MLMR
Summer 2018

MAST Machine Learning Exercises

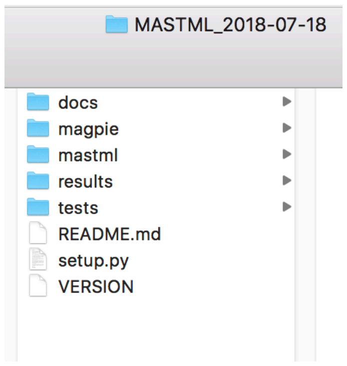
During this exercise, we will get hands-on experience building, training, and optimizing machine learning models. In doing so, we will explore issues of Cross Validation (CV) for evaluating model performance, the effectiveness of different model types, and model optimization. By the end of the exercise you should be able to:

1. Explain how different CV schemes can change how we interpret model performance.
2. Use CV metrics such as RMSE and R^2 to choose a model type and optimize its hyperparameters.
3. Run the MAST-ML code package to quickly iterate through a number of ML models and CV schemes.

1 Setting up MAST-ML

Attached with this lab handout was a MASTML_code_MLMR.zip file containing the MAST-ML code package and the necessary data and input files we will use throughout these exercises. You can also download MASTML from (https://github.com/uw-cmg/MAST-ML/tree/dev_Ryan). Documentation for MASTML is available at (https://github.com/uw-cmg/MAST-ML/blob/dev_Ryan/docs/how_to_mastml.md). To setup MASTML:

1. Download and unzip the MASTML_code_MLMR.zip file into a directory of your choice
2. Enter the MASTML directory created from unzipping the file. You should see the following directory tree:

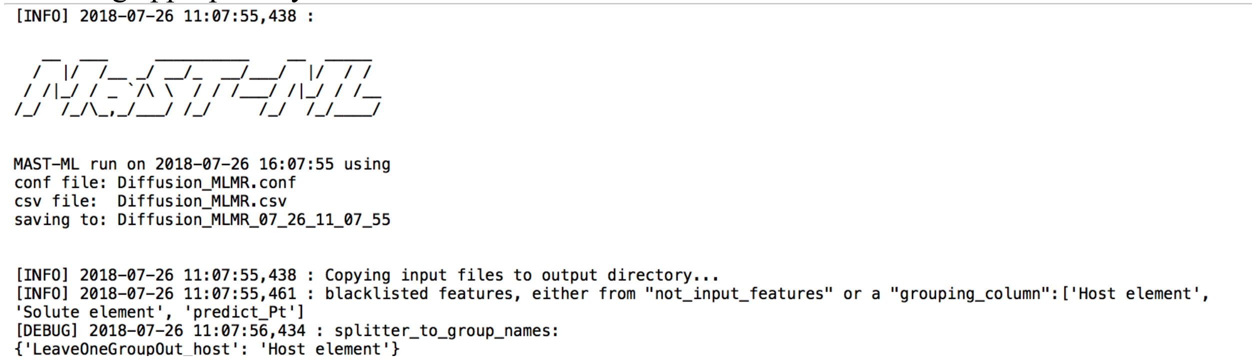


To install the required dependencies, run 'python setup.py install'. Once this is complete, all functionalities of MASTML should be ready to go. When performing your first MASTML run, if you receive a RuntimeError saying Python is not installed as a framework, this is related to an issue with matplotlib and Mac OS. To resolve it, create a file ~/.matplotlib/matplotlibrc in the ~/.matplotlib directory and add the following single line to that file: "backend: TkAgg".

MASTML takes two files as input: a data file in csv format and an input file with form <FileName> .conf. The path of data file for this workshop is tests/csv/Diffusion_MLMR.csv. The path of the input file is tests/conf/Diffusion_MLMR.conf. As performing the first MASTML run will take a few minutes, go ahead and run it now, then while it is running, continue reading below. **To run MASTML with these files, enter the following command in your terminal while in the main MASTML directory as shown in the previous screenshot:**

```
python3 -m mastml.mastml tests/conf/Diffusion_MLMR.conf
tests/csv/Diffusion_MLMR.csv -o results/Diffusion_MLMR
```

If you see a MASTML logo and initial output that looks like the following, then the run is executing appropriately:



```
[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'}
```

2 Dataset Introduction

The dataset we'll be working with today consists of migration barriers for dilute solute diffusion through a host elemental metal calculated using Density Functional Theory. Each host and each solute are single elements. An example of a data point in the dataset could be the migration barrier of dilute Cu diffusing through a host Al crystal. The dataset contains 15 host elements that span FCC, BCC, and HCP crystal structures. For each host, there is an average of around 27 solutes, resulting in a total dataset size of about 400 data points. This dataset is an extension of that published and studied in references [1,2].

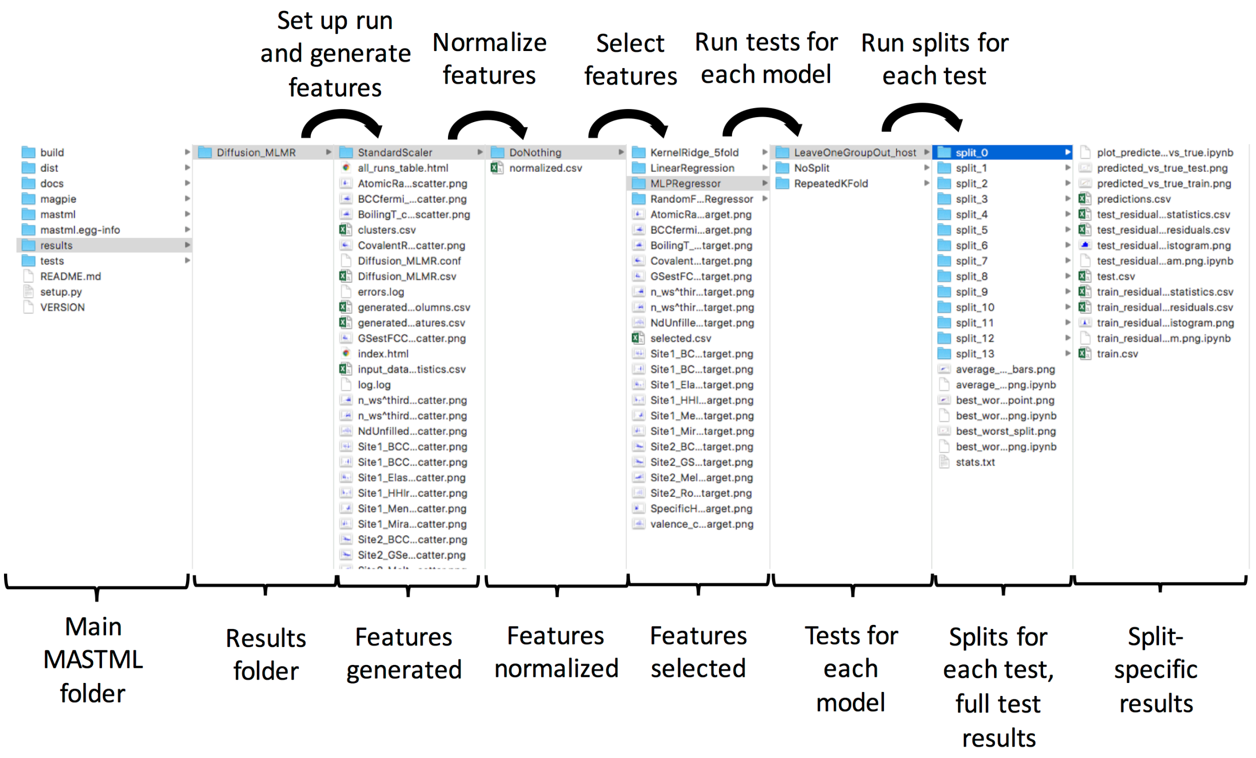
The dataset also has pre-generated and selected features that we'll use throughout the exercise to train our ML models. These features were generated from an elemental property database of physical and chemical properties of elements in their atomic form, which has been compiled from the Materials Agnostic Platform for Informatics and Exploration (MAGPIE)[3]. From this broad list of features, the most promising features have been selected using feature selection routines in the Scikit-learn python code package[4]. Both of these steps- the generation and selection of features, can be performed with MAST-ML, but will not be the focus of today's exercise. Through the exercises we'll be focusing on how we can take this dataset of migration barriers and relevant features and build useful machine learning models to see (1) how well different models and CV schemes perform to predict the migration barriers of materials in this dataset and (2) use these models and CV schemes to make predictions on solute migration in Pt, which will serve as our prediction dataset for final model assessment.

3 Machine Learning Key Concepts

The first section of exercises highlights some key considerations when fitting machine learning models. We will assess different CV schemes and use the fitted results to understand the limits of model performance. And specifically, how to optimize model hyperparameters to obtain the best performance.

The first MASTML run we will do will fit 4 models to the diffusion dataset: multivariate linear regression, kernel ridge regression with a Gaussian kernel, random forest, and a basic neural network. For each test, 3 different data splits will be performed to assess model performance: a single full fit (all data used, no CV), 5 tests of random 5-fold cross-validation, and leave out group CV, where each group *i* corresponds to all the entries associated with a host element *i* in the input data file (thus we have 15 groups, associated with the 15 host elements).

The MASTML run that you executed above should be nearly or totally complete by now. Under the results/Diffusion_MLMR directory there will be a series of sub-directories which correspond to each section of the MASTML workflow as discussed in the lecture slides.



The first task for these exercises is to pull out some of the high-level results to make some initial comparisons between the model types and CV types chosen.

Open the stats.txt file for a given model and CV test, which contains a summary of each test that was written and use the information there to fill in Table 1 with the RMSE values for each combination of CV scheme and Model Type. Do the same for R^2 values in Table 2.

Table 1. Summary of CV RMSE statistics

	NoSplit (Full fit)	Random 5-Fold	Leave Out Host
Multivariate Linear			
Random Forest			
Gaussian Kernel Ridge Regression			
Neural Network			

Table 2. Summary of CV R^2 statistics

	NoSplit (Full fit)	Random 5-Fold	Leave Out Host
Multivariate Linear			
Random Forest			

Gaussian Kernel Ridge Regression			
Neural Network			

3.1 Cross Validation Comparison

By running the Diffusion_MLMR.conf input file with MASTML, we have built a series of different models and performed a series of CV tests on each model. For this section, we'll focus on the results for one model type and see how the different CV tests that were performed could influence the conclusion we draw from the test. The overall goal of CV testing is to assess the predictive capability of a model, and each CV scheme can give different information about that predictive capability.

We've performed 3 different versions of CV in the Diffusion_MLMR.conf run: a single full fit (all data used, no CV), 5 tests of random 5-fold cross-validation, and leave out group CV, where the groups are denoted as each different host element in the input data file. The single full fit (denoted "NoSplit") fits all of the data at once with no CV, the 5-fold CV leaves 20% of the data out at random and then predicts that back. But what if we wanted to do something a bit more materials specific? Often in materials research our data can be heavily clustered, and therefore leaving out random chunks of data might not be an accurate representation of predictive ability. The final CV scheme tackles this by leaving out groups of data that we would expect to be very similar. The Leave Out Host CV test groups the data based on host elements and leaves out each host one by one during CV.

Compare the values of RMSE for the various CV schemes and consider the following questions:

What is the range of values?
Are there trends that are consistent across CV scheme?
Is one type generally more optimistic about performance?
If we wanted to predict new host data, which CV scheme might we choose?
Compare the same things using the R^2 metric. Does this give the same conclusions? If not this may suggest that conclusions aren't clear.

3.2 Model Type Comparison

Along with considering CV scheme, model type can also dramatically change performance. In the Diffusion_MLMR.conf run we've generated four model types of varying complexity. Using the same results in Table 1 and Table 2 lets now look at how performance can vary across models.

Consider the following questions:

For Leave Out Host CV which model has the best performance? Compare Leave Out Host RMSE scores.
Is this consistent across all CV schemes?
If the answer is no, then does the CV scheme selected influence model choice?

3.3 Hyperparameter Optimization

MASTML currently supports grid search and genetic algorithm methods to optimize hyperparameters. Due to its higher speed, we will use the genetic search for this workshop. There is a separate data file (Diffusion_MLMR_hyperparam.csv) and input file (Diffusion_MLMR_hyperparam.conf) to run hyperparameter optimization. If you open the data file, you may notice the data for Pt host has been removed! What we will do here is optimize a model of your choice on the diffusion data that does not contain Pt, then use your optimized model in the next section to try to accurately predict the Pt migration barriers as accurately as possible. Open up the Diffusion_MLMR_hyperparam.conf file and uncomment the parameter block coinciding with the model you'd like to optimize (note that optimizing the neural network is quite slow by comparison with the other models. If you are running short on time it's suggested to select another model type to look at during the activity. Mabye try NN in your own time if interested.). Make sure you only have one block of [GeneticSearch] and its parameters uncommented. You can run the hyperparameter in a similar way as a standard MASTML run, but use this command in your terminal:

```
python3 -m mastml.search.search
tests/conf/Diffusion_MLMR_hyperparam.conf
tests/csv/Diffusion_MLMR_hyperparam.csv -o
results/Diffusion_MLMR_hyperparam
```

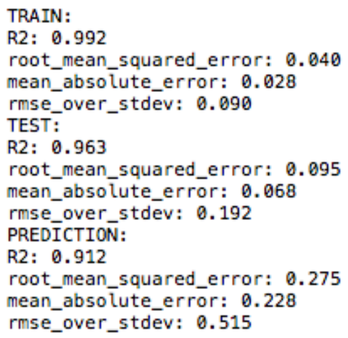
Once the optimization has finished, you will see the optimized parameters printed out to your screen, and saved in the final generation folder with the title "OPTIMIZED_PARAMS". You can also view the associated README file to see the average RMSE values of the top individuals for a given generation and the associated set of parameters used to obtain the RMSE. Note that for each individual, a given number of random K-fold splits are performed as denoted in the input file.

4 Competition! Predicting Diffusion for the Pt host material

Now that we've worked with the diffusion dataset a bit let's try to do something fun. Using the ideas presented above let's try to fit the best model that we can on the data set we have and predict a new host's migration barriers. Pick a model type and CV scheme from above and try to find the best set of hyperparameters you can. Once you've decided on a model, predict the migration values for the new dataset and report your RMSE error. We'll compare across the group and see whose model performs the best!

4.1 Predict a new host's diffusion!

To do this, run the same command you used in your initial exploration of several models, but with a few key differences: in your Diffusion_MLMR.conf input file, be sure to uncomment the line "validation_column = predict_Pt". The other is to specify the model hyperparameters you want to use based on your optimization performed previously. To do this edit the respective lines under the [models] section of the input file. This will make it so that the Pt data are never used in the train/test splits in CV, and instead are used solely as a prediction data set. When you run your favorite model with this setting, you'll notice that for each CV split, in addition to train.csv and test.csv files being generated, there is also a prediction.csv, and the associated average predicted statistics for the Pt host will be added to the stats.txt file for each CV scheme performed. The stats.txt file should now have output that looks like this:


```
TRAIN:
R2: 0.992
root_mean_squared_error: 0.040
mean_absolute_error: 0.028
rmse_over_stdev: 0.090
TEST:
R2: 0.963
root_mean_squared_error: 0.095
mean_absolute_error: 0.068
rmse_over_stdev: 0.192
PREDICTION:
R2: 0.912
root_mean_squared_error: 0.275
mean_absolute_error: 0.228
rmse_over_stdev: 0.515
```

Here, the R^2 and RMSE values under "PREDICTION" are those of predicting the Pt host migration barriers from your model. See how well you can do!