# **DQ0 SDK Documentation**

Release 1.0.10.2

**Gradient Zero** 

Apr 19, 2021

### Table of Contents

I	Subp	ибраскаges					
	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				
	Pytho	on Module Index	91				
	Index		93				

## **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 DOO 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.

Copyright 2020, Gradient Zero All rights reserved

```
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'})
```

Copyright 2020, Gradient Zero All rights reserved

### 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.
            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.
        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

Copyright 2020, Gradient Zero All rights reserved

```
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:

**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

Copyright 2020, Gradient Zero All rights reserved

```
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:

### **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

Copyright 2020, Gradient Zero All rights reserved

```
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

Copyright 2020, Gradient Zero All rights reserved

```
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:**

Cancels the experiment run

```
get_error()
    get_results()
        Gets the results of the running model or data experiment.
             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 represents 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 informa-
tion.
State ID values provided by dq0-main:
       • 0 - StateCreated
       • 1 - StatePrepared
       • 2 - StateRunning
       • 3 - StateFailed
       • 4 - StateStopped
Copyright 2020, Gradient Zero All rights reserved
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 )
```

### dq0.sdk.cli.utils package

Utils package contains helper functions for the SDK API

Updates the state representation

```
Copyright 2020, Gradient Zero All rights reserved
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
Copyright 2020, Gradient Zero All rights reserved
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
Copyright 2020, Gradient Zero All rights reserved
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.
Copyright 2020, Gradient Zero All rights reserved
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
        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

Copyright 2020, Gradient Zero All rights reserved

```
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

Copyright 2020, Gradient Zero All rights reserved

```
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

### Copyright 2020, Gradient Zero All rights reserved

```
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.

Copyright 2020, Gradient Zero All rights reserved

```
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:**

```
{\tt dq0.sdk.cli.runner.QueryRunner} \ {\tt 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 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.
        path (str): Absolute path to the feather file.
    read ( **kwargs )
        Read feather data source
            kwargs: keyword arguments.
```

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.
        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 columns 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.
        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 columnn to read.
        Returns:
            parquet data as pandas dataframe
class dq0.sdk.data.binary.SAS ( path )
    Bases: dg0.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.
        path (str): Absolute path to the spss file.
    read ( **kwargs )
        Read spss data source
        Args:
             kwargs: keyword arguments.
             spss data as pandas dataframe
class dq0.sdk.data.binary.Stata ( path )
    Bases: dg0.sdk.data.source.Source
    Data Source for stata data.
    Provides function to read in stata data.
        path (str): Absolute path to the stata file.
    read ( **kwargs )
        Read stata data source
        Args:
             kwargs: keyword arguments.
             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.
Copyright 2020, Gradient Zero All rights reserved
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.
Copyright 2020, Gradient Zero All rights reserved
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.
Copyright 2020, Gradient Zero All rights reserved
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.
Copyright 2020, Gradient Zero All rights reserved
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.
        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.

Copyright 2020, Gradient Zero All rights reserved

```
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 column 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.

Copyright 2020, Gradient Zero All rights reserved

```
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 columns 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.

Copyright 2020, Gradient Zero All rights reserved

```
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.
Copyright 2020, Gradient Zero All rights reserved
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.
Copyright 2020, Gradient Zero All rights reserved
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.

Copyright 2020, Gradient Zero All rights reserved

```
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.

#### Aros:

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.

### Copyright 2020, Gradient Zero All rights reserved

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.

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

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 ( <math>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 dg0.sdk.data.sgl.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
             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'
        connection_string (str): The sqlite connection string.
    execute ( query, **kwargs )
        Execute SQL query
        Args:
             query: SQL Query to execute kwargs: keyword arguments
             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!
Copyright 2020, Gradient Zero All rights reserved
class dg0.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
Copyright 2020, Gradient Zero All rights reserved
```

```
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.
Copyright 2020, Gradient Zero All rights reserved
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>'
        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.
Copyright 2020, Gradient Zero All rights reserved
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>'
        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.
Copyright 2020, Gradient Zero All rights reserved
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
            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.
Copyright 2020, Gradient Zero All rights reserved
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'
        connection_string (str): The postgresql connection string.
    execute ( query, **kwargs )
        Execute the Postgres query
            query: SQL Query to execute kwargs: keyword arguments
            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/S-AP/sqlalchemy-hana

Copyright 2020, Gradient Zero All rights reserved

```
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:
```

dq0.sdk.data.sql.sap hana module

Data Source for SAP Hana.

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

SQL ResultSet as pandas dataframe

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

Copyright 2020, Gradient Zero All rights reserved

```
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/snowflaked-b/snowflake-sqlalchemy

Copyright 2020, Gradient Zero All rights reserved

```
class dq0.sdk.data.sql.snowflake.Snowflake(connection_string)
Bases: dq0.sdk.data.sql.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.
Copyright 2020, Gradient Zero All rights reserved
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 connec-
        tion 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 execute SQL queries
             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.
             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.
Copyright 2020, Gradient Zero All rights reserved
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
```

```
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.
Copyright 2020, Gradient Zero All rights reserved
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.
Copyright 2020, Gradient Zero All rights reserved
class dq0.sdk.data.text.json.JSON ( path )
    Bases: dq0.sdk.data.source.Source
    Data Source for ISON 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. Copyright 2020, Gradient Zero All rights reserved

Raises:

```
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.

Copyright 2020, Gradient Zero All rights reserved

```
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

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.
```

#### Aros:

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.

```
\verb|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
{\tt 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.
        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_pre-
    fix')
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 float-
    ing-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 float-
              ing-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

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.

Copyright 2020, Gradient Zero All rights reserved

```
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

Copyright 2020, Gradient Zero All rights reserved

```
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 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 ison.

## dq0.sdk.data.transform module

Data Transform class.

## Copyright 2020, Gradient Zero

## 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.

Copyright 2021, Gradient Zero All rights reserved

```
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.

Copyright 2021, Gradient Zero All rights reserved

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
```

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)

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)
                                     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',
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

dg0.sdk.estimators.ensemble.sklearn ensemble module

Sklearn ensemble models.

Copyright 2021, Gradient Zero All rights reserved

Bases:  $dq0.sdk.estimators.base\_mixin.ClassifierMixin, \\dq0.sdk.estimators.estimator.Estimator$ 

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

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

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

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 Im module

Diffprivlib linear models.

Copyright 2021, Gradient Zero All rights reserved

 $\begin{tabular}{ll} $class$ & dq0.sdk.estimators.linear_model.diffprivlib_lm. LinearRegressionDP & (arget_epsilon=1, bounds_X=None, bounds_y=None, fit_intercept=True, copy_X=True, accountant=None, **kwargs*) \\ \end{tabular}$ 

Bases:  $dq0.sdk.estimators.base\_mixin.RegressorMixin, \\ dq0.sdk.estimators.estimator.Estimator$ 

Diffprivlib linear regression

 $\begin{array}{lll} \textbf{class} & \texttt{dq0.sdk.estimators.linear\_model.diffprivlib\_lm.LogisticRegressionDP} & ( & \textit{target\_epsilon=1}, & \textit{data\_norm=None}, & \textit{tol=0.0001}, & \textit{C=1.0}, & \textit{fit\_intercept=True}, & \textit{max\_iter=100}, & \textit{verbose=0}, \\ & \textit{warm\_start=False}, & \textit{n\_jobs=None}, & \textit{accountant=None}, & **kwargs \\ ) \end{array}$ 

Bases: dq0.sdk.estimators.base\_mixin.ClassifierMixin, dq0.sdk.estimators.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 )
                                     dq0.sdk.estimators.base_mixin.RegressorMixin,
    dq0.sdk.estimators.estimator.Estimator
        dq0.sdk.estimators.linear_model.sklearn_lm.LinearRegression
fit_intercept=True, normalize=False, copy_X=True, n_jobs=None, positive=False, **kwargs)
                                     dq0.sdk.estimators.base_mixin.RegressorMixin,
    dq0.sdk.estimators.estimator.Estimator
    Sklearn linear regression wrapper
class
          dq0.sdk.estimators.linear model.sklearn lm.LoqisticRegression
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)
                                     dg0.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 )
                                   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
Copyright 2021, Gradient Zero All rights reserved
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)
       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.
Copyright 2021, Gradient Zero All rights reserved
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.k-
    eras.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>, metric-
s=['accuracy', 'mae'], batch_size=250, epochs=2, n_layers=[10, 10], **kwargs)
                                    dq0.sdk.estimators.tf.keras_base.NN_Classifier,
    Bases:
    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=<ten-
    sorflow.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=['accura-
cy', 'mae'], batch_size=250, epochs=2, n_layers=[10, 10], **kwargs)
                                    dq0.sdk.estimators.tf.keras_base.NN_Classifier,
    dq0.sdk.estimators.base_mixin.ClassifierMixin,
```

1.4. dq0.sdk.estimators package

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.

Copyright 2021, Gradient Zero All rights reserved

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.k-eras.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

Copyright 2021, Gradient Zero All rights reserved

```
\textbf{class} \; \texttt{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 DO0

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.

Copyright 2021, Gradient Zero All rights reserved

```
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.

Copyright 2020, Gradient Zero All rights reserved

```
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.

Copyright 2020, Gradient Zero All rights reserved

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 state
>>> ../dq0 model predict --input-path </path/to/numpy.npy>
>>> ../dq0 model state
```

## Copyright 2020, Gradient Zero All rights reserved

```
class dq0.sdk.examples.census.preprocessed.model.user_model.UserModel
```

## Bases:

```
\verb|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.

Copyright 2020, Gradient Zero All rights reserved

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 state
>>> ../dq0 model predict --input-path </path/to/numpy.npy>
>>> ../dq0 model state
```

## Copyright 2020, Gradient Zero All rights reserved

```
class dq0.sdk.examples.census.raw.model.user_model.UserModel
```

## Bases:

```
\verb|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.

Copyright 2020, Gradient Zero All rights reserved

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.

Copyright 2020, Gradient Zero All rights reserved

```
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.

Copyright 2020, Gradient Zero All rights reserved

```
{\bf class} \; {\tt 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.

Copyright 2020, Gradient Zero All rights reserved

```
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.

Copyright 2020, Gradient Zero All rights reserved

## 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
```

## Copyright 2020, Gradient Zero All rights reserved

```
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.

Copyright 2020, Gradient Zero All rights reserved

## 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  $^{\wedge}$  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.

Copyright 2020, Gradient Zero

```
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.

Copyright 2020, Gradient Zero All rights reserved

## 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.

Copyright 2020, Gradient Zero All rights reserved

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"

Copyright 2020, Gradient Zero All rights reserved

 ${\bf class}~{\tt dq0.sdk.examples.newsgroups.network.model.user\_model.{\tt UserModel}$ 

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.

 $dq0.sdk. examples. news groups. network. model. user\_model\_from\_preprocessed\_data\ module$ 

Convolutional Neural Network model implementation for "20 Newsgroups"

Copyright 2020, Gradient Zero All rights reserved

#### class

dq0.sdk.examples.newsgroups.network.model.user\_model\_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

 ${\tt dq0.sdk.models.tf.neural\_network\_classification.NeuralNetworkClassification}$ 

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.

Copyright 2020, Gradient Zero All rights reserved

dq0.sdk.examples.pneumonia.data\_preparation.train\_test\_split\_reshape

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.

Copyright 2020, Gradient Zero All rights reserved

```
class dq0.sdk.examples.pneumonia.my_model.UserModel
```

Bases:

 ${\tt dq0.sdk.models.tf.neural\_network\_classification.NeuralNetworkClassification} \\ {\tt CNN~with~pre-training}$ 

```
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.

Copyright 2020, Gradient Zero All rights reserved

```
class dq0.sdk.examples.pneumonia.my_model_int_encoding.UserModel
    Bases:
```

dq0.sdk.models.tf.neural\_network\_classification.NeuralNetworkClassification

df,

```
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.
             path (str): The model path
    save ( path )
        Saves the model.
        Save the model in binary format on local storage.
             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
Copyright 2020, Gradient Zero All rights reserved
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.
             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: dg0.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\_k-wargs: 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()
```

## Copyright 2020, Gradient Zero All rights reserved

```
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.

```
\verb"evaluate" ( \textit{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:**

Х, у

dq0.sdk.models.tf.neural network classification module

Neural Network multi class classification model

Copyright 2020, Gradient Zero All rights reserved

# 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

Copyright 2020, Gradient Zero All rights reserved

```
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

Copyright 2020, Gradient Zero All rights reserved

```
\textbf{class} \; \texttt{dq0.sdk.models.tf.neural\_network\_regression.} \\ \textbf{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

Copyright 2020, Gradient Zero All rights reserved

```
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

Copyright 2020, Gradient Zero All rights reserved

```
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\_k-wargs: 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

Copyright 2020, Gradient Zero All rights reserved

```
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.
             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
        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.
             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.

Copyright 2020, Gradient Zero All rights reserved

```
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

Copyright 2021, Gradient Zero All rights reserved

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 )

 $\pmb{Bases:} \ \texttt{dq0.sdk.pipeline.transformer.transformer.Transformer\_1\_to\_N$ 

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='12', *, 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: dg0.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.
            dq0.sdk.pipeline.transformer.transformer.PowerTransformer
                                                                                      (
method='yeo-johnson', *, standardize=True, copy=True, **kwargs )
   Bases: dq0.sdk.pipeline.transformer.transformer.Transformer 1 to 1
        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
            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
                                                         (
                                                                              config_path=None,
         dq0.sdk.pipeline.pipeline.Pipeline
                                                              steps=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.
                                              root_dir='./dq0/sdk/pipeline/transformer/transformer.py',
    get_steps_from_config
    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.

Copyright 2020, Gradient Zero All rights reserved

```
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

Copyright 2020, Gradient Zero All rights reserved

## dq0.sdk.utils.parse\_args module

```
Copyright 2021, Gradient Zero All rights reserved
```

```
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()
```

Copyright 2020, Gradient Zero All rights reserved

```
class dq0.sdk.utils.yaml_config.YamlConfig ( yaml_path, yaml_dict=None, custom_object-
s={'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
```

```
dq0.sdk.data.preprocessing,33
d
                                             dg0.sdk.data.preprocessing.preprocessing,
da0
                                             dq0.sdk.data.source,52
   dq0.sdk, XX
                                             dq0.sdk.data.sql,34
   dq0.sdk.cli,1
                                             dq0.sdk.data.sql.big_query,37
   dq0.sdk.cli.api,7
                                             dq0.sdk.data.sql.drill,37
   dq0.sdk.cli.api.client,8
                                             dq0.sdk.data.sql.mssql,38
   dq0.sdk.cli.api.routes,9
                                             dq0.sdk.data.sql.mysql,38
   dq0.sdk.cli.data,15
                                             dq0.sdk.data.sql.oracle,39
   dq0.sdk.cli.experiment,16
                                             dq0.sdk.data.sql.postgresql,39
   dq0.sdk.cli.model,17
                                             dq0.sdk.data.sql.redshift,39
   dq0.sdk.cli.project,19
                                             dq0.sdk.data.sql.sap hana,40
   dq0.sdk.cli.query,21
                                             dq0.sdk.data.sql.snowflake,40
   dq0.sdk.cli.runner,9
                                             dq0.sdk.data.sql.sql,41
   dq0.sdk.cli.runner.data_runner,
                                             dq0.sdk.data.sql.sqlite,42
          10
                                             dq0.sdk.data.text,42
   dq0.sdk.cli.runner.model_runner,
                                             dq0.sdk.data.text.csv,43
                                             dq0.sdk.data.text.json,43
   dq0.sdk.cli.runner.query_runner,
                                             dq0.sdk.data.transform, 52
                                             dq0.sdk.data.utils,43
   dq0.sdk.cli.runner.runner,13
                                             dq0.sdk.data.utils.plotting,43
   dq0.sdk.cli.runner.state,14
                                             dq0.sdk.data.utils.util,46
   dq0.sdk.cli.utils,14
                                             dq0.sdk.errors,53
   dq0.sdk.cli.utils.code, 15
                                             dq0.sdk.errors.errors,53
   dq0.sdk.cli.utils.uuid,15
                                             dq0.sdk.estimators,54
   dq0.sdk.data,22
                                             dq0.sdk.estimators.base_mixin,60
   dq0.sdk.data.binary,23
                                             dq0.sdk.estimators.data_handler,
   dq0.sdk.data.binary.excel,25
   dq0.sdk.data.binary.feather,26
                                             dq0.sdk.estimators.data_handler.base,
   dq0.sdk.data.binary.hdf5,26
   dq0.sdk.data.binary.odf,26
                                             dq0.sdk.estimators.data_handler.csv,
   dq0.sdk.data.binary.orc, 27
                                                    55
   dq0.sdk.data.binary.parquet,27
                                             dq0.sdk.estimators.data_handler.utils,
   dq0.sdk.data.binary.sas,27
   dq0.sdk.data.binary.spss,28
                                             dq0.sdk.estimators.ensemble, 56
   dq0.sdk.data.binary.stata,28
                                             dq0.sdk.estimators.ensemble.sklearn_ensemble
   dq0.sdk.data.data_source_factory,
          51
                                             dq0.sdk.estimators.estimator,61
   dq0.sdk.data.image,28
                                             dq0.sdk.estimators.linear_model,
   dq0.sdk.data.image.image, 29
   dq0.sdk.data.metadata,29
                                             dq0.sdk.estimators.linear_model.diffprivlib_
   dq0.sdk.data.metadata.metadata,
```

30

```
dq0.sdk.estimators.linear_model.sklearn_lm,
                                         dq0.sdk.examples.newsgroups.network.model.us
dq0.sdk.estimators.SVM, 54
                                         dq0.sdk.examples.newsgroups.network.model.us
dq0.sdk.estimators.SVM.sklearn_svm,
                                         dq0.sdk.examples.newsgroups.network.model.us
dq0.sdk.estimators.tf,58
dq0.sdk.estimators.tf.keras base,
                                         dq0.sdk.examples.newsgroups.network.run_demo
                                               70
dq0.sdk.estimators.tf.keras_dense_classifier,
                                         dq0.sdk.examples.patient,70
\verb|dq0.sdk.estimators.tf.keras_dense_regres \verb|dq0.sdk.examples.patient.model.user_model|,
      60
                                         dq0.sdk.examples.pneumonia.data_preparation,
dq0.sdk.examples,61
dq0.sdk.examples.census,61
dq0.sdk.examples.census.bayesian,
                                         dq0.sdk.examples.pneumonia.my_model,
dq0.sdk.examples.census.bayesian.model.udq0.adkeexamples.pneumonia.my_model_int_enco
dq0.sdk.examples.census.bayesian.run_demdq0.sdk.models,72
                                         dq0.sdk.models.bayes,73
dq0.sdk.examples.census.eda,64
                                         dq0.sdk.models.bayes.naive_bayesian_model,
dq0.sdk.examples.census.preprocessed,
                                         dq0.sdk.models.model,84
dq0.sdk.examples.census.preprocessed.modeq0usek_models.tf,75
                                         dq0.sdk.models.tf.neural_network,
dq0.sdk.examples.census.preprocessed.run_demo, 78
                                         dq0.sdk.models.tf.neural_network_classificat
      63
dq0.sdk.examples.census.raw,63
                                                79
dq0.sdk.examples.census.raw.model.user_mdq@lsdk.models.tf.neural_network_multiclass_
                                         dq0.sdk.models.tf.neural_network_regression,
dq0.sdk.examples.census.raw.run_demo,
      64
dq0.sdk.examples.cifar,64
                                         dq0.sdk.models.tf.neural_network_yaml,
                                                80
dq0.sdk.examples.cifar.model.user_model,
                                         dq0.sdk.models.tf.tf_hub,81
dq0.sdk.examples.cifar.model.user_model_dqa,sdk.models.user,82
                                         dq0.sdk.models.user.user_model,
dq0.sdk.examples.cifar.run_demo,
                                         dq0.sdk.models.yaml_configs,83
                                         dq0.sdk.models.yaml_configs.tf_hub_models,
dq0.sdk.examples.har.model.user_model,
                                         dq0.sdk.pipeline,85
dq0.sdk.examples.har.run_demo,67
dq0.sdk.examples.medical_insurance,
                                         dq0.sdk.pipeline.pipeline,87
                                         dq0.sdk.pipeline.pipeline_config,
dq0.sdk.examples.medical_insurance.model.user_model,
                                         dq0.sdk.pipeline.transformer.transformer,
dq0.sdk.examples.medical_insurance.run_demo,
                                         dq0.sdk.projects,88
                                         dq0.sdk.projects.project,88
dq0.sdk.examples.newsgroups,68
                                         dq0.sdk.utils,89
dq0.sdk.examples.newsgroups.bayesian,
                                         dq0.sdk.utils.managed_classes,89
dq0.sdk.examples.newsgroups.bayesian.modeq0usek.mbdes,parse_args,89
                                         dq0.sdk.utils.yaml_config,89
dq0.sdk.examples.newsgroups.bayesian.model.user_model_old_needs_SOURCE,
dq0.sdk.examples.newsgroups.bayesian.run_demo,
dq0.sdk.examples.newsgroups.network,
```

92 Python Module Index

	method), 9
A	cancel() (dq0.sdk.cli.runner.model_runner
AdaBoostClassifier (class in dq0.sdk.estima-	ModelRunner method), 12
tors.ensemble.sklearn_ensemble), 56	cancel() (dq0.sdk.cli.runner.ModelRunner
AdaBoostRegressor (class in dq0.sdk.estima-	method), 10
tors.ensemble.sklearn_ensemble), 56	cancel() (dq0.sdk.cli.runner.query_runner
add_function() (in module dq0.sdk.cli.utils	QueryRunner method), 13
code), 15	cancel() (dq0.sdk.cli.runner.QueryRunner
add_shared_axis_labels() (in module dq0.sdk	method), 10
data.utils.plotting), 44	cancel() (dq0.sdk.cli.runner.runner.Runner
all() (dq0.sdk.cli.Data method), 5	method), 13
all() (dq0.sdk.cli.data.Data method), 15	case_insensitive_str_comparison() (in module
as_dict() (dq0.sdk.cli.Data method), 5	dq0.sdk.data.utils.util), 46
as_dict() (dq0.sdk.cli.data.Data method), 15	check_data_structure_type_consistency() (in module dq0.sdk.data.utils.util), 46
attach_data_source() (dq0.sdk.cli.Project	check_for_valid_numerical_encoding_of_labels()
method), 3	(in module dq0.sdk.data.utils.util), 46
attach_data_source() (dq0.sdk.cli.project.Project	check_signature() (in module dq0.sdk.cli.utils
method), 19	code), 15
attach_data_source() (dq0.sdk.projects.Project	checkSDKResponse() (in module dq0.sdk.er-
method), 88	rors), 53
attach_data_source() (dq0.sdk.projects.projects.project method), 89	checkSDKResponse() (in module dq0.sdk.er-
throject method), 69	rors.errors), 54
В	ClassifierMixin (class in dq0.sdk.estima-
	tors.base_mixin), 60
BaggingClassifier (class in dq0.sdk.estima-	Client (class in dq0.sdk.cli.api), 7
tors.ensemble.sklearn_ensemble), 56	Client (class in dq0.sdk.cli.api.client), 8
BaggingRegressor (class in dq0.sdk.estimators.ensemble.sklearn_ensemble), 56	ClusterMixin (class in dq0.sdk.estima-
BasicDataHandler (class in dq0.sdk.estimators	tors.base_mixin), 60
data_handler.base), 55	Column (class in dq0.sdk.data.metadata.meta-
BiclusterMixin (class in dq0.sdk.estima-	data), 31
tors.base_mixin), 60	ColumnSelector (class in dq0.sdk.pipeline
BigQuery (class in dq0.sdk.data.sql), 34	transformer.transformer), 85 combine_with() (dq0.sdk.data.metadata.Meta-
BigQuery (class in dq0.sdk.data.sql.big_query),	data method), 29
37	combine_with() (dq0.sdk.data.metadata.meta-
Binarizer (class in dq0.sdk.pipeline.trans-	data.Metadata method), 31
former.transformer), 85	commit() (dq0.sdk.cli.Project method), 3
	commit() (dq0.sdk.cli.project.Project method),
C	19
cancel() (dq0.sdk.cli.runner.data_runner	compute_confusion_matrix() (in module dq0.s-
DataRunner method), 11	dk.data.utils.plotting), 44
cancel() (dq0.sdk.cli.runner.DataRunner	compute_metrics_scores() (in module dq0.sdk

data.utils.util), 47	dq0.sdk.cli.runner.data_runner (module), 10
concatenate_train_test_datasets() (in module	dq0.sdk.cli.runner.model_runner (module), 11
dq0.sdk.data.utils.util), 47	dq0.sdk.cli.runner.query_runner (module), 12
concatenate_train_test_datasets_np_array() (in	dq0.sdk.cli.runner.runner (module), 13
module dq0.sdk.data.utils.util), 47	dq0.sdk.cli.runner.state (module), 14
concatenate_train_test_datasets_pd_Dataframes()	dq0.sdk.cli.utils (module), 14
(in module dq0.sdk.data.utils.util), 47	dq0.sdk.cli.utils.code (module), 15
copy_obj_attributes() (in module dq0.sdk.da-	dq0.sdk.cli.utils.uuid (module), 15
ta.utils.util), 47	dq0.sdk.data (module), 22
create_dataset() (in module dq0.sdk.exam-	dq0.sdk.data.binary (module), 23
ples.pneumonia.data_preparation), 71	dq0.sdk.data.binary.excel (module), 25
create_df() (in module dq0.sdk.examples.pneu-	dq0.sdk.data.binary.feather (module), 26
monia.data_preparation), 71	dq0.sdk.data.binary.hdf5 (module), 26
create_feature_vector() (in module dq0.sdk.ex-	dq0.sdk.data.binary.odf (module), 26
amples.pneumonia.data_preparation),	dq0.sdk.data.binary.orc (module), 27
71	dq0.sdk.data.binary.parquet (module), 27
create_from_type() (in module dq0.sdk.data	dq0.sdk.data.binary.sas (module), 27
data_source_factory), 52	dq0.sdk.data.binary.spss (module), 28
CSV (class in dq0.sdk.data.text), 42	dq0.sdk.data.binary.stata (module), 28
CSV (class in dq0.sdk.data.text.csv), 43	dq0.sdk.data.data_source_factory (module), 51
CSVDataHandler (class in dq0.sdk.estimators	dq0.sdk.data.image (module), 28
data_handler.csv), 55	dq0.sdk.data.image.image (module), 29
	dq0.sdk.data.metadata (module), 29
D	dq0.sdk.data.metadata.metadata (module), 30
Data (class in dq0.sdk.cli), 5	dq0.sdk.data.preprocessing (module), 33
Data (class in dq0.sdk.cli, 3	dq0.sdk.data.preprocessing.preprocessing
data_handler_factory() (in module dq0.sdk.esti-	(module), 33
mators.data_handler.utils), 56	dq0.sdk.data.source (module), 52
	dq0.sdk.data.sql (module), 34
dataframe_has_columns_of_these_types() (in	dq0.sdk.data.sql.big_query (module), 37
module dq0.sdk.data.utils.util), 48	dq0.sdk.data.sql.drill (module), 37
DataRunner (class in dq0.sdk.cli.runner), 9	dq0.sdk.data.sql.mssql (module), 38
DataRunner (class in dq0.sdk.cli.runner	dq0.sdk.data.sql.mysql (module), 38
data_runner), 11	dq0.sdk.data.sql.oracle (module), 39
datasets_are_equal() (in module dq0.sdk.da-	dq0.sdk.data.sql.postgresql (module), 39
ta.utils.util), 48	dq0.sdk.data.sql.redshift (module), 39
detach_data_source() (dq0.sdk.cli.Project	dq0.sdk.data.sql.sap_hana (module), 40
method), 3	dq0.sdk.data.sql.snowflake (module), 40
detach_data_source() (dq0.sdk.cli.projec-	dq0.sdk.data.sql.sql (module), 41
t.Project method), 19	dq0.sdk.data.sql.sqlite (module), 42
detach_data_source() (dq0.sdk.projects.Project	dq0.sdk.data.text (module), 42
method), 88	dq0.sdk.data.text.csv (module), 43
detach_data_source() (dq0.sdk.projects.projec-	dq0.sdk.data.text.json (module), 43
t.Project method), 89	dq0.sdk.data.transform (module), 52
distribution() (dq0.sdk.cli.Data method), 5	dq0.sdk.data.utils (module), 43
distribution() (dq0.sdk.cli.data.Data method),	dq0.sdk.data.utils.plotting (module), 43
16	dq0.sdk.data.utils.util (module), 46
dq0.sdk (module), 1	dq0.sdk.errors (module), 53
dq0.sdk.cli (module), 1	dq0.sdk.errors.errors (module), 53
dq0.sdk.cli.api (module), 7	dq0.sdk.estimators (module), 54
dq0.sdk.cli.api.client (module), 8	dq0.sdk.estimators.base_mixin (module), 60
dq0.sdk.cli.api.routes (module), 9	dq0.sdk.estimators.data_handler (module), 55
dq0.sdk.cli.data (module), 15	dq0.sdk.estimators.data_handler.base (mod-
dq0.sdk.cli.experiment (module), 16	ule), 55
dq0.sdk.cli.model (module), 17	dq0.sdk.estimators.data_handler.csv (module),
dq0.sdk.cli.project (module), 19	55
dq0.sdk.cli.query (module), 21	dq0.sdk.estimators.data_handler.utils (mod-
dq0.sdk.cli.runner (module), 9	ule), 55

dq0.sdk.estimators.ensemble (module), 56	(module), 68
dq0.sdk.estimators.ensemble.sklearn_ensemble	dq0.sdk.examples.newsgroups.bayesian.run_demo
(module), 56	(module), 68
dq0.sdk.estimators.estimator (module), 61	dq0.sdk.examples.newsgroups.network (mod-
dq0.sdk.estimators.linear_model (module), 57	ule), 69
dq0.sdk.estimators.linear_model.diffprivlib_lm	dq0.sdk.examples.newsgroups.network.model.user_model
(module), 57	(module), 69
dq0.sdk.estimators.linear_model.sklearn_lm	dq0.sdk.examples.newsgroups.network.model.user_model_from
(module), 57	(module), 69
dq0.sdk.estimators.SVM (module), 54	dq0.sdk.examples.newsgroups.network.model.user_model_old_
dq0.sdk.estimators.SVM.sklearn_svm (mod- ule), 54	(module), 70
dq0.sdk.estimators.tf (module), 58	dq0.sdk.examples.newsgroups.network.run_demo (module), 70
dq0.sdk.estimators.tf.keras_base (module), 58	dq0.sdk.examples.patient (module), 70
dq0.sdk.estimators.tf.keras_dense_classifier	dq0.sdk.examples.patient.model.user_model
(module), 59	(module), 70
dq0.sdk.estimators.tf.keras_dense_regressor	dq0.sdk.examples.pneumonia.data_preparation
(module), 60	(module), 71
dq0.sdk.examples (module), 61	dq0.sdk.examples.pneumonia.my_model
dq0.sdk.examples.census (module), 61	(module), 71
dq0.sdk.examples.census.bayesian (module), 61	dq0.sdk.examples.pneumonia.my_model_int_encoding
dq0.sdk.examples.census.bayesian.model.user_model	
(module), 61	dq0.sdk.models (module), 72
dq0.sdk.examples.census.bayesian.run_demo	dq0.sdk.models.bayes (module), 73
(module), 62	dq0.sdk.models.bayes.naive_bayesian_model
dq0.sdk.examples.census.eda (module), 64	(module), 74
dq0.sdk.examples.census.preprocessed (mod-	dq0.sdk.models.model (module), 84
ule), 62	dq0.sdk.models.tf (module), 75
$dq0.sdk. examples. census. preprocessed. model. user\_mathemath{ \  \  }$	nodatel.sdk.models.tf.neural_network (module), 78
(module), 62	dq0.sdk.models.tf.neural_network_classification
dq0.sdk.examples.census.preprocessed.run_demo	(module), 79
(module), 63	dq0.sdk.models.tf.neural_network_multiclass_classification
dq0.sdk.examples.census.raw (module), 63	(module), 79
dq0.sdk.examples.census.raw.model.user_model	dq0.sdk.models.tf.neural_network_regression
(module), 63	(module), 80
dq0.sdk.examples.census.raw.run_demo (mod-	dq0.sdk.models.tf.neural_network_yaml (mod-
ule), 64	ule), 80
dq0.sdk.examples.cifar (module), 64	dq0.sdk.models.tf.tf_hub (module), 81
dq0.sdk.examples.cifar.model.user_model	dq0.sdk.models.user (module), 82
(module), 64	dq0.sdk.models.user.user_model (module), 82
dq0.sdk.examples.cifar.model.user_model_old	dq0.sdk.models.yaml_configs (module), 83
(module), 65 dq0.sdk.examples.cifar.run_demo (module), 65	dq0.sdk.models.yaml_configs.tf_hub_models (module), 83
dq0.sdk.examples.har.model.user_model (mod-	dq0.sdk.pipeline (module), 85
ule), 66	dq0.sdk.pipeline (module), 87
dq0.sdk.examples.har.run_demo (module), 67	dq0.sdk.pipeline.pipeline_config (module), 87
dq0.sdk.examples.medical_insurance (module),	dq0.sdk.pipeline.transformer.transformer
67	(module), 85
dq0.sdk.examples.medical_insurance.model.user_mo	
(module), 67	dq0.sdk.projects.project (module), 88
dq0.sdk.examples.medical_insurance.run_demo	dq0.sdk.utils (module), 89
(module), 67	dq0.sdk.utils.managed_classes (module), 89
dq0.sdk.examples.newsgroups (module), 68	dq0.sdk.utils.parse_args (module), 89
dq0.sdk.examples.newsgroups.bayesian (mod-	dq0.sdk.utils.yaml_config (module), 89
ule), 68	DQ0SDKError, 53, 54
dq0.sdk.examples.newsgroups.bayesian.model.user_r	
(module), 68	Drill (class in dq0.sdk.data.sql.drill), 38
dall edk evamples newsgaroune havesian model user t	

ta.metadata.Metadata method), 30	method), 36
drop_columns_with_key_value() (dq0.sdk.da-	execute() (dq0.sdk.data.sql.postgresql.Post-
ta.metadata.metadata.Metadata	greSQL method), 39
method), 31 dump_model() (in module dq0.sdk.data.utils.u-	execute() (dq0.sdk.data.sql.Redshift method),
til), 48	execute() (dq0.sdk.data.sql.redshift.Redshift
dump_yaml() (dq0.sdk.utils.yaml_config.Yaml-	method), 40
Config method), 90	execute() (dq0.sdk.data.sql.sap_hana.SAPHana
dump_yaml() (dq0.sdk.utils.YamlConfig	method), 40
method), 89	execute() (dq0.sdk.data.sql.SAPHana method),
_	36
E	execute() (dq0.sdk.data.sql.Snowflake method), 37
eda() (in module dq0.sdk.examples.cen-	execute() (dq0.sdk.data.sql.snowflake.S-
sus.eda), 64	nowflake method), 41
ElasticNet (class in dq0.sdk.estimators.linear model.sklearn_lm), 57	execute() (dq0.sdk.data.sql.sql.SQL method), 41
empty_folder() (in module dq0.sdk.data.utils.u-	execute() (dq0.sdk.data.sql.SQLite method), 37
til), 48	execute() (dq0.sdk.data.sql.sqlite.SQLite
estimate_freq_of_labels() (in module dq0.sdk	method), 42
data.utils.util), 48	execute() (dq0.sdk.data.Transform method), 22
Estimator (class in dq0.sdk.estimators.estima-	execute() (dq0.sdk.data.transform.Transform method), 53
tor), 61	Experiment (class in dq0.sdk.cli), 1
evaluate() (dq0.sdk.examples.pneumonia.my	Experiment (class in dq0.sdk.cli.experiment), 17
model.UserModel method), 71 evaluate() (dq0.sdk.examples.pneumonia.my	extract_count_features_from_text_corpus() (in
model_int_encoding.UserModel	module dq0.sdk.data.preprocess-
method), 72	ing.preprocessing), 33
evaluate() (dq0.sdk.models.bayes	ExtraTreesClassifier (class in dq0.sdk.estima-
naive_bayesian_model.NaiveBayesianModel	tors.ensemble.sklearn_ensemble), 56
method), 74	ExtraTreesRegressor (class in dq0.sdk.estimators.ensemble.sklearn_ensemble), 56
evaluate() (dq0.sdk.models.bayes.Naive-	tors.erisembre.skiearit_erisembre), 50
BayesianModel method), 73	F
evaluate() (dq0.sdk.models.tf.neural_net- work.NeuralNetwork method), 78	fatal_error() (in module dq0.sdk.errors), 53
evaluate() (dq0.sdk.models.tf.NeuralNetwork	fatal_error() (in module dq0.sdk.errors.errors),
method), 75	54
Excel (class in dq0.sdk.data.binary), 23	Feather (class in dq0.sdk.data.binary), 23
Excel (class in dq0.sdk.data.binary.excel), 25	Feather (class in dq0.sdk.data.binary.feather),
execute() (dq0.sdk.cli.Query method), 6	26
execute() (dq0.sdk.cli.query.Query method), 21	fit() (dq0.sdk.estimators.estimator.Estimator
execute() (dq0.sdk.data.sql.big_query.BigQuery method), 37	method), 61 fit() (dq0.sdk.estimators.tf.keras_base.Neural-
execute() (dq0.sdk.data.sql.BigQuery method),	NetworkBase method), 59
34	fit() (dq0.sdk.models.bayes.naive_bayesian
execute() (dq0.sdk.data.sql.Drill method), 35	model.NaiveBayesianModel method),
execute() (dq0.sdk.data.sql.drill.Drill method),	74
38	fit() (dq0.sdk.models.bayes.NaiveBayesian-
execute() (dq0.sdk.data.sql.MSSQL method), 35	Model method), 73
execute() (dq0.sdk.data.sql.mssql.MSSQL	fit() (dq0.sdk.models.Model method), 72
method), 38 execute() (dq0.sdk.data.sql.MySQL method), 35	fit() (dq0.sdk.models.model.Model method), 84 fit() (dq0.sdk.models.tf.neural_network.Neural-
execute() (dq0.sdk.data.sql.mysql.MySQL	Network method), 78
method), 39	fit() (dq0.sdk.models.tf.NeuralNetwork
execute() (dq0.sdk.data.sql.Oracle method), 35	method), 75
execute() (dq0.sdk.data.sql.oracle.Oracle	fit() (dq0.sdk.models.user.user_model.User-
method), 39	Model method), 83
execute() (dq0.sdk.data.sql.PostgreSQL	fit() (dq0.sdk.models.user.UserModel method),

82	cli.Project method), 3
fit() (dq0.sdk.pipeline.pipeline.Pipeline	get_attached_data_sources() (dq0.sdk
method), 87	cli.project.Project method), 20
fit() (dq0.sdk.pipeline.transformer.transformer	get_available_data_sources() (dq0.sdk.cli.Data
ColumnSelector method), 85	static method), 5
fit() (dq0.sdk.pipeline.transformer.transformer	get_available_data_sources() (dq0.sdk.cli.data
Transformer method), 86	Data static method), 16
fit() (dq0.sdk.pipeline.transformer.transformer	get_available_data_sources() (dq0.sdk
Transformer_1_to_1 method), 87	cli.Project method), 3
fit() (dq0.sdk.pipeline.transformer.transformer	get_available_data_sources() (dq0.sdk
Transformer_1_to_N method), 87	cli.project.Project method), 20
fit_predict() (dq0.sdk.estimators.base_mixin	<pre>get_categorical_and_quantitative_features_list()</pre>
ClusterMixin method), 60	(in module dq0.sdk.data.utils.util), 48
fit_transform() (dq0.sdk.estimators.base_mix-	get_clone() (dq0.sdk.models.bayes
in.TransformerMixin method), 61	naive_bayesian_model.NaiveBayesianModel
fit_transform() (dq0.sdk.pipeline.pipeline.Pipe-	method), 74
line method), 87	get_clone() (dq0.sdk.models.bayes.Naive-
<pre>fit_transform() (dq0.sdk.pipeline.transformer</pre>	BayesianModel method), 73
transformer. Transformer method), 86	get_clone() (dq0.sdk.models.Model method), 72
<pre>fit_transform() (dq0.sdk.pipeline.transformer</pre>	get_clone() (dq0.sdk.models.model.Model
transformer.Transformer_1_to_1	method), 84
method), 87	get_clone() (dq0.sdk.models.tf.neural_net-
fit_transform() (dq0.sdk.pipeline.transformer	work.NeuralNetwork method), 78
transformer.Transformer_1_to_N	get_clone() (dq0.sdk.models.tf.NeuralNetwork
method), 87	method), 75
<pre>fix_limitation_of_Keras_fit_and_predict_functions()</pre>	get_connection() (dq0.sdk.data.sql.sql.SQL
(in module dq0.sdk.models.tf.neu-	method), 41
ral_network), 79	get_data_info() (dq0.sdk.cli.Data static
for_data() (dq0.sdk.cli.Query method), 6	method), 5
for_data() (dq0.sdk.cli.query.Query method), 21	get_data_info() (dq0.sdk.cli.data.Data static
format_float_lower_than_1() (in module dq0.s-	method), 16
dk.data.utils.util), 48	get_data_info() (dq0.sdk.cli.Project method), 4
from_meta() (dq0.sdk.data.metadata.metadata	get_data_info() (dq0.sdk.cli.project.Project
Column static method), 31	method), 20
from_meta() (dq0.sdk.data.metadata.metada-	get_dataset_names() (dq0.sdk.cli.Query
ta.Schema static method), 32	method), 6
from_meta() (dq0.sdk.data.metadata.metada-	get_dataset_names() (dq0.sdk.cli.query.Query
ta.Table static method), 32	method), 22
FunctionTransformer (class in dq0.sdk.pipe-	get_error() (dq0.sdk.cli.runner.runner.Runner
line.transformer.transformer), 85	method), 14
	get_feature_target_cols() (dq0.sdk.data.metada-
G	ta.Metadata method), 30
get() (dq0.sdk.cli.api.Client method), 7	get_feature_target_cols() (dq0.sdk.data.metada-
get() (dq0.sdk.cli.api.client.Client method), 8	ta.metadata.Metadata method), 31
get_all_schema_names() (dq0.sdk.data.metada-	get_fn() (in module dq0.sdk.data.utils.util), 49
ta.Metadata method), 30	get_header() (dq0.sdk.data.metadata.Metadata
get_all_schema_names() (dq0.sdk.data.metada-	method), 30
ta.metadata.Metadata method), 31	get_header() (dq0.sdk.data.metadata.metada-
get_all_table_names() (dq0.sdk.data.metada-	ta.Metadata method), 31
ta.Metadata method), 30	get_input_dim() (dq0.sdk.estimators.data_han-
get_all_table_names() (dq0.sdk.data.metada-	dler.csv.CSVDataHandler method), 55
ta.metadata.Metadata method), 31	get_last_data_run() (dq0.sdk.cli.Experiment
get_all_tables() (dq0.sdk.data.metadata.Meta-	method), 1
data method), 30	get_last_data_run() (dq0.sdk.cli.experiment.Ex-
get_all_tables() (dq0.sdk.data.metadata.meta-	periment method), 17
data.Metadata method), 31	get_last_model_run() (dq0.sdk.cli.Experiment
get_attached_data_sources() (dq0.sdk	method), 1
	get_last_model_run() (dq0.sdk.cli.experiment

Experiment method), 17	module dq0.sdk.data.utils.util), 49
get_model() (dq0.sdk.cli.runner.model_runner ModelRunner method), 12	instantiate_metrics_from_name() (in module dq0.sdk.data.utils.util), 49
get_model() (dq0.sdk.cli.runner.ModelRunner method), 10	is_numeric() (in module dq0.sdk.data.utils.u- til), 49
get_output_dim() (dq0.sdk.estimators data_handler.csv.CSVDataHandler method), 55	is_valid_uuid() (in module dq0.sdk.cli.utils), 15 is_valid_uuid() (in module dq0.sdk.cli.utils.u-uid), 15
get_param_configuration_for_publication_quality_pl (in module dq0.sdk.data.utils.plot-	ot() J
ting), 44 get_params() (dq0.sdk.pipeline.pipeline.Pipe- line method), 87	JSON (class in dq0.sdk.data.text), 42 JSON (class in dq0.sdk.data.text.json), 43
get_percentage_freq_of_values() (in module dq0.sdk.data.utils.util), 49	K
get_results() (dq0.sdk.cli.runner.runner.Runner method), 14	KBinsDiscretizer (class in dq0.sdk.pipeline transformer.transformer), 85
get_sample_data() (dq0.sdk.cli.Project method),	Keras_Dense_Classifier_Binary (class in dq0.sd-k.estimators.tf.keras_dense_classifier),
get_sample_data() (dq0.sdk.cli.project.Project method), 20	59 Keras_Dense_Classifier_Integer (class in dq0.s-
get_state() (dq0.sdk.cli.runner.data_runner DataRunner method), 11	dk.estimators.tf.keras_dense_classifier), 59
get_state() (dq0.sdk.cli.runner.DataRunner method), 9	Keras_Dense_Classifier_OHE (class in dq0.sd-k.estimators.tf.keras_dense_classifier),
get_state() (dq0.sdk.cli.runner.model_runner ModelRunner method), 12	59 Keras_Dense_Regressor (class in dq0.sdk.esti-
get_state() (dq0.sdk.cli.runner.ModelRunner method), 10	mators.tf.keras_dense_regressor), 60 KernelCenterer (class in dq0.sdk.pipeline.trans-
get_state() (dq0.sdk.cli.runner.query_runner QueryRunner method), 13	former.transformer), 85
get_state() (dq0.sdk.cli.runner.QueryRunner method), 10	L shallbing viscon (along in def) adds ningling trans
get_state() (dq0.sdk.cli.runner.runner.Runner	LabelBinarizer (class in dq0.sdk.pipeline.transformer), 85
method), 14 get_steps_from_config() (dq0.sdk.pipe-	LabelEncoder (class in dq0.sdk.pipeline.transformer), 85
line.pipeline_config.PipelineConfig method), 87	Lasso (class in dq0.sdk.estimators.linear_mod- el.sklearn_lm), 58
GradientBoostingClassifier (class in dq0.sdk.estimators.ensemble.sklearn_ensemble),	layer_factory() (in module dq0.sdk.estimators.t-f.keras_base), 59
56 GradientBoostingRegressor (class in dq0.sd-	LinearRegression (class in dq0.sdk.estimators
k.estimators.ensemble.sklearn_ensemble), 57	linear_model.sklearn_lm), 58 LinearRegressionDP (class in dq0.sdk.estima- tors.linear_model.diffprivlib_lm), 57
H	LinearSVC (class in dq0.sdk.estima- tors.SVM.sklearn_svm), 54
handle_missing_data() (in module dq0.sdk.da-	LinearSVR (class in dq0.sdk.estimators.SVM.sklearn_svm), 54
ta.preprocessing.preprocessing), 33 HDF5 (class in dq0.sdk.data.binary), 23 HDF5 (class in dq0.sdk.data.binary.hdf5), 26	list_subfloders() (in module dq0.sdk.data.util-s.util), 49
I	load() (dq0.sdk.cli.Project static method), 4 load() (dq0.sdk.cli.project.Project static
Image (class in dq0.sdk.data.image), 28 Image (class in dq0.sdk.data.image.image), 29 info() (dq0.sdk.cli.Project method), 4	method), 20 load() (dq0.sdk.models.bayes.naive_bayesian model.NaiveBayesianModel method), 74
info() (dq0.sdk.cli.project.Project method), 20 initialize_rnd_numbers_generators_state() (in	load() (dq0.sdk.models.bayes.NaiveBayesian- Model method) 74

load() (dqU.sdk.models.Model method), 72	
load() (dq0.sdk.models.model.Model method),	N
84	NaiveBayesianModel (class in dq0.sdk.model-
load() (dq0.sdk.models.tf.neural_network.Neu-	s.bayes), 73
ralNetwork method), 79	NaiveBayesianModel (class in dq0.sdk.model-
load() (dq0.sdk.models.tf.neural_network_yam-	s.bayes.naive_bayesian_model), 74
l.NeuralNetworkYaml method), 80	NeuralNetwork (class in dq0.sdk.models.tf), 75
load() (dq0.sdk.models.tf.NeuralNetwork	NeuralNetwork (class in dq0.sdk.models.tf.neu-
method), 75	ral_network), 78
load() (dq0.sdk.models.tf.NeuralNetworkYaml	NeuralNetworkBase (class in dq0.sdk.estima-
method), 76	tors.tf.keras_base), 59
load() (dq0.sdk.models.user_model.User-	NeuralNetworkClassification (class in dq0.sdk
Model method), 83	models.tf), 76
load() (dq0.sdk.models.user.UserModel	NeuralNetworkClassification (class in dq0.sdk
method), 82	models.tf.neural_network_classification),
load_img_in_dir() (in module dq0.sdk.exam-	79
ples.pneumonia.data_preparation), 71	NeuralNetworkMultiClassClassification (class
load_model_from_file() (in module dq0.sdk.da-	in dq0.sdk.models.tf), 76
ta.utils.util), 49	NeuralNetworkMultiClassClassification (class
load_params_from_config_file() (in module dq0.sdk.data.utils.util), 49	in dq0.sdk.models.tf.neural_network
LogisticRegression (class in dq0.sdk.estima-	multiclass_classification), 80
tors.linear_model.sklearn_lm), 58	NeuralNetworkRegression (class in dq0.sdk
LogisticRegressionDP (class in dq0.sdk.estima-	models.tf), 76
tors.linear_model.diffprivlib_lm), 57	NeuralNetworkRegression (class in dq0.sdk
toro.intear_intoaci.amprivino_int/, o/	models.tf.neural_network_regression),
M	80
	NeuralNetworkYaml (class in dq0.sdk.model-
manage_rnd_num_generators_state() (in	s.tf), 76
module dq0.sdk.data.utils.util), 50	NeuralNetworkYaml (class in dq0.sdk.models.t- f.neural_network_yaml), 80
MaxAbsScaler (class in dq0.sdk.pipeline.trans-	NN_Classifier (class in dq0.sdk.estimators.tf.k-
former.transformer), 85 mean() (dq0.sdk.cli.Data method), 5	eras_base), 58
mean() (dq0.sdk.cli.data.Data method), 16	NN_Regressor (class in dq0.sdk.estimators.tf.k-
Metadata (class in dq0.sdk.data.metadata), 29	eras_base), 58
Metadata (class in dq0.sdk.data.metadata.meta-	Normalizer (class in dq0.sdk.pipeline.trans-
data), 31	former.transformer), 86
MinMaxScaler (class in dq0.sdk.pipeline.trans-	numerical_datasets_are_equal() (in module
former.transformer), 86	dq0.sdk.data.utils.util), 50
missing_values_table() (in module dq0.sdk.da-	NuSVC (class in dq0.sdk.estima-
ta.utils.util), 50	tors.SVM.sklearn_svm), 54
Model (class in dq0.sdk.cli), 2	NuSVR (class in dq0.sdk.estima-
Model (class in dq0.sdk.cli.model), 18	tors.SVM.sklearn_svm), 55
Model (class in dq0.sdk.models), 72	
Model (class in dq0.sdk.models.model), 84	0
ModelRunner (class in dq0.sdk.cli.runner), 9	ODF (class in dq0.sdk.data.binary), 24
ModelRunner (class in dq0.sdk.cli.runner.mod-	ODF (class in dq0.sdk.data.binary.odf), 26
el_runner), 11	OneClassSVM (class in dq0.sdk.estima-
move_files() (in module dq0.sdk.data.utils.util),	tors.SVM.sklearn_svm), 55
50	OneHotEncoder (class in dq0.sdk.pipeline
MSSQL (class in dq0.sdk.data.sql), 35	transformer.transformer), 86
MSSQL (class in dq0.sdk.data.sql.mssql), 38	Oracle (class in dq0.sdk.data.sql), 35
MultiLabelBinarizer (class in dq0.sdk.pipeline	Oracle (class in dq0.sdk.data.sql.oracle), 39
transformer.transformer), 86	ORC (class in dq0.sdk.data.binary), 24
MySQL (class in dq0.sdk.data.sql), 35	ORC (class in dq0.sdk.data.binary.orc), 27
MySQL (class in dq0.sdk.data.sql.mysql), 38	OrdinalEncoder (class in dq0.sdk.pipeline
	transformer.transformer), 86

P	predict_proba() (dq0.sdk.estimators.tf.k- eras_base.NN_Regressor method), 59
Parquet (class in dq0.sdk.data.binary), 24 Parquet (class in dq0.sdk.data.binary.parquet),	preprocess() (dq0.sdk.cli.Experiment method), 2
27	preprocess() (dq0.sdk.cli.experiment.Experi-
parse_kwargs() (in module dq0.sdk.utils	ment method), 17
parse_args), 89	preprocess() (dq0.sdk.examples.cen-
parse_value() (in module dq0.sdk.utils	sus.bayesian.model.user_model.UserModel method), 62
parse_args), 89	preprocess() (dq0.sdk.examples.census.raw
perform_stratified_random_sampling() (in	model.user_model.UserModel
module dq0.sdk.data.utils.util), 50	method), 63
Pipeline (class in dq0.sdk.pipeline.pipeline), 87	preprocess() (dq0.sdk.examples.har.mod-
PipelineConfig (class in dq0.sdk.pipe-	el.user_model.UserModel method), 66
line.pipeline_config), 87	preprocess() (dq0.sdk.models.Model method),
plot_bars() (in module dq0.sdk.data.utils.plot-	72
ting), 44 plot_confusion_matrix() (in module dq0.sdk	preprocess() (dq0.sdk.models.model.Model
data.utils.plotting), 44	method), 84
plot_confusion_matrix_for_scikit_classifier() (in	preprocess() (dq0.sdk.models.user.user_mod-
module dq0.sdk.data.utils.plotting), 44	el.UserModel method), 83
plot_decision_tree() (in module dq0.sdk.data.u-	preprocess() (dq0.sdk.models.user.UserModel
tils.plotting), 45	method), 82
plot_hist() (in module dq0.sdk.data.utils.plot-	preprocess_dataset() (in module dq0.sdk.exam-
ting), 45	ples.census.eda), 64
PolynomialFeatures (class in dq0.sdk.pipeline	pretty_display_string_on_terminal() (in module
transformer.transformer), 86	dq0.sdk.data.utils.util), 50
post() (dq0.sdk.cli.api.Client method), 7	pretty_print_dict() (in module dq0.sdk.data.u-
post() (dq0.sdk.cli.api.client.Client method), 8	tils.util), 50
PostgreSQL (class in dq0.sdk.data.sql), 36	pretty_print_strings_list() (in module dq0.sdk
PostgreSQL (class in dq0.sdk.data.sql.post-	data.utils.util), 50 print_dataset_info() (in module dq0.sdk.data.u-
gresql), 39	tils.util), 50
PowerTransformer (class in dq0.sdk.pipeline	print_details_about_df_columns() (in module
transformer.transformer), 86	dq0.sdk.data.utils.util), 50
predict() (dq0.sdk.cli.Model method), 2	print_evaluation_res() (in module dq0.sdk.da-
predict() (dq0.sdk.cli.model.Model method), 18	ta.utils.util), 50
predict() (dq0.sdk.estimators.base_mixin.Clas- sifierMixin method), 60	print_full_df() (in module dq0.sdk.data.utils.u-
predict() (dq0.sdk.estimators.base_mixin.Re-	til), 51
gressorMixin method), 60	<pre>print_human_readable_elapsed_time_value()</pre>
predict() (dq0.sdk.estimators.tf.keras_base.N-	(in module dq0.sdk.data.utils.util), 51
N_Classifier method), 58	print_summary_stats() (in module dq0.sdk.da-
predict() (dq0.sdk.estimators.tf.keras_base.N-	ta.utils.util), 51
N_Regressor method), 58	Project (class in dq0.sdk.cli), 3
predict() (dq0.sdk.models.tf.neural_net-	Project (class in dq0.sdk.cli.project), 19
work.NeuralNetwork method), 79	Project (class in dq0.sdk.projects), 88
predict() (dq0.sdk.models.tf.neural_net-	Project (class in dq0.sdk.projects.project), 88
work_yaml.NeuralNetworkYaml	
method), 80	Q
predict() (dq0.sdk.models.tf.NeuralNetwork	QuantileTransformer (class in dq0.sdk.pipe-
method), 76	line.transformer.transformer), 86
predict() (dq0.sdk.models.tf.NeuralNet-	Query (class in dq0.sdk.cli), 6
workYaml method), 77	Query (class in dq0.sdk.cli.query), 21
predict_proba() (dq0.sdk.estimators.base_mix-	query() (dq0.sdk.cli.Data method), 6
in.ClassifierMixin method), 60	query() (dq0.sdk.cli.data.Data method), 16
predict_proba() (dq0.sdk.estimators.base_mix-	QueryRunner (class in dq0.sdk.cli.runner), 10
in.RegressorMixin method), 61	QueryRunner (class in dq0.sdk.cli.runner
predict_proba() (dq0.sdk.estimators.tf.k- eras_base.NN_Classifier method), 58	query_runner), 12
crao_baben vi v_Crabbilier incurba/, ou	

	method), 89
R	redirect_stdout_stderr_streams_to_file() (in
RandomForestClassifier (class in dq0.sdk.esti-	module dq0.sdk.data.utils.util), 51
mators.ensemble.sklearn_ensemble),	Redshift (class in dq0.sdk.data.sql), 36
57	Redshift (class in dq0.sdk.data.sql.redshift), 40
RandomForestRegressor (class in dq0.sdk.esti-	refresh() (dq0.sdk.cli.Data method), 6
mators.ensemble.sklearn_ensemble),	refresh() (dq0.sdk.cli.data.Data method), 16
57	register() (dq0.sdk.cli.Model method), 2
read() (dq0.sdk.data.binary.Excel method), 23	register() (dq0.sdk.cli.model.Model method), 18
read() (dq0.sdk.data.binary.excel.Excel	RegressorMixin (class in dq0.sdk.estima-
method), 25	tors.base_mixin), 60
read() (dq0.sdk.data.binary.Feather method), 23	replace_function() (in module dq0.sdk.cli.utils
read() (dq0.sdk.data.binary.feather.Feather	code), 15
method), 26	replace_model_parent_class() (in module
read() (dq0.sdk.data.binary.HDF5 method), 23	dq0.sdk.cli.utils.code), 15
read() (dq0.sdk.data.binary.hdf5.HDF5	restore_stdout_stderr_streams() (in module
method), 26	dq0.sdk.data.utils.util), 51
read() (dq0.sdk.data.binary.ORC method), 24	Ridge (class in dq0.sdk.estimators.linear_mod-
read() (dq0.sdk.data.binary.orc.ORC method),	el.sklearn_lm), 58
27	RidgeClassifier (class in dq0.sdk.estimators.lin-
read() (dq0.sdk.data.binary.Parquet method),	ear_model.sklearn_lm), 58
24	RobustScaler (class in dq0.sdk.pipeline.trans-
read() (dq0.sdk.data.binary.parquet.Parquet	former.transformer), 86
method), 27	run() (dq0.sdk.cli.Experiment method), 2
read() (dq0.sdk.data.binary.SAS method), 24	run() (dq0.sdk.cli.experiment.Experiment
read() (dq0.sdk.data.binary.sas.SAS method), 28	method), 17
read() (dq0.sdk.data.binary.SPSS method), 25	run_all() (dq0.sdk.models.tf.neural_net-
read() (dq0.sdk.data.binary.spss.SPSS method),	work_yaml.NeuralNetworkYaml
28	method), 81
read() (dq0.sdk.data.binary.Stata method), 25	run_all() (dq0.sdk.models.tf.NeuralNet-
read() (dq0.sdk.data.binary.stata.Stata method),	workYaml method), 77
28	Runner (class in dq0.sdk.cli.runner.runner), 13
read() (dq0.sdk.data.image.Image method), 29	
read() (dq0.sdk.data.image.image.Image	S
method), 29	SAPHana (class in dq0.sdk.data.sql), 36
read() (dq0.sdk.data.Source method), 22	SAPHana (class in dq0.sdk.data.sql.sap_hana),
read() (dq0.sdk.data.source.Source method), 52	40
read() (dq0.sdk.data.sql.sql.SQL method), 41	SAS (class in dq0.sdk.data.binary), 24
read() (dq0.sdk.data.text.CSV method), 42	SAS (class in dq0.sdk.data.binary.sas), 27
read() (dq0.sdk.data.text.csv.CSV method), 43	save() (dq0.sdk.models.bayes.naive_bayesian
read() (dq0.sdk.data.text.JSON method), 42	model.NaiveBayesianModel method),
read() (dq0.sdk.data.text.json.JSON method), 43	75
read_from_yaml() (dq0.sdk.data.metada-	save() (dq0.sdk.models.bayes.NaiveBayesian-
ta.Metadata method), 30	Model method), 74
read_from_yaml() (dq0.sdk.data.metada-	save() (dq0.sdk.models.Model method), 73
ta.metadata.Metadata method), 31	save() (dq0.sdk.models.model.Model method),
read_from_yaml() (dq0.sdk.pipeline.pipeline	84
config.PipelineConfig method), 88	save() (dq0.sdk.models.tf.neural_network.Neu-
read_from_yaml_file() (dq0.sdk.data.metada-	ralNetwork method), 79
ta.Metadata method), 30	save() (dq0.sdk.models.tf.neural_network_yam-
read_from_yaml_file() (dq0.sdk.data.metada-	l.NeuralNetworkYaml method), 81
ta.metadata.Metadata method), 31	save() (dq0.sdk.models.tf.NeuralNetwork
read_from_yaml_file() (dq0.sdk.pipe-	method), 76
line.pipeline_config.PipelineConfig	save() (dq0.sdk.models.tf.NeuralNetworkYaml
method), 88	method), 77
read_yaml_file() (dq0.sdk.utils.yaml_con-	save() (dq0.sdk.models.user.user_model.User-
fig. YamlConfig method), 90	Model method), 83
read_yaml_file() (dq0.sdk.utils.YamlConfig	<i>"</i>

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.Yaml-Config 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), 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), 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.k $eras\_base. Neural Network Base$ method), 59 setup\_data() (dq0.sdk.examples.cen $sus.bayesian.model.user\_model.UserModel\\$ method), 62 setup\_data() (dq0.sdk.examples.census.prepro-

cessed.model.user\_model.UserModel

el.user\_model.UserModel method), 65

setup\_data() (dq0.sdk.examples.census.raw.-

setup\_data() (dq0.sdk.examples.cifar.mod-

setup\_data() (dq0.sdk.examples.cifar.mod-

 $model.user\_model.UserModel$ 

el.user\_model\_old.UserModel

method), 63

method), 64

method), 65 setup\_data() (dq0.sdk.examples.har.model.user\_model.UserModel method), 66 setup\_data() (dq0.sdk.examples.medical\_insur $ance.model.user\_model.UserModel\\$ 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.newsgroup $s.network.model.user\_model.UserModel$ 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), 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

102 Index

method), 60

setup\_model() (dq0.sdk.estimators.tf.keras\_-

dense\_regressor.Keras\_Dense\_Regressor

method), 60	Source (class in dq0.sdk.data), 22
setup_model() (dq0.sdk.examples.cen-	Source (class in dq0.sdk.data.source), 52
sus.bayesian.model.user_model.UserModel	sparse_scipy_matrix_to_Pandas_df() (in
method), 62	module dq0.sdk.data.utils.util), 51
setup_model() (dq0.sdk.examples.census.pre-	SPSS (class in dq0.sdk.data.binary), 25
processed.model.user_model.UserModel	SPSS (class in dq0.sdk.data.binary.spss), 28
method), 63	SQL (class in dq0.sdk.data.sql.sql), 41
setup_model() (dq0.sdk.examples.census.raw	SQLite (class in dq0.sdk.data.sql), 37
model.user_model.UserModel	SQLite (class in dq0.sdk.data.sql.sqlite), 42
method), 64	StandardScaler (class in dq0.sdk.pipeline.trans-
setup_model() (dq0.sdk.examples.cifar.mod-	former.transformer), 86
el.user_model.UserModel method), 65	Stata (class in dq0.sdk.data.binary), 25
setup_model() (dq0.sdk.examples.cifar.mod-	Stata (class in dq0.sdk.data.binary.stata), 28
el.user_model_old.UserModel	State (class in dq0.sdk.cli.runner.state), 14
method), 65	str_to_bool() (in module dq0.sdk.data.utils.u-
setup_model() (dq0.sdk.examples.har.mod-	til), 51
el.user_model.UserModel method), 67	string_contains_numeric_value() (in module
setup_model() (dq0.sdk.examples.medical_in-	dq0.sdk.data.utils.util), 51
surance.model.user_model.UserModel	SVC (class in dq0.sdk.estima-
method), 67	tors.SVM.sklearn_svm), 55
setup_model() (dq0.sdk.examples.newsgroup-	SVR (class in dq0.sdk.estima-
s.bayesian.model.user_model.UserModel	tors.SVM.sklearn_svm), 55
method), 68	
setup_model() (dq0.sdk.examples.newsgroup-	T
s.bayesian.model.user_model_old_needs_SC	OURGE User Model salk data metadata meta-
method), 68	data), 32
setup_model() (dq0.sdk.examples.newsgroup-	tensorflow_tensor_to_numpy_ndarray() (in
s.network.model.user_model.UserModel	module dq0.sdk.data.utils.util), 51
method), 69	
setup_model() (dq0.sdk.examples.newsgroup-	TFHub (class in dq0.sdk.models.tf), 77 TFḤub (class in dq0.sdk.models.tf.tf_hub), 81
s.network.model.user_model_from_preproce	essed data User Model
method), 69	mothod) 20
1-10 (1-0 - 11 - 10 - 11 - 11 - 11 - 11	method), 30
s.network.model.user model old needs SO	to_dict() (dq0.sdk.data.metadata.metadata URCE.UserModel Column method), 31
method), 70	
setup_model() (dq0.sdk.examples.patient.mod-	to_dict() (dq0.sdk.data.metadata.metada-
el.user_model.UserModel method), 70	ta.Metadata method), 32
setup_model() (dq0.sdk.examples.pneumoni-	to_dict() (dq0.sdk.data.metadata.metada-
a.my_model.UserModel method), 71	ta.Schema method), 32
setup_model() (dq0.sdk.examples.pneumoni-	to_dict() (dq0.sdk.data.metadata.metada-
a.my_model_int_encoding.UserModel	ta.Table method), 32
method), 72	to_dict_sm() (dq0.sdk.data.metadata.Metadata
setup_model() (dq0.sdk.models.Model	method), 30
method), 73	to_dict_sm() (dq0.sdk.data.metadata.metada-
setup_model() (dq0.sdk.models.model.Model	ta.Metadata method), 32
method), 85	to_json() (dq0.sdk.data.image.Image method),
setup_model() (dq0.sdk.models.tf.neural_net-	29
work_yaml.NeuralNetworkYaml	to_json() (dq0.sdk.data.image.image.Image
method), 81	method), 29
setup_model() (dq0.sdk.models.tf.NeuralNet-	to_json() (dq0.sdk.data.Source method), 22
workYaml method), 77	to_json() (dq0.sdk.data.source.Source method),
	52
setup_model() (dq0.sdk.models.user.user_mod-	to_json() (dq0.sdk.data.sql.sql.SQL method), 41
el.UserModel method), 83	to_string() (dq0.sdk.models.bayes
setup_model() (dq0.sdk.models.user.User-	naive_bayesian_model.NaiveBayesianModel
Model method), 82	method), 75
Snowflake (class in dq0.sdk.data.sql), 36	to_string() (dq0.sdk.models.bayes.Naive-
Snowflake (class in dq0.sdk.data.sql.s-	BayesianModel method), 74
nowflake), 40	

to_string() (dq0.sdk.models.Model method), 73	
to_string() (dq0.sdk.models.model.Model	U
method), 85	univariate_feature_selection() (in module
to_string() (dq0.sdk.models.tf.neural_network	dq0.sdk.data.preprocessing.preprocessing),
classification.NeuralNetworkClassification	34
method), 79	update() (dq0.sdk.cli.runner.state.State
to_string() (dq0.sdk.models.tf.neural_network	method) 14
multiclass_classification.NeuralNetworkMu	ltiClassClassificationid() (dq0.sdk.cli.Project
method), 80	method), 5
to_string() (dq0.sdk.models.tf.neural_net-	update_commit_uuid() (dq0.sdk.cli.projec-
work_regression.NeuralNetworkRegression	t.Project method), 21
method), 80	UserModel (class in dq0.sdk.examples.cen-
to_string() (dq0.sdk.models.tf.neural_net-	sus.bayesian.model.user_model), 62
work_yaml.NeuralNetworkYaml	UserModel (class in dq0.sdk.examples.cen-
method), 81	sus.preprocessed.model.user_model),
to_string() (dq0.sdk.models.tf.NeuralNetwork-	62
Classification method), 76	UserModel (class in dq0.sdk.examples.cen-
to_string() (dq0.sdk.models.tf.NeuralNetwork-	sus.raw.model.user_model), 63
MultiClassClassification method), 76	UserModel (class in dq0.sdk.examples.cifar
to_string() (dq0.sdk.models.tf.NeuralNet-	model.user_model), 64
workRegression method), 76	UserModel (class in dq0.sdk.examples.cifar
to_string() (dq0.sdk.models.tf.NeuralNet-	model.user_model_old), 65
workYaml method), 77	UserModel (class in dq0.sdk.examples.har
to_yaml() (dq0.sdk.data.metadata.Metadata	model.user_model), 66
method), 30	UserModel (class in dq0.sdk.examples.medi-
to_yaml() (dq0.sdk.data.metadata.metada-	cal_insurance.model.user_model), 67
ta.Metadata method), 32	UserModel (class in dq0.sdk.examples.news-
to_yaml_file() (dq0.sdk.data.metadata.Meta-	groups.bayesian.model.user_model),
data method), 30	68
to_yaml_file() (dq0.sdk.data.metadata.metada-	UserModel (class in dq0.sdk.examples.news-
ta.Metadata method), 32	groups.bayesian.model.user_model_old_needs_SOURG
train_test_split() (in module dq0.sdk.data.pre-	68
processing.preprocessing), 33	UserModel (class in dq0.sdk.examples.news-
train_test_split_reshape() (in module dq0.sd-	groups.network.model.user_model),
k.examples.pneumonia.data_preparation), 71	69
· <del>-</del>	UserModel (class in dq0.sdk.examples.news-
Transform (class in dq0.sdk.data), 22 Transform (class in dq0.sdk.data transform), 53	groups.network.model.user_model_from_preprocessed
Transform (class in dq0.sdk.data.transform), 53 transform() (dq0.sdk.pipeline.transformer	69
transformer.ColumnSelector method),	UserModel (class in dq0.sdk.examples.news-
85	groups.network.model.user_model_old_needs_SOURC
transform() (dq0.sdk.pipeline.transformer	70
transformer.Transformer method), 86	UserModel (class in dq0.sdk.examples.patient
transform() (dq0.sdk.pipeline.transformer	model.user_model), 70
transformer.Transformer_1_to_1	UserModel (class in dq0.sdk.examples.pneu-
method), 87	monia.my_model), 71
transform() (dq0.sdk.pipeline.transformer	UserModel (class in dq0.sdk.examples.pneu-
transformer.Transformer_1_to_N	monia.my_model_int_encoding), 71
method), 87	UserModel (class in dq0.sdk.models.user), 82
Transformer (class in dq0.sdk.pipeline.trans-	UserModel (class in dq0.sdk.models.user.user
former.transformer), 86	model), 83
Transformer_1_to_1 (class in dq0.sdk.pipeline	V
transformer.transformer), 86	V
Transformer_1_to_N (class in dq0.sdk.pipe-	visualize_categorical_distribution() (in module
line.transformer.transformer), 87	dq0.sdk.data.utils.plotting), 46
TransformerMixin (class in dq0.sdk.estima-	visualize_continuous_distribution() (in module
tors.base_mixin), 61	dq0.sdk.data.utils.plotting), 46
<i>,</i> ·	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

# Υ

YamlConfig (class in dq0.sdk.utils), 89 YamlConfig (class in dq0.sdk.utils.yaml\_config), 90