]

splitter = TimeRangeBaseExtraFold(3)
print(splitter.get_base_folds([1, 2, 3, 4]),
        splitter.get_base_folds([1, 2, 3, 4, 5]))
# [[0], [1], [2], [3]]
# [[0], [1], [2, 3], [4]]
```

Методы

get_base_folds(time_col)	Разбивает данные на фолды.

get_base_folds (*time_col*)
Разбивает данные на фолды.

Результирующие фолды содержат не исходные данные, но номера соответствующих строк.

Параметры time_col (subscriptable) – Перечислимый объект с данными для разбиения (pandas.Series, numpy.ndarray, list, tuple, etc.)

Результат Список с номерами строк фолдов.

Тип результата list[list[int]]

Raises NotEnoughElementsError – The length of time_col is too small for splitting.

evaluation_tools.model_selection.TimeRangeByEntityBase

class evaluation_tools.model_selection.TimeRangeByEntityBase (*n_splits*)
Базовй класс для создания сплитов с распределением данных каждой сущности равномерно по фолдам в пределах одного сплита.

Важно: В текущей реализации класса входные данные должны быть предварительно отсортированы по полям сущность, время (entity, time).

Группирует данные по имени сущности, создает фолды для каждой группы и объединяет соответствующие фолды (первый с первым, второй со вторым и т.д.)

See more in Data Scientist User Guide

Параметры n_splits (int) – Количество сплитов.

Raises NotEnoughSplitsError – Cross-validation required at least 2 train/test splits, n_splits parameter should be 2 or more.

Наследуется классами:

- TimeRangeByEntityKFold
- TimeRangeConsecutivePairsByEntityCycledKFold

Примеры

```
import pandas as pd
from model_selection import TimeRangeByEntityBase


y = ['Jerry', 'Jerry', 'Jerry', 'Tom', 'Tom', 'Tom']

splitter = TimeRangeByEntityBase(2)
print(list(splitter.get_base_folds(y)))
# [[0, 1, 3, 4], [2, 5]]

splitter = TimeRangeByEntityBase(3)
print(list(splitter.get_base_folds(y)))
# [[0, 3], [1, 4], [2, 5]]
```

Методы

get_base_folds(entities)	Разбивает данные на фолды.

get_base_folds (*entities*)

Разбивает данные на фолды.

результирующие фолды содержат не исходные данные, но номера соответствующих строк. Количество сплитов выбирается при создании экземпляра класса.

Параметры entities (subscriptable) – Перечислимый объект (pandas.Series, numpy.ndarray, list, tuple, etc.) содержащий имена сущностей.

Результат Список с номерами строк фолдов.

Тип результата list[list[int]]

Raises NotEnoughElementsError – The length of time_col is too small for splitting.

evaluation_tools.model_selection.TimeRangeByEntityBaseExtraFold

class evaluation_tools.model_selection.TimeRangeByEntityBaseExtraFold (*n_splits*)

Базовый класс для создания сплитов с распределением данных каждой сущности равномерно по фолдам в пределах одного сплита, создает один дополнительный фолд.

Важно: В текущей реализации класса входные данные должны быть предварительно отсортированы по полям сущность, время (entity, time).

Группирует данные по имени сущности, создает фолды для каждой группы и объединяет соответствующие фолды (первый с первым, второй со вторым и т.д.)

Создание дополнительного фолда позволяет использовать этот класс в качестве родительского классами TimeRangeConsecutiveByEntityKFold и TimeRangeConsecutivePairsByEntityKFold для корректного создания сплитов.

See more in Data Scientist User Guide

Параметры n_splits (int) – Количество сплитов.

Наследуется классами:

- TimeRangeConsecutiveByEntityKFold,
- TimeRangeConsecutivePairsByEntityKFold.

Примеры

```
import pandas as pd
from model_selection import TimeRangeByEntityBaseExtraFold


y = ['Jerry', 'Jerry', 'Jerry', 'Tom', 'Tom', 'Tom']

splitter = TimeRangeByEntityBaseExtraFold(2)
print(list(splitter.get_base_folds(y)))
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# [[0, 3], [1, 4], [2, 5]]


splitter = TimeRangeByEntityBaseExtraFold(3)
print(list(splitter.get_base_folds(y)))
# raise NotEnoughElementsError
```

Методы

get_base_folds(entities)	Разбивает данные на фолды.

get_base_folds (*entities*)

Разбивает данные на фолды.

Результирующие фолды содержат не исходные данные, но номера соответствующих строк. Количество сплитов выбирается при создании экземпляра класса.

Параметры entities (subscriptable) – Перечислимый объект (pandas.Series, numpy.ndarray, list, tuple, etc.) содержащий имена сущностей.

Результат Список с номерами строк фолдов.

Тип результата list[list[int]]

Raises NotEnoughElementsError – The length of time_col is too small for splitting or there are not enough elements in input column after distributing data by entities.

evaluation_tools.model_selection.TimeRangeByEntityKFold

class evaluation_tools.model_selection.TimeRangeByEntityKFold (*n_splits*)

Создает сплиты с группировкой по имени сущности; 3 сплита будут реализованы как (2+3, 1), (1+3, 2), (1+2, 3), где 1, 2, 3 - входные фолды.

Использует класс TimeRangeByEntityBase для создания входных фолдов.

See more in Data Scientist User Guide

Параметры n_splits (int) – Количество сплитов.

Примеры

```
import pandas as pd
from model_selection import TimeRangeByEntityKFold


df = pd.DataFrame({'time': [1500000000, 1510000000, 1520000000, 1530000000, 1550000000,␣
↪1580000000],
                   'visitor': ['Jerry', 'Jerry', 'Jerry', 'Tom', 'Tom', 'Tom']})

splitter = TimeRangeByEntityKFold(2)

print("folds:{}\n".format(splitter.get_base_folds(df['visitor'])))
for train_idx, test_idx in splitter.split(df['visitor']):
    train, test = df.iloc[train_idx], df.iloc[test_idx]
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    print("train: \n{}".format(train))
    print("test: \n{}\n".format(test))
```

Результат примера:

```
folds:[[0, 1, 3, 4], [2, 5]]
train:
        time visitor
2  1520000000   Jerry
5  1580000000     Tom
test:
        time visitor
0  1500000000   Jerry
1  1510000000   Jerry
3  1530000000     Tom
4  1550000000     Tom

train:
        time visitor
0  1500000000   Jerry
1  1510000000   Jerry
3  1530000000     Tom
4  1550000000     Tom
test:
        time visitor
2  1520000000   Jerry
5  1580000000     Tom
```

Методы

split(entities)	Генератор сплитов.

split (*entities*)

Генератор сплитов.

Результирующие фолды содержат не исходные данные, но номера соответствующих строк. Количество сплитов выбирается при создании экземпляра класса.

Параметры entities (subscriptable) – Перечислимый объект (pandas.Series, numpy.ndarray, list, tuple, etc.) содержащий имена сущностей.

Возвращает генератор *tuple* – Сплиты.

evaluation_tools.model_selection.TimeRangeByTimeKFold

class evaluation_tools.model_selection.TimeRangeByTimeKFold (*n_splits*)

Создает сплиты с сортировкой по времени; 3 сплита будут реализованы как (2+3, 1), (1+3, 2), (1+2, 3), где 1, 2, 3 - входные фолды.

Использует класс TimeRangeBase для создания входных фолдов.

See more in Data Scientist User Guide

Параметры n_splits (int) – Количество сплитов.

Примеры

```
import pandas as pd
from model_selection import TimeRangeByTimeKFold


df = pd.DataFrame({'time': [1500000000, 1510000000, 1520000000, 1530000000, 1550000000,␣
↪1580000000],
                   'visitor': ['Jerry', 'Jerry', 'Jerry', 'Tom', 'Tom', 'Tom']})

splitter = TimeRangeByTimeKFold(3)

print("folds:{}\n".format(splitter.get_base_folds(df['time'])))
for train_idx, test_idx in splitter.split(df['time']):
    train, test = df.iloc[train_idx], df.iloc[test_idx]
    print("train: \n{}".format(train))
    print("test: \n{}\n".format(test))
```

Результат примера:

```
folds:[[0, 1], [2, 3], [4, 5]]

train:
        time visitor
2  1520000000   Jerry
3  1530000000     Tom
4  1550000000     Tom
5  1580000000     Tom
test:
        time visitor
0  1500000000   Jerry
1  1510000000   Jerry

train:
        time visitor
0  1500000000   Jerry
1  1510000000   Jerry
4  1550000000     Tom
5  1580000000     Tom
test:
        time visitor
2  1520000000   Jerry
3  1530000000     Tom

train:
        time visitor
0  1500000000   Jerry
1  1510000000   Jerry
2  1520000000   Jerry
3  1530000000     Tom
test:
        time visitor
4  1550000000     Tom
5  1580000000     Tom
```

Методы

split(time_col)	Генератор сплитов.

split (*time_col*)
Генератор сплитов.

результирующие фолды содержат не исходные данные, но номера соответствующих строк. Количество сплитов выбирается при создании экземпляра класса.

Параметры time_col (subscriptable) – Перечислимый объект с отметками времени (pandas.Series, numpy.ndarray, list, tuple, etc.)

Возвращает генератор *tuple* – Сплиты.

evaluation_tools.model_selection.TimeRangeConsecutiveByEntityKFold

class evaluation_tools.model_selection.TimeRangeConsecutiveByEntityKFold (*n_splits*)
Создает сплиты с группировкой данных по имени сущности; 3 сплита будут реализованы как (1, 2), (1+2, 3), (1+2+3, 4), где 1, 2, 3, 4 - входные фолды.

Использует класс TimeRangeByEntityBaseExtraFold для создания входных фолдов.

See more in Data Scientist User Guide

Параметры n_splits (int) – Количество сплитов.

Примеры

```
import pandas as pd
from model_selection import TimeRangeConsecutiveByEntityKFold


df = pd.DataFrame({'time': [1500000000, 1510000000, 1520000000, 1530000000, 1550000000,␣
↪1580000000],
                   'visitor': ['Jerry', 'Jerry', 'Jerry', 'Tom', 'Tom', 'Tom']})

splitter = TimeRangeConsecutiveByEntityKFold(2)

print("folds:{}\n".format(splitter.get_base_folds(df['visitor'])))
for train_idx, test_idx in splitter.split(df['visitor']):
    train, test = df.iloc[train_idx], df.iloc[test_idx]
    print("train: \n{}".format(train))
    print("test: \n{}\n".format(test))
```

Результат примера:

```
folds:[[0, 3], [1, 4], [2, 5]]

train:
        time visitor
0  1500000000   Jerry
3  1530000000     Tom
test:
        time visitor
1  1510000000   Jerry
4  1550000000     Tom

train:
        time visitor
0  1500000000   Jerry
3  1530000000     Tom
1  1510000000   Jerry
```

(continues on next page)

(продолжение с предыдущей страницы)

```
4  1550000000     Tom
test:
        time visitor
2  1520000000   Jerry
5  1580000000     Tom
```

Методы

split(entities)	Генератор сплитов.

split (*entities*)

Генератор сплитов.

результирующие фолды содержат не исходные данные, но номера соответствующих строк. Количество сплитов выбирается при создании экземпляра класса.

Параметры entities (subscriptable) – Перечислимый объект (pandas.Series, numpy.ndarray, list, tuple, etc.) содержащий имена сущностей.

Возвращает генератор *tuple* – Сплиты.

evaluation_tools.model_selection.TimeRangeConsecutiveByTimeKFold

class evaluation_tools.model_selection.TimeRangeConsecutiveByTimeKFold (*n_splits*)

Создает сплиты с сортировкой по времени; 3 сплита будут реализованы как (1, 2), (1+2, 3), (1+2+3, 4), где 1, 2, 3, 4 - входные фолды.

Использует класс TimeRangeBaseExtraFold для предварительного создания фолдов.

See more in Data Scientist User Guide

Параметры n_splits (int) – Количество сплитов.

Примеры

```
import pandas as pd
from model_selection import TimeRangeConsecutiveByTimeKFold


df = pd.DataFrame({'time': [1500000000, 1510000000, 1520000000, 1530000000, 1550000000,␣
↪1580000000],
                   'visitor': ['Jerry', 'Jerry', 'Jerry', 'Tom', 'Tom', 'Tom']})

splitter = TimeRangeConsecutiveByTimeKFold(3)

print("folds:{}\n".format(splitter.get_base_folds(df['time'])))
for train_idx, test_idx in splitter.split(df['time']):
    train, test = df.iloc[train_idx], df.iloc[test_idx]
    print("train: \n{}".format(train))
    print("test: \n{}\n".format(test))
```

Результат примера:

```
folds:[[0, 1], [2], [3], [4, 5]]

train:
        time visitor
0  1500000000   Jerry
1  1510000000   Jerry
test:
        time visitor
2  1520000000   Jerry

train:
        time visitor
0  1500000000   Jerry
1  1510000000   Jerry
2  1520000000   Jerry
test:
        time visitor
3  1530000000     Tom

train:
        time visitor
0  1500000000   Jerry
1  1510000000   Jerry
2  1520000000   Jerry
3  1530000000     Tom
test:
        time visitor
4  1550000000     Tom
5  1580000000     Tom
```

Методы

split(time_col)	Генератор сплитов.

split (*time_col*)

Генератор сплитов.

результирующие фолды содержат не исходные данные, но номера соответствующих строк. Количество сплитов выбирается при создании экземпляра класса.

Параметры time_col (subscriptable) – Перечислимый объект с отметками времени (pandas.Series, numpy.ndarray, list, tuple, etc.)

Возвращает генератор *tuple* – Сплиты.

evaluation_tools.model_selection.TimeRangeConsecutiveOnesBase

class evaluation_tools.model_selection.TimeRangeConsecutiveOnesBase

Базовый класс для создания сплитов на основе групп последовательных событий.

Важно: В текущей реализации класса входные данные должны быть предварительно отсортированы по полям сущность, время (entity, time).

Generates folds. One fold consists of sequence of ones (events) and preceding zeroes. See more in Data Scientist User Guide

Наследуется классом TimeRangeConsecutiveOnesByTimeKFold.

Примеры

```
import pandas as pd
from model_selection import TimeRangeConsecutiveOnesBase

y = pd.Series([1, 1, 0, 0, 1, 1, 0, 1, 1, 0])

splitter = TimeRangeConsecutiveOnesBase()
for f in splitter.get_base_folds(y):
    print(f, end=" ")

# [0 1] [2 3 4 5] [6 7 8 9]
```

Методы

get_base_folds(y)	Возвращает генератор фолдов.

get_base_folds (*y*)
Возвращает генератор фолдов.

Параметры y (subscriptable) – Перечислимый объект с данными для разбиения (pandas.Series, numpy.ndarray, list, tuple, etc.)

Возвращает генератор *ndarray* – Фолды.

Raises NoEventsError – No events in input column.

evaluation_tools.model_selection.TimeRangeConsecutiveOnesByTimeKFold

class evaluation_tools.model_selection.TimeRangeConsecutiveOnesByTimeKFold
Создает сплиты на основе групп последовательных событий. Из 4 групп создаются сплиты (1, 2), (1+2, 3), (1+2+3, 4).

Использует класс TimeRangeConsecutiveOnesBase для предварительного создания фолдов.

See more in Data Scientist User Guide

Примеры

```
import pandas as pd
from model_selection import TimeRangeConsecutiveOnesByTimeKFold


df = pd.DataFrame({'time': [1510000000, 1520000000, 1530000000, 1550000000, 1580000000,
                            1600000000, 1610000000, 1630000000, 1700000000, 1710000000],
                   'event': [1, 1, 0, 0, 1, 1, 0, 1, 1, 0]})

splitter = TimeRangeConsecutiveOnesByTimeKFold()
for train_idx, test_idx in splitter.split(df['event']):
    train, test = df.iloc[train_idx], df.iloc[test_idx]
    print("train: \n{}".format(train))
    print("test: \n{}\n".format(test))
```

Результат примера:

```
train:
        time    event
0  1510000000       1
1  1520000000       1
test:
        time    event
2  1530000000       0
3  1550000000       0
4  1580000000       1
5  1600000000       1

train:
        time    event
0  1510000000       1
1  1520000000       1
2  1530000000       0
3  1550000000       0
4  1580000000       1
5  1600000000       1
test:
        time    event
6  1610000000       0
7  1630000000       1
8  1700000000       1
9  1710000000       0
```

Методы

split(y)	Генератор сплитов.

split (*y*)
Генератор сплитов.

результирующие фолды содержат не исходные данные, но номера соответствующих строк.

Параметры y (subscriptable) – Перечислимый объект с отметками времени (pandas.Series, numpy.ndarray, list, tuple, etc.)

Возвращает генератор *tuple* – Сплиты.

Raises NotEnoughEventsGroupsError – Cross-validation required at least 2 train/test splits. The number of events groups in your data must be 3 or more.

evaluation_tools.model_selection.TimeRangeConsecutivePairsByEntityCycledKFold

class evaluation_tools.model_selection.TimeRangeConsecutivePairsByEntityCycledKFold (*n_splits*)
Создает сплиты с группировкой данных по имени сущности; 3 сплита будут реализованы как (3, 1), (1, 2), (2, 3), где 1, 2, 3 - входные фолды.

Вы можете определить количество фолдов, которые будут объединены. Например, если этот параметр равен 2, тогда 3 сплита будут реализованы как: (2+3, 1), (3+1, 2), (1+2, 3).

Использует класс TimeRangeByEntityBase для создания входных фолдов.

See more in Data Scientist User Guide

Параметры n_splits (int) – Количество сплитов.

Примеры

```
import pandas as pd
from model_selection import TimeRangeConsecutivePairsByEntityCycledKFold


df = pd.DataFrame({'time': [1500000000, 1510000000, 1520000000, 1530000000, 1550000000,␣
↪1580000000],
                   'visitor': ['Jerry', 'Jerry', 'Jerry', 'Tom', 'Tom', 'Tom']})

splitter = TimeRangeConsecutivePairsByEntityCycledKFold(3)

print("folds:{}\n".format(splitter.get_base_folds(df['visitor'])))
for train_idx, test_idx in splitter.split(df['visitor'], 2):
    train, test = df.iloc[train_idx], df.iloc[test_idx]
    print("train: \n{}".format(train))
    print("test: \n{}\n".format(test))
```

Результат примера:

```
folds:[[0, 3], [1, 4], [2, 5]]

train:
        time visitor
1  1510000000   Jerry
4  1550000000     Tom
2  1520000000   Jerry
5  1580000000     Tom
test:
        time visitor
0  1500000000   Jerry
3  1530000000     Tom

train:
        time visitor
2  1520000000   Jerry
5  1580000000     Tom
0  1500000000   Jerry
3  1530000000     Tom
test:
        time visitor
1  1510000000   Jerry
4  1550000000     Tom

train:
        time visitor
0  1500000000   Jerry
3  1530000000     Tom
1  1510000000   Jerry
4  1550000000     Tom
test:
        time visitor
2  1520000000   Jerry
5  1580000000     Tom
```

Методы

split(entities[, prev_parts])	Генератор сплитов.

split (*entities*, *prev_parts=1*)
Генератор сплитов.

Результирующие фолды содержат не исходные данные, но номера соответствующих строк. Количество сплитов выбирается при создании экземпляра класса.

Параметры

- entities (subscriptable) – Перечислимый объект (pandas.Series, numpy.ndarray, list, tuple, etc.) содержащий имена сущностей.
- prev_parts (int, optional) – Количество объединяемых фолдов.

Возвращает генератор *tuple* – Сплиты.

evaluation_tools.model_selection.TimeRangeConsecutivePairsByEntityKFold

class evaluation_tools.model_selection.TimeRangeConsecutivePairsByEntityKFold (*n_splits*)
Создает сплиты с группировкой по имени сущности; 3 сплита будут реализованы как (1, 2), (2, 3), (3, 4), где 1, 2, 3, 4 - входные фолды.

Использует класс TimeRangeByEntityBaseExtraFold для создания входных фолдов.

See more in Data Scientist User Guide

Параметры n_splits (int) – Количество сплитов.

Примеры

```
import pandas as pd
from model_selection import TimeRangeConsecutivePairsByEntityKFold


df = pd.DataFrame({'time': [1500000000, 1510000000, 1520000000, 1530000000, 1550000000,␣
↪1580000000],
                   'visitor': ['Jerry', 'Jerry', 'Jerry', 'Tom', 'Tom', 'Tom']})

splitter = TimeRangeConsecutivePairsByEntityKFold(2)

print("folds:{}\n".format(splitter.get_base_folds(df['visitor'])))
for train_idx, test_idx in splitter.split(df['visitor']):
    train, test = df.iloc[train_idx], df.iloc[test_idx]
    print("train: \n{}".format(train))
    print("test: \n{}\n".format(test))
```

Результат примера:

```
folds:[[0, 3], [1, 4], [2, 5]]

train:
        time visitor
0  1500000000   Jerry
3  1530000000     Tom
test:
        time visitor
1  1510000000   Jerry
4  1550000000     Tom
```

(continues on next page)

(продолжение с предыдущей страницы)

```
train:
        time visitor
1  1510000000   Jerry
4  1550000000     Tom
test:
        time visitor
2  1520000000   Jerry
5  1580000000     Tom
```

Методы

split(entities)	Генератор сплитов.

split (*entities*)

Генератор сплитов.

результирующие фолды содержат не исходные данные, но номера соответствующих строк. Количество сплитов выбирается при создании экземпляра класса.

Параметры entities (subscriptable) – Перечислимый объект (pandas.Series, numpy.ndarray, list, tuple, etc.) содержащий имена сущностей.

Возвращает генератор *tuple* – Сплиты.

evaluation_tools.model_selection.TimeRangeConsecutivePairsByTimeCycledKFold

class evaluation_tools.model_selection.TimeRangeConsecutivePairsByTimeCycledKFold (*n_splits*)

Создает сплиты с сортировкой по времени. 3 сплита будут реализованы как (3, 1), (1, 2), (2, 3), где 1, 2, 3 - входные фолды.

Вы можете определить количество фолдов, которые будут объединены. Например, если этот параметр равен 2, тогда 3 сплита будут реализованы как: (2+3, 1), (3+1, 2), (1+2, 3).

Использует класс TimeRangeBase для создания входных фолдов.

See more in Data Scientist User Guide

Параметры n_splits (int) – Количество сплитов.

Примеры

```
import pandas as pd
from model_selection import TimeRangeConsecutivePairsByTimeCycledKFold


df = pd.DataFrame({'time': [1500000000, 1510000000, 1520000000, 1530000000, 1550000000,␣
↪1580000000],
                   'visitor': ['Jerry', 'Jerry', 'Jerry', 'Tom', 'Tom', 'Tom']})

splitter = TimeRangeConsecutivePairsByTimeCycledKFold(3)

print("folds:{}\n".format(splitter.get_base_folds(df['time'])))
for train_idx, test_idx in splitter.split(df['time'], 2):
    train, test = df.iloc[train_idx], df.iloc[test_idx]
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    print("train: \n{}".format(train))
    print("test: \n{}\n".format(test))
```

Результат примера:

```
folds:[[0, 1], [2, 3], [4, 5]]

train:
        time visitor
2  1520000000   Jerry
3  1530000000     Tom
4  1550000000     Tom
5  1580000000     Tom
test:
        time visitor
0  1500000000   Jerry
1  1510000000   Jerry

train:
        time visitor
4  1550000000     Tom
5  1580000000     Tom
0  1500000000   Jerry
1  1510000000   Jerry
test:
        time visitor
2  1520000000   Jerry
3  1530000000     Tom

train:
        time visitor
0  1500000000   Jerry
1  1510000000   Jerry
2  1520000000   Jerry
3  1530000000     Tom
test:
        time visitor
4  1550000000     Tom
5  1580000000     Tom
```

Методы

split(time_col[, prev_parts])	Генератор сплитов.

split (*time_col*, *prev_parts=1*)
Генератор сплитов.

результирующие фолды содержат не исходные данные, но номера соответствующих строк. Количество сплитов выбирается при создании экземпляра класса.

Параметры

- time_col (subscriptable) – Перечислимый объект с отметками времени (pandas.Series, numpy.ndarray, list, tuple, etc.)
- prev_parts (int, optional) – Количество объединяемых фолдов. По умолчанию равно 1.

Возвращает генератор *tuple* – Сплиты.

evaluation_tools.model_selection.TimeRangeConsecutivePairsByTimeKFold

class evaluation_tools.model_selection.TimeRangeConsecutivePairsByTimeKFold (*n_splits*)

Создает сплиты с сортировкой по времени. 3 сплита будут реализованы как (1, 2), (2, 3), (3, 4), где 1, 2, 3, 4 - входные фолды.

Использует класс TimeRangeBaseExtraFold для создания входных фолдов.

See more in Data Scientist User Guide

Параметры n_splits (int) – Количество сплитов.

Примеры

```
import pandas as pd
from model_selection import TimeRangeConsecutivePairsByTimeKFold


df = pd.DataFrame({'time': [1500000000, 1510000000, 1520000000, 1530000000, 1550000000,␣
↪1580000000],
                   'visitor': ['Jerry', 'Jerry', 'Jerry', 'Tom', 'Tom', 'Tom']})

splitter = TimeRangeConsecutivePairsByTimeKFold(3)

print("folds:{}\n".format(splitter.get_base_folds(df['time'])))
for train_idx, test_idx in splitter.split(df['time']):
    train, test = df.iloc[train_idx], df.iloc[test_idx]
    print("train: \n{}".format(train))
    print("test: \n{}\n".format(test))
```

Результат примера:

```
folds:[[0, 1], [2], [3], [4, 5]]

train:
        time visitor
0  1500000000   Jerry
1  1510000000   Jerry
test:
        time visitor
2  1520000000   Jerry

train:
        time visitor
2  1520000000   Jerry
test:
        time visitor
3  1530000000     Tom

train:
        time visitor
3  1530000000     Tom
test:
        time visitor
4  1550000000     Tom
5  1580000000     Tom
```

Методы

split(time_col)	Генератор сплитов.

split (*time_col*)
Генератор сплитов.

результирующие фолды содержат не исходные данные, но номера соответствующих строк. Количество сплитов выбирается при создании экземпляра класса.

Параметры time_col (subscriptable) – Перечислимый объект с отметками времени (pandas.Series, numpy.ndarray, list, tuple, etc.)

Возвращает генератор *tuple* – Сплиты.

Исключения (Exceptions):

ModelSelectionError	Базовый класс исключений для модуля model_selection.
NoEventsError	Вызывается в случае, если во входном столбце не обозначены события (нет единиц).
NotEnoughElementsError	Вызывается в случае, если элементов во входном столбце недостаточно для создания требуемых сплитов.
NotEnoughEventsGroupsError	Вызывается в случае, если групп событий во входном столбце недостаточно для создания требуемых сплитов.
NotEnoughSplitsError	Вызывается в случае, если количество сплитов меньше 2.

evaluation_tools.model_selection.ModelSelectionError

class evaluation_tools.model_selection.ModelSelectionError
Base exception class for model_selection module.

evaluation_tools.model_selection.NoEventsError

class evaluation_tools.model_selection.NoEventsError
Raised if there is no events (ones) in input column.

evaluation_tools.model_selection.NotEnoughElementsError

class evaluation_tools.model_selection.NotEnoughElementsError
Raised if the number of elements in the input column is not enough to create the required splits.

evaluation_tools.model_selection.NotEnoughEventsGroupsError

class evaluation_tools.model_selection.NotEnoughEventsGroupsError
Raised if the number of groups of events in the input column is not enough to create the required splits.

evaluation_tools.model_selection.NotEnoughSplitsError

class evaluation_tools.model_selection.NotEnoughSplitsError
Raised if the number of splits is less than 2.

8.1.3 evaluation_tools.models_player_requests

Содержит класс Requests и исключения для работы с сервисомModels Player через объекты Python.

Requests(host, port)	Содержит методы, предоставляющие функциональность HTTP API сервиса Models Player.

evaluation_tools.models_player_requests.Requests

class evaluation_tools.models_player_requests.Requests (*host*, *port*)

Содержит методы, предоставляющие функциональность HTTP API сервиса Models Player.

Таким образом, используя данный класс, можно работать с сервисом через Python.

See documentation of Models Player HTTP API with examples on *this page*

Параметры

- host (str) – Хост, на котором запущен сервис Models Player.
- port (str) – Номер порта.

Примеры

Пример создания объекта класса:

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
```

Методы

download_model(filename[, save_folder])	Загружает архив с моделью на локальный диск пользователя.
get_models_settings(**model_kwargs)	Возвращает параметры запущенных моделей.
get_models_stats(**model_kwargs)	Возвращает статистику (время запуска, время инициализации и т.д.) для всех запущенных моделей.
list_archives_dir()	Возвращает список архивов с моделями.
list_models_dir()	Возвращает список моделей (директорий моделей).
list_running_models()	Возвращает список запущенных моделей (директорий моделей).
pack_model(model_id[, rm_sources])	Метод для упаковки модели в архив.
predict_batch(model_id, features)	Возвращает предсказание модели на наборе входных векторов.
remove_archive(filename)	Метод для удаления архива модели из директории с архивами.
remove_model_dir(model_id)	Удаляет директорию модели.
start_all_models()	Запускает все модели в директории моделей.
start_model(model_id)	Запускает модель с выбранным model_id.
stop_all_models()	Останавливает все запущенные модели.
stop_model(model_id)	Останавливает модель с выбранным ID.
unpack_model(filename)	Метод для распаковки модели из архива в директорию.
upload_model(filename)	Загружает архив с моделью на сервер.

download_model (*filename*, *save_folder='./'*)

Загружает архив с моделью на локальный диск пользователя.

Идентифицирует архив по параметру filename, переданном в строке GET-запроса. Имя файла должно иметь расширение .zip или .tar.gz.

Возвращает пользователю модель в архиве .tar.gz.

Параметры

- filename (str) – Имя архива.
- save_folder (str, optional) – Относительный путь к сохраняемому файлу; если не указан - файл будет сохранен в текущую директорию. (Значение по умолчанию = „./“)

Raises

- DownloadError – Status code not 200 or request returns an error.
- NotImplementedError – In case of using the save_folder arg: it is not implemented yet.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.download_model('model_1.tar.gz')
```

get_models_settings (**model_kwargs)

Возвращает параметры запущенных моделей.

Ответ включает в себя идентификатор модели, порт и конфигурацию (лог-файл по умолчанию, имя модели, имя задачи, идентификатор процесса (pid) и т.д.)

Можно получить параметры конкретной модели, если указать директорию, ID или имя класса (имя Python-класса, в котором описывается поведение модели в модуле Python). Если модель не идентифицируется, метод возвращает информацию обо всех зарегистрированных моделях.

Параметры **model_kwargs – Метод может принимать в качестве аргумента один из следующих параметров: **model_id** (имя директории модели), **model_dir** (путь к модели с именем директории), или **model_classname** (имя класса модели в соответствующем модуле Python).

Результат Параметры конкретной модели, если она идентифицирована, в противном случае - параметры всех зарегистрированных моделей.

Тип результата list[dict]

Raises ModelsInfoError – Status code not 200.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')

print(mp.get_models_settings())  # for all models
print(mp.get_models_settings(model_id='model_1'))  # for one model
    in other case - settings about all registered models.
```

get_models_stats (**model_kwargs)

Возвращает статистику (время запуска, время инициализации и т.д.) для всех запущенных моделей.

Параметры **model_kwargs – Метод может принимать в качестве аргумента один

из следующих параметров: **model_id** (имя директории модели), **model_dir** (путь к модели с именем директории), или **model_classname** (имя класса модели в соответствующем модуле Python).

Результат Статистика по конкретной модели, если модель идентифицирована, в противном случае - статистика по всем зарегистрированным моделям в виде списка.

Тип результата list[dict]

Raises ModelsInfoError – Status code not 200.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
print(mp.get_models_stats())
```

list_archives_dir ()

Возвращает список архивов с моделями.

Директория с архивами задается в конфигурации Models Player.

Результат Список файлов в директории с архивами моделей.

Тип результата list

Raises ModelsListError – Status code not 200.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.list_archives_dir()
```

list_models_dir ()

Возвращает список моделей (директорий моделей).

Путь к директории задается в конфигурации Models Player.

Результат Список моделей (директорий моделей).

Тип результата list

Raises ModelsListError – Status code not 200.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.list_models_dir()
```

list_running_models ()

Возвращает список запущенных моделей (директорий моделей).

Путь к директории задается в конфигурации Models Player.

Результат Список запущенных моделей.

Тип результата list

Raises ModelsListError – Status code not 200.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.list_running_models()
```

pack_model (*model_id*, *rm_sources=None*)

Метод для упаковки модели в архив.

Определяет модель по ID. Архивирует модель в формате .tar.gz и добавляет в директорию с архивами.

Параметры

- model_id (str) – Имя директории модели.
- rm_sources (bool, optional) – Если Истина - удаляет директорию с моделью, если модель не запущена. (Значение по умолчанию = None)

Результат Список файлов в директории с архивами моделей.

Тип результата list

Raises PackError – Status code not 200 or request returns an error.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.pack_model('model_1')
```

predict_batch (*model_id*, *features*)

Делает предсказание запущенной модели на наборе векторов. Принимает список векторов X и возвращает список с предсказаниями.

Параметры

- model_id (str) – ID модели для предсказаний;
- features (list[list]) – Список векторов признаков.

Результат Список словарей {«y_pred»: result} для каждого вектора X.

Тип результата list[dict]

Raises PredictError – Status code not 200.

Примеры

```
import pandas as pd
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
x = pd.DataFrame({"f1": [1, 2, 3, 4, 5], "f2": [0, 1, 1, 1, 0]})

mp.predict_batch('model_1', x.values.tolist())
```

remove_archive (*filename*)

Метод для удаления архива модели из директории с архивами.

Идентифицирует архив по параметру „filename“. Имя файла должно иметь расширение .zip или .tar.gz.

Если удаление прошло успешно, возвращает список файлов в директории с архивами.

Параметры filename (str) – Имя архива.

Результат Список файлов в директории с архивами моделей.

Тип результата list

Raises RemoveModelsError – Status code not 200 or request returns an error.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.remove_archive('model_1.tar.gz')
```

remove_model_dir (*model_id*)

Удаляет директорию модели.

Идентифицирует модель по параметру model_id, переданному в строке GET-запроса.

Если удаление прошло успешно, возвращает список файлов в директории с моделями.

Параметры model_id (str) – Имя директории модели (model_id).

Результат Список папок моделей в директории с моделями.

Тип результата list

Raises RemoveModelsError – Status code not 200 or request returns an error.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.remove_model_dir('model_1')
```

start_all_models ()

Запускает все модели в директории моделей.

Целевая директория задается в конфигурации Models Player. Каждая модель запускается в новом процессе, но только если не была запущена ранее.

Для каждой модели метод возвращает один из следующих статусов:

- ignored (если по время запуска модели произошли некоторые ошибки);
- started (если модель была запущена успешно);
- working (если модель уже была запущена на момент попытки старта).

Результат Расположение и статусы запущенных моделей.

Тип результата dict

Raises StartAllModelsError – Status code not 200 or request returns an error.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.start_all_models()
```

start_model (*model_id*)

Запускает модель с выбранным model_id.

Модель запускается в новом процессе; запуск происходит только в том случае, если она не была запущена ранее.

Также метод возвращает следующие статусы:

- ignored (если по время запуска модели произошли некоторые ошибки);
- started (если модель была запущена успешно);
- working (если модель уже была запущена на момент попытки старта).

Параметры model_id (str) – Идентификатор модели.

Результат Директория и статус модели.

Тип результата dict

Raises StartOneModelError – Status code not 200 or request returns an error.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.start_one_model('model_1')
```

stop_all_models ()

Останавливает все запущенные модели.

Результат Расположение моделей и сообщения о результатах их остановки.

Тип результата dict

Raises StopAllModelsError – Status code not 200 or request returns an error.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.stop_all_models()
```

stop_model (*model_id*)

Останавливает модель с выбранным ID.

Параметры model_id (str) – Имя директории модели.

Результат Расположение и результат остановки модели.

Тип результата dict

Raises StopOneModelError – Status code not 200 or request returns an error.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.stop_one_model('model_1')
```

unpack_model (*filename*)

Метод для распаковки модели из архива в директорию.

Идентифицирует архив по параметру „filename“. Имя файла должно иметь расширение .zip или .tar.gz. Распакованная модель попадает в директорию с моделями (путь определяется в конфигурации Models Player).

Параметры filename (str) – Имя архива.

Результат Список каталогов в директории с моделями.

Тип результата list

Raises UnpackError – Status code not 200 or request returns an error.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.unpack_model('model_1.tar.gz')
```

upload_model (*filename*)

Загружает архив с моделью на сервер.

Метод для загрузки в Models Player архива с моделью.

Сохраняет архив в соответствующую директорию (путь настраивается в конфигурации Models Player). Файл должен иметь расширение .zip или .tar.gz.

Для корректной работы директория models_archives должна существовать.

Параметры filename (str) – Имя файла архива в формате .tar.gz или .zip.

Результат Список файлов в директории с архивами моделей.

Тип результата list

Raises UploadError – Status code not 200 or request returns an error.

Примеры

```
import models_player_requests

mp = models_player_requests.Requests(host='0.0.0.0', port='10005')
mp.upload_model('model_1.tar.gz')
```

Исключения (Exceptions):

DownloadError	Вызывается в случае, если во время загрузки архива с сервера запрос возвращает ошибку, или возвращаемый код состояния отличается от «OK».
ModelArchiveError	This is a base class for exceptions related to model archive.
ModelsInfoError	Вызывается в случае неудачного завершения запроса информации по моделям.
ModelsListError	Вызывается в случае, если при запросе списка запущенных моделей возвращаемый код состояния отличается от «OK».
PackError	Вызывается в случае, если во время архивирования модели запрос возвращает ошибку, или возвращаемый код состояния отличается от «OK».
PredictError	Вызывается в случае, если при вызове метода Requests.predict_batch возвращаемый код состояния отличается от «OK».
RemoveModelsError	Вызывается в случае, если во время удаления директории или архива модели запрос возвращает ошибку, или возвращаемый код состояния отличается от «OK».
StartAllModelsError	Вызывается в случае, если во время запуска всех моделей запрос возвращает ошибку, или возвращаемый код состояния отличается от «OK».
StartOneModelError	Вызывается в случае, если во время запуска одной модели запрос возвращает ошибку, или возвращаемый код состояния отличается от «OK».
StopAllModelsError	Вызывается в случае, если во время остановки всех моделей запрос возвращает ошибку, или возвращаемый код состояния отличается от «OK».
StopOneModelError	Вызывается в случае, если при остановке одной модели запрос возвращает ошибку, или возвращаемый код состояния отличается от «OK».
UnpackError	Вызывается в случае, если во время распаковки модели из архива в директорию запрос возвращает ошибку, или возвращаемый код состояния отличается от «OK».
UploadError	Вызывается при некорректном завершении загрузки архива модели на сервер.

evaluation_tools.models_player_requests.DownloadError

class evaluation_tools.models_player_requests.DownloadError

Вызывается в случае, если во время загрузки архива с сервера запрос возвращает ошибку, или возвращаемый код состояния отличается от «OK».

evaluation_tools.models_player_requests.ModelArchiveError

class evaluation_tools.models_player_requests.ModelArchiveError

This is a base class for exceptions related to model archive.

evaluation_tools.models_player_requests.ModelsInfoError

class evaluation_tools.models_player_requests.ModelsInfoError
Вызывается в случае неудачного завершения запроса информации по моделям.

evaluation_tools.models_player_requests.ModelsListError

class evaluation_tools.models_player_requests.ModelsListError
Raised if status code not „OK“ for request list of running models, models directories or models archives.

evaluation_tools.models_player_requests.PackError

class evaluation_tools.models_player_requests.PackError
Вызывается в случае, если во время архивирования модели запрос возвращает ошибку, или возвращаемый код состояния отличается от «OK».

evaluation_tools.models_player_requests.PredictError

class evaluation_tools.models_player_requests.PredictError
Вызывается в случае, если при вызове метода Requests.predict_batch возвращаемый код состояния отличается от «OK».

evaluation_tools.models_player_requests.RemoveModelsError

class evaluation_tools.models_player_requests.RemoveModelsError
Raised if status code not „OK“ or request returns an error during removing model directory or archive.

evaluation_tools.models_player_requests.StartAllModelsError

class evaluation_tools.models_player_requests.StartAllModelsError
Raised if status code not „OK“ or request returns an error during starting all models.

evaluation_tools.models_player_requests.StartOneModelError

class evaluation_tools.models_player_requests.StartOneModelError
Raised if status code not „OK“ or request returns an error during starting one model work.

evaluation_tools.models_player_requests.StopAllModelsError

class evaluation_tools.models_player_requests.StopAllModelsError
Raised if status code not „OK“ or request returns an error during stop all models.

evaluation_tools.models_player_requests.StopOneModelError

class evaluation_tools.models_player_requests.StopOneModelError
Raised if status code not „OK“ or request returns an error during stop one model.

evaluation_tools.models_player_requests.UnpackError

class evaluation_tools.models_player_requests.UnpackError

Вызывается в случае, если во время распаковки модели из архива в директорию запрос возвращает ошибку, или возвращаемый код состояния отличается от «OK».

evaluation_tools.models_player_requests.UploadError

class evaluation_tools.models_player_requests.UploadError

Исключение, выбрасывается при некорректном завершении загрузки архива модели на сервер.

8.1.4 evaluation_tools.notebook_downloader

Содержит класс NotebookDownloader для загрузки Jupyter-ноутбуков из хранилища сабмитов на локальный диск пользователя.

NotebookDownloader([config_name, . . .])	Класс для загрузки jupyter-ноутбуков из хранилища результатов моделирования на локальный диск.

evaluation_tools.notebook_downloader.NotebookDownloader

class evaluation_tools.notebook_downloader.NotebookDownloader(*config_name=None*, *mongo_config=None*, *tool_config=None*, *logger=None*, *verbose=False*, *config_not_found_error=True*, *use_config_from_mongo_directly=False*, ***kwargs*)

Класс для загрузки jupyter-ноутбуков из хранилища результатов моделирования на локальный диск.

Параметры

- config_name (str) – Name of tool configuration.
- mongo_config (dict, optional) – Configuration for config database with parameters: host, port, db. In case of None, connection settings will be obtained from ProjectContextManager. (Default value = None)
- tool_config (dict, optional) – is being used as config of tool is specified instead of getting via config_db_connector.ConfigConnector. (Default value = None)
- logger (logging.Logger, optional) – Logger object from logging module. (Default value = None)
- verbose (bool, optional) – Controls the verbosity when requesting features. (Default value = False)
- config_not_found_error (bool, optional) – This flag blocks raising error when tool config not found. This is using for tests and tools without config. (Default value = True)

Примеры

```
>>> from notebook_downloader import NotebookDownloader
>>>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
>>>
>>> nd = NotebookDownloader(config_name="test", config_not_found_error=False)
>>> # 'config_not_found_error' should be False while NotebookDownloader has not config.
>>> # 'config_name' may be arbitrary.
>>> submit = nd.download_notebook_to_file("5d9d9ecbd81b2b6e1e0a2062")
```

Методы

download_notebook_to_file(submit_id[, . . .])	Метод для загрузки ноутбука из хранилища результатов моделирования.
get_config_storage_name()	Overrided parent method

download_notebook_to_file (*submit_id*, *filename='notebook.ipynb'*, *attach_filename='attachments.gz'*, *overwrite=False*)

Метод для загрузки ноутбука из хранилища результатов моделирования.

Параметры

- submit_id (str) – Submit ID to search for submit record and get notebook file.
- filename (str, optional) – Path with filename to save notebook. (Default value = „notebook.ipynb“). It could be several notebooks and in this case they will be named as named in „attachments“.
- attach_filename (str, optional) – Path with filename to save attachments. (Default value = „attachments.gz“)
- overwrite (bool) – Если Истина - метод переписывает файлы с одинаковыми именами, в противном случае (если Ложь), новый файл создается с добавлением постфикса __[номер] перед расширением .ipynb. (Значение по умолчанию = False)

Результат Tuple (filename, record), where the „filename“ is a list of of saved notebook_names, „record“ - submits storage record with notebook description (author, version, message, etc.)

Тип результата tuple

Raises

- SubmitNotFoundError – There is no notebooks with a requested name in the database.
- MoreThanOneSubmitFoundError – There is more than one notebook with requested name in database.

Примеры

```
>>> from notebook_downloader import NotebookDownloader
>>>
>>>
>>> submit_id = '5dd257a2781af7723a25ab58'
>>> nd = NotebookDownloader(config_name="test", config_not_found_error=False)
>>>
>>> nd.download_notebook_to_file(submit_id, filename='my_notebook.ipynb')
```

get_config_storage_name ()

Overrided parent method

Результат Name of the tool's config storage name. In this case - mongo collection name.

Тип результата str

—

Исключения (Exceptions):

MoreThanOneSubmitFoundError	Вызывается в случае, если при попытке загрузить из хранилища ноутбук с результатами моделирования найдено несколько ноутбуков с требуемым названием.
SubmitNotFoundError	Вызывается в случае, если в базе данных нет ноутбуков с запрашиваемым именем.

evaluation_tools.notebook_downloader.MoreThanOneSubmitFoundError

class evaluation_tools.notebook_downloader.MoreThanOneSubmitFoundError
Raised if there is more than one notebook with requested name in database.

evaluation_tools.notebook_downloader.SubmitNotFoundError

class evaluation_tools.notebook_downloader.SubmitNotFoundError
Raised if there is no one notebook with a requested name in the database.

8.1.5 evaluation_tools.raw_data_loader

Модуль содержит класс RawDataLoader, созданный на основе FeatureLoader. RawDataLoader используется для загрузки данных для разведывательного анализа данных.

RawDataLoader([config_name, mongo_config, . . .])	Загружает исходные данные из Data Service по HTTP API.

evaluation_tools.raw_data_loader.RawDataLoader

class evaluation_tools.raw_data_loader.RawDataLoader(*config_name=None*, *mongo_config=None*, *tool_config=None*, *logger=None*, *verbose=False*, *config_not_found_error=True*, *use_config_from_mongo_directly=False*, ***kwargs*)

Gets raw data from Data Service via http API. The class is inherited from FeatureLoader and has the same parameters.

Параметры

- config_name (str) – Name of tool configuration.
- mongo_config (dict, optional) – Configuration for database with parameters: host, port, db. In case of None, connection settings will be obtained from ProjectContextManager. (Default value = None)
- tool_config (dict, optional) – is being used as config of tool is specified instead of getting via config_db_connector.ConfigConnector. (Default value = None)
- logger (logging.Logger, optional) – Logger object from logging module. (Default value = None)
- verbose (bool, optional) – Controls the verbosity when requesting features. (Default value = False)

- config_not_found_error (bool, optional) – This flag blocks raising error when tool config not found. This is using for tests and tools without config. (Default value = True)
- feature_store_config (dict, optional) – dict with configuration of the REST API of Feature Store Service. Configuration must be written in form: { "host" : "127.0.0.1", "port" : 8590, "ssh_tunnel" : {"ssh_host" : "some.host.ip", "ssh_port" : 8080 }}. This parameter is transmitted via kwargs. (Default value = None)
- tmp_dir (str, optional) – The full path to the directory for storing temporary files. This parameter is transmitted via kwargs. (Default value = „/tmp/dataskai“)

Примеры

```
>>> from task_loader import TaskLoader
>>>
>>>
>>> task_loader = TaskLoader(config_name="raw_data__aero__fw_classification_v1_with_
↪fstrings",
...         subtool_config_from_context=True)
>>> raw_data_loader = task_loader.raw_data_loader
>>>
>>> dtypes = raw_data_loader.load_data_dtypes(record_to_use="cfds_events")
>>> print(f"Raw data dtypes: {dtypes}")
Raw data dtypes: {"parent_report_time": "np.int64", "parent_report_id": "np.int64" ...}
>>>
>>> df = raw_data_loader.load_data_one_record(record_to_use="cfds_events")
>>> print(f"Shape of the loaded raw data: {df.shape}")
Shape of the loaded raw data: (321292, 12)
```

Methods

get_config_storage_name()	Overrided parent method
load_data_dtypes(record_to_use[, ...])	Получает из конфигурационного файла словарь типов, в котором ключами являются имена столбцов, а значениями — типы данных.
load_data_one_record(record_to_use[, ...])	Loads data for specified record and (optionally) specified columns.
load_features(*args, **kwargs)	Загружает и возвращает признаки, соответствующие объекту FeatureLoader.
load_features_dtypes(*args, **kwargs)	Возвращает словарь типов для признаков, соответствующих объекту FeatureLoader.

get_config_storage_name ()
Overrided parent method

Результат Name of the tool's config storage name. In this case - mongo collection name.

Тип результата str

load_data_dtypes (*record_to_use*, *feature_selector=<function RawDataLoader.<lambda>>*)
Получает из конфигурационного файла словарь типов, в котором ключами являются имена столбцов, а значениями — типы данных.

Метод полезен, когда при загрузке признаков возникают ошибки с преобразованием типов данных. Для устранения таких ошибок предусмотрен метод _modify_config.

Types casting is applied to data when RawDataLoader tries to return pandas.DataFrame, when data is already downloaded from Data Service datasets and cached locally.

Если в конфигурационном файле не указан тип данных для конкретного столбца, применяется

тип данных по умолчанию (указанный в конфигурационном файле).

Параметры

- record_to_use (str) – Имя записи из конфигурационного файла.
- feature_selector (function, optional) – Булева функция вида str -> bool для отбора желаемых столбцов по именам. (Значение по умолчанию = lambda c: True)

Результат Словарь из конфигурационного файла. Ключами словаря являются имена столбцов, а значениями — типы данных.

Тип результата dict

load_data_one_record (*record_to_use*, *feature_selector=<function RawDataLoader.<lambda>>*, *verbose=False*)

Loads data for specified record and (optionally) specified columns.

Укажите в record_to_use имя записи из конфигурационного файла. Чтобы посмотреть список всех доступных имён записей, используйте _list_records_names. Запрос с недоступным именем так же выведет список всех доступных имён записей.

Опционально можно задать параметр feature_selector с помощью булевой функции для имён столбцов. Позволяет избежать загрузки всех столбцов, когда нужна только их часть.

Параметры

- record_to_use (str) – Имя записи из конфигурационного файла.
- feature_selector (function, optional) – Булева функция вида str -> bool для отбора желаемых столбцов по именам. (Значение по умолчанию = lambda c: True)
- verbose (bool) – Logging flag.

Результат Исходные данные.

Тип результата pandas.DataFrame

load_features (**args*, ***kwargs*)

Загружает и возвращает признаки, соответствующие объекту FeatureLoader.

Возвращает наборы признаков, соответствующие параметру records_to_use, который содержит имя набора (см. self.list_records, параметр name в списке feature_records) или их список. Если records_to_use имеет значение None, метод выполнит загрузку всех доступных для объекта FeatureLoader наборов признаков. В случае если метод загружает несколько объектов pd. DataFrame, они объединяются в один по значениям индексов. Дублирующие столбцы удаляются.

Параметр feature_selector — функция, которая используется для определения релевантных признаков.

Параметры

- records_to_use (str or list[str], optional) – Имя набора признаков для загрузки или список таких имён. Загружаются только релевантные признаки. (Значение по умолчанию = None)
- feature_selector (function(str), optional) – Функция „column_name -> bool“, которая используется для отбора релевантных признаков. Если возвращаемое значение feature_selector(feature_column) == True, это означает, что признак необходимо включить в выборку. Если возвращаемое значение feature_selector(feature_column) == False, признак в выборку не включается. (Значение по умолчанию = lambda c: True)

Результат Релевантные признаки.

Тип результата pandas.DataFrame

load_features_dtypes (**args*, ***kwargs*)

Возвращает словарь типов для признаков, соответствующих объекту FeatureLoader.

Возвращает типы данных соответственно параметру records_to_use, который содержит имя набора признаков (см self.list_records, параметр name в списке feature_records) или список наборов. Если records_to_use имеет значение None, метод выполнит загрузку всех доступных для объекта FeatureLoader наборов признаков. Повторения типов данных удаляются.

Параметр feature_selector — функция, которая используется для определения релевантных признаков.

Параметры

- records_to_use (str or list[str], optional) – Имя набора признаков для загрузки или список таких имен. Загружаются только релевантные признаки. (Значение по умолчанию = None)
- feature_selector (function(str), optional) – Функция „column_name -> bool“, которая используется для отбора релевантных признаков. Если возвращаемое значение feature_selector(feature_column) == True, это означает, что признак необходимо включить в выборку. Если возвращаемое значение feature_selector(feature_column) == False, признак в выборку не включается. (Значение по умолчанию = lambda c: True)

Результат Имена признаков с указанием типов данных.

Тип результата dict

—

Исключения (Exceptions):

RawDataLoaderError	Класс, который используется в качестве базового для всех классов исключений модуля raw_data_loader.
RecordNameNotExistsError	Возникает когда не задано имя записи, или запись с таким именем отсутствует.
UnsuitableMethodError	«Заглушка» для методов, унаследованных от FeatureLoader, но не использующихся в RawDataLoader .

evaluation_tools.raw_data_loader.RawDataLoaderError

class evaluation_tools.raw_data_loader.RawDataLoaderError

This is a base class for all exceptions within the raw_data_loader module.

evaluation_tools.raw_data_loader.RecordNameNotExistsError

class evaluation_tools.raw_data_loader.RecordNameNotExistsError

Raised if required record name isn't specified or not exists.

evaluation_tools.raw_data_loader.UnsuitableMethodError

class evaluation_tools.raw_data_loader.UnsuitableMethodError

Used as dummy plug for unsuitable methods inherited from FeatureLoader.

8.1.6 evaluation_tools.submitter

Инструменты для отправки результатов моделирования в хранилище и скачивания jupyter-ноутбуков с решениями.

Submitter([config_name, mongo_config, . . .])	Класс для отправки результатов моделирования на сервер.

evaluation_tools.submitter.Submitter

class evaluation_tools.submitter.Submitter (*config_name=None*, *mongo_config=None*, *tool_config=None*, *logger=None*, *verbose=False*, *config_not_found_error=True*, *use_config_from_mongo_directly=False*, ***kwargs*)

Класс для отправки результатов моделирования на сервер.

When a Submitter object is created, it connects to the storage and loads required configuration parameters, such as dataframe dimensions, allowed columns names, authors and etc.

Если вместе с результатами моделирования вы отправляете на сервер data scaler, он должен быть в списке разрешенных объектов. По умолчанию разрешены следующие data scalers библиотеки scikit-learn: StandardScaler, Binarizer, FunctionTransformer, Imputer, KernelCenterer, LabelBinarizer, LabelEncoder, MultiLabelBinarizer, MaxAbsScaler, MinMaxScaler, Normalizer, OneHotEncoder, PolynomialFeatures, QuantileTransformer, RobustScaler.

Вы можете заменить этот список своим, используя параметр allowed_scalers.

Также по умолчанию Submitter содержит список моделей, для которых необязательно задавать параметры заполнения неопределенных (NA) значений: Booster, XGBClassifier, XGBModel, XGBRegressor, Booster, LGBMClassifier, LGBMModel, LGBMRanker, LGBMRegressor, CatBoost, CatBoostClassifier, CatBoostRegressor

Переписать этот список можно, используя параметр models_without_additional_data в конфигурации Submitter.

Параметры

- config_name (str) – Name of tool configuration.
- mongo_config (dict, optional) – Configuration for database with parameters: host, port, db. In case of None, connection settings will be obtained from ProjectContextManager. (Default value = None)
- tool_config (dict, optional) – is being used as config of tool is specified instead of getting via config_db_connector.ConfigConnector. (Default value = None)
- logger (logging.Logger, optional) – Logger object from logging module. (Default value = None)
- verbose (bool, optional) – Controls the verbosity when requesting features. (Default value = False) config_not_found_error(bool, optional): This flag blocks raising error when tool config not found. This is using for tests and tools without config. (Default value = True)

Examples

```
>>> import pandas
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.preprocessing import StandardScaler
```

(continues on next page)

(продолжение с предыдущей страницы)

```
>>> from submitter import Submitter
>>>
>>>
>>> sb = Submitter(config_name="test_config")
>>> sb.submit_results(results_df=pandas.DataFrame(data={"tar": [i for i in range(10)]}),
...         message="mock message",
...         author_name="Mock.Author",
...         model_name="Logistic Regression",
...         model_version="1.0",
...         task_name="Mock_task",
...         model=LogisticRegression(),
...         model_feature_columns=["A", "B", "C"],
...         model_fill_na_values=[0, 0, 0],
...         data_scaler=StandardScaler(),
...         attachments=["tests//files//notebook_for_test.ipynb"])
'5f0dd010d1cb784c40b918f1'
```

Методы

get_config_storage_name()	Overrided parent method
submit_results(results_df, message, . . . [, . . .])	Принимает результаты моделирования, проводит валидацию, и если она успешна - отправляет решение на сервер.

get_config_storage_name ()
Overrided parent method

Результат Name of the tool's config storage name. In this case - mongo collection name.

Тип результата str

submit_results (*results_df*, *message*, *author_name*, *model_name*, *model_version*, *task_name*, *feature_mining_comments=''*, *tags=None*, *model=None*, *model_feature_columns=None*, *model_feature_importances=None*, *model_fill_na_values=None*, *data_scaler=None*, *attachments=None*)
Принимает результаты моделирования, проводит валидацию, и если она успешна - отправляет решение на сервер.

Валидация включает в себя проверку данных и метаданных, в том числе:

- количество строк и столбцов;
- имена столбцов;
- поля метаданных (разрешения для автора сабмита, имени задачи);
- параметры модели: признаки, параметры заполнения неопределенных (NA) значений и т.п.;
- объект data scaler и разрешения для него.

Method also tries to attach current Jupyter notebook to submit. If succeed - notebook will be sent to database in «attachments» archive.

Параметры

- results_df (pandas.DataFrame) – DataFrame с решением, столбцы называются соответственно столбцам целевой переменной;
- message (str) – имя задачи.

- author_name (str) – Автор отправляемого решения.
- model_name (str) – Имя модели.
- model_version (str) – Версия модели.
- task_name (str) – Имя текущей задачи.
- feature_mining_comments (str, optional) – Text comment to explain what features were used and how do they mined from input data. (Default value = „“)
- tags (str, optional) – Дополнительные строковые теги для быстрого поиска решения в таблице результатов. (Значение по умолчанию = None)
- model (object, optional) – Модель в виде объекта. (Значение по умолчанию = None)
- model_feature_columns (list[str], optional) – Имена признаков, которые используются в модели. (Значение по умолчанию = None)
- model_feature_importances (list[float], optional) – Важность признаков. (Значение по умолчанию = None)
- model_fill_na_values (list, optional) – Значения для заполнения пропусков в данных (обязательно для моделей, которые не умеют работать с NaN). (Значение по умолчанию = None)
- data_scaler (object, optional) – Обученный скейлер. (Значение по умолчанию = None)
- data_scaler – Обученный скейлер. (Значение по умолчанию = None)
- attachments (list[str], optional) – A list with paths to files which should be included in submit. If submit is done from Jupyter notebook, it will also be added to attachments. (Default value = None)

Результат ID отправки результатов.

Тип результата bson.objectid.ObjectId

Raises

- AttachmentsNotFoundError – There is no attachments in submit and submit is done not from Jupyter notebook.
- Exception – Something wrong happened on submit.

—

Исключения (Exceptions):

AttachmentsNotFoundError	Raised if there is no attachments in submit.
AuthorNameError	Вызывается в случае, если имя автора сабмита не определено или не разрешено в текущей конфигурации.
DataFrameColumnNameError	Вызывается в случае, если имена столбцов в отправляемом DataFrame не разрешены в текущей конфигурации.
DataFrameLengthError	Вызывается в случае, если длина DataFrame, содержащего результат моделирования, не совпадает с указанным в конфигурации (параметр y_test_len).
DataFrameWidthError	Вызывается в случае, если ширина датафрейма (количество столбцов), содержащего результат моделирования, не совпадает с указанным в конфигурации (параметр y_test_columns_len).
FeatureColumnsDataTypeError	Вызывается в случае, если признаки передаются в сабмите не в виде списка.

continues on next page

Таблица 8.39 – продолжение с предыдущей страницы

FeatureColumnsLengthError	Вызывается в случае, если модель - XGBoost, но количество признаков модели не совпадает с указанным при сабмите (параметр model_feature_columns).
FeatureColumnsNotFoundError	Вызывается в случае, если при отправке результатов признаки модели не передаются явно и не могут быть получены способом по умолчанию.
FeatureImportanceDataTypeError	Вызывается в случае, если величины, обозначающие важность признаков, отправляются не в виде списка.
FeatureImportanceLengthError	Вызывается в случае, если количество признаков не равно количеству числовых значений, обозначающих важность признаков.
ForbiddenTaskNameError	Вызывается в случае, если имя задачи (параметр **message**) не входит в список разрешенных имен (указывается в конфигурации).
NaNValuesDataTypeError	Вызывается в случае, если значения для заполнения пустых полей (имеющих неопределенное значение), переданы не в виде списка (class list).
NaNValuesLengthError	Вызывается в случае, если количество признаков не соответствует количеству значений для заполнения неопределенных полей (NA).
NaNValuesNotFilledError	Вызывается в случае, если для модели необходимы, но не указаны параметры заполнения неопределенных значений.
NaNValuesTargetError	Вызывается в случае, если в отправляемом датафрейме с результатами моделирования содержатся пустые (неопределенные) значения, и это не разрешено в конфигурации.
ScalerNotAllowedError	Вызывается в случае, если скейлер объявлен, но не входит в список разрешенных для использования.
SubmitterError	Базовый родительский класс для специальных исключений модуля submitter.
TaskNameError	Вызывается в случае, если обнаружено некорректное имя задачи.

evaluation_tools.submitter.AttachmentsNotFoundError

class evaluation_tools.submitter.AttachmentsNotFoundError
Raised if there is no attachments in submit.

evaluation_tools.submitter.AuthorNameError

class evaluation_tools.submitter.AuthorNameError
Raised if author name isn't specified or author is not permitted for submit.

evaluation_tools.submitter.DataFrameColumnNameError

class evaluation_tools.submitter.DataFrameColumnNameError
Raised if names of columns in submitted dataframe not in allowed columns list.

evaluation_tools.submitter.DataFrameLengthError

class evaluation_tools.submitter.DataFrameLengthError
Raised if length (number of lines) on submitted DataFrame with result (predicted target variable) does not match with required (y_test_len parameter in configuration).

evaluation_tools.submitter.DataFrameWidthError

class evaluation_tools.submitter.DataFrameWidthError
Raised if number of columns in submitted DataFrame with result (predicted target variable) does not match

with required (y_test_columns_len parameter in configuration).

evaluation_tools.submitter.FeatureColumnsDataTypeError

class evaluation_tools.submitter.FeatureColumnsDataTypeError
Raised if model feature columns not submitted as a list.

evaluation_tools.submitter.FeatureColumnsLengthError

class evaluation_tools.submitter.FeatureColumnsLengthError
Raised if your model is a XGBoost model but the number of feature columns does not match the one specified in submit (model_feature_columns parameter).

evaluation_tools.submitter.FeatureColumnsNotFoundError

class evaluation_tools.submitter.FeatureColumnsNotFoundError
Raised if model feature columns are not specified and cannot be obtained from default values.

evaluation_tools.submitter.FeatureImportanceDataTypeError

class evaluation_tools.submitter.FeatureImportanceDataTypeError
Raised if model feature importances not submitted as a list.

evaluation_tools.submitter.FeatureImportanceLengthError

class evaluation_tools.submitter.FeatureImportanceLengthError
Raised if numbers of model feature columns and feature importances are not equal.

evaluation_tools.submitter.ForbiddenTaskNameError

class evaluation_tools.submitter.ForbiddenTaskNameError
Raised if task name (**message** parameter) aren't in whitelist of messages (describes in configuration).

evaluation_tools.submitter.NaNValuesDataTypeError

class evaluation_tools.submitter.NaNValuesDataTypeError
Raised if values for filling NA are not submitted as a list.

evaluation_tools.submitter.NaNValuesLengthError

class evaluation_tools.submitter.NaNValuesLengthError
Raised if number of model feature columns does not match to number of values for filling NA.

evaluation_tools.submitter.NaNValuesNotFilledError

class evaluation_tools.submitter.NaNValuesNotFilledError
Raised if values for filling NA are required for model but aren't specified.

evaluation_tools.submitter.NaNValuesTargetError

class evaluation_tools.submitter.NaNValuesTargetError
Raised if submitted dataframe with target contents NA values but they are not allowed.

evaluation_tools.submitter.ScalerNotAllowedError

class evaluation_tools.submitter.ScalerNotAllowedError
Raised if data scaler is defined but isn't in list of allowed or default scalers.

Read about allowed by default scalers in Submitter page. Additionally allowed scalers are specified in Submitter configuration.

evaluation_tools.submitter.SubmitterError

class evaluation_tools.submitter.SubmitterError
Base parent class for submitter module exceptions.

evaluation_tools.submitter.TaskNameError

class evaluation_tools.submitter.TaskNameError
Raised if name of task is incorrect.

8.1.7 evaluation_tools.target_loader

Содержит класс для загрузки целевой переменной из хранилища.

TargetLoader([config_name, mongo_config, . . .])	Класс для загрузки целевой переменной.

evaluation_tools.target_loader.TargetLoader

class evaluation_tools.target_loader.TargetLoader(*config_name=None*, *mongo_config=None*, *tool_config=None*, *logger=None*, *verbose=False*, *config_not_found_error=True*, *use_config_from_mongo_directly=False*, ***kwargs*)
Класс для загрузки целевой переменной.

Параметры

- config_name (str, optional) – Name of the tool configuration.
- mongo_config (dict, optional) – Configuration for database with parameters: host, port, db. In case of None, connection settings will be obtained from ProjectContextManager. (Default value = None)
- tool_config (dict, optional) – is being used as config of tool is specified instead of getting via config_db_connector.ConfigConnector. (Default value = None)
- logger (logging.Logger, optional) – Logger object from logging module. (Default value = None)
- verbose (bool, optional) – Controls the verbosity when requesting features. (Default value = True)

- config_not_found_error (bool, optional) – This flag blocks raising error when tool config not found. This is using for tests and tools without config. (Default value = True)

Примеры

```
>>> import target_loader
>>>
>>>
>>> tl = target_loader.TargetLoader(config_name='some_formulated_task_name')
>>> train, test = tl.get_train_test()
```

Methods

get_config_cache_key()	Данная функция создает словарь из конфигурации объекта, полностью идентифицирующий данный объект.
get_config_storage_name()	Overrided parent method
get_train_test()	Возвращает учебную и тестовую выборки для текущей постановки задачи.
mask_target_from_object(obj)	Функция для удаления поля целевой переменной из объекта предметной области.
rebuild_target()	Drop target cash

get_config_cache_key ()
Данная функция создает словарь из конфигурации объекта, полностью идентифицирующий данный объект.

Результат Словарь, полностью идентифицирующий объект target_loader.

Тип результата dict

get_config_storage_name ()
Overrided parent method

Результат Name of the tool's config storage name. In this case - mongo collection name.

Тип результата str

get_train_test ()
Возвращает учебную и тестовую выборки для текущей постановки задачи.

Результат Список с обучающей и тестовой выборкой. Выборки имеют тип pandas DataFrame. Тестовая выборка не содержит значений целевой переменной.

Тип результата list

mask_target_from_object (*obj*)
Функция для удаления поля целевой переменной из объекта предметной области.

Параметры obj (type(obj)) – Объект предметной области, являющийся главным в текущей задаче и содержащий целевую переменную.

Результат Измененный объект, не содержащий целевой переменной.

Тип результата type(obj)

rebuild_target ()
Drop target cash

—

Исключения (Exceptions):

TargetLoaderError	Базовый класс исключений для модуля target_loader.

evaluation_tools.target_loader.TargetLoaderError

class evaluation_tools.target_loader.TargetLoaderError
Base exception for target_loader module.

8.1.8 evaluation_tools.task_loader

Содержит класс TaskLoader для инициализации экземпляров Data Science инструментов платформы с помощью единственного вызова с указанием конфига и имени задачи.

TaskLoader([config_name, mongo_config, . . .])	Данный класс позволяет вам создавать экземпляры инструментов DATASKAI для анализа данных с помощью одной общей конфигурации.

evaluation_tools.task_loader.TaskLoader

class evaluation_tools.task_loader.TaskLoader(*config_name=None*, *mongo_config=None*, *tool_config=None*, *logger=None*, *verbose=False*, *config_not_found_error=True*, *use_config_from_mongo_directly=False*, ***kwargs*)
Данный класс позволяет вам создавать экземпляры инструментов DATASKAI для анализа данных с помощью одной общей конфигурации.

Параметры

- config_name (str, optional) – Имя конфигурации.
- mongo_config (dict, optional) – Configuration for database with parameters: host, port, db. In case of None, connection settings will be obtained from ProjectContextManager. (Default value = None)
- tool_config (dict, optional) – is being used as config of tool is specified instead of getting via config_db_connector.ConfigConnector. (Default value = None)
- logger (logging.Logger, optional) – Logger object from logging module. (Default value = None)
- verbose (bool, optional) – Controls the verbosity when requesting features. (Default value = True)
- config_not_found_error (bool, optional) – This flag blocks raising error when tool config not found. This is using for tests and tools without config. (Default value = True)
- subtool_config_from_context (bool) – Flag, which shows that DS-tools configs should be gotten from Project Context. This parameter should be transmitted via kwargs.

Примеры

```
from task_loader import TaskLoader


config = {
    "config_name" : "task_name",
    "mongo_config" : {
        "host" : "192.168.1.1",
        "port" : 27017,
        "db" : "test_db"
    },
    "description_as_markdown" : {
        "en" : "some markdown task description in english",
        "ru" : "some markdown task description in russian"
    },
    "autofill_parameters" : [
        {
            "function" : "self.submitter.submit_results",
            "parameter_name" : "task_name",
            "parameter_value" : "default task name"
        }
    ],
    "used_config_names" : {
        "submitter.Submitter" : {
            'field_names': ['submitter1', 'submitter2', 'submitter3']
            "config_names" : ["submitter_config_name1", "submitter_config_name2",
↪"submitter_config_name3"]
        },
        "target_loader.TargetLoader" : {
            'field_names': ['target_loader1', 'target_loader2']
            "config_names" : ["target_loader_config_name1", "target_loader_config_name2"]
        },
        "feature_loader.FeatureLoader" : {
            'field_names': ['feature_loader']
            "config_names" : ["feature_loader_config_name"]
        },
        "raw_data_loader.RawDataLoader" : {
            'field_names': ['raw_data_loader1', 'raw_data_loader2']
            "config_names" : ["raw_data_loader_config_name1", "raw_data_loader_config_
↪name2"]
        }
        "task_loader.TaskLoader" : {
            'field_names': ['task_loader2']
            "config_names" : ["task_loader_config_name2"]
        }
    }
}

task_l = TaskLoader(config_name='raw_data__aero__fw_classification_v1_with_fstrings')
task_l2 = TaskLoader(config_name='task_name', tool_config=config)

tl = task_l.target_loader
fl = task_l.feature_loader
submitter = task_l.submitter
new_task_loader = task_l.task_loader
```

Methods

display_description_in_jupyter([language])	Отображает описание текущей задачи в Jupyter notebook.
get_config_storage_name()	Overrided parent method
list_task_description_languages()	Выводит список языков, для которых доступно описание задачи.

display_description_in_jupyter (*language='en'*)

Отображает описание текущей задачи в Jupyter notebook.

The descriptions are stored in «markdown» format. The description may contain templates which will be replaced with corresponding values from task configuration json (config). Example of template:

{fieldname.fieldname_1[0].fieldname_2},

where fieldname_1 is an array.

If field name contains dot (.), it should be escaped by «» :

{field\.name.fieldname_1}.

Примеры

```
>>>task_loader = TaskLoader(task_name, mongo_config)
>>>task_loader.display_description_in_jupyter()
Text of task description loaded from config and converted from markdown
with replaced templates (in Jupyter Notebook).
```

This templates are being applied during config loading.

Параметры language (str, optional) – Код языка, на котором будет запрошено описание задачи. Вероятно доступны: «en», «ru». (Значение по умолчанию = „en“)

get_config_storage_name ()

Overrided parent method

Результат Name of the tool's config storage name. In this case - mongo collection name.

Тип результата str

list_task_description_languages ()

Выводит список языков, для которых доступно описание задачи.

Выводит список кодов языков, для которых доступно описание задачи. Код языка используются при вызове метода display_description_in_jupyter как параметр.

Результат Список доступных языков для получения описания задачи.

Тип результата list[str]

Raises TaskLoaderDescriptionNotExistsError – Raised if task has no description.

—

Исключения (Exceptions):

TaskLoaderConfigAbsenceError	Raised if no config data is specified during initialization
TaskLoaderDescriptionLanguageNotExistsError	Возникает, если для данного языка нет описания задачи в базе данных.
TaskLoaderDescriptionNotExistsError	Возникает, если для данной задачи нет описания в базе данных.
TaskLoaderError	Класс, который используется в качестве базового для всех исключений модуля task_loader .

evaluation_tools.task_loader.TaskLoaderConfigAbsenceError

class evaluation_tools.task_loader.TaskLoaderConfigAbsenceError
Raised if no config data is specified during initialization

evaluation_tools.task_loader.TaskLoaderDescriptionLanguageNotExistsError

class evaluation_tools.task_loader.TaskLoaderDescriptionLanguageNotExistsError
Возникает, если для данного языка нет описания задачи в базе данных.

evaluation_tools.task_loader.TaskLoaderDescriptionNotExistsError

class evaluation_tools.task_loader.TaskLoaderDescriptionNotExistsError
Возникает, если для данной задачи нет описания в базе данных.

evaluation_tools.task_loader.TaskLoaderError

class evaluation_tools.task_loader.TaskLoaderError
This is a base class for all exceptions within the task_loader module.

8.1.9 evaluation_tools.utils

Модуль содержит дополнительные DS-инструменты.

AdversarialTrainTestValidator(classifier[, . . .])	Этот класс позволяет проверить различия между тренировочной и тестовой выборкой.

evaluation_tools.utils.AdversarialTrainTestValidator

class evaluation_tools.utils.AdversarialTrainTestValidator (*classifier*, *scoring='roc_auc'*, *threshold=0.7*, *stopping_rounds=5*, *cv=None*, *verbose=0*, *n_jobs=None*)
Этот класс позволяет проверить различия между тренировочной и тестовой выборкой.

Он также предоставляет список признаков, которые могут быть удалены для того чтобы тренировочная и тестовая выборка стали похожи.

Параметры

- classifier (object) – Объект, который используется для fit данных и предоставляет атрибуты ‘‘ coef_‘‘ или ‘‘ feature_importances_‘‘.
- scoring (str or callable or list or tuple or dict or None) – Пороговое значение метрики, когда мы можем сказать, что тренировочная выборка отличается от тестовой (по умолчанию: “roc_auc”).
- threshold (float) – Пороговое значение метрики, когда мы можем сказать, что тренировочная выборка отличается от тестовой (по умолчанию: “roc_auc”).
- stopping_rounds (int) – Количество раундов, когда текущие значение метрики должено быть меньше порога, чтобы остановить итерации (по умолчанию: 5).
- cv (int or Iterator) – Определяет стратегию разделения кросс-валидация (необяза-

тельно).

- verbose (int) – Флаг управления сообщениями отладки (необязательно, по умолчанию: 0).
- n_jobs (int or None) – Количество ядер для параллельной проверки при кросс-валидации (необязательно, по умолчанию: None).

removed_features
Удаленные признаки, которые были удалены для достижения порогового значения.

Type list

remained_features
Все признаки из предоставленной выборки, кроме remove_features.

Type list

steps
Информация о каждом шаге итерации, возвращает список словарей с ключами: feature_to_remove, remove_features, remained_features, score, coefs.

Type list[dict]

Примеры

```
classifier = ensemble.RandomForestClassifier(
    n_jobs=60,
    max_depth=2,
    n_estimators=5,
    random_state=SEED
)
splitter = model_selection.StratifiedKFold(
    n_splits=5,
    shuffle=True,
    random_state=0,
)
validator = AdversarialTrainTestValidator(
    classifier,
    scoring='roc_auc',
    cv=splitter,
    threshold=0.7,
    n_jobs=60,
    verbose=10,
)
validator.validate(x_train, x_test)
```

Методы

validate(x_train, x_test)	Запуск процесса проверки.

validate (*x_train*, *x_test*)
Запуск процесса проверки.

Параметры

- x_train (pandas.DataFrame) – Обучающая выборка.
- x_test (pandas.DataFrame) – Тестовая выборка.

Результат True, если обучающая и тестовая выборки похожи, иначе False.

Тип результата bool

—

—

8.1.10 evaluation_tools.pipeline.feature_union

The module wraps sklearn.pipeline.FeatureUnion for working with pandas.DataFrame.

FeatureUnion(transformer_list[, n_jobs, . . .])	Wraps sklearn.pipeline.FeatureUnion for working with pandas.DataFrame.

evaluation_tools.pipeline.feature_union.FeatureUnion

class evaluation_tools.pipeline.feature_union.FeatureUnion (*transformer_list*, *n_jobs=None*, *transformer_weights=None*)

Wraps sklearn.pipeline.FeatureUnion for working with pandas.DataFrame.

It has the same interfaces and the same behavior except it returns pandas.DataFrame for transform and fit_transform.

Examples

```
import pandas as pd
from sklearn.preprocessing import *
from evaluation_tools.pipeline import make_union, wrap_transformer

pipeline = make_union(
    wrap_transformer(QuantileTransformer(n_quantiles=3)),
    wrap_transformer(PolynomialFeatures(include_bias=False)),
    wrap_transformer(OneHotEncoder(categories='auto', sparse=False)),
)

train = pd.DataFrame([[1], [2], [3]], columns=['feature_name'])
pipeline.fit_transform(train).columns.tolist() # => ['quantiletransformer__feature_name',
                                                    'polynomialfeatures__x0',
↪'polynomialfeatures__x0^2',
                                                    'onehotencoder__x0_1', 'onehotencoder_
↪_x0_2', 'onehotencoder__x0_3']
```

Методы

fit_transform(x[, y])	Wraps sklearn.pipeline.FeatureUnion.fit_transform for working with pandas.DataFrame.
get_feature_names()	Gets feature names from all transformers after fit.
transform(x)	Wraps sklearn.pipeline.FeatureUnion.transform for working with pandas.DataFrame.

fit_transform (*x*, *y=None*, ***fit_params*)

Wraps sklearn.pipeline.FeatureUnion.fit_transform for working with pandas.DataFrame.

get_feature_names ()

Gets feature names from all transformers after fit. If a feature with the same name is in several

transformers, then a prefix with the name of the transformer is added. If a feature with the same name is in one transformer then method call makes a warning.

Результат Names of the features produced by transform.

Тип результата list[str]

transform (*x*)
Wraps sklearn.pipeline.FeatureUnion.transform for working with pandas.DataFrame.

make_union(*transformers, **kwargs)	Wraps sklearn.pipeline.make_union for working with pandas.DataFrame.

evaluation_tools.pipeline.feature_union.make_union

evaluation_tools.pipeline.feature_union.make_union (**transformers*, ***kwargs*)
Wraps sklearn.pipeline.make_union for working with pandas.DataFrame.

8.1.11 evaluation_tools.pipeline.pipeline

The module wraps sklearn.pipeline.Pipeline for working with pandas.DataFrame.

Pipeline(steps[, memory])	Wraps sklearn.pipeline.Pipeline for working with pandas.DataFrame.

evaluation_tools.pipeline.pipeline.Pipeline

class evaluation_tools.pipeline.pipeline.Pipeline (*steps*, *memory=None*)
Wraps sklearn.pipeline.Pipeline for working with pandas.DataFrame.

It has the same interfaces and the same behavior except it returns pandas.DataFrame for transform and fit_transform.

Examples

```
import pandas as pd
from sklearn.preprocessing import *
from evaluation_tools.pipeline import make_pipeline, wrap_transformer

pipeline = make_pipeline(
    wrap_transformer(PolynomialFeatures(include_bias=False)),
    wrap_transformer(RobustScaler()),
)
train = pd.DataFrame([[1], [2], [3]], columns=['feature_name'])
pipeline.fit_transform(train).columns.tolist() # => ['x0', 'x0^2']
```

Методы и атрибуты

fit_transform(x[, y])	Wraps sklearn.pipeline.Pipeline.fit_transform for working with pandas.DataFrame.
get_feature_names()	Gets feature names from last transformers after fit.

continues on next page

Таблица 8.57 – продолжение с предыдущей страницы

transform(x)	Wraps sklearn.pipeline.Pipeline.transform for working with pandas.DataFrame.

fit_transform (*x*, *y=None*, ***fit_params*)
Wraps sklearn.pipeline.Pipeline.fit_transform for working with pandas.DataFrame.

get_feature_names ()
Gets feature names from last transformers after fit.

Результат Names of the features produced by transform.

Тип результата list[str]

transform (*x*)
Wraps sklearn.pipeline.Pipeline.transform for working with pandas.DataFrame.

make_pipeline(*steps, **kwargs)	Wraps sklearn.pipeline.make_pipeline for working with pandas.DataFrame.

evaluation_tools.pipeline.pipeline.make_pipeline

evaluation_tools.pipeline.pipeline.make_pipeline (**steps*, ***kwargs*)
Wraps sklearn.pipeline.make_pipeline for working with pandas.DataFrame.

8.1.12 evaluation_tools.pipeline.utils

A set of utils for easy working with pipelines.

wrap_transformer(transformer[, ...])	The class (who looks like a method) allow wrapping sklearn transformer for working with pandas.DataFrame.

evaluation_tools.pipeline.utils.wrap_transformer

class evaluation_tools.pipeline.utils.wrap_transformer (*transformer*, *auto_fill_feature_names=False*, *feature_names=None*)
The class (who looks like a method) allow wrapping sklearn transformer for working with pandas.DataFrame.

Примеры

```
import pandas as pd
from sklearn.preprocessing import *
from evaluation_tools.pipeline import make_pipeline

transfomer = wrap_transformer(RobustScaler())
train = pd.DataFrame([[1], [2], [3]], columns=['feature_name'])
transfomer.fit_transform(train).columns.tolist() # => ['feature_name']
```

Параметры

- transformer (object) – Transformer with implemented fit, transform methods.
- auto_fill_feature_names (bool) – Flag for auto-filling feature names:

<TRANSFORMER_NAME>_column_<NUMBER>.

- feature_names (array[str]) – Feature names.

Результат The return value. True for success, False otherwise.

Тип результата bool

Методы

fit(x[, y])	Fit the transfomer and store feature names.
fit_transform(x[, y])	Fit and transform the transfomer and store feature names
get_feature_names()	Get feature names from last transformers after fit.
get_params(**kwargs)	Get parameters for this estimator.
set_params(transformer[, . . .])	Set parameters for this estimator.
transform(x)	Transform the transfomer and store feature names

fit (*x*, *y=None*, ***fit_params*)
Fit the transfomer and store feature names.

Параметры

- x (iterable) – Training data. Must fulfill input requirements of first step of the pipeline.
- y (iterable) – Training targets. Must fulfill label requirements for all steps of the pipeline. Default None.
- **fit_params (dict[str]) – Parameters passed to the fit method of each step, where each parameter name is prefixed such that parameter p for step s has key s__p.

Результат This estimator (self).

Тип результата object

fit_transform (*x*, *y=None*, ***fit_params*)
Fit and transform the transfomer and store feature names

Параметры

- x (iterable) – Training data. Must fulfill input requirements of first step of the pipeline.
- y (iterable) – Training targets. Must fulfill label requirements for all steps of the pipeline. Default None.
- **fit_params (dict[str]) – Parameters passed to the fit method of each step, where each parameter name is prefixed such that parameter p for step s has key s__p.

Результат Transformed samples.

Тип результата pandas.DataFrame

get_feature_names ()
Get feature names from last transformers after fit.

Результат Names of the features produced by transform.

Тип результата list[str]

get_params (***kwargs*)
Get parameters for this estimator.

Параметры **kwargs (dict[obj]) – Not used. For backward compatibility.

Результат Params with values.

Тип результата dict[str]

set_params (*transformer*, *auto_fill_feature_names=False*)
Set parameters for this estimator.

Параметры

- transformer (object) – Transformer with implemented fit, transform methods.
- auto_fill_feature_names (bool) – Flag for auto-filling feature names: <TRANSFORMER_NAME>_column_<NUMBER>.

Результат This estimator (self).

Тип результата object

transform (*x*)
Transform the transfomer and store feature names

Параметры x (iterable) – Training data. Must fulfill input requirements of first step of the pipeline.

Результат Transformed samples.

Тип результата pandas.DataFrame

extract_params(pipeline[, excluded_class_names])	Extract params from transformers.
flat_transformers(pipeline)	Flat transformers.

evaluation_tools.pipeline.utils.extract_params

evaluation_tools.pipeline.utils.extract_params (*pipeline*, *excluded_class_names=None*)
Extract params from transformers.

Параметры

- pipeline (object) – Pipeline.
- excluded_class_names (array[str]) – List of excluded transformers.

Результат Params for transformers.

Тип результата array[str]

evaluation_tools.pipeline.utils.flat_transformers

evaluation_tools.pipeline.utils.flat_transformers (*pipeline*)
Flat transformers.

Параметры pipeline (object) – Pipeline.

Результат Flattened transformers.

Тип результата array

—

8.2 Дополнительные инструменты для разработчиков

8.2.1 evaluation_tools.config_cache_manager

Модуль содержит инструменты для кэширования результатов работы долго выполняющихся методов. Вы можете использовать его для методов менеджеров данных, таких как TargetLoader, FeatureLoader и т.д.

ArcticWorker()	Class contains methods to work with cache using Arctic and mongo database.
CachedConfigManager()	Класс содержит инструменты для кэширования.
SubjectDomainCacheWorker()	Class contains methods to work with cache using Redis database.

evaluation_tools.config_cache_manager.ArcticWorker

class evaluation_tools.config_cache_manager.ArcticWorker

Class contains methods to work with cache using Arctic and mongo database.

Methods

set_config_connector(config_connector)	Method to set config connector to infer connection settings to config database.

set_config_connector (*config_connector*)

Method to set config connector to infer connection settings to config database.

Параметры config_connector (helpers.config_db_connector.ConfigConnector) – config connector object to infer connections

evaluation_tools.config_cache_manager.CachedConfigManager

class evaluation_tools.config_cache_manager.CachedConfigManager

Класс содержит инструменты для кэширования.

It is used as an ancestor for data managers (TargetLoader and etc.). To cache the work of some method you should apply the cached_call decorator. The result of the method work will be saved in a cache storage.

Ключ кэша состоит из трёх частей: конфигурация менеджера, работающего с данными, имя декорируемого метода и аргументы, которые использует метод.

- **manager configuration**: all manager state is defined by it's configuration, to be precise - some subset of configuration fields. This fields and their values should be filtered in according object method get_config_cache_key and returned as *dict*;
- **method name**: имя декорируемого метода;
- **''args'' and ''kwargs'' of called method**. kwargs is flatted, sorted and then converted to strings with default separator. Args is converted to strings and joined with default separator. Then two strings is concatted together and forms last part of cache key.

Все три части ключа объединяются и передаются в функцию hashlib.md5.hexdigest(). Результат работы hexdigest() становится ключом в хранилище кэшированных данных.

Примеры

```
import time
import config_cache_manager


class UserDataManager(config_cache_manager.CachedConfigManager):
    """User data manager."""
    def __init__(self):
        self._config = {
            'config_name' : 'default_congig2',
            'config_version' : '0.1',
            'cache_mongodb_address': '10.30.16.181:27017'}
        super(UserDataManager, self).__init__(self._config['cache_mongodb_address'])
        self._long_run_func = config_cache_manager.mongo_cached_call(self._long_run_func)

    def _long_run_func(self, cycles):
        """Decorated method."""
        s = 0
        for i in range(cycles):
            for j in range(cycles):
                s += i*j
        return s

    def get_config_cache_key(self):
        """Implementation of CachedConfigManager.get_config_cache_key abstract method."""
        cache_key = dict()
        cache_key['config_name'] = self._config['config_name']
        cache_key['config_version'] = self._config['config_version']
        return cache_key

    def get_result(self, cycles):
        return self._long_run_func(self, cycles)


utl = UserDataManager()
cycles = 10000

start = time.time()
utl.get_result(cycles)
print(time.time() - start)
# 10.789832830429077

start = time.time()
utl.get_result(cycles)
print(time.time() - start)
# 0.06432080268859863
```

Методы

get_config_cache_key()	Абстрактный метод для генерации части ключа из конфигурации объекта.

get_config_cache_key ()

Абстрактный метод для генерации части ключа из конфигурации объекта.

Когда метод используется объектом класса-наследника, объект должен выполнить отбор нужных параметров словаря self._config и создать из них новый словарь, который будет специфицировать поведение объекта и будет использован как часть ключа в хранилище кэшированных данных.

Must return dictionary with keys and values which specify object behavior.

evaluation_tools.config_cache_manager.SubjectDomainCacheWorker

class evaluation_tools.config_cache_manager.SubjectDomainCacheWorker
Class contains methods to work with cache using Redis database.

Methods

set_config_redis([host_port_string])	Set subject_domain_cache connection settings.

set_config_redis (*host_port_string=''*)
Set subject_domain_cache connection settings.

Параметры host_port_string (str) – host-port string separated by semicolon.

cached_call(some_func[, mongo_to_redis])	Декоратор для кэширования результатов работы долго выполняющихся методов.

evaluation_tools.config_cache_manager.cached_call

evaluation_tools.config_cache_manager.cached_call (*some_func*, *mongo_to_redis=True*, **args*, ***kwargs*)
A decorator for caching the results of long running methods work. The decorated method must be a member of CachedConfigManager inherited object.

See an example of usage on CachedConfigManager class page.

Параметры

- some_func (function) – A method of an object which should be decorated.
- mongo_to_redis (bool) – If true cache is sending from mongo to redis, vise versa otherwise.
- *args – Arguments of the method.
- **kwargs – Key-value arguments of the method.

Результат A function with caching functionality.

Тип результата function

Исключения (Exceptions):

CacheKeyNotFoundError	Вызывается в случае, если происходит попытка найти в хранилище несуществующий кэш-ключ.

evaluation_tools.config_cache_manager.CacheKeyNotFoundError

class evaluation_tools.config_cache_manager.CacheKeyNotFoundError
Raised when a non-existing cache key is requested from a storage.

Глава 9

Дополнительная информация

Построено Sphinx, git commit: 01f31f4bd8134d549f4e6afab4830fdf3d9ed15b.

Содержание модулей Python

e

m