
MAterials Simulation Toolkit for Machine Learning (MAST-ML) Documentation

Release 2.0

University of Wisconsin-Madison Computational Materials Group

Apr 13, 2021

Contents

1	Acknowledgements	1
2	Installing MAST-ML	3
2.1	Hardware and Data Requirements	3
2.2	Terminal installation (Linux or linux-like terminal on Mac)	3
2.3	Windows installation	4
2.4	Startup	6
3	MAST-ML Input File	9
3.1	Input file sections	9
4	MAST-ML overview slides	21
5	Running MAST-ML on Google Colab	31
6	MAST-ML tutorial	33
6.1	Introduction	33
6.2	Your first MAST-ML run	33
6.3	Cleaning input data	35
6.4	Feature generation and normalization	36
6.5	Training and evaluating your first model	37
6.6	Feature selection and learning curves	40
6.7	Hyperparameter optimization	44
6.8	Random leave-out versus leave-out-group cross-validation	46
6.9	Making predictions by importing a previously fit model	51
6.10	Predicting values for new, extrapolated data	53
7	Code Documentation: Metrics	57
7.1	mastml.metrics Module	57
8	Code Documentation: Configuration file parser	59
8.1	mastml.conf_parser Module	59
9	Code Documentation: Data cleaner	61
9.1	mastml.data_cleaner Module	61
10	Code Documentation: Data loader	65
10.1	mastml.data_loader Module	65

11 Code Documentation: Learning curve	67
11.1 mastml.learning_curve Module	67
12 Code Documentation: Clusterers	69
12.1 mastml.legos.clusterers Module	69
13 Code Documentation: Data splitters	71
13.1 mastml.legos.data_splitters Module	71
14 Code Documentation: Utils	77
14.1 mastml.utils Module	77
15 Code Documentation: MAST-ML Driver	83
15.1 mastml.mastml_driver Module	83
16 Code Documentation: Plot Helper	87
16.1 mastml.plot_helper Module	87
17 Code Documentation: HTML Helper	105
17.1 mastml.html_helper Module	105
18 Code Documentation: Feature Selectors	109
18.1 mastml.legos.feature_selectors Module	109
19 Code Documentation: Feature Normalizers	115
19.1 mastml.legos.feature_normalizers Module	115
20 Code Documentation: Randomizers	119
20.1 mastml.legos.randomizers Module	119
21 Code Documentation: Model Finder	121
21.1 mastml.legos.model_finder Module	121
22 Code Documentation: Utility Legos	127
22.1 mastml.legos.util_legos Module	127
23 Code Documentation: Feature Generators	131
23.1 mastml.legos.feature_generators Module	131
24 Indices and tables	143
Python Module Index	145
Index	147

Acknowledgements

Materials Simulation Toolkit for Machine Learning (MAST-ML)

MAST-ML is an open-source Python package designed to broaden and accelerate the use of machine learning in materials science research

Contributors

University of Wisconsin-Madison Computational Materials Group:

- Prof. Dane Morgan
- Dr. Ryan Jacobs
- Dr. Tam Mayeshiba
- Ben Afflerbach
- Dr. Henry Wu

University of Kentucky contributors:

- Luke Harold Miles
- Robert Max Williams
- Prof. Raphael Finkel

MAST-ML documentation:

An overview of code documentation and tutorials for getting started with MAST-ML can be found [here](#)

Funding

This work was and is funded by the National Science Foundation (NSF) SI2 award No. 1148011 and DMREF award number DMR-1332851

Citing MAST-ML

If you find MAST-ML useful, please cite the following publication:

Jacobs, R., Mayeshiba, T., Afflerbach, B., Miles, L., Williams, M., Turner, M., Finkel, R., Morgan, D., “The Materials Simulation Toolkit for Machine Learning (MAST-ML): An automated open source toolkit to accelerate data- driven

materials research”, Computational Materials Science 175 (2020), 109544. <https://doi.org/10.1016/j.commatsci.2020.109544>

Code Repository

MAST-ML is available on PyPi: `pip install mastml`

MAST-ML is available on *Github* <<https://github.com/uw-cmg/MAST-ML>>

`git clone --single-branch master https://github.com/uw-cmg/MAST-ML`

2.1 Hardware and Data Requirements

2.1.1 Hardware

PC or Mac computer capable of running python 3.

2.1.2 Data

- Numeric data file in the form of .csv or .xlsx file. There must be at least some target feature data, so that models can be fit.
- First row of file (each column) should have a text name (as string) which is how columns will be referenced later in the input file.
- If working in Jupyter environment, can also directly pass in a pandas dataframe

2.2 Terminal installation (Linux or linux-like terminal on Mac)

2.2.1 Install Python3

Install Python 3: for easier installation of numpy and scipy dependencies, download Anaconda from <https://www.continuum.io/downloads>

Create a conda environment

Create an environment:

```
conda create --name MAST_ML python=3.7
conda activate MAST_ML
pip install mastml
```

Set up Jupyter notebooks

There is no separate setup for Jupyter notebooks necessary; once MASTML has been run and created a notebook, then in the terminal, navigate to a directory housing the notebook and type:

```
jupyter notebook
```

and a browser window with the notebook should appear.

2.2.2 Install the MAST-ML package

Pip install MAST-ML from PyPi:

```
pip install mastml
```

Alternatively, git clone the Github repository, for example:

```
git clone https://github.com/uw-cmg/MAST-ML
```

Clone from “master” unless instructed specifically to use another branch. Ask for access if you cannot find this code.

Check status.github.com for issues if you believe github may be malfunctioning

Run:

```
python setup.py install
```

Imports that don't work

First try anaconda install, and if that gives errors try pip install Example: `conda install numpy` , or `pip install numpy`
Put the path to the installed MAST-ML folder in your PYTHONPATH if it isn't already

2.3 Windows installation

2.3.1 Install Python3

Install Python 3: for easier installation of numpy and scipy dependencies, download anaconda from <https://www.continuum.io/downloads>

Create a conda environment

From the Anaconda Navigator, go to Environments and create a new environment Select python version **3.6**

Under “Channels”, along with defaults channel, “Add” the “materials” channel. The Channels list should now read:

```
defaults
materials
```

(may be the “matsci” channel instead of the “materials” channel; this channel is used to install pymatgen)

Set up the Spyder IDE and Jupyter notebooks

From the Anaconda Navigator, go to Home With the newly created environment selected, click on “Install” below Jupyter. Click on “Install” below Spyder.

Once the MASTML has been run and has created a jupyter notebook (run MASTML from a location inside the anaconda environment, so that the notebook will also be inside the environment tree), from the Anaconda Navigator, go to Environments, make sure the environment is selected, press the green arrow button, and select Open jupyter notebook.

2.3.2 Install the MAST-ML package

Pip install MAST-ML from PyPi:

```
pip install mastml
```

Alternatively, git clone the Github repository, for example:

```
git clone https://github.com/uw-cmg/MAST-ML
```

Clone from “master” unless instructed specifically to use another branch. Ask for access if you cannot find this code.

Check status.github.com for issues if you believe github may be malfunctioning

Run:

```
python setup.py install
```

Imports that don’t work

First try anaconda install, and if that gives errors try pip install Example: conda install numpy , or pip install numpy Put the path to the installed MAST-ML folder in your PYTHONPATH if it isn’t already

2.3.3 Windows 10 install: step-by-step guide (credit Joe Kern)

First, figure out if your computer is 32 or 64-bit. Type “system information” in your search bar. Look at system type. x86 is a 32-bit computer, x64 is a 64-bit.

Second, download an environment manager. Environments are directories in your computer that store dependencies. For instance, one program you run might be dependent on version 1.0 of another program x. However, another program you have might be dependent on version 2.0 of program x. Having multiple environments allows you utilize both programs and dependencies on your computer. I will recommend you download anaconda, not because it is the best, but because it is an environment manager I know how to get working with MAST-ML. Feel free to experiment with other managers. Download the Python 3.7 version at <https://www.anaconda.com/distribution/>, just follow the installation instructions. Pick the graphical installer that corresponds with your computer system (64 bit or 32 bit).

Third, download Visual studio. Some of the MAST-ML dependencies require C++ distributables in order to run. Visual Studio Code is a code editor made for Windows 10. The dependencies for MAST-ML will look in the Visual

Studio Code folder for these C++ distributables when they download. There may be another way to download these these C++ distributables without Visual Studio Code, but I am not sure how to do that. Go here to download <https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2017>

Fourth, download Visual Studio with C++ build tools and restart the computer

Fifth, Open anaconda navigator. Click Environments and create at the bottom. Name it MASTML and make it Python 3.6. DO NOT MAKE IT Python 3.7 or Python version 2.6 or 2.7. Some dependencies do not work with those other version.

Sixth, click the arrow next to your environment name and open a command shell. In the command line type “pip install “ and then copy paste the dependency names from the dependency file into your command prompt.

Seventh, test if MAST-ML runs. There are multiple ways to do this, but I will outline one. Navigate to your MAST-ML folder in the command prompt. To do this, you need to know the command ‘cd’. Typing ‘cd’ will let you change the directory you command prompt is operating in. In order to navigate to your mast-ml folder, right click the folder and click properties. Copy the location and in the command prompt type ‘cd’ and paste the location after. Add a ‘Mast-ml’ or whatever your folder is called to the end of the pasted value so you can get to mastml

Finally, copy paste `python -m mastml.mastml_driver mastml/tests/conf/example_input.conf mastml/tests/csv/example_data.csv -o results/mastml_tutorial` into your command prompt and run. If it all works, you’re good to go.

2.4 Startup

2.4.1 Locate the examples folder

In the installed MASTML directory, navigate to the `tests` folder.

Under `tests/conf`, The file `example_input.conf` will use the `example_data.xlsx` data file located in `tests/csv` to run an example.

2.4.2 Run the MASTML command

The format is `python3 -m mastml.mastml_driver <path to config file> <path to data .xlsx file> -o <path to results folder>`

For example, to conduct the test run above, while in the MASTML install directory:

```
python3 -m mastml.mastml_driver tests/conf/example_input.conf tests/csv/example_data.  
↪xlsx -o results/example_results
```

This is a terminal command. For Windows, assuming setup has been followed as above, go to the Anaconda Navigator, Environments, select the environment, click the green arrow button, and Open terminal.

When you execute the above command, you’ll know it’s working if you begin to see output on your screen.

2.4.3 Check output

`index.html` should be created, linking to certain representative plots for each test

For this example, output will be located in subfolders in the `results/example_results` folder.

Check the following to see if the run completed successfully:

```
A log.log file is generated and the last line contains the phrase "Making html file_
↳of all run stats..."
An index.html file that gives some summary plots from all the tests that were run
A series of subfolders with names "StandardScaler"->"DoNothing"->"KernelRidge", with
↳the following three directories
within the "KernelRidge" directory: "LeaveOneGroupOut_host", "NoSplit", and
↳"RepeatedKFold"
```

You can compare all of these files with those given in the /example_results directory which should match.

MAST-ML Input File

This document provides an overview of the various sections and fields of the MAST-ML input file.

A full template input file can be downloaded here: [MASTML_InputFile](#)

3.1 Input file sections

3.1.1 General Setup

The “GeneralSetup” section of the input file allows the user to specify an assortment of basic MAST-ML parameters, ranging from which column names in the .xlsx file to use as features for fitting (i.e. X data) or to fit to (i.e. y data), as well as which metrics to employ in fitting a model, among other things.

Example:

```
[GeneralSetup]
    input_features = feature_1, feature_2, etc. or "Auto"
    input_target = target_feature
    randomizer = False
    metrics = root_mean_squared_error, mean_absolute_error, etc. or "Auto"
    input_other = additional_feature_1, additional_feature_2
    input_grouping = grouping_feature_1
    input_testdata = validation_feature_1
```

- **input_features** List of input X features
- **input_target** Target y feature
- **randomizer** Whether or not to randomize y feature data. Useful for establishing a null “baseline” test
- **metrics** Which metrics to evaluate model fits
- **input_other** Additional features that are not to be fitted on (i.e. not X features)
- **input_grouping** Feature names that provide information on data grouping

- **input_test** Feature name that designates whether data will be used for validation (set rows as 1 or 0 in csv file)

3.1.2 Data Cleaning

The “DataCleaning” section of the input file allows the user to clean their data to remove rows or columns that contain empty or NaN fields, or fill in these fields using imputation or principal component analysis methods.

Example:

```
[DataCleaning]
  cleaning_method = remove, imputation, ppca
  imputation_strategy = mean, median
```

- **cleaning_method** Method of data cleaning. “remove” simply removes columns with missing data. “imputation” uses basic operation to fill in missing values. “ppca” uses principal component analysis to fill in missing values.
- **imputation_strategy** Only valid field if doing imputation, selects method to impute missing data by using mean, median, etc. of the column

3.1.3 Clustering

Optional section to perform clustering of data using well-known clustering algorithms available in scikit-learn. Note that the subsection names must match the corresponding name of the routine in scikit-learn. More information on clustering routines and the parameters to set for each routine can be found here: <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster> For the purpose of this full input file, we use the scikit-learn default parameter values. Note that not all parameters are listed.

Example:

```
[Clustering]
  [[AffinityPropagation]]
    damping = 0.5
    max_iter = 200
    convergence_iter = 15
    affinity = euclidean
  [[AgglomerativeClustering]]
    n_clusters = 2
    affinity = euclidean
    compute_full_tree = auto
    linkage = ward
  [[Birch]]
    threshold = 0.5
    branching_factor = 50
    n_clusters = 3
  [[DBSCAN]]
    eps = 0.5
    min_samples = 5
    metric = euclidean
    algorithm = auto
    leaf_size = 30
  [[KMeans]]
    n_clusters = 8
    n_init = 10
    max_iter = 300
    tol = 0.0001
  [[MiniBatchKMeans]]
```

(continues on next page)

(continued from previous page)

```
n_clusters = 8
max_iter = 100
batch_size = 100
[[MeanShift]]
[[SpectralClustering]]
n_clusters = 8
n_init = 10
gamma = 1.0
affinity = rbf
```

3.1.4 Feature Generation

Optional section to perform feature generation based on properties of the constituent elements. These routines were custom written for MAST-ML, except for PolynomialFeatures. For more information on the MAST-ML custom routines, consult the MAST-ML online documentation. For more information on PolynomialFeatures, see: <http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html>

Example:

```
[FeatureGeneration]
[[Magpie]]
composition_feature = Material Compositions
feature_types = composition_avg, arithmetic_avg, max, min, difference
[[MaterialsProject]]
composition_feature = Material Compositions
api_key = my_api_key
[[Citrine]]
composition_feature = Material Compositions
api_key = my_api_key
[[ContainsElement]]
composition_feature = Host element
all_elements = False
element = Al
new_name = has_Al
[[PolynomialFeatures]]
degree=2
interaction_only=False
include_bias=True
```

- **composition_feature** Name of column in csv file containing material compositions
- **feature_types** Types of elemental features to output. If None is specified, all features are output. Note “elements” refers to properties of constituent elements
- **api_key** Your API key to access the Materials Project or Citrine. Register for your account at Materials Project: <https://materialsproject.org> or at Citrine: <https://citrine.com>
- **all_elements** For ContainsElement, whether or not to scan all data rows to assess all elements present in data set
- **element** For ContainsElement, name of element of interest. Ignored if all_elements = True
- **new_name** For ContainsElement, name of new feature column to generate. Ignored if all_elements = True

3.1.5 Feature Normalization

Optional section to perform feature normalization of the input or generated features using well-known feature normalization algorithms available in scikit-learn. Note that the subsection names must match the corresponding name of the routine in scikit-learn. More information on normalization routines and the parameters to set for each routine can be found here: <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing> . For the purpose of this full input file, we use the scikit-learn default parameter values. Note that not all parameters are listed, and only the currently listed normalization routines are supported. In addition, MeanStdevScaler is a custom written normalization routine for MAST-ML. Additional information on MeanStdevScaler can be found in the online MAST-ML documentation.

Example:

```
[FeatureNormalization]
  [[Binarizer]]
    threshold = 0.0
  [[MaxAbsScaler]]
  [[MinMaxScaler]]
  [[Normalizer]]
    norm = l2
  [[QuantileTransformer]]
    n_quantiles = 1000
    output_distribution = uniform
  [[RobustScaler]]
    with_centering = True
    with_scaling = True
  [[StandardScaler]]
  [[MeanStdevScaler]]
    mean = 0
    stdev = 1
```

3.1.6 Learning Curve

Optional section to perform learning curve analysis on a dataset. Two types of learning curves will be generated: a data learning curve (score vs. amount of training data) and a feature learning curve (score vs. number of features).

Example:

```
[LearningCurve]
  estimator = KernelRidge_learn
  cv = RepeatedKFold_learn
  scoring = root_mean_squared_error
  n_features_to_select = 5
  selector_name = MASTMLFeatureSelector
```

- **estimator** A scikit-learn model/estimator. The name needs to match an entry in the [Models] section. Note this model will be removed from the [Models] list after the learning curve is generated.
- **cv** A scikit-learn cross validation generator. The name needs to match an entry in the [DataSplits] section. Note this method will be removed from the [DataSplits] list after the learning curve is generated.
- **scoring** A scikit-learn scoring method compatible with MAST-ML. See the MAST-ML online documentation at https://htmlpreview.github.io/?https://raw.githubusercontent.com/uw-cmg/MAST-ML/dev_Ryan_2018-10-29/docs/build/html/3_metrics.html for more information.
- **n_features_to_select** The max number of features to use for the feature learning curve.

- **selector_name** Method to conduct feature selection for the feature learning curve. The name needs to match an entry in the [FeatureSelection] section. Note this method will be removed from the [FeatureSelection] section after the learning curve is generated.

3.1.7 Feature Selection

Optional section to perform feature selection using routines in scikit-learn, mlxtend and custom-written for MAST-ML. Note that the subsection names must match the corresponding name of the routine in scikit-learn. More information on selection routines and the parameters to set for each routine can be found here: http://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_selection . For the purpose of this full input file, we use the scikit-learn default parameter values. Note that not all parameters are listed, and only the currently listed selection routines are supported. In addition, MASTMLFeatureSelector is a custom written selection routine for MAST-ML. Additional information on MASTMLFeatureSelector can be found in the online MAST-ML documentation. Finally, SequentialFeatureSelector is a routine available from the mlxtend package, which documentation can be found here: <http://rasbt.github.io/mlxtend/>

Example:

```
[FeatureSelection]
  [[GenericUnivariateSelect]]
  [[SelectPercentile]]
  [[SelectKBest]]
  [[SelectFpr]]
  [[SelectFdr]]
  [[SelectFwe]]
  [[RFE]]
    estimator = RandomForestRegressor_selectRFE
    n_features_to_select = 5
    step = 1
  [[SequentialFeatureSelector]]
    estimator = RandomForestRegressor_selectSFS
    k_features = 5
  [[RFECV]]
    estimator = RandomForestRegressor_selectRFECV
    step = 1
    cv = LeaveOneGroupOut_selectRFECV
    min_features_to_select = 1
  [[SelectFromModel]]
    estimator = KernelRidge_selectfrommodel
    max_features = 5
  [[VarianceThreshold]]
    threshold = 0.0
  [[PCA]]
    n_components = 5
  [[MASTMLFeatureSelector]]
    estimator = KernelRidge_selectMASTML
    n_features_to_select = 5
    cv = LeaveOneGroupOut_selectMASTML
    # Any features you want to keep from the start, then use these to_
↪subsequently do forward selection
    manually_selected_features = myfeature_1, myfeature_2
  [[EnsembleModelFeatureSelector]]
    # A scikit-learn model/estimator. Needs to have estimator feature ranking.
↪The name needs to match an entry in the [Models] section.
    estimator = RandomForestRegressor_selectEnsemble
    # number of features to select
    k_features = 5
```

(continues on next page)

(continued from previous page)

```
[[PearsonSelector]]
    # threshold for removal of redundant features
    threshold_between_features = 0.9
    # threshold for removal of features not sufficiently correlated with target
    threshold_with_target = 0.8
    # whether to remove features that are highly correlated with each other (i.e.,
    ↪redundant)
    remove_highly_correlated_features = True
    # number of features to select
    k_features = 5
```

- **estimator** A scikit-learn model/estimator. The name needs to match an entry in the [Models] section. Note this model will be removed from the [Models] list after the learning curve is generated.
- **n_features_to_select** The max number of features to select
- **step** For RFE and RFECV, the number of features to remove in each step
- **k_features** For SequentialFeatureSelector, the max number of features to select.
- **cv** A scikit-learn cross validation generator. The name needs to match an entry in the [DataSplits] section. Note this method will be removed from the [DataSplits] list after the learning curve is generated.

3.1.8 Data Splits

Optional section to perform data splits using cross validation routines in scikit-learn, and custom-written for MAST-ML. Note that the subsection names must match the corresponding name of the routine in scikit-learn. More information on selection routines and the parameters to set for each routine can be found here: http://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection . For the purpose of this full input file, we use the scikit-learn default parameter values. Note that not all parameters are listed, and only the currently listed data split routines are supported. In addition, NoSplit is a custom written selection routine for MAST-ML, which simply produces a full data fit with no cross validation. Additional information on NoSplit can be found in the online MAST-ML documentation.

Example:

```
[DataSplits]
[[NoSplit]]
[[KFold]]
    shuffle = True
    n_splits = 10
[[RepeatedKFold]]
    n_splits = 5
    n_repeats = 10
    # Here, an example of another instance of RepeatedKFold, this one being used in,
    ↪the [LearningCurve] section above.
[[RepeatedKFold_learn]]
    n_splits = 5
    n_repeats = 10
[[GroupKFold]]
    n_splits = 3
[[LeaveOneOut]]
[[LeavePOut]]
    p = 10
[[RepeatedStratifiedKFold]]
    n_splits = 5
    n_repeats = 10
```

(continues on next page)

(continued from previous page)

```
[[StratifiedKFold]]
    n_splits = 3
[[ShuffleSplit]]
    n_splits = 10
[[StratifiedShuffleSplit]]
    n_splits = 10
[[LeaveOneGroupOut]]
    # The column name in the input csv file containing the group labels
    grouping_column = Host element
    # Here, an example of another instance of LeaveOneGroupOut, this one being used
    ↪ in the [FeatureSelection] section above.
[[LeaveOneGroupOut_selectMASTML]]
    # The column name in the input csv file containing the group labels
    grouping_column = Host element
    # Here, an example of another instance of LeaveOneGroupOut, this one being used
    ↪ based on the creation of the "has_Al"
    # group from the [[ContainsElement]] routine present in the [FeatureGeneration]
    ↪ section.
[[LeaveOneGroupOut_Al]]
    grouping_column = has_Al
    # Here, an example of another instance of LeaveOneGroupOut, this one being used
    ↪ based on the creation of clusters
    # from the [[KMeans]] routine present in the [Clustering] section.
[[LeaveOneGroupOut_kmeans]]
    grouping_column = KMeans
[[LeaveCloseCompositionsOut]]
    # Set the distance threshold in composition space
    dist_threshold=0.1
[[Bootstrap]]
    # Data set size
    n = 378
    # Number of bootstrap resamplings to perform
    n_bootstraps = 10
    # Training set size
    train_size = 303
    # Validation/test set size
    test_size = 75
```

3.1.9 Models

Optional section to denote different models/estimators for model fitting from scikit-learn. Note that the subsection names must match the corresponding name of the routine in scikit-learn. More information on different model routines and the parameters to set for each routine can be found here for ensemble methods: <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble> and here for kernel ridge and linear methods: http://scikit-learn.org/stable/modules/classes.html#module-sklearn.kernel_ridge and here for neural network methods: http://scikit-learn.org/stable/modules/classes.html#module-sklearn.neural_network and here for support vector machine and decision tree methods: <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.svm> . For the purpose of this full input file, we use the scikit-learn default parameter values. Note that not all parameters are listed, and only the currently listed data split routines are supported.

Example:

```
[Models]
    # Ensemble methods
```

(continues on next page)

(continued from previous page)

```
[[AdaBoostClassifier]]
    n_estimators = 50
    learning_rate = 1.0
[[AdaBoostRegressor]]
    n_estimators = 50
    learning_rate = 1.0
[[BaggingClassifier]]
    n_estimators = 50
    max_samples = 1.0
    max_features = 1.0
[[BaggingRegressor]]
    n_estimators = 50
    max_samples = 1.0
    max_features = 1.0
[[ExtraTreesClassifier]]
    n_estimators = 10
    criterion = gini
    min_samples_split = 2
    min_samples_leaf = 1
[[ExtraTreesRegressor]]
    n_estimators = 10
    criterion = mse
    min_samples_split = 2
    min_samples_leaf = 1
[[GradientBoostingClassifier]]
    loss = deviance
    learning_rate = 1.0
    n_estimators = 100
    subsample = 1.0
    criterion = friedman_mse
    min_samples_split = 2
    min_samples_leaf = 1
[[GradientBoostingRegressor]]
    loss = ls
    learning_rate = 0.1
    n_estimators = 100
    subsample = 1.0
    criterion = friedman_mse
    min_samples_split = 2
    min_samples_leaf = 1
[[RandomForestClassifier]]
    n_estimators = 10
    criterion = gini
    min_samples_leaf = 1
    min_samples_split = 2
[[RandomForestRegressor]]
    n_estimators = 10
    criterion = mse
    min_samples_leaf = 1
    min_samples_split = 2
# Here, an example of another instance of RandomForestRegressor, this one being
used based by the [[EnsembleFeatureSelector]]
# method from the [FeatureSelection] section.
[[RandomForestRegressor_selectEnsemble]]
    n_estimators = 100
    criterion = mse
[[XGBoostClassifier]]
```

(continues on next page)

(continued from previous page)

```
[[XGBoostRegressor]]
n_estimators = 100
objective = reg:squarederror

# Kernel ridge and linear methods

[[KernelRidge]]
alpha = 1
kernel = linear
# Here, an example of another instance of KernelRidge, this one being used based_
→by the [[MASTMLFeatureSelector]]
# method from the [FeatureSelection] section.
[[KernelRidge_selectMASTML]]
alpha = 1
kernel = linear
# Here, an example of another instance of KernelRidge, this one being used based_
→in the [LearningCurve] section.
[[KernelRidge_learn]]
alpha = 1
kernel = linear

[[ARDRegression]]
n_iter = 300
[[BayesianRidge]]
n_iter = 300
[[ElasticNet]]
alpha = 1.0
[[HuberRegressor]]
epsilon = 1.35
max_iter = 100
[[Lars]]
[[Lasso]]
alpha = 1.0
[[LassoLars]]
alpha = 1.0
max_iter = 500
[[LassoLarsIC]]
criterion = aic
max_iter = 500
[[LinearRegression]]
[[LogisticRegression]]
penalty = 12
C = 1.0
[[Perceptron]]
alpha = 0.0001
[[Ridge]]
alpha = 1.0
[[RidgeClassifier]]
alpha = 1.0
[[SGDClassifier]]
loss = hinge
penalty = 12
alpha = 0.0001
[[SGDRegressor]]
loss = squared_loss
penalty = 12
alpha = 0.0001
```

(continues on next page)

(continued from previous page)

```
# Neural networks

[[MLPClassifier]]
    hidden_layer_sizes = 100,
    activation = relu
    solver = adam
    alpha = 0.0001
    batch_size = auto
    learning_rate = constant
[[MLPRegressor]]
    hidden_layer_sizes = 100,
    activation = relu
    solver = adam
    alpha = 0.0001
    batch_size = auto
    learning_rate = constant
[[KerasRegressor]]
    [[Layer1]]
        layer_type = Dense
        neuron_num= 100
        input_dim= 287 #typically equal to n_features
        kernel_initializer= random_normal
        activation=relu
    [[Layer2]]
        layer_type = Dense
        neuron_num= 50
        kernel_initializer= random_normal
        activation=relu
    [[Layer3]]
        layer_type = Dense
        neuron_num= 25
        kernel_initializer= random_normal
        activation=relu
    [[Layer4]]
        layer_type = Dense
        neuron_num= 1
        kernel_initializer= random_normal
        activation=linear
    [[FitParams]]
        epochs=20
        batch_size=25
        loss = mean_squared_error
        optimizer = adam
        metrics = mse
        verbose=1
        shuffle = True
        #validation_split = 0.2

# Support vector machine methods

[[LinearSVC]]
    penalty = l2
    loss = squared_hinge
    tol = 0.0001
    C = 1.0
[[LinearSVR]]
```

(continues on next page)

(continued from previous page)

```

epsilon = 0.1
loss = epsilon_insensitive
tol = 0.0001
C = 1.0
[[NuSVC]]
    nu = 0.5
    kernel = rbf
    degree = 3
[[NuSVR]]
    nu = 0.5
    C = 1.0
    kernel = rbf
    degree = 3
[[SVC]]
    C = 1.0
    kernel = rbf
    degree = 3
[[SVR]]
    C = 1.0
    kernel = rbf
    degree = 3

# Decision tree methods

[[DecisionTreeClassifier]]
    criterion = gini
    splitter = best
    min_samples_split = 2
    min_samples_leaf = 1
[[DecisionTreeRegressor]]
    criterion = mse
    splitter = best
    min_samples_split = 2
    min_samples_leaf = 1
[[ExtraTreeClassifier]]
    criterion = gini
    splitter = random
    min_samples_split = 2
    min_samples_leaf = 1
[[ExtraTreeRegressor]]
    criterion = mse
    splitter = random
    min_samples_split = 2
    min_samples_leaf = 1

```

3.1.10 Misc Settings

This section controls which types of plots MAST-ML will write to the results directory and other miscellaneous settings.

Example:

```

[MiscSettings]
    plot_target_histogram = True
    plot_train_test_plots = True

```

(continues on next page)

(continued from previous page)

```
plot_predicted_vs_true = True
plot_predicted_vs_true_average = True
plot_best_worst_per_point = True
plot_each_feature_vs_target = False
plot_error_plots = True
rf_error_method = stdev
rf_error_percentile = 95
normalize_target_feature = False
```

- **plot_target_histogram** Whether or not to output target data histograms
- **plot_train_test_plots** Whether or not to output parity plots within each CV split
- **plot_predicted_vs_true** Whether or not to output summarized parity plots
- **plot_predicted_vs_true_average** Whether or not to output averaged parity plots
- **plot_best_worst_per_point** Whether or not to output parity plot showing best and worst split per point
- **plot_each_feature_vs_target** Whether or not to show plots of target feature as a function of each individual input feature
- **plot_error_method** Whether or not to show the individual and average plots of the normalized errors
- **rf_error_method** If using random forest, whether to calculate error bars with stdev or confidence intervals (confint)
- **rf_error_percentile** If using confint above, the confidence interval to use to calculate the error bars
- **normalize_target_feature** Whether or not to normalize the target feature values

CHAPTER 4

MAST-ML overview slides

The information for this MAST-ML overview shown on this page is available for download here:

[MASTMLoverview](#)

Let's begin with an overview of what MAST-ML is and what it can do:

What is MAST-ML?

MAST-ML is an open-source Python package designed to broaden and accelerate the use of machine learning in materials science research

MAST-ML:

- Leverages canonical machine learning packages (e.g. scikit-learn) to enable the easy construction and execution of general machine learning analysis pipelines
 - Codifies best practices of in-depth statistical analysis on user-defined model assessment tests (e.g. leave out group CV)
 - Enables data-driven materials research on a faster scale by automating execution and assessment of analysis pipelines, particularly for non-experts
-

Here is currently what MAST-ML can do as well as how to acquire it:

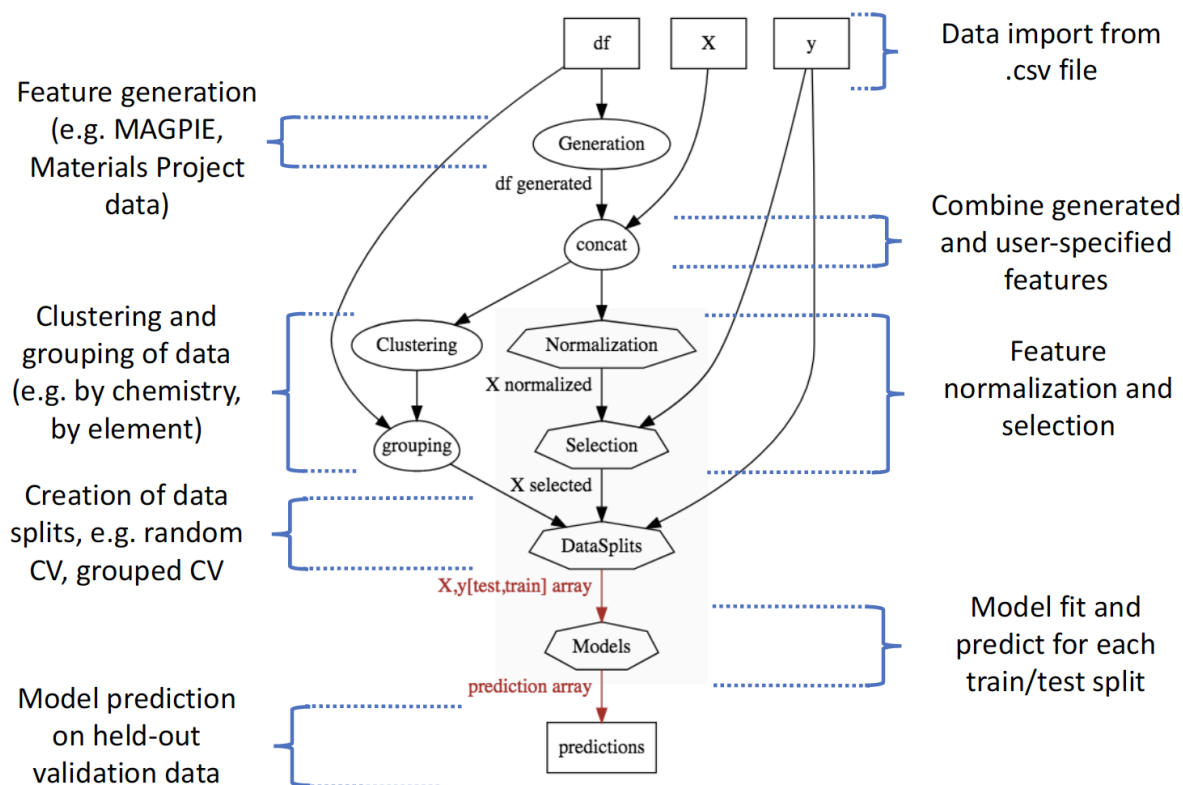
MAST-ML scope and capabilities

- The focus of MAST-ML is currently on supervised learning problems, with emphasis on its application to materials research problems
- MAST-ML supports the full library of scikit-learn modules, and is currently being extended to support tensorflow with Keras
- MAST-ML allows for the simultaneous execution of an arbitrary combination of data preprocessing, feature generation/selection, model types and model evaluation metrics
- MAST-ML is publicly available on GitHub (<https://github.com/uw-cmg/MAST-ML>) (pull/download master branch)



An overview of the general machine learning workflow that MAST-ML executes. Continuing development will focus on making the workflows more flexible and general

MAST-ML workflow



MAST-ML uses a text-based input file (.conf extension) which consists of different sections (corresponding to each part of the workflow) and specific subsections (e.g. different machine learning models to test, different feature selection algorithms, etc.). The input file is discussed in much greater detail [here](#):

[MAST-ML Input File](#)

and an input file with the full range of capabilities can be downloaded [here](#):

MASTMLinputfile

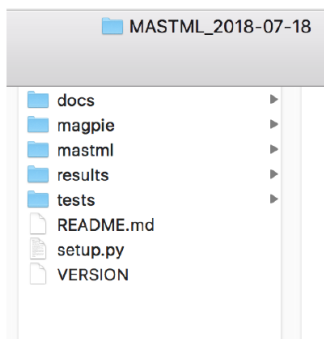
MAST-ML sample input

<pre>[GeneralSetup] input_features = Auto target_feature = Reduced barrier (eV) randomizer = False metrics = Auto not_input_features = Host element, Solute element, predict_Pt validation_column = predict_Pt</pre>	<p>General setup: names of input and target features, which feature to predict on, etc.</p>
<pre>[FeatureNormalization] [[StandardScaler]]</pre>	<p>Method to normalize features</p>
<pre>[DataSplits] [[NoSplit]] [[RepeatedKFold]] n_splits = 5 n_repeats = 5 [[LeaveOneGroupOut_host]] grouping_column = Host element</pre>	<p>How to split up data for testing, e.g. full fit (“NoSplit”), random CV, leave out group</p>
<pre>[Models] [[LinearRegression]] [[KernelRidge_5fold]] alpha = 0.009 gamma = 0.027 kernel = rbf [[RandomForestRegressor]] criterion = mse max_depth = 10 max_leaf_nodes = 200 min_samples_leaf = 1 min_samples_split = 2 n_estimators = 10 [[MLPRegressor]] #hidden_layer_sizes = 50, 4 hidden_layer_sizes = 296, 25 activation = relu solver = adam alpha = 0.001 batch_size = 20 learning_rate = constant</pre>	<p>Which models to test on and their associated parameters. Note that all model and parameter names are the same as in scikit-learn!</p>
<pre>[PlotSettings] feature_learning_curve = False data_learning_curve = False target_histogram = True train_test_plots = True predicted_vs_true = True predicted_vs_trueBars = True best_worst_per_point = True feature_vs_target = True</pre>	<p>Plotting controls: decide what is output</p>

Running MAST-ML is easily done with a single-line command in a Terminal/command line, your favorite IDE, or within a Jupyter notebook

Running MAST-ML

(1) Navigate to your main MAST-ML directory:



(2) In your terminal or IDE, run the command (one line):

```
python3 -m mastml.mastml_driver tests/conf/example_input.conf tests/csv/example_data.csv -o results/example_results
```

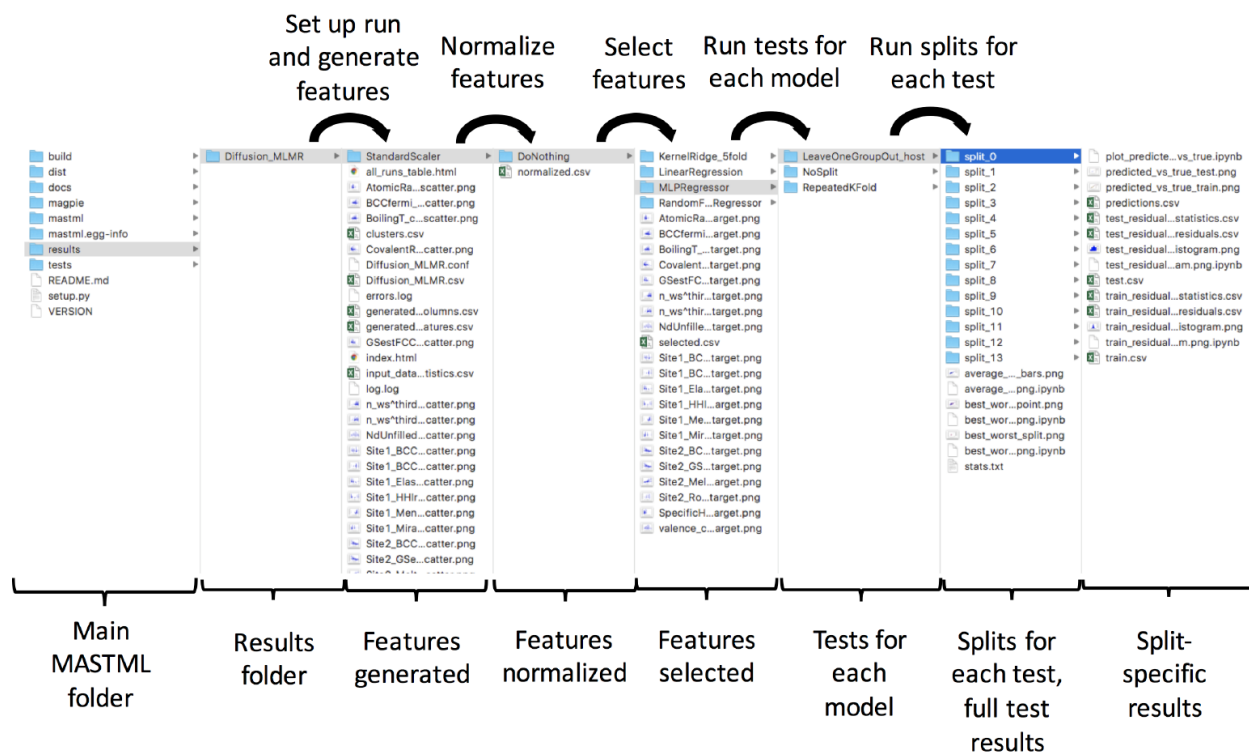
← Call module
← Path to input
← Path to data
← Path to results

(3) If it's working, you'll start seeing output on your screen:

```
[INFO] 2018-07-26 11:07:55,438 :  
  
MAST-ML run on 2018-07-26 16:07:55 using  
conf file: Diffusion_MLMR.conf  
csv file: Diffusion_MLMR.csv  
saving to: Diffusion_MLMR_07_26_11_07_55  
  
[INFO] 2018-07-26 11:07:55,438 : Copying input files to output directory...  
[INFO] 2018-07-26 11:07:55,461 : blacklisted features, either from "not_input_features" or a "grouping_column": ['Host element',  
'Solute element', 'predict_Pt']  
[DEBUG] 2018-07-26 11:07:56,434 : splitter_to_group_names:  
{'LeaveOneGroupOut_host': 'Host element'}
```

MAST-ML output takes the form of a full directory tree of results, with each level of the tree corresponding to a different portion of the machine learning workflow

MAST-ML high-level output



The last three figures demonstrate some example output of a few machine learning analysis features MAST-ML offers. Here, the ability to generate and select features is shown.

MAST-ML feature generation and selection

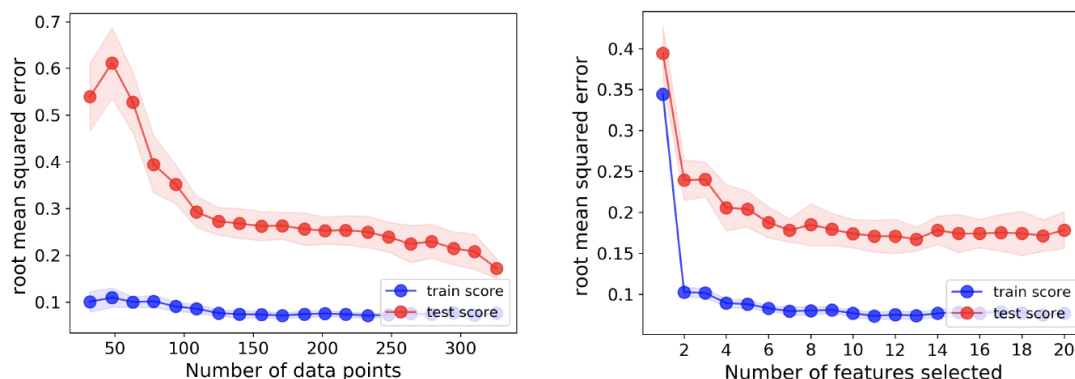
Generation (MAGPIE, Materials Project, Citrination)

100s or 1000s of features...



Host element	Reduced barrier (eV)	SecondionizationEnergy	ShearModulus	SpaceGroupNumber	SpecificHeatCapacity	ThermalConductivity	ThermalExpansionCoefficient	ThirdIonizationEnergy_min_value
I Ag	0	21.49	30	225	0.235	429	18.9	34.83
I Ag	-0.090141676	21.49	30	225	0.235	429	18.9	34.83
I Ag	0.259138544	21.49	30	225	0.235	429	18.9	34.83
I Ag	-0.022200405	21.49	30	225	0.235	429	18.9	34.83
I Ag	0.317672341	21.49	30	225	0.235	429	18.9	34.83
I Ag	0.202185741	21.49	30	225	0.235	429	18.9	34.83
I Ag	0.250571478	21.49	30	225	0.235	429	18.9	34.83
I Ag	-0.001431337	21.49	30	225	0.235	429	18.9	34.83
I Ag	0.164968058	21.49	30	225	0.235	429	18.9	34.83
I Ag	0.248163228	21.49	30	225	0.235	429	18.9	34.83
I Ag	-0.146976238	21.49	30	225	0.235	429	18.9	34.83
I Al	0	18.828	26	225	0.9	237	23.1	28.447
I Al	-0.12503	18.828	26	225	0.9	237	23.1	28.447
I Al	-0.14243	18.828	26	225	0.9	237	23.1	28.447

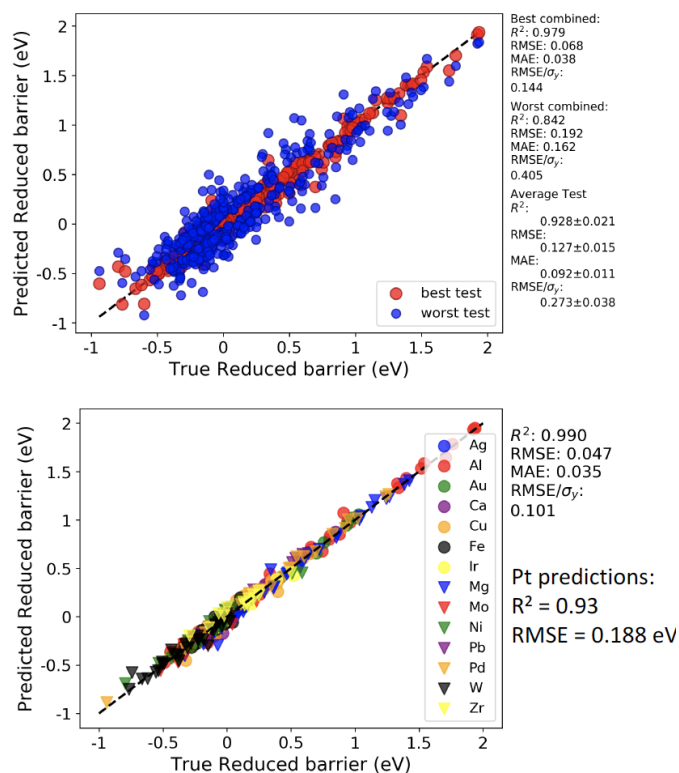
Selection and learning curves (Random Forest on Diffusion data)



A core feature of MAST-ML is the many pieces of statistical analysis regarding model assessment, which forms the basis of interpreting the quality and extensibility of a machine learning model.

MAST-ML model assessment

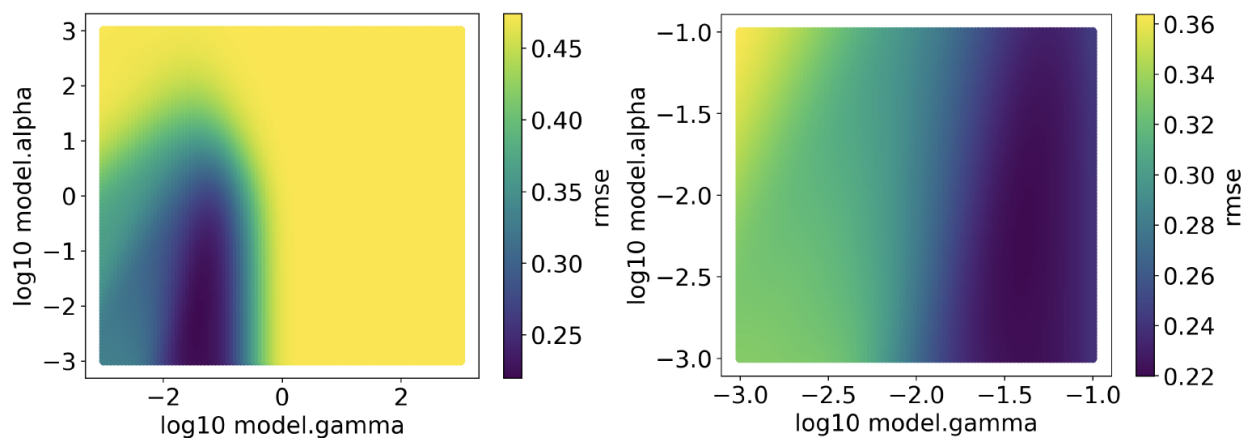
- A blizzard of statistics:
 - Output of every train/test split and prediction
 - Averages over every split and error bars for each point
 - Best/worst on per-split and per-point basis
 - Per-group and per-cluster train/test visualization
 - Output as:
 - Spreadsheets
 - Histograms
 - Parity/scatter plots
 - HTML summary file



Finally, MAST-ML offers the ability to easily optimize the model hyperparameters used in your analysis

MAST-ML hyperparameter optimization

- MAST-ML currently supports hyperparameter optimization using grid search and a genetic algorithm (GA).
- Example heat maps of running grid search to optimize the α and γ parameters in a KernelRidge model on the diffusion data set from the work of Wu, *et al.* Comp. Mat. Sci. (2017)



Running MAST-ML on Google Colab

In addition to running MAST-ML on your own machine or computing cluster, MAST-ML can be run using cloud resources on Google Colab. This can be advantageous as you don't have to worry about installing MAST-ML yourself, and all output files can be saved directly to your Google Drive.

MAST-ML comes with a notebook called MASTML_Colab.ipynb that you can open in Google Colab

MASTML_Colab.ipynb

Once you open the notebook in Google Colab, it will look something like this:

PRO

File Edit View Insert Runtime Tools Help

MASTML.pyipnb

☆

File Edit View Insert Runtime Tools Help All changes saved

Files

sample_data

pip install mastml and pinned versions of few dependencies. It will take a couple minutes

Don't worry about the package incompatibility warnings

!pip install mastml

!pip install mdf_toolbox --upgrade

!pip install scikit-learn==0.23.2

Requirement already satisfied: imageio=2.3.0 in /usr/local/lib/python3.6/dist-packages (from mastml) (2.4.1)

Requirement already satisfied: certifi=2020.12.05 in /usr/local/lib/python3.6/dist-packages (from mastml) (2020.12.5)

Requirement already satisfied: Shapely in /usr/local/lib/python3.6/dist-packages (from imageio=0.2.5->mastml) (1.7.1)

Requirement already satisfied: opencv-python in /usr/local/lib/python3.6/dist-packages (from imageio=0.2.5->mastml) (4.5.1.2)

Requirement already satisfied: scikit-image=0.11.0 in /usr/local/lib/python3.6/dist-packages (from imageio=0.2.5->mastml) (0.11.0)

Requirement already satisfied: Pillow in /usr/local/lib/python3.6/dist-packages (from imageio=0.2.5->mastml) (7.0.0)

Requirement already satisfied: sphinx=1.7 in /usr/local/lib/python3.6/dist-packages (from pygments>=2.0.0->mastml) (1.7.0)

Requirement already satisfied: iniconfig in /usr/local/lib/python3.6/dist-packages (from pytest>=5.0.1->mastml) (1.1.1)

Requirement already satisfied: tomli in /usr/local/lib/python3.6/dist-packages (from pytest>=5.0.1->mastml) (0.10.2)

Requirement already satisfied: keras-preprocessing=1.1.2 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (1.1.2)

Requirement already satisfied: h5py=2.10.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (2.10.0)

Requirement already satisfied: typing-extensions=3.7.4 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (3.7.4)

Requirement already satisfied: gast=0.3.3 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (0.3.3)

Requirement already satisfied: tensorflow-estimator<2.5.0,>=2.4.0rc0 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (2.4.0rc0)

Requirement already satisfied: opt-einsum=3.3.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (3.3.0)

Requirement already satisfied: wrapt=1.12.1 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (1.12.1)

Requirement already satisfied: tensorboard=2.4 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (2.4.0)

Requirement already satisfied: astunparse=1.6.3 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (1.6.3)

Requirement already satisfied: grpcio=1.32.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (1.32.0)

Requirement already satisfied: wheel=0.35 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (0.35)

Requirement already satisfied: termcolor=1.1.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (1.1.0)

Requirement already satisfied: protobuf=3.9.2 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (3.9.2)

Requirement already satisfied: absl-py=0.10 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (0.10)

Requirement already satisfied: google-pasta=0.2 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (0.2)

Requirement already satisfied: flatbuffers=1.12.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow>=1.15.0->mastml) (1.12.0)

Collecting pyjwt[crypto]<2.0.0,>=1.5.3

Downloading https://files.pythonhosted.org/packages/87/8b/6a9f14b5f781697e51259d81657e6048fd3a113229cf346880b8754556

Collecting ruamel.yaml.clib==0.1.2; platform_python_implementation == "CPython" and python_version < "3.9"

Downloading https://files.pythonhosted.org/packages/88/ff/ec25dc01ef04232a9e68ff18492e37dfa01f1f58172e702ad4f38536d41

552kB 37.9MB/s

There are a few blocks of code in this notebook. The first block performs a pip install of MAST-ML for this Colab

session. The second block links your Google Drive to the Colab instance so MAST-ML can save your run output directly to your Google Drive.

The one thing you'll need to do from here is to upload a data file (.csv or .xlsx format) and MAST-ML input file (.conf format) to this Colab session. Files can be uploaded by pressing the vertical arrow on the left side of the screen, by the file directory tree.

```
example_input.conf
```

```
example_data.xlsx
```

Note that when a Colab session ends, the files you upload will be deleted. Since your output will be saved to your Google Drive, the data and input files will be deleted. Note that MAST-ML automatically saves a copy of both of these files to your output directory for each run you do.

6.1 Introduction

This document provides step-by-step tutorials of conducting and analyzing different MAST-ML runs. For this tutorial, we will be using the dataset `example_data.xlsx` in the `tests/csv/` folder and input file `example_input.conf` in `tests/conf/`.

MAST-ML requires two files to run: The first is the text-based input file (`.conf` extension). This file contains all of the key settings for MAST-ML, for example, which models to fit and how to normalize your input feature matrix. The second file is the data file (`.csv` or `.xlsx` extension). This is the data file containing the input feature columns and values (X values) and the corresponding y data to fit models to. The data file may contain other columns that are dedicated to constructing groups of data for specific tests, or miscellaneous notes, which columns can be selectively left out so they are not used in the fitting. This will be discussed in more detail below.

Throughout this tutorial, we will be modifying the input file to add and remove different sections and values. For a complete and more in-depth discussion of the input file and its myriad settings, the reader is directed to the dedicated input file section:

MAST-ML Input File

The data contained in the `example_data.csv` file consist of a previously selected matrix of X features created from combinations of elemental properties, for example the average atomic radius of the elements in the material. The y data values used for fitting are listed in the “Scaled activation energy (eV)” column, and are DFT-calculated migration barriers of dilute solute diffusion, referenced to the host system. For example, the value of Ag solute diffusing through a Ag host is set to zero. The “Host element” and “Solute element” columns denote which species comprise the corresponding reduced migration barrier.

6.2 Your first MAST-ML run

It’s time to conduct your very first MAST-ML run! First, we will set up the most basic input file, which will only import your data and input file, and do nothing else except copy the input files to the results directory and output a basic histogram of the target data. Open the `example_input.conf` file (or create your own new file), and write the following in your input file:

Example:

```
[GeneralSetup]
input_features = Auto
input_target = Scaled activation energy (eV)
randomizer = False
metrics = Auto
input_other = Material composition, Host element, Solute element, predict_Pt
```

The General Setup section contains high-level control about how your input data file is parsed. Additional details of each parameter can be found in the MAST-ML Input File section in this documentation. Briefly, setting “input_features” to “Auto” will automatically assign all columns to be part of the X feature matrix, except those that are associated with target_feature or not_input_features. The option “randomizer” will shuffle all of your y-data, which can be useful for running a “null” test. The “metrics” option is used to denote which metrics to eventually evaluate your models on, such as mean_absolute_error. Using “Auto” provides a catalogue of standard metrics which is generally sufficient for many problems. Finally, the “not_input_features” field is used to denote any feature columns you don’t want to use in fitting. If some columns contain text notes, these will need to be added here too.

There are two ways to execute a MAST-ML run. The first is to run it via a Terminal or IDE command line by directly calling the main MAST-ML driver module. Here, the python -m (for module) command is invoked on the mastml.masml_driver module, and the paths containing the input file and data file are passed in. Lastly, the argument -o (for output) is used together with the path to put all results files and folders.

Example:

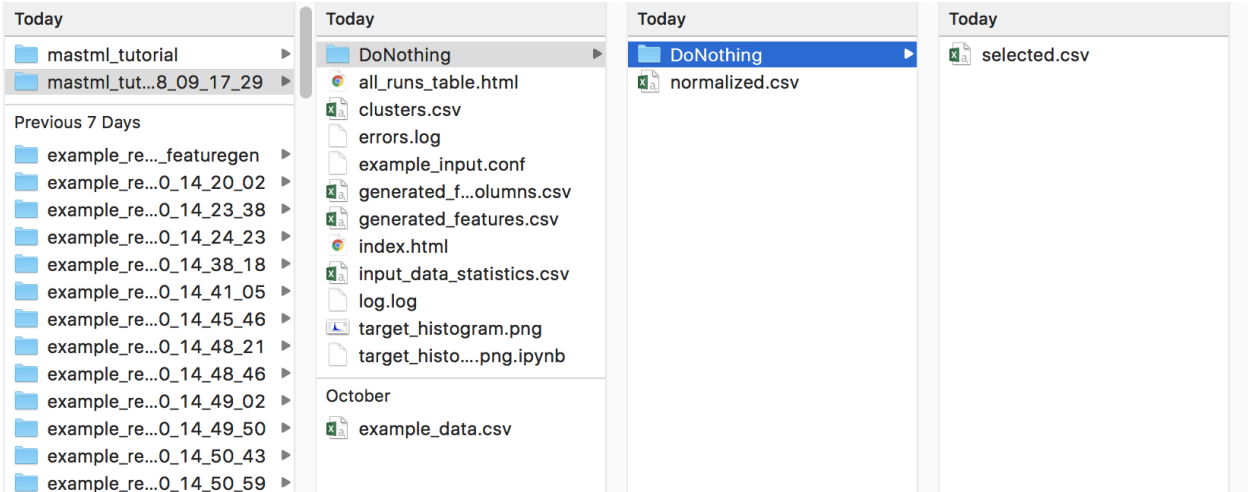
```
python3 -m mastml.masml_driver tests/conf/example_input.conf tests/csv/example_data.
↪xlsx -o results/mastml_tutorial
```

The second way is to run MAST-ML through a Jupyter notebook by importing mastml and running the mastml_driver main() method and supply the paths to the input file, data file

Example:

```
import mastml_driver
conf_path = 'tests/conf/example_input.conf'
data_path = 'tests/conf/example_data.csv'
results_path = 'results/mastml_tutorial'
mastml_driver.main(conf_path, data_path, results_path)
```

Let’s examine the output from this first run. Below is a screenshot of a Mac directory output tree in the results/mastml_tutorial folder. Note that you can re-use the same output folder name, and the date and time of the run will be appended so no work will be lost. Each level of the directory tree corresponds to a step in the general supervised learning workflow that MAST-ML uses. The first level is general data input and feature generation, the second level is numerical manipulation of features, and the third level is selection of features. Since we did not do any feature manipulation in this run, the output selected.csv, normalized.csv and generated_features.csv are all the same, and are the same file as the copied input data file, example_data.csv. In the main directory tree, there is also a log.log and errors.log file, which summarize the inner details of the MAST-ML run and flag any errors that may have occurred. There are two .html files which provide very high-level summaries of data plots and file links that may be of interest, to make searching for these files easier. Finally, there is some generated data about the statistics of your input target data. A histogram named target_histogram.png is created, and basic statistical summary of your data is saved in the input_data_statistics.csv file.



6.3 Cleaning input data

Now, let's imagine a slightly more complicated (but realistic) scenario where some of the value of your X feature matrix are not known. Open your `example_data.csv` file, and randomly remove some values of the X feature columns in your dataset. Don't remove any y data values in the "Reduced Barrier (eV)" column. You'll need to add the following section to your input file to handle cleaning of the input data:

Example:

```
[DataCleaning]
cleaning_method = imputation
imputation_strategy = mean
```

What this does is perform data imputation, where each missing value will be replaced with the mean value for that particular feature column. Other data cleaning options include imputation with median values, simply removing rows of data with missing values, or performing a probabilistic principal component analysis to fill in missing values.

From inspecting the data file in the parent directory to that in the subsequent directories, you can see that the missing values (here, the first 10 rows of the first several features were removed) have been replaced with the mean values for each respective feature column:

After data cleaning with imputation:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1																					
2	-0.0012068	-0.0101132	-0.0066092	0.01328052	0.00505528	-1.217816	-1.1886842	-0.7687543	0.21836133	-0.5898459	-0.7553359	-0.9611396	-1.3889356	1.04388056	-1.0578981	-0.7380744	0.92849204	-1.6769659	-1.4764603	-0.1939197	(
3	-0.0012068	-0.0101132	-0.0066092	0.01328052	0.00505528	-1.217816	-0.0395783	0.15850604	-0.8550293	-0.5898459	-0.7553359	-0.5430047	-0.3697167	-0.2300168	-0.3886101	-0.7380744	-0.866396	-1.24773	-0.7416888	-0.1939197	(
4	-0.0012068	-0.0101132	-0.0066092	0.01328052	0.00505528	-1.217816	-0.8258087	0.11095423	-0.8550293	-0.5898459	-0.7553359	-0.6829295	0.30976261	-0.9460356	-0.287857	-0.7380744	-0.9039853	0.03997786	-0.0952687	-0.1939197	(
5	-0.0012068	-0.0101132	-0.0066092	0.01328052	0.00505528	-1.217816	-0.4024539	-0.5072193	-0.8550293	-0.5898459	-0.7553359	-0.5430047	-1.3889356	0.77367197	-0.5181497	-0.7380744	-0.631463	-1.6769659	-1.3617284	-0.1939197	(
6	-0.0012068	-0.0101132	-0.0066092	0.01328052	0.00505528	-1.217816	-0.4024539	0.20605786	-0.8550293	-0.5898459	-0.7553359	-0.5790791	-0.029977	-0.5857918	-0.287857	-0.7380744	-0.9791639	0.03997786	-0.6322014	-0.1939197	(
7	-0.0012068	-0.0101132	-0.0066092	0.01328052	0.00505528	-1.217816	-0.8258087	-0.1743566	-0.8550293	-0.5898459	-0.7553359	-1.0163443	0.30976261	-0.8028065	-0.176309	-0.7380744	-0.9885612	0.4692138	-0.236792	-0.1939197	(
8	-0.0012068	-0.0101132	-0.0066092	0.01328052	0.00505528	-1.217816	0.08138018	0.15850604	-0.8550293	-0.5898459	-0.7553359	-0.5506568	-0.7094563	0.22287843	-0.3058486	-0.7380744	-0.9321773	-1.24773	-1.1093543	-0.1939197	(
9	-0.0012068	-0.0101132	-0.0066092	0.01328052	0.00505528	-1.217816	0.3232972	-0.9827375	-0.8550293	-0.5898459	-0.7553359	-0.9960231	1.66872114	0.00465102	-0.14034342	-0.7380744	0.92849204	-1.24773	-1.3360137	-0.5807058	(
10	-0.0012068	-0.0101132	-0.0066092	0.01328052	0.00505528	-1.217816	-0.2814954	-0.3883398	-0.8550293	-0.5898459	-0.7553359	-0.3462353	1.3289815	-0.4351508	-0.0215811	-0.7380744	0.92849204	-0.818494	-1.8553839	-0.1939197	(
11	-0.0012068	-0.0101132	-0.0066092	0.01328052	0.00505528	-1.217816	-0.7048502	-0.1030289	-0.8550293	-0.5898459	-0.7553359	-0.2806455	0.98924187	-0.7788033	-0.1439241	-0.7380744	-0.2273783	-0.3892581	-0.6309595	-0.1939197	(
12	-1.1379409	0.40250421	0.26304467	-0.6269246	1.48718093	-1.217816	0.20233869	-0.8638579	-0.8550293	-0.5898459	-0.7553359	-1.6470994	-1.3889356	1.49421587	-0.5073547	-0.7380744	-0.1240078	-1.6769659	-1.5249205	-0.3658247	(
13	-0.8687861	1.16944744	0.0521955	-0.6009658	0.74407334	0.93940978	-1.1886842	-0.6260989	-1.9284199	-0.7285132	1.80140537	-0.5708803	-1.3889356	0.75455071	-1.0578981	-0.8158282	0.79692957	-0.818494	1.13600944	-0.064991	(
14	-0.5318144	1.16944744	0.0521955	-0.6009658	1.02133691	0.93940978	0.26281795	-0.6974266	0.21836133	-0.7285132	1.80140537	-0.7660099	-1.3889356	1.04388056	1.33498664	-0.8158282	0.79692957	-1.24773	-0.1702254	-0.1939197	(
15	-0.693817	1.16944744	0.0521955	0.28615074	0.57573244	0.93940978	-1.0677257	-0.5072193	-0.8550293	-0.7285132	1.80140537	-1.6121182	-1.3889356	0.48723393	0.99314596	-0.8158282	0.42103678	0.03997786	0.68218206	-0.064991	(
16	-0.4173518	1.16944744	0.0521955	-0.5631762	0.88289301	0.93940978	-0.2814954	-0.1981326	1.29175196	-0.7285132	1.80140537	-0.3866823	-1.3889356	0.91271209	1.72000719	-0.8158282	0.79692957	-0.818494	0.51679968	-0.064991	(
17	-1.3036881	1.16944744	0.0521955	1.61507952	0.67098851	0.93940978	0.44425572	-1.1729447	1.29175196	-0.7285132	1.80140537	-1.0928657	-1.3889356	0.68715917	1.74159713	-0.8158282	0.79692957	0.03997786	1.38205918	-1.0534444	(

6.4 Feature generation and normalization

For this run, we are going to first generate a large X feature matrix based on a suite of elemental properties. Then, we are going to normalize the feature matrix so that all values in a given feature column have a mean of zero and a standard deviation equal to one.

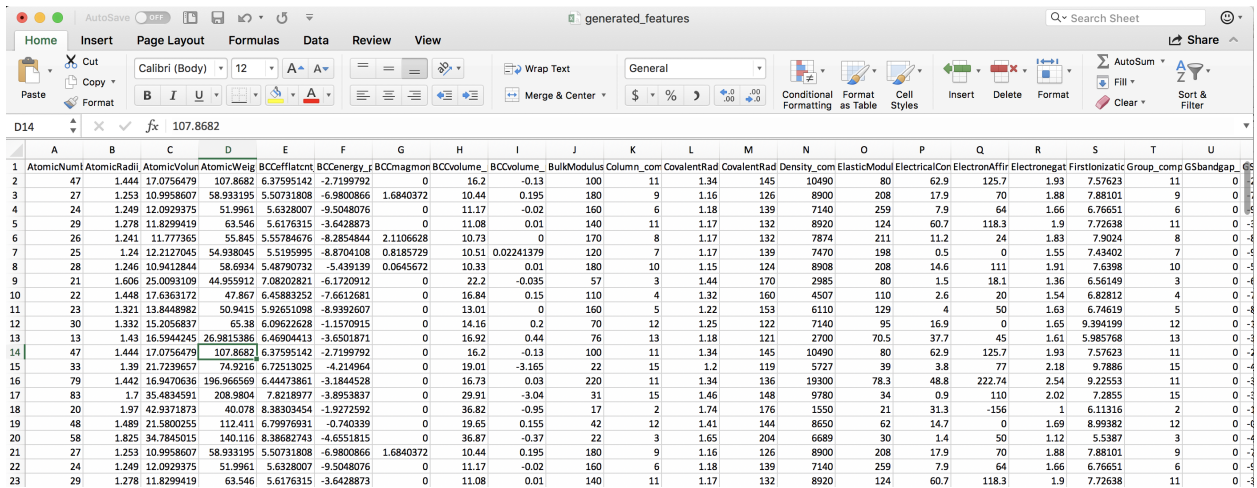
To perform the feature generation and normalization steps, add these sections to your input file. Use the same file from the previous run, which contains the GeneralSetup and DataCleaning sections, and use your data file with the values you previously removed. (Note that you can use the pristine original data file too, and the data cleaning step will simply do nothing). For the purpose of this example, we are going to generate elemental features using the MAGPIE approach, using compositions as specified in the “Solute element” column of the data file. Note that if multiple elements are present, features containing the average (both mean and composition-weighted averages) of the elements present will be calculated. The value specified in the composition_feature parameter must be a column name in your data file which contains the material compositions.

Example:

```
[FeatureGeneration]
  [[Magpie]]
    composition_feature = Solute element
    feature_types = composition_avg, arithmetic_avg, max, min, difference,
elements

[FeatureNormalization]
  [[StandardScaler]]
```

After performing this run, we can see that the .csv files in the feature generation and normalization folders of the results directory tree are now updated to reflect the generated and normalized X feature matrices. There are now many more features in the generated_features.csv file:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	AtomicNum	AtomicRadii	AtomicVolun	AtomicWeig	BCoefflatnc	BCEnergy_f	BCMnagmon	BCVolume_	BCVolume_	BulkModulus	Column_com	CovalentRad	CovalentRad	Density_com	ElasticModul	ElectricalCon	ElectronAffir	Electronegat	Firstionizati	Group_comp	GSbandgap
2	47	1.444	17.0756479	107.8682	6.37595142	-2.7199792	0	16.2	-0.13	100	11	1.34	145	10490	80	62.9	125.7	1.93	7.57623	11	0
3	27	1.253	10.9585607	58.93135	5.50731808	-6.9800866	1.6840372	10.44	0.195	180	9	1.16	126	8900	208	17.9	70	1.88	7.88101	9	0
4	24	1.249	12.0292937	51.9961	5.6328007	-9.5048076	0	11.17	-0.02	160	6	1.18	139	7140	259	7.9	64	1.66	6.76651	6	0
5	29	1.278	11.8299419	63.546	5.6176315	-3.6428873	0	11.08	0.01	140	11	1.17	132	8920	124	60.7	118.3	1.9	7.72638	11	0
6	26	1.241	11.777365	55.845	5.55784676	-8.2854844	2.1106628	10.73	0	170	8	1.17	132	7874	211	11.2	24	1.83	7.9024	8	0
7	25	1.24	12.2127045	54.938045	5.5195995	-8.8704108	0.8185729	10.51	0.02241379	120	7	1.17	139	7470	198	0.5	0	1.55	7.43402	7	0
8	28	1.246	10.9412844	58.6934	5.48790732	-5.439139	0.0645672	10.33	0.01	180	10	1.15	124	8908	208	14.6	111	1.91	7.6398	10	0
9	21	1.606	25.0093109	44.95912	7.08202821	-6.1720912	0	22.2	-0.035	57	3	1.44	170	2985	80	1.5	18.1	1.36	6.56149	3	0
10	22	1.448	17.6363172	47.867	6.45883252	-7.6612681	0	16.84	0.15	110	4	1.32	160	4507	110	2.6	20	1.54	6.82812	4	0
11	23	1.321	13.8448982	50.9415	5.92651098	-8.9392607	0	13.01	0	160	5	1.22	153	6110	129	4	50	1.63	6.74619	5	0
12	30	1.332	15.2056837	65.38	6.09622628	-1.1570915	0	14.16	0.2	70	12	1.25	122	7140	95	16.9	0	1.65	9.394199	12	0
13	13	1.43	16.5944245	26.9815386	6.4904413	-3.6501871	0	16.92	0.44	76	13	1.18	121	2700	70.5	37.7	45	1.61	5.98578	13	0
14	47	1.444	17.0756479	107.8682	6.37595142	-2.7199792	0	16.2	-0.13	100	11	1.34	145	10490	80	62.9	125.7	1.93	7.57623	11	0
15	33	1.39	21.7239657	74.9216	6.72513025	-4.214964	0	19.01	-3.165	22	15	1.2	119	5727	39	3.8	77	2.18	9.7886	15	0
16	79	1.442	16.9470636	196.966569	6.44473861	-3.1844528	0	16.73	0.03	220	11	1.34	136	19300	78.3	48.8	222.74	2.54	9.22553	11	0
17	83	1.7	35.4834591	208.9804	7.8218977	-3.8953837	0	29.91	-3.04	31	15	1.46	148	9780	34	0.9	110	2.02	7.2855	15	0
18	20	1.97	42.9371873	40.078	8.38303454	-1.972592	0	36.82	-0.95	17	2	1.74	176	1550	21	31.3	-156	1	6.11316	2	0
19	48	1.489	21.9802055	112.411	6.79970931	-7.740339	0	19.65	0.155	42	12	1.41	144	8650	62	14.7	0	1.69	8.99382	12	0
20	58	1.825	34.7845015	140.116	8.36682743	-4.6551815	0	36.87	-0.37	22	3	1.65	204	6689	30	1.4	50	1.12	5.5387	3	0
21	27	1.253	10.9585607	58.93135	5.50731808	-6.9800866	1.6840372	10.44	0.195	180	9	1.16	126	8900	208	17.9	70	1.88	7.88101	9	0
22	24	1.249	12.0292937	51.9961	5.6328007	-9.5048076	0	11.17	-0.02	160	6	1.18	139	7140	259	7.9	64	1.66	6.76651	6	0
23	29	1.278	11.8299419	63.546	5.6176315	-3.6428873	0	11.08	0.01	140	11	1.17	132	8920	124	60.7	118.3	1.9	7.72638	11	0

Note that feature columns that are identical in all values are removed automatically. We can see that the normalized feature set consists of each column having mean zero and standard deviation of one:

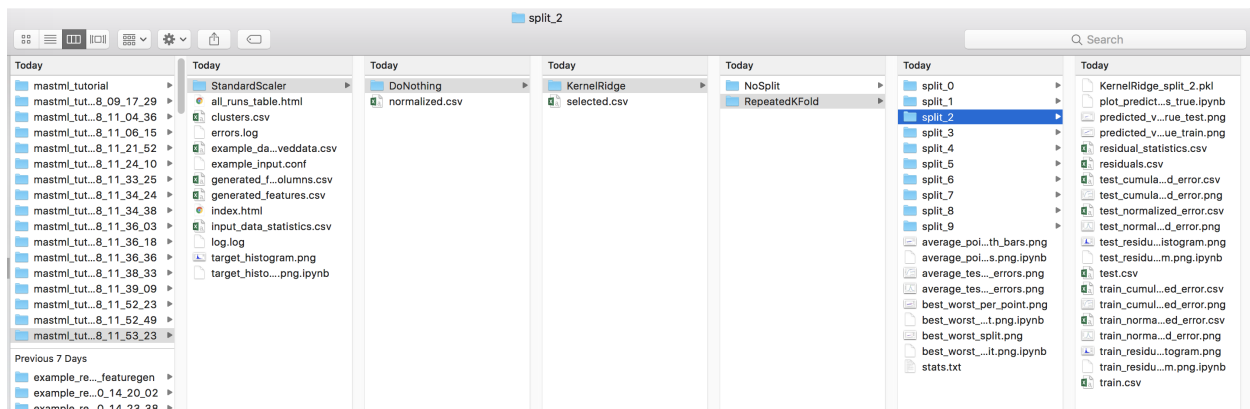
	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ
1	AtomicNum	AtomicRadii	AtomicVolun	AtomicWeig	BCCoefflatnc	BCEnergy_f	BCMnagmon	BCVolume_	BCVolume_	BulkModulus	Column_com	CovalentRad	CovalentRad	Density_com	ElasticModul	ElectricalCon	ElectronAffir	Electronegat	FirstIonizati	Group_comp	GSbandgap
2	0.11576176	-0.0488203	-0.3286616	0.04812277	-0.240759	1.02133691	-0.3093644	-0.3073781	0.27941581	-0.1926648	0.62661482	0.14875859	-0.0741463	0.15229148	-0.792842	3.06037953	0.83323764	0.29586562	0.00675915	0.62661482	-0.2054785
3	-0.8260289	-1.0997791	-1.0817547	-0.8257321	-1.3533457	-0.2484572	3.60155203	-1.0921886	0.44244326	0.5991084	0.1631772	-1.1069961	-0.9908638	-0.1425515	0.33176394	0.27415853	0.07312325	0.1623038	0.27145838	0.11631772	-0.2054785
4	-0.9672975	-1.1217888	-0.9458617	-0.949611	-1.1926293	-1.0009912	-0.3093644	-0.9927247	0.33459433	0.40116509	-0.6491279	-0.9674438	-0.363636	-0.4688427	0.69475969	-0.3450017	-0.0087562	0.2532682	-0.6964768	-0.6491279	-0.2054785
5	-0.7314898	-0.9622191	-0.9784385	-0.7433591	-1.2120589	0.74624918	-0.3093644	-1.0049874	0.34964302	0.20322179	0.62661482	-0.10372064	-0.701374	-0.1388072	-0.2661114	2.92416428	0.73225297	0.21572853	0.13716334	0.62661482	-0.2054785
6	-0.8731184	-1.165808	-0.9495191	-1.2886347	-0.3755522	4.59232412	-0.10526755	0.34462679	0.50013675	-0.1388038	-0.10372064	-0.701374	-0.3327494	0.35311663	-0.1406788	-0.5546193	0.02874198	0.29035544	0.29035544	-0.1388038	-0.2054785
7	-0.9202079	-1.1713104	-0.9310263	-0.8907053	-1.337624	-0.811899	1.59164482	-1.0826599	0.35587006	0.00527849	-0.3939794	-1.0372064	-0.363636	-0.4076563	0.20508831	-0.8031803	-0.8821372	-0.7192042	-0.1167492	-0.3939794	-0.2054785
8	-0.7789393	-1.138296	-1.088515	-0.8300142	-1.3782171	0.21084721	-0.159417	-1.1071763	0.34964302	0.5991084	0.1746627	-1.1767317	-1.0873004	-0.1410322	0.33176394	0.09883566	0.6263295	0.2424089	0.06196924	0.37146627	-0.2054785
9	-1.108566	0.84256882	0.6540801	-1.0753308	0.63624291	-0.0076211	-0.3093644	0.51013282	0.3270699	-0.6182429	-1.4145736	0.84638507	1.13206098	-1.2392345	-0.5792842	-0.7412642	-0.6351342	-1.2267391	-0.8745352	-1.4145736	-0.2054785
10	-1.0614765	-0.0268107	-0.2592125	-1.0233462	-1.346001	-0.4514944	-0.3093644	-0.2201769	0.41987023	-0.0936932	-1.159425	0.00923329	0.64957808	-0.9570356	-0.3657573	-0.6731566	-0.6092057	-0.7459166	-0.424969	-1.159425	-0.2054785
11	-1.014387	-0.7256158	-0.7288492	-0.9684434	-0.8164283	-0.8324208	-0.3093644	-0.7420214	0.34462679	0.40116509	-0.9042765	-0.6883932	0.31184005	-0.6598183	-0.2305236	-0.5864742	-0.1998083	-0.5055053	-0.7141246	-0.9042765	-0.2054785
12	-0.6847603	-0.6650894	-0.560291	-0.7106085	-0.5990471	1.48718093	-0.3093644	-0.5853318	0.44495138	-0.4895798	0.88176336	-0.4791052	-1.1838569	-0.4688427	-0.4725207	0.21224251	-0.8821372	-0.4520806	1.58565212	0.88176336	-0.2054785
13	-1.4852823	-0.125854	-0.38827	-1.3963075	-0.1215205	0.74407334	-0.3093644	-0.2092768	0.56534088	-0.4301968	1.13691191	-0.9674438	-1.2321052	-1.2920772	-0.646901	1.50009577	-0.2680412	-0.55893	-1.3745456	1.13691191	-0.2054785
14	0.11576176	-0.0488203	-0.3286616	0.04812277	-0.240759	1.02133691	-0.3093644	-0.3073781	0.27941581	-0.1926648	0.62661482	0.14875859	-0.0741463	0.15229148	-0.792842	3.06037953	0.83323764	0.29586562	0.00675915	0.62661482	-0.2054785
15	-0.5434917	-0.3459501	0.24711777	-0.5402198	0.2064894	0.57573244	-0.3093644	0.07548952	-1.2430098	-0.9646437	1.64720901	-0.8279185	-1.3286018	-0.7308315	-0.8711043	-0.5988574	0.16864929	0.96367471	1.92818655	1.64720901	-0.2054785
16	1.62262676	-0.0598251	-0.3445891	1.6391933	-0.1526524	0.882893	-0.3093644	-0.2351464	0.39697548	0.949995	0.62661482	0.14875859	-0.5083809	1.78578165	-0.5913841	2.18736362	2.157950157	1.9253198	1.43916431	0.62661482	-0.2054785
17	1.81098469	1.5979462	1.9514833	1.85372972	1.61129246	0.67098851	-0.3093644	1.56063433	-1.1803069	-0.8755692	1.64720901	0.98591037	0.0705986	0.02064812	-0.966821	-0.7784138	0.61888637	0.53627689	-0.2457378	1.64720901	-0.2054785
18	-1.1556556	2.8454321	2.87476421	-1.1624379	2.33002891	1.25761995	-0.3093644	2.50213439	-0.131915	-1.0141295	-1.6697221	2.39826453	1.42155073	-1.5053024	-0.9992204	1.10383323	-3.0110034	-2.188842	1.2639066	-1.6697221	-0.2054785
19	0.1628513	0.19877867	0.2292881	0.12924564	0.30209142	1.61140076	-0.3093644	0.16269068	0.42237835	-0.7667004	0.88176336	0.63709713	-0.1223946	-0.1888688	-0.7074003	0.07602726	-0.8821372	-0.3452311	1.23792585	0.88176336	-0.2054785
20	0.63374661	2.04759489	1.86490459	0.62398653	2.33488706	0.44451847	-0.3093644	2.50894698	1.15902631	-0.9646437	-1.4145736	2.31140068	2.77250285	-0.552464	-0.9351624	-0.1998083	-1.8678358	-1.762821	-1.4145736	-0.2054785	
21	-0.8260289	-1.0997791	-1.0817547	-0.8257321	-1.3533457	-0.2484572	3.60155203	-1.0921886	0.44244326	0.5991084	0.1631772	-1.1069961	-0.9908638	-0.1425515	0.33176394	0.27415853	0.07312325	0.1623038	0.27145838	0.11631772	-0.2054785
22	-0.9672975	-1.1217888	-0.9458617	-0.949611	-1.1926293	-1.0009912	-0.3093644	-0.9927247	0.33459433	0.40116509	-0.6491279	-0.9674438	-0.363636	-0.4688427	0.69475969	-0.3450017	-0.0087562	0.2532682	-0.6964768	-0.6491279	-0.2054785
23	-0.7314898	-0.9622191	-0.9784385	-0.7433591	-1.2120589	0.74624918	-0.3093644	-1.0049874	0.34964302	0.20322179	0.62661482	-0.10372064	-0.701374	-0.1388072	-0.2661114	2.92416428	0.73225297	0.21572853	0.13716334	0.62661482	-0.2054785
24	-0.8731184	-1.165808	-0.9495191	-1.2886347	-0.3755522	4.59232412	-0.10526755	0.34462679	0.50013675	-0.1388038	-0.10372064	-0.701374	-0.3327494	0.35311663	-0.1406788	-0.5546193	0.02874198	0.29035544	0.29035544	-0.1388038	-0.2054785
25	-0.9202079	-1.1713104	-0.9310263	-0.8907053	-1.337624	-0.811899	1.59164482	-1.0826599	0.35587006	0.00527849	-0.3939794	-1.0372064	-0.363636	-0.4076563	0.20508831	-0.8031803	-0.8821372	-0.7192042	-0.1167492	-0.3939794	-0.2054785
26	-0.7789393	-1.138296	-1.088515	-0.8300142	-1.3782171	0.21084721	-0.159417	-1.1071763	0.34964302	0.5991084	0.1746627	-1.1767317	-1.0873004	-0.1410322	0.33176394	0.09883566	0.6263295	0.2424089	0.06196924	0.37146627	-0.2054785
27	-1.108566	0.84256882	0.6540801	-1.0753308	0.63624291	-0.0076211	-0.3093644	0.51013282	0.3270699	-0.6182429	-1.4145736	0.84638507	1.13206098	-1.2392345	-0.5792842	-0.7412642	-0.6351342	-1.2267391	-0.8745352	-1.4145736	-0.2054785
28	-1.0614765	-0.0268107	-0.2592125	-1.0233462	-1.346001	-0.4514944	-0.3093644	-0.2201769	0.41987023	-0.0936932	-1.159425	0.00923329	0.64957808	-0.9570356	-0.3657573	-0.6731566	-0.6092057	-0.7459166	-0.424969	-1.159425	-0.2054785
29	-1.014387	-0.7256158	-0.7288492	-0.9684434	-0.8164283	-0.8324208	-0.3093644	-0.7420214	0.34462679	0.40116509	-0.9042765	-0.6883932	0.31184005	-0.6598183	-0.2305236	-0.5864742	-0.1998083	-0.5055053	-0.7141246	-0.9042765	-0.2054785
30	-0.6847603	-0.6650894	-0.560291	-0.7106085	-0.5990471	1.48718093	-0.3093644	-0.5853318	0.44495138	-0.4895798	0.88176336	-0.4791052	-1.1838569	-0.4688427	-0.4725207	0.21224251	-0.8821372	-0.4520806	1.58565212	0.88176336	-0.2054785
31	-1.4852823	-0.125854	-0.38827	-1.3963075	-0.1215205	0.74407334	-0.3093644	-0.2092768	0.56534088	-0.4301968	1.13691191	-0.9674438	-1.2321052	-1.2920772	-0.646901	1.50009577	-0.2680412	-0.55893	-1.3745456	1.13691191	-0.2054785
32	0.11576176	-0.0488203	-0.3286616	0.04812277	-0.240759	1.02133691	-0.3093644	-0.3073781	0.27941581	-0.1926648	0.62661482	0.14875859	-0.0741463	0.15229148	-0.792842	3.06037953	0.83323764	0.29586562	0.00675915	0.62661482	-0.2054785
33	-0.5434917	-0.3459501	0.24711777	-0.5402198	0.2064894	0.57573244	-0.3093644	0.07548952	-1.2430098	-0.9646437	1.64720901	-0.8279185	-1.3286018	-0.7308315	-0.8711043	-0.5988574	0.16864929	0.96367471	1.92818655	1.64720901	-0.2054785
34	1.62262676	-0.0598251	-0.3445891	1.6391933	-0.1526524	0.882893	-0.3093644	-0.2351464	0.39697548	0.949995	0.62661482	0.14875859	-0.5083809	1.78578165	-0.5913841	2.18736362	2.157950157	1.9253198	1.43916431	0.62661482	-0.2054785
35	1.81098469	1.5979462	1.9514833	1.85372972	1.61129246	0.67098851	-0.3093644	1.56063433	-1.1803069	-0.8755692	1.64720901	0.98591037	0.0705986	0.02064812	-0.966821	-0.7784138	0.61888637	0.53627689	-0.2457378	1.64720901	-0.2054785
36	-1.1556556	2.8454321	2.87476421	-1.1624379	2.33002891	1.25761995	-0.3093644	2.50213439	-0.131915	-1.0141295	-1.6697221	2.39826453	1.42155073	-1.5053024	-0.9992204	1.10383323	-3.0110034	-2.188842	1.2639066	-1.6697221	-0.2054785
37	0.1628513	0.19877867	0.2292881	0.12924564	0.30209142	1.61140076	-0.3093644	0.16269068	0.42237835	-0.7667004	0.88176336	0.63709713	-0.1223946	-0.1888688	-0.7074003	0.07602726	-0.8821372	-0.3452311	1.23792585	0.88176336	-0.2054785
38	0.63374661	2.04759489	1.86490459	0.62398653	2.33488706	0.44451847	-0.3093644	2.50894698	1.15902631	-0.9646437	-1.4145736	2.31140068	2.77250285	-0.552464	-0.9351624	-0.1998083	-1.8678358	-1.762821	-1.4145736	-0.2054785	
39	-0.8260289	-1.0997791	-1.0817547	-0.8257321	-1.3533457	-0.2484572	3.60155203	-1.0921886	0.44244326	0.5991084	0.1631772	-1.1069961	-0.9908638	-0.1425515	0.33176394	0.27415853	0.07312325	0.1623038	0.27145838	0.11631772	-0.2054785
40	-0.9672975	-1.1217888	-0.9458617	-0.949611	-1.1926293	-1.0009912	-0.3093644	-0.9927247	0.33459433	0.40116509	-0.6491279	-0.9674438	-0.363636	-0.4688427	0.69475969	-0.3450017	-0.0087562	0.2532682	-0.6964768	-0.6491279	-0.2054785

Example:

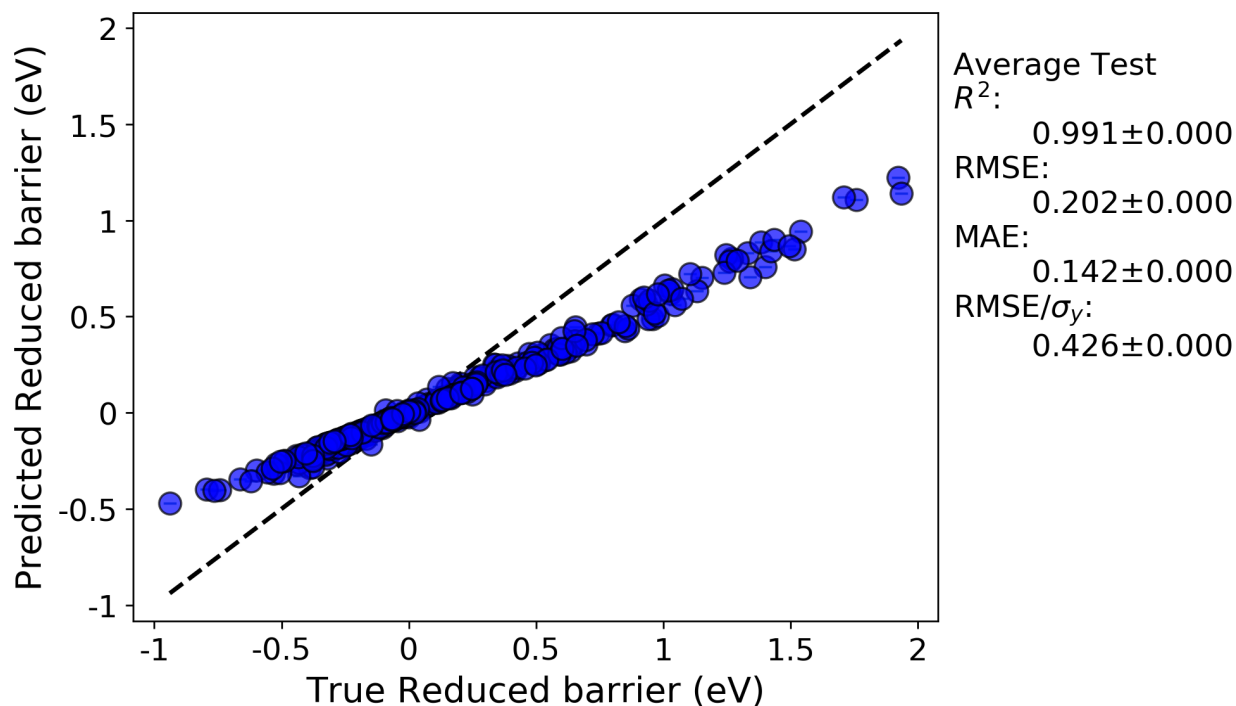
```
[Models]
[[KernelRidge]]
    kernel = rbf
    alpha = 1
    gamma = 1

[DataSplits]
[[NoSplit]]
[[RepeatedKFold]]
    n_splits = 5
    n_repeats = 2
```

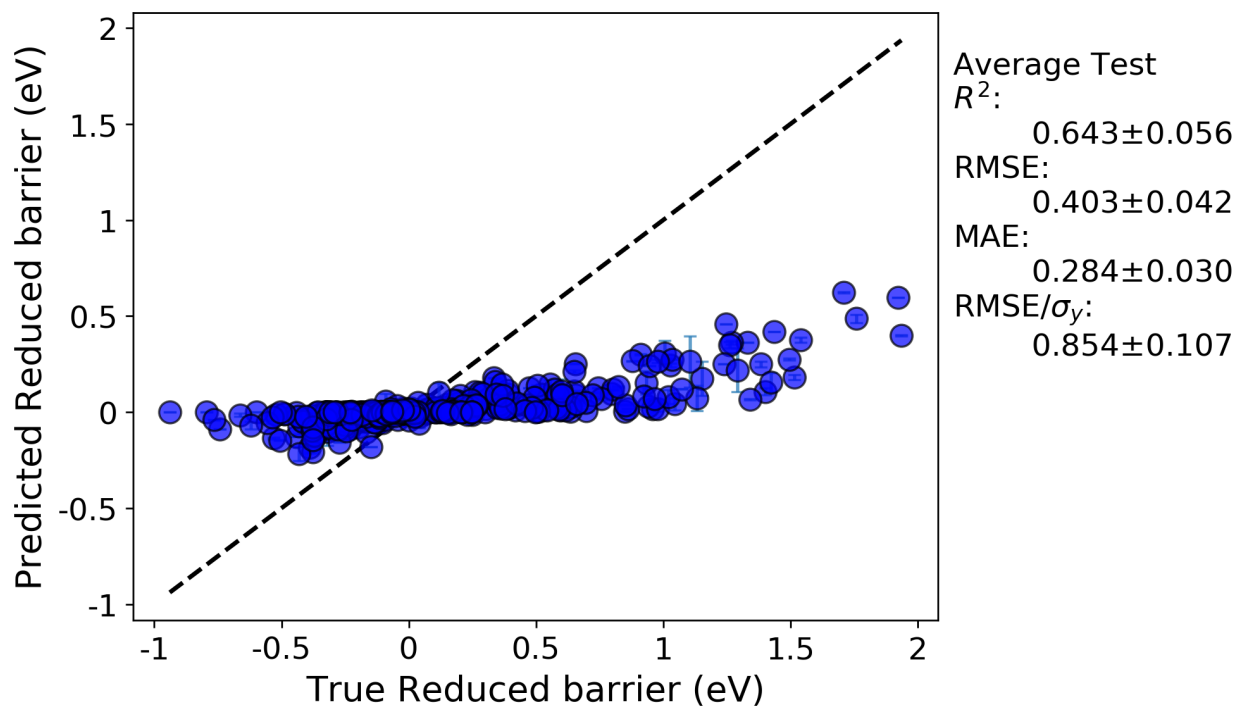
Below is a snapshot of the resulting directory tree generated from this MAST-ML run. You'll immediately notice the tree is deeper now, with a new level corresponding to each model we've fit (here just the single `KernelRidge` model), and, for each model, folders corresponding to each `DataSplit` test we denoted in the input file. For each data split method, there are folders and corresponding data plots and files for each hold-out split of the test. For instance, with the `RepeatedKfold` test, there were 10 total splits, which are labeled as `split_0` through `split_9`. Contained in each folder are numerous files, such as different data parity plots of predicted vs. actual values, histograms of residuals, .csv files for all plotted data, a .pkl file of the exported trained model, and .ipynb Jupyter notebooks useful for custom modifications of the data plots.



Below is a parity plot from the NoSplit (full data fit) run. The R-squared value is high, but there is significant mean error. This suggests that the model parameters are not optimal (which shouldn't be surprising considering we just picked them arbitrarily).



From examining the parity plot from the RepeatedKfold run (this is the 'average_points_with_bars.png' plot), which has the averaged values over all 10 splits, we can see that the predictions from random cross validation result in both a very low R-squared value and a high error. Essentially, cross-validation has shown that this model has no predictive ability. It seems our issues are two-fold: nonoptimal hyperparameters, and over-fitting. The over-fitting is evident due to the much worse before of the cross-validated parity plot compared to the full fit.



6.6 Feature selection and learning curves

As mentioned above, one problem with our current model is over-fitting. To further understand and minimize the effect of over-fitting, it is often necessary to construct learning curves and perform feature selection to obtain a reduced feature set which most accurately describes your data. To do this, we are going to add two additional sections to our input file.

The first section is related to feature selection. Here, we will use the `SequentialFeatureSelector` algorithm, which performs forward selection of features. We will select a total of 20 features, and use a `KernelRidge` model to evaluate the selected features. Here, we have denoted our estimator as “`KernelRidge_select`”. The models used in feature selection and learning curves are removed from the model queue, because in general one may want to use a different model for this step of the analysis than what will ultimately be used for fitting. Therefore, we need to also amend our models list to have this new `KernelRidge_select` model, as shown below.

Example:

```
[FeatureSelection]
  [[SequentialFeatureSelector]]
    estimator = KernelRidge_select
    k_features = 20

[Models]
  [[KernelRidge]]
    kernel = rbf
    alpha = 1
    gamma = 1
  [[KernelRidge_select]]
    kernel = rbf
    alpha = 1
    gamma = 1
```

The second section we will add is to plot learning curves. There are two types of learning curves MAST-ML can make: a data learning curve and feature learning curve. The former is a plot of the metric of interest versus the amount of training data used in the fits. The latter is a plot of the metric of interest versus the number of features comprising the X feature matrix. In the example `LearningCurve` input file section shown below, we are going to use a `KernelRidge` model, a random k-fold cross-validation and the `root_mean_squared_error` to evaluate our learning curves. We will also use a maximum of 20 features, and use the `SelectKBest` algorithm to assess the choice of features.

Example:

```
[LearningCurve]
  estimator = KernelRidge_learn
  cv = RepeatedKFold_learn
  scoring = root_mean_squared_error
  n_features_to_select = 20
  selector_name = SelectKBest
```

As with the above example of `FeatureSelection`, we need to add the `KernelRidge_learn` and `RepeatedKFold_learn` entries to the `Models` and `DataSplits` sections of our input file, respectively. At this point in the tutorial, the complete input file should look like this:

Example:

```
[GeneralSetup]
  input_features = Auto
  input_target = Reduced barrier (eV)
  randomizer = False
```

(continues on next page)

(continued from previous page)

```

metrics = Auto
input_other = Host element, Solute element, predict_Pt

[DataCleaning]
  cleaning_method = imputation
  imputation_strategy = mean

[FeatureGeneration]
  [[Magpie]]
    composition_feature = Solute element

[FeatureNormalization]
  [[StandardScaler]]

[FeatureSelection]
  [[SequentialFeatureSelector]]
    estimator = KernelRidge_select
    k_features = 20

[LearningCurve]
  estimator = KernelRidge_learn
  cv = RepeatedKFold_learn
  scoring = root_mean_squared_error
  n_features_to_select = 20
  selector_name = SelectKBest

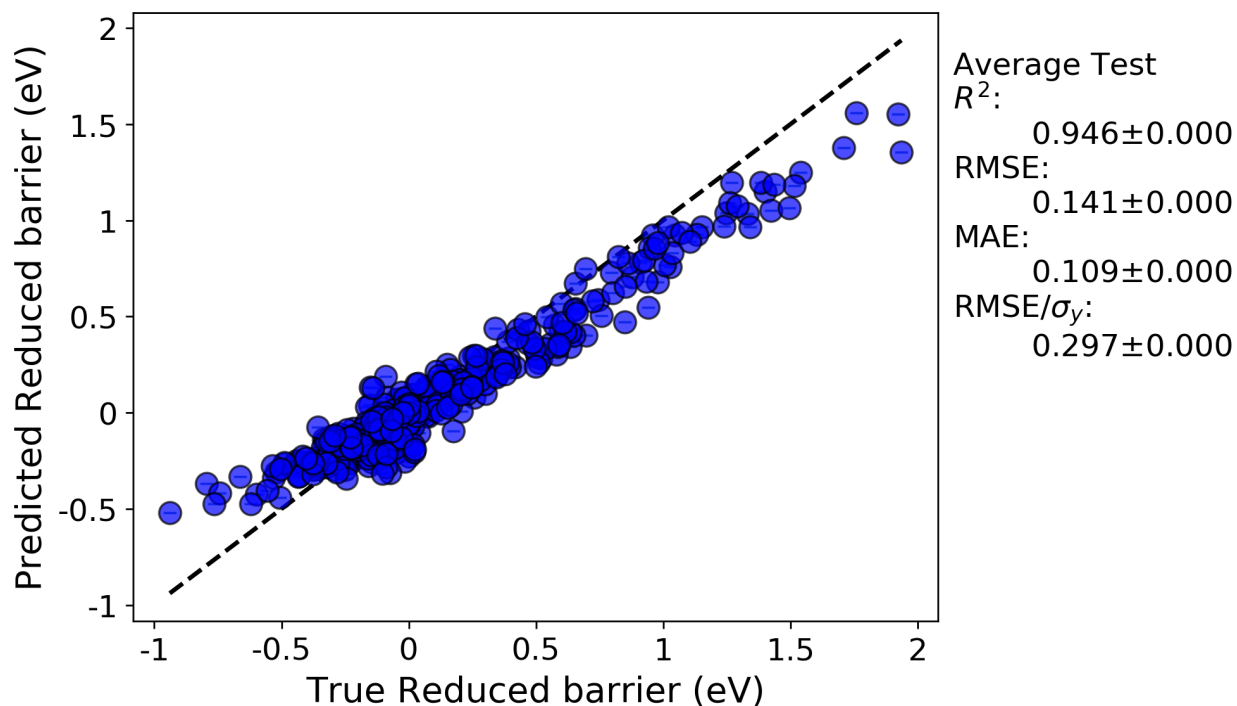
[Models]
  [[KernelRidge]]
    kernel = rbf
    alpha = 1
    gamma = 1
  [[KernelRidge_select]]
    kernel = rbf
    alpha = 1
    gamma = 1
  [[KernelRidge_learn]]
    kernel = rbf
    alpha = 1
    gamma = 1

[DataSplits]
  [[NoSplit]]
  [[RepeatedKFold]]
    n_splits = 5
    n_repeats = 2
  [[RepeatedKFold_learn]]
    n_splits = 5
    n_repeats = 2

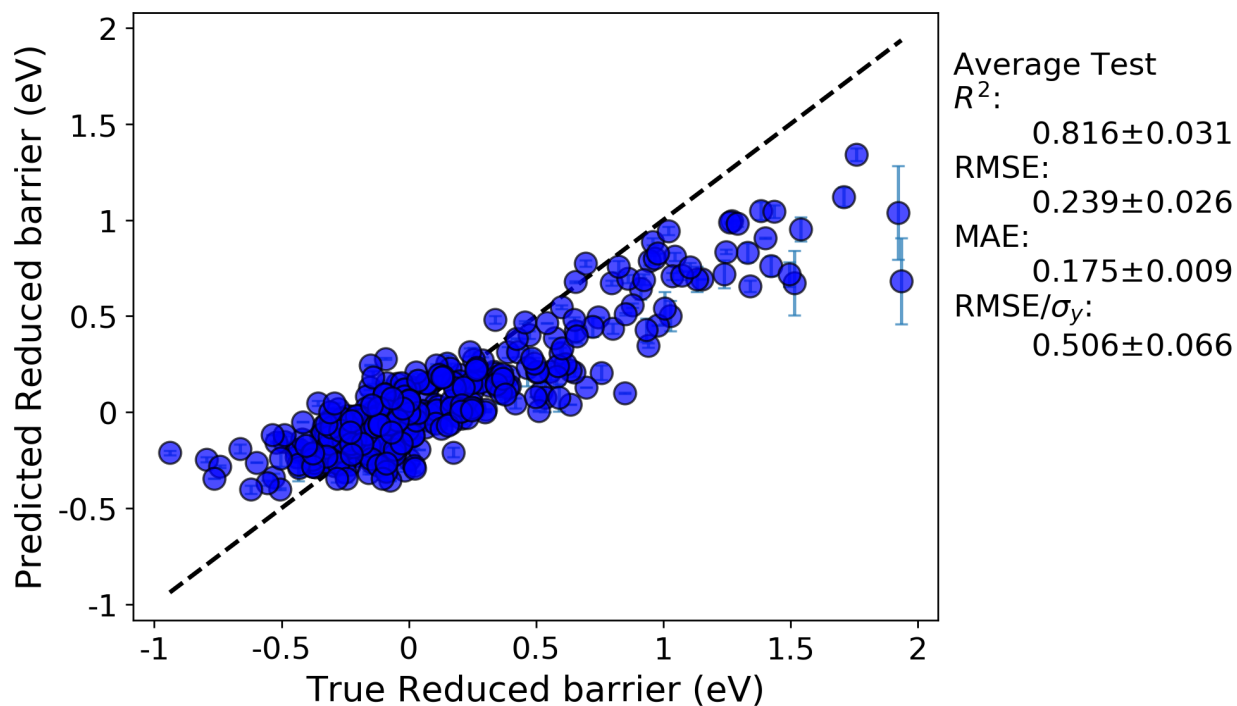
```

Let's take a look at the same full fit and RepeatedKFold random cross-validation tests for this run:

Full-fit:

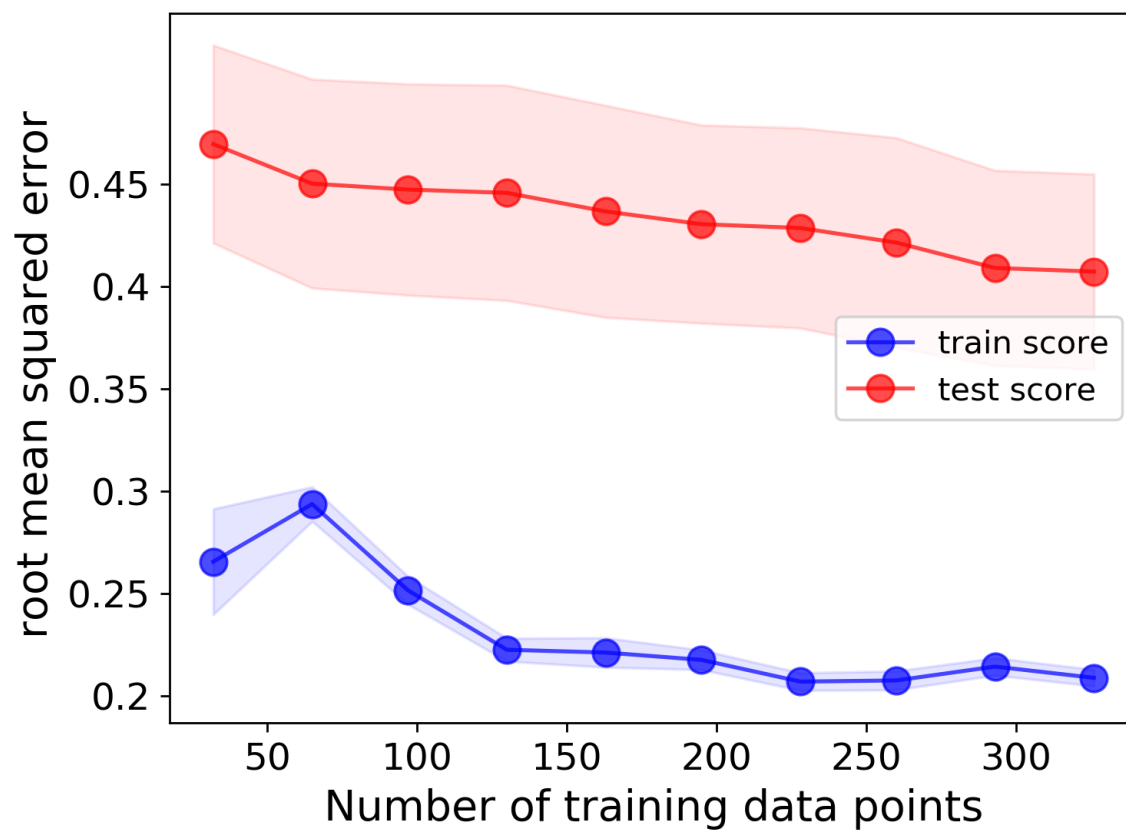


Random leave out cross-validation:

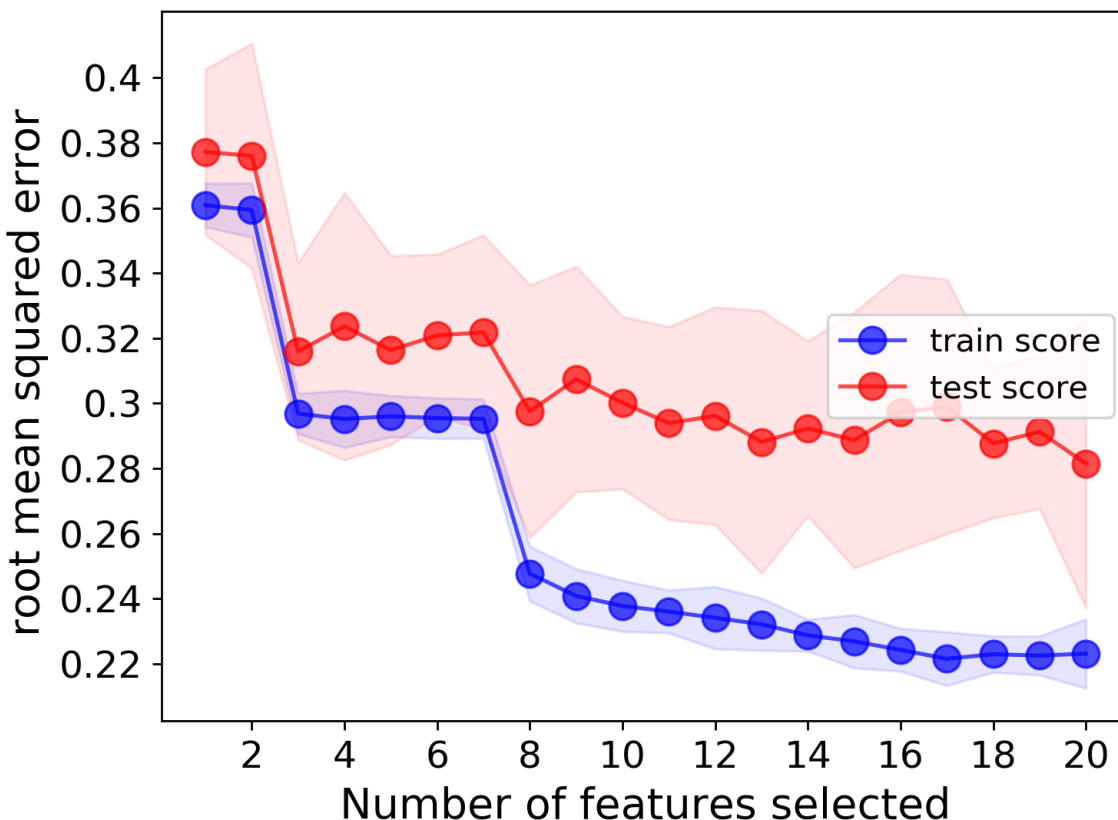


What we can see is, now that we down-selected features from more than 300 features in the previous run to just 20 here, that the fits have noticeably improved and the problem of over-fitting has been minimized. Below, we can look at the plotted learning curves

Data learning curve:



Feature learning curve:



We can clearly see that, as expected, having more training data will result in better test scores, and adding more features (up to a certain point) will also result in better fits. Based on these learning curves, one may be able to argue that additional features should could be used to further lower the error.

6.7 Hyperparameter optimization

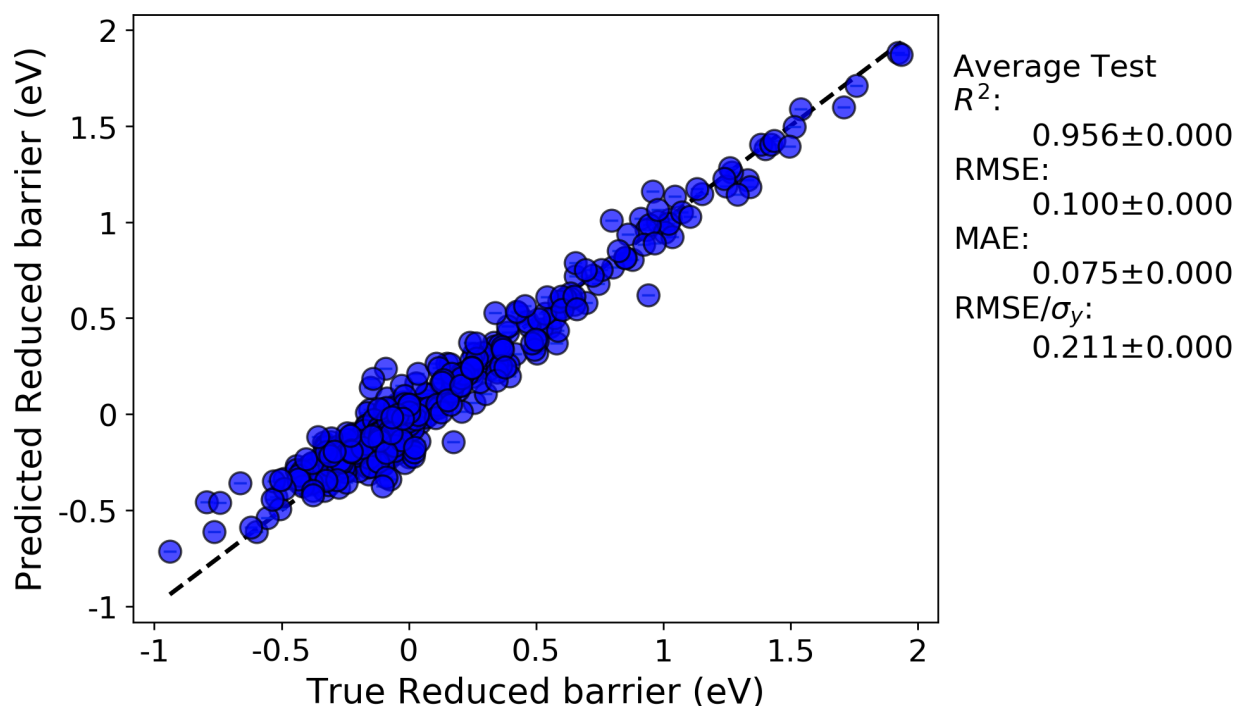
Next, we will consider optimization of the model hyperparameters, in order to use a better optimized model with a selected feature set to minimize the model errors. To do this, we need to add the HyperOpt section to our input file, as shown below. Here, we are optimizing our KernelRidge model, specifically its `root_mean_squared_error`, by using our RepeatedKfold random leave-out cross-validation scheme. The `param_names` field provides the parameter names to optimize. Here, we are optimizing the KernelRidge alpha and gamma parameters. Parameters must be delineated with a semicolon. The `param_values` field provides a bound on the values to search over. Here, the minimum value is -5, max is 5, 100 points are analyzed, and the numerical scaling is logarithmic, meaning it ranges from 10^{-5} to 10^5 . If “lin” instead of “log” would have been specified, the scale would be linear with 100 values ranging from -5 to 5.

Example:

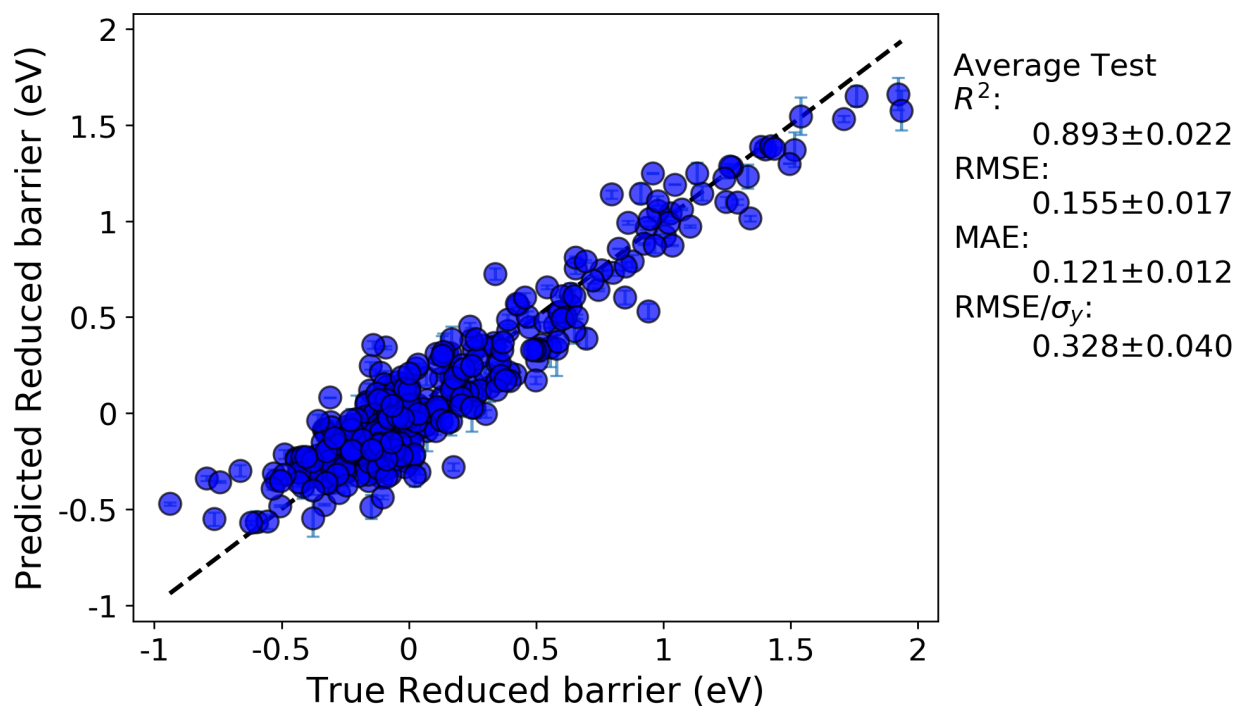
```
[HyperOpt]
[[GridSearch]]
    estimator = KernelRidge
    cv = RepeatedKfold
    param_names = alpha ; gamma
    param_values = -5 5 100 log float ; -5 5 100 log float
    scoring = root_mean_squared_error
```

Let’s take a final look at the same full fit and RepeatedKfold random cross-validation tests for this run:

Full-fit:



Random leave out cross-validation:



What we can see is, now that we down-selected features from more than 300 features in the previous run to just 20, along with optimizing the hyperparameters of our KernelRidge model, our fits are once again improved. The hyperparameter optimization portion of this workflow outputs the hyperparameter values and cross-validation scores for each step of, in this case, the GridSearch that we performed. All of this information is saved in the KerenlRidge.csv

file in the GridSearch folder in the results directory tree. For this run, the optimal hyperparameters were $\alpha = 0.034$ and $\gamma = 0.138$

6.8 Random leave-out versus leave-out-group cross-validation

Here, we will use our selected feature set and optimized KernelRidge hyperparameters from the previous section to do a new kind of cross-validation test: leave out group (LOG) CV. To do this, you will modify the α and γ values in the Models section, KernelRidge model in your input file. In addition, you can rename the selected.csv data file to a new name, for example “example_data_selected.csv”, and use the path to this new data file for this new run, as we will not be performing feature selection again (to save time).

We will compare these results to the results of LOG cross-validation with the random cross-validation. Our input data file had a column called “Host element”. This is a natural grouping to use for this problem, as it is interesting to assess our fits when training on a set of host elements and predicted the values of an entirely new host element set, without having ever trained on that set. Modify your input file to match what is shown below. Note that we have commented out the sections that we no longer want with the # symbol. You can either comment out the sections or remove them entirely.

Example:

```
[GeneralSetup]
    input_features = Auto
    input_target = Reduced barrier (eV)
    randomizer = False
    metrics = Auto
    input_other = Host element, Solute element, predict_Pt
    input_grouping = Host element

#[DataCleaning]
#    cleaning_method = imputation
#    imputation_strategy = mean

#[FeatureGeneration]
#    [[Magpie]]
#        composition_feature = Solute element

[FeatureNormalization]
    [[StandardScaler]]

#[FeatureSelection]
#    [[SequentialFeatureSelector]]
#        estimator = KernelRidge_select
#        k_features = 20

#[LearningCurve]
#    estimator = KernelRidge_learn
#    cv = RepeatedKfold_learn
#    scoring = root_mean_squared_error
#    n_features_to_select = 20
#    selector_name = SelectKBest

[Models]
    [[KernelRidge]]
        kernel = rbf
        alpha = 0.034
        gamma = 0.138
```

(continues on next page)

(continued from previous page)

```

# [[KernelRidge_select]]
#     kernel = rbf
#     alpha = 1
#     gamma = 1
# [[KernelRidge_learn]]
#     kernel = rbf
#     alpha = 1
#     gamma = 1

[DataSplits]
[[NoSplit]]
[[RepeatedKFold]]
    n_splits = 5
    n_repeats = 2
# [[RepeatedKFold_learn]]
#     n_splits = 5
#     n_repeats = 2
[[LeaveOneGroupOut]]
    grouping_column = Host element

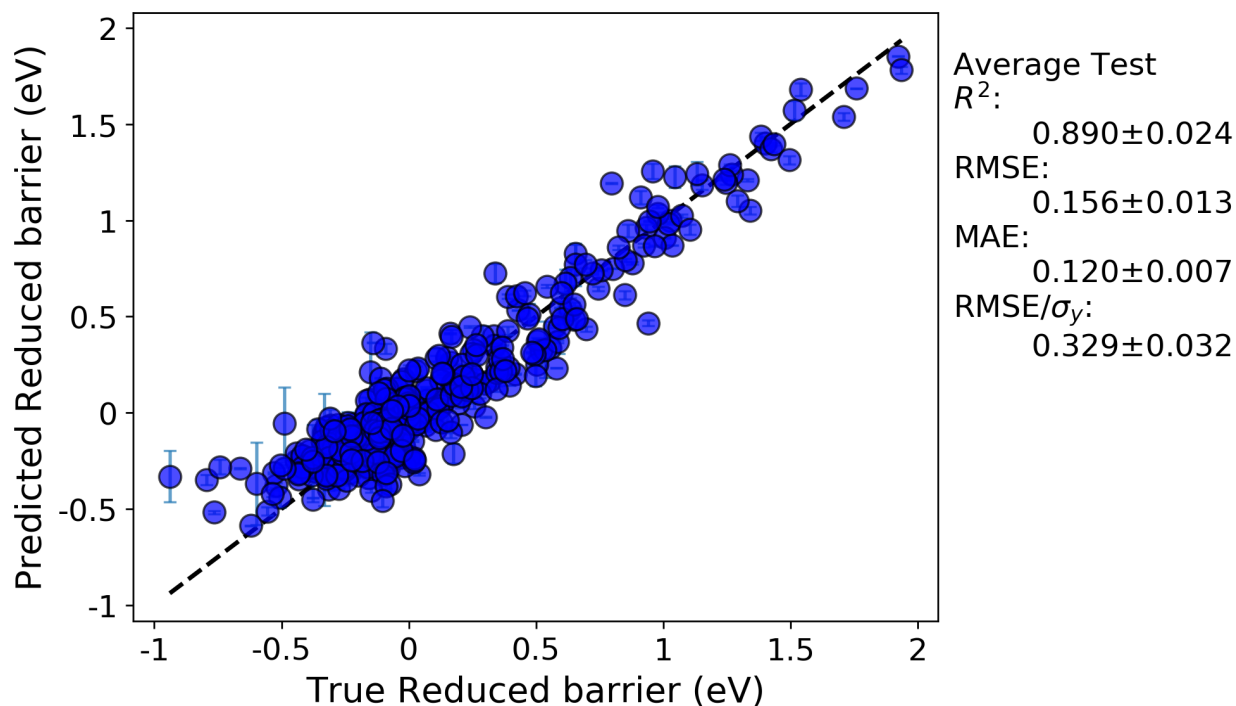
#[HyperOpt]
#     [[GridSearch]]
#         estimator = KernelRidge
#         cv = RepeatedKFold
#         param_names = alpha ; gamma
#         param_values = -5 5 100 log ; -5 5 100 log
#         scoring = root_mean_squared_error

```

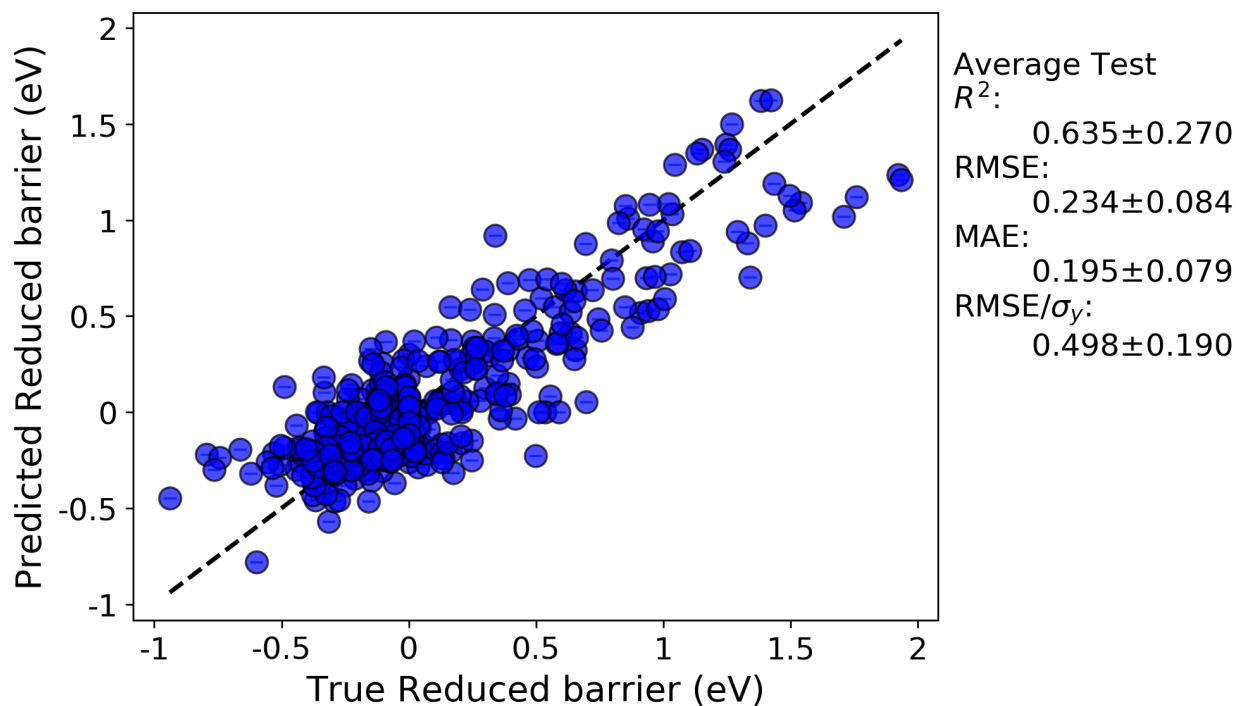
The main new additions to this input file is under the General Setup section, where the parameter `grouping_feature` needs to be added, and the addition of `LeaveOutGroup` to the `DataSplits` section.

By doing this run, we can assess the model fits resulting from the random cross-validation and the LOG cross-validation.

Random cross-validation:



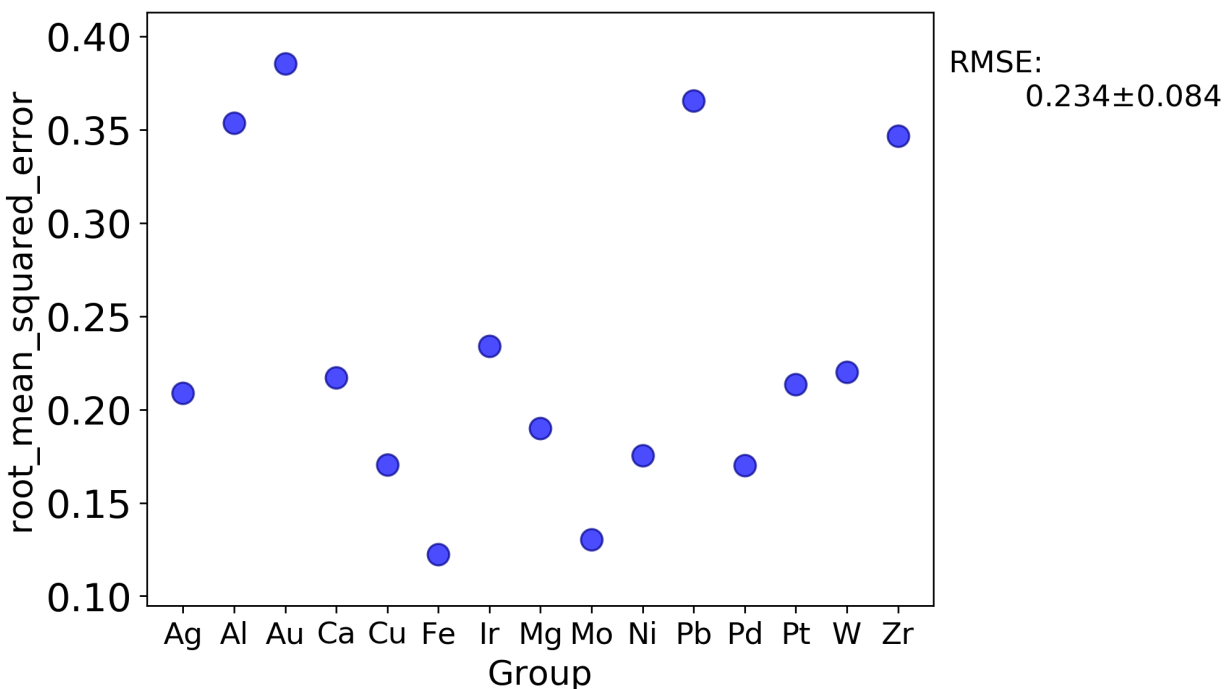
LOG cross-validation:



We can immediately see the R-squared and errors are both worse for the LOG cross-validation test compared to the random cross-validation test. This is likely because the LOG test is a more rigorous test of model extrapolation, because the test scores in each case are for data for which host elements were never included in the training set. In addition, a minor effect contributing to the reduced accuracy may be due to the fact that the model hyperparameters were optimized by evaluating the root mean squared error for a random cross-validation test. If instead the parameters

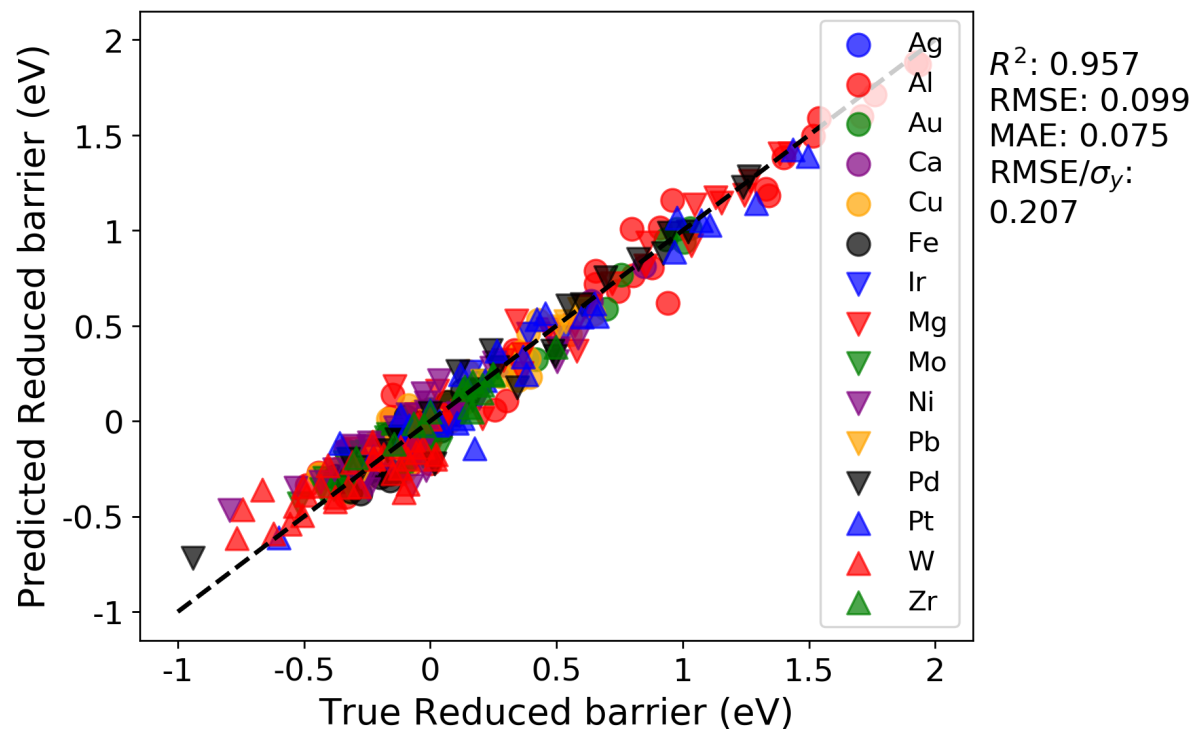
were optimized using the LOG test, the resulting fits would likely be improved.

There are a couple additional plots that are usual output for a LOG test that are worth drawing attention to. The first is a plot of each metric test value for each group. This enables one to quickly assess which groups perform better or worse than others.

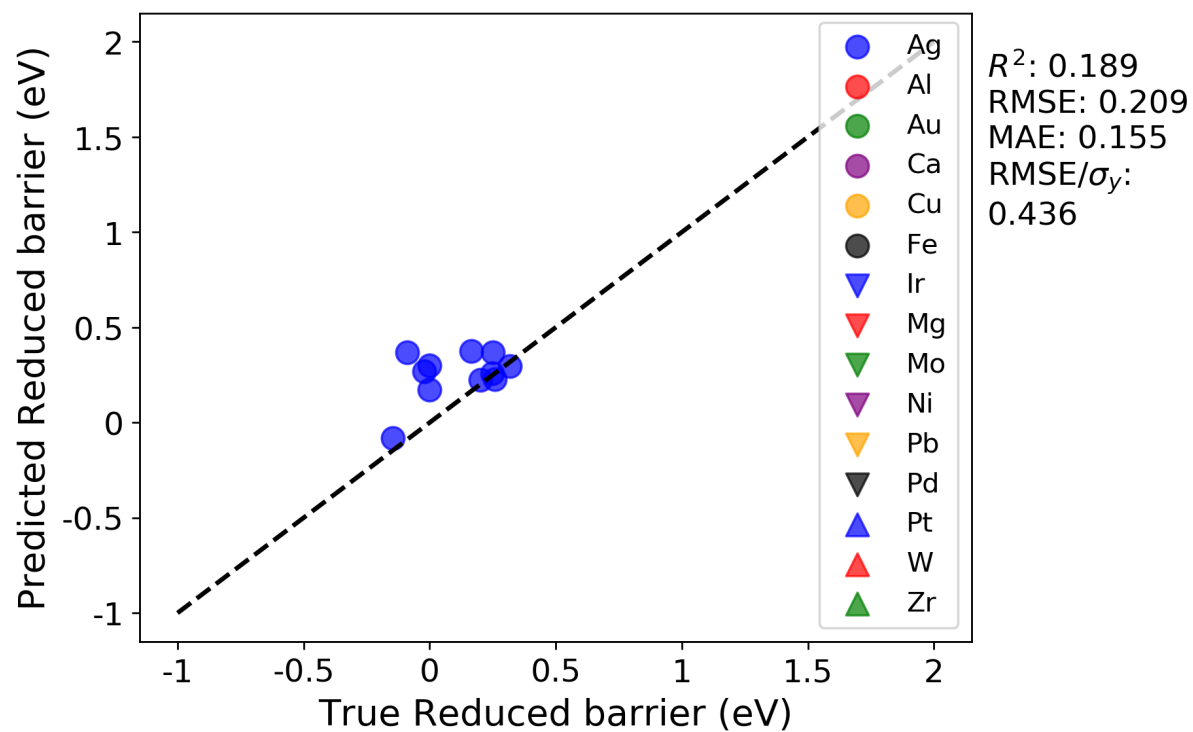


In addition, the parity plots for each split are now plotted with symbols denoting each group, which can help assess clustering of groups and goodness of fit on a per-group basis.

Training on all groups except Ag:



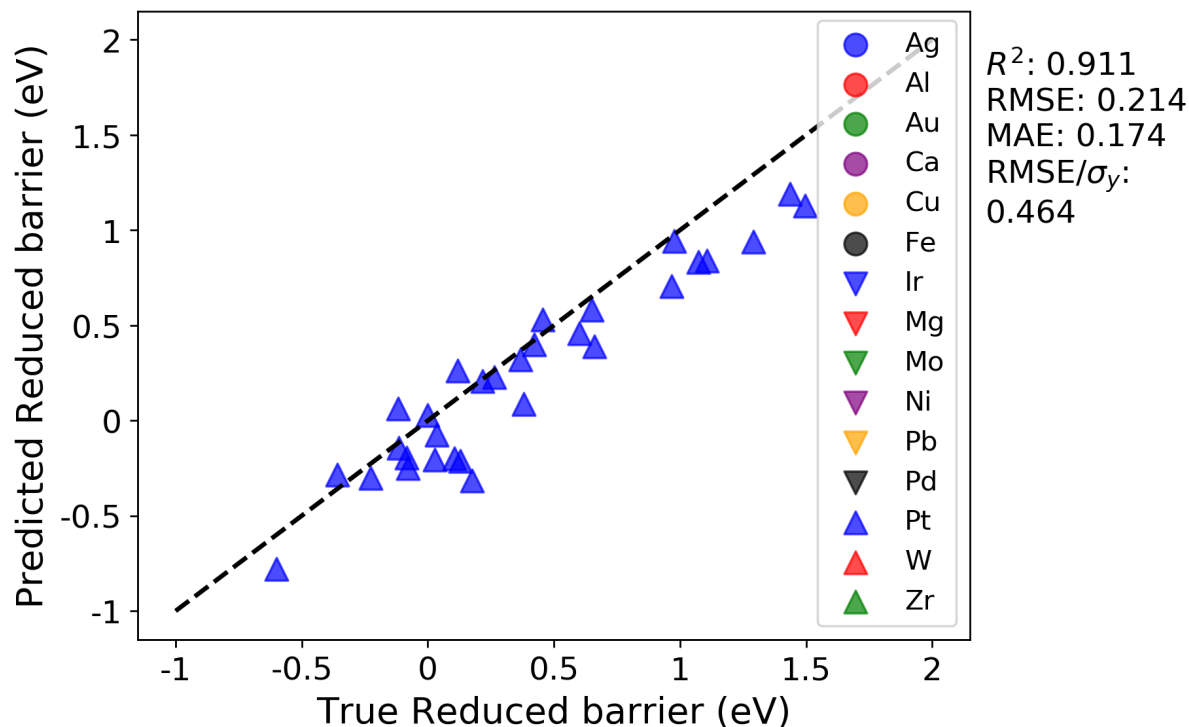
Testing on just Ag as the left-out host element:



6.9 Making predictions by importing a previously fit model

Here, we are going to import a previously fit model, and use it to predict the migration barriers for those data points with Pt as the host element.

In your previous run, the LOG test split where the Pt host values were predicted is in the split_12 folder. The parity plot for Pt test data should look like the below plot for your previous run:



Here, we are going to import the model that was fitted to all the groups except Pt, and use MAST-ML's data validation function as detailed above to obtain this same plot, but with using Pt as the validation data and the imported, previously trained model. If one were to extend this data set to include, for example, U as a host element, any number of previously trained models could be used to predict the migration barrier values for U. To import this model, save the KernelRidge_split_12.pkl file from your previous run into the /models/ folder (it is at the same level as the /tests/ folder in your main MAST-ML directory). To import this model into your next run, you can create a new field in the Models section, as shown below:

Example:

```
[Models]
# [[KernelRidge]]
#   kernel = rbf
#   alpha = 0.034
#   gamma = 0.138
# [[KernelRidge_select]]
#   kernel = rbf
#   alpha = 1
#   gamma = 1
# [[KernelRidge_learn]]
#   kernel = rbf
#   alpha = 1
#   gamma = 1
```

(continues on next page)

(continued from previous page)

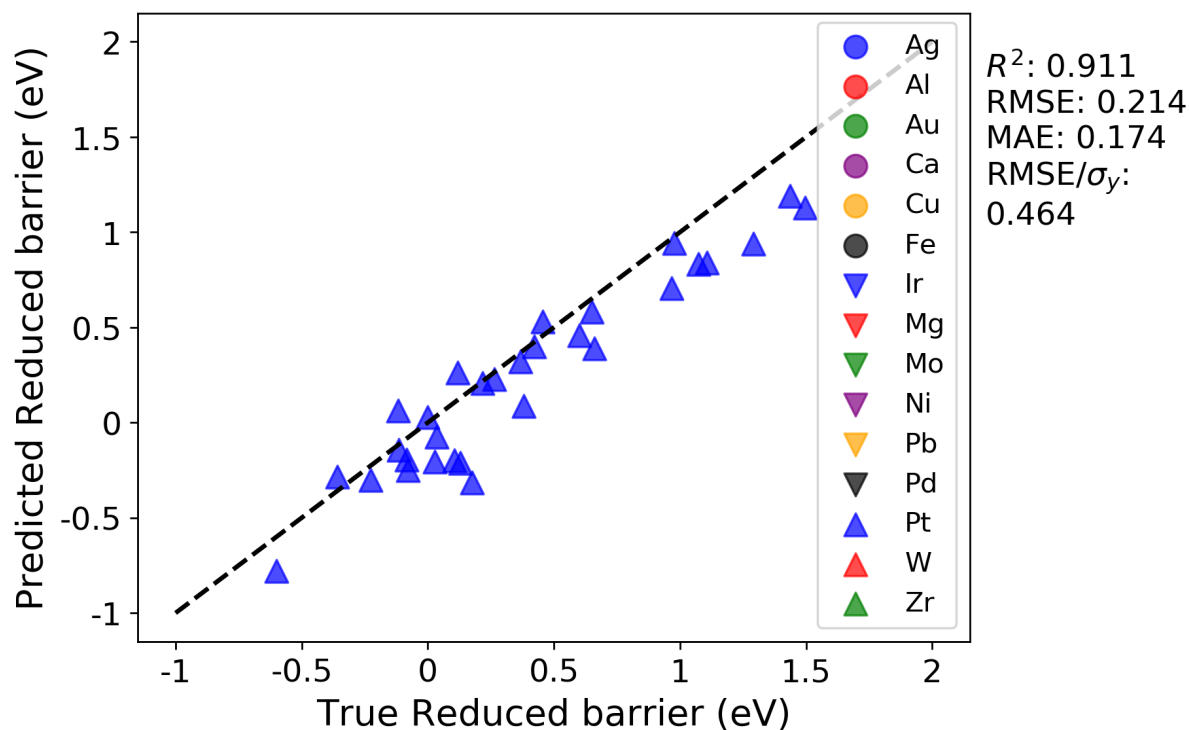
```
[[ModelImport]]
    model_path = models/KernelRidge_split_12.pkl
```

As we are only interested in assessing the fit on Pt for this example, we can change the DataSplits section to only have the LOG test:

Example:

```
[DataSplits]
    #[[NoSplit]]
    [[RepeatedKfold]]
        n_splits = 5
        n_repeats = 2
    #[[RepeatedKfold_learn]]
    #     n_splits = 5
    #     n_repeats = 2
    [[LeaveOneGroupOut]]
        grouping_column = Host element
```

From running this model and inspecting the test data parity plot in split_12 (the folder for Pt group, we obtain this parity plot:



As a comparison, this plot is exactly the same as the above plot from the previous run. This is the expected result, and demonstrates that the previously fit model was successfully imported and used to predict the Pt values. By inspecting the other groups, for example split_1, which is for Ag, the R squared and errors indicate a better fit than our previous run. This better fit is expected, as the model we saved from the previous run contained Ag in the training data, so these predictions on Ag should be improved (note that this defeats the purpose of the LOG test, but shows that the trained model we imported is behaving as expected).

6.10 Predicting values for new, extrapolated data

As a final example, we are going to use our model to predict the migration barriers for those data points with Pt as the host element. Your data file already has a column with the title “predict_Pt”, with values equal to 0 in all rows except where Pt is the host, in which case the value is 1. In the GeneralSetup section of your input file, add the parameter validation_columns, and have it equal to “predict_Pt”, as shown below. This will make it so that the data with Pt as the host element will never be involved in the model training. This feature is a convenient way to isolate part of your data, or some new part of your data, to only function as a validation data set. This way, whenever a model is trained and tested on the remaining data, an additional prediction will also be calculated, which here is for the Pt host data.

Example:

```
[GeneralSetup]
input_features = Auto
input_target = Reduced barrier (eV)
randomizer = False
metrics = Auto
input_other = Host element, Solute element, predict_Pt
input_grouping = Host element
input_testdata = predict_Pt
```

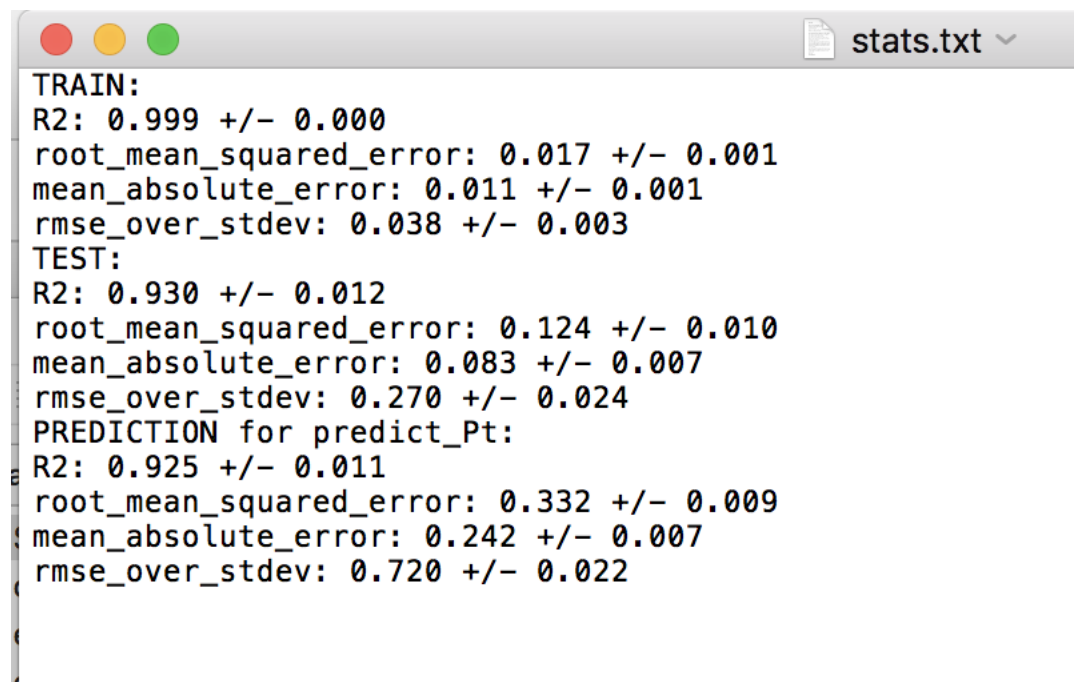
For this test, let’s run both the random cross-validation and LOG test. As a reminder, we need to un-comment the random cross-validation test in the DataSplits section:

Example:

```
[DataSplits]
#[[NoSplit]]
[[RepeatedKFold]]
    n_splits = 5
    n_repeats = 2
#[[RepeatedKFold_learn]]
#    n_splits = 5
#    n_repeats = 2
[[LeaveOneGroupOut]]
    grouping_column = Host element
```

When running this test, you’ll notice there are fewer splits in the LOG test folder now. This is because Pt is only treated as a final “validation” or “extrapolation” data set, and is never involved in the training or test set in any split. For each split in the random and LOG CV tests, there is a “stats.txt” file which is written, which provides the average train, test and prediction results. The prediction results are for the Pt validation data. Below are screenshots of the stats.txt file for the random and LOG tests.

Random cross-validation:

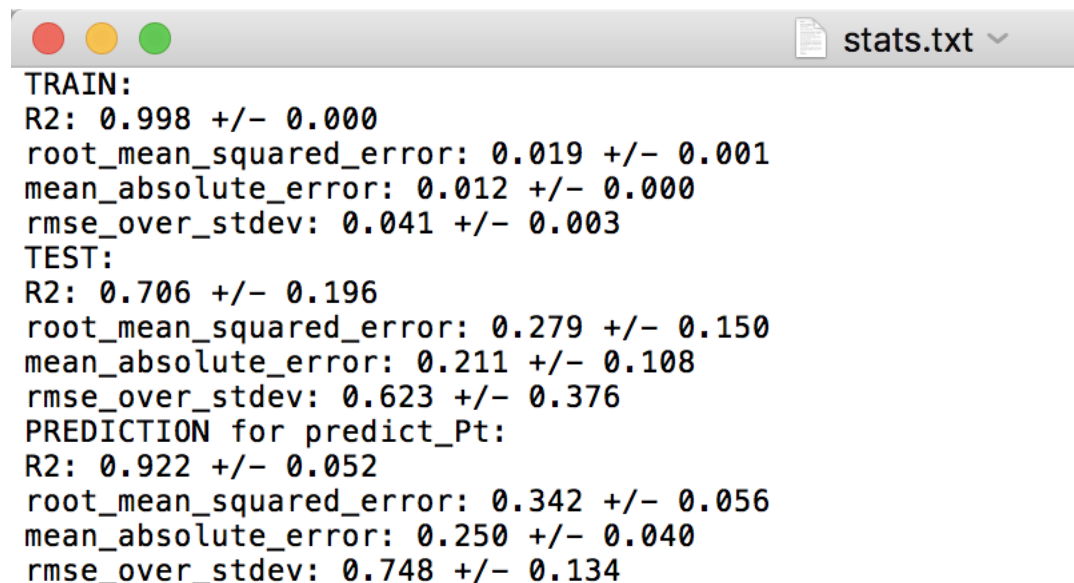


```

stats.txt
TRAIN:
R2: 0.999 +/- 0.000
root_mean_squared_error: 0.017 +/- 0.001
mean_absolute_error: 0.011 +/- 0.001
rmse_over_stdev: 0.038 +/- 0.003
TEST:
R2: 0.930 +/- 0.012
root_mean_squared_error: 0.124 +/- 0.010
mean_absolute_error: 0.083 +/- 0.007
rmse_over_stdev: 0.270 +/- 0.024
PREDICTION for predict_Pt:
R2: 0.925 +/- 0.011
root_mean_squared_error: 0.332 +/- 0.009
mean_absolute_error: 0.242 +/- 0.007
rmse_over_stdev: 0.720 +/- 0.022

```

LOG cross-validation:



```

stats.txt
TRAIN:
R2: 0.998 +/- 0.000
root_mean_squared_error: 0.019 +/- 0.001
mean_absolute_error: 0.012 +/- 0.000
rmse_over_stdev: 0.041 +/- 0.003
TEST:
R2: 0.706 +/- 0.196
root_mean_squared_error: 0.279 +/- 0.150
mean_absolute_error: 0.211 +/- 0.108
rmse_over_stdev: 0.623 +/- 0.376
PREDICTION for predict_Pt:
R2: 0.922 +/- 0.052
root_mean_squared_error: 0.342 +/- 0.056
mean_absolute_error: 0.250 +/- 0.040
rmse_over_stdev: 0.748 +/- 0.134

```

For the random cross-validation, the R-squared and error values are higher for the predict_Pt dataset compared to the average of the testing datasets. This is to be expected, as Pt is never involved in model training. Further, we can see that the predictions for predict_Pt are slightly worse in the case of the LOG cross-validation test compared to the random cross-validation test. This also makes sense, as each training split of the LOG test tends to result in worse predictive performance (i.e. worse model training), relative to the random cross-validation case, as discussed in the above test when we compared the results of the random and LOG cross-validation tests.

This concludes the MAST-ML tutorial document! There are some other features of MAST-ML which were not explicitly discussed in this tutorial, such as forming data clusters. Consult the MAST-ML Input File section of this

documentation for a more in-depth overview of all the possible options for different MAST-ML runs.

7.1 mastml.metrics Module

This module contains constructors for different model score metrics. Most model metrics are obtained from scikit-learn, while others are custom variations.

The full list of score functions in scikit-learn can be found at: http://scikit-learn.org/stable/modules/model_evaluation.html

7.1.1 Functions

<code>adjusted_r2_score(y_true, y_pred[, n_features])</code>	Method that calculates the adjusted R^2 value
<code>check_and_fetch_names(metric_names, ...)</code>	Method that checks whether chosen metrics to evaluate models are appropriate for user-specified models (e.g.
<code>r2_score_fitted(y_true, y_pred)</code>	Method that calculates the R^2 value
<code>r2_score_noint(y_true, y_pred)</code>	Method that calculates the R^2 value without fitting the y-intercept
<code>rmse_over_stdev(y_true, y_pred[, train_y])</code>	Method that calculates the root mean squared error (RMSE) of a set of data, divided by the standard deviation of the training data set.
<code>root_mean_squared_error(y_true, y_pred)</code>	Method that calculates the root mean squared error (RMSE)

adjusted_r2_score

`mastml.metrics.adjusted_r2_score(y_true, y_pred, n_features=None)`

Method that calculates the adjusted R^2 value

Args: `y_true`: (numpy array), array of true y data values `y_pred`: (numpy array), array of predicted y data values
`n_features`: (int), number of features used in the fit

Returns: (float): score of adjusted R^2

check_and_fetch_names

`mastml.metrics.check_and_fetch_names` (*metric_names, is_classification*)

Method that checks whether chosen metrics to evaluate models are appropriate for user-specified models (e.g. classification vs. regression models)

Args: `metric_names`: (numpy array), array of true y data values `is_classification`: (bool), whether the task is a classification task

Returns: functions (dict): dict containing the appropriate metric objects (e.g. classification vs. regression metrics)

r2_score_fitted

`mastml.metrics.r2_score_fitted` (*y_true, y_pred*)

Method that calculates the R^2 value

Args: `y_true`: (numpy array), array of true y data values `y_pred`: (numpy array), array of predicted y data values

Returns: (float): score of R^2

r2_score_noint

`mastml.metrics.r2_score_noint` (*y_true, y_pred*)

Method that calculates the R^2 value without fitting the y-intercept

Args: `y_true`: (numpy array), array of true y data values `y_pred`: (numpy array), array of predicted y data values

Returns: (float): score of R^2 with no y-intercept

rmse_over_stdev

`mastml.metrics.rmse_over_stdev` (*y_true, y_pred, train_y=None*)

Method that calculates the root mean squared error (RMSE) of a set of data, divided by the standard deviation of the training data set.

Args: `y_true`: (numpy array), array of true y data values `y_pred`: (numpy array), array of predicted y data values
`train_y`: (numpy array), array of training y data values

Returns: (float): score of RMSE divided by standard deviation of training data

root_mean_squared_error

`mastml.metrics.root_mean_squared_error` (*y_true, y_pred*)

Method that calculates the root mean squared error (RMSE)

Args: `y_true`: (numpy array), array of true y data values `y_pred`: (numpy array), array of predicted y data values

Returns: (float): score of RMSE

Code Documentation: Configuration file parser

8.1 mastml.conf_parser Module

The `conf_parser` module is used for handling, parsing, and checking MAST-ML input configuration files

8.1.1 Functions

<code>check_models_mixed(model_names)</code>	Method used to check whether the user has mixed regression and classification tasks
<code>fix_types(maybe_list)</code>	Method that returns true datatype of values passed as string or list of strings, parsed from configuration file
<code>make_scorer(score_func, *[, ...])</code>	Make a scorer from a performance metric or loss function.
<code>mybool(string)</code>	Method that converts a string equal to 'True' or 'False' into type bool
<code>parse_conf_file(filepath[, from_dict])</code>	Method that accepts the filepath of an input configuration file and returns its parsed dictionary

fix_types

`mastml.conf_parser.fix_types` (*maybe_list*)

Method that returns true datatype of values passed as string or list of strings, parsed from configuration file

Args: *maybe_list*: (list, str), a list of strings or just a string whose datatype should be e.g. int or list of float

Returns: *maybe_list*: (list, bool, int, float): a list of items or other data type converted from string to correct data type

mybool

`mastml.conf_parser.mybool` (*string*)

Method that converts a string equal to 'True' or 'False' into type bool

Args: string: (str), a string as 'True' or 'False'

Returns: bool: (bool): bool as True or False

parse_conf_file

`mastml.conf_parser.parse_conf_file` (*filepath, from_dict=False*)

Method that accepts the filepath of an input configuration file and returns its parsed dictionary

Args: filepath: (str), path to config file, or a dict of config values directly

Returns: conf: (dict): dictionary parsed from config file

9.1 mastml.data_cleaner Module

The `data_cleaner` module is used to clean missing or NaN values from pandas dataframes (e.g. removing NaN, imputation, etc.)

9.1.1 Functions

<code>columns_with_strings(df)</code>	Method that ascertains which columns in data contain string entries
<code>flag_outliers(df, conf_not_input_features, ...)</code>	Method that scans values in each X feature matrix column and flags values that are larger than 3 standard deviations from the average of that column value.
<code>imputation(df, strategy[, cols_to_leave_out])</code>	Method that imputes values to the missing places based on the median, mean, etc.
<code>orth(A[, rcond])</code>	Construct an orthonormal basis for the range of A using SVD
<code>ppca(df[, cols_to_leave_out])</code>	Method that performs a recursive PCA routine to use PCA of known columns to fill in missing values in particular column
<code>remove(df, axis)</code>	Method that removes a full column or row of data values if one column or row contains NaN or is blank

columns_with_strings

`mastml.data_cleaner.columns_with_strings(df)`

Method that ascertains which columns in data contain string entries

Args: `df`: (dataframe), pandas dataframe containing data

Returns: `str_columns`: (list), list containing indices of columns containing strings

flag_outliers

`mastml.data_cleaner.flag_outliers(df, conf_not_input_features, savepath, n_stdevs=3)`

Method that scans values in each X feature matrix column and flags values that are larger than 3 standard deviations from the average of that column value. The index and column values of potentially problematic points are listed and written to an output file.

Args: df: (dataframe), pandas dataframe containing data

Returns: None, just writes results to file

imputation

`mastml.data_cleaner.imputation(df, strategy, cols_to_leave_out=None)`

Method that imputes values to the missing places based on the median, mean, etc. of the data in the column

Args: df: (dataframe), pandas dataframe containing data strategy: (str), method of imputation, e.g. median, mean, etc. cols_to_leave_out: (list), list of column indices to not include in imputation

Returns: df: (dataframe): dataframe with NaN or missing values resolved via imputation

ppca

`mastml.data_cleaner.ppca(df, cols_to_leave_out=None)`

Method that performs a recursive PCA routine to use PCA of known columns to fill in missing values in particular column

Args: df: (dataframe), pandas dataframe containing data cols_to_leave_out: (list), list of column indices to not include in imputation

Returns: df: (dataframe): dataframe with NaN or missing values resolved via imputation

remove

`mastml.data_cleaner.remove(df, axis)`

Method that removes a full column or row of data values if one column or row contains NaN or is blank

Args: df: (dataframe), pandas dataframe containing data axis: (int), whether to remove rows (axis=0) or columns (axis=1)

Returns: df: (dataframe): dataframe with NaN or missing values removed

9.1.2 Classes

<code>PPCA()</code>	Class to perform probabilistic principal component analysis (PPCA) to fill in missing data.
<code>SimpleImputer(*[, missing_values, strategy, ...])</code>	Imputation transformer for completing missing values.

PPCA

class `mastml.data_cleaner.PPCA`

Bases: `object`

Class to perform probabilistic principal component analysis (PPCA) to fill in missing data.

This PPCA routine was taken directly from <https://github.com/allentran/pca-magic>. Due to import errors, for ease of use we have elected to copy the module here. This github repo was last accessed on 8/27/18. The code comprising the PPCA class below was not developed by and is not owned by the University of Wisconsin-Madison MAST-ML development team.

Methods Summary

`fit(data[, d, tol, min_obs, verbose])`

`load(fpath)`

`save(fpath)`

`transform([data])`

Methods Documentation

fit (*data*, *d=None*, *tol=0.0001*, *min_obs=10*, *verbose=False*)

load (*fpath*)

save (*fpath*)

transform (*data=None*)

9.1.3 Class Inheritance Diagram



10.1 mastml.data_loader Module

The data_loader module is used for importing data from user-specified csv or xlsx file to MAST-ML

10.1.1 Functions

<code>load_data(file_path[, input_features, ...])</code>	Method that accepts the filepath of an input data file and returns a full dataframe and parsed X and y dataframes
--	---

load_data

`mastml.data_loader.load_data` (*file_path*, *input_features=None*, *input_target=None*, *input_grouping=None*, *feature_blacklist=[]*)

Method that accepts the filepath of an input data file and returns a full dataframe and parsed X and y dataframes

Args: *file_path*: (str), path to data file

input_features: (str), column names to be used as input features (X data). If 'Auto', then takes all columns that are not listed in *target_feature* or *feature_blacklist* fields.

target_feature: (str), column name for data to be fit to (y data).

grouping_feature: (str), column names used to group data in user-defined grouping scheme

Returns: *df*: (dataframe), full dataframe of the input X data (y data is removed)

X: (dataframe), dataframe containing only the X data from the data file

X_noinput: (dataframe), dataframe containing the columns of the original X data that are not used as input features

X_grouped: (dataframe), dataframe containing the columns of the original X data that correspond to a data grouping scheme

y: (dataframe), dataframe containing only the y data from the data file

11.1 mastml.learning_curve Module

This module contains methods to construct learning curves, which evaluate some cross-validation performance metric (e.g. RMSE) as a function of amount of training data (i.e. a sample learning curve) or as a function of the number of features used in the fitting (i.e. a feature learning curve).

11.1.1 Functions

<code>f_regression(X, y, *[, center])</code>	Univariate linear regression tests.
<code>feature_learning_curve(X, y, estimator, cv, ...)</code>	Method that calculates data used to plot a feature learning curve, e.g.
<code>learning_curve(estimator, X, y, *[, groups, ...])</code>	Learning curve.
<code>sample_learning_curve(X, y, estimator, cv, ...)</code>	Method that calculates data used to plot a sample learning curve, e.g.

feature_learning_curve

`mastml.learning_curve.feature_learning_curve` (*X, y, estimator, cv, scoring, selector_name, savepath, n_features_to_select=None, Xgroups=None*)

Method that calculates data used to plot a feature learning curve, e.g. the RMSE of a cross-validation routine using a specified model and a given number of features

Args: *X*: (numpy array), array of X data values

y: (numpy array), array of y data values

estimator: (scikit-learn model object), a scikit-learn model used for fitting

cv: (scikit-learn cross validation object), a scikit-learn cross validation object to construct train/test splits

scoring: (scikit-learn metric object), a scikit-learn metric to use as a scorer

selector_name: (str), name of a scikit-learn or MAST-ML feature selection routine

n_features_to_select: (int), total number of features to select, i.e. stopping criterion for number of features

Xgroups: (list), list of row indices corresponding to each group

Returns: train_sizes: (numpy array), array of fractions of training data used in learning curve

train_mean: (numpy array), array of means of training data scores for each number of features

test_mean: (numpy array), array of means of testing data scores for each number of features

train_stddev: (numpy array), array of standard deviations of training data scores for each number of features

test_stddev: (numpy array), array of standard deviations of testing data scores for each number of features

sample_learning_curve

mastml.learning_curve.sample_learning_curve(*X*, *y*, *estimator*, *cv*, *scoring*, *Xgroups=None*)

Method that calculates data used to plot a sample learning curve, e.g. the RMSE of a cross-validation routine using a specified model and a given fraction of the total training data

Args: X: (numpy array), array of X data values

y: (numpy array), array of y data values

estimator: (scikit-learn model object), a scikit-learn model used for fitting

cv: (scikit-learn cross validation object), a scikit-learn cross validation object to construct train/test splits

scoring: (scikit-learn metric object), a scikit-learn metric to use as a scorer

Xgroups: (list), list of row indices corresponding to each group

Returns: train_sizes: (numpy array), array of fractions of training data used in learning curve

train_mean: (numpy array), array of means of training data scores for each training data fraction

test_mean: (numpy array), array of means of testing data scores for each training data fraction

train_stddev: (numpy array), array of standard deviations of training data scores for each training data fraction

test_stddev: (numpy array), array of standard deviations of testing data scores for each training data fraction

12.1 mastml.legos.clusterers Module

The clusterers module is used for instantiating cluster algorithm objects from scikit-learn. More information is available at <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster>

13.1 mastml.legos.data_splitters Module

The `data_splitters` module contains a collection of classes for generating (`train_indices`, `test_indices`) pairs from a dataframe or a numpy array.

For more information and a list of scikit-learn splitter classes, see: http://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection

13.1.1 Classes

<code>BaseEstimator</code>	Base class for all estimators in scikit-learn.
<code>Bootstrap(n[, n_bootstraps, train_size, ...])</code>	# Note: Bootstrap taken directly from sklearn Github (https://github.com/scikit-learn/scikit-learn/blob/0.11.X/sklearn/cross_validation.py) # which was necessary as it was later removed from more recent sklearn releases Random sampling with replacement cross-validation iterator Provides train/test indices to split data in train test sets while resampling the input <code>n_bootstraps</code> times: each time a new random split of the data is performed and then samples are drawn (with replacement) on each side of the split to build the training and test sets.
<code>JustEachGroup()</code>	Class to train the model on one group at a time and test it on the rest of the data This class wraps scikit-learn's <code>LeavePGroupsOut</code> with <code>P</code> set to <code>n-1</code> .
<code>LeaveCloseCompositionsOut([dist_threshold, ...])</code>	Leave-P-out where you exclude materials with compositions close to those the test set
<code>LeaveOutPercent([percent_leave_out, n_repeats])</code>	Class to train the model using a certain percentage of data as training data

Continued on next page

Table 1 – continued from previous page

<code>NearestNeighbors(*[, n_neighbors, radius, ...])</code>	Unsupervised learner for implementing neighbor searches.
<code>NoSplit()</code>	Class to just train the model on the training data and test it on that same data.
<code>SplittersUnion(splitters)</code>	Class to take the union of two separate splitting routines, so that many splitting routines can be performed at once
<code>TransformerMixin</code>	Mixin class for all transformers in scikit-learn.

Bootstrap

```
class mastml.legos.data_splitters.Bootstrap(n,      n_bootstraps=3,      train_size=0.5,
                                           test_size=None,      n_train=None,
                                           n_test=None, random_state=0)
```

Bases: object

Note: Bootstrap taken directly from sklearn Github (https://github.com/scikit-learn/scikit-learn/blob/0.11.X/sklearn/cross_validation.py) # which was necessary as it was later removed from more recent sklearn releases
Random sampling with replacement cross-validation iterator Provides train/test indices to split data in train test sets while resampling the input n_bootstraps times: each time a new random split of the data is performed and then samples are drawn (with replacement) on each side of the split to build the training and test sets. Note: contrary to other cross-validation strategies, bootstrapping will allow some samples to occur several times in each splits. However a sample that occurs in the train split will never occur in the test split and vice-versa. If you want each sample to occur at most once you should probably use ShuffleSplit cross validation instead.

Args:

n [int] Total number of elements in the dataset.

n_bootstraps [int (default is 3)] Number of bootstrapping iterations

train_size [int or float (default is 0.5)] If int, number of samples to include in the training split (should be smaller than the total number of samples passed in the dataset). If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split.

test_size [int or float or None (default is None)] If int, number of samples to include in the training set (should be smaller than the total number of samples passed in the dataset). If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If None, n_test is set as the complement of n_train.

random_state [int or RandomState] Pseudo number generator state used for random sampling.

Attributes Summary

indices

Methods Summary

get_n_splits([X, y, groups])

split(X, y[, groups])

Attributes Documentation

indices = True

Methods Documentation

get_n_splits (*X=None, y=None, groups=None*)

split (*X, y, groups=None*)

JustEachGroup

class mastml.legos.data_splitters.**JustEachGroup**

Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin

Class to train the model on one group at a time and test it on the rest of the data This class wraps scikit-learn's LeavePGroupsOut with P set to n-1. More information is available at: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.LeavePGroupsOut.html

Args: None (only object instance)

Methods: get_n_splits: method to calculate the number of splits to perform

Args: groups: (numpy array), array of group labels

Returns: (int), number of unique groups, indicating number of splits to perform

split: method to perform split into train indices and test indices

Args: X: (numpy array), array of X features y: (numpy array), array of y data groups: (numpy array), array of group labels

Returns: (numpy array), array of train and test indices

Methods Summary

get_n_splits([X, y, groups])

split(X, y, groups)

Methods Documentation

get_n_splits (*X=None, y=None, groups=None*)

split (*X, y, groups*)

LeaveCloseCompositionsOut

class mastml.legos.data_splitters.**LeaveCloseCompositionsOut** (*dist_threshold=0.1, nn_kwargs=None*)

Bases: sklearn.model_selection._split.BaseCrossValidator

Leave-P-out where you exclude materials with compositions close to those the test set

Computes the distance between the element fraction vectors. For example, the L_2 distance between Al and Cu is $\sqrt{2}$ and the L_1 distance between Al and Al0.9Cu0.1 is 0.2.

Consequently, this splitter requires a list of compositions as the input to *split* rather than the features.

Args:

dist_threshold (float): Entries must be farther than this distance to be included in the training set

nn_kwargs (dict): Keyword arguments for the scikit-learn NearestNeighbor class used to find nearest points

Methods Summary

<code>get_n_splits([X, y, groups])</code>	Returns the number of splitting iterations in the cross-validator
<code>split(X[, y, groups])</code>	Generate indices to split data into training and test set.

Methods Documentation

get_n_splits (*X=None, y=None, groups=None*)

Returns the number of splitting iterations in the cross-validator

split (*X, y=None, groups=None*)

Generate indices to split data into training and test set.

X [array-like of shape (n_samples, n_features)] Training data, where n_samples is the number of samples and n_features is the number of features.

y [array-like of shape (n_samples,)] The target variable for supervised learning problems.

groups [array-like of shape (n_samples,), default=None] Group labels for the samples used while splitting the dataset into train/test set.

train [ndarray] The training set indices for that split.

test [ndarray] The testing set indices for that split.

LeaveOutPercent

class mastml.legos.data_splitters.**LeaveOutPercent** (*percent_leave_out=0.2, n_repeats=5*)

Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin

Class to train the model using a certain percentage of data as training data

Args: percent_leave_out (float): fraction of data to use in training (must be > 0 and < 1)

n_repeats (int): number of repeated splits to perform (must be >= 1)

Methods: get_n_splits: method to return the number of splits to perform

Args: groups: (numpy array), array of group labels

Returns: (int), number of unique groups, indicating number of splits to perform

split: method to perform split into train indices and test indices

Args: X: (numpy array), array of X features y: (numpy array), array of y data groups: (numpy array), array of group labels

Returns: (numpy array), array of train and test indices

Methods Summary

```
get_n_splits([X, y, groups])
split(X, y[, groups])
```

Methods Documentation

get_n_splits (*X=None, y=None, groups=None*)

split (*X, y, groups=None*)

NoSplit

class mastml.legos.data_splitters.**NoSplit**

Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin

Class to just train the model on the training data and test it on that same data. Sometimes referred to as a “Full fit” or a “Single fit”, equivalent to just plotting y vs. x.

Args: None (only object instance)

Methods: get_n_splits: method to calculate the number of splits to perform

Args: None

Returns: (int), always 1 as only a single split is performed

split: method to perform split into train indices and test indices

Args: X: (numpy array), array of X features

Returns: (numpy array), array of train and test indices (all data used as train and test for NoSplit)

Methods Summary

```
get_n_splits([X, y, groups])
split(X, y[, groups])
```

Methods Documentation

get_n_splits (*X=None, y=None, groups=None*)

split (*X, y, groups=None*)

SplittersUnion

class mastml.legos.data_splitters.**SplittersUnion** (*splitters*)

Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin

Class to take the union of two separate splitting routines, so that many splitting routines can be performed at once

Args: splitters: (list), a list of scikit-learn splitter objects

Methods: get_n_splits: method to calculate the number of splits to perform across all splitters

Args: X: (numpy array), array of X features y: (numpy array), array of y data groups: (numpy array), array of group labels

Returns: (int), number of total splits to be conducted

`split`: method to perform split into train indices and test indices

Args: `X`: (numpy array), array of X features `y`: (numpy array), array of y data groups: (numpy array), array of group labels

Returns: (numpy array), array of train and test indices

Methods Summary

`get_n_splits(X, y[, groups])`

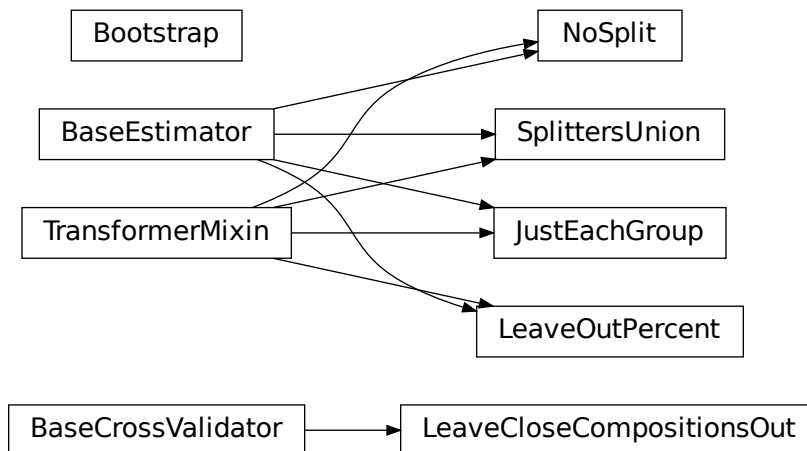
`split(X, y[, groups])`

Methods Documentation

get_n_splits (`X`, `y`, `groups=None`)

split (`X`, `y`, `groups=None`)

13.1.2 Class Inheritance Diagram



14.1 mastml.utils Module

The utils module contains a collection of miscellaneous methods and error handling used throughout MAST-ML

14.1.1 Functions

<code>activate_logging(savepath, paths[, ...])</code>	Method to create MAST-ML logger file
<code>ceil</code>	Return the ceiling of x as an Integral.
<code>floor</code>	Return the floor of x as an Integral.
<code>join(a, *p)</code>	Join two or more pathname components, inserting '/' as needed.
<code>log(x, [base=math.e])</code>	Return the logarithm of x to the given base.
<code>log_header(paths, log)</code>	Method to create header for MAST-ML logger
<code>nice_range(lower, upper)</code>	Method to create a range of values, including the specified start and end points, with nicely spaced intervals
<code>verboselize_logger(log, verbosity)</code>	

activate_logging

```

mastml.utils.activate_logging(savepath, paths, logger_name='mastml', to_screen=True,
                             to_file=True, verbosity=0)

```

Method to create MAST-ML logger file

Args:

savepath: (str), string specifying the save path

paths: (list), list containing strings of path locations for config file, data file, and results folder

logger_name: (str), name of logger file

to_screen: (bool), whether or not to write the log contents to the screen during a run

to_file: (bool), whether or not to write the log contents to a file in the savepath

verbosity: (int), controls the amount of output produced in the logger. Accepted values:

- 0 shows everything
- 1 hides debug
- 2 hides info (so no stdout except print)
- 3 hides warning
- 4 hides error
- 5 hides all output

Returns:

None

log_header

`mastml.utils.log_header(paths, log)`

Method to create header for MAST-ML logger

Args:

paths: (list), list containing strings of path locations for config file, data file, and results folder

log: (logging object), a python log

Returns:

None

nice_range

`mastml.utils.nice_range(lower, upper)`

Method to create a range of values, including the specified start and end points, with nicely spaced intervals

Args:

lower: (float or int), lower bound of range to create

upper: (float or int), upper bound of range to create

Returns:

(list), list of numerical values in established range

verboselize_logger

`mastml.utils.verboselize_logger(log, verbosity)`

14.1.2 Classes

<code>BetweenFilter(min_level, max_level)</code>	Class to aid in handling logger display levels
<code>ConfError</code>	Class representing error in input configuration file

Continued on next page

Table 2 – continued from previous page

<i>FileNotFoundError</i>	Class representing error raised when a needed file cannot be found
<i>FileTypeError</i>	Class representing error raised when an improper file extension is used
<i>InvalidConfParameters</i>	Class representing error raised when you have invalid input configuration file parameters
<i>InvalidConfSection</i>	Class representing error raised when an invalid section name is present in the input configuration file
<i>InvalidConfSubSection</i>	Class representing error raised when an invalid subsection name is present in the input configuration file
<i>InvalidModel</i>	Class representing error when model does not exist
<i>InvalidValue</i>	Class representing error raised when an invalid value has been used
<i>MastError</i>	Base class for MAST-ML specific errors that should be shown to the user
<i>MissingColumnError</i>	Class representing error raised when your csv doesn't have the specified column
defaultdict	defaultdict(default_factory[, ...]) -> dict with default factory

BetweenFilter

class mastml.utils.**BetweenFilter** (*min_level*, *max_level*)

Bases: object

Class to aid in handling logger display levels

Args:

min_level: (int), minimum verbosity level

max_level: (int), maximum verbosity level

Methods:

filter: Method to return logging level of logging.logRecord object

Args:

logRecord: (python logging.logRecord object)

Returns:

(int) logging level of logging.logRecord object, which is between the min and max provided levels

Methods Summary

filter(logRecord)

Methods Documentation

filter (*logRecord*)

ConfError

exception `mastml.utils.ConfError`
Class representing error in input configuration file

FileNotFoundError

exception `mastml.utils.FileNotFoundError`
Class representing error raised when a needed file cannot be found

FileTypeError

exception `mastml.utils.FileTypeError`
Class representing error raised when an improper file extension is used

InvalidConfParameters

exception `mastml.utils.InvalidConfParameters`
Class representing error raised when you have invalid input configuration file parameters

InvalidConfSection

exception `mastml.utils.InvalidConfSection`
Class representing error raised when an invalid section name is present in the input configuration file

InvalidConfSubSection

exception `mastml.utils.InvalidConfSubSection`
Class representing error raised when an invalid subsection name is present in the input configuration file

InvalidModel

exception `mastml.utils.InvalidModel`
Class representing error when model does not exist

InvalidValue

exception `mastml.utils.InvalidValue`
Class representing error raised when an invalid value has been used

MastError

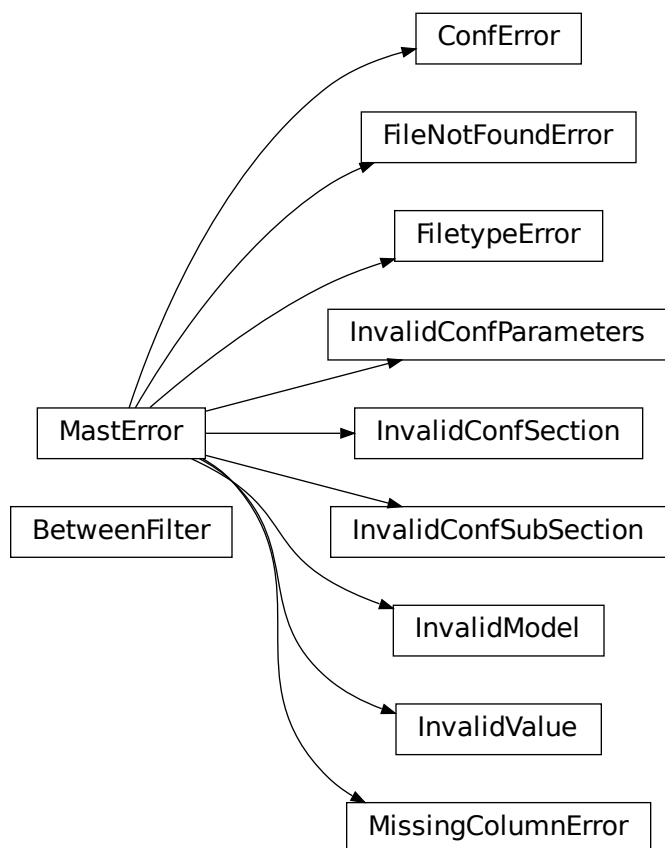
exception `mastml.utils.MastError`
Base class for MAST-ML specific errors that should be shown to the user

MissingColumnError

exception `mastml.utils.MissingColumnError`

Class representing error raised when your csv doesn't have the specified column

14.1.3 Class Inheritance Diagram



15.1 mastml.mastml_driver Module

Main MAST-ML module responsible for executing the workflow of a MAST-ML run

15.1.1 Functions

<code>check_paths(conf_path, data_path, outdir)</code>	This method is responsible for error handling of the user-specified paths for the configuration file, data file, and output directory.
<code>clone(estimator, *[, safe])</code>	Constructs a new unfitted estimator with the same parameters.
<code>deepcopy(x[, memo, _nil])</code>	Deep copy operation on arbitrary Python objects.
<code>get_commandline_args()</code>	This method is responsible for parsing and checking the command-line execution of MAST-ML inputted by the user.
<code>join(a, *p)</code>	Join two or more pathname components, inserting '/' as needed.
<code>main(conf_path, data_path[, outdir, verbosity])</code>	This method is responsible for setting up the initial stage of the MAST-ML run, such as parsing input directories to designate where data will be imported and results saved to, as well as creation of the MAST-ML run log.
<code>make_scorer(score_func, *[, ...])</code>	Make a scorer from a performance metric or loss function.
<code>mastml_run(conf_path, data_path, outdir)</code>	This method is responsible for conducting the main MAST-ML run workflow
<code>reduce(function, sequence[, initial])</code>	Apply a function of two arguments cumulatively to the items of a sequence, from left to right, so as to reduce the sequence to a single value.

check_paths

`mastml.mastml_driver.check_paths (conf_path, data_path, outdir)`

This method is responsible for error handling of the user-specified paths for the configuration file, data file, and output directory.

Args:

`conf_path`: (str), the path supplied by the user which contains the input configuration file

`data_path`: (str), the path supplied by the user which contains the input data file (as CSV or XLSX)

`outdir`: (str), the path supplied by the user which determines where the output results are saved to

Returns:

`conf_path`: (str), the path supplied by the user which contains the input configuration file

`data_path`: (str), the path supplied by the user which contains the input data file (as CSV or XLSX)

`outdir`: (str), the path supplied by the user which determines where the output results are saved to

get_commandline_args

`mastml.mastml_driver.get_commandline_args ()`

This method is responsible for parsing and checking the command-line execution of MAST-ML inputted by the user.

Args:

None

Returns:

(str), the path supplied by the user which contains the input configuration file

(str), the path supplied by the user which contains the input data file (as CSV or XLSX)

(str), the path supplied by the user which determines where the output results are saved to

`verbosity`: (int), the verbosity level of the MAST-ML log, which determines the amount of information written to the log.

main

`mastml.mastml_driver.main (conf_path, data_path, outdir='/home/docs/checkouts/readthedocs.org/user_builds/mastmldocs/c', verbosity=0)`

This method is responsible for setting up the initial stage of the MAST-ML run, such as parsing input directories to designate where data will be imported and results saved to, as well as creation of the MAST-ML run log.

Args:

`conf_path`: (str), the path supplied by the user which contains the input configuration file

`data_path`: (str), the path supplied by the user which contains the input data file (as CSV or XLSX)

`outdir`: (str), the path supplied by the user which determines where the output results are saved to

`verbosity`: (int), the verbosity level of the MAST-ML log, which determines the amount of information written to the log.

Returns:

outdir: (str), the path supplied by the user which determines where the output results are saved to (needed by other calls in MAST-ML)

mastml_run

`mastml.mastml_driver.mastml_run` (*conf_path*, *data_path*, *outdir*)

This method is responsible for conducting the main MAST-ML run workflow

Args:

conf_path: (str), the path supplied by the user which contains the input configuration file

data_path: (str), the path supplied by the user which contains the input data file (as CSV or XLSX)

outdir: (str), the path supplied by the user which determines where the output results are saved to

Returns:

None

16.1 mastml.plot_helper Module

This module contains a collection of functions which make plots (saved as png files) using matplotlib, generated from some model fits and cross-validation evaluation within a MAST-ML run.

This module also contains a method to create python notebooks containing plotted data and the relevant source code from this module, to enable the user to make their own modifications to the created plots in a straightforward way (useful for tweaking plots for a presentation or publication).

16.1.1 Functions

<code>auc(x, y)</code>	Compute Area Under the Curve (AUC) using the trapezoidal rule.
<code>ceil</code>	Return the ceiling of x as an Integral.
<code>confusion_matrix(y_true, y_pred, *[, ...])</code>	Compute confusion matrix to evaluate the accuracy of a classification.
<code>figaspect(arg)</code>	Calculate the width and height for a figure with a specified aspect ratio.
<code>floor</code>	Return the floor of x as an Integral.
<code>get_divisor(high, low)</code>	Method to obtain a sensible divisor based on range of two values
<code>get_histogram_bins(y_df)</code>	Method to obtain the number of bins to use when plotting a histogram
<code>ipynb_maker(plot_func)</code>	This method creates Jupyter Notebooks so user can modify and regenerate the plots produced by MAST-ML.
<code>join(a, *p)</code>	Join two or more pathname components, inserting '/' as needed.
<code>log(x, [base=math.e])</code>	Return the logarithm of x to the given base.

Continued on next page

Table 1 – continued from previous page

<code>make_axes_locatable(axes)</code>	
<code>make_axis_same(ax, max1, min1)</code>	Method to make the x and y ticks for each axis the same.
<code>make_error_plots(run, path, ...[, groups])</code>	
<code>make_fig_ax([aspect_ratio, x_align, left])</code>	Method to make matplotlib figure and axes objects.
<code>make_fig_ax_square([aspect, aspect_ratio])</code>	Method to make square shaped matplotlib figure and axes objects.
<code>make_train_test_plots(run, path, ...[, groups])</code>	General plotting method used to execute sequence of specific plots of train-test data analysis
<code>mark_inset(parent_axes, inset_axes, loc1, ...)</code>	Draw a box to mark the location of an area represented by an inset axes.
<code>nice_mean(ls)</code>	Method to return mean of a list or equivalent array with NaN values
<code>nice_names()</code>	
<code>nice_range(lower, upper)</code>	Method to create a range of values, including the specified start and end points, with nicely spaced intervals
<code>nice_std(ls)</code>	Method to return standard deviation of a list or equivalent array with NaN values
<code>parse_error_data(dataset_stdev, ...)</code>	
<code>plot_1d_heatmap(xs, heats, savepath[, ...])</code>	Method to plot a heatmap for values of a single variable; used for plotting GridSearch results in hyperparameter optimization.
<code>plot_2d_heatmap(xs, ys, heats, savepath[, ...])</code>	Method to plot a heatmap for values of two variables; used for plotting GridSearch results in hyperparameter optimization.
<code>plot_3d_heatmap(xs, ys, zs, heats, savepath)</code>	Method to plot a heatmap for values of three variables; used for plotting GridSearch results in hyperparameter optimization.
<code>plot_average_cumulative_normalized_error(y_true, ...[, ...])</code>	Method to plot the cumulative normalized residual errors of a model prediction
<code>plot_average_normalized_error(y_true, ...[, ...])</code>	Method to plot the normalized residual errors of a model prediction
<code>plot_best_worst_per_point(y_true, ...[, ...])</code>	Method to create a parity plot (predicted vs.
<code>plot_best_worst_split(y_true, best_run, ...)</code>	Method to create a parity plot (predicted vs.
<code>plot_confusion_matrix(y_true, y_pred, ...[, ...])</code>	Method used to generate a confusion matrix for a classification run.
<code>plot_cumulative_normalized_error(y_true, ...)</code>	Method to plot the cumulative normalized residual errors of a model prediction
<code>plot_keras_history(model_history, savepath, ...)</code>	
<code>plot_learning_curve(train_sizes, train_mean, ...)</code>	Method used to plot both data and feature learning curves
<code>plot_learning_curve_convergence(train_sizes, ...)</code>	Method used to plot both the convergence of data and feature learning curves as a function of amount of data or features
<code>plot_metric_vs_group(metric, groups, stats, ...)</code>	Method to plot the value of a particular calculated metric (e.g.
<code>plot_metric_vs_group_size(metric, groups, ...)</code>	Method to plot the value of a particular calculated metric (e.g.
<code>plot_normalized_error(y_true, y_pred, ...[, ...])</code>	Method to plot the normalized residual errors of a model prediction

Continued on next page

Table 1 – continued from previous page

<code>plot_precision_recall_curve(y_true, y_pred, ...)</code>	Method to calculate and plot the precision-recall curve for classification model results
<code>plot_predicted_vs_true(train_quad, ...)</code>	Method to create a parity plot (predicted vs.
<code>plot_predicted_vs_trueBars(y_true, ..., ...)</code>	Method to calculate parity plot (predicted vs.
<code>plot_real_vs_predicted_error(y_true, ...)</code>	
<code>plot_residuals_histogram(y_true, y_pred, ...)</code>	Method to calculate and plot the histogram of residuals from regression model
<code>plot_roc_curve(y_true, y_pred, savepath)</code>	Method to calculate and plot the receiver-operator characteristic curve for classification model results
<code>plot_scatter(x, y, savepath[, groups, ...])</code>	Method to create a general scatter plot
<code>plot_stats(fig, stats[, x_align, y_align, ...])</code>	Method that prints stats onto the plot.
<code>plot_target_histogram(y_df, savepath[, ...])</code>	Method to plot the histogram of true y values
<code>precision_recall_curve(y_true, probas_pred, *)</code>	Compute precision-recall pairs for different probability thresholds.
<code>prediction_intervals(model, X, ...)</code>	Method to calculate prediction intervals when using Random Forest and Gaussian Process regression models.
<code>r2_score(y_true, y_pred, *[, sample_weight, ...])</code>	R ² (coefficient of determination) regression score function.
<code>recursive_max(arr)</code>	Method to recursively find the max value of an array of iterables.
<code>recursive_max_and_min(arr)</code>	Method to recursively return max and min of values or iterables in array
<code>recursive_min(arr)</code>	Method to recursively find the min value of an array of iterables.
<code>roc_curve(y_true, y_score, *[, pos_label, ...])</code>	Compute Receiver operating characteristic (ROC).
<code>round_down(num, divisor)</code>	Method to return a rounded down number
<code>round_up(num, divisor)</code>	Method to return a rounded up number
<code>rounder(delta)</code>	Method to obtain number of decimal places to report on plots
<code>stat_to_string(name, value, nice_names)</code>	Method that converts a metric object into a string for displaying on a plot
<code>trim_array(arr_list)</code>	Method used to trim a set of arrays to make all arrays the same shape
<code>wraps(wrapped[, assigned, updated])</code>	Decorator factory to apply update_wrapper() to a wrapper function
<code>zoomed_inset_axes(parent_axes, zoom[, loc, ...])</code>	Create an anchored inset axes by scaling a parent axes.

get_divisor

`mastml.plot_helper.get_divisor (high, low)`

Method to obtain a sensible divisor based on range of two values

Args:

high: (float), a max data value

low: (float), a min data value

Returns:

divisor: (float), a number used to make sensible axis ticks

get_histogram_bins

`mastml.plot_helper.get_histogram_bins(y_df)`

Method to obtain the number of bins to use when plotting a histogram

Args:

`y_df`: (pandas Series or numpy array), array of y data used to construct histogram

Returns:

`num_bins`: (int), the number of bins to use when plotting a histogram

ipy nb_maker

`mastml.plot_helper.ipynb_maker(plot_func)`

This method creates Jupyter Notebooks so user can modify and regenerate the plots produced by MAST-ML.

Args:

`plot_func`: (plot_helper method), a plotting method contained in `plot_helper.py` which contains the `@ipynb_maker` decorator

Returns:

(plot_helper method), the same `plot_func` as used as input, but after having written the Jupyter notebook with source code to create plot

make_axis_same

`mastml.plot_helper.make_axis_same(ax, max1, min1)`

Method to make the x and y ticks for each axis the same. Useful for parity plots

Args:

`ax`: (matplotlib axis object), a matplotlib axes object

`max1`: (float), the maximum value of a particular axis

`min1`: (float), the minimum value of a particular axis

Returns:

None

make_error_plots

`mastml.plot_helper.make_error_plots(run, path, is_classification, label, model, train_X, test_X, rf_error_method, rf_error_percentile, is_validation, validation_column_name, validation_X, groups=None)`

make_fig_ax

`mastml.plot_helper.make_fig_ax(aspect_ratio=0.5, x_align=0.65, left=0.1)`

Method to make matplotlib figure and axes objects. Using Object Oriented interface from https://matplotlib.org/gallery/api/agg_oo_sgskip.html

Args:

aspect_ratio: (float), aspect ratio for figure and axes creation

x_align: (float), x position to draw edge of figure. Needed so can display stats alongside plot

left: (float), the leftmost position to draw edge of figure

Returns:

fig: (matplotlib fig object), a matplotlib figure object with the specified aspect ratio

ax: (matplotlib ax object), a matplotlib axes object with the specified aspect ratio

make_fig_ax_square

`mastml.plot_helper.make_fig_ax_square` (*aspect='equal', aspect_ratio=1*)

Method to make square shaped matplotlib figure and axes objects. Using Object Oriented interface from

https://matplotlib.org/gallery/api/agg_oo_sgskip.html

Args:

aspect: (str), 'equal' denotes x and y aspect will be equal (i.e. square)

aspect_ratio: (float), aspect ratio for figure and axes creation

Returns:

fig: (matplotlib fig object), a matplotlib figure object with the specified aspect ratio

ax: (matplotlib ax object), a matplotlib axes object with the specified aspect ratio

make_train_test_plots

`mastml.plot_helper.make_train_test_plots` (*run, path, is_classification, label, model, train_X, test_X, groups=None*)

General plotting method used to execute sequence of specific plots of train-test data analysis

Args:

run: (dict), a particular split_result from masml_driver

path: (str), path to save the generated plots and analysis of split_result designated in 'run'

is_classification: (bool), whether or not the analysis is a classification task

label: (str), name of the y data variable being fit

model: (scikit-learn model object), a scikit-learn model/estimator

train_X: (numpy array), array of X features used in training

test_X: (numpy array), array of X features used in testing

groups: (numpy array), array of group names

Returns:

None

nice_mean

`mastml.plot_helper.nice_mean(ls)`

Method to return mean of a list or equivalent array with NaN values

Args:

ls: (list), list of values

Returns:

(numpy array), array containing mean of list of values or NaN if list has no values

nice_names

`mastml.plot_helper.nice_names()`

nice_range

`mastml.plot_helper.nice_range(lower, upper)`

Method to create a range of values, including the specified start and end points, with nicely spaced intervals

Args:

lower: (float or int), lower bound of range to create

upper: (float or int), upper bound of range to create

Returns:

(list), list of numerical values in established range

nice_std

`mastml.plot_helper.nice_std(ls)`

Method to return standard deviation of a list or equivalent array with NaN values

Args:

ls: (list), list of values

Returns:

(numpy array), array containing standard deviation of list of values or NaN if list has no values

parse_error_data

`mastml.plot_helper.parse_error_data(dataset_stdev, path_to_test, data_test_type)`

plot_1d_heatmap

`mastml.plot_helper.plot_1d_heatmap(xs, heats, savepath, xlabel='x', heatlabel='heats')`

Method to plot a heatmap for values of a single variable; used for plotting GridSearch results in hyperparameter optimization.

Args:

xs: (numpy array), array of first variable values to plot heatmap against
 heats: (numpy array), array of heat values to plot
 savepath: (str), path to save the 1D heatmap to
 xlabel: (str), the x-axis label
 heatlabel: (str), the heat value axis label

plot_2d_heatmap

`mastml.plot_helper.plot_2d_heatmap(xs, ys, heats, savepath, xlabel='x', ylabel='y', heatlabel='heat')`

Method to plot a heatmap for values of two variables; used for plotting GridSearch results in hyperparameter optimization.

Args:

xs: (numpy array), array of first variable values to plot heatmap against
 ys: (numpy array), array of second variable values to plot heatmap against
 heats: (numpy array), array of heat values to plot
 savepath: (str), path to save the 2D heatmap to
 xlabel: (str), the x-axis label
 ylabel: (str), the y-axis label
 heatlabel: (str), the heat value axis label

plot_3d_heatmap

`mastml.plot_helper.plot_3d_heatmap(xs, ys, zs, heats, savepath, xlabel='x', ylabel='y', zlabel='z', heatlabel='heat')`

Method to plot a heatmap for values of three variables; used for plotting GridSearch results in hyperparameter optimization.

Args:

xs: (numpy array), array of first variable values to plot heatmap against
 ys: (numpy array), array of second variable values to plot heatmap against
 zs: (numpy array), array of third variable values to plot heatmap against
 heats: (numpy array), array of heat values to plot
 savepath: (str), path to save the 2D heatmap to
 xlabel: (str), the x-axis label
 ylabel: (str), the y-axis label
 zlabel: (str), the z-axis label
 heatlabel: (str), the heat value axis label

plot_average_cumulative_normalized_error

```
mastml.plot_helper.plot_average_cumulative_normalized_error(y_true, y_pred,
                                                            savepath,
                                                            has_model_errors,
                                                            err_avg=None)
```

Method to plot the cumulative normalized residual errors of a model prediction

Args:

- y_true: (numpy array), array containing the true y data values
- y_pred: (numpy array), array containing the predicted y data values
- savepath: (str), path to save the plotted cumulative normalized error plot
- model: (scikit-learn model/estimator object), a scikit-learn model object
- X: (numpy array), array of X features
- avg_stats: (dict), dict of calculated average metrics over all CV splits

Returns:

None

plot_average_normalized_error

```
mastml.plot_helper.plot_average_normalized_error(y_true, y_pred, savepath,
                                                  has_model_errors, err_avg=None)
```

Method to plot the normalized residual errors of a model prediction

Args:

- y_true: (numpy array), array containing the true y data values
- y_pred: (numpy array), array containing the predicted y data values
- savepath: (str), path to save the plotted normalized error plot
- model: (scikit-learn model/estimator object), a scikit-learn model object
- X: (numpy array), array of X features
- avg_stats: (dict), dict of calculated average metrics over all CV splits

Returns:

None

plot_best_worst_per_point

```
mastml.plot_helper.plot_best_worst_per_point(y_true, y_pred_list, savepath, metrics_dict,
                                              avg_stats, title='best worst per point', label='target_value')
```

Method to create a parity plot (predicted vs. true values) of the set of best and worst CV scores for each individual data point.

Args:

y_true: (numpy array), array of true y data
y_pred_list: (list), list of numpy arrays containing predicted y data for each CV split
savepath: (str), path to save plots to
metrics_dict: (dict), dict of scikit-learn metric objects to calculate score of predicted vs. true values
avg_stats: (dict), dict of calculated average metrics over all CV splits
title: (str), title of the best_worst_per_point plot
label: (str), label used for axis labeling

Returns:

None

plot_best_worst_split

`mastml.plot_helper.plot_best_worst_split` (*y_true, best_run, worst_run, savepath, title='Best Worst Overlay', label='target_value'*)

Method to create a parity plot (predicted vs. true values) of just the best scoring and worst scoring CV splits

Args:

y_true: (numpy array), array of true y data
best_run: (dict), the best scoring split_result from mastml_driver
worst_run: (dict), the worst scoring split_result from mastml_driver
savepath: (str), path to save plots to
title: (str), title of the best_worst_split plot
label: (str), label used for axis labeling

Returns:

None

plot_confusion_matrix

`mastml.plot_helper.plot_confusion_matrix` (*y_true, y_pred, savepath, stats, normalize=False, title='Confusion matrix', cmap=<matplotlib.colors.LinearSegmentedColormap object>*)

Method used to generate a confusion matrix for a classification run. Additional information can be found at: http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

Args:

y_true: (numpy array), array containing the true y data values
y_pred: (numpy array), array containing the predicted y data values
savepath: (str), path to save the plotted confusion matrix
stats: (dict), dict of training or testing statistics for a particular run
normalize: (bool), whether or not to normalize data output as truncated float vs. double
title: (str), title of the confusion matrix plot

cmap: (matplotlib colormap), the color map to use for confusion matrix plotting

Returns:

None

plot_cumulative_normalized_error

```
mastml.plot_helper.plot_cumulative_normalized_error(y_true, y_pred, savepath,
                                                    model, rf_error_method,
                                                    rf_error_percentile, X=None,
                                                    Xtrain=None, Xtest=None)
```

Method to plot the cumulative normalized residual errors of a model prediction

Args:

y_true: (numpy array), array containing the true y data values

y_pred: (numpy array), array containing the predicted y data values

savepath: (str), path to save the plotted cumulative normalized error plot

model: (scikit-learn model/estimator object), a scikit-learn model object

X: (numpy array), array of X features

avg_stats: (dict), dict of calculated average metrics over all CV splits

Returns:

None

plot_keras_history

```
mastml.plot_helper.plot_keras_history(model_history, savepath, plot_type)
```

plot_learning_curve

```
mastml.plot_helper.plot_learning_curve(train_sizes, train_mean, test_mean, train_stdev,
                                       test_stdev, score_name, learning_curve_type,
                                       savepath='data_learning_curve')
```

Method used to plot both data and feature learning curves

Args:

train_sizes: (numpy array), array of x-axis values, such as fraction of data used or number of features

train_mean: (numpy array), array of training data mean values, averaged over some type/number of CV splits

test_mean: (numpy array), array of test data mean values, averaged over some type/number of CV splits

train_stdev: (numpy array), array of training data standard deviation values, from some type/number of CV splits

test_stdev: (numpy array), array of test data standard deviation values, from some type/number of CV splits

score_name: (str), type of score metric for learning curve plotting; used in y-axis label

learning_curve_type: (str), type of learning curve employed: 'sample_learning_curve' or 'feature_learning_curve'

savepath: (str), path to save the plotted learning curve to

Returns:

None

plot_learning_curve_convergence

mastml.plot_helper.plot_learning_curve_convergence(*train_sizes*, *test_mean*,
score_name, *learning_curve_type*,
savepath)

Method used to plot both the convergence of data and feature learning curves as a function of amount of data or features

used, respectively.

Args:

train_sizes: (numpy array), array of x-axis values, such as fraction of data used or number of features

test_mean: (numpy array), array of test data mean values, averaged over some type/number of CV splits

score_name: (str), type of score metric for learning curve plotting; used in y-axis label

learning_curve_type: (str), type of learning curve employed: 'sample_learning_curve' or 'feature_learning_curve'

savepath: (str), path to save the plotted convergence learning curve to

Returns:

None

plot_metric_vs_group

mastml.plot_helper.plot_metric_vs_group(*metric*, *groups*, *stats*, *avg_stats*, *savepath*)

Method to plot the value of a particular calculated metric (e.g. RMSE, R², etc) for each data group

Args:

metric: (str), name of a calculation metric

groups: (numpy array), array of group names

stats: (dict), dict of training or testing statistics for a particular run

avg_stats: (dict), dict of calculated average metrics over all CV splits

savepath: (str), path to save plots to

Returns:

None

plot_metric_vs_group_size

`mastml.plot_helper.plot_metric_vs_group_size` (*metric, groups, stats, avg_stats, savepath*)

Method to plot the value of a particular calculated metric (e.g. RMSE, R^2 , etc) as a function of the size of each group.

Args:

`metric`: (str), name of a calculation metric

`groups`: (numpy array), array of group names

`stats`: (dict), dict of training or testing statistics for a particular run

`avg_stats`: (dict), dict of calculated average metrics over all CV splits

`savepath`: (str), path to save plots to

Returns:

None

plot_normalized_error

`mastml.plot_helper.plot_normalized_error` (*y_true, y_pred, savepath, model, rf_error_method, rf_error_percentile, X=None, Xtrain=None, Xtest=None*)

Method to plot the normalized residual errors of a model prediction

Args:

`y_true`: (numpy array), array containing the true y data values

`y_pred`: (numpy array), array containing the predicted y data values

`savepath`: (str), path to save the plotted normalized error plot

`model`: (scikit-learn model/estimator object), a scikit-learn model object

`X`: (numpy array), array of X features

`avg_stats`: (dict), dict of calculated average metrics over all CV splits

Returns:

None

plot_precision_recall_curve

`mastml.plot_helper.plot_precision_recall_curve` (*y_true, y_pred, savepath*)

Method to calculate and plot the precision-recall curve for classification model results

Args:

`y_true`: (numpy array), array of true y data values

`y_pred`: (numpy array), array of predicted y data values

`savepath`: (str), path to save the plotted precision-recall curve

Returns:

None

plot_predicted_vs_true

`mastml.plot_helper.plot_predicted_vs_true` (*train_quad, test_quad, outdir, label*)

Method to create a parity plot (predicted vs. true values)

Args:

`train_quad`: (tuple), tuple containing 4 numpy arrays: true training y data, predicted training y data, training metric data, and groups used in training

`test_quad`: (tuple), tuple containing 4 numpy arrays: true test y data, predicted test y data, testing metric data, and groups used in testing

`outdir`: (str), path to save plots to

`label`: (str), label used for axis labeling

Returns:

None

plot_predicted_vs_trueBars

`mastml.plot_helper.plot_predicted_vs_trueBars` (*y_true, y_pred_list, avg_stats, savepath, title='best worst with bars', label='target_value', groups=None*)

Method to calculate parity plot (predicted vs. true) of average predictions, averaged over all CV splits, with error

bars on each point corresponding to the standard deviation of the predicted values over all CV splits.

Args:

`y_true`: (numpy array), array of true y data

`y_pred_list`: (list), list of numpy arrays containing predicted y data for each CV split

`avg_stats`: (dict), dict of calculated average metrics over all CV splits

`savepath`: (str), path to save plots to

`title`: (str), title of the best_worst_per_point plot

`label`: (str), label used for axis labeling

Returns:

None

plot_real_vs_predicted_error

`mastml.plot_helper.plot_real_vs_predicted_error` (*y_true, savepath, model, data_test_type*)

plot_residuals_histogram

`mastml.plot_helper.plot_residuals_histogram` (*y_true, y_pred, savepath, stats, title='residuals histogram', label='residuals')*

Method to calculate and plot the histogram of residuals from regression model

Args:

`y_true`: (numpy array), array of true y data values
`y_pred`: (numpy array), array of predicted y data values
`savepath`: (str), path to save the plotted precision-recall curve
`stats`: (dict), dict of training or testing statistics for a particular run
`title`: (str), title of residuals histogram
`label`: (str), label used for axis labeling

Returns:

None

plot_roc_curve

`mastml.plot_helper.plot_roc_curve(y_true, y_pred, savepath)`

Method to calculate and plot the receiver-operator characteristic curve for classification model results

Args:

`y_true`: (numpy array), array of true y data values
`y_pred`: (numpy array), array of predicted y data values
`savepath`: (str), path to save the plotted ROC curve

Returns:

None

plot_scatter

`mastml.plot_helper.plot_scatter(x, y, savepath, groups=None, xlabel='x', label='target data')`

Method to create a general scatter plot

Args:

`x`: (numpy array), array of x data
`y`: (numpy array), array of y data
`savepath`: (str), path to save plots to
`groups`: (list), list of group labels
`xlabel`: (str), label used for x-axis labeling
`label`: (str), label used for y-axis labeling

Returns:

None

plot_stats

`mastml.plot_helper.plot_stats(fig, stats, x_align=0.65, y_align=0.9, font_dict={}, fontsize=14)`

Method that prints stats onto the plot. Goes off screen if they are too long or too many in number.

Args:

fig: (matplotlib figure object), a matplotlib figure object
stats: (dict), dict of statistics to be included with a plot
x_align: (float), float denoting x position of where to align display of stats on a plot
y_align: (float), float denoting y position of where to align display of stats on a plot
font_dict: (dict), dict of matplotlib font options to alter display of stats on plot
fontsize: (int), the fontsize of stats to display on plot

Returns:

None

plot_target_histogram

`mastml.plot_helper.plot_target_histogram(y_df, savepath, title='target histogram', label='target values')`

Method to plot the histogram of true y values

Args:

y_df: (pandas dataframe), dataframe of true y data values
savepath: (str), path to save the plotted precision-recall curve
title: (str), title of residuals histogram
label: (str), label used for axis labeling

Returns:

None

prediction_intervals

`mastml.plot_helper.prediction_intervals(model, X, rf_error_method, rf_error_percentile, Xtrain, Xtest)`

Method to calculate prediction intervals when using Random Forest and Gaussian Process regression models.

Prediction intervals for random forest adapted from <https://blog.datadive.net/prediction-intervals-for-random-forests/>

Args:

model: (scikit-learn model/estimator object), a scikit-learn model object
X: (numpy array), array of X features
method: (str), type of error bar to formulate (e.g. “stddev” is standard deviation of predicted errors, “confint” is error bar as confidence interval)
percentile: (int), percentile for which to form error bars

Returns:

err_up: (list), list of upper bounds of error bars for each data point
err_down: (list), list of lower bounds of error bars for each data point

recursive_max

`mastml.plot_helper.recursive_max(arr)`

Method to recursively find the max value of an array of iterables.

Credit: <https://www.linkedin.com/pulse/ask-recursion-during-coding-interviews-identify-good-talent-veteanu/>

Args:

arr: (numpy array), an array of values or iterables

Returns:

(float), max value in arr

recursive_max_and_min

`mastml.plot_helper.recursive_max_and_min(arr)`

Method to recursively return max and min of values or iterables in array

Args:

arr: (numpy array), an array of values or iterables

Returns:

(tuple), tuple containing max and min of arr

recursive_min

`mastml.plot_helper.recursive_min(arr)`

Method to recursively find the min value of an array of iterables.

Credit: <https://www.linkedin.com/pulse/ask-recursion-during-coding-interviews-identify-good-talent-veteanu/>

Args:

arr: (numpy array), an array of values or iterables

Returns:

(float), min value in arr

round_down

`mastml.plot_helper.round_down(num, divisor)`

Method to return a rounded down number

Args:

num: (float), a number to round down

divisor: (int), divisor to denote how to round down

Returns:

(float), the rounded-down number

round_up

`mastml.plot_helper.round_up(num, divisor)`

Method to return a rounded up number

Args:

num: (float), a number to round up

divisor: (int), divisor to denote how to round up

Returns:

(float), the rounded-up number

rounder

`mastml.plot_helper.rounder(delta)`

Method to obtain number of decimal places to report on plots

Args:

delta: (float), a float representing the change in two y values on a plot, used to obtain the plot axis spacing size

Return:

(int), an integer denoting the number of decimal places to use

stat_to_string

`mastml.plot_helper.stat_to_string(name, value, nice_names)`

Method that converts a metric object into a string for displaying on a plot

Args:

name: (str), long name of a stat metric or quantity

value: (float), value of the metric or quantity

Return:

(str), a string of the metric name, adjusted to look nicer for inclusion on a plot

trim_array

`mastml.plot_helper.trim_array(arr_list)`

Method used to trim a set of arrays to make all arrays the same shape

Args:

arr_list: (list), list of numpy arrays, where arrays are different sizes

Returns:

arr_list: (), list of trimmed numpy arrays, where arrays are same size

17.1 mastml.html_helper Module

Module for generating an HTML file, called index.html, which contains an overview of the key data and plots from a MAST-ML run. Images of cross-validation parity plots, data histograms, data statistics, and links to the relevant files are all provided.

17.1.1 Functions

<code>attr(*args, **kwargs)</code>	Set attributes on the current active tag context
<code>get_current([default])</code>	get the current tag being used as a with context or decorated function.
<code>gmtime([seconds])</code>	tm_sec, tm_wday, tm_yday, tm_isdst)
<code>is_test_image(path)</code>	Method used to assess whether an image is for testing data
<code>is_train_image(path)</code>	Method used to assess whether an image is for training data
<code>join(a, *p)</code>	Join two or more pathname components, inserting '/' as needed.
<code>make_html(outdir)</code>	Method used to create the main index.html file
<code>make_image(src[, title])</code>	Method used to generate and show an image of a fixed width.
<code>make_link(href)</code>	Method used to generate a link to a particular file created from a MAST-ML run.
<code>relpath(path[, start])</code>	Return a relative version of a path
<code>show_combo(combo_dir, outdir)</code>	Method used to collect combinations of data analysis (e.g.

Continued on next page

Table 1 – continued from previous page

<code>simple_section(filepath, outdir)</code>	Method used to create a section name for a particular analysis combination that will be displayed in the index.html file.
<code>strftime(format[, tuple])</code>	Convert a time tuple to a string according to a format specification.

is_test_image

`mastml.html_helper.is_test_image(path)`

Method used to assess whether an image is for testing data

Args:

path: (str), source path of the image to be displayed

Returns:

(bool), True if path is an image (.png) and is for testing data (has ‘test’ in path)

is_train_image

`mastml.html_helper.is_train_image(path)`

Method used to assess whether an image is for training data

Args:

path: (str), source path of the image to be displayed

Returns:

(bool), True if path is an image (.png) and is for training data (has ‘train’ in path)

make_html

`mastml.html_helper.make_html(outdir)`

Method used to create the main index.html file

Args:

outdir: (str), user-specified output path which designates where all results of MAST-ML run are written

Returns:

None

make_image

`mastml.html_helper.make_image(src, title=None)`

Method used to generate and show an image of a fixed width. The image will be displayed in the appropriate section of the index.html file

Args:

src: (str), source path of the image to be displayed

title: (str), title for the image

Returns:

None

make_link

`mastml.html_helper.make_link(href)`

Method used to generate a link to a particular file created from a MAST-ML run. The link will be displayed next to the appropriate data or image in the index.html file

Args:

href: (str), filename to generate link for

Returns:

(dominate.tags.html_tag object), hyperlink to filename

show_combo

`mastml.html_helper.show_combo(combo_dir, outdir)`

Method used to collect combinations of data analysis (e.g. parity plots of train and test data in a CV split) and required file paths and display them in the output index.html file.

Args:

combo_dir: (str), path containing the relevant data to combine as output in the index.html file

outdir: (str), user-specified output path which designates where all results of MAST-ML run are written

Returns:

None

simple_section

`mastml.html_helper.simple_section(filepath, outdir)`

Method used to create a section name for a particular analysis combination that will be displayed in the index.html file.

Args:

filepath: (str), path containing the relevant data to combine as output in the index.html file

outdir: (str), user-specified output path which designates where all results of MAST-ML run are written

Returns:

None

Code Documentation: Feature Selectors

18.1 mastml.legos.feature_selectors Module

This module contains a collection of classes and methods for selecting features, and interfaces with scikit-learn feature selectors. More information on scikit-learn feature selectors is available at:

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_selection

18.1.1 Functions

<code>cov(m[, y, rowvar, bias, ddof, fweights, ...])</code>	Estimate a covariance matrix, given data and weights.
<code>dataframify_new_column_names(transform, name)</code>	Method which transforms output of scikit-learn feature selectors to dataframe, and adds column names
<code>dataframify_selector(transform)</code>	Method which transforms output of scikit-learn feature selectors from array to dataframe.
<code>fitify_just_use_values(fit)</code>	Method which enables a feature selector fit method to operate on dataframes
<code>pearsonr(x, y)</code>	Pearson correlation coefficient and p-value for testing non-correlation.
<code>root_mean_squared_error(y_true, y_pred)</code>	Method that calculates the root mean squared error (RMSE)
<code>wraps(wrapped[, assigned, updated])</code>	Decorator factory to apply <code>update_wrapper()</code> to a wrapper function

dataframify_new_column_names

`mastml.legos.feature_selectors.dataframify_new_column_names` (*transform, name*)
Method which transforms output of scikit-learn feature selectors to dataframe, and adds column names

Args:

`transform`: (function), a scikit-learn feature selector that has a transform method

name: (str), name of the feature selector

Returns:

new_transform: (function), an amended version of the transform method that returns a dataframe

dataframify_selector

mastml.legos.feature_selectors.**dataframify_selector** (*transform*)

Method which transforms output of scikit-learn feature selectors from array to dataframe. Enables preservation of column names.

Args:

transform: (function), a scikit-learn feature selector that has a transform method

Returns:

new_transform: (function), an amended version of the transform method that returns a dataframe

fitify_just_use_values

mastml.legos.feature_selectors.**fitify_just_use_values** (*fit*)

Method which enables a feature selector fit method to operate on dataframes

Args:

fit: (function), a scikit-learn feature selector object with a fit method

Returns:

new_fit: (function), an amended version of the fit method that uses dataframes as input

18.1.2 Classes

BaseEstimator	Base class for all estimators in scikit-learn.
<i>EnsembleModelFeatureSelector</i> (estimator, ...)	Class custom-written for MAST-ML to conduct selection of features with ensemble model feature importances
<i>MASTMLFeatureSelector</i> (estimator, ..., [...])	Class custom-written for MAST-ML to conduct forward selection of features with flexible model and cv scheme
PCA([n_components, copy, whiten, ...])	Principal component analysis (PCA).
<i>PearsonSelector</i> (threshold_between_features, ...)	Class custom-written for MAST-ML to conduct selection of features based on Pearson correlation coefficient between features and target.
SequentialFeatureSelector(estimator[, ...])	Sequential Feature Selection for Classification and Regression.
TransformerMixin	Mixin class for all transformers in scikit-learn.
constructor	alias of sklearn.feature_selection._variance_threshold.VarianceThreshold

EnsembleModelFeatureSelector

class mastml.legos.feature_selectors.**EnsembleModelFeatureSelector** (*estimator*,
k_features)

Bases: object

Class custom-written for MAST-ML to conduct selection of features with ensemble model feature importances

Args:

estimator: (scikit-learn model/estimator object), a scikit-learn model/estimator

k_features: (int), the number of features to select

Methods:

fit: performs feature selection

Args:

X: (dataframe), dataframe of X features

y: (dataframe), dataframe of y data

Returns:

None

transform: performs the transform to generate output of only selected features

Args:

X: (dataframe), dataframe of X features

Returns:

dataframe: (dataframe), dataframe of selected X features

Methods Summary

fit(*X*, *y*)

transform(*X*)

Methods Documentation

fit (*X*, *y=None*)

transform (*X*)

MASTMLFeatureSelector

class mastml.legos.feature_selectors.**MASTMLFeatureSelector** (*estimator*,
n_features_to_select,
cv, *manually_selected_features=[]*)

Bases: object

Class custom-written for MAST-ML to conduct forward selection of features with flexible model and cv scheme

Args:

estimator: (scikit-learn model/estimator object), a scikit-learn model/estimator

`n_features_to_select`: (int), the number of features to select

`cv`: (scikit-learn cross-validation object), a scikit-learn cross-validation object

`manually_selected_features`: (list), a list of features manually set by the user. The feature selector will first start from this list of features and sequentially add features until `n_features_to_select` is met.

Methods:

`fit`: performs feature selection

Args:

`X`: (dataframe), dataframe of X features

`y`: (dataframe), dataframe of y data

`Xgroups`: (dataframe), dataframe of group labels

Returns:

None

`transform`: performs the transform to generate output of only selected features

Args:

`X`: (dataframe), dataframe of X features

Returns:

dataframe: (dataframe), dataframe of selected X features

Methods Summary

`fit(X, y, savepath[, Xgroups])`

`transform(X)`

Methods Documentation

fit (*X*, *y*, *savepath*, *Xgroups*=None)

transform (*X*)

PearsonSelector

class `mastml.legos.feature_selectors.PearsonSelector` (*threshold_between_features*,
threshold_with_target, *remove_highly_correlated_features*,
k_features)

Bases: `object`

Class custom-written for MAST-ML to conduct selection of features based on Pearson correlation coefficient between features and target. Can also be used for dimensionality reduction by removing redundant features highly correlated with each other.

Args:

`threshold_between_features`: (float), the threshold to decide whether redundant features are removed. Should be a decimal value between 0 and 1. Only used if `remove_highly_correlated_features` is True

`threshold_with_target`: (float), the threshold to decide whether a given feature is sufficiently correlated with the target feature and thus kept as a selected feature. Should be a decimal value between 0 and 1.

`remove_highly_correlated_features`: (bool), whether to remove features highly correlated with each other

`k_features`: (int), the number of features to select

Methods:

`fit`: performs feature selection

Args:

`X`: (dataframe), dataframe of X features

`y`: (dataframe), dataframe of y data

Returns:

None

`transform`: performs the transform to generate output of only selected features

Args:

`X`: (dataframe), dataframe of X features

Returns:

dataframe: (dataframe), dataframe of selected X features

Methods Summary

`fit(X, savepath[, y, Xgroups])`

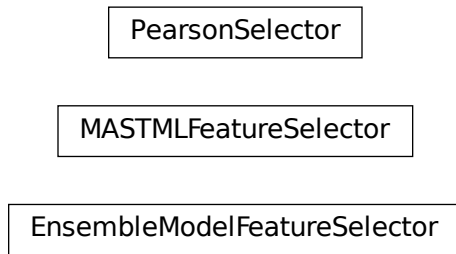
`transform(X)`

Methods Documentation

fit (*X*, *savepath*, *y=None*, *Xgroups=None*)

transform (*X*)

18.1.3 Class Inheritance Diagram



Code Documentation: Feature Normalizers

19.1 mastml.legos.feature_normalizers Module

This module contains a collection of classes and methods for normalizing features. Also included is connection with scikit-learn methods. See <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing> for more info.

19.1.1 Functions

<code>dataframify(transform)</code>	Method which is a decorator transforms output of scikit-learn feature normalizers from array to dataframe.
<code>wraps(wrapped[, assigned, updated])</code>	Decorator factory to apply <code>update_wrapper()</code> to a wrapper function

dataframify

`mastml.legos.feature_normalizers.dataframify(transform)`

Method which is a decorator transforms output of scikit-learn feature normalizers from array to dataframe. Enables preservation of column names.

Args:

transform: (function), a scikit-learn feature selector that has a transform method

Returns:

new_transform: (function), an amended version of the transform method that returns a dataframe

19.1.2 Classes

<code>BaseEstimator</code>	Base class for all estimators in scikit-learn.
<code>Binarizer(*[, threshold, copy])</code>	Binarize data (set feature values to 0 or 1) according to a threshold.
<code>MaxAbsScaler(*[, copy])</code>	Scale each feature by its maximum absolute value.
<code>MeanStdevScaler([features, mean, stdev])</code>	Class designed to normalize input data to a specified mean and standard deviation
<code>MinMaxScaler([feature_range, copy, clip])</code>	Transform features by scaling each feature to a given range.
<code>Normalizer([norm, copy])</code>	Normalize samples individually to unit norm.
<code>OneHotEncoder(*[, categories, drop, sparse, ...])</code>	Encode categorical features as a one-hot numeric array.
<code>QuantileTransformer(*[, n_quantiles, ...])</code>	Transform features using quantiles information.
<code>RobustScaler(*[, with_centering, ...])</code>	Scale features using statistics that are robust to outliers.
<code>StandardScaler(*[, copy, with_mean, with_std])</code>	Standardize features by removing the mean and scaling to unit variance
<code>TransformerMixin</code>	Mixin class for all transformers in scikit-learn.

MeanStdevScaler

class `mastml.legos.feature_normalizers.MeanStdevScaler` (*features=None, mean=0, stdev=1*)

Bases: `sklearn.base.BaseEstimator`, `sklearn.base.TransformerMixin`

Class designed to normalize input data to a specified mean and standard deviation

Args:

mean: (int/float), specified normalized mean of the data

stdev: (int/float), specified normalized standard deviation of the data

Methods:

`fit`: Obtains initial mean and stdev of data

Args:

df: (dataframe), dataframe of values to be normalized

Returns:

(self, the object instance)

`transform`: Normalizes the data to new mean and stdev values

Args:

df: (dataframe), dataframe of values to be normalized

Returns:

(dataframe), dataframe containing re-normalized data and any data that wasn't normalized

`inverse_transform`: Un-normalizes the data to the old mean and stdev values

Args:

df: (dataframe), dataframe of values to be un-normalized

Returns:

(dataframe), dataframe containing un-normalized data and any data that wasn't normalized

Methods Summary

fit(df[, y])

inverse_transform(df)

transform(df)

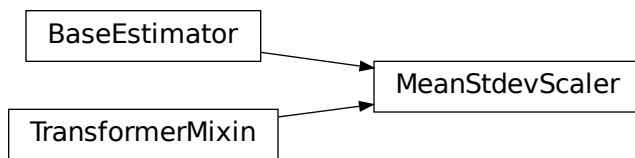
Methods Documentation

fit (df, y=None)

inverse_transform (df)

transform (df)

19.1.3 Class Inheritance Diagram



20.1 mastml.legos.randomizers Module

This module contains a class used to randomize the input y data, in order to create a “null model” for testing how rigorous other machine learning model predictions are.

20.1.1 Classes

<i>Randomizer()</i>	Class which randomizes X-y pairings by shuffling the y values
---------------------	---

Randomizer

class mastml.legos.randomizers.**Randomizer**

Bases: object

Class which randomizes X-y pairings by shuffling the y values

Args:

None

Methods:

fit: just passes through; present to maintain scikit-learn structure

Args:

None

transform: randomizes the values of a dataframe

Args:

df: (dataframe), a dataframe with data to be randomized

Returns:

(dataframe), a dataframe with randomized data

Methods Summary

fit()

transform(df)

Methods Documentation

fit()

transform(df)

20.1.2 Class Inheritance Diagram

Randomizer

21.1 mastml.legos.model_finder Module

This module provides a `name_to_constructor` dict for all models/estimators in scikit-learn, plus a couple test models and error handling functions

21.1.1 Functions

<code>check_models_mixed(model_names)</code>	Method used to check whether the user has mixed regression and classification tasks
<code>find_model(model_name)</code>	Method used to check model names conform to scikit-learn model/estimator names

check_models_mixed

`mastml.legos.model_finder.check_models_mixed(model_names)`

Method used to check whether the user has mixed regression and classification tasks

Args:

`model_names`: (list), list containing names of models/estimators

Returns:

(bool), whether or not a classifier was found, or raises exception if both regression and classification models present.

find_model

`mastml.legos.model_finder.find_model(model_name)`

Method used to check model names conform to scikit-learn model/estimator names

Args:

model_name: (str), the name of a model/estimator

Returns:

(str), the scikit-learn model name or raises InvalidModel error

21.1.2 Classes

<i>AlwaysFive</i> ([constant])	Class used as a test model that always predicts a value of 5.
<i>EnsembleRegressor</i> (n_estimators, num_samples, ...)	
<i>KerasRegressor</i> (conf_dict)	
<i>ModelImport</i> (model_path)	Class used to import pickled models from previous machine learning fits
<i>RandomGuesser</i> ()	Class used as a test model that always predicts random values for y data.

AlwaysFive

class mastml.legos.model_finder.**AlwaysFive** (constant=5)

Bases: sklearn.base.RegressorMixin

Class used as a test model that always predicts a value of 5.

Args:

constant: (int), the value to predict. Always 5 by default

Methods:

fit: Just passes through to maintain scikit-learn structure

predict: Provides predicted model values based on X features

Args:

X: (numpy array), array of X features

Returns:

(numpy array), prediction array where all values are equal to constant

Methods Summary

fit(X, y[, groups])

predict(X)

Methods Documentation

fit (X, y, groups=None)

predict (X)

EnsembleRegressor

```
class mastml.legos.model_finder.EnsembleRegressor (n_estimators, num_samples,  
                                                model_list, num_models)
```

Bases: object

Methods Summary

build_models()

fit(X, Y)

predict(X[, return_std])

setup(path)

stats_check_models(X, Y)

Methods Documentation

build_models ()

fit (*X, Y*)

predict (*X, return_std=False*)

setup (*path*)

stats_check_models (*X, Y*)

KerasRegressor

```
class mastml.legos.model_finder.KerasRegressor (conf_dict)
```

Bases: object

Methods Summary

build_model()

fit(X, Y)

predict(X)

summary()

Methods Documentation

build_model ()

fit (*X, Y*)

predict (*X*)

summary ()

ModelImport

```
class mastml.legos.model_finder.ModelImport (model_path)
```

Bases: object

Class used to import pickled models from previous machine learning fits

Args:

model_path (str): string designating the path to load the saved .pkl model file

Methods:

fit: Does nothing, present for compatibility purposes

Args:

X: Nonetype

y: Nonetype

groups: Nonetype

predict: Provides predicted model values based on X features

Args:

X: (numpy array), array of X features

Returns:

(numpy array), prediction array using imported model

Methods Summary

<code>fit([X, y, groups])</code>	Only here for compatibility
<code>predict(X)</code>	

Methods Documentation

fit (*X=None, y=None, groups=None*)

Only here for compatibility

predict (*X*)

RandomGuesser

class mastml.legos.model_finder.**RandomGuesser**

Bases: sklearn.base.RegressorMixin

Class used as a test model that always predicts random values for y data.

Args:

None

Methods:

fit: Constructs possible predicted values based on y data

Args:

y: (numpy array), array of y data

predict: Provides predicted model values based on X features

Args:

X: (numpy array), array of X features

Returns:

(numpy array), prediction array where all values are random selections of y data

Methods Summary

fit(X, y[, groups])

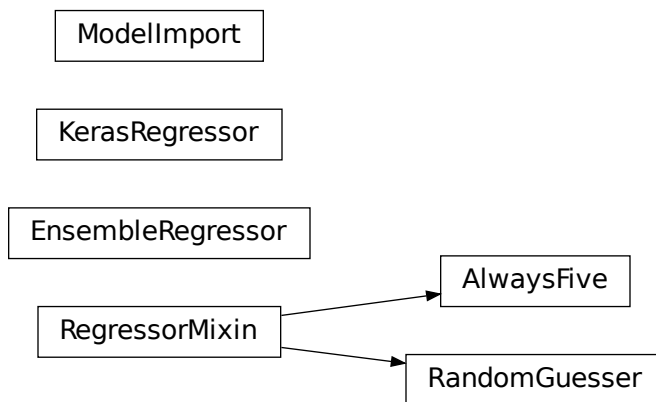
predict(X)

Methods Documentation

fit (X, y, groups=None)

predict (X)

21.1.3 Class Inheritance Diagram



22.1 mastml.legos.util_legos Module

This module contains a collection of classes for debugging and control flow

22.1.1 Classes

<code>BaseEstimator</code>	Base class for all estimators in scikit-learn.
<code>DataFrameFeatureUnion(transforms)</code>	Class for unioning dataframe generators (sklearn.pipeline.FeatureUnion always puts out arrays)
<code>DoNothing()</code>	Class for having a “null” transform where the output is the same as the input.
<code>TransformerMixin</code>	Mixin class for all transformers in scikit-learn.

DataFrameFeatureUnion

class mastml.legos.util_legos.**DataFrameFeatureUnion** (*transforms*)

Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin

Class for unioning dataframe generators (sklearn.pipeline.FeatureUnion always puts out arrays)

Args:

transforms: (list), list of scikit-learn functions, i.e. objects with a .fit or .transform method

Methods:

fit: Applies the .fit method for each transform

Args:

X: (numpy array), array of X features

transform: Transforms the output of the scikit-learn transformer into a dataframe

Args:

X: (numpy array), array of X features

Returns:

(dataframe), concatenated dataframe after all scikit-learn transforms have been completed

Methods Summary

fit(X[, y])
transform(X)

Methods Documentation

fit (X, y=None)

transform (X)

DoNothing

class mastml.legos.util_legos.**DoNothing**

Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin

Class for having a “null” transform where the output is the same as the input. Needed by MAST-ML as a placeholder if certain workflow aspects are not performed.

Args:

None

Methods:

fit: does nothing, just returns object instance. Needed to maintain same structure as scikit-learn classes

Args:

X: (numpy array), array of X features

transform: passes the input back out, in this case the array of X features

Args:

X: (numpy array), array of X features

Returns:

X: (numpy array), array of X features

Methods Summary

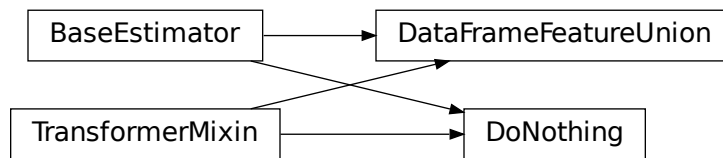
fit(X[, y])
transform(X)

Methods Documentation

fit (*X*, *y=None*)

transform (*X*)

22.1.2 Class Inheritance Diagram



Code Documentation: Feature Generators

23.1 mastml.legos.feature_generators Module

This module contains a collection of classes for generating input features to fit machine learning models to.

23.1.1 Functions

<code>clean_dataframe(df)</code>	Method to clean dataframes after feature generation has occurred, to remove columns that have a single missing or NaN value, or remove a row that is fully empty
----------------------------------	--

clean_dataframe

`mastml.legos.feature_generators.clean_dataframe(df)`

Method to clean dataframes after feature generation has occurred, to remove columns that have a single missing or NaN value, or remove a row that is fully empty

Args:

df: (dataframe), a post feature generation dataframe that needs cleaning

Returns:

df: (dataframe), the cleaned dataframe

23.1.2 Classes

<code>BaseEstimator</code>	Base class for all estimators in scikit-learn.
<code>ContainsElement(composition_feature, ...[, ...])</code>	Class to generate new categorical features (i.e.

Continued on next page

Table 2 – continued from previous page

<i>DataframeUtilities</i>	Class of basic utilities for dataframe manipulation, and exchanging between dataframes and numpy arrays
<i>Magpie</i> (composition_feature[, feature_types])	Class that wraps MagpieFeatureGeneration, giving it scikit-learn structure
<i>MagpieFeatureGeneration</i> (dataframe, ...)	Class to generate new features using Magpie data and dataframe containing material compositions
<i>MaterialsProject</i> (composition_feature, api_key)	Class that wraps MaterialsProjectFeatureGeneration, giving it scikit-learn structure
<i>MaterialsProjectFeatureGeneration</i> (dataframe, ...)	Class to generate new features using Materials Project data and dataframe containing material compositions. Datarame must have a column named “Material compositions”.
<i>Matminer</i> (structural_features, structure_col)	Class to generate structural features from matminer structure module. Args: structural_features: the structure feature(s) the user wants to instantiate and generate. structure_col: the dataframe column that contains the pymatgen structure object.
<i>NoGenerate</i> ()	Class for having a “null” transform where the output is the same as the input.
<i>PolynomialFeatures</i> ([features, degree, ...])	Class to generate polynomial features using scikit-learn’s polynomial features method. More info at: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html
<i>SklearnPolynomialFeatures</i>	alias of sklearn.preprocessing._data.PolynomialFeatures
<i>TransformerMixin</i>	Mixin class for all transformers in scikit-learn.

ContainsElement

```
class mastml.legos.feature_generators.ContainsElement(composition_feature,
                                                    element, new_name,
                                                    all_elements=False)
```

Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin

Class to generate new categorical features (i.e. values of 1 or 0) based on whether an input composition contains a certain designated element

Args:

composition_feature: (str), string denoting a chemical composition to generate elemental features from

element: (str), string representing the name of an element

new_name: (str), the name of the new feature column to be generated

all_elements: (bool), whether to generate new features for all elements present from all compositions in the dataset.

Methods:

fit: pass through, needed to maintain scikit-learn class structure

Args:

df: (dataframe), dataframe of input X and y data

transform: generate new element-specific features

Args:

df: (dataframe), dataframe of input X and y data

Returns:

df_trans: (dataframe), dataframe with generated element-specific features

Methods Summary

fit(df[, y])

transform(df[, y])

Methods Documentation

fit (df, y=None)

transform (df, y=None)

DataframeUtilities

class mastml.legos.feature_generators.**DataframeUtilities**

Bases: object

Class of basic utilities for dataframe manipulation, and exchanging between dataframes and numpy arrays

Args:

None

Methods:

merge_dataframe_columns : merge two dataframes by concatenating the column names (duplicate columns omitted)

Args:

dataframe1: (dataframe), a pandas dataframe object

dataframe2: (dataframe), a pandas dataframe object

Returns:

dataframe: (dataframe), merged dataframe

merge_dataframe_rows : merge two dataframes by concatenating the row contents (duplicate rows omitted)

Args:

dataframe1: (dataframe), a pandas dataframe object

dataframe2: (dataframe), a pandas dataframe object

Returns:

dataframe: (dataframe), merged dataframe

get_dataframe_statistics : obtain basic statistics about data contained in the dataframe

Args:

dataframe: (dataframe), a pandas dataframe object

Returns:

dataframe_stats: (dataframe), dataframe containing input dataframe statistics

dataframe_to_array : transform a pandas dataframe to a numpy array

Args:

dataframe: (dataframe), a pandas dataframe object

Returns:

array: (numpy array), a numpy array representation of the inputted dataframe

array_to_dataframe : transform a numpy array to a pandas dataframe

Args:

array: (numpy array), a numpy array

Returns:

dataframe: (dataframe), a pandas dataframe representation of the inputted numpy array

concatenate_arrays : merge two numpy arrays by concatenating along the columns

Args:

Xarray: (numpy array), a numpy array object

yarray: (numpy array), a numpy array object

Returns:

array: (numpy array), a numpy array merging the two input arrays

assign_columns_as_features : adds column names to dataframe based on the x and y feature names

Args:

dataframe: (dataframe), a pandas dataframe object

x_features: (list), list containing x feature names

y_feature: (str), target feature name

Returns:

dataframe: (dataframe), dataframe containing same data as input, with columns labeled with features

save_all_dataframe_statistics : obtain dataframe statistics and save it to a csv file

Args:

dataframe: (dataframe), a pandas dataframe object

data_path: (str), file path to save dataframe statistics to

Returns:

fname: (str), name of file dataframe stats saved to

Methods Summary

```

array_to_dataframe(array)
assign_columns_as_features(dataframe,
..., ...)
concatenate_arrays(X_array, y_array)
dataframe_to_array(dataframe)
get_dataframe_statistics(dataframe)
merge_dataframe_columns(dataframe1,
dataframe2)
merge_dataframe_rows(dataframe1,
dataframe2)
save_all_dataframe_statistics(dataframe,
...)

```

Methods Documentation

```

classmethod array_to_dataframe (array)
classmethod assign_columns_as_features (dataframe, x_features, y_feature, re-
move_first_row=True)
classmethod concatenate_arrays (X_array, y_array)
classmethod dataframe_to_array (dataframe)
classmethod get_dataframe_statistics (dataframe)
classmethod merge_dataframe_columns (dataframe1, dataframe2)
classmethod merge_dataframe_rows (dataframe1, dataframe2)
classmethod save_all_dataframe_statistics (dataframe, configdict)

```

Magpie

```

class mastml.legos.feature_generators.Magpie (composition_feature, feature_types=None)
Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin

```

Class that wraps MagpieFeatureGeneration, giving it scikit-learn structure

Args:

composition_feature: (str), string denoting a chemical composition to generate elemental features from

Methods:

fit: pass through, copies input columns as pre-generated features

Args:

df: (dataframe), input dataframe containing X and y data

transform: generate Magpie features

Args:

df: (dataframe), input dataframe containing X and y data

Returns:

df: (dataframe), output dataframe containing generated features, original features and y data

Methods Summary

```
fit(df[, y])
```

```
transform(df)
```

Methods Documentation

```
fit (df, y=None)
```

```
transform (df)
```

MagpieFeatureGeneration

```
class mastml.legos.feature_generators.MagpieFeatureGeneration (dataframe, com-
position_feature,
feature_types)
```

Bases: object

Class to generate new features using Magpie data and dataframe containing material compositions

Args:

dataframe: (pandas dataframe), dataframe containing x and y data and feature names

composition_feature: (str), string denoting a chemical composition to generate elemental features from

feature_types: (list), list containing types of magpie features to include in the final dataframe. Options include ["composition_avg", "arithmetic_avg", "max", "min", "difference", "elements"]. Specifying nothing will include all features.

Methods:

generate_magpie_features : generates magpie feature set based on compositions in dataframe

Args:

None

Returns:

dataframe: (dataframe) : dataframe containing magpie feature set

Methods Summary

```
generate_magpie_features()
```

Methods Documentation

```
generate_magpie_features ()
```

MaterialsProject

```
class mastml.legos.feature_generators.MaterialsProject (composition_feature,
api_key)
```

Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin

Class that wraps MaterialsProjectFeatureGeneration, giving it scikit-learn structure

Args:

composition_feature: (str), string denoting a chemical composition to generate elemental features from

mapi_key: (str), string denoting your Materials Project API key

Methods:

fit: pass through, copies input columns as pre-generated features

Args:

df: (dataframe), input dataframe containing X and y data

transform: generate Materials Project features

Args:

df: (dataframe), input dataframe containing X and y data

Returns:

df: (dataframe), output dataframe containing generated features, original features and y data

Methods Summary

fit(df[, y])

transform(df)

Methods Documentation

fit (df, y=None)

transform (df)

MaterialsProjectFeatureGeneration

```
class mastml.legos.feature_generators.MaterialsProjectFeatureGeneration (dataframe,
                                                                    mapi_key,
                                                                    com-
                                                                    po-
                                                                    si-
                                                                    tion_feature)
```

Bases: object

Class to generate new features using Materials Project data and dataframe containing material compositions
Datarame must have a column named “Material compositions”.

Args: dataframe: (dataframe), dataframe containing x and y data and feature names

mapi_key: (str), string denoting your Materials Project API key

composition_feature: (str), string denoting a chemical composition to generate elemental features from

Methods:

generate_materialsproject_features : generates materials project feature set based on compositions in dataframe

Args:

None

Returns: dataframe: (dataframe), dataframe containing materials project feature set

Methods Summary

`generate_materialsproject_features()`

Methods Documentation

generate_materialsproject_features()

Matminer

class mastml.legos.feature_generators.**Matminer** (*structural_features*, *structure_col*)

Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin

Class to generate structural features from matminer structure module Args:

structural_features: the structure feature(s) the user wants to instantiate and generate *structure_col*: the dataframe column that contains the pymatgen structure object. Matminer needs a pymatgen structure object in order to instantiate the structural feature

Methods: fit: pass through, needed to maintain scikit-learn class structure Args:

df: (dataframe), dataframe of input x and y data

transform: main method that iterates through rows of dataframe to create pymatgen structure objects for matminer routines. Iterates through list of structural features from conf file and instantiates each structure; drops unused dataframe columns and returns the generated features dataframe Args:

df: (dataframe), dataframe containing the path of file to create pymatgen structure object which is under the *structure_col* column

Returns: (dataframe), the generated features dataframe

Methods Summary

<code>fit(df[, y])</code>	
<code>retrieve_AFLOW(criteria, properties[, ...])</code>	
<code>retrieve_MDF(criteria[, anonymous, ...])</code>	
<code>retrieve_MPDS(criteria[, properties, ...])</code>	
<code>retrieve_citrine(criteria, properties, ...)</code>	Gets a Pandas dataframe object from data retrieved from the Citrine API.
<code>retrieve_mp(criteria[, properties, ...])</code>	Gets data from MP in a dataframe format.
<code>transform(df[, y])</code>	

Methods Documentation

fit (*df*, *y=None*)

retrieve_AFLOW (*criteria, properties, files=None, request_size=10000, request_limit=0, index_auid=True*)

retrieve_MDF (*criteria, anonymous=False, properties=None, unwind_arrays=True*)

retrieve_MPDS (*criteria, properties=None, api_key=None, endpoint=None*)

retrieve_citrine (*criteria, properties, common_fields, secondary_fields, print_properties_options, api_key*)

Gets a Pandas dataframe object from data retrieved from the Citrine API. Args:

criteria (dict): see `get_data` method for supported keys except `prop`; `prop` should be included in `properties`.

properties ([str]): requested properties/fields/columns. For example, ["Seebeck coefficient", "Band gap"]. If unsure about the exact words, capitalization, etc try something like ["gap"] and "max_results": 3 and `print_properties_options=True` to see the exact options for this field

common_fields ([str]): fields that are common to all the requested properties. Common example can be "chemicalFormula". Look for suggested common fields after a quick query for more info

secondary_fields (bool): if `True`, fields not included in `properties` may be added to the output (e.g. references). Recommended only if `len(properties)==1`

print_properties_options (bool): whether to print available options for "properties" and "common_fields" arguments.

api_key: (str) Your Citrine API key, or `None` if you've set the `CITRINE_KEY` environment variable

return: (object) Pandas dataframe object containing the results notes/bugs: `criteria` needs a dictionary, not specified in `get_data()` as mentioned,

and example on documentation webpage does not work. What to fix for dataframe integration into mastml?

retrieve_mp (*criteria, properties=['band_gap', 'volume', 'density', 'formation_energy_per_atom'], index_mpid=True, api_key=None*)

Gets data from MP in a dataframe format. See `api_link` for more details. Args:

criteria (dict): (str/dict) see `MPRester.query()` for a description of this parameter. String examples: "mp-1234", "Fe2O3", "Li-Fe-O", "*2O3". Dict example: {"band_gap": {"\$gt": 1}}

properties ([str]): (list) see `MPRester.query()` for a description of this parameter. Example: ["formula", "formation_energy_per_atom"]

plus: "structure", "initial_structure", "final_structure", "bandstructure" (line mode), "bandstructure_uniform", "phonon_bandstructure", "phonon_ddb", "phonon_bandstructure", "phonon_dos". Note that for a long list of compounds, it may

take a long time to retrieve some of these objects.

index_mpid (bool): (bool) Whether to set the `materials_id` as the dataframe index.

api_key: (str) Your Materials Project API key, or `None` if you've set up your pymatgen config.

Returns (pandas.DataFrame): containing results notes/bugs: works pretty great, API easy to use and accurate. What to fix for

dataframe integration into mastml?

```
transform (df, y=None)
```

NoGenerate

```
class mastml.legos.feature_generators.NoGenerate
```

```
Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin
```

Class for having a “null” transform where the output is the same as the input. Needed by MAST-ML as a placeholder if certain workflow aspects are not performed.

Args:

None

Methods:

fit: does nothing, just returns object instance. Needed to maintain same structure as scikit-learn classes

Args:

X: (dataframe), dataframe of X features

transform: passes the input back out, in this case the array of X features

Args:

X: (dataframe), dataframe of X features

Returns:

(dataframe), dataframe of X features

Methods Summary

```
fit(X[, y])
```

```
transform(X)
```

Methods Documentation

```
fit (X, y=None)
```

```
transform (X)
```

PolynomialFeatures

```
class mastml.legos.feature_generators.PolynomialFeatures (features=None, degree=2, interaction_only=False, include_bias=True)
```

```
Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin
```

Class to generate polynomial features using scikit-learn’s polynomial features method More info at: <http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html>

Args:

degree: (int), degree of polynomial features

interaction_only: (bool), If true, only interaction features are produced: features that are products of at most degree distinct input features (so not $x[1]**2$, $x[0]*x[2]**3$, etc.).

include_bias: (bool), If True (default), then include a bias column, the feature in which all polynomial powers are zero (i.e. a column of ones - acts as an intercept term in a linear model).

Methods:

fit: conducts fit method of polynomial feature generation

Args:

df: (dataframe), dataframe of input X and y data

transform: generates dataframe containing polynomial features

Args:

df: (dataframe), dataframe of input X and y data

Returns:

(dataframe), dataframe containing new polynomial features, plus original features present

Methods Summary

`fit(df[, y])`

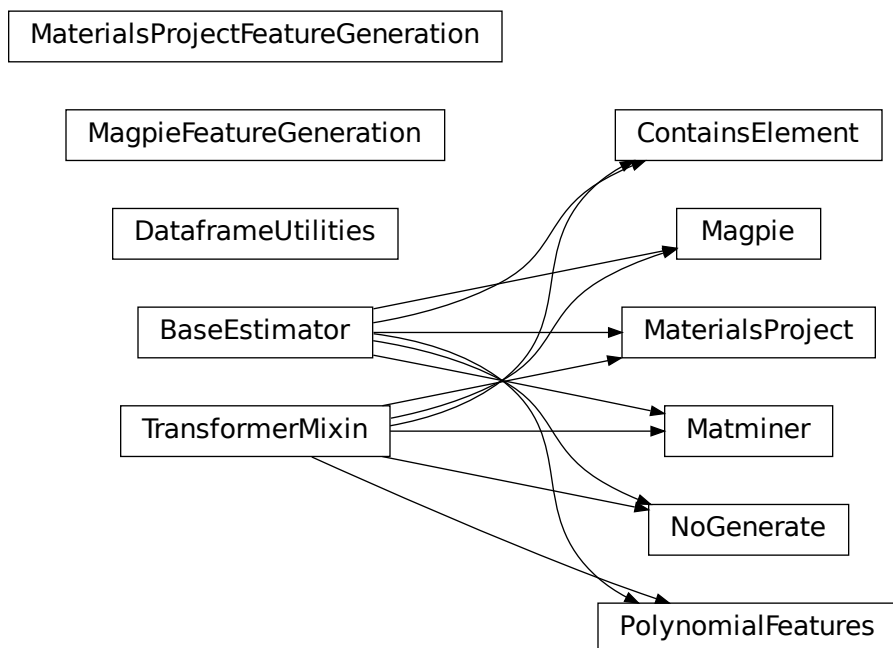
`transform(df)`

Methods Documentation

fit (df, y=None)

transform (df)

23.1.3 Class Inheritance Diagram



CHAPTER 24

Indices and tables

- `genindex`
- `modindex`
- `search`

m

- `mastml.conf_parser`, 59
- `mastml.data_cleaner`, 61
- `mastml.data_loader`, 65
- `mastml.html_helper`, 105
- `mastml.learning_curve`, 67
- `mastml.legos.clusterers`, 69
- `mastml.legos.data_splitters`, 71
- `mastml.legos.feature_generators`, 131
- `mastml.legos.feature_normalizers`, 115
- `mastml.legos.feature_selectors`, 109
- `mastml.legos.model_finder`, 121
- `mastml.legos.randomizers`, 119
- `mastml.legos.util_legos`, 127
- `mastml.mastml_driver`, 83
- `mastml.metrics`, 57
- `mastml.plot_helper`, 87
- `mastml.utils`, 77

A

activate_logging() (in module mastml.utils), 77
adjusted_r2_score() (in module mastml.metrics), 57
AlwaysFive (class in mastml.legos.model_finder), 122
array_to_dataframe() (mastml.legos.feature_generators.DataFrameUtilities class method), 135
assign_columns_as_features() (mastml.legos.feature_generators.DataFrameUtilities class method), 135

B

BetweenFilter (class in mastml.utils), 79
Bootstrap (class in mastml.legos.data_splitters), 72
build_model() (mastml.legos.model_finder.KerasRegressor class method), 123
build_models() (mastml.legos.model_finder.EnsembleRegressor class method), 123

C

check_and_fetch_names() (in module mastml.metrics), 58
check_models_mixed() (in module mastml.legos.model_finder), 121
check_paths() (in module mastml.mastml_driver), 84
clean_dataframe() (in module mastml.legos.feature_generators), 131
columns_with_strings() (in module mastml.data_cleaner), 61
concatenate_arrays() (mastml.legos.feature_generators.DataFrameUtilities class method), 135
ConfError, 80
ContainsElement (class in mastml.legos.feature_generators), 132

D

dataframe_to_array()

(mastml.legos.feature_generators.DataFrameUtilities class method), 135
DataFrameFeatureUnion (class in mastml.legos.util_legos), 127
DataFrameUtilities (class in mastml.legos.feature_generators), 133
dataframify() (in module mastml.legos.feature_normalizers), 115
dataframify_new_column_names() (in module mastml.legos.feature_selectors), 109
dataframify_selector() (in module mastml.legos.feature_selectors), 110
DoNothing (class in mastml.legos.util_legos), 128

E

EnsembleModelFeatureSelector (class in mastml.legos.feature_selectors), 111
EnsembleRegressor (class in mastml.legos.model_finder), 123

F

feature_learning_curve() (in module mastml.learning_curve), 67
FileNotFoundError, 80
FileTypeError, 80
filter() (mastml.utils.BetweenFilter method), 79
find_model() (in module mastml.legos.model_finder), 121
fit() (mastml.data_cleaner.PPCA method), 63
fit() (mastml.legos.feature_generators.ContainsElement method), 133
fit() (mastml.legos.feature_generators.Magpie method), 136
fit() (mastml.legos.feature_generators.MaterialsProject method), 137
fit() (mastml.legos.feature_generators.Matminer method), 138
fit() (mastml.legos.feature_generators.NoGenerate method), 140

[fit\(\)](#) (*mastml.legos.feature_generators.PolynomialFeatureGenerator* method), 141
[fit\(\)](#) (*mastml.legos.feature_normalizers.MeanStdevScaler* method), 117
[fit\(\)](#) (*mastml.legos.feature_selectors.EnsembleModelFeatureSelector* method), 111
[fit\(\)](#) (*mastml.legos.feature_selectors.MASTMLFeatureSelector* method), 112
[fit\(\)](#) (*mastml.legos.feature_selectors.PearsonSelector* method), 113
[fit\(\)](#) (*mastml.legos.model_finder.AlwaysFive* method), 122
[fit\(\)](#) (*mastml.legos.model_finder.EnsembleRegressor* method), 123
[fit\(\)](#) (*mastml.legos.model_finder.KerasRegressor* method), 123
[fit\(\)](#) (*mastml.legos.model_finder.ModelImport* method), 124
[fit\(\)](#) (*mastml.legos.model_finder.RandomGuesser* method), 125
[fit\(\)](#) (*mastml.legos.randomizers.Randomizer* method), 120
[fit\(\)](#) (*mastml.legos.util_legos.DataFrameFeatureUnion* method), 128
[fit\(\)](#) (*mastml.legos.util_legos.DoNothing* method), 129
[fitify_just_use_values\(\)](#) (in module *mastml.legos.feature_selectors*), 110
[fix_types\(\)](#) (in module *mastml.conf_parser*), 59
[flag_outliers\(\)](#) (in module *mastml.data_cleaner*), 62

G

[generate_magpie_features\(\)](#) (*mastml.legos.feature_generators.MagpieFeatureGeneration* method), 136
[generate_materialsproject_features\(\)](#) (*mastml.legos.feature_generators.MaterialsProjectFeatureGeneration* method), 138
[get_commandline_args\(\)](#) (in module *mastml.mastml_driver*), 84
[get_dataframe_statistics\(\)](#) (*mastml.legos.feature_generators.DataFrameUtility* class method), 135
[get_divisor\(\)](#) (in module *mastml.plot_helper*), 89
[get_histogram_bins\(\)](#) (in module *mastml.plot_helper*), 90
[get_n_splits\(\)](#) (*mastml.legos.data_splitters.Bootstrap* method), 73
[get_n_splits\(\)](#) (*mastml.legos.data_splitters.EachGroup* method), 73
[get_n_splits\(\)](#) (*mastml.legos.data_splitters.LeaveCloseCompositionsOut* method), 74

H

[get_n_splits\(\)](#) (*mastml.legos.data_splitters.LeaveOutPercent* method), 75
[get_n_splits\(\)](#) (*mastml.legos.data_splitters.NoSplit* method), 75
[get_n_splits\(\)](#) (*mastml.legos.data_splitters.SplittersUnion* method), 76
[imputation\(\)](#) (in module *mastml.data_cleaner*), 62
[indices](#) (*mastml.legos.data_splitters.Bootstrap* attribute), 72
[InvalidConfParameters](#), 80
[InvalidConfSection](#), 80
[InvalidConfSubSection](#), 80
[InvalidModel](#), 80
[InvalidValue](#), 80
[inverse_transform\(\)](#) (*mastml.legos.feature_normalizers.MeanStdevScaler* method), 117
[ipynb_maker\(\)](#) (in module *mastml.plot_helper*), 90
[is_test_image\(\)](#) (in module *mastml.html_helper*), 106
[is_train_image\(\)](#) (in module *mastml.html_helper*), 106

J

[JustEachGroup](#) (class in *mastml.legos.data_splitters*), 73

K

[KerasRegressor](#) (class in *mastml.legos.model_finder*), 123

L

[LeaveCloseCompositionsOut](#) (class in *mastml.legos.data_splitters*), 73
[LeaveOutPercent](#) (class in *mastml.legos.data_splitters*), 74
[load\(\)](#) (*mastml.data_cleaner.PPCA* method), 63
[load_data\(\)](#) (in module *mastml.data_loader*), 65
[log_header\(\)](#) (in module *mastml.utils*), 78

M

[Magpie](#) (class in *mastml.legos.feature_generators*), 135
[MagpieFeatureGeneration](#) (class in *mastml.legos.feature_generators*), 136
[main\(\)](#) (in module *mastml.mastml_driver*), 84
[make_axis_same\(\)](#) (in module *mastml.plot_helper*), 90
[make_error_plots\(\)](#) (in module *mastml.plot_helper*), 90
[make_fig_ax\(\)](#) (in module *mastml.plot_helper*), 90
[make_fig_ax_square\(\)](#) (in module *mastml.plot_helper*), 91

make_html() (in module mastml.html_helper), 106
make_image() (in module mastml.html_helper), 106
make_link() (in module mastml.html_helper), 107
make_train_test_plots() (in module mastml.plot_helper), 91
MastError, 80
mastml.conf_parser (module), 59
mastml.data_cleaner (module), 61
mastml.data_loader (module), 65
mastml.html_helper (module), 105
mastml.learning_curve (module), 67
mastml.legos.clusterers (module), 69
mastml.legos.data_splitters (module), 71
mastml.legos.feature_generators (module), 131
mastml.legos.feature_normalizers (module), 115
mastml.legos.feature_selectors (module), 109
mastml.legos.model_finder (module), 121
mastml.legos.randomizers (module), 119
mastml.legos.util_legos (module), 127
mastml.mastml_driver (module), 83
mastml.metrics (module), 57
mastml.plot_helper (module), 87
mastml.utils (module), 77
mastml_run() (in module mastml.mastml_driver), 85
MASTMLFeatureSelector (class in mastml.legos.feature_selectors), 111
MaterialsProject (class in mastml.legos.feature_generators), 136
MaterialsProjectFeatureGeneration (class in mastml.legos.feature_generators), 137
Matminer (class in mastml.legos.feature_generators), 138
MeanStdevScaler (class in mastml.legos.feature_normalizers), 116
merge_dataframe_columns() (mastml.legos.feature_generators.DataframeUtilities class method), 135
merge_dataframe_rows() (mastml.legos.feature_generators.DataframeUtilities class method), 135
MissingColumnError, 81
ModelImport (class in mastml.legos.model_finder), 123
mybool() (in module mastml.conf_parser), 60

N

nice_mean() (in module mastml.plot_helper), 92
nice_names() (in module mastml.plot_helper), 92
nice_range() (in module mastml.plot_helper), 92
nice_range() (in module mastml.utils), 78
nice_std() (in module mastml.plot_helper), 92

NoGenerate (class in mastml.legos.feature_generators), 140
NoSplit (class in mastml.legos.data_splitters), 75

P

parse_conf_file() (in module mastml.conf_parser), 60
parse_error_data() (in module mastml.plot_helper), 92
PearsonSelector (class in mastml.legos.feature_selectors), 112
plot_1d_heatmap() (in module mastml.plot_helper), 92
plot_2d_heatmap() (in module mastml.plot_helper), 93
plot_3d_heatmap() (in module mastml.plot_helper), 93
plot_average_cumulative_normalized_error() (in module mastml.plot_helper), 94
plot_average_normalized_error() (in module mastml.plot_helper), 94
plot_best_worst_per_point() (in module mastml.plot_helper), 94
plot_best_worst_split() (in module mastml.plot_helper), 95
plot_confusion_matrix() (in module mastml.plot_helper), 95
plot_cumulative_normalized_error() (in module mastml.plot_helper), 96
plot_keras_history() (in module mastml.plot_helper), 96
plot_learning_curve() (in module mastml.plot_helper), 96
plot_learning_curve_convergence() (in module mastml.plot_helper), 97
plot_metric_vs_group() (in module mastml.plot_helper), 97
plot_metric_vs_group_size() (in module mastml.plot_helper), 98
plot_normalized_error() (in module mastml.plot_helper), 98
plot_precision_recall_curve() (in module mastml.plot_helper), 98
plot_predicted_vs_true() (in module mastml.plot_helper), 99
plot_predicted_vs_trueBars() (in module mastml.plot_helper), 99
plot_real_vs_predicted_error() (in module mastml.plot_helper), 99
plot_residuals_histogram() (in module mastml.plot_helper), 99
plot_roc_curve() (in module mastml.plot_helper), 100

`plot_scatter()` (in module `mastml.plot_helper`), 100
`plot_stats()` (in module `mastml.plot_helper`), 100
`plot_target_histogram()` (in module `mastml.plot_helper`), 101
`PolynomialFeatures` (class in `mastml.legos.feature_generators`), 140
`PPCA` (class in `mastml.data_cleaner`), 62
`ppca()` (in module `mastml.data_cleaner`), 62
`predict()` (`mastml.legos.model_finder.AlwaysFive` method), 122
`predict()` (`mastml.legos.model_finder.EnsembleRegressor` method), 123
`predict()` (`mastml.legos.model_finder.KerasRegressor` method), 123
`predict()` (`mastml.legos.model_finder.ModelImport` method), 124
`predict()` (`mastml.legos.model_finder.RandomGuesser` method), 125
`prediction_intervals()` (in module `mastml.plot_helper`), 101

R

`r2_score_fitted()` (in module `mastml.metrics`), 58
`r2_score_noint()` (in module `mastml.metrics`), 58
`RandomGuesser` (class in `mastml.legos.model_finder`), 124
`Randomizer` (class in `mastml.legos.randomizers`), 119
`recursive_max()` (in module `mastml.plot_helper`), 102
`recursive_max_and_min()` (in module `mastml.plot_helper`), 102
`recursive_min()` (in module `mastml.plot_helper`), 102
`remove()` (in module `mastml.data_cleaner`), 62
`retrieve_AFLOW()` (`mastml.legos.feature_generators.Matminer` method), 138
`retrieve_citrine()` (`mastml.legos.feature_generators.Matminer` method), 139
`retrieve_MDF()` (`mastml.legos.feature_generators.Matminer` method), 139
`retrieve_mp()` (`mastml.legos.feature_generators.Matminer` method), 139
`retrieve_MPDS()` (`mastml.legos.feature_generators.Matminer` method), 139
`rmse_over_stdev()` (in module `mastml.metrics`), 58
`root_mean_squared_error()` (in module `mastml.metrics`), 58
`round_down()` (in module `mastml.plot_helper`), 102
`round_up()` (in module `mastml.plot_helper`), 103
`rounder()` (in module `mastml.plot_helper`), 103

S

`sample_learning_curve()` (in module `mastml.learning_curve`), 68
`save()` (`mastml.data_cleaner.PPCA` method), 63
`save_all_dataframe_statistics()` (`mastml.legos.feature_generators.DataFrameUtilities` class method), 135
`setup()` (`mastml.legos.model_finder.EnsembleRegressor` method), 123
`show_combo()` (in module `mastml.html_helper`), 107
`simple_section()` (in module `mastml.html_helper`), 107
`split()` (`mastml.legos.data_splitters.Bootstrap` method), 73
`split()` (`mastml.legos.data_splitters.EachGroup` method), 73
`split()` (`mastml.legos.data_splitters.LeaveCloseCompositionsOut` method), 74
`split()` (`mastml.legos.data_splitters.LeaveOutPercent` method), 75
`split()` (`mastml.legos.data_splitters.NoSplit` method), 75
`split()` (`mastml.legos.data_splitters.SplittersUnion` method), 76
`SplittersUnion` (class in `mastml.legos.data_splitters`), 75
`stat_to_string()` (in module `mastml.plot_helper`), 103
`stats_check_models()` (`mastml.legos.model_finder.EnsembleRegressor` method), 123
`summary()` (`mastml.legos.model_finder.KerasRegressor` method), 123

T

`transform()` (`mastml.data_cleaner.PPCA` method), 63
`transform()` (`mastml.legos.feature_generators.ContainsElement` method), 133
`transform()` (`mastml.legos.feature_generators.Magpie` method), 136
`transform()` (`mastml.legos.feature_generators.MaterialsProject` method), 137
`transform()` (`mastml.legos.feature_generators.Matminer` method), 140
`transform()` (`mastml.legos.feature_generators.NoGenerate` method), 140
`transform()` (`mastml.legos.feature_generators.PolynomialFeatures` method), 141
`transform()` (`mastml.legos.feature_normalizers.MeanStdevScaler` method), 117
`transform()` (`mastml.legos.feature_selectors.EnsembleModelFeatureSe` method), 111

`transform()` (*mastml.legos.feature_selectors.MASTMLFeatureSelector*
method), [112](#)
`transform()` (*mastml.legos.feature_selectors.PearsonSelector*
method), [113](#)
`transform()` (*mastml.legos.randomizers.Randomizer*
method), [120](#)
`transform()` (*mastml.legos.util_legos.DataFrameFeatureUnion*
method), [128](#)
`transform()` (*mastml.legos.util_legos.DoNothing*
method), [129](#)
`trim_array()` (*in module mastml.plot_helper*), [103](#)

V

`verboselize_logger()` (*in module mastml.utils*),
[78](#)