
Nirdizati Research Documentation

Stefano Branchi, Chiara Di Francescomarino, Tõnis Kasekamp, S

Jun 14, 2019

1	Nirdizati Research backend	3
1.1	Running in a new environment	3
1.2	Docker Compose	3
1.3	Run an instance of the project	4
1.4	Advanced configuration	5
1.5	Labelling job	5
1.6	Contributors	5
2	Nirdizati Research frontend	7
2.1	Running in a new environment	7
2.2	Docker Compose	7
2.3	Run an instance of the project	7
2.4	Thanks to	8
2.5	Contributors	8
3	src	9
3.1	src package	9
	Python Module Index	81
	Index	83

Nirdizati Research backend

Master

Development

Django backend server for machine learning on event logs.

1.1 Running in a new environment

The docker build is available @ <https://hub.docker.com/r/nirdizatiresearch/predict-python/> in any case if you prefer to setup your environment on your own you can refer the [Dockerfile](#).

1.2 Docker Compose

On first run to setup the database, you can run:

```
docker-compose run server python manage.py migrate
```

To run the project:

```
docker-compose up redis server scheduler worker
```

To access a generic remote Django server you can use the ssh tunneling functionality as shown in the following sample:

```
ssh -L 8000:127.0.0.1:8000 <user>@<host>
```

1.3 Run an instance of the project

If you are familiar with docker-compose the [docker-compose](#) file is available, otherwise if you use PyCharm as IDE run the provided configurations. Finally, from the command line you can use the following sample commands to interact with our software.

Start server with

```
python manage.py runserver
```

Run tests with one of the following

```
python manage.py test
./manage.py test
```

NB: always run a redis-server in background if you want your server to accept any incoming post requests!

Start by running migrations and adding sample data

```
python manage.py migrate
python manage.py loaddata <your_file.json>
```

Start jobs from command line

```
curl --request POST \
  --header 'Content-Type: application/json' \
  --data-binary '{
    "type": "classification",
    "split_id": 1,
    "config": {
      "encodings": ["simpleIndex"],
      "clusterings": ["noCluster"],
      "methods": ["randomForest"],
      "label": {"type": "remaining_time"},
      "encoding": {"prefix_length": 3, "generation_type": "only", "padding": "zero_
↪padding"}
    }
  }' \
  http://localhost:8000/jobs/multiple
```

Creating a single split options.

- `$SPLIT_TYPE` has to be one of `split_sequential`, `split_random`, `split_temporal`, `split_strict_temporal`. By default `split_sequential`.
- `test_size` has to be from 0 to 1. By default 0.2

```
curl --request POST \
  --header 'Content-Type: application/json' \
  --data-binary '{
    "type": "single",
    "original_log": 1,
    "config": {
      "test_size": 0.2,
      "split_type": $SPLIT_TYPE
    }
  }' \
  http://localhost:8000/splits/
```


1.4 Advanced configuration

Prediction methods accept configuration for sklearn classification/regression methods. The Job config must contain a dict with only the supported options for that method. The dict name must take the format “type.method”. For classification randomForest this would be `classification.randomForest`. Advanced configuration is optional. Look at `jobs/job_creator.py` for default values. For example, the configuration for classification KNN would have to be like:

```
curl --request POST \
  --header 'Content-Type: application/json' \
  --data-binary '{
    "type": "classification",
    "split_id": 1,
    "config": {
      "encodings": ["simpleIndex"],
      "clusterings": ["noCluster"],
      "methods": ["knn"],
      "classification.knn": {
        "n_neighbors": 5,
        "weights": "uniform"
      },
      "label": {"type": "remaining_time"},
      "encoding": {"prefix_length": 3, "generation_type": "up_to", "padding": "no_
↪padding"}
    }
  }' \
  http://localhost:8000/jobs/multiple
```

1.5 Labelling job

Log encoding and labelling can be tested before prediction. It supports all the same values as classification and regression jobs but the method and clustering.

```
curl --request POST \
  --header 'Content-Type: application/json' \
  --data-binary '{
    "type": "labelling",
    "split_id": 5,
    "config": {
      "label": {"type": "remaining_time"},
      "encoding": {"prefix_length": 3, "generation_type": "up_to", "padding": "no_
↪padding"}
    }
  }' \
  http://localhost:8000/jobs/multiple
```

1.6 Contributors

- @stebranchi Stefano Branchi
- @dfmchiara Chiara Di Francescomarino
- @TKasekamp Tōnis Kasekamp

- [@mrsonuk](#) Santosh Kumar
- [@fmmaggi](#) Fabrizio Maggi
- [@WilliamsRizzi](#) Williams Rizzi
- [@HitLuca](#) Luca Simonetto

Nirdizati Research frontend

Master

Development

React frontend to perform Predictive Monitoring analysis over event logs.

2.1 Running in a new environment

The docker build is available @ <https://hub.docker.com/r/nirdizatiresearch/predict-react/> in any case if you prefer to setup your environment on your own you can refer the [Dockerfile](#).

2.2 Docker Compose

To run the project:

```
docker-compose up react-client
```

2.3 Run an instance of the project

If you are familiar with docker-compose the [docker-compose](#) file is available, otherwise if you use pycharm as IDE is available the run configuration in the [runConfiguration](#) settings.

package.json contains all supported commands for this project.

Install required components:

```
npm install
```

Run build:

```
npm run build
```

Run tests:

```
npm run test
```

Run start:

```
npm run start
```

2.4 Thanks to

This project was bootstrapped with [Create React App](#) and [Storybook](#).

2.5 Contributors

- [@stebranchi](#) Stefano Branchi
- [@dfmchiara](#) Chiara Di Francescomarino
- [@TKasekamp](#) Tönis Kasekamp
- [@mrsonuk](#) Santosh Kumar
- [@fmmaggi](#) Fabrizio Maggi
- [@WilliamsRizzi](#) Williams Rizzi
- [@HitLuca](#) Luca Simonetto

3.1 src package

3.1.1 Subpackages

src.cache package

Subpackages

Submodules

src.cache.apps module

```
class src.cache.apps.CacheConfig(app_name, app_module)
    Bases: django.apps.config.AppConfig
    name = 'src.cache'
```

src.cache.cache module

```
src.cache.cache.dump_to_cache(path, obj, prefix="")
src.cache.cache.get_digested(candidate_path)
    Return type str
src.cache.cache.get_labelled_logs(job)
    Return type (<class 'pandas.core.frame.DataFrame'>, <class 'pandas.core.frame.DataFrame'>)
src.cache.cache.get_loaded_logs(split)
```

Return type (<class 'pandas.core.frame.DataFrame'>, <class 'pandas.core.frame.DataFrame'>, <class 'pandas.core.frame.DataFrame'>)

`src.cache.cache.load_from_cache(path, prefix=)`

`src.cache.cache.put_labelled_logs(job, train_df, test_df)`

`src.cache.cache.put_loaded_logs(split, train_df, test_df, additional_columns)`

src.cache.models module

class `src.cache.models.Cache(id)`

Bases: `src.common.models.CommonModel`

exception `DoesNotExist`

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception `MultipleObjectsReturned`

Bases: `django.core.exceptions.MultipleObjectsReturned`

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

labelledlog

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

loadedlog

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

objects = <django.db.models.manager.Manager object>

class `src.cache.models.LabelledLog(id, cache_ptr, train_log_path, test_log_path, split, encoding, labelling)`

Bases: `src.cache.models.Cache`

exception `DoesNotExist`

Bases: `src.cache.models.DoesNotExist`

exception `MultipleObjectsReturned`

Bases: `src.cache.models.MultipleObjectsReturned`

cache_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

cache_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

encoding

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

encoding_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

labelling

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

labelling_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

split

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

split_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

test_log_path

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

train_log_path

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

```
class src.cache.models.LoadedLog(id, cache_ptr, train_log_path, test_log_path, additional_columns_path, split)
```

Bases: `src.cache.models.Cache`

exception DoesNotExist

Bases: `src.cache.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.cache.models.MultipleObjectsReturned`

additional_columns_path

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

cache_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

cache_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

split

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

split_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

test_log_path

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

train_log_path

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

Module contents

src.clustering package

Subpackages

Submodules

src.clustering.apps module

```
class src.clustering.apps.ClusteringConfig(app_name, app_module)
    Bases: django.apps.config.AppConfig
    name = 'src.clustering'
```

src.clustering.clustering module

clustering methods and functionalities

```
class src.clustering.clustering.Clustering(clustering)
    Bases: object

    clustering related tasks, stores both the clustered data and the models trained on each cluster

cluster_data(input_df)
    clusters the input DataFrame

    Parameters input_df (DataFrame) – input DataFrame
    Return type dict
    Returns dictionary containing the clustered data

fit(training_df)
    clusters the input DataFrame

    Parameters training_df (DataFrame) – training DataFrame
    Return type None

classmethod load_model(job)

predict(test_df)
    TODO: complete

    Parameters test_df (DataFrame) – testing DataFrame
    Return type Series
    Returns TODO: complete
```

src.clustering.methods_default_config module

```
src.clustering.methods_default_config.clustering_kmeans()
```

src.clustering.models module

```
class src.clustering.models.Clustering(*args, **kwargs)
    Bases: src.common.models.CommonModel

    Container of Classification to be shown in frontend

exception DoesNotExist
    Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
    Bases: django.core.exceptions.MultipleObjectsReturned
```

clustering_method

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_clustering_method_display (*, field=<django.db.models.fields.CharField: clustering_method>)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

static init (clustering='noCluster', configuration={})

job_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

kmeans

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

model_path

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

nocluster

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

objects = <model_utils.managers.InheritanceManager object>

to_dict ()

class src.clustering.models.ClusteringMethods

Bases: enum.Enum

An enumeration.

KMEANS = 'kmeans'

NO_CLUSTER = 'noCluster'

```
class src.clustering.models.KMeans(id, model_path, clustering_method, clustering_ptr,
                                   n_clusters, init, n_init, max_iter, tol, precompute_distances, random_state, copy_x, algorithm)
```

Bases: `src.clustering.models.Clustering`

exception DoesNotExist

Bases: `src.clustering.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.clustering.models.MultipleObjectsReturned`

algorithm

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

clustering_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

clustering_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

copy_x

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_algorithm_display (*, field=<django.db.models.fields.CharField: algorithm>)

get_init_display (*, field=<django.db.models.fields.CharField: init>)

get_precompute_distances_display (*, field=<django.db.models.fields.CharField: precompute_distances>)

max_iter

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_clusters

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_init

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

precompute_distances

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

random_state

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict ()

Return type `dict`

tol

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class `src.clustering.models.NoCluster` (*id, model_path, clustering_method, clustering_ptr*)

Bases: `src.clustering.models.Clustering`

exception `DoesNotExist`

Bases: `src.clustering.models.DoesNotExist`

exception `MultipleObjectsReturned`

Bases: `src.clustering.models.MultipleObjectsReturned`

clustering_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

clustering_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

Module contents

src.common package

Subpackages

Submodules

src.common.apps module

class `src.common.apps.CommonConfig` (*app_name, app_module*)

Bases: `django.apps.config.AppConfig`

name = `'src.common'`

src.common.models module

class `src.common.models.CommonModel` (**args, **kwargs*)

Bases: `django.db.models.base.Model`

class `Meta`

Bases: `object`

abstract = `False`

get_full_dict ()

to_dict ()

Return type `dict`

Module contents

src.core package

Subpackages

Submodules

src.core.common module

common methods used in the core package

`src.core.common.get_method_config(job)`
returns the method configuration dictionary

Parameters `job` (*Job*) – job configuration

Return type (<class 'str'>, <class 'dict'>)

Returns method string and method configuration dict

src.core.core module

Module contents

src.encoding package

Subpackages

Submodules

src.encoding.apps module

class `src.encoding.apps.EncodingConfig(app_name, app_module)`
Bases: `django.apps.config.AppConfig`

`name = 'src.encoding'`

src.encoding.boolean_frequency module

`src.encoding.boolean_frequency.boolean(log, event_names, label, encoding)`

Return type `DataFrame`

`src.encoding.boolean_frequency.frequency(log, event_names, label, encoding)`

Return type `DataFrame`

src.encoding.common module

`src.encoding.common.encode_label_log(run_log, encoding, job_type, labelling, event_names=None, additional_columns=None, fit_encoder=False)`

```
src.encoding.common.encode_label_logs(training_log, test_log, job, additional_columns=None)
```

src.encoding.complex_last_payload module

```
src.encoding.complex_last_payload.complex(log, labelling, encoding, additional_columns)
```

Return type DataFrame

```
src.encoding.complex_last_payload.last_payload(log, labelling, encoding, additional_columns)
```

Return type DataFrame

src.encoding.encoder module

```
class src.encoding.encoder.Encoder(df, encoding)
```

Bases: object

```
encode(df, encoding)
```

Return type None

src.encoding.encoding_container module

```
class src.encoding.encoding_container.EncodingContainer
```

Bases: *src.encoding.encoding_container.EncodingContainer*

Inner object describing encoding configuration.

```
static encode(df)
```

Return type None

```
static init_label_encoder(df)
```

Return type None

```
is_all_in_one()
```

Return type bool

```
is_boolean()
```

Return type bool

```
is_complex()
```

Return type bool

```
is_zero_padding()
```

Return type bool

src.encoding.encoding_parser module

```
class src.encoding.encoding_parser.DataEncoder(task, is_targets_dataset=False)
```

Bases: object

support class for EncodingParser, tasked with actual parsing/one-hot encoding

```

class DataTypes
    Bases: enum.Enum

    possible data types for each column

    CATEGORICAL = 'categorical'

    NUMERIC = 'numeric'

build_encoders (data)
    builds an encoder for each column

    first the base headers are extracted (prefix_1 -> prefix, org:resources:Amount_1 -> org_resources:Amount)
    and then a dictionary of LabelEncoders is built. Numerical data stores min and max instead of a LabelEncoder.

    Parameters data (DataFrame) – input dataframe

    Return type None

encode_data (data, train=True)
    encodes the input data

    actual data encoding, using the built encoders. For each column type the right encoding is done (to
    class/normalization)

    Parameters

        • data (DataFrame) – input dataframe

        • train (bool) – flag indicating whether the input is a train dataframe or a test one

    Return type None

get_n_classes_x ()
    returns the number of training/test classes

    returns the highest number of classes for the encoded dataframe, adding 1 if there are numerical values.
    The structure is [one-hot encoding, normalized_value] for each variable, such that a categorical variable
    becomes [0 0 0 1 0.0] where a numerical value becomes [0 0 0 0 0.263]

    Returns number of training/test classes + 1 (for numerical values)

get_numerical_limits (header='label')
    returns the numerical limits for the input header

    returns the min and max value from the stored LabelEncoders, using header as index

    Parameters header – label associated with the data we want to extract min and max from

    Returns min and max values associated with the column_header_

to_one_hot (data)
    one hot encoding

    transforms the encoded data into the one-hot representation

    Parameters data (DataFrame) – input dataframe

    Return type ndarray

    Returns one-hot encoded array

class src.encoding.encoding_parser.EncodingParser (encoding, binary_target, task)
    Bases: object

```

parses the encoded datasets into a suitable format for the keras models (0-1 float range, one-hot encodable classes etc.), plus minor utils

denormalize_predictions (*predictions*)

denormalizes the predictive_model predictions

denormalizes the predictions using the stored y min and max

Parameters **predictions** (ndarray) – predictive_model predictions

Return type ndarray

Returns denormalized predictions

get_n_classes_x ()

parse_targets (*targets*)

parses the target dataset

encodes the target dataset based on the encoding given in the init method. Stores min and max value/classes number based on the encoding :type targets: DataFrame :param targets: input dataset :rtype: ndarray :return: parsed input dataset

parse_testing_dataset (*test_data*)

parses the test dataset

encodes the test dataset based on the encoding given in the init method :type test_data: DataFrame :param test_data: input dataset :rtype: ndarray :return: parsed input dataset

parse_training_dataset (*train_data*)

parses the training dataset

encodes the training dataset based on the encoding given in the init method :type train_data: DataFrame :param train_data: input dataset :rtype: ndarray :return: parsed input dataset

src.encoding.models module

class src.encoding.models.**DataEncodings**

Bases: enum.Enum

An enumeration.

LABEL_ENCODER = 'label_encoder'

ONE_HOT_ENCODER = 'one_hot'

class src.encoding.models.**Encoding**(*id, data_encoding, value_encoding, add_elapsed_time, add_remaining_time, add_executed_events, add_resources_used, add_new_traces, features, prefix_length, padding, task_generation_type*)

Bases: [src.common.models.CommonModel](#)

exception **DoesNotExist**

Bases: django.core.exceptions.ObjectDoesNotExist

exception **MultipleObjectsReturned**

Bases: django.core.exceptions.MultipleObjectsReturned

add_elapsed_time

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

add_executed_events

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

add_new_traces

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

add_remaining_time

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

add_resources_used

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

data_encoding

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

features

A placeholder class that provides a way to set the attribute on the model.

get_data_encoding_display (*, field=<django.db.models.fields.CharField: data_encoding>)

get_task_generation_type_display (*, field=<django.db.models.fields.CharField: task_generation_type>)

get_value_encoding_display (*, field=<django.db.models.fields.CharField: value_encoding>)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

job_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

labelledlog_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

objects = <django.db.models.manager.Manager object>

padding

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

prefix_length

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

task_generation_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

Return type dict

value_encoding

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class src.encoding.models.TaskGenerationTypes

Bases: enum.Enum

An enumeration.

ALL_IN_ONE = 'all_in_one'

ONLY_THIS = 'only'

UP_TO = 'up_to'

class src.encoding.models.ValueEncodings

Bases: enum.Enum

An enumeration.

BOOLEAN = 'boolean'

COMPLEX = 'complex'

FREQUENCY = 'frequency'

LAST_PAYLOAD = 'lastPayload'

SIMPLE_INDEX = 'simpleIndex'

src.encoding.simple_index module

```
src.encoding.simple_index.add_trace_row(trace, encoding, labelling, event_index, col-  
                                         umn_len, attribute_classifier=None, exe-  
                                         cuted_events=None, resources_used=None,  
                                         new_traces=None)
```

Row in data frame

```
src.encoding.simple_index.simple_index(log, labelling, encoding)
```

Return type DataFrame

Module contents**src.evaluation package**

Subpackages

Submodules

src.evaluation.apps module

```
class src.evaluation.apps.EvaluationConfig(app_name, app_module)
    Bases: django.apps.config.AppConfig
    name = 'src.evaluation'
```

src.evaluation.models module

```
class src.evaluation.models.BinaryClassificationMetrics(id, elapsed_time, eval-
    uation_ptr, f1_score,
    accuracy, precision,
    recall, classification-
    metrics_ptr, true_positive,
    true_negative,
    false_negative,
    false_positive, auc)
    Bases: src.evaluation.models.ClassificationMetrics
```

exception DoesNotExist

Bases: src.evaluation.models.DoesNotExist

exception MultipleObjectsReturned

Bases: src.evaluation.models.MultipleObjectsReturned

auc

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

classificationmetrics_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Restaurant.place is a ForwardOneToOneDescriptor instance.

classificationmetrics_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

false_negative

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

false_positive

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

Return type dict

true_negative

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

true_positive

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

```
class src.evaluation.models.ClassificationMetrics (id, elapsed_time, evaluation_ptr,  
fl_score, accuracy, precision, re-  
call)
```

Bases: `src.evaluation.models.Evaluation`

exception DoesNotExist

Bases: `src.evaluation.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.evaluation.models.MultipleObjectsReturned`

accuracy

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

binaryclassificationmetrics

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

evaluation_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

evaluation_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

f1_score

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

multiclassclassificationmetrics

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

precision

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

recall

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

Return type dict

class `src.evaluation.models.Evaluation(id, elapsed_time)`

Bases: `src.common.models.CommonModel`

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

classificationmetrics

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

elapsed_time

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

static init (*prediction_type, results, binary=False*)

job_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

objects = <model_utils.managers.InheritanceManager object>

regressionmetrics

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

timeseriespredictionmetrics

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

to_dict()

Return type `dict`

```
class src.evaluation.models.MulticlassClassificationMetrics (id, elapsed_time,
                                                             evaluation_ptr,
                                                             fl_score, accuracy,
                                                             precision, recall, classification-
                                                             metrics_ptr)
```

Bases: `src.evaluation.models.ClassificationMetrics`

exception DoesNotExist

Bases: `src.evaluation.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.evaluation.models.MultipleObjectsReturned`

classificationmetrics_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

classificationmetrics_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

```
class src.evaluation.models.RegressionMetrics (id, elapsed_time, evaluation_ptr, rmse,
                                                mae, rscore, mape)
```

Bases: `src.evaluation.models.Evaluation`

exception DoesNotExist

Bases: `src.evaluation.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.evaluation.models.MultipleObjectsReturned`

evaluation_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

evaluation_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

mae

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

mape

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

rmse

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

rscore

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict ()

Return type dict

class `src.evaluation.models.TimeSeriesPredictionMetrics` (*id, elapsed_time, evaluation_ptr, nlevenshtein*)

Bases: `src.evaluation.models.Evaluation`

exception DoesNotExist

Bases: `src.evaluation.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.evaluation.models.MultipleObjectsReturned`

evaluation_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

evaluation_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

nlevenshtein

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict ()

Return type dict

Module contents**src.hyperparameter_optimization package**

Subpackages

Submodules

src.hyperparameter_optimization.apps module

```
class src.hyperparameter_optimization.apps.HyperparameterOptimizationConfig (app_name,  
                                                                           app_module)  
    Bases: django.apps.config.AppConfig  
    name = 'src.hyperparameter_optimization'
```

src.hyperparameter_optimization.hyperopt_spaces module

hyperopt search spaces for each prediction method

src.hyperparameter_optimization.hyperopt_wrapper module

src.hyperparameter_optimization.methods_default_config module

```
src.hyperparameter_optimization.methods_default_config.hyperparameter_optimization_hyperopt
```

src.hyperparameter_optimization.models module

```
class src.hyperparameter_optimization.models.HyperOpt (id,      optimization_method,  
                                                         hyperparameteroptimiza-  
                                                         tion_ptr,  max_evaluations,  
                                                         performance_metric,  algo-  
                                                         rithm_type)  
    Bases: src.hyperparameter_optimization.models.HyperparameterOptimization  
exception DoesNotExist  
    Bases: src.hyperparameter_optimization.models.DoesNotExist  
exception MultipleObjectsReturned  
    Bases: src.hyperparameter_optimization.models.MultipleObjectsReturned  
algorithm_type  
    A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is  
    executed.  
get_algorithm_type_display (*, field=<django.db.models.fields.CharField: algorithm_type>)  
get_performance_metric_display (*, field=<django.db.models.fields.CharField: perfor-  
                                     mance_metric>)  
hyperparameteroptimization_ptr  
    Accessor to the related object on the forward side of a one-to-one relation.  
    In the example:
```

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

Restaurant.place is a ForwardOneToOneDescriptor instance.

hyperparameteroptimization_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

max_evaluations

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

performance_metric

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

```
class src.hyperparameter_optimization.models.HyperOptAlgorithms
```

Bases: `enum.Enum`

An enumeration.

```
RANDOM_SEARCH = 'random_search'
```

```
TPE = 'tpe'
```

```
class src.hyperparameter_optimization.models.HyperOptLosses
```

Bases: `enum.Enum`

An enumeration.

```
ACC = 'acc'
```

```
AUC = 'auc'
```

```
F1SCORE = 'f1score'
```

```
FALSE_NEGATIVE = 'false_negative'
```

```
FALSE_POSITIVE = 'false_positive'
```

```
MAE = 'mae'
```

```
MAPE = 'mape'
```

```
PRECISION = 'precision'
```

```
RECALL = 'recall'
```

```
RMSE = 'rmse'
```

```
RSCORE = 'rscore'
```

```
TRUE_NEGATIVE = 'true_negative'
```

```
TRUE_POSITIVE = 'true_positive'
```

```
class src.hyperparameter_optimization.models.HyperparameterOptimization(id,  
op-  
ti-  
miza-  
tion_method)
```

Bases: `src.common.models.CommonModel`

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

get_optimization_method_display (*, field=<django.db.models.fields.CharField: optimization_method>)

hyperopt

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

static init (configuration={'type': 'hyperopt'})

job_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child (Model):  
    parent = ForeignKey (Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

objects = <model_utils.managers.InheritanceManager object>

optimization_method

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class src.hyperparameter_optimization.models.HyperparameterOptimizationMethods

Bases: enum.Enum

An enumeration.

HYPEROPT = 'hyperopt'

NONE = 'none'

Module contents

src.jobs package

Subpackages

Submodules

src.jobs.admin module

src.jobs.apps module

```
class src.jobs.apps.JobsConfig(app_name, app_module)
    Bases: django.apps.config.AppConfig
    name = 'src.jobs'
```

src.jobs.job_creator module

```
src.jobs.job_creator.generate(split, payload)
src.jobs.job_creator.generate_labelling(split, payload)
src.jobs.job_creator.get_prediction_method_config(predictive_model, prediction_method, payload)
src.jobs.job_creator.update(split, payload, generation_type='classification')
```

src.jobs.json_renderer module

```
class src.jobs.json_renderer.CustomJSONEncoder(*, skipkeys=False, ensure_ascii=True,
    check_circular=True, allow_nan=True,
    sort_keys=False, indent=None, separators=None, default=None)
    Bases: rest_framework.utils.encoders.JSONEncoder
```

```
iterencode(o, _one_shot=False)
    Encode the given object and yield each string representation as available.
    For example:
```

```
for chunk in JSONEncoder().iterencode(bigobject):
    mysocket.write(chunk)
```

```
class src.jobs.json_renderer.PalJSONRenderer
    Bases: rest_framework.renderers.JSONRenderer
    encoder_class
        alias of CustomJSONEncoder
```

src.jobs.models module

```
class src.jobs.models.Job(id, created_date, modified_date, error, status, type, create_models, split,
    encoding, labelling, clustering, predictive_model, evaluation, hyperparameter_optimizer, incremental_train)
    Bases: src.common.models.CommonModel
```

```
exception DoesNotExist
    Bases: django.core.exceptions.ObjectDoesNotExist
```

```
exception MultipleObjectsReturned
    Bases: django.core.exceptions.MultipleObjectsReturned
```

```
base_model
    Accessor to the related objects manager on the reverse side of a many-to-one relation.
    In the example:
```

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

clustering

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

clustering_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

create_models

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

encoding

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

encoding_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

error

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

evaluation

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

evaluation_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

```
get_next_by_created_date (*, field=<django.db.models.fields.DateTimeField: created_date>,
                           is_next=True, **kwargs)
```

```
get_next_by_modified_date (*, field=<django.db.models.fields.DateTimeField: modified_date>, is_next=True, **kwargs)
```

```
get_previous_by_created_date (*, field=<django.db.models.fields.DateTimeField: created_date>, is_next=False, **kwargs)
```

```
get_previous_by_modified_date (*, field=<django.db.models.fields.DateTimeField: modified_date>, is_next=False, **kwargs)
```

```
get_status_display (*, field=<django.db.models.fields.CharField: status>)
```

```
get_type_display (*, field=<django.db.models.fields.CharField: type>)
```

hyperparameter_optimizer

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

hyperparameter_optimizer_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

incremental_train

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

incremental_train_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

labelling

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

labelling_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

modified_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

predictive_model

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

predictive_model_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

split

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

split_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

status

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

Return type dict

type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class `src.jobs.models.JobStatuses`

Bases: `enum.Enum`

An enumeration.

COMPLETED = 'completed'

CREATED = 'created'

ERROR = 'error'

```

    RUNNING = 'running'

class src.jobs.models.JobTypes
    Bases: enum.Enum

    An enumeration.

    LABELLING = 'labelling'

    PREDICTION = 'prediction'

    UPDATE = 'update'

class src.jobs.models.ModelType
    Bases: enum.Enum

    An enumeration.

    CLASSIFIER = 'classification'

    CLUSTERER = 'clusterer'

    REGRESSOR = 'regression'

    TIME_SERIES_PREDICTOR = 'time_series_prediction'

```

src.jobs.serializers module

```

class src.jobs.serializers.JobSerializer(instance=None, data=<class
    'rest_framework.fields.empty'>, **kwargs)
    Bases: rest_framework.serializers.ModelSerializer

    class Meta
        Bases: object

        fields = ('id', 'created_date', 'modified_date', 'error', 'status', 'type', 'config')

        model
            alias of src.jobs.models.Job

        get_config(job)

```

src.jobs.tasks module

src.jobs.urls module

src.jobs.views module

src.jobs.ws_publisher module

```

src.jobs.ws_publisher.publish(object)
    Publish an object to websocket listeners :param object: A Django predictive_model :return: {type: object class
    name, data: OBJECT}

```

Module contents

src.labelling package

Subpackages

Submodules

src.labelling.apps module

```
class src.labelling.apps.LabellingConfig(app_name, app_module)
    Bases: django.apps.config.AppConfig
    name = 'src.labelling'
```

src.labelling.common module

```
src.labelling.common.add_labels(encoding, labelling, prefix_length, trace, at-
                                tribute_classifier=None, executed_events=None, re-
                                sources_used=None, new_traces=None)
    Adds any number of label cells with last as label
```

```
src.labelling.common.compute_label_columns(columns, encoding, labelling)
```

Return type list

```
src.labelling.common.get_intercase_attributes(log, encoding)
    Dict of kwargs These intercase attributes are expensive operations!!!
```

```
src.labelling.common.next_event_name(trace, prefix_length)
    Return the event event name at prefix length or 0 if out of range.
```

src.labelling.label_container module

```
class src.labelling.label_container.LabelContainer
    Bases: src.labelling.label_container.LabelContainer
```

Inner object describing labelling state. For no labelling use NO_LABEL

This is a horrible hack and should be split into a label container and a container for encoding options, like what to add to the encoded log.

src.labelling.models module

```
class src.labelling.models.LabelTypes
    Bases: enum.Enum
    An enumeration.
    ATTRIBUTE_NUMBER = 'attribute_number'
    ATTRIBUTE_STRING = 'attribute_string'
    DURATION = 'duration'
    NEXT_ACTIVITY = 'next_activity'
```



```

NO_LABEL = 'no_label'
REMAINING_TIME = 'remaining_time'
class src.labelling.models.Labelling(id, type, attribute_name, threshold_type, threshold, re-
                                     sults)
    Bases: src.common.models.CommonModel
    exception DoesNotExist
        Bases: django.core.exceptions.ObjectDoesNotExist
    exception MultipleObjectsReturned
        Bases: django.core.exceptions.MultipleObjectsReturned
    attribute_name
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.
    get_threshold_type_display(*, field=<django.db.models.fields.CharField: threshold_type>)
    get_type_display(*, field=<django.db.models.fields.CharField: type>)
    id
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.
    job_set
        Accessor to the related objects manager on the reverse side of a many-to-one relation.
        In the example:
        

```
class Child(Model):
 parent = ForeignKey(Parent, related_name='children')
```


        Parent.children is a ReverseManyToOneDescriptor instance.
        Most of the implementation is delegated to a dynamically defined manager class built by
        create_forward_many_to_many_manager() defined below.
    labelledlog_set
        Accessor to the related objects manager on the reverse side of a many-to-one relation.
        In the example:
        

```
class Child(Model):
 parent = ForeignKey(Parent, related_name='children')
```


        Parent.children is a ReverseManyToOneDescriptor instance.
        Most of the implementation is delegated to a dynamically defined manager class built by
        create_forward_many_to_many_manager() defined below.
    objects = <django.db.models.manager.Manager object>
    results
        A placeholder class that provides a way to set the attribute on the model.
    threshold
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.
    threshold_type
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.

```

to_dict()

type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class `src.labelling.models.ThresholdTypes`

Bases: `enum.Enum`

An enumeration.

THRESHOLD_CUSTOM = 'threshold_custom'

THRESHOLD_MEAN = 'threshold_mean'

Module contents

src.logs package

Subpackages

Submodules

src.logs.admin module

src.logs.apps module

class `src.logs.apps.LogsConfig(app_name, app_module)`

Bases: `django.apps.config.AppConfig`

name = 'src.logs'

src.logs.log_service module

`src.logs.log_service.create_log(log, name, folder='cache/log_cache')`

`src.logs.log_service.create_properties(log)`

Create read-only dict with methods in this class

Return type dict

src.logs.models module

class `src.logs.models.Log(*args, **kwargs)`

Bases: `src.common.models.CommonModel`

A XES log file on disk

exception `DoesNotExist`

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception `MultipleObjectsReturned`

Bases: `django.core.exceptions.MultipleObjectsReturned`

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

log

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

original_log

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

path

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

properties

A placeholder class that provides a way to set the attribute on the model.

real_log

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

test_log

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`to_dict()`

`training_log`

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

src.logs.serializers module

```
class src.logs.serializers.LogSerializer(instance=None, data=<class
                                     'rest_framework.fields.empty'>, **kwargs)
    Bases: rest_framework.serializers.ModelSerializer

    class Meta
        Bases: object

        fields = ('id', 'name', 'properties')

    model
        alias of src.logs.models.Log
```

src.logs.urls module

src.logs.views module

```
class src.logs.views.LogDetail(**kwargs)
    Bases: rest_framework.mixins.RetrieveModelMixin, rest_framework.generics.
    GenericAPIView

    get(request, *args, **kwargs)

    queryset

    serializer_class
        alias of src.logs.serializers.LogSerializer

class src.logs.views.LogList(**kwargs)
    Bases: rest_framework.mixins.ListModelMixin, rest_framework.generics.
    GenericAPIView

    get(request, *args, **kwargs)

    post(request)

    queryset
```

```

serializer_class
    alias of src.logs.serializers.LogSerializer
class src.logs.views.SplitDetail (**kwargs)
    Bases: rest_framework.mixins.RetrieveModelMixin, rest_framework.generics.GenericAPIView
    get (request, *args, **kwargs)
    queryset
    serializer_class
        alias of src.split.serializers.SplitSerializer
src.logs.views.get_log_stats (self, request, *args, **kwargs)
    Get log statistics
    DEPRECATED ENDPOINT. LOGS HAVE PROPERTIES.
    End URL with * events for event_by_date * resources for resources_by_date * executions for event_executions
    * traceAttributes for trace_attributes * eventsInTrace for events_in_trace * newTraces for new_trace_start
src.logs.views.upload_multiple (self, request, *args, **kwargs)

```

Module contents

src.predictive_model package

Subpackages

src.predictive_model.classification package

Subpackages

Submodules

src.predictive_model.classification.apps module

```

class src.predictive_model.classification.apps.ClassificationConfig (app_name,
                                                                    app_module)
    Bases: src.predictive_model.apps.PredictiveModelConfig
    name = 'src.predictive_model.classification'

```

src.predictive_model.classification.classification module

src.predictive_model.classification.custom_classification_models module

```

class src.predictive_model.classification.custom_classification_models.NNClassifier (**kwargs)
    Bases: sklearn.base.ClassifierMixin
    Neural Network classifier, implements the same methods as the sklearn models to make it simple to add
    fit (train_data, targets)
        creates and fits the predictive_model

```

first the encoded data is parsed, then the `predictive_model` created and then trained

Parameters

- **train_data** (DataFrame) – encoded training dataset
- **targets** (ndarray) – encoded target dataset

Return type None

predict (*test_data*)

returns `predictive_model` predictions

parses the encoded test dataset, then returns the `predictive_model` predictions

Parameters **test_data** (DataFrame) – encoded test dataset

Return type ndarray

Returns `predictive_model` predictions

predict_proba (*test_data*)

returns the classification probability

parses the test dataset and returns the raw prediction probabilities of the `predictive_model`

Parameters **test_data** (DataFrame) – encoded test dataset

Return type ndarray

Returns `predictive_model` prediction probabilities

reset ()

placeholder to allow use with other sklearn algorithms

Return type None

src.predictive_model.classification.methods_default_config module

```
src.predictive_model.classification.methods_default_config.classification_decision_tree()  
src.predictive_model.classification.methods_default_config.classification_incremental_adapt()  
src.predictive_model.classification.methods_default_config.classification_incremental_hoeffding()  
src.predictive_model.classification.methods_default_config.classification_incremental_naive_bayes()  
src.predictive_model.classification.methods_default_config.classification_incremental_perceptron()  
src.predictive_model.classification.methods_default_config.classification_incremental_sgd()  
src.predictive_model.classification.methods_default_config.classification_knn()  
src.predictive_model.classification.methods_default_config.classification_nn()  
src.predictive_model.classification.methods_default_config.classification_random_forest()  
src.predictive_model.classification.methods_default_config.classification_xgboost()
```

src.predictive_model.classification.models module

```
class src.predictive_model.classification.models.AdaptiveHoeffdingTree(id,
                                                                    model_path,
                                                                    pre-
                                                                    dic-
                                                                    tive_model,
                                                                    pre-
                                                                    dic-
                                                                    tion_method,
                                                                    pre-
                                                                    dic-
                                                                    tive-
                                                                    model_ptr,
                                                                    clas-
                                                                    sifi-
                                                                    ca-
                                                                    tion_ptr,
                                                                    grace_period,
                                                                    split_criterion,
                                                                    split_confidence,
                                                                    tie_threshold,
                                                                    re-
                                                                    move_poor_atts,
                                                                    leaf_prediction,
                                                                    nb_threshold)
```

Bases: *src.predictive_model.classification.models.Classification*

exception DoesNotExist

Bases: *src.predictive_model.classification.models.DoesNotExist*

exception MultipleObjectsReturned

Bases: *src.predictive_model.classification.models.MultipleObjectsReturned*

classification_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

classification_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

```
get_leaf_prediction_display(*, field=<django.db.models.fields.CharField:
                             leaf_prediction>)
```

```
get_split_criterion_display(* ,field=<django.db.models.fields.CharField: split_criterion>)
```

grace_period

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

leaf_prediction

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

nb_threshold

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

remove_poor_atts

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

split_confidence

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

split_criterion

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

tie_threshold

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

```
class src.predictive_model.classification.models.Classification(*args,  
                                                                **kwargs)
```

Bases: `src.predictive_model.models.PredictiveModel`

Container of Classification to be shown in frontend

exception DoesNotExist

Bases: `src.predictive_model.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.models.MultipleObjectsReturned`

adaptivehoffdingtree

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

decisiontree

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

hoffdingtree

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:


```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

static init (*configuration*)

Return type *PredictiveModel*

knn

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

naivebayes

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

neuralnetwork

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

perceptron

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

predictivemodel_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Restaurant.place is a ForwardOneToOneDescriptor instance.

predictivemodel_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

randomforest

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

sgdclassifier

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

xgboost

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

class src.predictive_model.classification.models.**ClassificationMethods**

Bases: enum.Enum

An enumeration.

ADAPTIVE_TREE = 'adaptiveTree'

DECISION_TREE = 'decisionTree'

HOEFFDING_TREE = 'hoeffdingTree'

KNN = 'knn'

MULTINOMIAL_NAIVE_BAYES = 'multinomialNB'

NN = 'nn'

PERCEPTRON = 'perceptron'

RANDOM_FOREST = 'randomForest'

SGDClassifier = 'SGDClassifier'

XGBOOST = 'xgboost'

class src.predictive_model.classification.models.**DecisionTree**(*id*, *model_path*,
predictive_model,
prediction_method, *predictivemodel_ptr*,
classification_ptr,
max_depth,
min_samples_split,
min_samples_leaf)

Bases: `src.predictive_model.classification.models.Classification`

exception DoesNotExist

Bases: `src.predictive_model.classification.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.classification.models.MultipleObjectsReturned`

classification_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

classification_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

max_depth

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

min_samples_leaf

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

min_samples_split

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

```
class src.predictive_model.classification.models.HoeffdingTree(id, model_path,
                                                               predic-
                                                               tive_model, pre-
                                                               diction_method,
                                                               predictive-
                                                               model_ptr, clas-
                                                               sification_ptr,
                                                               grace_period,
                                                               split_criterion,
                                                               split_confidence,
                                                               tie_threshold,
                                                               re-
                                                               move_poor_atts,
                                                               leaf_prediction,
                                                               nb_threshold)
```

Bases: `src.predictive_model.classification.models.Classification`

exception DoesNotExist

Bases: `src.predictive_model.classification.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.classification.models.MultipleObjectsReturned`

classification_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

classification_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

```
get_leaf_prediction_display (*, field=<django.db.models.fields.CharField:  
                             leaf_prediction>)
```

```
get_split_criterion_display (*, field=<django.db.models.fields.CharField: split_criterion>)
```

grace_period

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

leaf_prediction

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

nb_threshold

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

remove_poor_atts

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

split_confidence

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

split_criterion

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

tie_threshold

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict ()

```
class src.predictive_model.classification.models.Knn (id, model_path, predic-  
                                                     tive_model, prediction_method,  
                                                     predictivemodel_ptr, clas-  
                                                     sification_ptr, n_neighbors,  
                                                     weights)
```

Bases: `src.predictive_model.classification.models.Classification`

exception DoesNotExist

Bases: `src.predictive_model.classification.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.classification.models.MultipleObjectsReturned`

classification_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

classification_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_weights_display (*, *field*=<django.db.models.fields.CharField: weights>)

n_neighbors

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict ()

weights

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

```
class src.predictive_model.classification.models.NaiveBayes (id,      model_path,
                                                            predictive_model,
                                                            prediction_method,
                                                            predictivemodel_ptr,
                                                            classification_ptr,
                                                            alpha, fit_prior)
```

Bases: `src.predictive_model.classification.models.Classification`

exception DoesNotExist

Bases: `src.predictive_model.classification.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.classification.models.MultipleObjectsReturned`

alpha

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

classification_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

classification_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

fit_prior

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

```
class src.predictive_model.classification.models.NeuralNetwork(id, model_path,
                                                             predic-
                                                             tive_model, pre-
                                                             diction_method,
                                                             predictive-
                                                             model_ptr, clas-
                                                             sification_ptr,
                                                             n_hidden_layers,
                                                             n_hidden_units,
                                                             activation,
                                                             n_epochs,
                                                             dropout_rate)
```

Bases: `src.predictive_model.classification.models.Classification`

exception DoesNotExist

Bases: `src.predictive_model.classification.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.classification.models.`
`MultipleObjectsReturned`

activation

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

classification_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

classification_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

dropout_rate

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_activation_display(* , field=<django.db.models.fields.CharField: activation>)

n_epochs

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_hidden_layers

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_hidden_units

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

```
class src.predictive_model.classification.models.Perceptron (id, model_path,
                                                         predictive_model,
                                                         prediction_method,
                                                         predictivemodel_ptr,
                                                         classification_ptr,
                                                         penalty, alpha,
                                                         fit_intercept, tol,
                                                         shuffle, eta0, val-
                                                         idation_fraction,
                                                         n_iter_no_change)
```

Bases: `src.predictive_model.classification.models.Classification`

exception DoesNotExist

Bases: `src.predictive_model.classification.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.classification.models.MultipleObjectsReturned`

alpha

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

classification_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model) :
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

classification_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

eta0

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

fit_intercept

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_penalty_display (*, *field*=<`django.db.models.fields.CharField: penalty`>)

n_iter_no_change

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

penalty

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

shuffle

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict ()

tol

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

validation_fraction

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

```
class src.predictive_model.classification.models.RandomForest (id, model_path,  
predictive_model,  
prediction_method, predic-  
tivemodel_ptr,  
classification_ptr,  
n_estimators,  
max_depth,  
max_features)
```

Bases: `src.predictive_model.classification.models.Classification`

exception DoesNotExist

Bases: `src.predictive_model.classification.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.classification.models.`
`MultipleObjectsReturned`

classification_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model) :  
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

classification_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

max_depth

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

max_features

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_estimators

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict ()


```
class src.predictive_model.classification.models.SGDClassifier(id, model_path,
                                                             predic-
                                                             tive_model, pre-
                                                             diction_method,
                                                             predictive-
                                                             model_ptr, clas-
                                                             sification_ptr,
                                                             loss, penalty,
                                                             alpha, ll_ratio,
                                                             fit_intercept,
                                                             tol, epsilon,
                                                             learning_rate,
                                                             eta0, power_t,
                                                             n_iter_no_change,
                                                             valida-
                                                             tion_fraction,
                                                             average)
```

Bases: `src.predictive_model.classification.models.Classification`

exception DoesNotExist

Bases: `src.predictive_model.classification.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.classification.models.MultipleObjectsReturned`

alpha

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

average

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

classification_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model) :
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

classification_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

epsilon

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

eta0

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

fit_intercept

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_learning_rate_display (*,field=<django.db.models.fields.CharField: learning_rate>)

get_loss_display (*,field=<django.db.models.fields.CharField: loss>)

get_penalty_display (*,field=<django.db.models.fields.CharField: penalty>)

ll_ratio

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

learning_rate

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

loss

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_iter_no_change

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

penalty

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

power_t

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict ()

tol

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

validation_fraction

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

```
class src.predictive_model.classification.models.XGBoost (id, model_path, pre-
    dictive_model, predic-
    tion_method, predictive-
    model_ptr, classifica-
    tion_ptr, n_estimators,
    max_depth)
```

Bases: `src.predictive_model.classification.models.Classification`

exception DoesNotExist

Bases: `src.predictive_model.classification.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.classification.models.
MultipleObjectsReturned`

classification_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField (Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

classification_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

max_depth

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_estimators

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

Module contents

src.predictive_model.regression package

Subpackages

Submodules

src.predictive_model.regression.apps module

```
class src.predictive_model.regression.apps.RegressionConfig(app_name,
                                                         app_module)
    Bases: src.predictive_model.apps.PredictiveModelConfig
    name = 'src.predictive_model.regression'
```

src.predictive_model.regression.custom_regression_models module

```
class src.predictive_model.regression.custom_regression_models.NNRegressor(**kwargs)
    Bases: sklearn.base.RegressorMixin
```

Neural Network regressor, implements the same methods as the sklearn models to make it simple to add

```
fit(train_data, targets)
    creates and fits the predictive_model
    first the encoded data is parsed, then the predictive_model created and then trained
```

Parameters

- **train_data** (*DataFrame*) – encoded training dataset
- **targets** (*ndarray*) – encoded target dataset

Return type *None*

```
predict(test_data)
    returns predictive_model predictions
    parses the encoded test dataset, then returns the predictive_model predictions
```

Parameters **test_data** (*DataFrame*) – encoded test dataset

Return type ndarray

Returns predictive_model predictions

reset()

placeholder to allow use with other sklearn algorithms

Return type None

src.predictive_model.regression.methods_default_config module

src.predictive_model.regression.methods_default_config.**regression_lasso()**

src.predictive_model.regression.methods_default_config.**regression_linear()**

src.predictive_model.regression.methods_default_config.**regression_nn()**

src.predictive_model.regression.methods_default_config.**regression_random_forest()**

src.predictive_model.regression.methods_default_config.**regression_xgboost()**

src.predictive_model.regression.models module

```
class src.predictive_model.regression.models.Lasso(id, model_path, predictive_model,
                                                    prediction_method, predictive-
                                                    model_ptr, regression_ptr, alpha,
                                                    fit_intercept, normalize)
```

Bases: *src.predictive_model.regression.models.Regression*

exception DoesNotExist

Bases: *src.predictive_model.regression.models.DoesNotExist*

exception MultipleObjectsReturned

Bases: *src.predictive_model.regression.models.MultipleObjectsReturned*

alpha

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

fit_intercept

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

normalize

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

regression_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

regression_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

```
class src.predictive_model.regression.models.Linear(id,      model_path,      predic-
                                                    tive_model, prediction_method,
                                                    predictivemodel_ptr, regres-
                                                    sion_ptr, fit_intercept, normal-
                                                    ize)
```

Bases: *src.predictive_model.regression.models.Regression*

exception DoesNotExist

Bases: *src.predictive_model.regression.models.DoesNotExist*

exception MultipleObjectsReturned

Bases: *src.predictive_model.regression.models.MultipleObjectsReturned*

fit_intercept

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

normalize

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

regression_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

regression_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

```
class src.predictive_model.regression.models.NeuralNetwork(id,      model_path,
                                                    predictive_model,
                                                    prediction_method,
                                                    predictivemodel_ptr,
                                                    regression_ptr,
                                                    n_hidden_layers,
                                                    n_hidden_units, ac-
                                                    tivation,  n_epochs,
                                                    dropout_rate)
```

Bases: *src.predictive_model.regression.models.Regression*

exception DoesNotExist

Bases: *src.predictive_model.regression.models.DoesNotExist*

exception MultipleObjectsReturned

Bases: *src.predictive_model.regression.models.MultipleObjectsReturned*

activation

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

dropout_rate

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_activation_display (*, field=<django.db.models.fields.CharField: activation>)

n_epochs

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_hidden_layers

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_hidden_units

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

regression_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

regression_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict ()

```
class src.predictive_model.regression.models.RandomForest (id, model_path, pre-  
                                                            dictive_model, predic-  
                                                            tion_method, predic-  
                                                            tivemodel_ptr, regres-  
                                                            sion_ptr, n_estimators,  
                                                            max_features,  
                                                            max_depth)
```

Bases: `src.predictive_model.regression.models.Regression`

exception DoesNotExist

Bases: `src.predictive_model.regression.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.regression.models.MultipleObjectsReturned`

max_depth

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

max_features

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_estimators

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

random_state = 21

regression_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

regression_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

class `src.predictive_model.regression.models.Regression(*args, **kwargs)`

Bases: `src.predictive_model.models.PredictiveModel`

Container of Regression to be shown in frontend

exception DoesNotExist

Bases: `src.predictive_model.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.models.MultipleObjectsReturned`

static init(configuration)**lasso**

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

linear

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

neuralnetwork

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

predictivemodel_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

predictivemodel_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

randomforest

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

xgboost

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

class src.predictive_model.regression.models.RegressionMethods

Bases: `enum.Enum`

An enumeration.

LASSO = 'lasso'

LINEAR = 'linear'

NN = 'nn'

RANDOM_FOREST = 'randomForest'

XGBOOST = 'xgboost'

```
class src.predictive_model.regression.models.XGBoost(id, model_path, predic-
                                                    tive_model, prediction_method,
                                                    predictivemodel_ptr, re-
                                                    gression_ptr, max_depth,
                                                    n_estimators)
```

Bases: `src.predictive_model.regression.models.Regression`

exception DoesNotExist

Bases: `src.predictive_model.regression.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.regression.models.MultipleObjectsReturned`

max_depth

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_estimators

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

regression_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):
    place = OneToOneField (Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

regression_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()**src.predictive_model.regression.regression module**

regression methods and functionalities

`src.predictive_model.regression.regression.regression(training_df, test_df, clusterer, job)`

main regression entry point

train and tests the regressor using the provided data

Parameters

- **clusterer** (*Clustering*) –
- **training_df** (`DataFrame`) – training `DataFrame`
- **test_df** (`DataFrame`) – testing `DataFrame`
- **job** (*Job*) – job configuration

Return type (<class 'dict'>, <class 'dict'>)

Returns predictive_model scores and split

`src.predictive_model.regression.regression.regression_single_log(input_df, model)`

single log regression

classifies a single log using the provided TODO: complete

Parameters

- **input_df** (`DataFrame`) – input `DataFrame`
- **model** (`dict`) – TODO: complete

Return type `DataFrame`

Returns predictive_model scores

Module contents

src.predictive_model.time_series_prediction package

Subpackages

Submodules

src.predictive_model.time_series_prediction.TimeSeriesPredictorMixin module

```
class src.predictive_model.time_series_prediction.TimeSeriesPredictorMixin.TimeSeriesPredi  
    Bases: object
```

src.predictive_model.time_series_prediction.apps module

```
class src.predictive_model.time_series_prediction.apps.TimeSeriesPredictionConfig(app_name,  
    Bases: src.predictive_model.apps.PredictiveModelConfig                                app_module  
    name = 'src.predictive_model.time_series_prediction'
```

src.predictive_model.time_series_prediction.custom_time_series_prediction_models module

```
class src.predictive_model.time_series_prediction.custom_time_series_prediction_models.RNN  
    Bases: src.predictive_model.time_series_prediction.TimeSeriesPredictorMixin,  
           TimeSeriesPredictorMixin
```

Recurrent Neural Network Time Series predictor, implements the same methods as the sklearn models to make it simple to add. This architecture is of the seq2seq type, taking as input a sequence (0...t) and outputting a sequence (1...t+1)

fit (*train_data*)
 creates and fits the predictive_model
 first the encoded data is parsed, then the predictive_model created and then trained

Parameters **train_data** (DataFrame) – encoded training dataset

Return type None

predict (*test_data*)
 returns predictive_model predictions
 parses the encoded test dataset, then returns the predictive_model predictions

Parameters **test_data** (DataFrame) – encoded test dataset

Return type ndarray

Returns predictive_model predictions

src.predictive_model.time_series_prediction.methods_default_config module

```
src.predictive_model.time_series_prediction.methods_default_config.time_series_prediction_1
```

src.predictive_model.time_series_prediction.models module

```
class src.predictive_model.time_series_prediction.models.RecurrentNeuralNetwork (id,  
model_path,  
pre-  
dic-  
tive_model,  
pre-  
dic-  
tion_method,  
pre-  
dic-  
tive-  
model_ptr,  
time-  
seriespre-  
dic-  
tion_ptr,  
n_units,  
rnn_type,  
n_epochs)
```

Bases: `src.predictive_model.time_series_prediction.models.
TimeSeriesPrediction`

exception DoesNotExist

Bases: `src.predictive_model.time_series_prediction.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `src.predictive_model.time_series_prediction.models.
MultipleObjectsReturned`

get_rnn_type_display (*,field=<django.db.models.fields.CharField: rnn_type>)

n_epochs

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_units

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

rnn_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

timeseriesprediction_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

timeseriesprediction_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`to_dict()`

class `src.predictive_model.time_series_prediction.models.TimeSeriesPrediction` (*args,
**kwargs)

Bases: `src.predictive_model.models.PredictiveModel`

Container of Classification to be shown in frontend

exception `DoesNotExist`

Bases: `src.predictive_model.models.DoesNotExist`

exception `MultipleObjectsReturned`

Bases: `src.predictive_model.models.MultipleObjectsReturned`

static `init` (configuration)

predictivemodel_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

predictivemodel_ptr_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

recurrentneuralnetwork

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model):  
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

class `src.predictive_model.time_series_prediction.models.TimeSeriesPredictionMethods`

Bases: `enum.Enum`

An enumeration.

`RNN = 'rnn'`

`src.predictive_model.time_series_prediction.time_series_prediction` module

time series prediction methods and functionalities

`src.predictive_model.time_series_prediction.time_series_prediction.time_series_prediction` (t

main time series prediction entry point

train and tests the time series predictor using the provided data

Parameters

- `clusterer` (*Clustering*) –

- **training_df** (DataFrame) – training DataFrame
- **test_df** (DataFrame) – testing DataFrame
- **job** (*Job*) – job configuration

Return type (<class 'dict'>, <class 'dict'>)

Returns predictive_model scores and split

`src.predictive_model.time_series_prediction.time_series_prediction.time_series_prediction_`

single log time series prediction

time series predicts a single log using the provided TODO: complete

Parameters

- **input_df** (DataFrame) – input DataFrame
- **model** (dict) – TODO: complete

Return type dict

Returns predictive_model scores

Module contents

Submodules

src.predictive_model.apps module

```
class src.predictive_model.apps.PredictiveModelConfig(app_name, app_module)
    Bases: django.apps.config.AppConfig
    name = 'src.predictive_model'
```

src.predictive_model.models module

```
class src.predictive_model.models.PredictiveModel(*args, **kwargs)
    Bases: src.common.models.CommonModel
```

Container of Classification to be shown in frontend

exception DoesNotExist

Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned

Bases: django.core.exceptions.MultipleObjectsReturned

classification

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

get_predictive_model_display (*, field=<django.db.models.fields.CharField: predictive_model>)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

static init (configuration)

job_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

model_path

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <model_utils.managers.InheritanceManager object>

prediction_method

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

predictive_model

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

regression

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

timeseriesprediction

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

to_dict ()

class src.predictive_model.models.PredictiveModels

Bases: enum.Enum

An enumeration.

CLASSIFICATION = 'classification'

```
REGRESSION = 'regression'
TIME_SERIES_PREDICTION = 'time_series_prediction'
```

Module contents

src.runtime package

Subpackages

Submodules

src.runtime.apps module

```
class src.runtime.apps.RuntimeConfig(app_name, app_module)
    Bases: django.apps.config AppConfig
    name = 'src.runtime'
```

src.runtime.models module

```
class src.runtime.models.DemoReplayer(id, running)
    Bases: src.common.models.CommonModel

    exception DoesNotExist
        Bases: django.core.exceptions.ObjectDoesNotExist

    exception MultipleObjectsReturned
        Bases: django.core.exceptions.MultipleObjectsReturned

    id
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.

    objects = <django.db.models.manager.Manager object>

    running
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.

    to_dict()

class src.runtime.models.XEvent(id, config, xid, trace)
    Bases: src.common.models.CommonModel

    exception DoesNotExist
        Bases: django.core.exceptions.ObjectDoesNotExist

    exception MultipleObjectsReturned
        Bases: django.core.exceptions.MultipleObjectsReturned

    config
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.
```

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>**to_dict()****trace**

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

trace_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

xid

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class src.runtime.models.XLog(id, config, real_log)

Bases: *src.common.models.CommonModel*

exception DoesNotExist

Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned

Bases: django.core.exceptions.MultipleObjectsReturned

config

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>**real_log**

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

real_log_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

xlog

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

```
class src.runtime.models.XTrace(id, config, name, xlog, completed, first_event, last_event,
                                n_events, error, real_log, reg_results, class_results, reg_actual,
                                duration, class_actual, reg_model, class_model)
```

Bases: `src.common.models.CommonModel`

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

class_actual

A placeholder class that provides a way to set the attribute on the model.

class_model

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

class_model_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class_results

A placeholder class that provides a way to set the attribute on the model.

completed

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

config

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

duration

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

error

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

first_event

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

last_event

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

n_events

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

real_log

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

reg_actual

A placeholder class that provides a way to set the attribute on the model.

reg_model

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

reg_model_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

reg_results

A placeholder class that provides a way to set the attribute on the model.

to_dict()**xlog**

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

xlog_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

xtrace

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

src.runtime.replay_core module**src.runtime.replayer module****src.runtime.serializers module**

```
class src.runtime.serializers.TraceSerializer(instance=None, data=<class
                                             'rest_framework.fields.empty'>,
                                             **kwargs)
    Bases: rest_framework.serializers.ModelSerializer

    class Meta
        Bases: object
        fields = ('id', 'name', 'completed', 'real_log', 'first_event', 'last_event', 'n_ev
        model
            alias of src.runtime.models.XTrace
```

src.runtime.tasks module**src.runtime.tests module****src.runtime.urls module****src.runtime.views module****Module contents****src.split package****Subpackages****Submodules**

src.split.apps module

```
class src.split.apps.SplitConfig(app_name, app_module)
    Bases: django.apps.config.AppConfig
    name = 'src.split'
```

src.split.models module

```
class src.split.models.Split(*args, **kwargs)
    Bases: src.common.models.CommonModel
    Container of Split to be shown in frontend

exception DoesNotExist
    Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
    Bases: django.core.exceptions.MultipleObjectsReturned

additional_columns
    A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
    executed.

get_splitting_method_display(* , field=<django.db.models.fields.CharField: splitting_method>)

get_type_display(* ,field=<django.db.models.fields.CharField: type>)

id
    A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
    executed.

job_set
    Accessor to the related objects manager on the reverse side of a many-to-one relation.

    In the example:
```

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

```
labelledlog_set
    Accessor to the related objects manager on the reverse side of a many-to-one relation.

    In the example:
```

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

loadedlog_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

objects = <django.db.models.manager.Manager object>

original_log

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

original_log_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

splitting_method

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

test_log

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

test_log_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

test_size

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

to_dict()

Return type dict

train_log

Accessor to the related object on the forward side of a many-to-one or one-to-one (via `ForwardOneToOneDescriptor` subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

train_log_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class src.split.models.SplitOrderingMethods

Bases: enum.Enum

An enumeration.

SPLIT_RANDOM = 'random'

SPLIT_SEQUENTIAL = 'sequential'

SPLIT_STRICT_TEMPORAL = 'strict_temporal'

SPLIT_TEMPORAL = 'temporal'

class src.split.models.SplitTypes

Bases: enum.Enum

An enumeration.

SPLIT_DOUBLE = 'double'

SPLIT_SINGLE = 'single'

src.split.serializers module

```
class src.split.serializers.CreateSplitSerializer(instance=None, data=<class
                                                    'rest_framework.fields.empty'>,
                                                    **kwargs)
```

Bases: rest_framework.serializers.ModelSerializer

class Meta

Bases: object

fields = ('original_log', 'splitting_method', 'test_size')

model

alias of *src.split.models.Split*

```
class src.split.serializers.SplitSerializer(instance=None, data=<class
                                                    'rest_framework.fields.empty'>, **kwargs)
```

Bases: rest_framework.serializers.ModelSerializer

class Meta

Bases: object

fields = ('id', 'original_log', 'type', 'splitting_method', 'test_log', 'training_log')

model

alias of *src.split.models.Split*

get_training_log (*split*)

src.split.splitting module

`src.split.splitting.prepare_logs(split)`
Returns training_log and test_log

src.split.urls module

src.split.views module

```
class src.split.views.SplitList(**kwargs)
    Bases: rest_framework.mixins.ListModelMixin, rest_framework.generics.
            GenericAPIView
    get (request, *args, **kwargs)
    static post (request)
    queryset
    serializer_class
        alias of src.split.serializers.SplitSerializer
```

Module contents

src.utils package

Subpackages

Submodules

src.utils.django_orm module

`src.utils.django_orm.duplicate_orm_row(obj)`

src.utils.event_attributes module

`src.utils.event_attributes.get_additional_columns(log)`

`src.utils.event_attributes.get_event_attributes(log)`

Get log event attributes that are not name or time

As log file is a list, it has no global event attributes. Getting from first event of first trace. This may be bad.

`src.utils.event_attributes.get_global_event_attributes(log)`

Get log event attributes that are not name or time

`src.utils.event_attributes.get_global_trace_attributes(log)`

`src.utils.event_attributes.unique_events(log)`

List of unique events using event concept:name

Adds all events into a list and removes duplicates while keeping order.

`src.utils.event_attributes.unique_events2(training_log, test_log)`

Combines unique events from two logs into one list.

Renamed to 2 because Python doesn't allow functions with same names. Python is objectively the worst language.

`src.utils.file_service` module

`src.utils.file_service.create_unique_name(name)`

Return type `str`

`src.utils.file_service.get_log(log)`

Read in event log from disk

Uses `xes_importer` to parse log.

Return type `EventLog`

`src.utils.file_service.save_result(results, job, start_time)`

`src.utils.log_metrics` module

`src.utils.log_metrics.avg_events_in_log(log)`

Returns the average number of events in any trace

:return 3

Return type `int`

`src.utils.log_metrics.event_executions(log)`

Creates dict of event execution count

Return {'Event A' 7, '2011-01-06': 8}

Return type `OrderedDict`

`src.utils.log_metrics.events_by_date(log)`

Creates dict of events by date ordered by date

Return {'2010-12-30' 7, '2011-01-06': 8}

Return type `OrderedDict`

`src.utils.log_metrics.events_in_trace(log)`

Creates dict of number of events in trace

Return {'4' 11, '3': 8}

Return type `OrderedDict`

`src.utils.log_metrics.max_events_in_log(log)`

Returns the maximum number of events in any trace

:return 3

Return type `int`

`src.utils.log_metrics.new_trace_start(log)`

Creates dict of new traces by date

Return {'2010-12-30' 1, '2011-01-06': 2}

Return type OrderedDict

`src.utils.log_metrics.resources_by_date(log)`

Creates dict of used unique resources ordered by date

Resource and timestamp delimited by &&. If this is in resources name, bad stuff will happen. Returns a dict with a date and the number of unique resources used on that day. :return {'2010-12-30': 7, '2011-01-06': 8}

Return type OrderedDict

`src.utils.log_metrics.std_var_events_in_log(log)`

Returns the standard variation of the average number of events in any trace

:return 3

Return type int

`src.utils.log_metrics.trace_attributes(log)`

Creates an array of dicts that describe trace attributes. Only looks at first trace. Filters out *concept:name*.

Return [{name 'name', type: 'string', example: 34}]

Return type list

src.utils.result_metrics module

`src.utils.result_metrics.calculate_auc(actual, scores, auc)`

Return type float

`src.utils.result_metrics.calculate_nlebenshtein(actual, predicted)`

Return type float

`src.utils.result_metrics.calculate_results_classification(actual, predicted)`

Return type dict

`src.utils.result_metrics.calculate_results_regression(input_df, label)`

Return type dict

`src.utils.result_metrics.calculate_results_time_series_prediction(actual, predicted)`

Return type dict

`src.utils.result_metrics.get_auc(actual, scores)`

Return type float

`src.utils.result_metrics.get_confusion_matrix(actual, predicted)`

Return type dict

src.utils.tests_utils module

`src.utils.tests_utils.create_test_clustering(clustering_type='noCluster', configuration={})`

Return type Clustering

```
src.utils.tests_utils.create_test_encoding(prefix_length=1, padding=False,
                                           value_encoding='simpleIndex',
                                           add_elapsed_time=False,
                                           add_remaining_time=False,
                                           add_resources_used=False,
                                           add_new_traces=False,
                                           add_executed_events=False,
                                           task_generation_type='only')
```

Return type *Encoding*

```
src.utils.tests_utils.create_test_hyperparameter_optimizer(hyperoptim_type='hyperopt',
                                                           perform-
                                                           mance_metric='acc',
                                                           max_evals=10)

src.utils.tests_utils.create_test_job(split=None, encoding=None, labelling=None,
                                       clustering=None, predictive_model=None,
                                       job_type='prediction', hyperparameter-
                                       optimizer=None)
```

```
src.utils.tests_utils.create_test_labelling(label_type='next_activity', at-
                                           tribute_name=None, thresh-
                                           old_type='threshold_mean', threshold=0.0)
```

Return type *Labelling*

```
src.utils.tests_utils.create_test_log(log_name='general_example.xes',
                                       log_path='cache/log_cache/test_logs/general_example.xes')
```

Return type *Log*

```
src.utils.tests_utils.create_test_predictive_model(predictive_model='classification',
                                                    predic-
                                                    tion_method='randomForest')
```

Return type *PredictiveModel*

```
src.utils.tests_utils.create_test_split(split_type='single',
                                         split_ordering_method='sequential',
                                         test_size=0.2, original_log=None,
                                         train_log=None, test_log=None)
```

src.utils.time_metrics module

```
src.utils.time_metrics.count_on_event_day(trace, date_dict, event_id)
    Finds the date of event and returns the value from date_dict :param date_dict one of the dicts from log_metrics.py
    :param event_id Event id :param trace Log trace

src.utils.time_metrics.duration(trace)
    Calculate the duration of a trace

src.utils.time_metrics.elapsed_time(trace, event)
    Calculate elapsed time by event in trace

src.utils.time_metrics.elapsed_time_id(trace, event_index)
    Calculate elapsed time by event index in trace

src.utils.time_metrics.remaining_time(trace, event)
    Calculate remaining time by event in trace
```

```
src.utils.time_metrics.remaining_time_id(trace, event_index)
```

Calculate remaining time by event index in trace

Module contents

3.1.2 Module contents

S

src, 79

src.cache, 12

src.cache.apps, 9

src.cache.cache, 9

src.cache.models, 10

src.clustering, 16

src.clustering.apps, 13

src.clustering.clustering, 13

src.clustering.methods_default_config, 13

src.clustering.models, 13

src.common, 17

src.common.apps, 16

src.common.models, 16

src.core, 17

src.core.common, 17

src.encoding, 22

src.encoding.apps, 17

src.encoding.boolean_frequency, 17

src.encoding.common, 17

src.encoding.complex_last_payload, 18

src.encoding.encoder, 18

src.encoding.encoding_container, 18

src.encoding.encoding_parser, 18

src.encoding.models, 20

src.encoding.simple_index, 22

src.evaluation, 27

src.evaluation.apps, 23

src.evaluation.models, 23

src.hyperparameter_optimization, 30

src.hyperparameter_optimization.apps, 28

src.hyperparameter_optimization.hyperopt_spaces, 28

src.hyperparameter_optimization.methods_default_config, 28

src.hyperparameter_optimization.models, 28

src.jobs, 36

src.jobs.admin, 30

src.jobs.apps, 31

src.jobs.job_creator, 31

src.jobs.json_renderer, 31

src.jobs.models, 31

src.jobs.serializers, 35

src.jobs.ws_publisher, 35

src.labelling, 38

src.labelling.apps, 36

src.labelling.common, 36

src.labelling.label_container, 36

src.labelling.models, 36

src.logs, 41

src.logs.admin, 38

src.logs.apps, 38

src.logs.log_service, 38

src.logs.models, 38

src.logs.serializers, 40

src.logs.urls, 40

src.logs.views, 40

src.predictive_model, 67

src.predictive_model.apps, 65

src.predictive_model.classification, 55

src.predictive_model.classification.apps, 41

src.predictive_model.classification.custom_classification, 41

src.predictive_model.classification.methods_default_config, 42

src.predictive_model.classification.models, 43

src.predictive_model.models, 65

src.predictive_model.regression, 62

src.predictive_model.regression.apps, 55

src.predictive_model.regression.custom_regression_model, 55

src.predictive_model.regression.methods_default_config, 56

```
src.predictive_model.regression.models,
56
src.predictive_model.regression.regression,
61
src.predictive_model.time_series_prediction,
65
src.predictive_model.time_series_prediction.apps,
62
src.predictive_model.time_series_prediction.custom_time_series_prediction_models,
62
src.predictive_model.time_series_prediction.methods_default_config,
62
src.predictive_model.time_series_prediction.models,
63
src.predictive_model.time_series_prediction.time_series_prediction,
64
src.predictive_model.time_series_prediction.TimeSeriesPredictorMixin,
62
src.runtime, 71
src.runtime.apps, 67
src.runtime.models, 67
src.runtime.serializers, 71
src.split, 75
src.split.apps, 72
src.split.models, 72
src.split.serializers, 74
src.split.splitting, 75
src.split.urls, 75
src.split.views, 75
src.utils, 79
src.utils.django_orm, 75
src.utils.event_attributes, 75
src.utils.file_service, 76
src.utils.log_metrics, 76
src.utils.result_metrics, 77
src.utils.tests_utils, 77
src.utils.time_metrics, 78
```

A

abstract (*src.common.models.CommonModel.Meta* attribute), 16
 ACC (*src.hyperparameter_optimization.models.HyperOptLosses* attribute), 29
 accuracy (*src.evaluation.models.ClassificationMetrics* attribute), 24
 activation (*src.predictive_model.classification.models.NeuralNetwork* attribute), 50
 activation (*src.predictive_model.regression.models.NeuralNetwork* attribute), 57
 ADAPTIVE_TREE (*src.predictive_model.classification.models.ClassificationMetrics* attribute), 46
 AdaptiveHoeffdingTree (class in *src.predictive_model.classification.models*), 43
 adaptivehoeffdingtree (*src.predictive_model.classification.models.ClassificationMetrics* attribute), 44
 AdaptiveHoeffdingTree.DoesNotExist, 43
 AdaptiveHoeffdingTree.MultipleObjectsReturned, 43
 add_elapsed_time (*src.encoding.models.Encoding* attribute), 20
 add_executed_events (*src.encoding.models.Encoding* attribute), 20
 add_labels() (in module *src.labelling.common*), 36
 add_new_traces (*src.encoding.models.Encoding* attribute), 21
 add_remaining_time (*src.encoding.models.Encoding* attribute), 21
 add_resources_used (*src.encoding.models.Encoding* attribute), 21
 add_trace_row() (in module *src.encoding.simple_index*), 22
 additional_columns (*src.split.models.Split* attribute), 72
 additional_columns_path (*src.cache.models.LoadedLog* attribute), 12
 algorithm (*src.clustering.models.KMeans* attribute), 15
 algorithm_type (*src.hyperparameter_optimization.models.HyperOpt* attribute), 28
 ALL_IN_ONE (*src.encoding.models.TaskGenerationTypes* attribute), 22
 alpha (*src.predictive_model.classification.models.NaiveBayes* attribute), 49
 alpha (*src.predictive_model.classification.models.Perceptron* attribute), 51
 alpha (*src.predictive_model.classification.models.SGDClassifier* attribute), 53
 alpha (*src.predictive_model.regression.models.Lasso* attribute), 56
 attribute_name (*src.labelling.models.Labelling* attribute), 37
 ATTRIBUTE_NUMBER (*src.labelling.models.LabelTypes* attribute), 36
 ATTRIBUTE_STRING (*src.labelling.models.LabelTypes* attribute), 36
 auc (*src.evaluation.models.BinaryClassificationMetrics* attribute), 23
 AUC (*src.hyperparameter_optimization.models.HyperOptLosses* attribute), 29
 average (*src.predictive_model.classification.models.SGDClassifier* attribute), 53
 avg_events_in_log() (in module *src.utils.log_metrics*), 76

B

base_model (*src.jobs.models.Job* attribute), 31
 BinaryClassificationMetrics (class in *src.evaluation.models*), 23
 binaryclassificationmetrics (*src.evaluation.models.ClassificationMetrics* attribute), 24

BinaryClassificationMetrics.DoesNotExist, 42
 23 classification_incremental_adaptive_tree()
 BinaryClassificationMetrics.MultipleObjectsReturned, 42
 23 (in module *src.predictive_model.classification.methods_default_config*)
 BOOLEAN (*src.encoding.models.ValueEncodings* attribute), 22
 classification_incremental_hoeffding_tree()
 (in module *src.predictive_model.classification.methods_default_config*)
 boolean() (in module 42
src.encoding.boolean_frequency), 17 classification_incremental_naive_bayes()
 build_encoders() (*src.encoding.encoding_parser.DataEncoder* (in module *src.predictive_model.classification.methods_default_config*), 19 42
 method), 19 classification_incremental_perceptron()
 (in module *src.predictive_model.classification.methods_default_config*)
 42
C classification_incremental_sgd_classifier()
 (in module *src.predictive_model.classification.methods_default_config*)
 42
 Cache (class in *src.cache.models*), 10 classification_knn() (in module
src.predictive_model.classification.methods_default_config),
 Cache.DoesNotExist, 10 42
 Cache.MultipleObjectsReturned, 10 classification_nn() (in module
src.predictive_model.classification.methods_default_config),
 cache_ptr (*src.cache.models.LabelledLog* attribute), 10 42
 10 classification_ptr
 cache_ptr (*src.cache.models.LoadedLog* attribute), 12 (in module *src.predictive_model.classification.models.AdaptiveHoeffdingTree*
 attribute), 11 43
 cache_ptr_id (*src.cache.models.LabelledLog* attribute), 11 classification_ptr
 (in module *src.predictive_model.classification.models.DecisionTree*
 attribute), 12 47
 cache_ptr_id (*src.cache.models.LoadedLog* attribute), 12 classification_ptr
 (in module *src.predictive_model.classification.models.HoeffdingTree*
 attribute), 47
 CacheConfig (class in *src.cache.apps*), 9 classification_ptr
 calculate_auc() (in module 48
src.utils.result_metrics), 77 (in module *src.predictive_model.classification.models.Knn*
 attribute), 48
 calculate_nleventshtein() (in module 49
src.utils.result_metrics), 77 (in module *src.predictive_model.classification.models.NaiveBayes*
 attribute), 49
 calculate_results_classification() (in 50
 module *src.utils.result_metrics*), 77 classification_ptr
 (in module *src.predictive_model.classification.models.Perceptron*
 attribute), 51
 calculate_results_regression() (in module 52
src.utils.result_metrics), 77 (in module *src.predictive_model.classification.models.RandomForest*
 attribute), 52
 calculate_results_time_series_prediction() (in module *src.predictive_model.classification.models.SGDClassifier*
 attribute), 53
 (in module *src.utils.result_metrics*), 77 classification_ptr
 (in module *src.predictive_model.classification.models.XGBoost*
 attribute), 54
 CATEGORICAL (*src.encoding.encoding_parser.DataEncoder.DataType* attribute), 19 55
 class_actual (*src.runtime.models.XTrace* attribute), 69 classification_ptr_id
 69 (in module *src.predictive_model.classification.models.AdaptiveHoeffdingTree*
 attribute), 55
 class_model (*src.runtime.models.XTrace* attribute), 69
 69
 class_model_id (*src.runtime.models.XTrace* attribute), 69
 69
 class_results (*src.runtime.models.XTrace* attribute), 69
 69
 Classification (class in *src.predictive_model.classification.models*), 44
 44
 classification (*src.predictive_model.models.PredictiveModel* attribute), 65
 65
 CLASSIFICATION (*src.predictive_model.models.PredictiveModels* attribute), 66
 66
 Classification.DoesNotExist, 44
 Classification.MultipleObjectsReturned, 44
 44
 classification_decision_tree() (in module *src.predictive_model.classification.methods_default_config*), 55
 55
 classification_ptr_id
 (in module *src.predictive_model.classification.models.AdaptiveHoeffdingTree*
 attribute), 55

`attribute`), 43
`classification_ptr_id`
`(src.predictive_model.classification.models.DecisionTree`
`attribute)`, 47
`classification_ptr_id`
`(src.predictive_model.classification.models.HoeffdingTree`
`attribute)`, 48
`classification_ptr_id`
`(src.predictive_model.classification.models.Knn`
`attribute)`, 49
`classification_ptr_id`
`(src.predictive_model.classification.models.NaiveBayes`
`attribute)`, 49
`classification_ptr_id`
`(src.predictive_model.classification.models.NeuralNetwork`
`attribute)`, 50
`classification_ptr_id`
`(src.predictive_model.classification.models.Perceptron`
`attribute)`, 51
`classification_ptr_id`
`(src.predictive_model.classification.models.RandomForest`
`attribute)`, 52
`classification_ptr_id`
`(src.predictive_model.classification.models.SGDClassifier`
`attribute)`, 53
`classification_ptr_id`
`(src.predictive_model.classification.models.XGBoost`
`attribute)`, 55
`classification_random_forest()` (in module
`src.predictive_model.classification.methods_default_config`)
42
`classification_xgboost()` (in module
`src.predictive_model.classification.methods_default_config`)
42
`ClassificationConfig` (class in
`src.predictive_model.classification.apps`),
41
`ClassificationMethods` (class in
`src.predictive_model.classification.models`), 46
`ClassificationMetrics` (class in
`src.evaluation.models`), 24
`classificationmetrics`
`(src.evaluation.models.Evaluation attribute)`,
25
`ClassificationMetrics.DoesNotExist`, 24
`ClassificationMetrics.MultipleObjectsReturned`,
24
`classificationmetrics_ptr`
`(src.evaluation.models.BinaryClassificationMetrics`
`attribute)`, 23
`classificationmetrics_ptr`
`(src.evaluation.models.MulticlassClassificationMetrics`
`attribute)`, 26
`classificationmetrics_ptr_id`
`(src.evaluation.models.BinaryClassificationMetrics`
`attribute)`, 23
`classificationmetrics_ptr_id`
`(src.evaluation.models.MulticlassClassificationMetrics`
`attribute)`, 26
`ClassificationMetricType` (`src.jobs.models.ModelType` attribute), 35
`cluster_data()` (`src.clustering.clustering.Clustering`
`method`), 13
`CLUSTERER` (`src.jobs.models.ModelType` attribute), 35
`Clustering` (class in `src.clustering.clustering`), 13
`Clustering` (class in `src.clustering.models`), 13
`Clustering` (`src.jobs.models.Job` attribute), 32
`Clustering.DoesNotExist`, 13
`Clustering.MultipleObjectsReturned`, 13
`Clustering_ptr_id` (`src.jobs.models.Job` attribute), 32
`clustering_kmeans()` (in module
`src.clustering.methods_default_config`), 13
`clustering_method`
`(src.clustering.models.Clustering attribute)`, 13
`clustering_ptr` (`src.clustering.models.KMeans` at-
tribute), 15
`clustering_ptr` (`src.clustering.models.NoCluster`
attribute), 16
`clustering_ptr_id` (`src.clustering.models.KMeans`
attribute), 15
`clustering_ptr_id`
`(src.clustering.models.NoCluster attribute)`, 16
`ClusteringConfig` (class in `src.clustering.apps`), 13
`ClusteringMethods` (class in `src.clustering.models`),
14
`CommonConfig` (class in `src.common.apps`), 16
`CommonModel` (class in `src.common.models`), 16
`CommonModel.Meta` (class in `src.common.models`), 16
`COMPLETED` (`src.jobs.models.JobStatuses` attribute), 34
`completed` (`src.runtime.models.XTrace` attribute), 69
`COMPLEX` (`src.encoding.models.ValueEncodings` at-
tribute), 22
`complex()` (in module
`src.encoding.complex_last_payload`), 18
`compute_label_columns()` (in module
`src.labelling.common`), 36
`config` (`src.runtime.models.XEvent` attribute), 67
`config` (`src.runtime.models.XLog` attribute), 68
`config` (`src.runtime.models.XTrace` attribute), 69
`copy_x` (`src.clustering.models.KMeans` attribute), 15
`create_log()` (in module `src.logs.log_service`), 38
`create_models` (`src.jobs.models.Job` attribute), 32
`create_properties()` (in module
`src.logs.log_service`), 38
`create_test_clustering()` (in module
`src.utils.tests_utils`), 77
`create_test_encoding()` (in module

[src.utils.tests_utils](#)), 77
[create_test_hyperparameter_optimizer\(\)](#)
 (in module [src.utils.tests_utils](#)), 78
[create_test_job\(\)](#) (in module [src.utils.tests_utils](#)),
 78
[create_test_labelling\(\)](#) (in module
[src.utils.tests_utils](#)), 78
[create_test_log\(\)](#) (in module [src.utils.tests_utils](#)),
 78
[create_test_predictive_model\(\)](#) (in module
[src.utils.tests_utils](#)), 78
[create_test_split\(\)](#) (in module
[src.utils.tests_utils](#)), 78
[create_unique_name\(\)](#) (in module
[src.utils.file_service](#)), 76
[CREATED](#) ([src.jobs.models.JobStatuses](#) attribute), 34
[created_date](#) ([src.jobs.models.Job](#) attribute), 32
[CreateSplitSerializer](#) (class in
[src.split.serializers](#)), 74
[CreateSplitSerializer.Meta](#) (class in
[src.split.serializers](#)), 74
[CustomJSONEncoder](#) (class in
[src.jobs.json_renderer](#)), 31

D

[data_encoding](#) ([src.encoding.models.Encoding](#) at-
 tribute), 21
[DataEncoder](#) (class in
[src.encoding.encoding_parser](#)), 18
[DataEncoder.DataTypes](#) (class in
[src.encoding.encoding_parser](#)), 18
[DataEncodings](#) (class in [src.encoding.models](#)), 20
[DECISION_TREE](#) ([src.predictive_model.classification.models.ClassificationModel](#)
 attribute), 46
[DecisionTree](#) (class in
[src.predictive_model.classification.models](#)), 46
[decisiontree](#) ([src.predictive_model.classification.models.ClassificationModel](#)
 attribute), 44
[DecisionTree.DoesNotExist](#), 47
[DecisionTree.MultipleObjectsReturned](#),
 47
[DemoReplayer](#) (class in [src.runtime.models](#)), 67
[DemoReplayer.DoesNotExist](#), 67
[DemoReplayer.MultipleObjectsReturned](#),
 67
[denormalize_predictions\(\)](#)
 ([src.encoding.encoding_parser.EncodingParser](#)
 method), 20
[dropout_rate](#) ([src.predictive_model.classification.models.ClassificationModel](#)
 attribute), 50
[dropout_rate](#) ([src.predictive_model.regression.models.RegressionModel](#)
 attribute), 57
[dump_to_cache\(\)](#) (in module [src.cache.cache](#)), 9

[duplicate_orm_row\(\)](#) (in module
[src.utils.django_orm](#)), 75
[DURATION](#) ([src.labelling.models.LabelTypes](#) attribute),
 36
[duration](#) ([src.runtime.models.XTrace](#) attribute), 69
[duration\(\)](#) (in module [src.utils.time_metrics](#)), 78

E

[elapsed_time](#) ([src.evaluation.models.Evaluation](#) at-
 tribute), 25
[elapsed_time\(\)](#) (in module [src.utils.time_metrics](#)),
 78
[elapsed_time_id\(\)](#) (in module
[src.utils.time_metrics](#)), 78
[encode\(\)](#) ([src.encoding.encoder.Encoder](#) method), 18
[encode\(\)](#) ([src.encoding.encoding_container.EncodingContainer](#)
 static method), 18
[encode_data\(\)](#) ([src.encoding.encoding_parser.DataEncoder](#)
 method), 19
[encode_label_log\(\)](#) (in module
[src.encoding.common](#)), 17
[encode_label_logs\(\)](#) (in module
[src.encoding.common](#)), 18
[Encoder](#) (class in [src.encoding.encoder](#)), 18
[encoder_class](#) ([src.jobs.json_renderer.PalJSONRenderer](#)
 attribute), 31
[Encoding](#) (class in [src.encoding.models](#)), 20
[encoding](#) ([src.cache.models.LabelledLog](#) attribute), 11
[encoding](#) ([src.jobs.models.Job](#) attribute), 32
[Encoding.DoesNotExist](#), 20
[Encoding.MultipleObjectsReturned](#), 20
[encoding_id](#) ([src.cache.models.LabelledLog](#) at-
 tribute), 11
[encoding_id](#) ([src.jobs.models.Job](#) attribute), 32
[EncodingConfig](#) (class in [src.encoding.apps](#)), 17
[EncodingContainer](#) (class in
[src.encoding.encoding_container](#)), 18
[EncodingParser](#) (class in
[src.encoding.encoding_parser](#)), 19
[epsilon](#) ([src.predictive_model.classification.models.SGDClassifier](#)
 attribute), 53
[error](#) ([src.jobs.models.Job](#) attribute), 32
[ERROR](#) ([src.jobs.models.JobStatuses](#) attribute), 34
[error](#) ([src.runtime.models.XTrace](#) attribute), 69
[eta0](#) ([src.predictive_model.classification.models.Perceptron](#)
 attribute), 51
[eta0](#) ([src.predictive_model.classification.models.SGDClassifier](#)
 attribute), 53
[EpsilonNormalizer](#) (class in [src.evaluation.models](#)), 25
[evaluation](#) ([src.jobs.models.Job](#) attribute), 32
[Evaluation.DoesNotExist](#), 25
[Evaluation.MultipleObjectsReturned](#), 25
[evaluation_id](#) ([src.jobs.models.Job](#) attribute), 32

[evaluation_ptr \(src.evaluation.models.ClassificationMetrics attribute\), 24](#)
[evaluation_ptr \(src.evaluation.models.RegressionMetrics attribute\), 26](#)
[evaluation_ptr \(src.evaluation.models.TimeSeriesPredictionMetrics attribute\), 27](#)
[evaluation_ptr_id \(src.evaluation.models.ClassificationMetrics attribute\), 24](#)
[evaluation_ptr_id \(src.evaluation.models.RegressionMetrics attribute\), 26](#)
[evaluation_ptr_id \(src.evaluation.models.TimeSeriesPredictionMetrics attribute\), 27](#)
[EvaluationConfig \(class in src.evaluation.apps\), 23](#)
[event_executions \(\) \(in module src.utils.log_metrics\), 76](#)
[events_by_date \(\) \(in module src.utils.log_metrics\), 76](#)
[events_in_trace \(\) \(in module src.utils.log_metrics\), 76](#)

F

[f1_score \(src.evaluation.models.ClassificationMetrics attribute\), 24](#)
[F1SCORE \(src.hyperparameter_optimization.models.HyperOptLosses attribute\), 29](#)
[false_negative \(src.evaluation.models.BinaryClassificationMetrics attribute\), 23](#)
[FALSE_NEGATIVE \(src.hyperparameter_optimization.models.HyperOptLosses attribute\), 29](#)
[false_positive \(src.evaluation.models.BinaryClassificationMetrics attribute\), 23](#)
[FALSE_POSITIVE \(src.hyperparameter_optimization.models.HyperOptLosses attribute\), 29](#)
[features \(src.encoding.models.Encoding attribute\), 21](#)
[fields \(src.jobs.serializers.JobSerializer.Meta attribute\), 35](#)
[fields \(src.logs.serializers.LogSerializer.Meta attribute\), 40](#)
[fields \(src.runtime.serializers.TraceSerializer.Meta attribute\), 71](#)
[fields \(src.split.serializers.CreateSplitSerializer.Meta attribute\), 74](#)
[fields \(src.split.serializers.SplitSerializer.Meta attribute\), 74](#)
[first_event \(src.runtime.models.XTrace attribute\), 69](#)
[fit \(\) \(src.clustering.clustering.Clustering method\), 13](#)
[fit \(\) \(src.predictive_model.classification.custom_classification_models.NNClassifier method\), 41](#)
[fit \(\) \(src.predictive_model.regression.custom_regression_models.NNRegressor method\), 55](#)

[fit_intercept \(src.predictive_model.time_series_prediction.custom_time_series_prediction_models.TimeSeriesPrediction method\), 62](#)
[fit_intercept \(src.predictive_model.classification.models.Perceptron attribute\), 51](#)
[fit_intercept \(src.predictive_model.classification.models.SGDClassifier attribute\), 53](#)
[fit_intercept \(src.predictive_model.regression.models.Lasso attribute\), 56](#)
[fit_intercept \(src.predictive_model.regression.models.Linear attribute\), 57](#)
[fit_prior \(src.predictive_model.classification.models.NaiveBayes attribute\), 49](#)
[FREQUENCY \(src.encoding.models.ValueEncodings attribute\), 22](#)
[frequency \(\) \(in module src.encoding.boolean_frequency\), 17](#)

G

[generate \(\) \(in module src.jobs.job_creator\), 31](#)
[generate_labelling \(\) \(in module src.jobs.job_creator\), 31](#)
[get \(\) \(src.logs.views.LogDetail method\), 40](#)
[get \(\) \(src.logs.views.LogList method\), 40](#)
[get \(\) \(src.logs.views.SplitDetail method\), 41](#)
[get \(\) \(src.split.views.SplitList method\), 75](#)
[get_activation_display \(\) \(src.predictive_model.classification.models.NeuralNetwork method\), 50](#)
[get_activation_display \(\) \(src.predictive_model.regression.models.NeuralNetwork method\), 58](#)
[get_additional_columns \(\) \(in module src.utils.event_attributes\), 75](#)
[get_algorithm_display \(\) \(src.clustering.models.KMeans method\), 15](#)
[get_algorithm_type_display \(\) \(src.hyperparameter_optimization.models.HyperOpt method\), 28](#)
[get_auc \(\) \(in module src.utils.result_metrics\), 77](#)
[get_clustering_method_display \(\) \(src.clustering.models.Clustering method\), 14](#)
[get_config \(\) \(src.jobs.serializers.JobSerializer method\), 35](#)
[get_confusion_matrix \(\) \(in module src.utils.result_metrics\), 77](#)
[get_data_encoding_display \(\) \(src.encoding.models.Encoding method\), 21](#)
[get_event_attributes \(\) \(in module src.cache.cache\), 9](#)
[get_event_attributes \(\) \(in module src.utils.event_attributes\), 75](#)

`get_full_dict()` (*src.common.models.CommonModel* method), 16
`get_global_event_attributes()` (in module *src.utils.event_attributes*), 75
`get_global_trace_attributes()` (in module *src.utils.event_attributes*), 75
`get_init_display()` (*src.clustering.models.KMeans* method), 15
`get_intercase_attributes()` (in module *src.labelling.common*), 36
`get_labelled_logs()` (in module *src.cache.cache*), 9
`get_leaf_prediction_display()` (*src.predictive_model.classification.models.AdaptiveHoeffdingTree* method), 43
`get_leaf_prediction_display()` (*src.predictive_model.classification.models.HoeffdingTree* method), 48
`get_learning_rate_display()` (*src.predictive_model.classification.models.SGDClassifier* method), 53
`get_loaded_logs()` (in module *src.cache.cache*), 9
`get_log()` (in module *src.utils.file_service*), 76
`get_log_stats()` (in module *src.logs.views*), 41
`get_loss_display()` (*src.predictive_model.classification.models.SGDClassifier* method), 54
`get_method_config()` (in module *src.core.common*), 17
`get_n_classes_x()` (*src.encoding.encoding_parser.DataEncoder* method), 19
`get_n_classes_x()` (*src.encoding.encoding_parser.EncodingParser* method), 20
`get_next_by_created_date()` (*src.jobs.models.Job* method), 33
`get_next_by_modified_date()` (*src.jobs.models.Job* method), 33
`get_numerical_limits()` (*src.encoding.encoding_parser.DataEncoder* method), 19
`get_optimization_method_display()` (*src.hyperparameter_optimization.models.HyperparameterOptimization* method), 29
`get_penalty_display()` (*src.predictive_model.classification.models.Perceptron* method), 51
`get_penalty_display()` (*src.predictive_model.classification.models.SGDClassifier* method), 54
`get_performance_metric_display()` (*src.hyperparameter_optimization.models.HyperOpt* method), 28
`get_precompute_distances_display()` (*src.clustering.models.KMeans* method), 15
`get_prediction_method_config()` (in module *src.jobs.job_creator*), 31
`get_predictive_model_display()` (*src.predictive_model.models.PredictiveModel* method), 65
`get_previous_by_created_date()` (*src.jobs.models.Job* method), 33
`get_previous_by_modified_date()` (*src.jobs.models.Job* method), 33
`get_rnn_type_display()` (*src.predictive_model.time_series_prediction.models.RecurrentNeuralNetwork* method), 63
`get_split_criterion_display()` (*src.predictive_model.classification.models.AdaptiveHoeffdingTree* method), 43
`get_split_criterion_display()` (*src.predictive_model.classification.models.HoeffdingTree* method), 48
`get_splitting_method_display()` (*src.split.models.Split* method), 72
`get_status_display()` (*src.jobs.models.Job* method), 33
`get_task_generation_type_display()` (*src.encoding.models.Encoding* method), 21
`get_threshold_type_display()` (*src.labelling.models.Labelling* method), 37
`get_training_log()` (*src.split.serializers.SplitSerializer* method), 74
`get_type_display()` (*src.jobs.models.Job* method), 33
`get_type_display()` (*src.labelling.models.Labelling* method), 37
`get_type_display()` (*src.split.models.Split* method), 72
`get_value_encoding_display()` (*src.encoding.models.Encoding* method), 21
`get_weights_display()` (*src.predictive_model.classification.models.Knn* method), 49
`grace_period` (*src.predictive_model.classification.models.AdaptiveHoeffdingTree* attribute), 43
`grace_period` (*src.predictive_model.classification.models.HoeffdingTree* attribute), 48
`HOEFFDING_TREE` (*src.predictive_model.classification.models.Classification* attribute), 46

HoeffdingTree (class in `src.jobs.models.Job` attribute), 33
 `src.predictive_model.classification.models`), 47
 hoeffdingtree (`src.predictive_model.classification.models.ClassificationModels` attribute), 44
 HoeffdingTree.DoesNotExist, 47
 HoeffdingTree.MultipleObjectsReturned, 47
 HyperOpt (class in `src.hyperparameter_optimization.models`), 28
 hyperopt (`src.hyperparameter_optimization.models.HyperparameterOptimizationMethods` attribute), 30
 HYPEROPT (`src.hyperparameter_optimization.models.HyperparameterOptimizationMethods` attribute), 30
 HyperOpt.DoesNotExist, 28
 HyperOpt.MultipleObjectsReturned, 28
 HyperOptAlgorithms (class in `src.hyperparameter_optimization.models`), 29
 HyperOptLosses (class in `src.hyperparameter_optimization.models`), 29
 hyperparameter_optimization_hyperopt ()
 (in module `src.hyperparameter_optimization.methods.default_config`), 28
 hyperparameter_optimizer (`src.jobs.models.Job` attribute), 33
 hyperparameter_optimizer_id
 (`src.jobs.models.Job` attribute), 33
 HyperparameterOptimization (class in `src.hyperparameter_optimization.models`), 29
 HyperparameterOptimization.DoesNotExist, is_all_in_one () (`src.encoding.encoding_container.EncodingContainer` method), 18
 HyperparameterOptimization.MultipleObjectsReturned, 29
 hyperparameteroptimization_ptr
 (`src.hyperparameter_optimization.models.HyperOpt` attribute), 28
 hyperparameteroptimization_ptr_id
 (`src.hyperparameter_optimization.models.HyperOpt` attribute), 29
 HyperparameterOptimizationConfig (class in `src.hyperparameter_optimization.apps`), 28
 HyperparameterOptimizationMethods (class in `src.hyperparameter_optimization.models`), 30

 I
 id (`src.cache.models.Cache` attribute), 10
 id (`src.clustering.models.Clustering` attribute), 14
 id (`src.encoding.models.Encoding` attribute), 21
 id (`src.evaluation.models.Evaluation` attribute), 25
 id (`src.hyperparameter_optimization.models.HyperparameterOptimizationMethods` attribute), 30

 J
 Job (class in `src.jobs.models`), 31
 Job.DoesNotExist, 31
 Job.MultipleObjectsReturned, 31
 job_set (`src.clustering.models.Clustering` attribute), 14
 job_set (`src.encoding.models.Encoding` attribute), 21
 job_set (`src.evaluation.models.Evaluation` attribute), 25
 job_set (`src.hyperparameter_optimization.models.HyperparameterOptimizationMethods` attribute), 30

job_set (*src.labelling.models.Labelling* attribute), 37
 job_set (*src.predictive_model.models.PredictiveModel* attribute), 66
 job_set (*src.split.models.Split* attribute), 72
 JobsConfig (class in *src.jobs.apps*), 31
 JobSerializer (class in *src.jobs.serializers*), 35
 JobSerializer.Meta (class in *src.jobs.serializers*), 35
 JobStatuses (class in *src.jobs.models*), 34
 JobTypes (class in *src.jobs.models*), 35

K

KMeans (class in *src.clustering.models*), 14
 kmeans (*src.clustering.models.Clustering* attribute), 14
 KMEANS (*src.clustering.models.ClusteringMethods* attribute), 14
 KMeans.DoesNotExist, 15
 KMeans.MultipleObjectsReturned, 15
 Knn (class in *src.predictive_model.classification.models*), 48
 knn (*src.predictive_model.classification.models.Classification* attribute), 45
 KNN (*src.predictive_model.classification.models.ClassificationMethods* attribute), 46
 Knn.DoesNotExist, 48
 Knn.MultipleObjectsReturned, 48

L

l1_ratio (*src.predictive_model.classification.models.SGDClassifier* attribute), 54
 LABEL_ENCODER (*src.encoding.models.DataEncodings* attribute), 20
 LabelContainer (class in *src.labelling.label_container*), 36
 LabelledLog (class in *src.cache.models*), 10
 labelledlog (*src.cache.models.Cache* attribute), 10
 LabelledLog.DoesNotExist, 10
 LabelledLog.MultipleObjectsReturned, 10
 labelledlog_set (*src.encoding.models.Encoding* attribute), 21
 labelledlog_set (*src.labelling.models.Labelling* attribute), 37
 labelledlog_set (*src.split.models.Split* attribute), 72
 Labelling (class in *src.labelling.models*), 37
 labelling (*src.cache.models.LabelledLog* attribute), 11
 labelling (*src.jobs.models.Job* attribute), 33
 LABELLING (*src.jobs.models.JobTypes* attribute), 35
 Labelling.DoesNotExist, 37
 Labelling.MultipleObjectsReturned, 37
 labelling_id (*src.cache.models.LabelledLog* attribute), 11
 labelling_id (*src.jobs.models.Job* attribute), 34

LabellingConfig (class in *src.labelling.apps*), 36
 LabelTypes (class in *src.labelling.models*), 36
 Lasso (class in *src.predictive_model.regression.models*), 56
 lasso (*src.predictive_model.regression.models.Regression* attribute), 59
 LASSO (*src.predictive_model.regression.models.RegressionMethods* attribute), 60
 Lasso.DoesNotExist, 56
 Lasso.MultipleObjectsReturned, 56
 last_event (*src.runtime.models.XTrace* attribute), 70
 LAST_PAYLOAD (*src.encoding.models.ValueEncodings* attribute), 22
 last_payload() (in module *src.encoding.complex_last_payload*), 18
 leaf_prediction (*src.predictive_model.classification.models.Adaptive* attribute), 43
 leaf_prediction (*src.predictive_model.classification.models.Hoeffding* attribute), 48
 learning_rate (*src.predictive_model.classification.models.SGDClassifier* attribute), 54
 Linear (class in *src.predictive_model.regression.models*), 57
 linear (*src.predictive_model.regression.models.Regression* attribute), 59
 LINEAR (*src.predictive_model.regression.models.RegressionMethods* attribute), 60
 Linear.DoesNotExist, 57
 Linear.MultipleObjectsReturned, 57
 load_from_cache() (in module *src.cache.cache*), 10
 load_model() (*src.clustering.clustering.Clustering* class method), 13
 LoadedLog (class in *src.cache.models*), 11
 loadedlog (*src.cache.models.Cache* attribute), 10
 LoadedLog.DoesNotExist, 12
 LoadedLog.MultipleObjectsReturned, 12
 loadedlog_set (*src.split.models.Split* attribute), 72
 Log (class in *src.logs.models*), 38
 log (*src.logs.models.Log* attribute), 39
 Log.DoesNotExist, 38
 Log.MultipleObjectsReturned, 38
 LogDetail (class in *src.logs.views*), 40
 LogList (class in *src.logs.views*), 40
 LogsConfig (class in *src.logs.apps*), 38
 LogSerializer (class in *src.logs.serializers*), 40
 LogSerializer.Meta (class in *src.logs.serializers*), 40
 loss (*src.predictive_model.classification.models.SGDClassifier* attribute), 54

M

mae (*src.evaluation.models.RegressionMetrics* attribute), 27

name (*src.hyperparameter_optimization.apps.HyperparameterOptimizationConfig* attribute), 28
 name (*src.jobs.apps.JobsConfig* attribute), 31
 name (*src.labelling.apps.LabellingConfig* attribute), 36
 name (*src.logs.apps.LogsConfig* attribute), 38
 name (*src.logs.models.Log* attribute), 39
 name (*src.predictive_model.apps.PredictiveModelConfig* attribute), 65
 name (*src.predictive_model.classification.apps.ClassificationConfig* attribute), 41
 name (*src.predictive_model.regression.apps.RegressionConfig* attribute), 55
 name (*src.predictive_model.time_series_prediction.apps.TimeSeriesPredictionConfig* attribute), 62
 name (*src.runtime.apps.RuntimeConfig* attribute), 67
 name (*src.runtime.models.XTrace* attribute), 70
 name (*src.split.apps.SplitConfig* attribute), 72
 nb_threshold (*src.predictive_model.classification.models.AdaptiveHoeffdingTree* attribute), 44
 nb_threshold (*src.predictive_model.classification.models.HoeffdingTree* attribute), 48
 NeuralNetwork (class in *src.predictive_model.classification.models*), 50
 NeuralNetwork (class in *src.predictive_model.regression.models*), 57
 neuralnetwork (*src.predictive_model.classification.models.Classification* attribute), 45
 neuralnetwork (*src.predictive_model.regression.models.Regression* attribute), 59
 NeuralNetwork.DoesNotExist, 50, 57
 NeuralNetwork.MultipleObjectsReturned, 50, 57
 new_trace_start () (in module *src.utils.log_metrics*), 76
 NEXT_ACTIVITY (*src.labelling.models.LabelTypes* attribute), 36
 next_event_name () (in module *src.labelling.common*), 36
 nlevenshtein (*src.evaluation.models.TimeSeriesPredictionMetrics* attribute), 27
 NN (*src.predictive_model.classification.models.ClassificationMethods* attribute), 46
 NN (*src.predictive_model.regression.models.RegressionMethods* attribute), 60
 NNClassifier (class in *src.predictive_model.classification.custom_classification_models*), 41
 NNRegressor (class in *src.predictive_model.regression.custom_regression_models*), 55
 NO_CLUSTER (*src.clustering.models.ClusteringMethods* attribute), 14
 NO_LABEL (*src.labelling.models.LabelTypes* attribute), 36
 NoCluster (class in *src.clustering.models*), 16
 nocluster (*src.clustering.models.Clustering* attribute), 14
 NoCluster.DoesNotExist, 16
 NoCluster.MultipleObjectsReturned, 16
 NONE (*src.hyperparameter_optimization.models.HyperparameterOptimizationConfig* attribute), 30
 normalize (*src.predictive_model.regression.models.Lasso* attribute), 56
 normalize (*src.predictive_model.regression.models.Linear* attribute), 57
 none (*src.encoding.encoding_parser.DataEncoder.DataTypes* attribute), 19

O

objects (*src.cache.models.Cache* attribute), 10
 objects (*src.clustering.models.Clustering* attribute), 14
 objects (*src.encoding.models.Encoding* attribute), 21
 objects (*src.evaluation.models.Evaluation* attribute), 25
 objects (*src.hyperparameter_optimization.models.HyperparameterOptimizationConfig* attribute), 30
 objects (*src.jobs.models.Job* attribute), 34
 objects (*src.labelling.models.Labelling* attribute), 37
 objects (*src.logs.models.Log* attribute), 39
 objects (*src.predictive_model.models.PredictiveModel* attribute), 66
 objects (*src.runtime.models.DemoReplayer* attribute), 67
 objects (*src.runtime.models.XEvent* attribute), 68
 objects (*src.runtime.models.XLog* attribute), 68
 objects (*src.runtime.models.XTrace* attribute), 70
 objects (*src.split.models.Split* attribute), 73
 ONE_HOT_ENCODER (*src.encoding.models.DataEncodings* attribute), 20
 ONLY_THIS (*src.encoding.models.TaskGenerationTypes* attribute), 22
 optimization_method (*src.hyperparameter_optimization.models.HyperparameterOptimizationConfig* attribute), 30
 original_log (*src.logs.models.Log* attribute), 39
 original_log (*src.split.models.Split* attribute), 73
 original_log_id (*src.split.models.Split* attribute), 73

P

padding (*src.encoding.models.Encoding* attribute), 21
 PajsonRenderer (class in *src.jobs.json_renderer*), 31
 parse_targets () (*src.encoding.encoding_parser.EncodingParser* method), 20

parse_testing_dataset ()
 (src.encoding.encoding_parser.EncodingParser method), 20
 parse_training_dataset ()
 (src.encoding.encoding_parser.EncodingParser method), 20
 path (src.logs.models.Log attribute), 39
 penalty (src.predictive_model.classification.models.Perceptron attribute), 51
 penalty (src.predictive_model.classification.models.SGDClassifier attribute), 54
 Perceptron (class in src.predictive_model.classification.models), 50
 perceptron (src.predictive_model.classification.models.ClassificationModel attribute), 45
 PERCEPTRON (src.predictive_model.classification.models.ClassificationModel attribute), 46
 Perceptron.DoesNotExist, 51
 Perceptron.MultipleObjectsReturned, 51
 performance_metric
 (src.hyperparameter_optimization.models.HyperOpt attribute), 29
 post () (src.logs.views.LogList method), 40
 post () (src.split.views.SplitList static method), 75
 power_t (src.predictive_model.classification.models.SGDClassifier attribute), 54
 precision (src.evaluation.models.ClassificationMetrics attribute), 24
 PRECISION (src.hyperparameter_optimization.models.HyperOptLoss attribute), 29
 precompute_distances
 (src.clustering.models.KMeans attribute), 15
 predict () (src.clustering.clustering.Clustering method), 13
 predict () (src.predictive_model.classification.custom_classification_models.NNClassifier method), 42
 predict () (src.predictive_model.regression.custom_regression_models.NNRegressor method), 55
 predict () (src.predictive_model.time_series_prediction.custom_time_series_prediction_models.RNNTimesSeriesPredictor method), 62
 predict_proba () (src.predictive_model.classification.custom_classification_models.NNClassifier method), 42
 PREDICTION (src.jobs.models.JobTypes attribute), 35
 prediction_method
 (src.predictive_model.models.PredictiveModel attribute), 66
 predictive_model (src.jobs.models.Job attribute), 34
 predictive_model (src.predictive_model.models.PredictiveModel attribute), 66
 predictive_model_id (src.jobs.models.Job attribute), 34
 PredictiveModel (class in src.predictive_model.models), 65
 PredictiveModel.DoesNotExist, 65
 PredictiveModel.MultipleObjectsReturned, 65
 predictivemodel_ptr
 (src.predictive_model.classification.models.Classification attribute), 45
 predictivemodel_ptr
 (src.predictive_model.regression.models.Regression attribute), 59
 predictivemodel_ptr
 (src.predictive_model.time_series_prediction.models.TimeSeriesPrediction attribute), 64
 predictivemodel_ptr_id
 (src.predictive_model.classification.models.Classification attribute), 45
 predictivemodel_ptr_id
 (src.predictive_model.regression.models.Regression attribute), 60
 predictivemodel_ptr_id
 (src.predictive_model.time_series_prediction.models.TimeSeriesPrediction attribute), 64
 PredictiveModelConfig (class in src.predictive_model.apps), 65
 PredictiveModels (class in src.predictive_model.models), 66
 prefix_length (src.encoding.models.Encoding attribute), 22
 put_labelled_logs () (in module src.cache.cache), 10
 put_loaded_logs () (in module src.cache.cache), 10
 queryset (src.logs.views.LogDetail attribute), 40
 queryset (src.logs.views.LogList attribute), 40
 queryset (src.logs.views.SplitDetail attribute), 41
 queryset (src.split.views.SplitList attribute), 75
 RandomForest (class in src.predictive_model.classification.models), 52
 RANDOM_FOREST (src.predictive_model.classification.models.Classification attribute), 46
 RANDOM_FOREST (src.predictive_model.regression.models.RegressionMethod attribute), 60
 RANDOM_SEARCH (src.hyperparameter_optimization.models.HyperOptAlgorithm attribute), 29
 random_state (src.clustering.models.KMeans attribute), 15
 random_state (src.predictive_model.regression.models.RandomForest attribute), 58
 RandomForest (class in src.predictive_model.classification.models), 52

RandomForest (class in attribute), 56
 src.predictive_model.regression.models), regression_ptr (src.predictive_model.regression.models.Linear
 58 attribute), 57
 randomforest (src.predictive_model.classification.models.LassoClassification_ptr (src.predictive_model.regression.models.NeuralNetwork
 attribute), 45 attribute), 58
 randomforest (src.predictive_model.regression.models.Regression_ptr (src.predictive_model.regression.models.RandomFore
 attribute), 60 attribute), 58
 RandomForest.DoesNotExist, 52, 58 regression_ptr (src.predictive_model.regression.models.XGBoost
 RandomForest.MultipleObjectsReturned, attribute), 61
 52, 58 regression_ptr_id
 real_log (src.logs.models.Log attribute), 39 (src.predictive_model.regression.models.Lasso
 real_log (src.runtime.models.XLog attribute), 68 attribute), 56
 real_log (src.runtime.models.XTrace attribute), 70 regression_ptr_id
 real_log_id (src.runtime.models.XLog attribute), 68 (src.predictive_model.regression.models.Linear
 recall (src.evaluation.models.ClassificationMetrics at- attribute), 57
 tribute), 25 regression_ptr_id
 RECALL (src.hyperparameter_optimization.models.HyperOptLosses (src.predictive_model.regression.models.NeuralNetwork
 attribute), 29 attribute), 58
 RecurrentNeuralNetwork (class in regression_ptr_id
 src.predictive_model.time_series_prediction.models), (src.predictive_model.regression.models.RandomForest
 63 attribute), 59
 recurrentneuralnetwork regression_ptr_id
 (src.predictive_model.time_series_prediction.models.TimeSeriesPrediction_ptr (src.predictive_model.regression.models.XGBoost
 attribute), 64 attribute), 61
 RecurrentNeuralNetwork.DoesNotExist, 63 regression_random_forest () (in module
 RecurrentNeuralNetwork.MultipleObjectsReturned, src.predictive_model.regression.methods_default_config),
 63 56
 reg_actual (src.runtime.models.XTrace attribute), 70 regression_single_log () (in module
 reg_model (src.runtime.models.XTrace attribute), 70 src.predictive_model.regression.regression), 61
 reg_model_id (src.runtime.models.XTrace attribute), regression_xgboost () (in module
 70 src.predictive_model.regression.methods_default_config),
 reg_results (src.runtime.models.XTrace attribute), 56
 70 RegressionConfig (class in
 Regression (class in src.predictive_model.regression.apps), 55
 src.predictive_model.regression.models), RegressionMethods (class in
 59 src.predictive_model.regression.models),
 regression (src.predictive_model.models.PredictiveModel 60
 attribute), 66 RegressionMetrics (class in
 REGRESSION (src.predictive_model.models.PredictiveModels src.evaluation.models), 26
 attribute), 66 regressionmetrics
 regression () (in module (src.evaluation.models.Evaluation attribute),
 src.predictive_model.regression.regression), 61 25
 Regression.DoesNotExist, 59 RegressionMetrics.DoesNotExist, 26
 Regression.MultipleObjectsReturned, 59 RegressionMetrics.MultipleObjectsReturned,
 regression_lasso () (in module 26
 src.predictive_model.regression.methods_default_config), 56
 regression_linear () (in module attribute), 37
 src.predictive_model.regression.methods_default_config), 56
 regression_nn () (in module remaining_time () (in module
 src.predictive_model.regression.methods_default_config), src.utils.time_metrics), 78
 56 remaining_time_id () (in module
 regression_ptr (src.predictive_model.regression.models.Lasso attribute), 44 remove_poor_atts (src.predictive_model.classification.models.Adaptive

remove_poor_atts(*src.predictive_model.classification.models.HoeffdingTree* attribute), 48
 reset() (*src.predictive_model.classification.custom_classification.models.NNClassifier* attribute), 42
 reset() (*src.predictive_model.regression.custom_regression.models.NNRegressor* attribute), 56
 resources_by_date() (in module *src.utils.log_metrics*), 77
 results(*src.labelling.models.Labelling* attribute), 37
 rmse(*src.evaluation.models.RegressionMetrics* attribute), 27
 RMSE(*src.hyperparameter_optimization.models.HyperOptLosses* attribute), 29
 RNN(*src.predictive_model.time_series_prediction.models.TimeSeriesPredictionMethods* attribute), 64
 rnn_type(*src.predictive_model.time_series_prediction.models.RecurrentNeuralNetwork* attribute), 63
 RNNTimeseriesPredictor (class in *src.predictive_model.time_series_prediction.custom_time_series_prediction.models*), 62
 rscore(*src.evaluation.models.RegressionMetrics* attribute), 27
 RSCORE(*src.hyperparameter_optimization.models.HyperOptLosses* attribute), 29
 RUNNING(*src.jobs.models.JobStatuses* attribute), 34
 running(*src.runtime.models.DemoReplayer* attribute), 67
 RuntimeConfig (class in *src.runtime.apps*), 67

S

save_result() (in module *src.utils.file_service*), 76
 serializer_class(*src.logs.views.LogDetail* attribute), 40
 serializer_class(*src.logs.views.LogList* attribute), 40
 serializer_class(*src.logs.views.SplitDetail* attribute), 41
 serializer_class(*src.split.views.SplitList* attribute), 75
 SGDClassifier (class in *src.predictive_model.classification.models*), 52
 sgdcassifier(*src.predictive_model.classification.models.SGDClassifier* attribute), 46
 SGDCLASSIFIER(*src.predictive_model.classification.models.SGDClassifierMethods* attribute), 46
 SGDClassifier.DoesNotExist, 53
 SGDClassifier.MultipleObjectsReturned, 53
 shuffle(*src.predictive_model.classification.models.Perceptron* attribute), 51
 SIMPLE_INDEX(*src.encoding.models.ValueEncodings* attribute), 22
 simple_index() (in module *src.encoding.simple_index*), 22
 split(*src.cache.models.LabellingLog* attribute), 11
 split(*src.cache.models.LoadedLog* attribute), 12
 split(*src.jobs.models.Job* attribute), 34
 split(*src.predictive_model.classification.models.NNClassifier* attribute), 72
 Split.MultipleObjectsReturned, 72
 split_confidence(*src.predictive_model.classification.models.Adaptive* attribute), 44
 split_confidence(*src.predictive_model.classification.models.HoeffdingTree* attribute), 48
 split_criterion(*src.predictive_model.classification.models.Adaptive* attribute), 44
 split_criterion(*src.predictive_model.classification.models.HoeffdingTree* attribute), 48
 SPLIT_DOUBLE(*src.split.models.SplitTypes* attribute), 74
 split_id(*src.cache.models.LabellingLog* attribute), 11
 split_id(*src.cache.models.LoadedLog* attribute), 12
 split_id(*src.jobs.models.Job* attribute), 34
 SPLIT_RANDOM(*src.split.models.SplitOrderingMethods* attribute), 74
 SPLIT_SEQUENTIAL(*src.split.models.SplitOrderingMethods* attribute), 74
 SPLIT_SINGLE(*src.split.models.SplitTypes* attribute), 74
 SPLIT_STRICT_TEMPORAL(*src.split.models.SplitOrderingMethods* attribute), 74
 SPLIT_TEMPORAL(*src.split.models.SplitOrderingMethods* attribute), 74
 SplitConfig (class in *src.split.apps*), 72
 SplitDetail (class in *src.logs.views*), 41
 SplitList (class in *src.split.views*), 75
 SplitOrderingMethods (class in *src.split.models*), 74
 SplitSerializer (class in *src.split.serializers*), 74
 SplitSerializer.Meta (class in *src.split.serializers*), 74
 splitting_method(*src.split.models.Split* attribute), 73
 SplitTypes (class in *src.split.models*), 74
 SRC (module), 79
 src.cache (module), 12
 src.cache.apps (module), 9
 src.cache.cache (module), 9
 src.cache.models (module), 10
 src.clustering (module), 16
 src.clustering.apps (module), 13
 src.clustering.clustering (module), 13
 src.clustering.methods_default_config (module), 13
 src.clustering.models (module), 13
 src.common (module), 17
 src.common.apps (module), 16

src.common.models (module), 16
src.core (module), 17
src.core.common (module), 17
src.encoding (module), 22
src.encoding.apps (module), 17
src.encoding.boolean_frequency (module), 17
src.encoding.common (module), 17
src.encoding.complex_last_payload (module), 18
src.encoding.encoder (module), 18
src.encoding.encoding_container (module), 18
src.encoding.encoding_parser (module), 18
src.encoding.models (module), 20
src.encoding.simple_index (module), 22
src.evaluation (module), 27
src.evaluation.apps (module), 23
src.evaluation.models (module), 23
src.hyperparameter_optimization (module), 30
src.hyperparameter_optimization.apps (module), 28
src.hyperparameter_optimization.hyperopt_spaces (module), 28
src.hyperparameter_optimization.methods_default_config_model (module), 28
src.hyperparameter_optimization.models (module), 28
src.jobs (module), 36
src.jobs.admin (module), 30
src.jobs.apps (module), 31
src.jobs.job_creator (module), 31
src.jobs.json_renderer (module), 31
src.jobs.models (module), 31
src.jobs.serializers (module), 35
src.jobs.ws_publisher (module), 35
src.labelling (module), 38
src.labelling.apps (module), 36
src.labelling.common (module), 36
src.labelling.label_container (module), 36
src.labelling.models (module), 36
src.logs (module), 41
src.logs.admin (module), 38
src.logs.apps (module), 38
src.logs.log_service (module), 38
src.logs.models (module), 38
src.logs.serializers (module), 40
src.logs.urls (module), 40
src.logs.views (module), 40
src.predictive_model (module), 67
src.predictive_model.apps (module), 65
src.predictive_model.classification (module), 55
src.predictive_model.classification.apps (module), 41
src.predictive_model.classification.custom_classification (module), 41
src.predictive_model.classification.methods_default_config_model (module), 42
src.predictive_model.classification.models (module), 43
src.predictive_model.models (module), 65
src.predictive_model.regression (module), 62
src.predictive_model.regression.apps (module), 55
src.predictive_model.regression.custom_regression_model (module), 55
src.predictive_model.regression.methods_default_config_model (module), 56
src.predictive_model.regression.models (module), 56
src.predictive_model.regression.regression (module), 61
src.predictive_model.time_series_prediction (module), 65
src.predictive_model.time_series_prediction.apps (module), 62
src.predictive_model.time_series_prediction.custom_classification_model (module), 62
src.predictive_model.time_series_prediction.methods_default_config_model (module), 62
src.predictive_model.time_series_prediction.models (module), 63
src.predictive_model.time_series_prediction.time_series_prediction (module), 64
src.predictive_model.time_series_prediction.TimeSeriesPrediction (module), 62
src.runtime (module), 71
src.runtime.apps (module), 67
src.runtime.models (module), 67
src.runtime.serializers (module), 71
src.split (module), 75
src.split.apps (module), 72
src.split.models (module), 72
src.split.serializers (module), 74
src.split.splitting (module), 75
src.split.urls (module), 75
src.split.views (module), 75
src.utils (module), 79
src.utils.django_orm (module), 75
src.utils.event_attributes (module), 75
src.utils.file_service (module), 76
src.utils.log_metrics (module), 76
src.utils.result_metrics (module), 77
src.utils.tests_utils (module), 77
src.utils.time_metrics (module), 78

status (*src.jobs.models.Job* attribute), 34
 std_var_events_in_log() (in module *src.utils.log_metrics*), 77

T

task_generation_type (*src.encoding.models.Encoding* attribute), 22
 TaskGenerationTypes (class in *src.encoding.models*), 22
 test_log (*src.logs.models.Log* attribute), 39
 test_log (*src.split.models.Split* attribute), 73
 test_log_id (*src.split.models.Split* attribute), 73
 test_log_path (*src.cache.models.LabelledLog* attribute), 11
 test_log_path (*src.cache.models.LoadedLog* attribute), 12
 test_size (*src.split.models.Split* attribute), 73
 threshold (*src.labelling.models.Labelling* attribute), 37
 THRESHOLD_CUSTOM (*src.labelling.models.ThresholdTypes* attribute), 38
 THRESHOLD_MEAN (*src.labelling.models.ThresholdTypes* attribute), 38
 threshold_type (*src.labelling.models.Labelling* attribute), 37
 ThresholdTypes (class in *src.labelling.models*), 38
 tie_threshold (*src.predictive_model.classification.models.AdaptiveHoeffdingTree* attribute), 44
 tie_threshold (*src.predictive_model.classification.models.AdaptiveHoeffdingTree* attribute), 48
 TIME_SERIES_PREDICTION (*src.predictive_model.models.PredictiveModels* attribute), 67
 time_series_prediction() (in module *src.predictive_model.time_series_prediction.time_series_prediction*), 64
 time_series_prediction_rnn() (in module *src.predictive_model.time_series_prediction.methods_default_config*), 62
 time_series_prediction_single_log() (in module *src.predictive_model.time_series_prediction.time_series_prediction*), 65
 TIME_SERIES_PREDICTOR (*src.jobs.models.ModelType* attribute), 35
 TimeSeriesPrediction (class in *src.predictive_model.time_series_prediction.models*), 64
 timeseriesprediction (*src.predictive_model.models.PredictiveModel* attribute), 66
 TimeSeriesPrediction.DoesNotExist, 64
 TimeSeriesPrediction.MultipleObjectsReturned, 64
 timeseriesprediction_ptr (*src.predictive_model.time_series_prediction.models.RecurrentNeuralNetwork* attribute), 63
 timeseriesprediction_ptr_id (*src.predictive_model.time_series_prediction.models.RecurrentNeuralNetwork* attribute), 63
 TimeSeriesPredictionConfig (class in *src.predictive_model.time_series_prediction.apps*), 62
 TimeSeriesPredictionMethods (class in *src.predictive_model.time_series_prediction.models*), 64
 TimeSeriesPredictionMetrics (class in *src.evaluation.models*), 27
 timeseriespredictionmetrics (*src.evaluation.models.Evaluation* attribute), 26
 TimeSeriesPredictionMetrics.DoesNotExist, 27
 TimeSeriesPredictionMetrics.MultipleObjectsReturned, 27
 TimeSeriesPredictorMixin (class in *src.predictive_model.time_series_prediction.TimeSeriesPredictorMixin*), 62
 to_dict() (*src.clustering.models.Clustering* method), 14
 to_dict() (*src.clustering.models.KMeans* method), 15
 to_dict() (*src.common.models.CommonModel* method), 16
 to_dict() (*src.encoding.models.Encoding* method), 22
 to_dict() (*src.evaluation.models.BinaryClassificationMetrics* method), 23
 to_dict() (*src.evaluation.models.ClassificationMetrics* method), 25
 to_dict() (*src.evaluation.models.Evaluation* method), 26
 to_dict() (*src.evaluation.models.RegressionMetrics* method), 27
 to_dict() (*src.evaluation.models.TimeSeriesPredictionMetrics* method), 27
 to_dict() (*src.hyperparameter_optimization.models.HyperOpt* method), 29
 to_dict() (*src.jobs.models.Job* method), 34
 to_dict() (*src.labelling.models.Labelling* method), 37
 to_dict() (*src.logs.models.Log* method), 40
 to_dict() (*src.predictive_model.classification.models.AdaptiveHoeffdingTree* method), 44
 to_dict() (*src.predictive_model.classification.models.DecisionTree* method), 47
 to_dict() (*src.predictive_model.classification.models.HoeffdingTree* method), 48
 to_dict() (*src.predictive_model.classification.models.Knn* method), 48

[method](#)), 49
[to_dict\(\)](#) ([src.predictive_model.classification.models.NaiveBayes](#) [tribute](#)), 12
[method](#)), 49
[to_dict\(\)](#) ([src.predictive_model.classification.models.NeuralNetwork](#) [tribute](#)), 12
[method](#)), 50
[to_dict\(\)](#) ([src.predictive_model.classification.models.Perceptron](#) [tribute](#)), 12
[method](#)), 51
[to_dict\(\)](#) ([src.predictive_model.classification.models.RandomForest](#) [tribute](#)), 12
[method](#)), 52
[to_dict\(\)](#) ([src.predictive_model.classification.models.SGDClassifier](#) [tribute](#)), 12
[method](#)), 54
[to_dict\(\)](#) ([src.predictive_model.classification.models.XGBoost](#) [tribute](#)), 12
[method](#)), 55
[to_dict\(\)](#) ([src.predictive_model.models.PredictiveModel](#) [tribute](#)), 12
[method](#)), 66
[to_dict\(\)](#) ([src.predictive_model.regression.models.Lasso](#) [tribute](#)), 12
[method](#)), 56
[to_dict\(\)](#) ([src.predictive_model.regression.models.Linear](#) [tribute](#)), 12
[method](#)), 57
[to_dict\(\)](#) ([src.predictive_model.regression.models.NeuralNetwork](#) [tribute](#)), 12
[method](#)), 58
[to_dict\(\)](#) ([src.predictive_model.regression.models.RandomForest](#) [tribute](#)), 12
[method](#)), 59
[to_dict\(\)](#) ([src.predictive_model.regression.models.XGBoost](#) [tribute](#)), 12
[method](#)), 61
[to_dict\(\)](#) ([src.predictive_model.time_series_prediction.models.RecurrentNeuralNetwork](#) [tribute](#)), 12
[method](#)), 63
[to_dict\(\)](#) ([src.runtime.models.DemoReplayer](#) [tribute](#)), 12
[method](#)), 67
[to_dict\(\)](#) ([src.runtime.models.XEvent](#) [tribute](#)), 12
[to_dict\(\)](#) ([src.runtime.models.XLog](#) [tribute](#)), 12
[to_dict\(\)](#) ([src.runtime.models.XTrace](#) [tribute](#)), 12
[to_dict\(\)](#) ([src.split.models.Split](#) [tribute](#)), 12
[to_one_hot\(\)](#) ([src.encoding.encoding_parser.DataEncoder](#) [tribute](#)), 12
[method](#)), 19
[tol](#) ([src.clustering.models.KMeans](#) [tribute](#)), 15
[tol](#) ([src.predictive_model.classification.models.Perceptron](#) [tribute](#)), 51
[tol](#) ([src.predictive_model.classification.models.SGDClassifier](#) [tribute](#)), 54
[TPE](#) ([src.hyperparameter_optimization.models.HyperOptAlgorithms](#) [tribute](#)), 29
[trace](#) ([src.runtime.models.XEvent](#) [tribute](#)), 68
[trace_attributes\(\)](#) ([src.runtime.models.XEvent](#) [tribute](#)), 68
[src.utils.log_metrics](#)), 77
[trace_id](#) ([src.runtime.models.XEvent](#) [tribute](#)), 68
[TraceSerializer](#) ([src.runtime.serializers](#)), 71
[TraceSerializer.Meta](#) ([src.runtime.serializers](#)), 71
[train_log](#) ([src.split.models.Split](#) [tribute](#)), 73
[train_log_id](#) ([src.split.models.Split](#) [tribute](#)), 74
[train_log_path](#) ([src.cache.models.LabelledLog](#) [tribute](#)), 11

[train_log_path](#) ([src.cache.models.LoadedLog](#) [tribute](#)), 11
[training_log](#) ([src.logs.models.Log](#) [tribute](#)), 40
[tributives](#) ([src.evaluation.models.BinaryClassificationMetrics](#) [tribute](#)), 23
[tribute](#) ([src.evaluation.models.BinaryClassificationMetrics](#) [tribute](#)), 23
[tribute](#) ([src.hyperparameter_optimization.models.HyperOptAlgorithms](#) [tribute](#)), 29
[tribute](#) ([src.evaluation.models.BinaryClassificationMetrics](#) [tribute](#)), 24
[tribute](#) ([src.hyperparameter_optimization.models.HyperOptAlgorithms](#) [tribute](#)), 29
[tribute](#) ([src.jobs.models.Job](#) [tribute](#)), 34
[tribute](#) ([src.labelling.models.Labelling](#) [tribute](#)), 38
[tribute](#) ([src.split.models.Split](#) [tribute](#)), 74
[tribute](#) ([src.encoding.models.TaskGenerationTypes](#) [tribute](#)), 22
[tribute](#) ([src.jobs.models.JobTypes](#) [tribute](#)), 35
[tribute](#) ([src.jobs.job_creator](#)), 31
[tribute](#) ([src.logs.views](#)), 41
[tribute](#) ([src.encoding.models.Encoding](#) [tribute](#)), 22
[tribute](#) ([src.encoding.models.ValueEncodings](#)), 22
[tribute](#) ([src.predictive_model.classification.models.Knn](#) [tribute](#)), 49
[tribute](#) ([src.runtime.models.XEvent](#) [tribute](#)), 67
[tribute](#) ([src.runtime.models.XEvent](#) [tribute](#)), 67
[tribute](#) ([src.runtime.models.XEvent](#) [tribute](#)), 67
[tribute](#) ([src.predictive_model.classification.models.XGBoost](#) [tribute](#)), 54
[tribute](#) ([src.predictive_model.regression.models.XGBoost](#) [tribute](#)), 60
[tribute](#) ([src.predictive_model.classification.models.ClassificationMethod](#) [tribute](#)), 46
[tribute](#) ([src.predictive_model.classification.models.ClassificationMethod](#) [tribute](#)), 46

U
[unique_events\(\)](#) ([src.utils.event_attributes](#)), 75
[unique_events2\(\)](#) ([src.utils.event_attributes](#)), 75
[UP_TO](#) ([src.encoding.models.TaskGenerationTypes](#) [tribute](#)), 22
[UPDATE](#) ([src.jobs.models.JobTypes](#) [tribute](#)), 35
[update\(\)](#) ([src.jobs.job_creator](#)), 31
[upload_multiple\(\)](#) ([src.logs.views](#)), 41
V
[validation_fraction](#) ([src.predictive_model.classification.models.Perceptron](#) [tribute](#)), 52
[validation_fraction](#) ([src.predictive_model.classification.models.SGDClassifier](#) [tribute](#)), 54
[value_encoding](#) ([src.encoding.models.Encoding](#) [tribute](#)), 22
[ValueEncodings](#) ([src.encoding.models](#)), 22
W
[weights](#) ([src.predictive_model.classification.models.Knn](#) [tribute](#)), 49
X
[XEvent](#) ([src.runtime.models](#)), 67
[XEvent.DoesNotExist](#), 67
[XEvent.MultipleObjectsReturned](#), 67
[XGBoost](#) ([src.predictive_model.classification.models](#)), 54
[XGBoost](#) ([src.predictive_model.regression.models](#)), 60
[xgboost](#) ([src.predictive_model.classification.models.ClassificationMethod](#) [tribute](#)), 46
[XGBOOST](#) ([src.predictive_model.classification.models.ClassificationMethod](#) [tribute](#)), 46

`xgboost` (*src.predictive_model.regression.models.Regression*
attribute), 60
`XGBOOST` (*src.predictive_model.regression.models.RegressionMethods*
attribute), 60
`XGBoost.DoesNotExist`, 54, 60
`XGBoost.MultipleObjectsReturned`, 54, 60
`xid` (*src.runtime.models.XEvent attribute*), 68
`XLog` (*class in src.runtime.models*), 68
`xlog` (*src.runtime.models.XLog attribute*), 68
`xlog` (*src.runtime.models.XTrace attribute*), 70
`XLog.DoesNotExist`, 68
`XLog.MultipleObjectsReturned`, 68
`xlog_id` (*src.runtime.models.XTrace attribute*), 70
`XTrace` (*class in src.runtime.models*), 69
`xtrace` (*src.runtime.models.XTrace attribute*), 71
`XTrace.DoesNotExist`, 69
`XTrace.MultipleObjectsReturned`, 69