Practical 1

Samuel Daulton, Taylor Killian, Andrew Petschek (ML Marauders)

February 9, 2016

1 Technical Approach

1.1 Initial Attempts

Our initial approach to expand on the sample code provided to us for this machine learning problem was to implement Principal Component Analysis (PCA) to gain a better understanding of the interdependencies between features and to effectively reduce our feature set. We calculated the top thirty to sixty components, and compared which number of principal components appeared to be best from the start. As stated in our results section below, PCA led to satisfactory reduction of RMSE.

We also performed some basic feature engineering upfront as well to couple with the PCA. There is a hypothesis that the number of branches influences the gap between HOMO and LUMO levels, and we pulled relevant compounds that were highlighted in a theoretical study on this topic conducted by Padtaya Pukpayat of the University of Nevada Las Vegas. This specific feature aggregates the number of active features, meaning if there are more features that are active in a compound, then there may be a higher orbital overlap and, thus, a lower HOMO-LUMO gap. A final initial feature we used was the presence of a benzene ring in the compound using the SMILES encoding; we could identify a benzene ring which are more conjugated (i.e., closer together in energy).

We combined this initial feature generation with PCA, and turned to the existing methods to see if it improved baseline. The results were promising, but we decided additional features needed to be engineered for full impact

1.2 Feature Engineering

To create new features, we turned to the RDkit. Specifically, we leveraged methods that process molecules objects and return integers. We felt integer outputs would work best

with our modeling efforts and avoided outputs that non-integerse (e.g., strings and vectors). 114 new features were created that spanned four main areas: general feature generation (e.g., number of branches, number of bonds), descriptors (e.g., molecular weight, number of valence electrons), calculation of Lipinksi parameters, and fragment descriptors (e.g., number of aliphatic carboxylic acids). The methods that generated these features act upon molecule objects, which we created using the RDKit API. Molecule objects take up a lot of memory, so our technical approach to feature generation was to divide the processing into four parts, which were ultimately merged together at the end.

1.3 Modeling Techniques

We not only built off of the simple linear regression and random forest model classes provided to us, but also approached this machine learning problem from other modeling angles as well.

To prevent overfitting on out-of-sample data sets when evaluating various hyperparameters for estimators, we split our training set into a training set and a validation set (67-33 split). This way, we avoided any knowledge of the test set leaking into our models and hurting their ability to generalize to new data. We had initially contemplated a cross-validation loop, where the creation of discrete training and validation sets would no longer be necessary. However, since we had roughly one million samples to begin with, we decided we would have sufficient data to carry out our modeling. using discrete training and validation.

When estimating hyperparamters for each of our models below, we employed a grid search. Since we do not know which values of hyperparameters may minimize our loss function across models, we run each model several times using an array of hyperparameter values ranging from small to large. At the end of this loop, we selected the hyperparameter value(s) that yielded a minimum loss.

Model	RMSE
Random Forest	0.2
Linear Regression	0.3

Table 1: Model Results

1.4 Lasso

1.5 ExtraTrees

1.6 Adaboost with ExtraTrees

1.7 Steepest Descent

2 Results

3 Discussion