DQ0 SDK Documentation

Release 1.0.10.1

Gradient Zero

Apr 16, 2021

1 Subpackages **1**
1.1 dq0.sdk.cli package 1
1.2 dq0.sdk.data package 22
1.3 dq0.sdk.errors package 53
1.4 dq0.sdk.estimators package 54
1.5 dq0.sdk.examples package 61
1.6 dq0.sdk.models package 72
1.7 dq0.sdk.pipeline package 85
1.8 dq0.sdk.projects package 88
1.9 dq0.sdk.utils package 89

Python Module Index **91**

Index **93**

Chapter 1

Subpackages

1.1 dq0.sdk.cli package

DQ0 SDK CLI Package

This package comprises the communication hub to the local DQ0 CLI API.

class dq0.sdk.cli.**Experiment** (*project=None*, *name=None*)

Bases: object

An experiment

Provides methods to train models and preprocess datasets.

Note:

There can be only one epxeriment for model and data each per project version. If you want to run mulitple experiments for one model in parallel use different project versions.

Example:

```
>>> # Create an experiment. Then call train and preprocess
>>> experiment = Experiment(project=project, name='experiment_1')
>>> run = experiment.train()
>>> run = experiment.preprocess()
```

Args:

project (dq0.sdk.cli.Project): The project
this experiment belongs to

name (str): The name of the new experiment

Attributes:

project (dq0.sdk.cli.Project): The project
this experiment belongs to

name (str): The of the experiment

get_last_data_run ( )

Returns the latest DataRunner.

Can be used to cancel zombie jobs for example.

Returns:

A new instance of the DataRunner class for the project.

get_last_model_run ( )

Returns the latest ModelRunner.

Can be used to cancel zombie jobs for example.

Returns:
A new instance of the ModelRunner class for the project.

preprocess ( )
Starts a preprocessing run
It calls the CLI command *data preprocess* and returns a Runner instance to watch to job.

Returns:
A new instance of the DataRunner class for the preprocess run.

run (*entry_point='train_use_dq0_makedp'*, *args=''*)
Starts a training run
It calls the CLI command *model train* and returns a Runner instance to watch to job

Returns:
A new instance of the ModelRunner class for the train run.

class dq0.sdk.cli.**Model** (*project=None*, *run_id=None*)
Bases: object
A model predict wrapper
Provides methods to call model predict

Example:

```
>>> # get the latest model
>>> model = project.get_latest_model()
>>>
>>> # check DQ0 privacy clearing
>>> if model.predict_allowed:
>>>
>>>     # call predict
>>>     run = model.predict(np.array([1, 2, 3]))
>>>
>>>     # wait for completion
>>>     run.wait_for_completion(verbose=True)
>>>
>>>     # get training results
>>>     print(run.get_results())
```

Args:

project (dq0.sdk.cli.Project): The project
this model belongs to

Attributes:

project (dq0.sdk.cli.Project): The project
this model belongs to

predict_allowed (bool): True if the model was checked by DQ0
and flagged as safe. Note that this attribute is here for convenience. The actual allowance check is done by dq0-main.

predict (*test_data*)
Starts a prediction run
It calls the CLI command *model predict* and returns a Runner instance to watch to job

Args:
test_data (numpy.array) data to perform prediction for

Returns:
New instance of the ModelRunner class representing the prediction run.

register (*run_id=None*, *model_path=None*)
Registers the model (to be later used for predict)

It calls the CLI command *model register*
If the job_uuid and model_path arguments are omitted, the SDK assumes a DQ0 secured training run and uses the hard coded default values.

Args:
run_id: the ID of the run containing the model artifact model_path: the relative path to the model artifact

class dq0.sdk.cli.**Project** (*name=None*, *create=True*)
Bases: object
A user project
Provides methods to create and manage a user project comprising of user_model and user_-source code.

Example:

```
>>> # Create a new project
>>> project = Project(name='some name')
```

```
>>> # Load a project (cd into project dir first)
>>> project = Project.load()
```

Args:
name (str): The name of the new project create (bool): True to create a new project via DQ0 CLI.

Default is True.

Attributes:
name (str): The name of the project project_uuid (str): The universally unique identifier of

the project

data_source_uuid (str): The universally unique identifier of
the project's currently attached data source
version (str): A version number of the project.

attach_data_source (*data=None*, *data_uuid=None*, *data_name=None*)
Attaches a new data source to the project.

Args:
data (dq0.sdk.cli.Data): Data instance of the data source to attach data_uuid (str): optional; The UUID of the new source to attach data_name (str): optional; The name of the new source to attach

commit ( )

detach_data_source (*data=None*, *data_uuid=None*, *data_name=None*)
Detaches a new data source to the project.

Args:
data (dq0.sdk.cli.Data): Data instance of the data source to attach data_uuid (str): optional; The UUID of the new source to attach data_name (str): optional; The name of the new source to attach

get_attached_data_sources ( )

get_available_data_sources ( )
Returns a list of available data sources.
The returned Data instances can be used for the attach_data_source method.

Returns:
A list of available data sources.

get_data_info (*data=None*, *data_uuid=None*)

Returns info of a given data source.
The returned dict contains information about the data source depending on the source's permissions set by the data owner.

Args:

data (dq0.sdk.cli.Data): Data instance of the requested data source data_uuid (str): optional; The UUID of the requested data source

Returns:

The data source information in JSON format

get_sample_data (*data=None*, *data_uuid=None*)

Returns sample data for a given data source.
Sample data is provided manually and is not available for every data source.

Args:

data (dq0.sdk.cli.Data): Data instance of the requested data source data_uuid (str): optional; The UUID of the requested data source

Returns:

The data source sample data

info ( )

Info returns information about the project.
It calls the CLI command *project info* and returns the results as JSON.

Returns:

Project info in JSON format

static load ( )

Load loads an existing project.
Load is a static function to create a new model instance from an existing local project.
It reads the .meta file of the current directory to collect all neccessary project information.

Returns:

New instance of the Project class for the loaded project

Raises:

FileNotFoundError: if the .meta project file was not found in
the current directory.

set_connection (*host='localhost'*, *port=9000*)

Updates the connection string for the API communication.
Passes the updated info to the API handler.

Args:

host (str): The host of the DQ0 CLI API Server port (int): The port of the DQ0 CLI API Server

set_model_code (*setup_data=None*, *setup_model=None*, *preprocess=None*, *parent_class_name=None*)

Sets the user defined setup_model and setup_data functions.
Saves the function code to user_model.py

Note:

This function will only work inside iphyton notebooks, otherwise the sources of the function arguments are not available.

Args:

setup_data (func, optional): user defined setup_data function setup_model (func, optional): user defined setup_model function preprocess (func, optional): user defined

preprocess function parent_class_name (str, optional): name of the parent class for UserModel

update_commit_uuid (*message*)

Updates the latest commit uuid from the given response message.

Args:

message (str): The response message after deploy.

class dq0.sdk.cli.**Data** (*source*, *project=None*)

Bases: object

A data source wrapper

Provides an interface for inspecting/interacting with data sources as well as query methods

Args:

project (dq0.sdk.cli.Project): The project
this data source belongs to

Attributes:

project (dq0.sdk.cli.Project): The project
this data source belongs to

uuid (:obj:str): UUID of data source name (:obj:str): Name of data source type (:obj:str): Type of data source

all (**args*)

All reset filter.

as_dict ( )

Returns data source representation as dictionary

distribution (*cols=None*)

Gets the differential private mean value of the given columns

Args:

cols: list of columns in the dataset to include. None for all available columns

static **get_available_data_sources** ( )

Returns a list of available data sources.

The returned Data instances can be used for the attach_data_source method.

Returns:

A list of available data sources.

static **get_data_info** (*data=None*, *data_uuid=None*)

Returns info of a given data source.

The returned dict contains information about the data source depending on the source's permissions set by the data owner.

Args:

data (dq0.sdk.cli.Data): Data instance of the requested data source data_uuid (str): optional; The UUID of the requested data source

Returns:

The data source information in JSON format

mean (*cols=None*)

Gets the differential private mean value of the given columns

Args:

cols: list of columns in the dataset to include. None for all available columns

query (*query*, *epsilon=1.0*, *tau=0.0*, *private_column=''*, *permissions=None*, *params=None*, *project=None*)

Run a query on this Data instance.

Args:

query: string containing SQL permissions: optional; e.g. 'households<75' params: optional; e.g. 'p1=123' epsilon: float; Epsilon value for differential private query. Default: 1.0 tau: float; Tau threshold value for private query. Default: 0.0 private_column: string; Private column for this query. Leave empty or omit for default value from metadata. project:*dq0.sdk.cli.project.Project* instance.

Returns:

`dq0.sdk.cli.runner.QueryRunner` instance

refresh ( )

Reloads data source information from database. Useful when instantiating from incomplete dictionaries or when the data has changed

where (**args*)

Where filter. TBD.

class `dq0.sdk.cli.`**Query** (*project*)

Bases: `object`

A query source wrapper

Provides methods to run query jobs. A query always needs to be run in a project context. Alternative to using the query method directly from a Data instance. Allows for querying multiple data sources.

Args:

project (`dq0.sdk.cli.Project`): The project
this query belongs to

Attributes:

project (`dq0.sdk.cli.Project`): The project
this query belongs to

execute (*query*, *epsilon=1.0*, *tau=0.0*, *private_column=''*, *permissions=None*, *params=None*)

Run a query on the data sources defined by this Query instance.

Args:

query: string containing SQL epsilon: float; Epsilon value for differential private query. Default: 1.0 tau: float; Tau threshold value for private query. Default: 0.0 private_column: string; Private column for this query. Leave empty or omit for default value from metadata. permissions: optional; e.g. 'households<75' params: optional; e.g. 'p1=123'

Returns:

`dq0.sdk.cli.runner.QueryRunner` instance

for_data (*data*)

Specifiy which datasets are used in query. Args:

data (`list`) list of `dq0.sdk.cli.Data` instances included in query. Alternatively, pass a single `dq0.sdk.cli.Data` instance.

Returns:

`dq0.sdk.cli.Query` instance with set datasets

get_dataset_names ( )

Returns used dataset names as a single comma-separated string

1.1.1 Subpackages

dq0.sdk.cli.api package

DQ0 SDK CLI API Package

This package comprises the communication classes to interact with the DQ0 CLI API.

class dq0.sdk.cli.api.**Client** (*host='localhost'*, *port=9000*)

Bases: object

A simple HTTP client

This class is used to communicate with the DQ0 CLI API. It shall not be used directly, but rather is called by Experiment, Model and Runner to commit actions and retrieve information via the DQ0 CLI API.

Example:

```
>>> # Create an instance
>>> client = Client()
>>>
>>> # make a request
>>> route = 'project'
>>> json_response = client.request(route)
```

Args:

host (str): The host of the DQ0 CLI API Server
(default 'localhost')

port (int): The port of the DQ0 CLI API Server (default 9000)

Attributes:

api (str): The complete API URL, host + port

get (*route*, *uuid=None*, *data=None*)

Make an HTTP GET request.

Calles the DQ0 CLI API with a GET request on the given route.

Returns the response as JSON. Throws an error on failure.

Args:

route (str): The API route to request. uuid (str): If set this value will replace route's ':uuid' placeholder data (optional, dict): GET data to pass.

Returns:

The HTTP response in JSON format

post (*route*, *uuid=None*, *data=None*)

Make an HTTP POST request.

Calles the DQ0 CLI API with a POST request on the given route.

Returns the response as JSON. Throws an error on failure.

Args:

route (str): The API route to request. uuid (str): If set this value will replace route's ':uuid' placeholder data (dict, optional): POST data to pass.

Returns:

The HTTP response in JSON format

set_connection (*host='localhost'*, *port=9000*, *verbose=True*)

Updates the connection string for the API communication.

Args:

host (str): The host of the DQ0 CLI API Server port (int): The port of the DQ0 CLI API Server

Submodules

dq0.sdk.cli.api.client module

Client communicates with the DQ0 CLI API via requests http calls

Client uses the requests library to perform HTTP requests.

class `dq0.sdk.cli.api.client.`**`Client`** (*host='localhost'*, *port=9000*)

Bases: `object`

A simple HTTP client

This class is used to communicate with the DQ0 CLI API. It shall not be used directly, but rather is called by Experiment, Model and Runner to commit actions and retrieve information via the DQ0 CLI API.

Example:

```
>>> # Create an instance
>>> client = Client()
>>>
>>> # make a request
>>> route = 'project'
>>> json_response = client.request(route)
```

Args:

host (`str`): The host of the DQ0 CLI API Server
(default 'localhost')
port (int): The port of the DQ0 CLI API Server (default 9000)

Attributes:

api (`str`): The complete API URL, host + port

`get` (*route*, *uuid=None*, *data=None*)

Make an HTTP GET request.

Calles the DQ0 CLI API with a GET request on the given route.

Returns the response as JSON. Throws an error on failure.

Args:

route (`str`): The API route to request. uuid (`str`): If set this value will replace route's ':uuid' placeholder data (optional, dict): GET data to pass.

Returns:

The HTTP response in JSON format

`post` (*route*, *uuid=None*, *data=None*)

Make an HTTP POST request.

Calles the DQ0 CLI API with a POST request on the given route.

Returns the response as JSON. Throws an error on failure.

Args:

route (`str`): The API route to request. uuid (`str`): If set this value will replace route's ':uuid' placeholder data (`dict`, optional): POST data to pass.

Returns:

The HTTP response in JSON format

`set_connection` (*host='localhost'*, *port=9000*, *verbose=True*)

Updates the connection string for the API communication.

Args:

host (`str`): The host of the DQ0 CLI API Server port (int): The port of the DQ0 CLI API Server

dq0.sdk.cli.api.routes module

Routes defines all available DQ0 API routes.

Use the a route to select an action and do the request like this:
client.request(routes.project.info, {'some': 'data'})

dq0.sdk.cli.runner package

Runner manages a running experiment.

There is an abstract base class for runner and two implementing child classes:

- ModelRunner
- DataRunner

class dq0.sdk.cli.runner.**DataRunner** (*project*)

Bases: dq0.sdk.cli.runner.runner.Runner

A running data experiment
Provides methods to get job status, wait for completion or cancel job.

Example:

```
>>> # call preprocess
>>> run = experiment.preprocess()
>>>
>>> # get status
>>> print(run.get_state())
>>>
>>> # wait for completion
>>> run.wait_for_completion(verbose=True)
>>>
>>> # or cancel
>>> run.cancel()
```

Args:

project (dq0.sdk.cli.Project): The project
this runner belongs to

cancel (*force=False*)

Cancels the data run.

Args:

force (bool, optional): Set to true to force the job to be
interrupted. Default is false where the job gracefully gets signalled to halt.

get_state ( )

Gets the current state of the running data experiment.

Returns:
The state in JSON format

class dq0.sdk.cli.runner.**ModelRunner** (*project*, *job_uuid*)

Bases: dq0.sdk.cli.runner.runner.Runner

A running model experiment
Provides methods to get job status, wait for completion or cancel job.

Example:

```
>>> # call train
>>> run = experiment.train()
>>>
```

```
>>> # get status
>>> print(run.get_state())
>>>
>>> # wait for completion
>>> run.wait_for_completion(verbose=True)
>>>
>>> # or cancel
>>> run.cancel()
```

Args:

project (dq0.sdk.cli.Project): The project
this runner belongs to

cancel (*force=False*)
Cancels the experiment run.

Args:

force (bool, optional): Set to true to force the job to be
interrupted. Default is false where the job gracefully gets signalled to halt.

get_model ( )
Returns a model instance for the given run.

Returns:
The model instance

get_state ( )
Gets the current state of the running model experiment.

Returns:
The state in JSON format

class dq0.sdk.cli.runner.**QueryRunner** (*project*, *query_uuid*)
Bases: dq0.sdk.cli.runner.runner.Runner
A running query
Provides methods to get job status, wait for completion or cancel job.

Args:

project (dq0.sdk.cli.Project): The project
this runner belongs to

cancel (*force=False*)
Cancels the running query.

Args:

force (bool, optional): Set to true to force the job to be
interrupted. Default is false where the job gracefully gets signalled to halt.

get_state ( )
Gets the current state of the running query.

Returns:
The state in JSON format

Submodules

dq0.sdk.cli.runner.data_runner module

Data Runner manages a running data experiment.

When starting an experiment with experiment.preprocess() a new DataRunner instance is returned.

Runner can tell the job's current state, it can wait for the job to complete, or it can (forcefully) cancel the job.

DataRunner wraps the following CLI commands:

- dq0 data state
- dq0 data cancel

class dq0.sdk.cli.runner.data_runner.**DataRunner** (*project*)

Bases: dq0.sdk.cli.runner.runner.Runner

A running data experiment

Provides methods to get job status, wait for completion or cancel job.

Example:

```
>>> # call preprocess
>>> run = experiment.preprocess()
>>>
>>> # get status
>>> print(run.get_state())
>>>
>>> # wait for completion
>>> run.wait_for_completion(verbose=True)
>>>
>>> # or cancel
>>> run.cancel()
```

Args:

project (dq0.sdk.cli.Project): The project
this runner belongs to

cancel (*force=False*)

Cancels the data run.

Args:

force (bool, optional): Set to true to force the job to be
interrupted. Default is false where the job gracefully gets signalled to halt.

get_state ( )

Gets the current state of the running data experiment.

Returns:
The state in JSON format

dq0.sdk.cli.runner.model_runner module

Model Runner manages a running model experiment.

When starting an experiment with experiment.train() or model.predict() a new ModelRunner instance is returned.

Runner can tell the job's current state, it can wait for the job to complete, or it can (forcefully) cancel the job.

ModelRunner wraps the following CLI commands:

- dq0 model state
- dq0 model cancel

class dq0.sdk.cli.runner.model_runner.**ModelRunner** (*project*, *job_uuid*)

Bases: dq0.sdk.cli.runner.runner.Runner

A running model experiment

Provides methods to get job status, wait for completion or cancel job.

Example:

```
>>> # call train
>>> run = experiment.train()
>>>
>>> # get status
>>> print(run.get_state())
>>>
>>> # wait for completion
>>> run.wait_for_completion(verbose=True)
>>>
>>> # or cancel
>>> run.cancel()
```

Args:

project (dq0.sdk.cli.Project): The project
this runner belongs to

cancel (*force=False*)
Cancels the experiment run.

Args:

force (bool, optional): Set to true to force the job to be
interrupted. Default is false where the job gracefully gets signalled to halt.

get_model ( )
Returns a model instance for the given run.

Returns:
The model instance

get_state ( )
Gets the current state of the running model experiment.

Returns:
The state in JSON format

dq0.sdk.cli.runner.query_runner module

Query Runner manages a running queries.

When starting an query with Data.query() or Query.execute() a new QueryRunner instance is returned.

Runner can tell the job's current state, it can wait for the job to complete, or it can (forcefully) cancel the job.

QueryRunner wraps the following CLI commands:

- dq0 query state
- dq0 query cancel

class dq0.sdk.cli.runner.query_runner.**QueryRunner** (*project*, *query_uuid*)
Bases: dq0.sdk.cli.runner.runner.Runner
A running query
Provides methods to get job status, wait for completion or cancel job.

Args:

project (dq0.sdk.cli.Project): The project
this runner belongs to

cancel (*force=False*)

Cancels the running query.

Args:

force (bool, optional): Set to true to force the job to be
interrupted. Default is false where the job gracefully gets signalled to halt.

get_state ( )

Gets the current state of the running query.

Returns:

The state in JSON format

dq0.sdk.cli.runner.runner module

Runner manages a running experiment.

When starting an experiment with experiment.train(), experiment.preprocess() or model.predict() a new Runner instance is returned.

Runner can tell the job's current state, it can wait for the job to complete, or it can (forcefully) cancel the job.

Runner wraps the following CLI commands:

- dq0 model state
- dq0 model cancel
- dq0 data state
- dq0 data cancel

class dq0.sdk.cli.runner.runner.**Runner** (*project*)

Bases: abc.ABC

A running experiment

Provides methods to get job status, wait for completion or cancel job.

Example:

```
>>> # call train
>>> run = experiment.train()
>>>
>>> # get status
>>> print(run.get_state())
>>>
>>> # wait for completion
>>> run.wait_for_completion(verbose=True)
>>>
>>> # or cancel
>>> run.cancel()
```

Args:

project (dq0.sdk.cli.Project): The project
this runner belongs to

Attributes:

project (dq0.sdk.cli.Project): The project
this runner belongs to

state (dq0.sdk.cli.runner.State) The runner's state

abstract cancel (*force=False*)

Cancels the experiment run

get_error ( )

get_results ( )
Gets the results of the running model or data experiment.

Returns:
The final state in JSON format or an empty dict if the run has not finished yet.

abstract get_state ( )
Gets the current state of the running model or data experiment.

Returns:
The state in JSON format

wait_for_completion (*verbose=False*)
Loops until the state reflects the end of the run.
This function is blocking.

Args:

verbose (bool, optional): Set to true to see periodic state outputs.
Default is false

dq0.sdk.cli.runner.state module

State reprersents the state of a running job.

A runner can tell the job's current state, this class provides the info.

The state will be updated by the runner. It provides a finished status and current state log information.

State ID values provided by dq0-main:

- 0 - StateCreated
- 1 - StatePrepared
- 2 - StateRunning
- 3 - StateFailed
- 4 - StateStopped

class dq0.sdk.cli.runner.state.**State**
Bases: object
A state for a running job (model or data)
Provides methods to get the job's state and log message.

Attributes:

finished (bool): The state's finished flag.
True if the run has finished.
message (str): The last log message of the run. results (dict): The run's state once finished.
progress (float): Progress.

set_results (*results*)
Update parsed run results

update (*response*)
Updates the state representation

dq0.sdk.cli.utils package

Utils package contains helper functions for the SDK API

dq0.sdk.cli.utils.**is_valid_uuid** (*uuid_to_test*, *version=4*)
Check if uuid_to_test is a valid UUID.

Submodules

dq0.sdk.cli.utils.code module

Utils package contains helper functions for the SDK API

dq0.sdk.cli.utils.code.**add_function** (*lines*, *code*)
Add the given code to the end of lines.

dq0.sdk.cli.utils.code.**check_signature** (*code*, *name*)
check method signatures. Used by set_code functions.

dq0.sdk.cli.utils.code.**replace_function** (*lines*, *code*)
Replace functions code. Used my set_code functions.

dq0.sdk.cli.utils.code.**replace_model_parent_class** (*lines*, *parent_class_name*)
Replace parent class. Used my set_code functions.

dq0.sdk.cli.utils.uuid module

Utils package contains helper functions for the SDK API

dq0.sdk.cli.utils.uuid.**is_valid_uuid** (*uuid_to_test*, *version=4*)
Check if uuid_to_test is a valid UUID.

1.1.2 Submodules

dq0.sdk.cli.data module

Data allows for the execution of db stats jobs

A data object will be created at runtime from a project instance.

class dq0.sdk.cli.data.**Data** (*source*, *project=None*)
Bases: object
A data source wrapper
Provides an interface for inspecting/interacting with data sources as well as query methods

Args:

project (dq0.sdk.cli.Project): The project
this data source belongs to

Attributes:

project (dq0.sdk.cli.Project): The project
this data source belongs to
uuid (:obj:str): UUID of data source name (:obj:str): Name of data source type (:obj:str): Type of data source

all (**args*)
All reset filter.

as_dict ( )
Returns data source representation as dictionary

distribution (*cols=None*)

Gets the differential private mean value of the given columns

Args:

cols: list of columns in the dataset to include. None for all available columns

static **get_available_data_sources** ( )

Returns a list of available data sources.

The returned Data instances can be used for the attach_data_source method.

Returns:

A list of available data sources.

static **get_data_info** (*data=None*, *data_uuid=None*)

Returns info of a given data source.

The returned dict contains information about the data source depending on the source's permissions set by the data owner.

Args:

data (`dq0.sdk.cli.Data`): Data instance of the requested data source data_uuid (`str`): optional; The UUID of the requested data source

Returns:

The data source information in JSON format

mean (*cols=None*)

Gets the differential private mean value of the given columns

Args:

cols: list of columns in the dataset to include. None for all available columns

query (*query*, *epsilon=1.0*, *tau=0.0*, *private_column=''*, *permissions=None*, *params=None*, *project=None*)

Run a query on this Data instance.

Args:

query: string containing SQL permissions: optional; e.g. 'households<75' params: optional; e.g. 'p1=123' epsilon: float; Epsilon value for differential private query. Default: 1.0 tau: float; Tau threshold value for private query. Default: 0.0 private_column: string; Private column for this query. Leave empty or omit for default value from metadata. project:*dq0.sdk.cli.project.Project* instance.

Returns:

`dq0.sdk.cli.runner.QueryRunner` instance

refresh ( )

Reloads data source information from database. Useful when instantiating from incomplete dictionaries or when the data has changed

where (**args*)

Where filter. TBD.

dq0.sdk.cli.experiment module

Experiment allows for the execution of training and preprocessing jobs

An experiment will be created at runtime. It has a project and a name.

Calling train or preprocess through the experiment will return a Runner instance that can be used to further control the job

Experiment wraps the following CLI commands:

- dq0 model train

- dq0 data preprocess

class dq0.sdk.cli.experiment.**Experiment** (*project=None, name=None*)

Bases: object

An experiment

Provides methods to train models and preprocess datasets.

Note:

There can be only one epxeriment for model and data each per project version. If you want to run mulitple experiments for one model in parallel use different project versions.

Example:

```
>>> # Create an experiment. Then call train and preprocess
>>> experiment = Experiment(project=project, name='experiment_1')
>>> run = experiment.train()
>>> run = experiment.preprocess()
```

Args:

project (dq0.sdk.cli.Project): The project
this experiment belongs to

name (str): The name of the new experiment

Attributes:

project (dq0.sdk.cli.Project): The project
this experiment belongs to

name (str): The of the experiment

get_last_data_run ( )

Returns the latest DataRunner.

Can be used to cancel zombie jobs for example.

Returns:

A new instance of the DataRunner class for the project.

get_last_model_run ( )

Returns the latest ModelRunner.

Can be used to cancel zombie jobs for example.

Returns:

A new instance of the ModelRunner class for the project.

preprocess ( )

Starts a preprocessing run

It calls the CLI command *data preprocess* and returns a Runner instance to watch to job.

Returns:

A new instance of the DataRunner class for the preprocess run.

run (*entry_point='train_use_dq0_makedp', args=''*)

Starts a training run

It calls the CLI command *model train* and returns a Runner instance to watch to job

Returns:

A new instance of the ModelRunner class for the train run.

dq0.sdk.cli.model module

Model allows for the execution of prediction jobs

A model will be created at runtime. It belongs to a project.

Calling predict through Model will return a ModelRunner instance that can be used to further control the job

Model wraps the following CLI commands:

- dq0 model predict

class dq0.sdk.cli.model.**Model** (*project=None*, *run_id=None*)

Bases: object

A model predict wrapper

Provides methods to call model predict

Example:

```
>>> # get the latest model
>>> model = project.get_latest_model()
>>>
>>> # check DQ0 privacy clearing
>>> if model.predict_allowed:
>>>
>>>     # call predict
>>>     run = model.predict(np.array([1, 2, 3]))
>>>
>>>     # wait for completion
>>>     run.wait_for_completion(verbose=True)
>>>
>>>     # get training results
>>>     print(run.get_results())
```

Args:

project (dq0.sdk.cli.Project): The project
this model belongs to

Attributes:

project (dq0.sdk.cli.Project): The project
this model belongs to

predict_allowed (bool): True if the model was checked by DQ0
and flagged as safe. Note that this attribute is here for convenience. The actual allowance check is done by dq0-main.

predict (*test_data*)

Starts a prediction run

It calls the CLI command *model predict* and returns a Runner instance to watch to job

Args:

test_data (numpy.array) data to perform prediction for

Returns:

New instance of the ModelRunner class representing the prediction run.

register (*run_id=None*, *model_path=None*)

Registers the model (to be later used for predict)

It calls the CLI command *model register*

If the job_uuid and model_path arguments are omitted, the SDK assumes a DQ0 secured training run and uses the hard coded default values.

Args:

run_id: the ID of the run containing the model artifact model_path: the relative path to the model artifact

dq0.sdk.cli.project module

Project represents a user project

This class provides methods to create and manage a user project comprising of user_model and user_-source code.

Project reads and writes the .meta file in the current project directory.

Project wraps the following CLI commands:

- dq0 project info
- dq0 project create [NAME]
- dq0 project deploy
- dq0 data list
- dq0 data attach

class `dq0.sdk.cli.project.`**`Project`** (*name=None*, *create=True*)

Bases: `object`

A user project

Provides methods to create and manage a user project comprising of user_model and user_-source code.

Example:

```
>>> # Create a new project
>>> project = Project(name='some name')
```

```
>>> # Load a project (cd into project dir first)
>>> project = Project.load()
```

Args:

name (`str`): The name of the new project create (bool): True to create a new project via DQ0 CLI.

Default is True.

Attributes:

name (`str`): The name of the project project_uuid (`str`): The universally unique identifier of the project

data_source_uuid (`str`): The universally unique identifier of the project's currently attached data source

version (`str`): A version number of the project.

`attach_data_source` (*data=None*, *data_uuid=None*, *data_name=None*)

Attaches a new data source to the project.

Args:

data (`dq0.sdk.cli.Data`): Data instance of the data source to attach data_uuid (`str`): optional; The UUID of the new source to attach data_name (`str`): optional; The name of the new source to attach

`commit` ( )

`detach_data_source` (*data=None*, *data_uuid=None*, *data_name=None*)

Detaches a new data source to the project.

Args:

data (`dq0.sdk.cli.Data`): Data instance of the data source to attach data_uuid (`str`):

optional; The UUID of the new source to attach data_name (`str`): optional; The name of the new source to attach

get_attached_data_sources ( )

get_available_data_sources ( )

Returns a list of available data sources.

The returned Data instances can be used for the attach_data_source method.

Returns:

A list of available data sources.

get_data_info (*data=None*, *data_uuid=None*)

Returns info of a given data source.

The returned dict contains information about the data source depending on the source's permissions set by the data owner.

Args:

data (`dq0.sdk.cli.Data`): Data instance of the requested data source data_uuid (`str`): optional; The UUID of the requested data source

Returns:

The data source information in JSON format

get_sample_data (*data=None*, *data_uuid=None*)

Returns sample data for a given data source.

Sample data is provided manually and is not available for every data source.

Args:

data (`dq0.sdk.cli.Data`): Data instance of the requested data source data_uuid (`str`): optional; The UUID of the requested data source

Returns:

The data source sample data

info ( )

Info returns information about the project.

It calls the CLI command *project info* and returns the results as JSON.

Returns:

Project info in JSON format

static load ( )

Load loads an existing project.

Load is a static function to create a new model instance from an existing local project.

It reads the .meta file of the current directory to collect all neccessary project information.

Returns:

New instance of the Project class for the loaded project

Raises:

FileNotFoundError: if the .meta project file was not found in
the current directory.

set_connection (*host='localhost'*, *port=9000*)

Updates the connection string for the API communication.

Passes the updated info to the API handler.

Args:

host (`str`): The host of the DQ0 CLI API Server port (int): The port of the DQ0 CLI API Server

set_model_code (*setup_data=None*, *setup_model=None*, *preprocess=None*, *parent_class_name=None*)

Sets the user defined setup_model and setup_data functions.
Saves the function code to user_model.py

Note:

This function will only work inside iphyton notebooks, otherwise the sources of the function arguments are not available.

Args:

setup_data (func, optional): user defined setup_data function setup_model (func, optional): user defined setup_model function preprocess (func, optional): user defined preprocess function parent_class_name (`str`, optional): name of the parent class for UserModel

update_commit_uuid (*message*)

Updates the latest commit uuid from the given response message.

Args:

message (`str`): The response message after deploy.

dq0.sdk.cli.query module

Query allows for the execution of db stats jobs

A query object will be created at runtime from a project instance.

class `dq0.sdk.cli.query.`**Query** (*project*)

Bases: `object`

A query source wrapper
Provides methods to run query jobs. A query always needs to be run in a project context. Alternative to using the query method directly from a Data instance. Allows for querying multiple data sources.

Args:

project (`dq0.sdk.cli.Project`): The project
this query belongs to

Attributes:

project (`dq0.sdk.cli.Project`): The project
this query belongs to

execute (*query*, *epsilon=1.0*, *tau=0.0*, *private_column=''*, *permissions=None*, *params=None*)

Run a query on the data sources defined by this Query instance.

Args:

query: string containing SQL epsilon: float; Epsilon value for differential private query. Default: 1.0 tau: float; Tau threshold value for private query. Default: 0.0 private_column: string; Private column for this query. Leave empty or omit for default value from metadata. permissions: optional; e.g. 'households<75' params: optional; e.g. 'p1=123'

Returns:

`dq0.sdk.cli.runner.QueryRunner` instance

for_data (*data*)

Specifiy which datasets are used in query. Args:

data (`list`) list of `dq0.sdk.cli.Data` instances included in query. Alternatively, pass a single `dq0.sdk.cli.Data` instance.

Returns:
dq0.sdk.cli.Query instance with set datasets

get_dataset_names ( )
Returns used dataset names as a single comma-separated string

1.2 dq0.sdk.data package

DQ0 SDK Data Package

This package contains the data connector abstract classes and implementing subclasses.

class dq0.sdk.data.**Source** (*path=None*, ***kwargs*)
Bases: abc.ABC
Abstract base class for all data connector sources available through the SDK.
Data sources classes provide a read method to read the data into memory or provide a data reader for the underlying source.

Args:
path (str, optional): Path to the data

Attributes:
uuid (str): The universally unique identifier of the data source. name (str): The data source's name type (str): The data source's distinct type (e.g. 'csv') description (str): The data source's description types: json object containing column type description data (pandas.DataFrame): The loaded data read_allowed (bool): True if this source can be read meta_allowed (bool): True if this source provides meta information types_allowed (bool): True if this source provides data type information stats_allowed (bool): True if this source provides statistics sample_allowed (bool): True if there is sample data for this source path (str): Path to the data (filepath, URI) sample_path (str): Path to the data containing sample data. (filepath, URI)

abstract read (***kwargs*)
Read data sources
This function should be used by child classes to read data or return a data handler to read streaming data.

Args:
kwargs: keyword arguments

Returns:
data read from the data source.

to_json ( )
Returns a json representation of this data sources information.

Returns:
data source description as json.

class dq0.sdk.data.**Transform**
Bases: dq0.sdk.projects.project.Project
Abstract base class for all transformations available through the SDK.
Transform classes provide a execute method to transform source data

Attributes:
model_type (str): type of this model instance. Options: 'keras'. uuid (str): UUID of this model. data_source (dq0.sdk.data.Source): dict of attached data sources.

abstract execute (*dataset=None*)
Execute transformation function

This function can be used by child classes to prepare data that dont need to be repeated for every training run.

1.2.1 Subpackages

dq0.sdk.data.binary package

DQ0 SDK Data Sources Binary package.

This package contains all binary table based data source implementation.

class dq0.sdk.data.binary.**Excel** (*path*)

Bases: dq0.sdk.data.source.Source

Data Source for MS Excel data.
Provides function to read in excel data.

Args:

path (str): Absolute path to the Excel file.

read (***kwargs*)

Read excel data source

Args:

kwargs: keyword arguments.

Should contain a 'sheet_name' argument to specify which excel sheet to load (None for all).

Returns:

excel data as pandas dataframe

class dq0.sdk.data.binary.**Feather** (*path*)

Bases: dq0.sdk.data.source.Source

Data Source for Apache Arrow Feather data.
Provides function to read in feather data.

Args:

path (str): Absolute path to the feather file.

read (***kwargs*)

Read feather data source

Args:

kwargs: keyword arguments.

Returns:

feather data as pandas dataframe

class dq0.sdk.data.binary.**HDF5** (*path*)

Bases: dq0.sdk.data.source.Source

Data Source for HDF5 PyTables data.
Provides function to read in hdf5 data.

Args:

path (str): Absolute path to the hdf5 file.

read (***kwargs*)

Read hdf5 data source

Args:

kwargs: keyword arguments.

Can contain a 'key' argument to specify a group within the pytable.

Returns:
hdf5 data as pandas dataframe

class dq0.sdk.data.binary.**ODF** (*path*)
Bases: dq0.sdk.data.binary.excel.Excel
Data Source for Open Document data.
Provides function to read in Open Document data.

Args:
path (str): Absolute path to the Open Document file.

class dq0.sdk.data.binary.**ORC** (*path*)
Bases: dq0.sdk.data.source.Source
Data Source for Apache ORC data.
Provides function to read in orc data.

Args:
path (str): Absolute path to the orc file.

read (***kwargs*)
Read orc data source

Args:

kwargs: keyword arguments.
May contain columns to define columsn to read.

Returns:
orc data as pandas dataframe

class dq0.sdk.data.binary.**Parquet** (*path*)
Bases: dq0.sdk.data.source.Source
Data Source for Apache Parquet data.
Provides function to read in parquet data.

Args:
path (str): Absolute path to the parquet file.

read (***kwargs*)
Read parquet data source

Args:

kwargs: keyword arguments.
May contain 'engine' to define whether to use pyarrow or fastparquet. May contain columns to define columsn to read.

Returns:
parquet data as pandas dataframe

class dq0.sdk.data.binary.**SAS** (*path*)
Bases: dq0.sdk.data.source.Source
Data Source for SAS data.
Provides function to read in SAS data.

Args:
path (str): Absolute path to the SAS file.

read (***kwargs*)
Read sas data source

Args:
kwargs: keyword arguments.

Returns:
sas data as pandas dataframe

class dq0.sdk.data.binary.**SPSS** (*path*)
Bases: dq0.sdk.data.source.Source
Data Source for SAS data.
Provides function to read in spss data.

Args:
path (str): Absolute path to the spss file.

read (***kwargs*)
Read spss data source

Args:
kwargs: keyword arguments.

Returns:
spss data as pandas dataframe

class dq0.sdk.data.binary.**Stata** (*path*)
Bases: dq0.sdk.data.source.Source
Data Source for stata data.
Provides function to read in stata data.

Args:
path (str): Absolute path to the stata file.

read (***kwargs*)
Read stata data source

Args:
kwargs: keyword arguments.

Returns:
stata data as pandas dataframe

Submodules

dq0.sdk.data.binary.excel module

Data Source for MS Excel files.

This source class provides access to Excel data as pandas dataframes.

class dq0.sdk.data.binary.excel.**Excel** (*path*)
Bases: dq0.sdk.data.source.Source
Data Source for MS Excel data.
Provides function to read in excel data.

Args:
path (str): Absolute path to the Excel file.

read (***kwargs*)
Read excel data source

Args:

kwargs: keyword arguments.
Should contain a 'sheet_name' argument to specify which excel sheet to load (None for all).

Returns:
excel data as pandas dataframe

dq0.sdk.data.binary.feather module

Data Source for Apache Arrow Feather files.

This source class provides access to feather data as pandas dataframes.

class dq0.sdk.data.binary.feather.**Feather** (*path*)

Bases: dq0.sdk.data.source.Source

Data Source for Apache Arrow Feather data.
Provides function to read in feather data.

Args:

path (str): Absolute path to the feather file.

read (***kwargs*)

Read feather data source

Args:

kwargs: keyword arguments.

Returns:

feather data as pandas dataframe

dq0.sdk.data.binary.hdf5 module

Data Source for HDF5 PyTables files.

This source class provides access to hdf5 data as pandas dataframes.

class dq0.sdk.data.binary.hdf5.**HDF5** (*path*)

Bases: dq0.sdk.data.source.Source

Data Source for HDF5 PyTables data.
Provides function to read in hdf5 data.

Args:

path (str): Absolute path to the hdf5 file.

read (***kwargs*)

Read hdf5 data source

Args:

kwargs: keyword arguments.

Can contain a 'key' argument to specify a group within the pytable.

Returns:

hdf5 data as pandas dataframe

dq0.sdk.data.binary.odf module

Data Source for Open Document files.

This source class provides access to Open Document data as pandas dataframes.

This class uses the read function of the ExcelSource class.

class dq0.sdk.data.binary.odf.**ODF** (*path*)

Bases: dq0.sdk.data.binary.excel.Excel

Data Source for Open Document data.
Provides function to read in Open Document data.

Args:

path (str): Absolute path to the Open Document file.

dq0.sdk.data.binary.orc module

Data Source for Apache ORC files.

This source class provides access to orc data as pandas dataframes.

class dq0.sdk.data.binary.orc.**ORC** (*path*)

Bases: dq0.sdk.data.source.Source

Data Source for Apache ORC data.
Provides function to read in orc data.

Args:

path (str): Absolute path to the orc file.

read (***kwargs*)

Read orc data source

Args:

kwargs: keyword arguments.

May contain columns to define columsn to read.

Returns:

orc data as pandas dataframe

dq0.sdk.data.binary.parquet module

Data Source for Apache Parquet files.

This source class provides access to parquet data as pandas dataframes.

class dq0.sdk.data.binary.parquet.**Parquet** (*path*)

Bases: dq0.sdk.data.source.Source

Data Source for Apache Parquet data.
Provides function to read in parquet data.

Args:

path (str): Absolute path to the parquet file.

read (***kwargs*)

Read parquet data source

Args:

kwargs: keyword arguments.

May contain 'engine' to define whether to use pyarrow or fastparquet. May contain columns to define columsn to read.

Returns:

parquet data as pandas dataframe

dq0.sdk.data.binary.sas module

Data Source for SAS files.

This source class provides access to SAS data as pandas dataframes.

class dq0.sdk.data.binary.sas.**SAS** (*path*)

Bases: dq0.sdk.data.source.Source

Data Source for SAS data.
Provides function to read in SAS data.

Args:

path (str): Absolute path to the SAS file.

read (***kwargs*)
Read sas data source

Args:
kwargs: keyword arguments.

Returns:
sas data as pandas dataframe

dq0.sdk.data.binary.spss module

Data Source for SPSS files.

This source class provides access to SPSS data as pandas dataframes.

class dq0.sdk.data.binary.spss.**SPSS** (*path*)
Bases: dq0.sdk.data.source.Source
Data Source for SAS data.
Provides function to read in spss data.

Args:
path (str): Absolute path to the spss file.

read (***kwargs*)
Read spss data source

Args:
kwargs: keyword arguments.

Returns:
spss data as pandas dataframe

dq0.sdk.data.binary.stata module

Data Source for Stata files.

This source class provides access to stata data as pandas dataframes.

class dq0.sdk.data.binary.stata.**Stata** (*path*)
Bases: dq0.sdk.data.source.Source
Data Source for stata data.
Provides function to read in stata data.

Args:
path (str): Absolute path to the stata file.

read (***kwargs*)
Read stata data source

Args:
kwargs: keyword arguments.

Returns:
stata data as pandas dataframe

dq0.sdk.data.image package

DQ0 SDK Image Data Sources dataset package.

class dq0.sdk.data.image.**Image** (*folderpath*)
Bases: dq0.sdk.data.source.Source
Data Source for image dataset.

Attributes:
folderpath (`string`): path or url to folder containing the images to load.

read ( )
Read the image data.

Returns:
data (`pandas.DataFrame`): image data as pd dataframe (no-channels, ch1, ch2, …)

to_json ( )
Returns a json representation of this data sources information.

Returns:
data source description as json.

Submodules

dq0.sdk.data.image.image module

Image Data Source.

This is a data source implementation for image data sets.

class `dq0.sdk.data.image.image.`**Image** (*folderpath*)
Bases: `dq0.sdk.data.source.Source`
Data Source for image dataset.

Attributes:
folderpath (`string`): path or url to folder containing the images to load.

read ( )
Read the image data.

Returns:
data (`pandas.DataFrame`): image data as pd dataframe (no-channels, ch1, ch2, …)

to_json ( )
Returns a json representation of this data sources information.

Returns:
data source description as json.

dq0.sdk.data.metadata package

DQ0 SDK Metadata Package

This package contains the data metadata handlers.

class `dq0.sdk.data.metadata.`**Metadata** (*filename=None*, *yaml=None*)
Bases: `object`
Metadata class.
Describes a data source via metadata information. See README.md for details.

Attributes:
name: parsed name property. description: parsed description property type: parsed type propertey. schemas: parsed schema metadata (see class Schema below) privacy_column: the unique privacy column for this data set

combine_with (*metadata*)
Combines two metadata instances. Adds the first schema of the given metadata object and all of its tables to this metadata object.

Returns:
The combined metadata object (self)

drop_columns_with_key_value (*key, value*)
Helper function that drops all columns from the metadata that have the given key value combination.

get_all_schema_names ( )
Helper function that returns a list of the names of all schemas in this metadata.

get_all_table_names ( )
Helper function that returns all available table names (across schemas) in this metadata.

get_all_tables (*only_names=False*)
Helper function that returns all available tables (across schemas) in this metadata.

get_feature_target_cols ( )
gets all column name from all tables and looks if is_feature or is defined

get_header ( )

read_from_yaml (*yaml_input*)
Reads metadata from the given yaml input.

Args:
yaml_input: open yaml file stream or yaml string.

read_from_yaml_file (*filename*)
Reads metadata from the given yaml file.

Args:
filename: the path to the yaml file.

to_dict (*sm=False*)
Returns a dict representation of this class.

Args:
sm: True to return the dict with the non-smartnoise properties stripped.

Returns:
Metadata as python dictionary.

to_dict_sm ( )
Returns a dict representation of this metadata in smartnoise format.

to_yaml (*sm=False*)
Writes metadata to a yaml string.

Args:
sm: True to return the dict with the non-smartnoise properties stripped.

Returns:
metadata as yaml string

to_yaml_file (*filename, sm=False*)
Writes metadata to a yaml file at the given path.

Args:
filename: the path to the yaml file. sm: True to return the dict with the non-smartnoise properties stripped.

Submodules

dq0.sdk.data.metadata.metadata module

Data Source Metadata information

Attributes and read / write functions for metadata structures.

class dq0.sdk.data.metadata.metadata.**Column** (*name*, *_type=''*, *bounded=None*, *lower=None*, *upper=None*, *use_auto_bounds=False*, *auto_bounds_prob=0.9*, *auto_lower=None*, *auto_upper=None*, *cardinality=0*, *allowed_values=None*, *private_id=False*, *selectable=False*, *mask=None*, *synthesizable=True*, *discrete=False*, *min_step=1.0*, *is_feature=False*, *is_target=False*)

Bases: object

Column represents a table column definition inside the metadata.

static from_meta (*column*, *meta*)

Create a column instance from the meta yaml part.

to_dict (*sm=False*)

Returns a dict representation of this class.

class dq0.sdk.data.metadata.metadata.**Metadata** (*filename=None*, *yaml=None*)

Bases: object

Metadata class.

Describes a data source via metadata information. See README.md for details.

Attributes:

name: parsed name property. description: parsed description property type: parsed type propertey. schemas: parsed schema metadata (see class Schema below) privacy_column: the unique privacy column for this data set

combine_with (*metadata*)

Combines two metadata instances. Adds the first schema of the given metadata object and all of its tables to this metadata object.

Returns:

The combined metadata object (self)

drop_columns_with_key_value (*key*, *value*)

Helper function that drops all columns from the metadata that have the given key value combination.

get_all_schema_names ( )

Helper function that returns a list of the names of all schemas in this metadata.

get_all_table_names ( )

Helper function that returns all available table names (across schemas) in this metadata.

get_all_tables (*only_names=False*)

Helper function that returns all available tables (across schemas) in this metadata.

get_feature_target_cols ( )

gets all column name from all tables and looks if is_feature or is defined

get_header ( )

read_from_yaml (*yaml_input*)

Reads metadata from the given yaml input.

Args:

yaml_input: open yaml file stream or yaml string.

read_from_yaml_file (*filename*)

Reads metadata from the given yaml file.

Args:

filename: the path to the yaml file.

to_dict (*sm=False*)

Returns a dict representation of this class.

Args:

sm: True to return the dict with the non-smartnoise properties stripped.

Returns:

Metadata as python dictionary.

to_dict_sm ( )

Returns a dict representation of this metadata in smartnoise format.

to_yaml (*sm=False*)

Writes metadata to a yaml string.

Args:

sm: True to return the dict with the non-smartnoise properties stripped.

Returns:

metadata as yaml string

to_yaml_file (*filename*, *sm=False*)

Writes metadata to a yaml file at the given path.

Args:

filename: the path to the yaml file. sm: True to return the dict with the non-smartnoise properties stripped.

class dq0.sdk.data.metadata.metadata.**Schema** (*name*, *size=0*, *connection=''*, *privacy_budget=0*, *privacy_budget_interval_days=0*, *synth_allowed=False*, *privacy_level=2*, *tables=None*)

Bases: object

Schema class.

Describes a data source unit via metadata information.

Attributes:

connection: data source connection URI name: name of the database size: the size of this database privacy_budget: parsed privacy budget property. privacy_budget_interval_days: parsed privacy budget reset interval in days. synth_allowed: true to allow synthesized data for exploration privacy_level: 0, 1, 2 in ascending order of privacy protection (default is 2). tables: parsed database metadata.

static from_meta (*schema*, *meta*)

Create a schema instance from the meta yaml part.

to_dict (*sm=False*)

Returns a dict representation of this class.

class dq0.sdk.data.metadata.metadata.**Table** (*name*, *use_original_header=True*, *header_row=None*, *row_privacy=False*, *rows=0*, *max_ids=1*, *sample_max_ids=True*, *use_dpsu=False*, *clamp_counts=False*, *clamp_columns=True*, *censor_dims=False*, *tau=None*, *columns=None*)

Bases: object

Table represents a table definition inside the metadata.

static from_meta (*table*, *meta*)

Create a table instance from the meta yaml part.

to_dict (*sm=False*)

Returns a dict representation of this class.

dq0.sdk.data.preprocessing package

DQ0 SDK Data Preprocessing Package

This package contains helper scripts for data preprocessing.

Submodules

dq0.sdk.data.preprocessing.preprocessing module

Data preprocessing utils.

Collection of functions for preprocessing datasets, including data-scrubbing, extraction of count features from corpora of documents, missing-data handling, etc.

Copyright 2020, Gradient Zero

`dq0.sdk.data.preprocessing.preprocessing.`**`extract_count_features_from_text_corpus`** (*tr_data_list*, *test_data_list*)

Extracts count features from the given list of documents.

Args:

tr_data_list (`list`): list of text documents test_data_list (`list`): list of text documents

Returns:

Sparse Scipy matrices with the extracted features

`dq0.sdk.data.preprocessing.preprocessing.`**`handle_missing_data`** (*dataset_df*, *mode='imputation'*, *imputation_method_for_cat_feats='unknown'*, *imputation_method_for_quant_feats='median'*, *categorical_features_list=None*, *quantitative_features_list=None*)

Fills missing data values.

Args:

dataset_df (`pandas.DataFrame`): the data frame to transform. mode (`str`): either 'imputation' or 'dropping' to fill or drop missing values. imputation_method_for_cat_feats (`str`): either 'unknown' or 'most_common_cat'.

'unkowon' will replace all missing categorical feature values by 'Unknown'. 'most_common_cat' will replace the missing values with the most common categorical feature.

imputation_method_for_quant_feats (str): Will replace the missing quantitative feature values

with either 'mean' or 'median' value.

categorical_features_list (`list`): list of categorical features quantitative_features_list (`list`): list of quantitative features

Returns:

Transformed data frame.

`dq0.sdk.data.preprocessing.preprocessing.`**`scale_pixels`** (*X_np_a*, *max_pixel_intensity*)

Scale pixel values to be in [0, 1] to help gradient-descent optimization.

Args:

X_np_a (`numpy.ndarray`): Matrix of pixel intensities max_pixel_intensity (int): normalization constant set by user

Returns:

matrix of scaled intensities of the pixels

`dq0.sdk.data.preprocessing.preprocessing.`**`train_test_split`** (*X_df*, *y_ts*, *num_tr_instances*)

Splits the given X and y data in train and test sets.

Assumption: train instances on top, test instances at the bottom

TODO:
Make this more robust by adding a column defining tr / test status and split based on it rather than on above assumption

Args:
X_df (pandas.DataFrame): data frame containing the X values y_ts (pandas.DataFrame): data frame containing the y values num_tr_instances (int): The number of desired training instances in

the resulting split.

Returns:
X_train_df (pandas.DataFrame): X train split X_test_df (pandas.DataFrame): X test split y_train_ts (pandas.DataFrame): y train split y_test_ts (pandas.DataFrame): y test split

dq0.sdk.data.preprocessing.preprocessing.**univariate_feature_selection** (*num_top_ranked_feats_to_keep*, *X_train*, *y_train*, *X_test*, *technique*, *feature_names_list=None*, *verbose=False*)
Univariate feature selection.

Args:
num_top_ranked_feats_to_keep (int): Keep top n features X_train (numpy.ndarray): Training input samples y_train (numpy.ndarray): Target values X_test (numpy.ndarray): Test samples technique (str): Selection technique.

Either 'chi-squared test' or 'mutual information'.
feature_names_list (list): List of features. verbose (bool): True to print output.

Returns:
X_train (numpy.ndarray): transformed X train set X_test (numpy.ndarray): transformed X test set selected_feature_lsit (list): List of selected features

dq0.sdk.data.sql package

DQ0 SDK Data Sources SQL package.

This package contains all SQL data source implementation.

class dq0.sdk.data.sql.**BigQuery** (*connection_string*)
Bases: dq0.sdk.data.sql.sql.SQL
Data Source for BigQuery data.
Provides function to read in BigQuery data.

Args:
connection_string (str): The BigQuery project.

execute (*query*, ***kwargs*)
Execute SQL query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

class dq0.sdk.data.sql.**Drill** (*connection_string*)
Bases: dq0.sdk.data.sql.sql.SQL
Data Source for Apache Drill data.
Provides function to read in drill data.
Drill connection string: 'drill+sadrill://<username>:<password>@<host>:<port>/<storage_plugin>?use_ssl=True'

Args:
connection_string (str): The drill connection string.

execute (*query*, ***kwargs*)
Execute drill query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

class dq0.sdk.data.sql.**MSSQL** (*connection_string*)
Bases: dq0.sdk.data.sql.sql.SQL
Data Source for MSSQL data.
Provides function to read in MSSQL data.
MSSQL connection string: 'mssql+pyodbc://<username>:<password>@<dsnname>'

Args:
connection_string (str): The mssql connection string.

execute (*query*, ***kwargs*)
Execute MSSQL query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

class dq0.sdk.data.sql.**MySQL** (*connection_string*)
Bases: dq0.sdk.data.sql.sql.SQL
Data Source for MySQL data.
Provides function to read in MySQL data.
MySQL connection string: 'mysql+mysqlconnector://<user>:<password>@<host>[:<port>]/<dbname>'

Args:
connection_string (str): The mysql connection string.

execute (*query*, ***kwargs*)
Execute MYSQL query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

class dq0.sdk.data.sql.**Oracle** (*connection_string*)
Bases: dq0.sdk.data.sql.sql.SQL
Data Source for Oracle data.
Provides function to read in Oracle data.
Oracle connection string: 'oracle+cx_oracle://user:pass@host:port/dbname[?key=value&key=value...]'

Args:
connection_string (str): The oracle connection string.

execute (*query*, ***kwargs*)
Execute Oracle SQL query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

class dq0.sdk.data.sql.**PostgreSQL** (*connection_string*)
Bases: dq0.sdk.data.sql.sql.SQL
Data Source for PostgreSQL data.
Provides function to read in PostgreSQL data.
PostgreSQL connection string: 'postgresql+psycopg2://user:password@/dbname'

Args:
connection_string (str): The postgresql connection string.

execute (*query*, ***kwargs*)
Execute the Postgres query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

class dq0.sdk.data.sql.**Redshift** (*connection_string*)
Bases: dq0.sdk.data.sql.sql.SQL
Data Source for Amazon Redshift data.
Provides function to read in Amazon Redshift data.
Amazon Redshift connection string: 'redshift+psycopg2://username@host.amazonaws.com:5439/database'

Args:
connection_string (str): The redshift connection string.

execute (*query*, ***kwargs*)
Execute Redshift SQL query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

class dq0.sdk.data.sql.**SAPHana** (*connection_string*)
Bases: dq0.sdk.data.sql.sql.SQL
Data Source for SAP Hana data.
Provides function to read in Snowflake data.
SAP Hana connection string: 'hana://<user>:<password>@<host>:<port>/'

Args:
connection_string (str): The saphana connection string.

execute (*query*, ***kwargs*)
Execute SAP SQL query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

class dq0.sdk.data.sql.**Snowflake** (*connection_string*)
Bases: dq0.sdk.data.sql.sql.SQL
Data Source for Snowflake data.
Provides function to read in Snowflake data.

Snowflake connection string: 'snowflake://<user>:<password>@<account>/'

Args:
connection_string (`str`): The snowflake connection string.

`execute` (*query*, ***kwargs*)
Execute Snowflake query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

class `dq0.sdk.data.sql.`**`SQLite`** (*connection_string*)
Bases: `dq0.sdk.data.sql.sql.SQL`
Data Source for SQLite data.
Provides function to read in SQLite data.
SQLite connection string: 'sqlite:///path/to/database.db'

Args:
connection_string (`str`): The sqlite connection string.

`execute` (*query*, ***kwargs*)
Execute SQL query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

Submodules

dq0.sdk.data.sql.big_query module

Data Source for Big Query.

This source class provides access to data from Google BigQuery as pandas dataframes.

The BigQuery adapter assumes the authentication is managed in the runtime environment!

class `dq0.sdk.data.sql.big_query.`**`BigQuery`** (*connection_string*)
Bases: `dq0.sdk.data.sql.sql.SQL`
Data Source for BigQuery data.
Provides function to read in BigQuery data.

Args:
connection_string (`str`): The BigQuery project.

`execute` (*query*, ***kwargs*)
Execute SQL query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

dq0.sdk.data.sql.drill module

Data Source for Apache Drill.

This source class provides access to data reveived via Apache Drill as pandas dataframes.

Based on sqlalchemy with drill driver extension: https://github.com/JohnOmernik/sqlalchemy-drill

class dq0.sdk.data.sql.drill.**Drill** (*connection_string*)

Bases: dq0.sdk.data.sql.sql.SQL

Data Source for Apache Drill data.

Provides function to read in drill data.

Drill connection string: 'drill+sadrill://<username>:<password>@<host>:<port>/<storage_plugin>?use_ssl=True'

Args:

connection_string (str): The drill connection string.

execute (*query*, ***kwargs*)

Execute drill query

Args:

query: SQL Query to execute kwargs: keyword arguments

Returns:

SQL ResultSet as pandas dataframe

dq0.sdk.data.sql.mssql module

Data Source for MSSQL.

This source class provides access to data from MSSQL as pandas dataframes.

Based on sqlalchemy with pyodbc driver.

class dq0.sdk.data.sql.mssql.**MSSQL** (*connection_string*)

Bases: dq0.sdk.data.sql.sql.SQL

Data Source for MSSQL data.

Provides function to read in MSSQL data.

MSSQL connection string: 'mssql+pyodbc://<username>:<password>@<dsnname>'

Args:

connection_string (str): The mssql connection string.

execute (*query*, ***kwargs*)

Execute MSSQL query

Args:

query: SQL Query to execute kwargs: keyword arguments

Returns:

SQL ResultSet as pandas dataframe

dq0.sdk.data.sql.mysql module

Data Source for MySQL.

This source class provides access to data from MySQL as pandas dataframes.

Based on sqlalchemy with mysqlconnector driver.

class dq0.sdk.data.sql.mysql.**MySQL** (*connection_string*)

Bases: dq0.sdk.data.sql.sql.SQL

Data Source for MySQL data.

Provides function to read in MySQL data.

MySQL connection string: 'mysql+mysqlconnector://<user>:<password>@<host>[:<port>]/<dbname>'

Args:

connection_string (str): The mysql connection string.

execute (*query*, ***kwargs*)
Execute MYSQL query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

dq0.sdk.data.sql.oracle module

Data Source for Oracle DB.

This source class provides access to data from Oracle DB as pandas dataframes.

Based on sqlalchemy with Oracle-CX driver.

class dq0.sdk.data.sql.oracle.**Oracle** (*connection_string*)
Bases: dq0.sdk.data.sql.sql.SQL
Data Source for Oracle data.
Provides function to read in Oracle data.
Oracle connection string: 'oracle+cx_oracle://user:pass@host:port/dbname[?key=value&key=-value…]'

Args:
connection_string (str): The oracle connection string.

execute (*query*, ***kwargs*)
Execute Oracle SQL query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

dq0.sdk.data.sql.postgresql module

Data Source for PostgreSQL.

This source class provides access to data from PostgreSQL as pandas dataframes.

Based on sqlalchemy with psycopg2 driver.

class dq0.sdk.data.sql.postgresql.**PostgreSQL** (*connection_string*)
Bases: dq0.sdk.data.sql.sql.SQL
Data Source for PostgreSQL data.
Provides function to read in PostgreSQL data.
PostgreSQL connection string: 'postgresql+psycopg2://user:password@/dbname'

Args:
connection_string (str): The postgresql connection string.

execute (*query*, ***kwargs*)
Execute the Postgres query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

dq0.sdk.data.sql.redshift module

Data Source for Amazon Redshift.

This source class provides access to Amazon Redshift data as pandas dataframes.

Based on sqlalchemy with sqlalchemy-redshift (psycopg2) driver extension: https://github.com/SAP/sqlalchemy-hana

class dq0.sdk.data.sql.redshift.**Redshift** (*connection_string*)

Bases: dq0.sdk.data.sql.sql.SQL

Data Source for Amazon Redshift data.

Provides function to read in Amazon Redshift data.

Amazon Redshift connection string: 'redshift+psycopg2://username@host.amazonaws.com:5439/database'

Args:

connection_string (str): The redshift connection string.

execute (*query*, ***kwargs*)

Execute Redshift SQL query

Args:

query: SQL Query to execute kwargs: keyword arguments

Returns:

SQL ResultSet as pandas dataframe

dq0.sdk.data.sql.sap_hana module

Data Source for SAP Hana.

This source class provides access to SAP Hana data as pandas dataframes.

Based on sqlalchemy with sqlalchemy-hana driver extension: https://github.com/SAP/sqlalchemy-hana

class dq0.sdk.data.sql.sap_hana.**SAPHana** (*connection_string*)

Bases: dq0.sdk.data.sql.sql.SQL

Data Source for SAP Hana data.

Provides function to read in Snowflake data.

SAP Hana connection string: 'hana://<user>:<password>@<host>:<port>/'

Args:

connection_string (str): The saphana connection string.

execute (*query*, ***kwargs*)

Execute SAP SQL query

Args:

query: SQL Query to execute kwargs: keyword arguments

Returns:

SQL ResultSet as pandas dataframe

dq0.sdk.data.sql.snowflake module

Data Source for Snowflake.

This source class provides access to data reveived Snowflake as pandas dataframes.

Based on sqlalchemy with snowflake-sqlalchemy driver extension: https://github.com/snowflakedb/snowflake-sqlalchemy

class dq0.sdk.data.sql.snowflake.**Snowflake** (*connection_string*)

Bases: dq0.sdk.data.sql.sql.SQL

Data Source for Snowflake data.
Provides function to read in Snowflake data.
Snowflake connection string: 'snowflake://<user>:<password>@<account>/'

Args:
connection_string (`str`): The snowflake connection string.

`execute` (*query*, ***kwargs*)
Execute Snowflake query

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

dq0.sdk.data.sql.sql module

Data Source base class for SQL-based data sources.

class `dq0.sdk.data.sql.sql.`**`SQL`** (*connection_string*)
Bases: `dq0.sdk.data.source.Source`
Data Source base class for SQL data.

Attributes:
query (`str`): SQL query. connection_string (`str`): General purpose SQL data source connection string. engine: the used sqlalchemy engine engine_connection: the active sql connection type: the datasource type

Args:
connection (`str`): General purpose SQL data source connection string.

abstract `execute` (*query=None*, ***kwargs*)
Execute SQL query
This function should be used by child classes to execeute SQL queries

Args:
query: SQL Query to execute kwargs: keyword arguments

Returns:
SQL ResultSet as pandas dataframe

`get_connection` ( )
Returns the active sql connection.
Initiates the connection if not already done.

Returns:
Active sql connection. Throws error if engine is not set.

`read` (***kwargs*)
Runs overriden 'execute' method with query parameter

Args:
kwargs: keyword arguments

Returns:
CSV data as pandas dataframe

`to_json` ( )
Returns a json representation of this data sources information.

Returns:
data source description as json.

dq0.sdk.data.sql.sqlite module

Data Source for SQLite.

This source class provides access to data from SQLite as pandas dataframes.

Based on sqlalchemy with standard sqlite driver.

class dq0.sdk.data.sql.sqlite.**SQLite** (*connection_string*)

Bases: dq0.sdk.data.sql.sql.SQL

Data Source for SQLite data.
Provides function to read in SQLite data.
SQLite connection string: 'sqlite:///path/to/database.db'

Args:

connection_string (str): The sqlite connection string.

execute (*query*, ***kwargs*)

Execute SQL query

Args:

query: SQL Query to execute kwargs: keyword arguments

Returns:

SQL ResultSet as pandas dataframe

dq0.sdk.data.text package

DQ0 SDK Data Sources Text package.

This package contains all text based data source implementation.

class dq0.sdk.data.text.**CSV** (*path*, *feature_cols=None*, *target_cols=None*, *header=None*)

Bases: dq0.sdk.data.source.Source

Data Source for CSV data.
Provides function to read in csv data.

Args:

path (str): Absolute path to the CSV file.

read (***kwargs*)

Read CSV data sources

Args:

kwargs: keyword arguments

Returns:

CSV data as pandas dataframe

class dq0.sdk.data.text.**JSON** (*path*)

Bases: dq0.sdk.data.source.Source

Data Source for JSON data.
Provides function to read in json data.

Args:

path (str): Absolute path to the JSON file.

read (***kwargs*)

Read json data sources

Args:

kwargs: keyword arguments

Returns:

json data as pandas dataframe

Raises:
IOError: if file was not found

Submodules

dq0.sdk.data.text.csv module

Data Source for CSV files.

This source class provides access to CSV data as pandas dataframes.

class dq0.sdk.data.text.csv.**CSV** (*path*, *feature_cols=None*, *target_cols=None*, *header=None*)
Bases: dq0.sdk.data.source.Source
Data Source for CSV data.
Provides function to read in csv data.

Args:
path (str): Absolute path to the CSV file.

read (***kwargs*)
Read CSV data sources

Args:
kwargs: keyword arguments

Returns:
CSV data as pandas dataframe

dq0.sdk.data.text.json module

Data Source for JSON files.

This source class provides access to JSON data as pandas dataframes.

class dq0.sdk.data.text.json.**JSON** (*path*)
Bases: dq0.sdk.data.source.Source
Data Source for JSON data.
Provides function to read in json data.

Args:
path (str): Absolute path to the JSON file.

read (***kwargs*)
Read json data sources

Args:
kwargs: keyword arguments

Returns:
json data as pandas dataframe

Raises:
IOError: if file was not found

dq0.sdk.data.utils package

DQ0 SDK Data Utils Package

This package contains general data helper functions.

Submodules

dq0.sdk.data.utils.plotting module

Helper functions for plotting data analysis results.

dq0.sdk.data.utils.plotting.**add_shared_axis_labels** (*fig*, *x_label*, *y_label*, *font_size=15*)

Add label to x and y axes shared by the subplots in the figure referred to by the input "fig" handle.

Input "fig" handle is assumed to be generated by, e.g:

fig, ax = plt.subplots(nrows=…,
ncols=…, sharex=True, sharey=True)

Args:

fig: multi-plots figure handle x_label: string with label for shared x-axis y_label: string with label for shared y-axis font_size: size of font for axes labels

dq0.sdk.data.utils.plotting.**compute_confusion_matrix** (*y_true*, *y_pred*, *normalize*)

Computes the confusion matrix.

Pass the labels list to the confusion_matrix function in order to index the confusion matrix based on the order of labels in the list. Using labels_list ensures that the confusion matrix matches the ticks in the figure plotting it.

From Scikit documentation:

labels: array-like of shape (n_classes), default=None List of labels to index the matrix. This may be used to reorder or select a subset of labels. If None is given, those that appear at least once in y_true or y_pred are used in sorted order.

Args:

y_true (numpy.ndarray): target y values. y_pred (numpy.ndarray): predicted y values. normalize (bool): True to normalize the matrix.

dq0.sdk.data.utils.plotting.**get_param_configuration_for_publication_quality_plot** ( )

Set matplotlib.pyplot parameters for publication-quality plot

Returns:

linewidth, markersize, figsize, fontsize

dq0.sdk.data.utils.plotting.**plot_bars** (*bar_heights*, ***kwargs*)

Generate a bar plot.

Args:

bar_heights (list): heights of the bars. **kwargs:

Returns:

dq0.sdk.data.utils.plotting.**plot_confusion_matrix** (*y_true*, *y_pred*, *output_folder*, *ticks_rotation='horizontal'*, *cmap=<matplotlib.colors.LinearSegmentedColormap object>*, *part_of_fn_describing_matrix=''*)

Print and plot the confusion matrix.

Args:

y_true (numpy.ndarray): target y values. y_pred (numpy.ndarray): predicted y values. output_folder (str): Path to the output folder for the matrix png image. ticks_rotation: can be 'horizontal', 'vertical' or float cmap: Matplotlib color map. part_of_fn_describing_matrix (str): function description for matrix.

dq0.sdk.data.utils.plotting.**plot_confusion_matrix_for_scikit_classifier** (*classifier*, *X_test_np_a*, *y_test_np_a*, *class_names=None*, *xticks_rotation='horizontal'*, *part_of_fn_describing_matrix=''*, *output_folder='../data/output/'*)

Plots a confusion matrix for the scikit classifier.

Args:

classifier: a Scikit trained classifier! E.g., after creating the
classifier object, its fit() method had been invoked.

X_test_np_a (numpy.ndarray): X test data y_test_np_a (numpy.ndarray): y test data

class_names (`list`): optional list of labels to index the confusion matrix.

This may be used to reorder or select a subset of labels. If None is given, those that appear at least once in y_true or y_pred are used in sorted order. To get the labels:

class_names = sklearn.utils.multiclass.unique_labels(y_true)

To use only the labels that appear in the data:

class_names = sklearn.utils.multiclass.unique_labels(y_true, y_pred)

xticks_rotation: can be 'horizontal', 'vertical' or float part_of_fn_describing_matrix (`str`): function description for matrix. output_folder (`str`): Path to the output folder for the matrix png image.

`dq0.sdk.data.utils.plotting.`**`plot_decision_tree`** (*dec_tree*, *folder*)

Plot scikit-learn decision tree.

Args:

dec_tree: The decision tree to plot. folder (`str`): Folder to save the figure to.

`dq0.sdk.data.utils.plotting.`**`plot_hist`** (*bin_edges*, *bin_heights*, ***kwargs*)

Plot a histogram. By default, the bins are coloured based on their heights. Coloring is based on a diverging colormap where highest bins are given reddish colors, while lower bins are given bluish colors.

Args:

bin_edges (list): edges of the bins. List lengths is number of bins

plus one.

bin_heights (list): heights of the bins. **kwargs:

`dq0.sdk.data.utils.plotting.`**`save_figure`** (*fig*, *figure_name*, *dpi=300*, *tracker=None*, *tracker_output_path=None*, *output_folder_path=None*)

Save figure referenced by input figure handle "fig". It also closes the figure.

Args:

fig: figure handle figure_name: name of figure (without file extension) dpi (int): dots per inch. For printing and most screens, 150 is pretty

good, 300 is clear, and 600 is spectacular. 1200 or higher can come in handy if you want to be able to do a lot of zooming in, but your image can start to get very big on disk at that resolution. Default: 300.

tracker: instance of tracker tracker_output_path (str): path to folder where the figure will be

saved.

output_folder_path (str): path to folder where the figure will be

saved, if not saved via tracker.

`dq0.sdk.data.utils.plotting.`**`scatterplot`** (*x*, *y*, *working_folder='../data/working/'*, *hue=None*, *part_of_fn_describing_data=''*)

Plots a scatterplot graph with seaborn.

Args:

x: names of x variables in data or vector data y: names of y variables in data or vector data working_folder (`str`): working directory hue: Grouping variable that will produce points with different colors part_of_fn_describing_data (`str`): function description for plot.

`dq0.sdk.data.utils.plotting.`**`select_bar_colors`** (*bar_heights*, *evenly_spaced_interval=False*)

Define a color for each bar in a bar-plot.

Args:

bar_heights (np.ndarray): list of heights of the bars. Order

matters: bar_heights[0] refers to the height of the leftmost bar.

evenly_spaced_interval (bool): Boolean flag. If False, bar coloring
based on height of bins. If true, bar coloring by order of the bins: the selected colors are equally spaced in the color map. The latter option may be useful if the bins are ordered by their height and the color map is sequential or diverging.

Returns:
list of colors

dq0.sdk.data.utils.plotting.**visualize_categorical_distribution** (*series*, *output_folder*, ***kwargs*)
Show the proportion of observations in each category using bars. Basically, it visualizes the discrete distribution of the data and the resulting plot can be interpreted as an histogram across a categorical, instead of quantitative, variable.

Args:

series (pandas.Series): categorical (aka discrete) data to be
visualized.

output_folder: path to folder where the generated figure will be
saved.

dq0.sdk.data.utils.plotting.**visualize_continuous_distribution** (*series*, *output_folder*, ***kwargs*)
Generate univariate histogram showing the data distribution. Histogram bars show the proportions of observations falling in each bin.

Args:
series (pandas.Series): quantitative data to be visualized. output_folder: path to folder where the generated figure will be

saved.

dq0.sdk.data.utils.util module

General data utility functions.

dq0.sdk.data.utils.util.**case_insensitive_str_comparison** (*string1*, *string2*)
Compare strings case insensitive.

dq0.sdk.data.utils.util.**check_data_structure_type_consistency** (*X_train*, *X_test*, *y_train*, *y_test*)
Check for type consistency among train and test X, y.

Type consistency is achieved iff:

- all X and y must are Pandas objects
- all X and y must are Numpy objects
- y_train, y_test must have the same number of dimensions

Mixture of Pandas and Numpy objects is not allowed.

Args:
X_train: Numpy array or Pandas DataFrame X_test: Numpy array or Pandas DataFrame y_train: Numpy (also non-dimensional) array or Pandas Series y_test: Numpy (also non-dimensional) array or Pandas Series

dq0.sdk.data.utils.util.**check_for_valid_numerical_encoding_of_labels** (*labels*)
Checks whether the labels are numerical labels satisfying the following requirements:

1. each label is an integer greater or equal to zero
2. the smallest label is zero

The labels encoded by applying "sklearn.preprocessing.LabelEncoder" satisfy above requirements.

Args:

labels: array-like list of labels to check. Can even be a column vector.

Returns:

is_valid: True if the input labels satisfy above requirements, False if not.

dq0.sdk.data.utils.util.**compute_metrics_scores** (*y*, *y_pred_np_a*, *metrics_list*)

Iterate through metrics_list and compute each metric in the list. Each list item is expected to be an instance of a tensorflow.keras.metrics class. So this function call must be preceded by the call to function instantiate_metrics_from_name.

Args:

y: vector with actual classification labels or regression scores y_pred_np_a: vector with predicted classification labels or regression

scores

metrics_list: list of instances of metric classes

Returns:

dictionary with (metric name, metric score) pairs

dq0.sdk.data.utils.util.**concatenate_train_test_datasets** (*X_train*, *X_test*, *y_train*, *y_test*)

Concetenates train and test datasets

Args:

X_train: Numpy array or Pandas DataFrame X_test: Numpy array or Pandas DataFrame y_train: Numpy (also non-dimensional) array or Pandas Series y_test: Numpy (also non-dimensional) array or Pandas Series

Returns:

Concatenated X and y

dq0.sdk.data.utils.util.**concatenate_train_test_datasets_np_array** (*X_train_np_a*, *X_test_np_a*, *y_train_np_a*, *y_test_np_a*)

Concatenates train and test datasets

Args:

X_train_np_a: numpy array X_test_np_a: numpy array y_train_np_a: numpy (also non-dimensional) array y_test_np_a: numpy (also non-dimensional) array

Returns:

Concatenated X and y

dq0.sdk.data.utils.util.**concatenate_train_test_datasets_pd_Dataframes** (*X_train_df*, *X_test_df*, *y_train_se*, *y_test_se*)

Concatenates train and test datasets

Args:

X_train_df: Pandas DataFrame X_test_df: Pandas DataFrame y_train_se: Pandas Series y_test_se: Pandas Series

Returns:

Concatenated X and y

dq0.sdk.data.utils.util.**copy_obj_attributes** (*obj_from*, *obj_to*, *attributes_l=None*)

Copy attributes of class instance (object) "obj_from" to class instance (object) "obj_to". The two objects are assumed to be instances of the same class.

Args:

obj_from: class instance (object) to copy from obj_to: class instance (object) to copy to attributes_l: list of attributes to be copied. If none, a blind copy is performed, where all

attributes of obj_from are copied to obj_to.

Returns:
obj_to

dq0.sdk.data.utils.util.**dataframe_has_columns_of_these_types** (*df, types_list*)
Returns true if the dataset has the given types.

Args:
df: The dataframe to inspect. types_list: list of types to check for.

Returns:
True if the types are present in the data frame.

dq0.sdk.data.utils.util.**datasets_are_equal** (*d1, d2*)
Compare two datasets for equality.

Args:
d1: Numpy array or Pandas DataFrame or Pandas Series d2: Numpy array or Pandas DataFrame or Pandas Series

Returns:
Boolean value True / False if d1 and d2 are / are not equal

dq0.sdk.data.utils.util.**dump_model** (*model, path='./data/output', name='model.pickle'*)
Pickle dump given model.

dq0.sdk.data.utils.util.**empty_folder** (*path_folder_tbr*)
Empties or creates the given folder.

dq0.sdk.data.utils.util.**estimate_freq_of_labels** (*y*)
Estimate the frequency of labels in y.

dq0.sdk.data.utils.util.**format_float_lower_than_1** (*float_value, abs_tol=1e-08*)
Generate a string representation for the input float number with the 0 value of the unit being removed if the absolute value of the float number is smaller than one. E.g, 0.234 is converted into ".234".

Args:
float_value (float): input float number abs_tol (float): tolerance value for equality to zero

Returns:
str representation of the input float number with redundant 0 for unit removed (if any).

dq0.sdk.data.utils.util.**get_categorical_and_quantitative_features_list** (*dataset, target_feature*)
Define list of categorical and quantitative features of the input dataset.

Naming convention:
"categorical" features: self-explanatory name "quantitative" features: continuous, discrete or ordinal values. The

term "quantitative" is preferred to "numerical" because also category labels can be numerical values.

"target" feature: it contains the learning signal (e.g., labels for
a classification problem)

Args:
dataset: Pandas DataFrame target_feature: feature containing the learning signal (e.g.,

labels for a classification problem)

Returns:
Python list of categorical and quantitative features

dq0.sdk.data.utils.util.**get_fn** (*file_path*)
Get filename and extension for file path.

dq0.sdk.data.utils.util.**get_percentage_freq_of_values** (*x_np_a*)
Compute percentage frequencies of values in the input Numpy array

Args:
x_np_a (numpy.ndarray): array of values parse

Returns:
dictionary with percentage frequencies of values

dq0.sdk.data.utils.util.**initialize_rnd_numbers_generators_state** (*seed=1*, *verbose=True*)
Initialize tf random generator.

Args:
seed (int, optional): random seed. Default is 1. verbose (bool, optional): Boolean flag to print seed used.

dq0.sdk.data.utils.util.**instantiate_metrics_from_name** (*metrics_list*)
Instantiate a metric class from tensorflow.keras.metrics
Iterate through metrics_list and replace each string (each string contains a metric name) with an instance of the corresponding metric class.

Args:

metrics_list: list of metrics defined by user. It may contain
objects of metric classes or strings with metric names.

Returns:
modified metrics_list

dq0.sdk.data.utils.util.**is_numeric** (*array*)
Determine whether the argument has a numeric datatype, when converted to a NumPy array.
Booleans, unsigned integers, signed integers, floats and complex numbers are the kinds of numeric datatype.

array : *array-like*
The array to check.

is_numeric : *bool*
True if the array has a numeric datatype, False if not.

dq0.sdk.data.utils.util.**list_subfloders** (*path='./'*, *s_prefix=None*)
List sub folders. Subfolder name starts with given s_prefix ('_starting_with_certain_name_prefix')

dq0.sdk.data.utils.util.**load_model_from_file** (*path*)
Load model from file.

dq0.sdk.data.utils.util.**load_params_from_config_file** (*yaml_file_path*)
Load parameters from YAML configuration file.

Args:
file_path (str): path to file. Defaults to *config.yml*

Raises:
FileNotFoundError: yaml config file not found

Returns:
parameters loaded from yaml file

dq0.sdk.data.utils.util.**manage_rnd_num_generators_state** (*action*)
Save and restore the internal states of the random number generators used

Args:
action (str): Manage action. Options: 'save'

dq0.sdk.data.utils.util.**missing_values_table** (*df*)
Generate per feature stats about missing values to preview the missing values and the % of missing values in each column

dq0.sdk.data.utils.util.**move_files** (*l_path_files*, *s_dest_folder*)
Move files.

dq0.sdk.data.utils.util.**numerical_datasets_are_equal** (*d1*, *d2*, *approx_error=0.0001*)
Compare two numerical datasets for equality. A small tolerance value is considered for floating-point error mitigation. I.e., values d1[i, j] and d2[i,j] are considered equal iff:

abs(d1[i,j] - d2[i,j]) < approx_error

Input Pandas DataFrames / Series (if any) must not contain non-numeric values. d1 and d2 types may be different.

Args:
d1: Numpy array or Pandas DataFrame or Pandas Series d2: Numpy array or Pandas DataFrame or Pandas Series approx_error: tolerance value for equality in floating-point arithmetic

Returns:
Boolean value True / False if d1 and d2 are / are not equal

dq0.sdk.data.utils.util.**perform_stratified_random_sampling** (*df*, *col_name*, *sample_size*)
Generate stratified sample of size "sample_size" where the proportion of instances with value "A" for "col_name" in the stratified sample matches the proportion of instances with value "A" for "col_name" in the larger DataFrame. This holds for every distinct value "A" of "col_name".

dq0.sdk.data.utils.util.**pretty_display_string_on_terminal** (*s*)
Trim string to fit on terminal (assuming 80-column display)

dq0.sdk.data.utils.util.**pretty_print_dict** (*d*, *indent_steps=1*, *indent_unit=' '*, *logger_fun=None*)
Print dictionary.

dq0.sdk.data.utils.util.**pretty_print_strings_list** (*l_strings*, *s_list_name=None*)
Print string list.

dq0.sdk.data.utils.util.**print_dataset_info** (*df_dataset*, *s_title*)
Print some info about the given dataset.

Args:
df_dataset: data frame to print. s_title: Title for print output.

dq0.sdk.data.utils.util.**print_details_about_df_columns** (*df_dataset*)
Print details about columns of dataset.

Args:
df_dataset: data frame to inspect.

dq0.sdk.data.utils.util.**print_evaluation_res** (*res*, *dataset_type*, *model_metrics=None*)
Print the results of call of trainer.evaluate()

Args:
res (`dict`): Results returned by trainer.evaluate() dataset_type (`str`): string with two

possible values: “training” or “test” model_metrics (`list`): list of metrics specified in user model

`dq0.sdk.data.utils.util.`**`print_full_df`** (*df_dataset*)
Print whole dataframe. By default, just reduced output is printed

`dq0.sdk.data.utils.util.`**`print_human_readable_elapsed_time_value`** (*elapsed_cpu_time_sec*, *s_tmp*)
Print elapsed time in human readable format.

`dq0.sdk.data.utils.util.`**`print_summary_stats`** (*ts*, *percentiles*, *s_col*)
Print stats.

`dq0.sdk.data.utils.util.`**`redirect_stdout_stderr_streams_to_file`** (*log_file*)
Redirect the stdout and stderr streams to the given log file.

`dq0.sdk.data.utils.util.`**`restore_stdout_stderr_streams`** (*file_stream*, *orig_stdout*, *orig_stderr*)
Restore back stdout and stderr.

`dq0.sdk.data.utils.util.`**`save_preprocessed_tr_and_te_datasets`** (*X_train*, *X_test*, *y_train*, *y_test*, *working_folder*)
Save train and test dataset

Args:
X_train: Pandas Dataframe or numpy array X_test: Pandas Dataframe or numpy array y_train: Pandas Series or numpy (also non-dimensional) array y_test: Pandas Series or numpy (also non-dimensional) array working_folder: str with file path

`dq0.sdk.data.utils.util.`**`sparse_scipy_matrix_to_Pandas_df`** (*sp_matr*, *sparse_representation=True*, *columns_names_list=None*)
Convert Scipy matrix to pandas dataframe.

Args:
sp_matr (`scipy.sparse.spmatrix`): The origin matrix sparse_representation (bool): True if the matrix is sparse columns_names_list: list of column names

Returns:
converted pandas dataframe.

`dq0.sdk.data.utils.util.`**`str_to_bool`** (*s*)
Convert string to bool

`dq0.sdk.data.utils.util.`**`string_contains_numeric_value`** (*s*)
Returns true if the string is convertible to float.

`dq0.sdk.data.utils.util.`**`tensorflow_tensor_to_numpy_ndarray`** (**args*)
Convert input tensorflow tensors into numpy.ndarray arrays.

Args:
args: tensorflow tensors

Returns:
np_arrays, a list of numpy.ndarray arrays. Order matters: np_arrays[i] is the conversion of args[i]. If np_arrays contains a single item, the item is returned rather than a list with just one item inside.

1.2.2 Submodules

dq0.sdk.data.data_source_factory module

Data Source Factory.

Helper function to create data source instance based on a given type.

dq0.sdk.data.data_source_factory.**create_from_type** (*type*, **args*)

Returns a matching data source instance based on the given type or None if no data source class for this type was found.

Args:

type: the type of the data source class to create. *args: positional arguments for the specific data source constructor.

Returns:

initialized data source class.

dq0.sdk.data.source module

Data Source abstract base class

The source class serves as the base class for all data sources.

Implementing subclasses have to define at least read

class dq0.sdk.data.source.**Source** (*path=None*, ***kwargs*)

Bases: abc.ABC

Abstract base class for all data connector sources available through the SDK.
Data sources classes provide a read method to read the data into memory or provide a data reader for the underlying source.

Args:

path (str, optional): Path to the data

Attributes:

uuid (str): The universally unique identifier of the data source. name (str): The data source's name type (str): The data source's distinct type (e.g. 'csv') description (str): The data source's description types: json object containing column type description data (pandas.DataFrame): The loaded data read_allowed (bool): True if this source can be read meta_allowed (bool): True if this source provides meta information types_allowed (bool): True if this source provides data type information stats_allowed (bool): True if this source provides statistics sample_allowed (bool): True if there is sample data for this source path (str): Path to the data (filepath, URI) sample_path (str): Path to the data containing sample data. (filepath, URI)

abstract read (***kwargs*)

Read data sources
This function should be used by child classes to read data or return a data handler to read streaming data.

Args:

kwargs: keyword arguments

Returns:

data read from the data source.

to_json ( )

Returns a json representation of this data sources information.

Returns:

data source description as json.

dq0.sdk.data.transform module

Data Transform class.

class dq0.sdk.data.transform.**Transform**

Bases: dq0.sdk.projects.project.Project

Abstract base class for all transformations available through the SDK.

Transform classes provide a execute method to transform source data

Attributes:

model_type (str): type of this model instance. Options: 'keras'. uuid (str): UUID of this model. data_source (dq0.sdk.data.Source): dict of attached data sources.

abstract execute (*dataset=None*)

Execute transformation function

This function can be used by child classes to prepare data that dont need to be repeated for every training run.

1.3 dq0.sdk.errors package

DQ0 SDK Error Package

exception dq0.sdk.errors.**DQ0SDKError**

Bases: Exception

General DQ0 SDK Error.

dq0.sdk.errors.**checkSDKResponse** (*response*)

Check an SDK response for error and raise a DQ0SDKError if neccessary.

Args:

response (dict): SDK response JSON dictionary.

dq0.sdk.errors.**fatal_error** (*error_msg*, *logger=None*, *log_key_string=None*)

Handle fatal errors.

Args:

error_msg: string with error message logger: Logger instance log_key_string (str): secret key to be appended to safe logging

messages (i.e., not harming data privacy). Safe logging messages are shown to DQ0 users without waiting for approval by the data owner / officer.

1.3.1 Submodules

dq0.sdk.errors.errors module

Error handling module

Guidelines for handling errors occurring in SDK and plugins Please use:

logger.warning(), for harmless warning messages. Program execution should not
be stopped;

logger.error(), for an error that DQ0 can recover from. E.g., log an error for

a parameter that has been assigned an infeasible value, assign a default feasible value to the parameter and continue program execution;

dq0.sdk.errors.errors..fatal_error(error_msg) for an error that DQ0 cannot recover
from. Program execution is stopped. Therefore, to handle fatal exception / error:

dq0.sdk.errors.errors.fatal_error(message)

should be preferred to:
logger.fatal(message) return 1. / sys.exit(1)

Optionally, fatal_error() accepts as input a logger instance and a log-key value. See below for details.

exception dq0.sdk.errors.errors.**DQ0SDKError**
Bases: Exception
General DQ0 SDK Error.

dq0.sdk.errors.errors.**checkSDKResponse** (*response*)
Check an SDK response for error and raise a DQ0SDKError if neccessary.

Args:
response (dict): SDK response JSON dictionary.

dq0.sdk.errors.errors.**fatal_error** (*error_msg*, *logger=None*, *log_key_string=None*)
Handle fatal errors.

Args:
error_msg: string with error message logger: Logger instance log_key_string (str): secret key to be appended to safe logging

messages (i.e., not harming data privacy). Safe logging messages are shown to DQ0 users without waiting for approval by the data owner / officer.

1.4 dq0.sdk.estimators package

DQ0 SDK Estimators

This package contains the estimators models a subclassing that follow the sklearn estimator interface

1.4.1 Subpackages

dq0.sdk.estimators.SVM package

Submodules

dq0.sdk.estimators.SVM.sklearn_svm module

Sklearn SVM models.

class dq0.sdk.estimators.SVM.sklearn_svm.**LinearSVC** (*penalty='l2'*, *loss='squared_hinge'*, ***, *dual=True*, *tol=0.0001*, *C=1.0*, *multi_class='ovr'*, *fit_intercept=True*, *intercept_scaling=1*, *class_weight=None*, *verbose=0*, *random_state=None*, *max_iter=1000*, ***kwargs*)
Bases: dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.SVM.sklearn_svm.**LinearSVR** (***, *epsilon=0.0*, *tol=0.0001*, *C=1.0*, *loss='epsilon_insensitive'*, *fit_intercept=True*, *intercept_scaling=1.0*, *dual=True*, *verbose=0*, *random_state=None*, *max_iter=1000*, ***kwargs*)
Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.SVM.sklearn_svm.**NuSVC** (***, *nu=0.5*, *kernel='rbf'*, *degree=3*, *gamma='scale'*, *coef0=0.0*, *shrinking=True*, *probability=False*, *tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=-1*, *decision_function_shape='ovr'*, *break_ties=False*, *random_state=None*, ***kwargs*)
Bases: dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.SVM.sklearn_svm.**NuSVR** (**, nu=0.5, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, tol=0.001, cache_size=200, verbose=False, max_iter=-1, **kwargs*)

Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.SVM.sklearn_svm.**OneClassSVM** (**, kernel='rbf', degree=3, gamma='scale', coef0=0.0, tol=0.001, nu=0.5, shrinking=True, cache_size=200, verbose=False, max_iter=-1, **kwargs*)

Bases: dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.SVM.sklearn_svm.**SVC** (**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None, **kwargs*)

Bases: dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.SVM.sklearn_svm.**SVR** (**, kernel='rbf', degree=3, gamma='scale', coef0=0.0, tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache_size=200, verbose=False, max_iter=-1, **kwargs*)

Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

dq0.sdk.estimators.data_handler package

Submodules

dq0.sdk.estimators.data_handler.base module

Base data handler.

Copyright 2021, Gradient Zero All rights reserved

class dq0.sdk.estimators.data_handler.base.**BasicDataHandler** (*pipeline_steps=None, pipeline_config_path=None, transformers_root_dir='.', log_key_string=''*)

Bases: abc.ABC

Basic Data Handler for all estimators

setup_data (*data_source, **kwargs*)

Empty setup data, just returns the data source

dq0.sdk.estimators.data_handler.csv module

Base data handler.

Copyright 2021, Gradient Zero All rights reserved

class dq0.sdk.estimators.data_handler.csv.**CSVDataHandler** (*pipeline_steps=None, pipeline_config_path=None, transformers_root_dir='.', log_key_string='.'*)

Bases: dq0.sdk.estimators.data_handler.base.BasicDataHandler

Basic CSV Data Handler for all estimators

get_input_dim (*X*)

get_output_dim (*y*)

setup_data (*data_source, train_size=0.66, **kwargs*)

Setup data from CSV file. Using the CSV data source.

dq0.sdk.estimators.data_handler.utils module

Utils for data handler.

Copyright 2021, Gradient Zero All rights reserved

dq0.sdk.estimators.data_handler.utils.**data_handler_factory** (*data_handler_instance*, *pipeline_steps=None*, *pipeline_config_path=None*, *transformers_root_dir='.'*, *log_key_string=''*)

dq0.sdk.estimators.ensemble package

Submodules

dq0.sdk.estimators.ensemble.sklearn_ensemble module

Sklearn ensemble models.

class dq0.sdk.estimators.ensemble.sklearn_ensemble.**AdaBoostClassifier** (*base_estimator=None*, ***, *n_estimators=50*, *learning_rate=1.0*, *algorithm='SAMME.R'*, *random_state=None*, ***kwargs*)

Bases: dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.ensemble.sklearn_ensemble.**AdaBoostRegressor** (*base_estimator=None*, ***, *n_estimators=50*, *learning_rate=1.0*, *loss='linear'*, *random_state=None*, ***kwargs*)

Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.ensemble.sklearn_ensemble.**BaggingClassifier** (*base_estimator=None*, *n_estimators=10*, ***, *max_samples=1.0*, *max_features=1.0*, *bootstrap=True*, *bootstrap_features=False*, *oob_score=False*, *warm_start=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, ***kwargs*)

Bases: dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.ensemble.sklearn_ensemble.**BaggingRegressor** (*base_estimator=None*, *n_estimators=10*, ***, *max_samples=1.0*, *max_features=1.0*, *bootstrap=True*, *bootstrap_features=False*, *oob_score=False*, *warm_start=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, ***kwargs*)

Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.ensemble.sklearn_ensemble.**ExtraTreesClassifier** (*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *bootstrap=False*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*, ***kwargs*)

Bases: dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.ensemble.sklearn_ensemble.**ExtraTreesRegressor** (*n_estimators=100*, ***, *criterion='mse'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *bootstrap=False*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *ccp_alpha=0.0*, *max_samples=None*, ***kwargs*)

Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.ensemble.sklearn_ensemble.**GradientBoostingClassifier** (***, *loss='deviance'*, *learning_rate=0.1*, *n_estimators=100*, *subsample=1.0*, *criterion='friedman_mse'*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_depth=3*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *init=None*, *random_state=None*, *max_features=None*, *verbose=0*, *max_leaf_nodes=None*, *warm_start=False*, *validation_fraction=0.1*, *n_iter_no_change=None*, *tol=0.0001*, *ccp_alpha=0.0*, ***kwargs*)

Bases: dq0.sdk.estimators.base_mixin.ClassifierMixin,

dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.ensemble.sklearn_ensemble.**GradientBoostingRegressor**(**, loss='ls', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None, max_features=None, alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False, validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0, **kwargs*)

Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.ensemble.sklearn_ensemble.**RandomForestClassifier**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None, **kwargs*)

Bases: dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.ensemble.sklearn_ensemble.**RandomForestRegressor**(*n_estimators=100, *, criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None, **kwargs*)

Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

dq0.sdk.estimators.linear_model package

Submodules

dq0.sdk.estimators.linear_model.diffprivlib_lm module

Diffprivlib linear models.

Copyright 2021, Gradient Zero All rights reserved

class dq0.sdk.estimators.linear_model.diffprivlib_lm.**LinearRegressionDP**(*target_epsilon=1, bounds_X=None, bounds_y=None, fit_intercept=True, copy_X=True, accountant=None, **kwargs*)

Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

Diffprivlib linear regression

class dq0.sdk.estimators.linear_model.diffprivlib_lm.**LogisticRegressionDP**(*target_epsilon=1, data_norm=None, tol=0.0001, C=1.0, fit_intercept=True, max_iter=100, verbose=0, warm_start=False, n_jobs=None, accountant=None, **kwargs*)

Bases: dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

Diffprivlib logistic regression

dq0.sdk.estimators.linear_model.sklearn_lm module

Sklearn linear models.

Copyright 2021, Gradient Zero All rights reserved

class dq0.sdk.estimators.linear_model.sklearn_lm.**ElasticNet**(*alpha=1.0, *, l1_ratio=0.5, fit_intercept=True, normalize=False, precompute=False, max_iter=1000, copy_X=True, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic', **kwargs*)

Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.linear_model.sklearn_lm.**Lasso** (*alpha=1.0*, ***, *fit_intercept=True*, *normalize=False*, *precompute=False*, *copy_X=True*, *max_iter=1000*, *tol=0.0001*, *warm_start=False*, *positive=False*, *random_state=None*, *selection='cyclic'*, ***kwargs*)

Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.linear_model.sklearn_lm.**LinearRegression** (***, *fit_intercept=True*, *normalize=False*, *copy_X=True*, *n_jobs=None*, *positive=False*, ***kwargs*)

Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

Sklearn linear regression wrapper

class dq0.sdk.estimators.linear_model.sklearn_lm.**LogisticRegression** (*penalty='l2'*, ***, *dual=False*, *tol=0.0001*, *C=1.0*, *fit_intercept=True*, *intercept_scaling=1*, *class_weight=None*, *random_state=None*, *solver='lbfgs'*, *max_iter=100*, *multi_class='auto'*, *verbose=0*, *warm_start=False*, *n_jobs=None*, *l1_ratio=None*, ***kwargs*)

Bases: dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

Sklearn logistic regression wrapper

class dq0.sdk.estimators.linear_model.sklearn_lm.**Ridge** (*alpha=1.0*, ***, *fit_intercept=True*, *normalize=False*, *copy_X=True*, *max_iter=None*, *tol=0.001*, *solver='auto'*, *random_state=None*, ***kwargs*)

Bases: dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

class dq0.sdk.estimators.linear_model.sklearn_lm.**RidgeClassifier** (*alpha=1.0*, ***, *fit_intercept=True*, *normalize=False*, *copy_X=True*, *max_iter=None*, *tol=0.001*, *class_weight=None*, *solver='auto'*, *random_state=None*, ***kwargs*)

Bases: dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

Sklearn RidgeClassifier

dq0.sdk.estimators.tf package

Submodules

dq0.sdk.estimators.tf.keras_base module

Base tensorflow keras classes for all estimator subclasses

class dq0.sdk.estimators.tf.keras_base.**NN_Classifier** (*data_source=None*, *log_key_string=''*, ***kwargs*)

Bases: dq0.sdk.estimators.tf.keras_base.NeuralNetworkBase

Keras neural network classification models with one hot encoded targets.

predict (*X*)

Return the class as index on the one-hot-encoding format.

predict_proba (*X*)

Returns the confidence scores.

class dq0.sdk.estimators.tf.keras_base.**NN_Regressor** (*data_source=None*, *log_key_string=''*, ***kwargs*)

Bases: dq0.sdk.estimators.tf.keras_base.NeuralNetworkBase

Keras neural network regression models.

predict (*X*)

predict_proba (*X*)

class dq0.sdk.estimators.tf.keras_base.**NeuralNetworkBase** (*data_source=None, log_key_string=''*, ***kwargs*)

Bases: dq0.sdk.estimators.estimator.Estimator

Base TF Network mixin.

fit (*X*, *y*, ***kwargs*)

Model fit method

setup_data (*data_handler_instance='CSV'*, *pipeline_steps=None*, *pipeline_config_path=None*, ***kwargs*)

Keras NN specific setup data. To get the input and output dimensions from the data handler.

dq0.sdk.estimators.tf.keras_base.**layer_factory** (*layers*, *n_layers*, ***kwargs*)

Helper function to create the layers given some parameters.

dq0.sdk.estimators.tf.keras_dense_classifier module

Keras dense neural network for classification with different target encoding.

class dq0.sdk.estimators.tf.keras_dense_classifier.**Keras_Dense_Classifier_Binary** (*optimizer='Adam'*, *loss=<tensorflow.python.keras.losses.BinaryCrossentropy object>*, *metrics=['accuracy', 'mae']*, *batch_size=250*, *epochs=2*, ***kwargs*)

Bases: dq0.sdk.estimators.tf.keras_base.NN_Classifier, dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

Dense Classifier with binary labels.

setup_model (*input_shape=None*, *n_layers=[10, 10]*, *optimizer='Adam'*, *loss=<tensorflow.python.keras.losses.BinaryCrossentropy object>*, *metrics=['accuracy', 'mae']*, *batch_size=250*, *epochs=2*, ***kwargs*)

Args: n_layers: list of int, for every element a layer with the number of units given in the list

class dq0.sdk.estimators.tf.keras_dense_classifier.**Keras_Dense_Classifier_Integer** (*optimizer='Adam'*, *loss=<tensorflow.python.keras.losses.SparseCategoricalCrossentropy object>*, *metrics=['accuracy', 'mae']*, *batch_size=250*, *epochs=2*, *n_layers=[10, 10]*, ***kwargs*)

Bases: dq0.sdk.estimators.tf.keras_base.NN_Classifier, dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

Dense Classifier with integer encoding labels.

setup_model (*input_shape=None*, *n_classes=None*, *n_layers=[10, 10]*, *optimizer='Adam'*, *loss=<tensorflow.python.keras.losses.SparseCategoricalCrossentropy object>*, *metrics=['accuracy', 'mae']*, *batch_size=250*, *epochs=2*, ***kwargs*)

Args: n_layers: list of int, for every element a layer with the number of units given in the list

class dq0.sdk.estimators.tf.keras_dense_classifier.**Keras_Dense_Classifier_OHE** (*optimizer='Adam'*, *loss=<tensorflow.python.keras.losses.CategoricalCrossentropy object>*, *metrics=['accuracy', 'mae']*, *batch_size=250*, *epochs=2*, *n_layers=[10, 10]*, ***kwargs*)

Bases: dq0.sdk.estimators.tf.keras_base.NN_Classifier, dq0.sdk.estimators.base_mixin.ClassifierMixin, dq0.sdk.estimators.estimator.Estimator

Keras sequential dense estimator for classification with OHE targets.

setup_model (*input_shape=None, n_classes=None, n_layers=[10, 10], optimizer='Adam', loss=<tensorflow.python.keras.losses.CategoricalCrossentropy object>, metrics=['accuracy', 'mae'], batch_size=250, epochs=2, **kwargs*)

Args: n_layers: list of int, for every element a layer with the number of units given in the list

dq0.sdk.estimators.tf.keras_dense_regressor module

Keras dense neural networks for regression targets.

class dq0.sdk.estimators.tf.keras_dense_regressor.**Keras_Dense_Regressor** (*optimizer='Adam', loss=<tensorflow.python.keras.losses.MeanAbsoluteError object>, metrics=['mae'], batch_size=250, epochs=2, n_layers=[10, 10], **kwargs*)

Bases: dq0.sdk.estimators.tf.keras_base.NN_Regressor, dq0.sdk.estimators.base_mixin.RegressorMixin, dq0.sdk.estimators.estimator.Estimator

setup_model (*input_shape=None, n_layers=[10, 10], optimizer='Adam', loss=<tensorflow.python.keras.losses.MeanAbsoluteError object>, metrics=['mae'], batch_size=250, epochs=2, **kwargs*)

Setup model function.

1.4.2 Submodules

dq0.sdk.estimators.base_mixin module

Base mixin classes for all estimator subclasses

class dq0.sdk.estimators.base_mixin.**BiclusterMixin**

Bases: object

Mixin class for all bicluster estimators in DQ0.

class dq0.sdk.estimators.base_mixin.**ClassifierMixin**

Bases: object

Mixin class for all classifier estimators in DQ0

predict (*X*)

Predict classes for given dataset.

predict_proba (*X*)

Get class probability for the given data X

score (*X, y, sample_weight=None*)

Return the mean accuracy on the given test data and labels.

class dq0.sdk.estimators.base_mixin.**ClusterMixin**

Bases: object

Mixin class for all cluster estimators in DQ0.

fit_predict (*X, y=None*)

class dq0.sdk.estimators.base_mixin.**RegressorMixin**

Bases: object

Mixin class for all regressor estimators in DQ0

predict (*X*)

Predict for given dataset.

predict_proba (*X*)
Predict classes for given dataset.

score (*X*, *y*, *sample_weight=None*)
Return the coefficient of determination `R^2` of the prediction.

class `dq0.sdk.estimators.base_mixin.`**TransformerMixin**
Bases: `object`
Mininclass for all transformers in DQ0

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

dq0.sdk.estimators.estimator module

Estimator abstract base class

The Model class serves as the base class for all models.

Implementing subclasses have to define setup_data and setup_model functions.

class `dq0.sdk.estimators.estimator.`**Estimator** (*data_source=None*, *log_key_string=''*, ***kwargs*)
Bases: `dq0.sdk.projects.project.Project`
Abstract base class

fit (*X*, *y=None*, ***kwargs*)
Model fit method

setup_data (*data_handler_instance='CSV'*, *pipeline_steps=None*, *pipeline_config_path=None*, *transformers_root_dir='.'*, ***kwargs*)
Setup data function using a data_handler None of the estimators handle data by themselfs. They make use of predefined data_handler. It is selected by the 'data_handler_instance' attribute.

Params:
data_handler_instane: string: as defined in dq0.sdk.estimators.data_handler_utils; default is CSV **kwargs: open kwargs

setup_model (***kwargs*)
Setup model function.

1.5 dq0.sdk.examples package

DQ0 SDK Examples.

1.5.1 Subpackages

dq0.sdk.examples.census package

Subpackages

dq0.sdk.examples.census.bayesian package

Subpackages

dq0.sdk.examples.census.bayesian.model namespace

Submodules

dq0.sdk.examples.census.bayesian.model.user_model module

Gaussian Naive Bayesian Model example for the adult census data set.

class dq0.sdk.examples.census.bayesian.model.user_model.**UserModel**

Bases: dq0.sdk.models.bayes.naive_bayesian_model.NaiveBayesianModel

Naive Bayesian classifier for the "Adult Census Income" dataset

SDK users instantiate this class to create and train the model.

preprocess ( )

Preprocess the data

Preprocess the data set. The input data is read from the attached source.

At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.

For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.

Use *self.data_source.read()* to read the attached data.

Returns:

preprocessed data

setup_data (***kwargs*)

Setup data function

This function can be used to prepare data or perform other tasks for the training run.

setup_model (***kwargs*)

Setup model function

Define the model here.

Submodules

dq0.sdk.examples.census.bayesian.run_demo module

Adult dataset example.

Run script to test the bayesian census model locally.

dq0.sdk.examples.census.preprocessed package

Subpackages

dq0.sdk.examples.census.preprocessed.model namespace

Submodules

dq0.sdk.examples.census.preprocessed.model.user_model module

Adult dataset example.

Neural network model definition

Example:

```
>>> ./dq0 project create --name demo
>>> cd demo
>>> copy user_model.py to demo/model/
>>> ../dq0 data list
>>> ../dq0 model attach --id <dataset id>
>>> ../dq0 project deploy
>>> ../dq0 model train
>>> ../dq0 model state
>>> ../dq0 model predict --input-path </path/to/numpy.npy>
>>> ../dq0 model state
```


class dq0.sdk.examples.census.preprocessed.model.user_model.**UserModel**

Bases:
dq0.sdk.models.tf.neural_network_classification.NeuralNetworkClassification

Derived from dq0.sdk.models.tf.NeuralNetwork class
Model classes provide a setup method for data and model definitions.

setup_data (***kwargs*)
Setup data function
This function can be used to prepare data or perform other tasks for the training run.
At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.
For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.
Use *self.data_source.read()* to read the attached data.

setup_model (***kwargs*)
Setup model function
Define the model here.

Submodules

dq0.sdk.examples.census.preprocessed.run_demo module

Adult dataset example.

Run script to test the exeuction locally.

dq0.sdk.examples.census.raw package

Subpackages

dq0.sdk.examples.census.raw.model namespace

Submodules

dq0.sdk.examples.census.raw.model.user_model module

Adult dataset example.

Neural network model definition

Example:

```
>>> ./dq0 project create --name demo
>>> cd demo
>>> copy user_model.py to demo/model/
>>> ../dq0 data list
>>> ../dq0 model attach --id <dataset id>
>>> ../dq0 project deploy
>>> ../dq0 model train
>>> ../dq0 model state
>>> ../dq0 model predict --input-path </path/to/numpy.npy>
>>> ../dq0 model state
```


class dq0.sdk.examples.census.raw.model.user_model.**UserModel**
Bases:
dq0.sdk.models.tf.neural_network_classification.NeuralNetworkClassification
Derived from dq0.sdk.models.tf.NeuralNetwork class
Model classes provide a setup method for data and model definitions.

preprocess ( )
Preprocess the data
Preprocess the data set. The input data is read from the attached source.
At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.
For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.

Use *self.data_source.read()* to read the attached data.

Returns:
preprocessed data

setup_data (***kwargs*)
Setup data function
This function can be used to prepare data or perform other tasks for the training run.
At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.
For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.
Use *self.data_source.read()* to read the attached data.

setup_model (***kwargs*)
Setup model function
Define the model here.

Submodules

dq0.sdk.examples.census.raw.run_demo module

Adult dataset example.

Run script to test the execution locally.

dq0.sdk.examples.census.raw.run_demo_probas_calibration module

Submodules

dq0.sdk.examples.census.eda module

Adult dataset example.

Run script to test the execution locally.

dq0.sdk.examples.census.eda.**eda** (*dataset*, *output_folder*)
Perform exploratory data analysis over the input dataset

Args:
dataset (pandas.DataFrame): dataset to be investigated

dq0.sdk.examples.census.eda.**preprocess_dataset** (*data_source*)

dq0.sdk.examples.census.eda.**visualize_filtered_data** (*series*, *conditions*, *output_folder*)

Args:
series (pandas.Series): data conditions (tuple): (lb, ub) output_folder (str): path to output folder

dq0.sdk.examples.cifar package

Subpackages

dq0.sdk.examples.cifar.model namespace

Submodules

dq0.sdk.examples.cifar.model.user_model module

Neural Network model for CIFAR-10 image dataset.

Use this class to train a classifier on CIFAR-10 image data.

class dq0.sdk.examples.cifar.model.user_model.**UserModel**
Bases:
dq0.sdk.models.tf.neural_network_classification.NeuralNetworkClassification

Convolutional Neural Network model implementation for Cifar-10 image data.
SDK users instantiate this class to create and train Keras models or subclass this class to define custom neural networks.

Attributes:

model_type (`str`): type of this model instance. Options: 'keras'. label_encoder (`sklearn.preprocessing.LabelEncoder`): sklearn class label encoder.

`setup_data` (***kwargs*)

Setup data function
This function can be used to prepare data or perform other tasks for the training run.
At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.
For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.
Use *self.data_source.read()* to read the attached data.

`setup_model` (***kwargs*)

Setup model function
Define the CNN model.

dq0.sdk.examples.cifar.model.user_model_old module

Neural Network model for CIFAR-10 image dataset.

Use this class to train a classifier on CIFAR-10 image data.

class `dq0.sdk.examples.cifar.model.user_model_old.`**`UserModel`**

Bases:
`dq0.sdk.models.tf.neural_network_classification.NeuralNetworkClassification`
Convolutional Neural Network model implementation for Cifar-10 image data.
SDK users instantiate this class to create and train Keras models or subclass this class to define custom neural networks.

Attributes:

model_type (`str`): type of this model instance. Options: 'keras'. label_encoder (`sklearn.preprocessing.LabelEncoder`): sklearn class label encoder.

`setup_data` ( )

Setup data function
This function can be used to prepare data or perform other tasks for the training run.
At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.
For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.
Use *self.data_source.read()* to read the attached data.

`setup_model` ( )

Setup model function
Define the CNN model.

Submodules

dq0.sdk.examples.cifar.run_demo module

CIFAR-10 example.

Run script to test the execution locally.

dq0.sdk.examples.har namespace

Subpackages

dq0.sdk.examples.har.model namespace

Submodules

dq0.sdk.examples.har.model.user_model module

Human Activity Recognition dataset example.

http://groupware.les.inf.puc-rio.br/har

Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz6bPytcguP

Neural network model definition

Example:

```
>>> ./dq0 project create --name demo
>>> cd demo
>>> copy user_model.py to demo/model/
>>> ../dq0 data list
>>> ../dq0 model attach --id <dataset id>
>>> ../dq0 project deploy
>>> ../dq0 model train
>>> ../dq0 model state
>>> ../dq0 model predict --input-path </path/to/numpy.npy>
>>> ../dq0 model state
```


class dq0.sdk.examples.har.model.user_model.**UserModel**

Bases:
dq0.sdk.models.tf.neural_network_classification.NeuralNetworkClassification

Derived from dq0.sdk.models.tf.NeuralNetwork class

Model classes provide a setup method for data and model definitions.

preprocess ( )

Preprocess the data

Preprocess the data set. The input data is read from the attached source.

At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.

For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.

Use *self.data_source.read()* to read the attached data.

Returns:

preprocessed data

setup_data (***kwargs*)

Setup data function

This function can be used to prepare data or perform other tasks for the training run.

At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.

For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.

Use *self.data_source.read()* to read the attached data.

setup_model (***kwargs*)
Setup model function
Define the model here.

Submodules

dq0.sdk.examples.har.run_demo module

Human Activity Recognition dataset example.

Run script to test the execution locally.

dq0.sdk.examples.medical_insurance package

Subpackages

dq0.sdk.examples.medical_insurance.model namespace

Submodules

dq0.sdk.examples.medical_insurance.model.user_model module

Neural network Model for the medical insurance dataset

https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/insurance.csv

1338 examples of beneficiaries in the insurance plan. Task: predict total medical expenses charged to the plan based on six attributes of the beneficiary:

age sex: gender, female / male bmi: body mass index (kg / m ^ 2), ratio of person's weight in kilograms

and height in meters squared. Ideally from 18.5 to 24.9

children: number of children covered by health insurance smoker: yes / no region: beneficiary's residential area in the US: northeast, southeast, southwest, northwest.

class dq0.sdk.examples.medical_insurance.model.user_model.**UserModel**
Bases:
dq0.sdk.models.tf.neural_network_regression.NeuralNetworkRegression
Derived from dq0.sdk.models.tf.NeuralNetworkRegression class
Model classes provide a setup method for data and model definitions.

setup_data (***kwargs*)
Setup data function
This function can be used to prepare data or perform other tasks for the training run.

setup_model (***kwargs*)
Setup model function
Define the model here.

Submodules

dq0.sdk.examples.medical_insurance.run_demo module

Adult dataset example.

Run script to test the execution locally.

dq0.sdk.examples.newsgroups package

Subpackages

dq0.sdk.examples.newsgroups.bayesian package

Subpackages

dq0.sdk.examples.newsgroups.bayesian.model namespace

Submodules

dq0.sdk.examples.newsgroups.bayesian.model.user_model module

Multinomial Naive Bayesian Model example for the 20Newsgroups dataset.

Copyright 2020, Gradient Zero All rights reserved

class dq0.sdk.examples.newsgroups.bayesian.model.user_model.**UserModel**

Bases: dq0.sdk.models.bayes.naive_bayesian_model.NaiveBayesianModel

Multinomial Naive Bayesian classifier for the "20 Newsgroups" dataset

SDK users instantiate this class to create and train the model.

setup_data (***kwargs*)

Setup data function

This function can be used to prepare data or perform other tasks for the training run.

At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.

For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.

Use *self.data_source.read()* to read the attached data.

setup_model (***kwargs*)

Setup model.

Define the model here.

dq0.sdk.examples.newsgroups.bayesian.model.user_model_old_needs_SOURCE module

Multinomial Naive Bayesian Model example for the 20Newsgroups dataset.

Copyright 2020, Gradient Zero All rights reserved

class dq0.sdk.examples.newsgroups.bayesian.model.user_model_old_needs_SOURCE.**UserModel**

Bases: dq0.sdk.models.bayes.naive_bayesian_model.NaiveBayesianModel

Multinomial Naive Bayesian classifier for the "20 Newsgroups" dataset

SDK users instantiate this class to create and train the model.

setup_data ( )

Setup data function

This function can be used to prepare data or perform other tasks for the training run.

At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.

For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.

Use *self.data_source.read()* to read the attached data.

setup_model ( )

Setup model.

Define the model here.

Submodules

dq0.sdk.examples.newsgroups.bayesian.run_demo module

20 Newsgroups dataset example.

Run script to test the execution locally.

dq0.sdk.examples.newsgroups.network package

Subpackages

dq0.sdk.examples.newsgroups.network.model namespace

Submodules

dq0.sdk.examples.newsgroups.network.model.user_model module

Convolutional Neural Network model implementation for "20 Newsgroups"

class dq0.sdk.examples.newsgroups.network.model.user_model.**UserModel**

Bases:
dq0.sdk.models.tf.neural_network_classification.NeuralNetworkClassification

Neural Network model implementation for "20 Newsgroups"

SDK users instantiate this class to create and train Keras models or subclass this class to define custom neural networks.

setup_data (***kwargs*)

Setup data function

This function can be used to prepare data or perform other tasks for the training run.

At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.

For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.

Use *self.data_source.read()* to read the attached data.

setup_model (***kwargs*)

Setup model function

Implementing child classes can use this method to define the model.

dq0.sdk.examples.newsgroups.network.model.user_model_from_preprocessed_data module

Convolutional Neural Network model implementation for "20 Newsgroups"

class
dq0.sdk.examples.newsgroups.network.model.user_model_from_preprocessed_data.**UserModel**

Bases:
dq0.sdk.models.tf.neural_network_classification.NeuralNetworkClassification

Neural Network model implementation for "20 Newsgroups"

SDK users instantiate this class to create and train Keras models or subclass this class to define custom neural networks.

setup_data ( )

Setup data function

This function can be used to prepare data or perform other tasks for the training run.

At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.

For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.

Use *self.data_source.read()* to read the attached data.

setup_model ( )

Setup model function

Implementing child classes can use this method to define the model.

dq0.sdk.examples.newsgroups.network.model.user_model_old_needs_SOURCE module

Convolutional Neural Network model implementation for "20 Newsgroups"

Copyright 2020, Gradient Zero All rights reserved

class
dq0.sdk.examples.newsgroups.network.model.user_model_old_needs_SOURCE.**UserModel**
Bases:
dq0.sdk.models.tf.neural_network_classification.NeuralNetworkClassification
Neural Network model implementation for "20 Newsgroups"
SDK users instantiate this class to create and train Keras models or subclass this class to define custom neural networks.

setup_data ( )
Setup data function
This function can be used to prepare data or perform other tasks for the training run.
At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.
For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.
Use *self.data_source.read()* to read the attached data.

setup_model ( )
Setup model function
Implementing child classes can use this method to define the model.

Submodules

dq0.sdk.examples.newsgroups.network.run_demo module

20 Newsgroups dataset example.

Run script to test the execution locally.

Copyright 2020, Gradient Zero All rights reserved

dq0.sdk.examples.patient package

Subpackages

dq0.sdk.examples.patient.model namespace

Submodules

dq0.sdk.examples.patient.model.user_model module

Neural network Model class for patient dataset:

https://synthea.mitre.org/downloads

Copyright 2020, Gradient Zero

class dq0.sdk.examples.patient.model.user_model.**UserModel**
Bases:
dq0.sdk.models.tf.neural_network_regression.NeuralNetworkRegression
Derived from dq0.sdk.models.tf.NeuralNetworkRegression class
Model classes provide a setup method for data and model definitions.

setup_data (***kwargs*)
Setup data function
This function can be used to prepare data or perform other tasks for the training run.

setup_model (***kwargs*)
Setup model function
Define the model here.

dq0.sdk.examples.pneumonia namespace

Submodules

dq0.sdk.examples.pneumonia.data_preparation module

Create feature vector for the Pneumonia image dataset. In a real setting this could not be done outside of DQ0. This should later be converted into a transform step in the project.

dq0.sdk.examples.pneumonia.data_preparation.**create_dataset** (*data_path*, *DOWNSAMPLE=True*)

dq0.sdk.examples.pneumonia.data_preparation.**create_df** (*X_train_feat_vec*, *X_test_feat_vec*, *idx_train*, *idx_test*, *idx_train_normal*, *idx_train_pneu*, *idx_test_normal*, *idx_test_pneu*)

dq0.sdk.examples.pneumonia.data_preparation.**create_feature_vector** (*X*, *model_url='https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/4'*)

Uses a pretrained neural network form tf_hub to transform the given dataset X into a feature vec.

dq0.sdk.examples.pneumonia.data_preparation.**load_img_in_dir** (*data_folder*, *resize_shape=(224, 224)*)

The images are reshaped and flattend to be used saved a CSV file

dq0.sdk.examples.pneumonia.data_preparation.**train_test_split_reshape** (*df*, *img_shape=(224, 224)*)

dq0.sdk.examples.pneumonia.keras_model module

dq0.sdk.examples.pneumonia.my_model module

Neural Network model for Pneumonia image dataset.

class dq0.sdk.examples.pneumonia.my_model.**UserModel**

Bases:
dq0.sdk.models.tf.neural_network_classification.NeuralNetworkClassification

CNN with pre-training

evaluate (*test_data=True*, *verbose=0*)

Model evaluate implementation.

Args:

test_data (bool): False to use train data instead of test

Default is True.

verbose (int): Verbose level, Default is 0

setup_data (***kwargs*)

Setup data function

This function can be used by child classes to prepare data or perform other tasks that dont need to be repeated for every training run.

setup_model (***kwargs*)

Set up model function

dq0.sdk.examples.pneumonia.my_model_int_encoding module

Neural Network model for Pneumonia image dataset.

class dq0.sdk.examples.pneumonia.my_model_int_encoding.**UserModel**

Bases:
dq0.sdk.models.tf.neural_network_classification.NeuralNetworkClassification

CNN with pre-training

evaluate (*test_data=True, verbose=0*)
Model evaluate implementation.

Args:

test_data (bool): False to use train data instead of test
Default is True.
verbose (int): Verbose level, Default is 0

setup_data (***kwargs*)
Set up data function

setup_model (***kwargs*)
Set up model function

1.5.2 Submodules

dq0.sdk.examples.wrapper_for_sdk_demos module

1.6 dq0.sdk.models package

DQ0 SDK Models Package

This package contains the model abstract classes and implementing subclasses.

class dq0.sdk.models.**Model**
Bases: dq0.sdk.projects.project.Project
Abstract base class for all models available through the SDK.
Model classes provide a setup method as well as the fit and predict ML model functions.

Attributes:
model_type (str): type of this model instance. Options: 'keras'. uuid (str): UUID of this model. data_source (dq0.sdk.data.Source): dict of attached data sources.

fit ( )
Train model on a dataset passed as input.

abstract get_clone ( )
Generates a new model with the same parameters, if they are not fit on the training data.
Generates a deep copy of the model without actually copying any attached dataset. It yields a new model with the same parameters that has not been fit on any data. Parameters fit to the training data like, e.g., model weights, are re-initialized in the clone.

Returns:
deep copy of model

abstract load (*path*)
Loads the model.
Implementing child classes should use this function to load the model from local storage.

Args:
path (str): The model path

preprocess ( )
Preprocess the data
Preprocess the data set. The input data is read from the attached source.
At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.

For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.
Use *self.data_source.read()* to read the attached data.

Returns:
preprocessed data

abstract save (*path*)
Saves the model.
Implementing child classes should use this function to save the model in binary format on local storage.

Args:
path (`str`): The model path

abstract setup_data (***kwargs*)
Setup data function
This function can be used by child classes to prepare data or perform other tasks that dont need to be repeated for every training run.

abstract setup_model (***kwargs*)
Setup model function
Implementing child classes can use this method to define the model.

abstract to_string ( )
Print model type.
Implementing child classes should use this function to print the model_type.

1.6.1 Subpackages

dq0.sdk.models.bayes package

DQ0 SDK Models Bayes Package

This package contains bayesian models subclassing the abstract model base class.

class `dq0.sdk.models.bayes.`**NaiveBayesianModel**
Bases: `dq0.sdk.models.model.Model`
Naive Bayesian classifier implementation.
Simple model representing a Bayesian classifier.

evaluate (*test_data=True*)
Model evaluate implementation.

Args:

test_data (bool): False to use train data instead of test
Default is True.

fit ( )
Model fit function learning a model from training data

get_clone (*trained=False*)
Generates a new model with the same parameters, if they are not fit on the training data.
Generates a deep copy of the model without actually copying any attached dataset. It yields a new model with the same parameters that has not been fit on any data. Parameters fit to the training data like, e.g., model weights, are re-initialized in the clone.

Args:

trained: if *True*, maintains current state including trained model
weights, etc. Otherwise, returns an unfitted model with the same initialization

params.

Returns:
deep copy of model

load (*path*)
Loads the model.
Load the model from local storage.

Args:
path (str): The model path

save (*path*)
Saves the model.
Save the model in binary format on local storage.

Args:
path (str): The model path

to_string ( )
Print model type.
Implementing child classes should use this function to print the model_type.

Submodules

dq0.sdk.models.bayes.naive_bayesian_model module

Naive Bayesian Model class

class dq0.sdk.models.bayes.naive_bayesian_model.**NaiveBayesianModel**
Bases: dq0.sdk.models.model.Model
Naive Bayesian classifier implementation.
Simple model representing a Bayesian classifier.

evaluate (*test_data=True*)
Model evaluate implementation.

Args:

test_data (bool): False to use train data instead of test
Default is True.

fit ( )
Model fit function learning a model from training data

get_clone (*trained=False*)
Generates a new model with the same parameters, if they are not fit on the training data.
Generates a deep copy of the model without actually copying any attached dataset. It yields a new model with the same parameters that has not been fit on any data. Parameters fit to the training data like, e.g., model weights, are re-initialized in the clone.

Args:

trained: if *True*, maintains current state including trained model
weights, etc. Otherwise, returns an unfitted model with the same initialization params.

Returns:
deep copy of model

load (*path*)
Loads the model.
Load the model from local storage.

Args:
path (str): The model path

save (*path*)
Saves the model.
Save the model in binary format on local storage.

Args:
path (str): The model path

to_string ( )
Print model type.
Implementing child classes should use this function to print the model_type.

dq0.sdk.models.tf package

DQ0 SDK Models Tensorflow Package

This package contains the tensorflow models subclassing the abstract model base class.

class dq0.sdk.models.tf.**NeuralNetwork**
Bases: dq0.sdk.models.model.Model
Neural Network model implementation.
SDK users can use this class to create and train Keras models or subclass this class to define custom neural networks.

Note:
fit, predict, and evaluate functions will be overriden at runtime when executed inside the DQ0 quarantine instance.

evaluate (*test_data=True*, *verbose=0*)
Model evaluate implementation.

Args:
test_data (bool): False to use train data instead of test
Default is True.
verbose (int): Verbose level, Default is 0

fit (*verbose=0*)
Model fit function learning a model from training data

get_clone (*trained=False*)
Generates a new model with the same parameters, if they are not fit on the training data.
Generates a deep copy of the model without actually copying any attached dataset. It yields a new model with the same parameters that has not been fit on any data. Parameters fit to the training data like, e.g., model weights, are re-initialized in the clone.

Args:
trained: if *True*, maintains current state including trained model
weights, etc. Otherwise, returns an unfitted model with the same initialization params.

Returns:
deep copy of model

load (*path*)
Loads the model.
Load the model from local storage.

Args:
path (str): The model path

predict (*x*)
Model predict function.
Model scoring.

Returns:
yhat: numerical matrix containing the predicted responses.

save (*path*)
Saves the model.
Save the model in binary format on local storage.

Args:
path (`str`): The model path

class `dq0.sdk.models.tf.`**NeuralNetworkClassification**
Bases: `dq0.sdk.models.tf.neural_network.NeuralNetwork`
Neural Network multi class classification model

to_string ( )
Print model type.
Implementing child classes should use this function to print the model_type.

class `dq0.sdk.models.tf.`**NeuralNetworkMultiClassClassification**
Bases: `dq0.sdk.models.tf.neural_network.NeuralNetwork`
Neural Network multi class classification model.

to_string ( )
Print model type.
Implementing child classes should use this function to print the model_type.

class `dq0.sdk.models.tf.`**NeuralNetworkRegression**
Bases: `dq0.sdk.models.tf.neural_network.NeuralNetwork`
Neural Network model implementation.

to_string ( )
Print model type.
Implementing child classes should use this function to print the model_type.

class `dq0.sdk.models.tf.`**NeuralNetworkYaml** (*yaml_path=None*)
Bases: `dq0.sdk.models.model.Model`
Neural Network defined by Yaml file.

Note:
fit, predict, and evaluate functions will be overriden at runtime when executed inside the DQ0 quarantine instance.

Args:
yaml_path (`str`): path to the model definition file.

Attributes:
model_type (`str`): type of this model instance. Options: 'keras'. yaml_config (`dq0.sdk.utils.YamlConfig`): yaml config reader yaml_dict (`dict`): Parsed yaml config dictionary. model (`tf.keras.Sequential`): the actual keras model. custom_objects (`dict`): A dictionary of additional model objects.

load (*path='model'*)
Loads the model.
Load the model from local storage.

Args:
path (`str`): The model path

predict (*x*)
Model predict function.
Model scoring.

Args:

x: Input data. It could be:
A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs). A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs). A dict mapping input names to the corresponding array/tensors, if the model has named inputs. A tf.data dataset. Should return a tuple of either (inputs, targets) or (inputs, targets, sample_weights). A generator or tf.keras.utils.Sequence returning (inputs, targets) or (inputs, targets, sample weights).

A more detailed description of unpacking behavior for iterator types (Dataset, generator, Sequence) is given below.

Returns:
yhat: numerical matrix containing the predicted responses.

run_all (*augment=False*)
Runs experiment
Does all the setup data, model, fit and evaluate

save (*path='model'*)
Saves the model.
Save the model in binary format on local storage.

Args:
path (`str`): The model path

setup_data (*augment=False*)
Setup data function
This function can be used by child classes to prepare data or perform other tasks that dont need to be repeated for every training run.

Args:
augment (bool): applies image augmenttion to training data

setup_model ( )
Setup model from yaml MODEL
This function converts the yaml MODEL:GRAPH: config to an instance of tf.keras.Sequential

to_string ( )
Print model type.
Implementing child classes should use this function to print the model_type.

class `dq0.sdk.models.tf.`**TFHub** (*tf_hub_url=None*)
Bases: `dq0.sdk.models.tf.neural_network_yaml.NeuralNetworkYaml`
Tensorflow Hub Network Model.
Uses NeuralNetworkYaml to read a TF Hub Yaml config to define the model.

setup_data (*augment=False*)
Setup Predefined data

args:
task (`str`): string specifying the task, i.e, im_clf or text_clf augment (bool): augment training data

returns:

X_train: true x or generator containing y y_train: true y or if x generator then None X_test: true x or generator containing y y_test: true y or if x generator then None fit_kwargs: steps_per_epoch and validation data if used

Submodules

dq0.sdk.models.tf.neural_network module

Neural Network Model

Basic tensorflow neural network implementation using Keras.

This can be used as a base class for UserModel definitions.

Example:

```
>>> import dq0.sdk.models.tf.NeuralNetworkClassification
>>>
>>> class MyAwsomeModel(NeuralNetwork):
>>>     def __init__(self):
>>>         super().__init__()
>>>
>>>     def setup_data(self):
>>>         # do something
>>>         pass
>>>
>>>     def setup_model(self):
>>>         # freely deinfe the tf / keras model
>>>         pass
>>>
>>> if __name__ == "__main__":
>>>     myModel = MyAwsomeModel()
>>>     myModel.setup_data()
>>>     myModel.setup_model()
>>>     myModel.fit()
>>>     myModel.save()
```


class dq0.sdk.models.tf.neural_network.**NeuralNetwork**

Bases: dq0.sdk.models.model.Model

Neural Network model implementation.

SDK users can use this class to create and train Keras models or subclass this class to define custom neural networks.

Note:

fit, predict, and evaluate functions will be overriden at runtime when executed inside the DQ0 quarantine instance.

evaluate (*test_data=True*, *verbose=0*)

Model evaluate implementation.

Args:

test_data (bool): False to use train data instead of test
Default is True.

verbose (int): Verbose level, Default is 0

fit (*verbose=0*)

Model fit function learning a model from training data

get_clone (*trained=False*)

Generates a new model with the same parameters, if they are not fit on the training data.

Generates a deep copy of the model without actually copying any attached dataset. It yields a new model with the same parameters that has not been fit on any data. Parameters fit to

the training data like, e.g., model weights, are re-initialized in the clone.

Args:

trained: if *True*, maintains current state including trained model
weights, etc. Otherwise, returns an unfitted model with the same initialization params.

Returns:
deep copy of model

load (*path*)
Loads the model.
Load the model from local storage.

Args:
path (`str`): The model path

predict (*x*)
Model predict function.
Model scoring.

Returns:
yhat: numerical matrix containing the predicted responses.

save (*path*)
Saves the model.
Save the model in binary format on local storage.

Args:
path (`str`): The model path

dq0.sdk.models.tf.neural_network.**fix_limitation_of_Keras_fit_and_predict_functions** (*X*, *y*, *batch_size*)
Fix limitation of Keras "fit", "predict" and "evaluate" functions.
Limitation of Keras "fit" function: size of training dataset (i.e., number of training samples) must be divisible by the minibatch size ("batch_size" parameter).
This function removes above limitation by making training robust for any number of minibatches.
The same limitation holds for Keras "evaluate" and "predict" functions, too. In the case of "evaluate" and "predict", if all the data to be predicted do not fit in the CPU/GPU RAM at the same time, predictions are done in batches. Args:

X: data matrix y: learning signal batch_size: batch size set in user model

Returns:
X, y

dq0.sdk.models.tf.neural_network_classification module

Neural Network multi class classification model

class
dq0.sdk.models.tf.neural_network_classification.**NeuralNetworkClassification**
Bases: dq0.sdk.models.tf.neural_network.NeuralNetwork
Neural Network multi class classification model

to_string ( )
Print model type.
Implementing child classes should use this function to print the model_type.

dq0.sdk.models.tf.neural_network_multiclass_classification module

Neural Network multi class classification model

class dq0.sdk.models.tf.neural_network_multiclass_classification.**NeuralNetworkMultiClassClass**

Bases: dq0.sdk.models.tf.neural_network.NeuralNetwork

Neural Network multi class classification model.

to_string ( )

Print model type.

Implementing child classes should use this function to print the model_type.

dq0.sdk.models.tf.neural_network_regression module

Neural Network multi class classification model

class dq0.sdk.models.tf.neural_network_regression.**NeuralNetworkRegression**

Bases: dq0.sdk.models.tf.neural_network.NeuralNetwork

Neural Network model implementation.

to_string ( )

Print model type.

Implementing child classes should use this function to print the model_type.

dq0.sdk.models.tf.neural_network_yaml module

Neural Network Model For Image Classification From Yaml

Basic tensorflow neural network implementation using Keras for image classification using a yaml config

class dq0.sdk.models.tf.neural_network_yaml.**NeuralNetworkYaml** (*yaml_path=None*)

Bases: dq0.sdk.models.model.Model

Neural Network defined by Yaml file.

Note:

fit, predict, and evaluate functions will be overriden at runtime when executed inside the DQ0 quarantine instance.

Args:

yaml_path (str): path to the model definition file.

Attributes:

model_type (str): type of this model instance. Options: 'keras'. yaml_config (dq0.sdk.utils.YamlConfig): yaml config reader yaml_dict (dict): Parsed yaml config dictionary. model (tf.keras.Sequential): the actual keras model. custom_objects (dict): A dictionary of additional model objects.

load (*path='model'*)

Loads the model.

Load the model from local storage.

Args:

path (str): The model path

predict (*x*)

Model predict function.

Model scoring.

Args:

x: Input data. It could be:

A Numpy array (or array-like), or a list of arrays (in case the model has multiple

inputs). A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs). A dict mapping input names to the corresponding array/tensors, if the model has named inputs. A tf.data dataset. Should return a tuple of either (inputs, targets) or (inputs, targets, sample_weights). A generator or tf.keras.utils.Sequence returning (inputs, targets) or (inputs, targets, sample weights).

A more detailed description of unpacking behavior for iterator types (Dataset, generator, Sequence) is given below.

Returns:
yhat: numerical matrix containing the predicted responses.

run_all (*augment=False*)
Runs experiment
Does all the setup data, model, fit and evaluate

save (*path='model'*)
Saves the model.
Save the model in binary format on local storage.

Args:
path (`str`): The model path

setup_data (*augment=False*)
Setup data function
This function can be used by child classes to prepare data or perform other tasks that dont need to be repeated for every training run.

Args:
augment (bool): applies image augmenttion to training data

setup_model ( )
Setup model from yaml MODEL
This function converts the yaml MODEL:GRAPH: config to an instance of tf.keras.Sequential

to_string ( )
Print model type.
Implementing child classes should use this function to print the model_type.

dq0.sdk.models.tf.tf_hub module

TF Hub pretrained Models

class `dq0.sdk.models.tf.tf_hub.`**TFHub** (*tf_hub_url=None*)
Bases: `dq0.sdk.models.tf.neural_network_yaml.NeuralNetworkYaml`
Tensorflow Hub Network Model.
Uses NeuralNetworkYaml to read a TF Hub Yaml config to define the model.

setup_data (*augment=False*)
Setup Predefined data

args:
task (`str`): string specifying the task, i.e, im_clf or text_clf augment (bool): augment training data

returns:
X_train: true x or generator containing y y_train: true y or if x generator then None X_test: true x or generator containing y y_test: true y or if x generator then None fit_kwargs: steps_per_epoch and validation data if used

dq0.sdk.models.user package

DQ0 SDK User Model Template project (used by dq0-cli when new project was created).

class dq0.sdk.models.user.**UserModel**

Bases: dq0.sdk.models.model.Model

Derived from dq0.sdk.models.Model class

Model classes provide a setup method for data and model definitions.

fit (***kwargs*)

Train model on a dataset passed as input.

Args:

kwargs (dict): dictionary of optional arguments

load (*path*)

Loads the model.

Load the model from local storage.

Args:

path (str): The model path

preprocess ( )

Preprocess the data

Preprocess the data set. The input data is read from the attached source.

At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.

For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.

Use *self.data_source.read()* to read the attached data.

Returns:

preprocessed data

save (*path*)

Saves the model.

Save the model in binary format on local storage.

Args:

path (str): The model path

setup_data ( )

Setup data function

This function can be used to prepare data or perform other tasks for the training run.

At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.

For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.

Use *self.data_source.read()* to read the attached data.

setup_model ( )

Setup model function

Define the model here.

Submodules

dq0.sdk.models.user.user_model module

User Model template

class dq0.sdk.models.user.user_model.**UserModel**

Bases: dq0.sdk.models.model.Model

Derived from dq0.sdk.models.Model class

Model classes provide a setup method for data and model definitions.

fit (***kwargs*)

Train model on a dataset passed as input.

Args:

kwargs (dict): dictionary of optional arguments

load (*path*)

Loads the model.

Load the model from local storage.

Args:

path (str): The model path

preprocess ( )

Preprocess the data

Preprocess the data set. The input data is read from the attached source.

At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.

For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.

Use *self.data_source.read()* to read the attached data.

Returns:

preprocessed data

save (*path*)

Saves the model.

Save the model in binary format on local storage.

Args:

path (str): The model path

setup_data ( )

Setup data function

This function can be used to prepare data or perform other tasks for the training run.

At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.

For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.

Use *self.data_source.read()* to read the attached data.

setup_model ( )

Setup model function

Define the model here.

dq0.sdk.models.yaml_configs package

DQ0 SDK YAML Configs

Submodules

dq0.sdk.models.yaml_configs.tf_hub_models module

Curated list of hub models

1.6.2 Submodules

dq0.sdk.models.model module

Model abstract base class

The Model class serves as the base class for all models.

Implementing subclasses have to define setup_data and setup_model functions.

class dq0.sdk.models.model.**Model**

Bases: dq0.sdk.projects.project.Project

Abstract base class for all models available through the SDK.
Model classes provide a setup method as well as the fit and predict ML model functions.

Attributes:

model_type (str): type of this model instance. Options: 'keras'. uuid (str): UUID of this model. data_source (dq0.sdk.data.Source): dict of attached data sources.

fit ( )

Train model on a dataset passed as input.

abstract get_clone ( )

Generates a new model with the same parameters, if they are not fit on the training data.
Generates a deep copy of the model without actually copying any attached dataset. It yields a new model with the same parameters that has not been fit on any data. Parameters fit to the training data like, e.g., model weights, are re-initialized in the clone.

Returns:

deep copy of model

abstract load (*path*)

Loads the model.
Implementing child classes should use this function to load the model from local storage.

Args:

path (str): The model path

preprocess ( )

Preprocess the data
Preprocess the data set. The input data is read from the attached source.
At runtime the selected datset is attached to this model. It is available as the *data_source* attribute.
For local testing call *model.attach_data_source(some_data_source)* manually before calling *setup_data()*.
Use *self.data_source.read()* to read the attached data.

Returns:

preprocessed data

abstract save (*path*)

Saves the model.
Implementing child classes should use this function to save the model in binary format on local storage.

Args:

path (str): The model path

abstract setup_data (***kwargs*)

Setup data function

This function can be used by child classes to prepare data or perform other tasks that dont need to be repeated for every training run.

abstract setup_model (***kwargs*)
Setup model function
Implementing child classes can use this method to define the model.

abstract to_string ( )
Print model type.
Implementing child classes should use this function to print the model_type.

1.7 dq0.sdk.pipeline package

1.7.1 Subpackages

dq0.sdk.pipeline.transformer namespace

Submodules

dq0.sdk.pipeline.transformer.transformer module

Pipeline transformers

class dq0.sdk.pipeline.transformer.transformer.**Binarizer** (**, threshold=0.0, copy=True, **kwargs*)
Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1

class dq0.sdk.pipeline.transformer.transformer.**ColumnSelector** (*selected_columns*)
Bases: dq0.sdk.pipeline.transformer.transformer.Transformer
Utility transformer to select specif columns of the DataFrame and drop the rest.

fit (*X, y=None*)
Nothing happens during fit. This is here for compatability with the pipline interface.

transform (*X*)

class dq0.sdk.pipeline.transformer.transformer.**FunctionTransformer** (*func=None, inverse_func=None, *, validate=False, accept_sparse=False, check_inverse=True, kw_args=None, inv_kw_args=None, **kwargs*)
Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1

class dq0.sdk.pipeline.transformer.transformer.**KBinsDiscretizer** (*n_bins=5, *, encode='onehot', strategy='quantile', dtype=None, **kwargs*)
Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_N

class dq0.sdk.pipeline.transformer.transformer.**KernelCenterer** (***kwargs*)
Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1

class dq0.sdk.pipeline.transformer.transformer.**LabelBinarizer** (**, neg_label=0, pos_label=1, sparse_output=False, **kwargs*)
Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_N

class dq0.sdk.pipeline.transformer.transformer.**LabelEncoder** (***kwargs*)
Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1

class dq0.sdk.pipeline.transformer.transformer.**MaxAbsScaler** (**, copy=True, **kwargs*)
Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1

class dq0.sdk.pipeline.transformer.transformer.**MinMaxScaler** (*feature_range=(0, 1)*, ***, *copy=True*, *clip=False*, ***kwargs*)

Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1

class dq0.sdk.pipeline.transformer.transformer.**MultiLabelBinarizer** (***, *classes=None*, *sparse_output=False*, ***kwargs*)

Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_N

class dq0.sdk.pipeline.transformer.transformer.**Normalizer** (*norm='l2'*, ***, *copy=True*, ***kwargs*)

Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1

class dq0.sdk.pipeline.transformer.transformer.**OneHotEncoder** (***, *categories='auto'*, *drop=None*, *sparse=True*, *dtype=<class 'numpy.float64'>*, *handle_unknown='error'*, ***kwargs*)

Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_N

class dq0.sdk.pipeline.transformer.transformer.**OrdinalEncoder** (***, *categories='auto'*, *dtype=<class 'numpy.float64'>*, *handle_unknown='error'*, *unknown_value=None*, ***kwargs*)

Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1

class dq0.sdk.pipeline.transformer.transformer.**PolynomialFeatures** (*degree=2*, ***, *interaction_only=False*, *include_bias=True*, *order='C'*, ***kwargs*)

Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_N

Take note is 1 to N mapping does not allows for interaction of the features.

class dq0.sdk.pipeline.transformer.transformer.**PowerTransformer** (*method='yeo-johnson'*, ***, *standardize=True*, *copy=True*, ***kwargs*)

Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1

class dq0.sdk.pipeline.transformer.transformer.**QuantileTransformer** (***, *n_quantiles=1000*, *output_distribution='uniform'*, *ignore_implicit_zeros=False*, *subsample=100000*, *random_state=None*, *copy=True*, ***kwargs*)

Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1

class dq0.sdk.pipeline.transformer.transformer.**RobustScaler** (***, *with_centering=True*, *with_scaling=True*, *quantile_range=(25.0, 75.0)*, *copy=True*, *unit_variance=False*, ***kwargs*)

Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1

class dq0.sdk.pipeline.transformer.transformer.**StandardScaler** (***, *copy=True*, *with_mean=True*, *with_std=True*, ***kwargs*)

Bases: dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1

class dq0.sdk.pipeline.transformer.transformer.**Transformer** (*input_col=None*, ***kwargs*)

Bases: abc.ABC

abstract fit (*X*, *y=None*)

fit_transform (*X*, *y=None*)

Call fit and then transform

abstract transform (*X*)

class dq0.sdk.pipeline.transformer.transformer.**Transformer_1_to_1** (*input_col=None*, ***kwargs*)

Bases: abc.ABC

Standart transformer with 1 to 1 column mappings

fit (*X*, *y=None*)

fit_transform (*X*, *y=None*)

transform (*X*)

class dq0.sdk.pipeline.transformer.transformer.**Transformer_1_to_N** (*input_col=None*, ***kwargs*)

Bases: dq0.sdk.pipeline.transformer.transformer.Transformer

Transformer with 1 to N column mappings (e.g. One-Hot-Encoding). The mapping if performed column wise so the N new columns can be named an track accordingly. A many to many mapping is not possible with this type of transformer

fit (*X*, *y=None*)

Sets up a separate transformer for every column in the DataFrame. Args:

X: pandas DataFrame y: None, ignored here. Only for compatability with pipeline

Retruns:

self

fit_transform (*X*, *y=None*)

Call fit and then transform

transform (*X*)

Transform X using the transformers per column

Args:

X: Dataframe that is to be one hot encoded

Returns:

Dataframe Xt

1.7.2 Submodules

dq0.sdk.pipeline.pipeline module

Copyright 2021, Gradient Zero All rights reserved

class dq0.sdk.pipeline.pipeline.**Pipeline** (*steps=None*, *config_path=None*, *transformers_root_dir='.'*, *log_key_string=''*, ***kwargs*)

Bases: object

fit (*X*, *y=None*, ***fit_params*)

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

get_params (*deep=True*)

dq0.sdk.pipeline.pipeline_config module

Copyright 2021, Gradient Zero All rights reserved

class dq0.sdk.pipeline.pipeline_config.**PipelineConfig** (*config_path*)

Bases: object

Helper class to set up a pipline with a given config yaml.

get_steps_from_config (*root_dir='./dq0/sdk/pipeline/transformer/transformer.py'*, *log_key_string=''*)

Goes though the list pipeline of the config and sets ups the setps list of tuples to initialize the pipeline with.

read_from_yaml (*yaml_input*)
Reads metadata from the given yaml input.

Args:
yaml_input: open yaml file stream or yaml string.

read_from_yaml_file (*filename*)
Reads metadata from the given yaml file.

Args:
filename: the path to the yaml file.

1.8 dq0.sdk.projects package

DQ0 SDK Projects Package

This package contains the project abstract base class.

class dq0.sdk.projects.**Project** (*data_source=None*)
Bases: abc.ABC
Abstract base class for all all models and data jobs.
Project classes provide a attach_data_source function that is used to assign a selected data source to a project (to be used by model or data jobs).

Attributes:
data_source (dq0.sdk.data.Source): attached data source.

attach_data_source (*data_source*)
Attach a data source to the project.
This function needs to be called at least once. All data operations will use one of the attached data sources.

Args:
data_source (dq0.sdk.data.Source): The new data source to attach

detach_data_source (*data_source*)
Detaches a data source from the project.

Args:
data_source (dq0.sdk.data.Source): The data source to remove

1.8.1 Submodules

dq0.sdk.projects.project module

Project abstract base class

The Project class serves as the base class for all models and data jobs.

class dq0.sdk.projects.project.**Project** (*data_source=None*)
Bases: abc.ABC
Abstract base class for all all models and data jobs.
Project classes provide a attach_data_source function that is used to assign a selected data source to a project (to be used by model or data jobs).

Attributes:
data_source (dq0.sdk.data.Source): attached data source.

attach_data_source (*data_source*)
Attach a data source to the project.
This function needs to be called at least once. All data operations will use one of the attached data sources.

Args:
data_source (dq0.sdk.data.Source): The new data source to attach

detach_data_source (*data_source*)
Detaches a data source from the project.

Args:
data_source (dq0.sdk.data.Source): The data source to remove

1.9 dq0.sdk.utils package

DQ0 SDK Utils Package

class dq0.sdk.utils.**YamlConfig** (*yaml_path*, *yaml_dict=None*, *custom_objects={'KerasLayer': <class 'tensorflow_hub.keras_layer.KerasLayer'>}*)
Bases: object
Yaml parser for tf.keras models
Yaml parser class for tf.Keras config files.

dump_yaml (*yaml_dict*)

read_yaml_file ( )
Reads yaml file
This function parses a yaml file to self.yaml_dict

save_yaml ( )
Save yaml dict to a yaml file

1.9.1 Submodules

dq0.sdk.utils.managed_classes module

Managed classes of custom_objects, Optimizers and Losses

dq0.sdk.utils.parse_args module

dq0.sdk.utils.parse_args.**parse_kwargs** (*kwargs*)

dq0.sdk.utils.parse_args.**parse_value** (*val*)

dq0.sdk.utils.yaml_config module

Process Yaml Config

Process yaml file to instantiate a basic tensorflow neural network implementation using Keras.

Example:
```python if __name__ == "__main__":
    yaml_path = 'your path' model = MyAwesomeModel(yaml_path) model.setup_model()
```

class dq0.sdk.utils.yaml_config.**YamlConfig** (*yaml_path, yaml_dict=None, custom_objects={'KerasLayer': <class 'tensorflow_hub.keras_layer.KerasLayer'>}*)

Bases: object

Yaml parser for tf.keras models

Yaml parser class for tf.Keras config files.

dump_yaml (*yaml_dict*)

read_yaml_file ( )

Reads yaml file

This function parses a yaml file to self.yaml_dict

save_yaml ( )

Save yaml dict to a yaml file

d

dq0
    dq0.sdk, XX
    dq0.sdk.cli, 1
    dq0.sdk.cli.api, 7
    dq0.sdk.cli.api.client, 8
    dq0.sdk.cli.api.routes, 9
    dq0.sdk.cli.data, 15
    dq0.sdk.cli.experiment, 16
    dq0.sdk.cli.model, 17
    dq0.sdk.cli.project, 19
    dq0.sdk.cli.query, 21
    dq0.sdk.cli.runner, 9
    dq0.sdk.cli.runner.data_runner, 10
    dq0.sdk.cli.runner.model_runner, 11
    dq0.sdk.cli.runner.query_runner, 12
    dq0.sdk.cli.runner.runner, 13
    dq0.sdk.cli.runner.state, 14
    dq0.sdk.cli.utils, 14
    dq0.sdk.cli.utils.code, 15
    dq0.sdk.cli.utils.uuid, 15
    dq0.sdk.data, 22
    dq0.sdk.data.binary, 23
    dq0.sdk.data.binary.excel, 25
    dq0.sdk.data.binary.feather, 26
    dq0.sdk.data.binary.hdf5, 26
    dq0.sdk.data.binary.odf, 26
    dq0.sdk.data.binary.orc, 27
    dq0.sdk.data.binary.parquet, 27
    dq0.sdk.data.binary.sas, 27
    dq0.sdk.data.binary.spss, 28
    dq0.sdk.data.binary.stata, 28
    dq0.sdk.data.data_source_factory, 51
    dq0.sdk.data.image, 28
    dq0.sdk.data.image.image, 29
    dq0.sdk.data.metadata, 29
    dq0.sdk.data.metadata.metadata, 30
    dq0.sdk.data.preprocessing, 33
    dq0.sdk.data.preprocessing.preprocessing, 33
    dq0.sdk.data.source, 52
    dq0.sdk.data.sql, 34
    dq0.sdk.data.sql.big_query, 37
    dq0.sdk.data.sql.drill, 37
    dq0.sdk.data.sql.mssql, 38
    dq0.sdk.data.sql.mysql, 38
    dq0.sdk.data.sql.oracle, 39
    dq0.sdk.data.sql.postgresql, 39
    dq0.sdk.data.sql.redshift, 39
    dq0.sdk.data.sql.sap_hana, 40
    dq0.sdk.data.sql.snowflake, 40
    dq0.sdk.data.sql.sql, 41
    dq0.sdk.data.sql.sqlite, 42
    dq0.sdk.data.text, 42
    dq0.sdk.data.text.csv, 43
    dq0.sdk.data.text.json, 43
    dq0.sdk.data.transform, 52
    dq0.sdk.data.utils, 43
    dq0.sdk.data.utils.plotting, 43
    dq0.sdk.data.utils.util, 46
    dq0.sdk.errors, 53
    dq0.sdk.errors.errors, 53
    dq0.sdk.estimators, 54
    dq0.sdk.estimators.base_mixin, 60
    dq0.sdk.estimators.data_handler, 55
    dq0.sdk.estimators.data_handler.base, 55
    dq0.sdk.estimators.data_handler.csv, 55
    dq0.sdk.estimators.data_handler.utils, 55
    dq0.sdk.estimators.ensemble, 56
    dq0.sdk.estimators.ensemble.sklearn_ensemble 56
    dq0.sdk.estimators.estimator, 61
    dq0.sdk.estimators.linear_model, 57
    dq0.sdk.estimators.linear_model.diffprivlib_ 57

dq0.sdk.estimators.linear_model.sklearn_lm, 57
dq0.sdk.estimators.SVM, 54
dq0.sdk.estimators.SVM.sklearn_svm, 54
dq0.sdk.estimators.tf, 58
dq0.sdk.estimators.tf.keras_base, 58
dq0.sdk.estimators.tf.keras_dense_classifier, 59
dq0.sdk.estimators.tf.keras_dense_regressor, 60
dq0.sdk.examples, 61
dq0.sdk.examples.census, 61
dq0.sdk.examples.census.bayesian, 61
dq0.sdk.examples.census.bayesian.model.user_model, 61
dq0.sdk.examples.census.bayesian.run_demo, 62
dq0.sdk.examples.census.eda, 64
dq0.sdk.examples.census.preprocessed, 62
dq0.sdk.examples.census.preprocessed.model.user_model, 62
dq0.sdk.examples.census.preprocessed.run_demo, 63
dq0.sdk.examples.census.raw, 63
dq0.sdk.examples.census.raw.model.user_model, 63
dq0.sdk.examples.census.raw.run_demo, 64
dq0.sdk.examples.cifar, 64
dq0.sdk.examples.cifar.model.user_model, 64
dq0.sdk.examples.cifar.model.user_model_old, 65
dq0.sdk.examples.cifar.run_demo, 65
dq0.sdk.examples.har.model.user_model, 66
dq0.sdk.examples.har.run_demo, 67
dq0.sdk.examples.medical_insurance, 67
dq0.sdk.examples.medical_insurance.model.user_model, 67
dq0.sdk.examples.medical_insurance.run_demo, 67
dq0.sdk.examples.newsgroups, 68
dq0.sdk.examples.newsgroups.bayesian, 68
dq0.sdk.examples.newsgroups.bayesian.model.user_model, 68
dq0.sdk.examples.newsgroups.bayesian.model.user_model_old_needs_SOURCE, 68
dq0.sdk.examples.newsgroups.bayesian.run_demo, 68
dq0.sdk.examples.newsgroups.network, 69
dq0.sdk.examples.newsgroups.network.model.us 69
dq0.sdk.examples.newsgroups.network.model.us 69
dq0.sdk.examples.newsgroups.network.model.us 70
dq0.sdk.examples.newsgroups.network.run_demo 70
dq0.sdk.examples.patient, 70
dq0.sdk.examples.patient.model.user_model, 70
dq0.sdk.examples.pneumonia.data_preparation, 71
dq0.sdk.examples.pneumonia.my_model, 71
dq0.sdk.examples.pneumonia.my_model_int_enco 71
dq0.sdk.models, 72
dq0.sdk.models.bayes, 73
dq0.sdk.models.bayes.naive_bayesian_model, 74
dq0.sdk.models.model, 84
dq0.sdk.models.tf, 75
dq0.sdk.models.tf.neural_network, 78
dq0.sdk.models.tf.neural_network_classificat 79
dq0.sdk.models.tf.neural_network_multiclass_ 79
dq0.sdk.models.tf.neural_network_regression, 80
dq0.sdk.models.tf.neural_network_yaml, 80
dq0.sdk.models.tf.tf_hub, 81
dq0.sdk.models.user, 82
dq0.sdk.models.user.user_model, 82
dq0.sdk.models.yaml_configs, 83
dq0.sdk.models.yaml_configs.tf_hub_models, 83
dq0.sdk.pipeline, 85
dq0.sdk.pipeline.pipeline, 87
dq0.sdk.pipeline.pipeline_config, 87
dq0.sdk.pipeline.transformer.transformer, 85
dq0.sdk.projects, 88
dq0.sdk.projects.project, 88
dq0.sdk.utils, 89
dq0.sdk.utils.managed_classes, 89
dq0.sdk.utils.parse_args, 89
dq0.sdk.utils.yaml_config, 89

A

AdaBoostClassifier (class in dq0.sdk.estimators.ensemble.sklearn_ensemble), 56
AdaBoostRegressor (class in dq0.sdk.estimators.ensemble.sklearn_ensemble), 56
add_function() (in module dq0.sdk.cli.utils.code), 15
add_shared_axis_labels() (in module dq0.sdk.data.utils.plotting), 44
all() (dq0.sdk.cli.Data method), 5
all() (dq0.sdk.cli.data.Data method), 15
as_dict() (dq0.sdk.cli.Data method), 5
as_dict() (dq0.sdk.cli.data.Data method), 15
attach_data_source() (dq0.sdk.cli.Project method), 3
attach_data_source() (dq0.sdk.cli.project.Project method), 19
attach_data_source() (dq0.sdk.projects.Project method), 88
attach_data_source() (dq0.sdk.projects.project.Project method), 89

B

BaggingClassifier (class in dq0.sdk.estimators.ensemble.sklearn_ensemble), 56
BaggingRegressor (class in dq0.sdk.estimators.ensemble.sklearn_ensemble), 56
BasicDataHandler (class in dq0.sdk.estimators.data_handler.base), 55
BiclusterMixin (class in dq0.sdk.estimators.base_mixin), 60
BigQuery (class in dq0.sdk.data.sql), 34
BigQuery (class in dq0.sdk.data.sql.big_query), 37
Binarizer (class in dq0.sdk.pipeline.transformer.transformer), 85

C

cancel() (dq0.sdk.cli.runner.data_runner.DataRunner method), 11
cancel() (dq0.sdk.cli.runner.DataRunner method), 9
cancel() (dq0.sdk.cli.runner.model_runner.ModelRunner method), 12
cancel() (dq0.sdk.cli.runner.ModelRunner method), 10
cancel() (dq0.sdk.cli.runner.query_runner.QueryRunner method), 13
cancel() (dq0.sdk.cli.runner.QueryRunner method), 10
cancel() (dq0.sdk.cli.runner.runner.Runner method), 13
case_insensitive_str_comparison() (in module dq0.sdk.data.utils.util), 46
check_data_structure_type_consistency() (in module dq0.sdk.data.utils.util), 46
check_for_valid_numerical_encoding_of_labels() (in module dq0.sdk.data.utils.util), 46
check_signature() (in module dq0.sdk.cli.utils.code), 15
checkSDKResponse() (in module dq0.sdk.errors), 53
checkSDKResponse() (in module dq0.sdk.errors.errors), 54
ClassifierMixin (class in dq0.sdk.estimators.base_mixin), 60
Client (class in dq0.sdk.cli.api), 7
Client (class in dq0.sdk.cli.api.client), 8
ClusterMixin (class in dq0.sdk.estimators.base_mixin), 60
Column (class in dq0.sdk.data.metadata.metadata), 31
ColumnSelector (class in dq0.sdk.pipeline.transformer.transformer), 85
combine_with() (dq0.sdk.data.metadata.Metadata method), 29
combine_with() (dq0.sdk.data.metadata.metadata.Metadata method), 31
commit() (dq0.sdk.cli.Project method), 3
commit() (dq0.sdk.cli.project.Project method), 19
compute_confusion_matrix() (in module dq0.sdk.data.utils.plotting), 44
compute_metrics_scores() (in module dq0.sdk.-

data.utils.util), 47
concatenate_train_test_datasets() (in module dq0.sdk.data.utils.util), 47
concatenate_train_test_datasets_np_array() (in module dq0.sdk.data.utils.util), 47
concatenate_train_test_datasets_pd_Dataframes() (in module dq0.sdk.data.utils.util), 47
copy_obj_attributes() (in module dq0.sdk.data.utils.util), 47
create_dataset() (in module dq0.sdk.examples.pneumonia.data_preparation), 71
create_df() (in module dq0.sdk.examples.pneumonia.data_preparation), 71
create_feature_vector() (in module dq0.sdk.examples.pneumonia.data_preparation), 71
create_from_type() (in module dq0.sdk.data.data_source_factory), 52
CSV (class in dq0.sdk.data.text), 42
CSV (class in dq0.sdk.data.text.csv), 43
CSVDataHandler (class in dq0.sdk.estimators.data_handler.csv), 55

D

Data (class in dq0.sdk.cli), 5
Data (class in dq0.sdk.cli.data), 15
data_handler_factory() (in module dq0.sdk.estimators.data_handler.utils), 56
dataframe_has_columns_of_these_types() (in module dq0.sdk.data.utils.util), 48
DataRunner (class in dq0.sdk.cli.runner), 9
DataRunner (class in dq0.sdk.cli.runner.data_runner), 11
datasets_are_equal() (in module dq0.sdk.data.utils.util), 48
detach_data_source() (dq0.sdk.cli.Project method), 3
detach_data_source() (dq0.sdk.cli.project.Project method), 19
detach_data_source() (dq0.sdk.projects.Project method), 88
detach_data_source() (dq0.sdk.projects.project.Project method), 89
distribution() (dq0.sdk.cli.Data method), 5
distribution() (dq0.sdk.cli.data.Data method), 16
dq0.sdk (module), 1
dq0.sdk.cli (module), 1
dq0.sdk.cli.api (module), 7
dq0.sdk.cli.api.client (module), 8
dq0.sdk.cli.api.routes (module), 9
dq0.sdk.cli.data (module), 15
dq0.sdk.cli.experiment (module), 16
dq0.sdk.cli.model (module), 17
dq0.sdk.cli.project (module), 19
dq0.sdk.cli.query (module), 21
dq0.sdk.cli.runner (module), 9
dq0.sdk.cli.runner.data_runner (module), 10
dq0.sdk.cli.runner.model_runner (module), 11
dq0.sdk.cli.runner.query_runner (module), 12
dq0.sdk.cli.runner.runner (module), 13
dq0.sdk.cli.runner.state (module), 14
dq0.sdk.cli.utils (module), 14
dq0.sdk.cli.utils.code (module), 15
dq0.sdk.cli.utils.uuid (module), 15
dq0.sdk.data (module), 22
dq0.sdk.data.binary (module), 23
dq0.sdk.data.binary.excel (module), 25
dq0.sdk.data.binary.feather (module), 26
dq0.sdk.data.binary.hdf5 (module), 26
dq0.sdk.data.binary.odf (module), 26
dq0.sdk.data.binary.orc (module), 27
dq0.sdk.data.binary.parquet (module), 27
dq0.sdk.data.binary.sas (module), 27
dq0.sdk.data.binary.spss (module), 28
dq0.sdk.data.binary.stata (module), 28
dq0.sdk.data.data_source_factory (module), 51
dq0.sdk.data.image (module), 28
dq0.sdk.data.image.image (module), 29
dq0.sdk.data.metadata (module), 29
dq0.sdk.data.metadata.metadata (module), 30
dq0.sdk.data.preprocessing (module), 33
dq0.sdk.data.preprocessing.preprocessing (module), 33
dq0.sdk.data.source (module), 52
dq0.sdk.data.sql (module), 34
dq0.sdk.data.sql.big_query (module), 37
dq0.sdk.data.sql.drill (module), 37
dq0.sdk.data.sql.mssql (module), 38
dq0.sdk.data.sql.mysql (module), 38
dq0.sdk.data.sql.oracle (module), 39
dq0.sdk.data.sql.postgresql (module), 39
dq0.sdk.data.sql.redshift (module), 39
dq0.sdk.data.sql.sap_hana (module), 40
dq0.sdk.data.sql.snowflake (module), 40
dq0.sdk.data.sql.sql (module), 41
dq0.sdk.data.sql.sqlite (module), 42
dq0.sdk.data.text (module), 42
dq0.sdk.data.text.csv (module), 43
dq0.sdk.data.text.json (module), 43
dq0.sdk.data.transform (module), 52
dq0.sdk.data.utils (module), 43
dq0.sdk.data.utils.plotting (module), 43
dq0.sdk.data.utils.util (module), 46
dq0.sdk.errors (module), 53
dq0.sdk.errors.errors (module), 53
dq0.sdk.estimators (module), 54
dq0.sdk.estimators.base_mixin (module), 60
dq0.sdk.estimators.data_handler (module), 55
dq0.sdk.estimators.data_handler.base (module), 55
dq0.sdk.estimators.data_handler.csv (module), 55
dq0.sdk.estimators.data_handler.utils (module), 55

dq0.sdk.estimators.ensemble (module), 56
dq0.sdk.estimators.ensemble.sklearn_ensemble (module), 56
dq0.sdk.estimators.estimator (module), 61
dq0.sdk.estimators.linear_model (module), 57
dq0.sdk.estimators.linear_model.diffprivlib_lm (module), 57
dq0.sdk.estimators.linear_model.sklearn_lm (module), 57
dq0.sdk.estimators.SVM (module), 54
dq0.sdk.estimators.SVM.sklearn_svm (module), 54
dq0.sdk.estimators.tf (module), 58
dq0.sdk.estimators.tf.keras_base (module), 58
dq0.sdk.estimators.tf.keras_dense_classifier (module), 59
dq0.sdk.estimators.tf.keras_dense_regressor (module), 60
dq0.sdk.examples (module), 61
dq0.sdk.examples.census (module), 61
dq0.sdk.examples.census.bayesian (module), 61
dq0.sdk.examples.census.bayesian.model.user_model (module), 61
dq0.sdk.examples.census.bayesian.run_demo (module), 62
dq0.sdk.examples.census.eda (module), 64
dq0.sdk.examples.census.preprocessed (module), 62
dq0.sdk.examples.census.preprocessed.model.user_model (module), 62
dq0.sdk.examples.census.preprocessed.run_demo (module), 63
dq0.sdk.examples.census.raw (module), 63
dq0.sdk.examples.census.raw.model.user_model (module), 63
dq0.sdk.examples.census.raw.run_demo (module), 64
dq0.sdk.examples.cifar (module), 64
dq0.sdk.examples.cifar.model.user_model (module), 64
dq0.sdk.examples.cifar.model.user_model_old (module), 65
dq0.sdk.examples.cifar.run_demo (module), 65
dq0.sdk.examples.har.model.user_model (module), 66
dq0.sdk.examples.har.run_demo (module), 67
dq0.sdk.examples.medical_insurance (module), 67
dq0.sdk.examples.medical_insurance.model.user_model (module), 67
dq0.sdk.examples.medical_insurance.run_demo (module), 67
dq0.sdk.examples.newsgroups (module), 68
dq0.sdk.examples.newsgroups.bayesian (module), 68
dq0.sdk.examples.newsgroups.bayesian.model.user_model (module), 68
dq0.sdk.examples.newsgroups.bayesian.model.user_model_old (module), 68
dq0.sdk.examples.newsgroups.bayesian.run_demo (module), 68
dq0.sdk.examples.newsgroups.network (module), 69
dq0.sdk.examples.newsgroups.network.model.user_model (module), 69
dq0.sdk.examples.newsgroups.network.model.user_model_from (module), 69
dq0.sdk.examples.newsgroups.network.model.user_model_old (module), 70
dq0.sdk.examples.newsgroups.network.run_demo (module), 70
dq0.sdk.examples.patient (module), 70
dq0.sdk.examples.patient.model.user_model (module), 70
dq0.sdk.examples.pneumonia.data_preparation (module), 71
dq0.sdk.examples.pneumonia.my_model (module), 71
dq0.sdk.examples.pneumonia.my_model_int_encoding (module), 71
dq0.sdk.models (module), 72
dq0.sdk.models.bayes (module), 73
dq0.sdk.models.bayes.naive_bayesian_model (module), 74
dq0.sdk.models.model (module), 84
dq0.sdk.models.tf (module), 75
dq0.sdk.models.tf.neural_network (module), 78
dq0.sdk.models.tf.neural_network_classification (module), 79
dq0.sdk.models.tf.neural_network_multiclass_classification (module), 79
dq0.sdk.models.tf.neural_network_regression (module), 80
dq0.sdk.models.tf.neural_network_yaml (module), 80
dq0.sdk.models.tf.tf_hub (module), 81
dq0.sdk.models.user (module), 82
dq0.sdk.models.user.user_model (module), 82
dq0.sdk.models.yaml_configs (module), 83
dq0.sdk.models.yaml_configs.tf_hub_models (module), 83
dq0.sdk.pipeline (module), 85
dq0.sdk.pipeline.pipeline (module), 87
dq0.sdk.pipeline.pipeline_config (module), 87
dq0.sdk.pipeline.transformer.transformer (module), 85
dq0.sdk.projects (module), 88
dq0.sdk.projects.project (module), 88
dq0.sdk.utils (module), 89
dq0.sdk.utils.managed_classes (module), 89
dq0.sdk.utils.parse_args (module), 89
dq0.sdk.utils.yaml_config (module), 89
DQ0SDKError, 53, 54
Drill (class in dq0.sdk.data.sql), 34
Drill (class in dq0.sdk.data.sql.drill), 38
drop_columns_SOURCE_value() (dq0.sdk.da-

ta.metadata.Metadata method), 30
drop_columns_with_key_value() (dq0.sdk.data.metadata.metadata.Metadata method), 31
dump_model() (in module dq0.sdk.data.utils.util), 48
dump_yaml() (dq0.sdk.utils.yaml_config.YamlConfig method), 90
dump_yaml() (dq0.sdk.utils.YamlConfig method), 89

E

eda() (in module dq0.sdk.examples.census.eda), 64
ElasticNet (class in dq0.sdk.estimators.linear_model.sklearn_lm), 57
empty_folder() (in module dq0.sdk.data.utils.util), 48
estimate_freq_of_labels() (in module dq0.sdk.data.utils.util), 48
Estimator (class in dq0.sdk.estimators.estimator), 61
evaluate() (dq0.sdk.examples.pneumonia.my_model.UserModel method), 71
evaluate() (dq0.sdk.examples.pneumonia.my_model_int_encoding.UserModel method), 72
evaluate() (dq0.sdk.models.bayes.naive_bayesian_model.NaiveBayesianModel method), 74
evaluate() (dq0.sdk.models.bayes.NaiveBayesianModel method), 73
evaluate() (dq0.sdk.models.tf.neural_network.NeuralNetwork method), 78
evaluate() (dq0.sdk.models.tf.NeuralNetwork method), 75
Excel (class in dq0.sdk.data.binary), 23
Excel (class in dq0.sdk.data.binary.excel), 25
execute() (dq0.sdk.cli.Query method), 6
execute() (dq0.sdk.cli.query.Query method), 21
execute() (dq0.sdk.data.sql.big_query.BigQuery method), 37
execute() (dq0.sdk.data.sql.BigQuery method), 34
execute() (dq0.sdk.data.sql.Drill method), 35
execute() (dq0.sdk.data.sql.drill.Drill method), 38
execute() (dq0.sdk.data.sql.MSSQL method), 35
execute() (dq0.sdk.data.sql.mssql.MSSQL method), 38
execute() (dq0.sdk.data.sql.MySQL method), 35
execute() (dq0.sdk.data.sql.mysql.MySQL method), 39
execute() (dq0.sdk.data.sql.Oracle method), 35
execute() (dq0.sdk.data.sql.oracle.Oracle method), 39
execute() (dq0.sdk.data.sql.PostgreSQL method), 36
execute() (dq0.sdk.data.sql.postgresql.PostgreSQL method), 39
execute() (dq0.sdk.data.sql.Redshift method), 36
execute() (dq0.sdk.data.sql.redshift.Redshift method), 40
execute() (dq0.sdk.data.sql.sap_hana.SAPHana method), 40
execute() (dq0.sdk.data.sql.SAPHana method), 36
execute() (dq0.sdk.data.sql.Snowflake method), 37
execute() (dq0.sdk.data.sql.snowflake.Snowflake method), 41
execute() (dq0.sdk.data.sql.sql.SQL method), 41
execute() (dq0.sdk.data.sql.SQLite method), 37
execute() (dq0.sdk.data.sql.sqlite.SQLite method), 42
execute() (dq0.sdk.data.Transform method), 22
execute() (dq0.sdk.data.transform.Transform method), 53
Experiment (class in dq0.sdk.cli), 1
Experiment (class in dq0.sdk.cli.experiment), 17
extract_count_features_from_text_corpus() (in module dq0.sdk.data.preprocessing.preprocessing), 33
ExtraTreesClassifier (class in dq0.sdk.estimators.ensemble.sklearn_ensemble), 56
ExtraTreesRegressor (class in dq0.sdk.estimators.ensemble.sklearn_ensemble), 56

F

fatal_error() (in module dq0.sdk.errors), 53
fatal_error() (in module dq0.sdk.errors.errors), 54
Feather (class in dq0.sdk.data.binary), 23
Feather (class in dq0.sdk.data.binary.feather), 26
fit() (dq0.sdk.estimators.estimator.Estimator method), 61
fit() (dq0.sdk.estimators.tf.keras_base.NeuralNetworkBase method), 59
fit() (dq0.sdk.models.bayes.naive_bayesian_model.NaiveBayesianModel method), 74
fit() (dq0.sdk.models.bayes.NaiveBayesianModel method), 73
fit() (dq0.sdk.models.Model method), 72
fit() (dq0.sdk.models.model.Model method), 84
fit() (dq0.sdk.models.tf.neural_network.NeuralNetwork method), 78
fit() (dq0.sdk.models.tf.NeuralNetwork method), 75
fit() (dq0.sdk.models.user.user_model.UserModel method), 83
fit() (dq0.sdk.models.user.UserModel method),

82
fit() (dq0.sdk.pipeline.pipeline.Pipeline method), 87
fit() (dq0.sdk.pipeline.transformer.transformer.ColumnSelector method), 85
fit() (dq0.sdk.pipeline.transformer.transformer.Transformer method), 86
fit() (dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1 method), 87
fit() (dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_N method), 87
fit_predict() (dq0.sdk.estimators.base_mixin.ClusterMixin method), 60
fit_transform() (dq0.sdk.estimators.base_mixin.TransformerMixin method), 61
fit_transform() (dq0.sdk.pipeline.pipeline.Pipeline method), 87
fit_transform() (dq0.sdk.pipeline.transformer.transformer.Transformer method), 86
fit_transform() (dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1 method), 87
fit_transform() (dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_N method), 87
fix_limitation_of_Keras_fit_and_predict_functions() (in module dq0.sdk.models.tf.neural_network), 79
for_data() (dq0.sdk.cli.Query method), 6
for_data() (dq0.sdk.cli.query.Query method), 21
format_float_lower_than_1() (in module dq0.sdk.data.utils.util), 48
from_meta() (dq0.sdk.data.metadata.metadata.Column static method), 31
from_meta() (dq0.sdk.data.metadata.metadata.Schema static method), 32
from_meta() (dq0.sdk.data.metadata.metadata.Table static method), 32
FunctionTransformer (class in dq0.sdk.pipeline.transformer.transformer), 85

G

get() (dq0.sdk.cli.api.Client method), 7
get() (dq0.sdk.cli.api.client.Client method), 8
get_all_schema_names() (dq0.sdk.data.metadata.Metadata method), 30
get_all_schema_names() (dq0.sdk.data.metadata.metadata.Metadata method), 31
get_all_table_names() (dq0.sdk.data.metadata.Metadata method), 30
get_all_table_names() (dq0.sdk.data.metadata.metadata.Metadata method), 31
get_all_tables() (dq0.sdk.data.metadata.Metadata method), 30
get_all_tables() (dq0.sdk.data.metadata.metadata.Metadata method), 31
get_attached_data_sources() (dq0.sdk.cli.Project method), 3
get_attached_data_sources() (dq0.sdk.cli.project.Project method), 20
get_available_data_sources() (dq0.sdk.cli.Data static method), 5
get_available_data_sources() (dq0.sdk.cli.data.Data static method), 16
get_available_data_sources() (dq0.sdk.cli.Project method), 3
get_available_data_sources() (dq0.sdk.cli.project.Project method), 20
get_categorical_and_quantitative_features_list() (in module dq0.sdk.data.utils.util), 48
get_clone() (dq0.sdk.models.bayes.naive_bayesian_model.NaiveBayesianModel method), 74
get_clone() (dq0.sdk.models.bayes.NaiveBayesianModel method), 73
get_clone() (dq0.sdk.models.Model method), 72
get_clone() (dq0.sdk.models.model.Model method), 84
get_clone() (dq0.sdk.models.tf.neural_network.NeuralNetwork method), 78
get_clone() (dq0.sdk.models.tf.NeuralNetwork method), 75
get_connection() (dq0.sdk.data.sql.sql.SQL method), 41
get_data_info() (dq0.sdk.cli.Data static method), 5
get_data_info() (dq0.sdk.cli.data.Data static method), 16
get_data_info() (dq0.sdk.cli.Project method), 4
get_data_info() (dq0.sdk.cli.project.Project method), 20
get_dataset_names() (dq0.sdk.cli.Query method), 6
get_dataset_names() (dq0.sdk.cli.query.Query method), 22
get_error() (dq0.sdk.cli.runner.runner.Runner method), 14
get_feature_target_cols() (dq0.sdk.data.metadata.Metadata method), 30
get_feature_target_cols() (dq0.sdk.data.metadata.metadata.Metadata method), 31
get_fn() (in module dq0.sdk.data.utils.util), 49
get_header() (dq0.sdk.data.metadata.Metadata method), 30
get_header() (dq0.sdk.data.metadata.metadata.Metadata method), 31
get_input_dim() (dq0.sdk.estimators.data_handler.csv.CSVDataHandler method), 55
get_last_data_run() (dq0.sdk.cli.Experiment method), 1
get_last_data_run() (dq0.sdk.cli.experiment.Experiment method), 17
get_last_model_run() (dq0.sdk.cli.Experiment method), 1
get_last_model_run() (dq0.sdk.cli.experiment.-

Experiment method), 17
get_model() (dq0.sdk.cli.runner.model_runner.ModelRunner method), 12
get_model() (dq0.sdk.cli.runner.ModelRunner method), 10
get_output_dim() (dq0.sdk.estimators.data_handler.csv.CSVDataHandler method), 55
get_param_configuration_for_publication_quality_plot() (in module dq0.sdk.data.utils.plotting), 44
get_params() (dq0.sdk.pipeline.pipeline.Pipeline method), 87
get_percentage_freq_of_values() (in module dq0.sdk.data.utils.util), 49
get_results() (dq0.sdk.cli.runner.runner.Runner method), 14
get_sample_data() (dq0.sdk.cli.Project method), 4
get_sample_data() (dq0.sdk.cli.project.Project method), 20
get_state() (dq0.sdk.cli.runner.data_runner.DataRunner method), 11
get_state() (dq0.sdk.cli.runner.DataRunner method), 9
get_state() (dq0.sdk.cli.runner.model_runner.ModelRunner method), 12
get_state() (dq0.sdk.cli.runner.ModelRunner method), 10
get_state() (dq0.sdk.cli.runner.query_runner.QueryRunner method), 13
get_state() (dq0.sdk.cli.runner.QueryRunner method), 10
get_state() (dq0.sdk.cli.runner.runner.Runner method), 14
get_steps_from_config() (dq0.sdk.pipeline.pipeline_config.PipelineConfig method), 87
GradientBoostingClassifier (class in dq0.sdk.estimators.ensemble.sklearn_ensemble), 56
GradientBoostingRegressor (class in dq0.sdk.estimators.ensemble.sklearn_ensemble), 57

H

handle_missing_data() (in module dq0.sdk.data.preprocessing.preprocessing), 33
HDF5 (class in dq0.sdk.data.binary), 23
HDF5 (class in dq0.sdk.data.binary.hdf5), 26

I

Image (class in dq0.sdk.data.image), 28
Image (class in dq0.sdk.data.image.image), 29
info() (dq0.sdk.cli.Project method), 4
info() (dq0.sdk.cli.project.Project method), 20
initialize_rnd_numbers_generators_state() (in module dq0.sdk.data.utils.util), 49
instantiate_metrics_from_name() (in module dq0.sdk.data.utils.util), 49
is_numeric() (in module dq0.sdk.data.utils.util), 49
is_valid_uuid() (in module dq0.sdk.cli.utils), 15
is_valid_uuid() (in module dq0.sdk.cli.utils.uuid), 15

J

JSON (class in dq0.sdk.data.text), 42
JSON (class in dq0.sdk.data.text.json), 43

K

KBinsDiscretizer (class in dq0.sdk.pipeline.transformer.transformer), 85
Keras_Dense_Classifier_Binary (class in dq0.sdk.estimators.tf.keras_dense_classifier), 59
Keras_Dense_Classifier_Integer (class in dq0.sdk.estimators.tf.keras_dense_classifier), 59
Keras_Dense_Classifier_OHE (class in dq0.sdk.estimators.tf.keras_dense_classifier), 59
Keras_Dense_Regressor (class in dq0.sdk.estimators.tf.keras_dense_regressor), 60
KernelCenterer (class in dq0.sdk.pipeline.transformer.transformer), 85

L

LabelBinarizer (class in dq0.sdk.pipeline.transformer.transformer), 85
LabelEncoder (class in dq0.sdk.pipeline.transformer.transformer), 85
Lasso (class in dq0.sdk.estimators.linear_model.sklearn_lm), 58
layer_factory() (in module dq0.sdk.estimators.tf.keras_base), 59
LinearRegression (class in dq0.sdk.estimators.linear_model.sklearn_lm), 58
LinearRegressionDP (class in dq0.sdk.estimators.linear_model.diffprivlib_lm), 57
LinearSVC (class in dq0.sdk.estimators.SVM.sklearn_svm), 54
LinearSVR (class in dq0.sdk.estimators.SVM.sklearn_svm), 54
list_subfloders() (in module dq0.sdk.data.utils.util), 49
load() (dq0.sdk.cli.Project static method), 4
load() (dq0.sdk.cli.project.Project static method), 20
load() (dq0.sdk.models.bayes.naive_bayesian_model.NaiveBayesianModel method), 74
load() (dq0.sdk.models.bayes.NaiveBayesianModel method), 74

load() (dq0.sdk.models.Model method), 72
load() (dq0.sdk.models.model.Model method), 84
load() (dq0.sdk.models.tf.neural_network.NeuralNetwork method), 79
load() (dq0.sdk.models.tf.neural_network_yaml.NeuralNetworkYaml method), 80
load() (dq0.sdk.models.tf.NeuralNetwork method), 75
load() (dq0.sdk.models.tf.NeuralNetworkYaml method), 76
load() (dq0.sdk.models.user.user_model.UserModel method), 83
load() (dq0.sdk.models.user.UserModel method), 82
load_img_in_dir() (in module dq0.sdk.examples.pneumonia.data_preparation), 71
load_model_from_file() (in module dq0.sdk.data.utils.util), 49
load_params_from_config_file() (in module dq0.sdk.data.utils.util), 49
LogisticRegression (class in dq0.sdk.estimators.linear_model.sklearn_lm), 58
LogisticRegressionDP (class in dq0.sdk.estimators.linear_model.diffprivlib_lm), 57

M

manage_rnd_num_generators_state() (in module dq0.sdk.data.utils.util), 50
MaxAbsScaler (class in dq0.sdk.pipeline.transformer.transformer), 85
mean() (dq0.sdk.cli.Data method), 5
mean() (dq0.sdk.cli.data.Data method), 16
Metadata (class in dq0.sdk.data.metadata), 29
Metadata (class in dq0.sdk.data.metadata.metadata), 31
MinMaxScaler (class in dq0.sdk.pipeline.transformer.transformer), 86
missing_values_table() (in module dq0.sdk.data.utils.util), 50
Model (class in dq0.sdk.cli), 2
Model (class in dq0.sdk.cli.model), 18
Model (class in dq0.sdk.models), 72
Model (class in dq0.sdk.models.model), 84
ModelRunner (class in dq0.sdk.cli.runner), 9
ModelRunner (class in dq0.sdk.cli.runner.model_runner), 11
move_files() (in module dq0.sdk.data.utils.util), 50
MSSQL (class in dq0.sdk.data.sql), 35
MSSQL (class in dq0.sdk.data.sql.mssql), 38
MultiLabelBinarizer (class in dq0.sdk.pipeline.transformer.transformer), 86
MySQL (class in dq0.sdk.data.sql), 35
MySQL (class in dq0.sdk.data.sql.mysql), 38

N

NaiveBayesianModel (class in dq0.sdk.models.bayes), 73
NaiveBayesianModel (class in dq0.sdk.models.bayes.naive_bayesian_model), 74
NeuralNetwork (class in dq0.sdk.models.tf), 75
NeuralNetwork (class in dq0.sdk.models.tf.neural_network), 78
NeuralNetworkBase (class in dq0.sdk.estimators.tf.keras_base), 59
NeuralNetworkClassification (class in dq0.sdk.models.tf), 76
NeuralNetworkClassification (class in dq0.sdk.models.tf.neural_network_classification), 79
NeuralNetworkMultiClassClassification (class in dq0.sdk.models.tf), 76
NeuralNetworkMultiClassClassification (class in dq0.sdk.models.tf.neural_network_multiclass_classification), 80
NeuralNetworkRegression (class in dq0.sdk.models.tf), 76
NeuralNetworkRegression (class in dq0.sdk.models.tf.neural_network_regression), 80
NeuralNetworkYaml (class in dq0.sdk.models.tf), 76
NeuralNetworkYaml (class in dq0.sdk.models.tf.neural_network_yaml), 80
NN_Classifier (class in dq0.sdk.estimators.tf.keras_base), 58
NN_Regressor (class in dq0.sdk.estimators.tf.keras_base), 58
Normalizer (class in dq0.sdk.pipeline.transformer.transformer), 86
numerical_datasets_are_equal() (in module dq0.sdk.data.utils.util), 50
NuSVC (class in dq0.sdk.estimators.SVM.sklearn_svm), 54
NuSVR (class in dq0.sdk.estimators.SVM.sklearn_svm), 55

O

ODF (class in dq0.sdk.data.binary), 24
ODF (class in dq0.sdk.data.binary.odf), 26
OneClassSVM (class in dq0.sdk.estimators.SVM.sklearn_svm), 55
OneHotEncoder (class in dq0.sdk.pipeline.transformer.transformer), 86
Oracle (class in dq0.sdk.data.sql), 35
Oracle (class in dq0.sdk.data.sql.oracle), 39
ORC (class in dq0.sdk.data.binary), 24
ORC (class in dq0.sdk.data.binary.orc), 27
OrdinalEncoder (class in dq0.sdk.pipeline.transformer.transformer), 86

P

Parquet (class in dq0.sdk.data.binary), 24
Parquet (class in dq0.sdk.data.binary.parquet), 27
parse_kwargs() (in module dq0.sdk.utils.parse_args), 89
parse_value() (in module dq0.sdk.utils.parse_args), 89
perform_stratified_random_sampling() (in module dq0.sdk.data.utils.util), 50
Pipeline (class in dq0.sdk.pipeline.pipeline), 87
PipelineConfig (class in dq0.sdk.pipeline.pipeline_config), 87
plot_bars() (in module dq0.sdk.data.utils.plotting), 44
plot_confusion_matrix() (in module dq0.sdk.data.utils.plotting), 44
plot_confusion_matrix_for_scikit_classifier() (in module dq0.sdk.data.utils.plotting), 44
plot_decision_tree() (in module dq0.sdk.data.utils.plotting), 45
plot_hist() (in module dq0.sdk.data.utils.plotting), 45
PolynomialFeatures (class in dq0.sdk.pipeline.transformer.transformer), 86
post() (dq0.sdk.cli.api.Client method), 7
post() (dq0.sdk.cli.api.client.Client method), 8
PostgreSQL (class in dq0.sdk.data.sql), 36
PostgreSQL (class in dq0.sdk.data.sql.postgresql), 39
PowerTransformer (class in dq0.sdk.pipeline.transformer.transformer), 86
predict() (dq0.sdk.cli.Model method), 2
predict() (dq0.sdk.cli.model.Model method), 18
predict() (dq0.sdk.estimators.base_mixin.ClassifierMixin method), 60
predict() (dq0.sdk.estimators.base_mixin.RegressorMixin method), 60
predict() (dq0.sdk.estimators.tf.keras_base.NN_Classifier method), 58
predict() (dq0.sdk.estimators.tf.keras_base.NN_Regressor method), 58
predict() (dq0.sdk.models.tf.neural_network.NeuralNetwork method), 79
predict() (dq0.sdk.models.tf.neural_network_yaml.NeuralNetworkYaml method), 80
predict() (dq0.sdk.models.tf.NeuralNetwork method), 76
predict() (dq0.sdk.models.tf.NeuralNetworkYaml method), 77
predict_proba() (dq0.sdk.estimators.base_mixin.ClassifierMixin method), 60
predict_proba() (dq0.sdk.estimators.base_mixin.RegressorMixin method), 61
predict_proba() (dq0.sdk.estimators.tf.keras_base.NN_Classifier method), 58
predict_proba() (dq0.sdk.estimators.tf.keras_base.NN_Regressor method), 59
preprocess() (dq0.sdk.cli.Experiment method), 2
preprocess() (dq0.sdk.cli.experiment.Experiment method), 17
preprocess() (dq0.sdk.examples.census.bayesian.model.user_model.UserModel method), 62
preprocess() (dq0.sdk.examples.census.raw.model.user_model.UserModel method), 63
preprocess() (dq0.sdk.examples.har.model.user_model.UserModel method), 66
preprocess() (dq0.sdk.models.Model method), 72
preprocess() (dq0.sdk.models.model.Model method), 84
preprocess() (dq0.sdk.models.user.user_model.UserModel method), 83
preprocess() (dq0.sdk.models.user.UserModel method), 82
preprocess_dataset() (in module dq0.sdk.examples.census.eda), 64
pretty_display_string_on_terminal() (in module dq0.sdk.data.utils.util), 50
pretty_print_dict() (in module dq0.sdk.data.utils.util), 50
pretty_print_strings_list() (in module dq0.sdk.data.utils.util), 50
print_dataset_info() (in module dq0.sdk.data.utils.util), 50
print_details_about_df_columns() (in module dq0.sdk.data.utils.util), 50
print_evaluation_res() (in module dq0.sdk.data.utils.util), 50
print_full_df() (in module dq0.sdk.data.utils.util), 51
print_human_readable_elapsed_time_value() (in module dq0.sdk.data.utils.util), 51
print_summary_stats() (in module dq0.sdk.data.utils.util), 51
Project (class in dq0.sdk.cli), 3
Project (class in dq0.sdk.cli.project), 19
Project (class in dq0.sdk.projects), 88
Project (class in dq0.sdk.projects.project), 88

Q

QuantileTransformer (class in dq0.sdk.pipeline.transformer.transformer), 86
Query (class in dq0.sdk.cli), 6
Query (class in dq0.sdk.cli.query), 21
query() (dq0.sdk.cli.Data method), 6
query() (dq0.sdk.cli.data.Data method), 16
QueryRunner (class in dq0.sdk.cli.runner), 10
QueryRunner (class in dq0.sdk.cli.runner.query_runner), 12

R

RandomForestClassifier (class in dq0.sdk.estimators.ensemble.sklearn_ensemble), 57
RandomForestRegressor (class in dq0.sdk.estimators.ensemble.sklearn_ensemble), 57
read() (dq0.sdk.data.binary.Excel method), 23
read() (dq0.sdk.data.binary.excel.Excel method), 25
read() (dq0.sdk.data.binary.Feather method), 23
read() (dq0.sdk.data.binary.feather.Feather method), 26
read() (dq0.sdk.data.binary.HDF5 method), 23
read() (dq0.sdk.data.binary.hdf5.HDF5 method), 26
read() (dq0.sdk.data.binary.ORC method), 24
read() (dq0.sdk.data.binary.orc.ORC method), 27
read() (dq0.sdk.data.binary.Parquet method), 24
read() (dq0.sdk.data.binary.parquet.Parquet method), 27
read() (dq0.sdk.data.binary.SAS method), 24
read() (dq0.sdk.data.binary.sas.SAS method), 28
read() (dq0.sdk.data.binary.SPSS method), 25
read() (dq0.sdk.data.binary.spss.SPSS method), 28
read() (dq0.sdk.data.binary.Stata method), 25
read() (dq0.sdk.data.binary.stata.Stata method), 28
read() (dq0.sdk.data.image.Image method), 29
read() (dq0.sdk.data.image.image.Image method), 29
read() (dq0.sdk.data.Source method), 22
read() (dq0.sdk.data.source.Source method), 52
read() (dq0.sdk.data.sql.sql.SQL method), 41
read() (dq0.sdk.data.text.CSV method), 42
read() (dq0.sdk.data.text.csv.CSV method), 43
read() (dq0.sdk.data.text.JSON method), 42
read() (dq0.sdk.data.text.json.JSON method), 43
read_from_yaml() (dq0.sdk.data.metadata.Metadata method), 30
read_from_yaml() (dq0.sdk.data.metadata.metadata.Metadata method), 31
read_from_yaml() (dq0.sdk.pipeline.pipeline_config.PipelineConfig method), 88
read_from_yaml_file() (dq0.sdk.data.metadata.Metadata method), 30
read_from_yaml_file() (dq0.sdk.data.metadata.metadata.Metadata method), 31
read_from_yaml_file() (dq0.sdk.pipeline.pipeline_config.PipelineConfig method), 88
read_yaml_file() (dq0.sdk.utils.yaml_config.YamlConfig method), 90
read_yaml_file() (dq0.sdk.utils.YamlConfig method), 89
redirect_stdout_stderr_streams_to_file() (in module dq0.sdk.data.utils.util), 51
Redshift (class in dq0.sdk.data.sql), 36
Redshift (class in dq0.sdk.data.sql.redshift), 40
refresh() (dq0.sdk.cli.Data method), 6
refresh() (dq0.sdk.cli.data.Data method), 16
register() (dq0.sdk.cli.Model method), 2
register() (dq0.sdk.cli.model.Model method), 18
RegressorMixin (class in dq0.sdk.estimators.base_mixin), 60
replace_function() (in module dq0.sdk.cli.utils.code), 15
replace_model_parent_class() (in module dq0.sdk.cli.utils.code), 15
restore_stdout_stderr_streams() (in module dq0.sdk.data.utils.util), 51
Ridge (class in dq0.sdk.estimators.linear_model.sklearn_lm), 58
RidgeClassifier (class in dq0.sdk.estimators.linear_model.sklearn_lm), 58
RobustScaler (class in dq0.sdk.pipeline.transformer.transformer), 86
run() (dq0.sdk.cli.Experiment method), 2
run() (dq0.sdk.cli.experiment.Experiment method), 17
run_all() (dq0.sdk.models.tf.neural_network_yaml.NeuralNetworkYaml method), 81
run_all() (dq0.sdk.models.tf.NeuralNetworkYaml method), 77
Runner (class in dq0.sdk.cli.runner.runner), 13

S

SAPHana (class in dq0.sdk.data.sql), 36
SAPHana (class in dq0.sdk.data.sql.sap_hana), 40
SAS (class in dq0.sdk.data.binary), 24
SAS (class in dq0.sdk.data.binary.sas), 27
save() (dq0.sdk.models.bayes.naive_bayesian_model.NaiveBayesianModel method), 75
save() (dq0.sdk.models.bayes.NaiveBayesianModel method), 74
save() (dq0.sdk.models.Model method), 73
save() (dq0.sdk.models.model.Model method), 84
save() (dq0.sdk.models.tf.neural_network.NeuralNetwork method), 79
save() (dq0.sdk.models.tf.neural_network_yaml.NeuralNetworkYaml method), 81
save() (dq0.sdk.models.tf.NeuralNetwork method), 76
save() (dq0.sdk.models.tf.NeuralNetworkYaml method), 77
save() (dq0.sdk.models.user.user_model.UserModel method), 83

save() (dq0.sdk.models.user.UserModel method), 82
save_figure() (in module dq0.sdk.data.utils.plotting), 45
save_preprocessed_tr_and_te_datasets() (in module dq0.sdk.data.utils.util), 51
save_yaml() (dq0.sdk.utils.yaml_config.YamlConfig method), 90
save_yaml() (dq0.sdk.utils.YamlConfig method), 89
scale_pixels() (in module dq0.sdk.data.preprocessing.preprocessing), 33
scatterplot() (in module dq0.sdk.data.utils.plotting), 45
Schema (class in dq0.sdk.data.metadata.metadata), 32
score() (dq0.sdk.estimators.base_mixin.ClassifierMixin method), 60
score() (dq0.sdk.estimators.base_mixin.RegressorMixin method), 61
select_bar_colors() (in module dq0.sdk.data.utils.plotting), 45
set_connection() (dq0.sdk.cli.api.Client method), 7
set_connection() (dq0.sdk.cli.api.client.Client method), 8
set_connection() (dq0.sdk.cli.Project method), 4
set_connection() (dq0.sdk.cli.project.Project method), 20
set_model_code() (dq0.sdk.cli.Project method), 4
set_model_code() (dq0.sdk.cli.project.Project method), 21
set_results() (dq0.sdk.cli.runner.state.State method), 14
setup_data() (dq0.sdk.estimators.data_handler.base.BasicDataHandler method), 55
setup_data() (dq0.sdk.estimators.data_handler.csv.CSVDataHandler method), 55
setup_data() (dq0.sdk.estimators.estimator.Estimator method), 61
setup_data() (dq0.sdk.estimators.tf.keras_base.NeuralNetworkBase method), 59
setup_data() (dq0.sdk.examples.census.bayesian.model.user_model.UserModel method), 62
setup_data() (dq0.sdk.examples.census.preprocessed.model.user_model.UserModel method), 63
setup_data() (dq0.sdk.examples.census.raw.model.user_model.UserModel method), 64
setup_data() (dq0.sdk.examples.cifar.model.user_model.UserModel method), 65
setup_data() (dq0.sdk.examples.cifar.model.user_model_old.UserModel method), 65
setup_data() (dq0.sdk.examples.har.model.user_model.UserModel method), 66
setup_data() (dq0.sdk.examples.medical_insurance.model.user_model.UserModel method), 67
setup_data() (dq0.sdk.examples.newsgroups.bayesian.model.user_model.UserModel method), 68
setup_data() (dq0.sdk.examples.newsgroups.bayesian.model.user_model_old_needs_SOURCE.Use method), 68
setup_data() (dq0.sdk.examples.newsgroups.network.model.user_model.UserModel method), 69
setup_data() (dq0.sdk.examples.newsgroups.network.model.user_model_from_preprocessed_data method), 69
setup_data() (dq0.sdk.examples.newsgroups.network.model.user_model_old_needs_SOURCE.Use method), 70
setup_data() (dq0.sdk.examples.patient.model.user_model.UserModel method), 70
setup_data() (dq0.sdk.examples.pneumonia.my_model.UserModel method), 71
setup_data() (dq0.sdk.examples.pneumonia.my_model_int_encoding.UserModel method), 72
setup_data() (dq0.sdk.models.Model method), 73
setup_data() (dq0.sdk.models.model.Model method), 84
setup_data() (dq0.sdk.models.tf.neural_network_yaml.NeuralNetworkYaml method), 81
setup_data() (dq0.sdk.models.tf.NeuralNetworkYaml method), 77
setup_data() (dq0.sdk.models.tf.tf_hub.TFHub method), 81
setup_data() (dq0.sdk.models.tf.TFHub method), 77
setup_data() (dq0.sdk.models.user.user_model.UserModel method), 83
setup_data() (dq0.sdk.models.user.UserModel method), 82
setup_model() (dq0.sdk.estimators.estimator.Estimator method), 61
setup_model() (dq0.sdk.estimators.tf.keras_dense_classifier.Keras_Dense_Classifier_Binary method), 59
setup_model() (dq0.sdk.estimators.tf.keras_dense_classifier.Keras_Dense_Classifier_Integer method), 59
setup_model() (dq0.sdk.estimators.tf.keras_dense_classifier.Keras_Dense_Classifier_OHE method), 60
setup_model() (dq0.sdk.estimators.tf.keras_dense_regressor.Keras_Dense_Regressor

method), 60
setup_model() (dq0.sdk.examples.census.bayesian.model.user_model.UserModel method), 62
setup_model() (dq0.sdk.examples.census.preprocessed.model.user_model.UserModel method), 63
setup_model() (dq0.sdk.examples.census.raw.model.user_model.UserModel method), 64
setup_model() (dq0.sdk.examples.cifar.model.user_model.UserModel method), 65
setup_model() (dq0.sdk.examples.cifar.model.user_model_old.UserModel method), 65
setup_model() (dq0.sdk.examples.har.model.user_model.UserModel method), 67
setup_model() (dq0.sdk.examples.medical_insurance.model.user_model.UserModel method), 67
setup_model() (dq0.sdk.examples.newsgroups.bayesian.model.user_model.UserModel method), 68
setup_model() (dq0.sdk.examples.newsgroups.bayesian.model.user_model_old_needs_SOURCE.UserModel method), 68
setup_model() (dq0.sdk.examples.newsgroups.network.model.user_model.UserModel method), 69
setup_model() (dq0.sdk.examples.newsgroups.network.model.user_model_from_preprocessed_data.UserModel method), 69
setup_model() (dq0.sdk.examples.newsgroups.network.model.user_model_old_needs_SOURCE.UserModel method), 70
setup_model() (dq0.sdk.examples.patient.model.user_model.UserModel method), 70
setup_model() (dq0.sdk.examples.pneumonia.my_model.UserModel method), 71
setup_model() (dq0.sdk.examples.pneumonia.my_model_int_encoding.UserModel method), 72
setup_model() (dq0.sdk.models.Model method), 73
setup_model() (dq0.sdk.models.model.Model method), 85
setup_model() (dq0.sdk.models.tf.neural_network_yaml.NeuralNetworkYaml method), 81
setup_model() (dq0.sdk.models.tf.NeuralNetworkYaml method), 77
setup_model() (dq0.sdk.models.user.user_model.UserModel method), 83
setup_model() (dq0.sdk.models.user.UserModel method), 82
Snowflake (class in dq0.sdk.data.sql), 36
Snowflake (class in dq0.sdk.data.sql.snowflake), 40
Source (class in dq0.sdk.data), 22
Source (class in dq0.sdk.data.source), 52
sparse_scipy_matrix_to_Pandas_df() (in module dq0.sdk.data.utils.util), 51
SPSS (class in dq0.sdk.data.binary), 25
SPSS (class in dq0.sdk.data.binary.spss), 28
SQL (class in dq0.sdk.data.sql.sql), 41
SQLite (class in dq0.sdk.data.sql), 37
SQLite (class in dq0.sdk.data.sql.sqlite), 42
StandardScaler (class in dq0.sdk.pipeline.transformer.transformer), 86
Stata (class in dq0.sdk.data.binary), 25
Stata (class in dq0.sdk.data.binary.stata), 28
State (class in dq0.sdk.cli.runner.state), 14
str_to_bool() (in module dq0.sdk.data.utils.util), 51
string_contains_numeric_value() (in module dq0.sdk.data.utils.util), 51
SVC (class in dq0.sdk.estimators.SVM.sklearn_svm), 55
SVR (class in dq0.sdk.estimators.SVM.sklearn_svm), 55

T

Table (class in dq0.sdk.data.metadata.metadata), 32
tensorflow_tensor_to_numpy_ndarray() (in module dq0.sdk.data.utils.util), 51
TFHub (class in dq0.sdk.models.tf), 77
TFHub (class in dq0.sdk.models.tf.tf_hub), 81
to_dict() (dq0.sdk.data.metadata.Metadata method), 30
to_dict() (dq0.sdk.data.metadata.metadata.Column method), 31
to_dict() (dq0.sdk.data.metadata.metadata.Metadata method), 32
to_dict() (dq0.sdk.data.metadata.metadata.Schema method), 32
to_dict() (dq0.sdk.data.metadata.metadata.Table method), 32
to_dict_sm() (dq0.sdk.data.metadata.Metadata method), 30
to_dict_sm() (dq0.sdk.data.metadata.metadata.Metadata method), 32
to_json() (dq0.sdk.data.image.Image method), 29
to_json() (dq0.sdk.data.image.image.Image method), 29
to_json() (dq0.sdk.data.Source method), 22
to_json() (dq0.sdk.data.source.Source method), 52
to_json() (dq0.sdk.data.sql.sql.SQL method), 41
to_string() (dq0.sdk.models.bayes.naive_bayesian_model.NaiveBayesianModel method), 75
to_string() (dq0.sdk.models.bayes.NaiveBayesianModel method), 74

to_string() (dq0.sdk.models.Model method), 73
to_string() (dq0.sdk.models.model.Model method), 85
to_string() (dq0.sdk.models.tf.neural_network_classification.NeuralNetworkClassification method), 79
to_string() (dq0.sdk.models.tf.neural_network_multiclass_classification.NeuralNetworkMultiClassClassification method), 80
to_string() (dq0.sdk.models.tf.neural_network_regression.NeuralNetworkRegression method), 80
to_string() (dq0.sdk.models.tf.neural_network_yaml.NeuralNetworkYaml method), 81
to_string() (dq0.sdk.models.tf.NeuralNetworkClassification method), 76
to_string() (dq0.sdk.models.tf.NeuralNetworkMultiClassClassification method), 76
to_string() (dq0.sdk.models.tf.NeuralNetworkRegression method), 76
to_string() (dq0.sdk.models.tf.NeuralNetworkYaml method), 77
to_yaml() (dq0.sdk.data.metadata.Metadata method), 30
to_yaml() (dq0.sdk.data.metadata.metadata.Metadata method), 32
to_yaml_file() (dq0.sdk.data.metadata.Metadata method), 30
to_yaml_file() (dq0.sdk.data.metadata.metadata.Metadata method), 32
train_test_split() (in module dq0.sdk.data.preprocessing.preprocessing), 33
train_test_split_reshape() (in module dq0.sdk.examples.pneumonia.data_preparation), 71
Transform (class in dq0.sdk.data), 22
Transform (class in dq0.sdk.data.transform), 53
transform() (dq0.sdk.pipeline.transformer.transformer.ColumnSelector method), 85
transform() (dq0.sdk.pipeline.transformer.transformer.Transformer method), 86
transform() (dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_1 method), 87
transform() (dq0.sdk.pipeline.transformer.transformer.Transformer_1_to_N method), 87
Transformer (class in dq0.sdk.pipeline.transformer.transformer), 86
Transformer_1_to_1 (class in dq0.sdk.pipeline.transformer.transformer), 86
Transformer_1_to_N (class in dq0.sdk.pipeline.transformer.transformer), 87
TransformerMixin (class in dq0.sdk.estimators.base_mixin), 61

U

univariate_feature_selection() (in module dq0.sdk.data.preprocessing.preprocessing), 34
update() (dq0.sdk.cli.runner.state.State method), 14
update_commit_uuid() (dq0.sdk.cli.Project method), 5
update_commit_uuid() (dq0.sdk.cli.project.Project method), 21
UserModel (class in dq0.sdk.examples.census.bayesian.model.user_model), 62
UserModel (class in dq0.sdk.examples.census.preprocessed.model.user_model), 62
UserModel (class in dq0.sdk.examples.census.raw.model.user_model), 63
UserModel (class in dq0.sdk.examples.cifar.model.user_model), 64
UserModel (class in dq0.sdk.examples.cifar.model.user_model_old), 65
UserModel (class in dq0.sdk.examples.har.model.user_model), 66
UserModel (class in dq0.sdk.examples.medical_insurance.model.user_model), 67
UserModel (class in dq0.sdk.examples.newsgroups.bayesian.model.user_model), 68
UserModel (class in dq0.sdk.examples.newsgroups.bayesian.model.user_model_old_needs_SOURC
68
UserModel (class in dq0.sdk.examples.newsgroups.network.model.user_model), 69
UserModel (class in dq0.sdk.examples.newsgroups.network.model.user_model_from_preprocessed
69
UserModel (class in dq0.sdk.examples.newsgroups.network.model.user_model_old_needs_SOURC
70
UserModel (class in dq0.sdk.examples.patient.model.user_model), 70
UserModel (class in dq0.sdk.examples.pneumonia.my_model), 71
UserModel (class in dq0.sdk.examples.pneumonia.my_model_int_encoding), 71
UserModel (class in dq0.sdk.models.user), 82
UserModel (class in dq0.sdk.models.user.user_model), 83

V

visualize_categorical_distribution() (in module dq0.sdk.data.utils.plotting), 46
visualize_continuous_distribution() (in module dq0.sdk.data.utils.plotting), 46
visualize_filtered_data() (in module dq0.sd-

k.examples.census.eda), 64

W

wait_for_completion() (dq0.sdk.cli.runner.runner.Runner method), 14
where() (dq0.sdk.cli.Data method), 6
where() (dq0.sdk.cli.data.Data method), 16

Y

YamlConfig (class in dq0.sdk.utils), 89
YamlConfig (class in dq0.sdk.utils.yaml_config), 90