Nirdizati Research Documentation

Stefano Branchi, Chiara Di Francescomarino, Tõnis Kasekamp, Sa

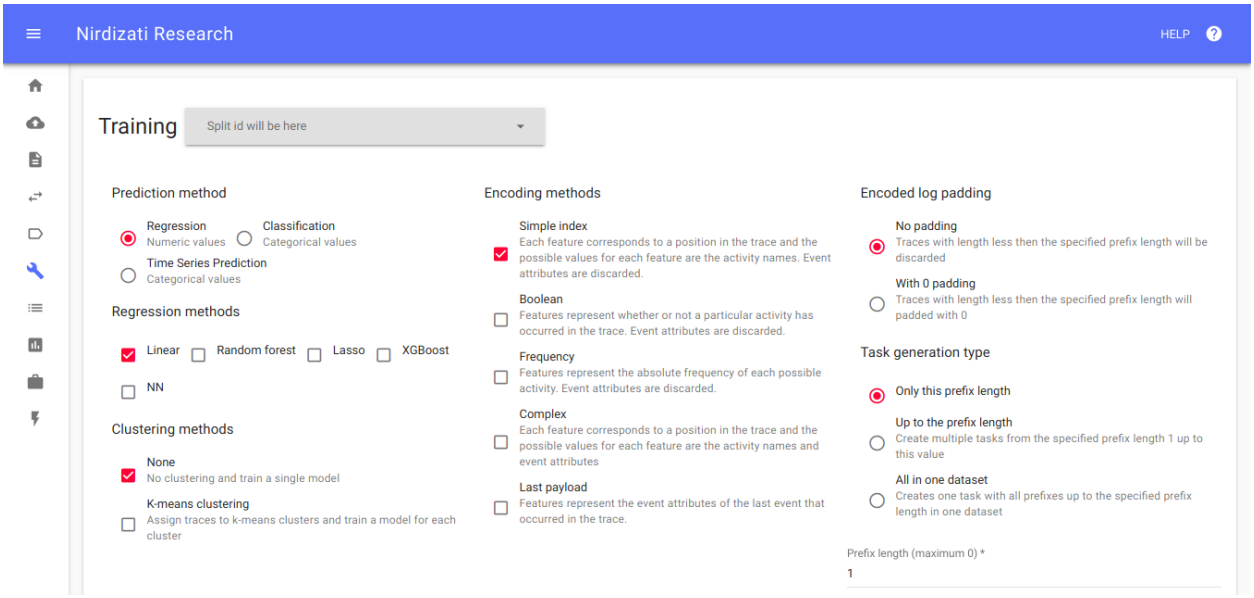
Jun 17, 2019

READMEs

A Web Application to Support Research in Predictive Monitoring Tasks

Nirdizati Research is a tool that aims at supporting expert users in finding the best predictive models suitable for specific datasets and prediction problems. The predictive models can then be used to provide predictions at runtime on incomplete traces or streams of events.

The tool offers a wide selection of features ranging from event log preprocessing options to hyperparmeter optimization as well as runtime feedback to the user.



CHAPTER 1

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 Tõnis Kasekamp

- @mrsonuk Santosh Kumar
- @fmmaggi Fabrizio Maggi
- @WilliamsRizzi Williams Rizzi
- @HitLuca Luca Simonetto

CHAPTER 2

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 yuor 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 Tõnis Kasekamp
- @mrsonuk Santosh Kumar
- @fmmaggi Fabrizio Maggi
- @WilliamsRizzi Williams Rizzi
- @HitLuca Luca Simonetto

CHAPTER 3

src

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 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*, *column_len*, *attribute_classifier=None*, *executed_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*, *evaluation_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*, *f1_score*, *accuracy*, *precision*, *recall*)

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*, *f1_score*, *accuracy*, *precision*, *recall*, *classificationmetrics_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*, *hyperparameteroptimization_ptr*, *max_evaluations*, *performance_metric*, *algorithm_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: performance_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*, *optimization_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*, *attribute_classifier=None*, *executed_events=None*, *resources_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*, *results*)

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_hoef**

src.predictive_model.classification.methods_default_config.**classification_incremental_naive**

src.predictive_model.classification.methods_default_config.**classification_incremental_perce**

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*, *predictive_model*, *prediction_method*, *predictive_model_ptr*, *classification_ptr*, *grace_period*, *split_criterion*, *split_confidence*, *tie_threshold*, *remove_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

adaptivehoeffdingtree
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.

hoeffdingtree
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*, *predictive_model*, *prediction_method*, *predictive_model_ptr*, *classification_ptr*, *grace_period*, *split_criterion*, *split_confidence*, *tie_threshold*, *remove_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*, *predictive_model*, *prediction_method*, *predictivemodel_ptr*, *classification_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*, *predictive_model*, *prediction_method*, *predictive_model_ptr*, *classification_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*, *validation_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*, *predictivemodel_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*, *predictive_model*, *prediction_method*, *predictive_model_ptr*, *classification_ptr*, *loss*, *penalty*, *alpha*, *l1_ratio*, *fit_intercept*, *tol*, *epsilon*, *learning_rate*, *eta0*, *power_t*, *n_iter_no_change*, *validation_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>*)

l1_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*, *predictive_model*, *prediction_method*, *predictive_model_ptr*, *classification_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*, *predictive_model*, *prediction_method*, *predictivemodel_ptr*, *regression_ptr*, *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

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*, *activation*, *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*, *predictive_model*, *prediction_method*, *predictivemodel_ptr*, *regression_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*, *predictive_model*, *prediction_method*, *predictivemodel_ptr*, *regression_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.**TimeSeriesPredic**
Bases: object

src.predictive_model.time_series_prediction.apps module

class src.predictive_model.time_series_prediction.apps.**TimeSeriesPredictionConfig**(*app_name*, *app_module*
Bases: *src.predictive_model.apps.PredictiveModelConfig*

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_**

src.predictive_model.time_series_prediction.models module

class src.predictive_model.time_series_prediction.models.**RecurrentNeuralNetwork**(*id*, *model_path*, *predictive_model*, *prediction_method*, *predictive_model_ptr*, *timeseriesprediction_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*
c
t
j

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_s**

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_l

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_nlevenshtein**(*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'*, *performance_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'*, *attribute_name=None*, *threshold_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'*, *prediction_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

Python Module Index

S

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X