mleap Documentation

Release 1.0.0

Viktor Kazakov

Mar 12, 2018

CONTENTS:

Welcome to `mleap` (Machine Learning Experiments Automation Package).

CHAPTER

ONE

DESCRIPTION OF THE PROJECT

The goal of `mleap` is to automate the workflow for training and comparing machine learning estimators. The package facilitates the training of a large number of estimators on multiple datasets. It also provides a statistical framework for comparing the performance of the trained estimators.

1.1 Overview

`mleap` seeks to expand the functionality of the scikit-learn package. It also aims at providing a seamless integration with other packages such as scipy, statsmodels and scikit-posthocs.

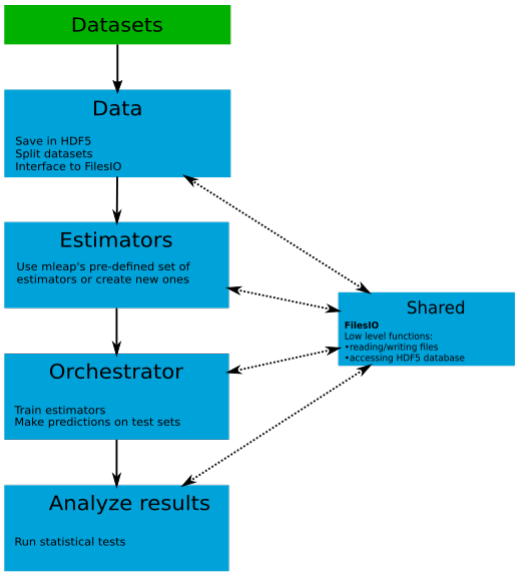
Note: Knowledge of all above mentioned packages is not necessarily required in order to work with `mleap`. However, in order to make full use of it and be able to expand its functionality understanding of scikit-learn is highly desirable.

`mleap` is comprised of the following main modules.

- *Estimators*
- *Data*
- *Experiments*
- *Analyze Results*
- *Shared*

The graph below shows on a high level the interaction between the modules.

The user should begin by saving the preprocessed datasets on its hard drive. The `Data` module is then evoked for saving the datasets in a HDF5 database. The second step of the process is to instantiate the machine learning estimator objects. The user has the choice of either using the pre-defined estimators in the `Estimators` module or defining its own by making use of the *mleap_estimator*. Once this step is completed the user can proceed to running the experiments by making use of the `Orchestrator` module. Once all estimators are trained on all datasets the user can carry out the final step of analyzing the results by evoking the `Analuze results` module. The `Shared` module provides an interface for reading and writing files as well as for accessing the HDF5 database files and is used by all other modules. This module can also be directly accessed by the end user.


Datasets
Data
Save in HDF5
Split datasets
Interface to FilesIO
Estimators
Use mleap's pre-defined set of estimators or create new ones
Shared
FilesIO
Low level functions:
•reading/writing files
•accessing HDF5 database
Orchestrator
Train estimators
Make predictions on test sets
Analyze results
Run statistical tests

CHAPTER

TWO

TUTORIAL

The code below will help you get started with mleap.

The following Jupyter Notebook contains the code used below.

`mleap.ipynb`.

2.1 Data Preprocessing and Estimator Training Phase

1. Get your data.

The included code downloads preprocessed datasets from the UCI Machine Learning Repository and stores them locally.

`delgado_datasets.py`

The enclosed code can be used as follows:

```
delgado = DownloadAndConvertDelgadoDatasets()
datasets, metadata = delgado.download_and_extract_datasets(verbose = False)
```

2. Define Input and Output HDF5 objects.

```
input_io = data.open_hdf5('data/delgado.hdf5', mode='a')
out_io = data.open_hdf5('data/classification.hdf5', mode='a')
```

3. Store your data in HDF5 database.

Once the datasets are stored locally we put them in an HDF5 database.

```
data.pandas_to_db(save_loc_hdf5='delgado_datasets/',
                  datasets=datasets,
                  dts_metadata=metadata,
                  save_loc_hdd='data/delgado.hdf5')
```

4. Split the data in test and train sets.

```
dts_names_list, dts_names_list_full_path = data.list_datasets(hdf5_io=input_io,
                                                              hdf5_group='delgado_datasets/')
split_dts_list = data.split_datasets(hdf5_in=input_io,
                                     hdf5_out=out_io,
                                     dataset_paths=dts_names_list_full_path)
```

5. Instantiate estimator objects and the experiments orchestrator class.

```
instantiated_models = instantiate_default_estimators(estimators=['all'])
orchest = Orchestrator(hdf5_input_io=input_io,
                       hdf5_output_io=out_io,
                       dts_names=dts_names_list,
                       original_datasets_group_h5_path='delgado_datasets/')
```

6. Run the experiments.

```
orchest.run(modelling_strategies=instantiated_models)
```

At this point mleap will train all instantiated estimators on all datasets that were passed to the constructor. The trained models will be saved on the disk.

After this process is finished we can make predictions on the test sets using the trained estimators. in the previous step.

7. Make predictions on the test sets.

```
orchest.predict_all(trained_models_dir='data/trained_models', estimators=instantiated_
↪models)
```

2.2 Analyze Results of Machine Learning Experiments Phase

After the training phase is completed the user can use the predictions on the test sets to run statistical tests.

The starting point is to calculate the prediction error achieved on the test sets. Two main methods are available for this purpose:

- `mleap.analyze_results.analyze_results.calculate_error_all_datasets()`. This method calculates the error achieved by each estimator on each of the datasets and returns the results in a dictionary format.
- `mleap.analyze_results.analyze_results.calculate_error_per_dataset()`. This method calculates the prediction error for each estimator on each datapoint for each dataset.

```
analyze = AnalyseResults(hdf5_output_io=out_io,
                         hdf5_input_io=input_io,
                         input_h5_original_datasets_group='delgado_datasets/',
                         output_h5_predictions_group='experiments/predictions/')
error_all_datasets = analyze.calculate_error_all_datasets(metric='mean_squared_error')
error_per_dataset, error_per_dataset_df = analyze.calculate_error_per_dataset(metric=
↪'mean_squared_error')
```

The output of these two functions can be used as an input to the flowing methods tests:

- `mleap.analyze_results.analyze_results.calculate_average_std()`
- `mleap.analyze_results.analyze_results.cohens_d()`
- `mleap.analyze_results.analyze_results.t_test()`
- `mleap.analyze_results.analyze_results.sign_test()`
- `mleap.analyze_results.analyze_results.t_test_with_bonferroni_correction()`
- `mleap.analyze_results.analyze_results.wilcoxon_test()`
- `mleap.analyze_results.analyze_results.friedman_test()`
- `mleap.analyze_results.analyze_results.nemenyi()`

Please refer to *Analyze Results* for additional information about their use.

2.3 Defining Estimators

`mleap` comes with a number of built-in estimators. However, the user also has the flexibility to define its own by inehriting from the *mleap_estimator* class. The user should at a minimum implement the `save()` and `build()` methods. Please refer to one of the implemented *Estimators* for additional information.

CHAPTER

THREE

MODULES

3.1 Estimators

This module contains the default estimators that come with mleap.

3.1.1 estimators

mleap.estimators.estimators.**instantiate_default_estimators**(*estimators*, *verbose=0*)

instatiates default estimators.

Parameters

- **estimators** (*array of strings*) – Estimator names, family class or type of task.
- **verbose** (*integer*) – The level of output displayed in the terminal. Default is 0 or no output. Higher number means more messages will be printed.

Return type *array of sklearn objects*

3.1.2 baseline_estimators

class mleap.estimators.baseline_estimators.**Baseline_Classifier**(*wrapped*, *wrapper*, *enabled=None*)

Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn dummy classifier class.

build(*strategy='stratified'*)

Builds and returns estimator class.

Parameters **strategy** (*string*) – as per scikit-learn dummy classifier documentation.

save(*dataset_name*)

Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

class mleap.estimators.baseline_estimators.**Baseline_Regressor**(*wrapped*, *wrapper*, *enabled=None*)

Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn dummy regressor

build(*strategy='median'*)
Builds and returns estimator class.

Parameters **strategy** (*string*) – as per scikit-learn dummy regressor documentation.

Return type *sklearn object*

save(*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

3.1.3 bayes_estimators

class mleap.estimators.bayes_estimators.**Bernoulli_Naive_Bayes**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Bernoulli Naive Bayes estimator.

build()
Builds and returns estimator class.

Return type *sklearn object*

save(*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

class mleap.estimators.bayes_estimators.**Gaussian_Naive_Bayes**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Naive Bayes estimator.

build()
Builds and returns estimator class.

Return type *sklearn object*

save(*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

3.1.4 ensemble_estimators

class mleap.estimators.ensemble_estimators.**Bagging_Classifier**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Bagging Classifier.

build(*hyperparameters=None*)
builds and returns estimator

Parameters **hyperparameters** (*dictionary*) – dictionary of hyperparameters to be used for tuning the estimator

Return type *GridsearchCV object*

save(*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

class mleap.estimators.ensemble_estimators.**Bagging_Regressor**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Bagging Regressor.

build(*hyperparameters=None*)
builds and returns estimator

Parameters **hyperparameters** (*dictionary*) – dictionary of hyperparameters to be used for tuning the estimator

Return type *GridsearchCV object*

save(*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

class mleap.estimators.ensemble_estimators.**Gradient_Boosting_Classifier**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Gradient Boosting Classifier.

build(*hyperparameters=None*)
builds and returns estimator

Parameters **hyperparameters** (*dictionary*) – dictionary of hyperparameters to be used for tuning the estimator

Return type *GridsearchCV object*

save(*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

class mleap.estimators.ensemble_estimators.**Gradient_Boosting_Regressor**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Gradient Boosting Regressor.

build(*hyperparameters=None*)
builds and returns estimator

Parameters **hyperparameters** (*dictionary*) – dictionary of hyperparameters to be used for tuning the estimator

Return type *GridsearchCV object*

save (*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

class mleap.estimators.ensemble_estimators.**Random_Forest_Classifier**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Random Forest Classifier.

build (*hyperparameters=None*)
builds and returns estimator

Parameters **hyperparameters** (*dictionary*) – dictionary of hyperparameters to be used for tuning the estimator

Return type *GridsearchCV object*

save (*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

class mleap.estimators.ensemble_estimators.**Random_Forest_Regressor**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Random Forest Regressor.

build (*hyperparameters=None*)
builds and returns estimator

Parameters **hyperparameters** (*dictionary*) – dictionary of hyperparameters to be used for tuning the estimator

Return type *GridsearchCV object*

save (*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

3.1.5 glm_estimators (Generalized Linear Models)

class mleap.estimators.glm_estimators.**Lasso**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Lasso.

build(*hyperparameters=None*)
builds and returns estimator

Parameters **hyperparameters** (*dictionary*) – dictionary of hyperparameters to be used for tuning the estimator

Return type *sklearn object with built-in cross-validationt*

save(*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

class mleap.estimators.glm_estimators.**Lasso_Lars**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Lasso Lars.

build(*hyperparameters=None*)
builds and returns estimator

Parameters **hyperparameters** (*dictionary*) – dictionary of hyperparameters to be used for tuning the estimator

Return type *sklearn object with built-in cross-validation*

save(*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

class mleap.estimators.glm_estimators.**Logistic_Regression**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Logistic Regression.

build(*hyperparameters=None*)
builds and returns estimator

Parameters **hyperparameters** (*dictionary*) – dictionary of hyperparameters to be used for tuning the estimator

Return type *GridsearchCV object*

save(*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

class mleap.estimators.glm_estimators.**Passive_Aggressive_Classifier**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Passive Aggressive Classifier.

build(*hyperparameters=None*)
builds and returns estimator

Parameters **hyperparameters** (*dictionary*) – dictionary of hyperparameters to be used for tuning the estimator

Return type *GridsearchCV object*

save (*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

class mleap.estimators.glm_estimators.**Ridge_Regression** (*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn Ridge Regression.

build (*hyperparameters=None*)
builds and returns estimator

Parameters **hyperparameters** (*dictionary*) – dictionary of hyperparameters to be used for tuning the estimator

Return type *sklearn object with built-in cross-validation*

save (*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

3.1.6 nn_estimators (Neural Network)

class mleap.estimators.nn_estimators.**Deep_NN_Classifier** (*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for a keras sequential model.

build (*num_classes*, *input_dim*, *num_samples*, *loss='mean_squared_error'*, *learning_rate=0.001*, *hyperparameters=None*)
builds and returns estimator

Parameters

- **num_classes** (*int*) – number of classes in datset
- **input_dim** (*int*) – number of features in dataset.
- **num_samples** (*int*) – number of samples in dataset.
- **loss** (*string*) – loss metric as per keras documentation.
- **learning_rate** (*float*) – learning rate for training the neural network.
- **hypehyperparameters** (*dictionary*) – dictionary used for tuning the network if Gridsearch is used.

Return type *keras object*

load (*path_to_model*)
Loads saved keras model from disk.

Parameters **path_to_model** (*string*) – path on disk where the object is saved.

save(*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

class mleap.estimators.nn_estimators.**Deep_NN_Regressor**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for a keras sequential model.

build(*input_dim*, *num_samples*, *loss='mean_squared_error'*, *learning_rate=0.001*, *hyperparameters=None*)
builds and returns estimator

Parameters

- **input_dim** (*int*) – number of features in dataset.
- **num_samples** (*int*) – number of samples in dataset.
- **loss** (*string*) – loss metric as per keras documentation.
- **learning_rate** (*float*) – learning rate for training the neural network.
- **hypehyperparameters** (*dictionary*) – dictionary used for tuning the network if Gridsearch is used.

Return type *keras object*

load(*path_to_model*)
Loads saved keras model from disk.

Parameters **path_to_model** (*string*) – path on disk where the object is saved.

save(*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

3.1.7 svm_estimators

class mleap.estimators.svm_estimators.**SVC_mleap**(*wrapped*, *wrapper*, *enabled=None*)
Bases: *mleap.estimators.mleap_estimator.MleapEstimator*

Wrapper for sklearn SVC estimator.

build(*hyperparameters=None*)
builds and returns estimator

Parameters **hyperparameters** (*dictionary*) – dictionary of hyperparameters to be used for tuning the estimator

Return type *GridsearchCV object*

save(*dataset_name*)
Saves estimator on disk.

Parameters **dataset_name** (*string*) – name of the dataset. Estimator will be saved under default folder structure */data/trained_models/<dataset name>/<model name>*

3.1.8 mleap_estimator

class mleap.estimators.mleap_estimator.**MleapEstimator**(*verbose=0*, *n_jobs=-1*, *num_cv_folds=3*, *refit=True*)

Bases: abc.ABC

Abstact base class that all mleap estimators should inherit from.

build()
Abstract method that needs to be implemented by all estimators. It should return an estimator object.

get_trained_model()
Getter method.

Return type *estimator object*

load(*path_to_model*)
Loads trained estimator from disk. The default implemented method loads a sklearn estimator from a pickle file. This method needs to be overwritten in the child estimator class if another framework/procedure for saving/loading is used.

Parameters **path_to_model** (*string*) – Location of the trained estimator.

save()
Abstract method that needs to be implemented by all estimators. Saves the trained model to disk.

set_trained_model(*trained_model*)
setter method for storing trained estimator in memory

Parameters **trained_model** (*estimator object*) – Trained sklearn, keras, etc. estimator object.

class mleap.estimators.mleap_estimator.**properties**(*estimator_family*, *tasks*, *name*)
Bases: object

Decorator class used for adding properties to mleap estimator classes. The properties that all mleap estimator objects must have are: estimator family, task (classification, regression), name of estimator.

3.2 Experiments

This module orchestrates the process of training the estimators on the datasets. It contains two main loops: over all estimators and over all datasets.

3.2.1 orchestrator

class mleap.experiments.orchestrator.**Orchestrator**(*hdf5_input_io*, *hdf5_output_io*, *dts_names*, *original_datasets_group_h5_path*, *experiments_predictions_group='experiments/predictions'*, *experiments_trained_models_dir='data/trained_models'*)

Bases: object

Orchestrates the sequencing of running the machine learning experiments.

Parameters

- **hdf5_input_io** (*FilesIO()*) – instance of *FilesIO()* with reference to the input file
- **hdf5_output_io** (*FilesIO()*) – instance of *FilesIO()* with reference to the output file
- **dts_names** (*array of strings*) – array with the names of the datasets on which experiments will be run.
- **experiments_predictions_group** (*string*) – path in HDF5 database where predictions will be saved
- **experiments_trained_models_dir** (*string*) – folder on disk where trained estimators will be saved.

predict_all (*trained_models_dir*, *estimators*)
Make predictions on test sets

Parameters

- **trained_models_dir** (*string*) – directory where the trained models are saved
- **estimators** (array of *mleap_estimator* objects) – *mleap_estimator* objects. The trained models are set as a property to the object.

run (*modelling_strategies*)
Main module for training the estimators. The inputs of the function are:

1. The input and output databse files containing the datasets.
2. The instantiated estimators.

The method iterates through all datasets in the input database file and trains all modelling strategies on these datasets. At the end of the process we make predictions on the test sets and store them in the output database file.

The class uses helper methods in the experiments class to avoid too many nested loops.

Parameters **modelling_strategies** (array of *mleap_estimator* objects) – array of estimators that will be used for training

3.3 Data

Contains interface methods for certain *files_io* methods.

3.3.1 data

class mleap.data.data.**Data** (*experiments_predictions_group='experiments/predictions'*, *split_datasets_group='split_dts_idx'*, *train_idx='train_idx'*, *test_idx='test_idx'*)
Bases: object

Interface class expanding the functionality of *FilesIO()*

Parameters

- **split_datasets_group** (*string*) – location in HDF5 database were the test/train split will be saved.
- **train_idx** (*string*) – name of group where the indexes of the samples used for training are saved

- **test_idx** (*string*) – name of group where the indexes of the samples used for testing are saved

list_datasets(*hdf5_group*, *hdf5_io*)
Returns sub group in parent HDF5 group.

Parameters

- **hdf5_group** (*string*) – Path to HDF5 parent group of which we are quering the subgroups.
- **hdf5_io** (*FilesIO()*) – Instance of *FilesIO()*

Return type tuple with array with dataset names and array with full path to datasets.

load_test_train_dts(*hdf5_out*, *hdf5_in*, *dts_name*, *dts_grp_path*)
Loads test/train data.

Parameters

- **hdf5_out** (*FilesIO()* object) – instance of *FilesIO()* object with output data.
- **hdf5_in** (*FilesIO()* object) – instance of *FilesIO()* object with input data.
- **dts_name** (*string*) – name of dataset for which the splits will be loaded.
- **dts_grp_path** (*string*) – path to root group where the original datasets are stored.

Return type tuple arrays in the form: X_train, X_test, y_train, y_test where X are the features and y are the lables.

load_train_test_split(*hdf5_out*, *dataset_name*)
Loads test train split form HDF5 database.

Parameters

- **hdf5_out** (*FilesIO()* object) – *FilesIO()* output object where the test/train index splits are stored.
- **dataset_name** (*string*) – name of dataset for which the splits will be loaded.

Return type tuple with train indices, test indices, train metadata and test metadata.

load_true_labels(*hdf5_in*, *dataset_loc*, *lables_idx*)
Loads labels for dataset

Parameters

- **hdf5_in** (*FilesIO()* object) – instance of *FilesIO()* object containing the original datasets
- **lables_idx** (*string*) – path the dataset.

Return type pandas DataFrame

open_hdf5(*hdf5_path*, *mode='a'*)

Parameters

- **hdf5_path** (*string*) – path to HDF5 file saved on disk.
- **mode** (*string*) – open and create file modes as per the h5py documentation.

pandas_to_db(*save_loc_hdf5*, *datasets*, *dts_metadata*, *input_io*)
Saves array of datasets in pandas DataFrame format in HDf5 Database. This represents an interface method for *save_datasets()*

Parameters

- **save_loc_hdf5** (*string*) – Root group in HDF5 database where the datasets will be saved.
- **datasets** (*array of pandas DataFrame*) – array of datasets formatted as pandas DataFrame.
- **dts_meta** (*array of dictionaries*) – Metadata for each dataset.
- **input_io** (*FilesIO()*) – Instance of *FilesIO()* class.

split_datasets(*hdf5_in*, *hdf5_out*, *dataset_paths*, *test_size=0.33*, *random_state=1*, *verbose=False*)

Splits datasets in test and train sets.

Parameters

- **hdf5_in** (*FilesIO()* object) – *FilesIO()* object where the original/input datasets are stored.
- **hdf5_out** (*FilesIO()* object) – *FilesIO()* object where the split/output test/train indices are stored.
- **dataset_paths** (*array of strings*) – full path to each dataset stored in the origial/input HDF5 database that will be split to test/train.
- **test_size** (*float*) – percentage of samples to be put in the test set.
- **random_state** (*integer*) – random state for test/train split.
- **verbose** (*boolean*) – if True prints progress messages in terminal.

Return type array of strings containing locations of split datasets.

3.4 Analyze Results

This modules contains methods for analyzing the results produced by the trained estimators and benchmarking their performance.

3.4.1 analyze_results

class mleap.analyze_results.analyze_results.**AnalyseResults**(*hdf5_output_io*, *hdf5_input_io*, *input_h5_original_datasets_group*, *output_h5_predictions_group*, *split_datasets_group='split_dts_idx'*, *train_idx='train_idx'*, *test_idx='test_idx'*)

Bases: object

Analyze results of machine learning experiments.

Parameters

- **hdf5_input_io** (*FilesIO()*) – Instance of *FilesIO()* class.
- **hdf5_input_io** – Instance of *FilesIO()* class.
- **input_h5_original_datasets_group** (*string*) – location in HDF5 database where the original datasets are stored.

- **output_h5_predictions_group** (*string*) – location in HDF5 where the prediction of the estimators will be saved.
- **split_datasets_group** (*string*) – location in HDF5 database where the test/train splits are saved.
- **train_idx** (*string*) – name of group where the train split index will be stored.
- **test_idx** (*string*) – name of group where the test split index will be stored.

calculate_average_std(*scores_dict*)
Calculates simple average and standard deviation.

Parameters **scores_dict** (*dictionary*) – Dictionary with estimators (keys) and corresponding prediction accuracies on different datasets.

Return type pandas DataFrame

calculate_error_all_datasets(*metric*)
Calculates the prediction error for each estimator on all test splits.

Parameters **metric** (*string*) – Loss metric. Supported values are: `accuracy`, `mean_squared_error`

Return type dictionary with keys representing the name of the estimator and values representing the loss achieved on each dataset.

calculate_error_per_dataset(*metric*)
Calculates the prediction error for each estimator on each datapoint for each dataset.

Parameters **metric** (*string*) – Loss metric. Supported values are: `accuracy`, `mean_squared_error`

Return type dictionary

cohens_d(*estimator_dict*)
Cohen's d is an effect size used to indicate the standardised difference between two means. The calculation is implemented natively (without the use of third-party libraries). More information can be found here: Cohen's d.

Parameters **estimator_dict** (*dictionay*) – dictionay with keys *names of estimators* and values *errors achieved by estimators on test datasets*.

Return type pandas DataFrame.

friedman_test(*observations*)
The Friedman test is a non-parametric statistical test used to detect differences in treatments across multiple test attempts. The procedure involves ranking each row (or block) together, then considering the values of ranks by columns. Implementation used: scipy.stats.

Parameters **observations** (*dictionary*) – Dictionary with errors on test sets achieved by estimators.

Return type tuple of dictionary, pandas DataFrame.

nemenyi(*obeservations*)
Post-hoc test run if the *friedman_test* reveals statistical significance. For more information see Nemenyi test. Implementation used scikit-posthocs.

Parameters **observations** (*dictionary*) – Dictionary with errors on test sets achieved by estimators.

Return type pandas DataFrame.

sign_test (*observations*)
Non-parametric test for testing consistent differences between pairs of obeservations. The test counts the number of observations that are greater, smaller and equal to the mean https://en.wikipedia.org/wiki/Sign_test.

Parameters **observations** (*dictionary*) – Dictionary with errors on test sets achieved by estimators.

Return type tuple of dictionary, pandas DataFrame

t_test (*observations*)
Runs t-test on all possible combinations between the estimators.

Parameters **observations** (*dictionary*) – Dictionary with errors on test sets achieved by estimators.

Return type tuple of dictionary, pandas DataFrame

t_test_with_bonferroni_correction (*observations*, *alpha=0.05*)
correction used to counteract multiple comparissons https://en.wikipedia.org/wiki/Bonferroni_correction

Parameters

- **observations** (*dictionary*) – Dictionary with errors on test sets achieved by estimators.
- **alpha** (*float*) – confidence level.

Return type tuple of dictionary, pandas DataFrame

wilcoxon_test (*observations*)
Wilcoxon signed-rank test. Tests whether two related paired samples come from the same distribution. In particular, it tests whether the distribution of the differences x-y is symmetric about zero

Parameters **observations** (*dictionary*) – Dictionary with errors on test sets achieved by estimators.

Return type tuple of dictionary, pandas DataFrame.

3.5 Shared

This module provides functionality that is used in the other modules. It contains prodominantly interface methods for working with HDF5 databases as well as mehtods for manipulating files on the hard drive.

3.5.1 files_io

class mleap.shared.files_io.**DiskOperations**
Bases: object

Class with high level disk operations for loading and saving estimators.

check_path_exists (*path_to_file*)
Checks whether path exists on disk

Parameters **path_to_file** (*string.*) – path to check whether exists or not

Return type *Boolean*

create_directory_on_hdd (*directory_path*)
Creates directory on hard drive

Parameters **directory_path** (*string.*) – path of directory that will be created.

save_keras_model (*trained_model*, *model_name*, *dataset_name*, *root_dir='data/trained_models'*)
Saves keras object to disk.

Parameters

- **trained_model** (*keras estimator object.*) – trained keras object to be saved on disk.
- **model_name** (*string*) – name of sklearn estimator.
- **dataset_name** (*string*) – name of dataset that the estimator was trained on.
- **root_dir** (*string*) – root dir where the trained estimators will be saved.

save_to_pickle (*trained_model*, *model_name*, *dataset_name*, *root_dir='data/trained_models'*)
Saves sklearn estimator to disk as pickle file.

Parameters

- **trained_model** (*sklearn estimator object.*) – trained sklearn object to be saved on disk.
- **model_name** (*string*) – name of sklearn estimator.
- **dataset_name** (*string*) – name of dataset that the estimator was trained on.
- **root_dir** (*string*) – root dir where the trained estimators will be saved.

class mleap.shared.files_io.**FilesIO**(*hdf5_filename*, *mode='a'*, *experiments_predictions_group='experiments/predictions'*)

Bases: object

Methods for manipulating HDF5 databases and datasets.

Parameters

- **hdf5_filename** (*string*) – full path where the database file will be stored.
- **mode** (*string*) – open and create file modes as per the h5py documentation.
- **experiments_predictions_group** (*string*) – Location in HDF5 database where estimator predictions will be saved.

check_h5_path_exists (*path_to_check*)
Checks whether path/group exists in HDF5 database

Parameters **path_to_check** (*string*) – path of group that will be checked.

Return type *boolean*

list_datasets (*hdf5_group*)
Lists all datasets/sub-groups in an HDF5 group

Parameters **hdf5_group** (*string*) – path to HDf5 group

Return type *array of strings*

load_dataset_h5 (*dataset_path*)
Loads dataset from HDF5 database. The dataset needs to have been saved as a numpy array originally

Parameters **dataset_path** (*string*) – path to dataset.

Return type *numpy array, metadata dictionary*

load_dataset_pd (*dataset_path*)
Loads dataset from HDF5 database. The dataset needs to have been saved as a pandas Datafram originally

Parameters **dataset_path** (*string*) – path to dataset.

Return type *pandas DataFrame, metadata dictionary*

load_predictions_for_dataset(*dataset_name*)
Loads predictions generated my trained estimator models.

Parameters **dataset_name** (*string*) – Name of dataset on which estimators were trained

Return type *numpy array in the form [[strategy name][predictions]]*

save_array_hdf5(*group*, *datasets*, *array_names*, *array_meta*)
Saves an array of data to HDF5 database

:type group:string :param group: path to group in HDF5 where the arrays will be saved

Parameters

- **datasets** (*array of arrays*) – array in the form [[x_1],[x_2],...[x_n]] representing the data that will be saved. Each x_i is saved in a different sub-group under the common parent HDF5 group.
- **array_names** (*array of strings*) – name of sub-groups where the dataset arrays will be saved
- **array_meta** (*array of dictionaries*) – metadata for each sub-group

save_datasets(*datasets*, *datasets_save_paths*, *dts_metadata*, *verbose=False*)
saves datasets in HDF5 database. dataset_names must contain full path.

Parameters

- **datasets** (*array of pandas DataFrame*) – array of datasets formatted as pandas DataFrame.
- **datasets_save_paths** (*array of string*) – Array with the save locations for each dataset.
- **dts_meta** (*array of dictionaries*) – Metadata for each dataset.
- **verbose** (*boolean*) – Display or not progress with saving the datasets.

save_ml_strategy_timestamps(*timestamps_df*, *dataset_name*)
Saves start and end times for training estimators.

Parameters

- **timestamps_df** (*DataFrame*) – Dataframe containing: [strategy_name, begin_timestamp, end_timestamp, difference between start and end timestamp]
- **dataset_name** (*string*) – name of dataset on which the estimator was trained.

save_prediction_to_db(*predictions*, *dataset_name*, *strategy_name*)
Saves the prediction of a single trained estimator in HDF5 database.

Parameters

- **predictions** (*numpy array*) – array with predictions
- **dataset_name** (*string*) – name of dataset on which the estimator was trained
- **strategy_name** (*string*) – name of estimator/strategy

save_predictions_to_db(*predictions*, *dataset_name*)
Bulk save of predictions generated by a several trained estimators.

Parameters

- **predictions** (*array*) – array in form [[estimator name][predictions]]
- **dataset_name** (*string*) – Name of dataset on which the estimator was trained.

split_dataset(*dataset_path*, *test_size=0.33*)

Splits dataset in train and test set.

Parameters

- **dataset_path** (*string*) – Path to dataset
- **test_size** (*float*) – Fraction of samples that will be put in test set

Return type tuple of four arrays (X_train, X_test, y_train, y_test)

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX

T

W