**Modeling Update, Weeks 3 / 4**

**Nils Persson**

*Feature Selection*

Random Forest Regression tended to perform well at prediction of ∆H, with MAE of 9.1 ± 2.3 kJ/mole. However, it was unclear which Magpie features were most important. The feature importances returned by the RF regressor had very high uncertainty, and were very low in absolute magnitude.

In Raccuglia *et al.*, (materials discovery using failed experiments), features were selected using Correlation-based Feature Selection (CFS), a filtering algorithm whose objective is to yield a set of features with high correlation to the target variable, but low correlation with each other.1 The algorithm is outlined in the thesis of Mark Hall and implemented in WEKA, a Java-based machine learning library.2 There is no Python implementation available for this algorithm; for now we will use the tools available in scikit-learn.

Scikit-learn offers an assortment of test- and model-based feature selection algorithms. Model-based selection is intuitive and straightforward: your model of choice is fit using subsets of your features, and the subset is scored based on the validation accuracy of the fitted model. Unfortunately, exhaustively testing all combinations of features scales factorially (?) in computation time. There is also the disadvantage that a model must be chosen; we will look at Random Forest, because it performed well enough (within margin of error of bagging / boosting regressors), and is a popular option that deserves more scrutiny.

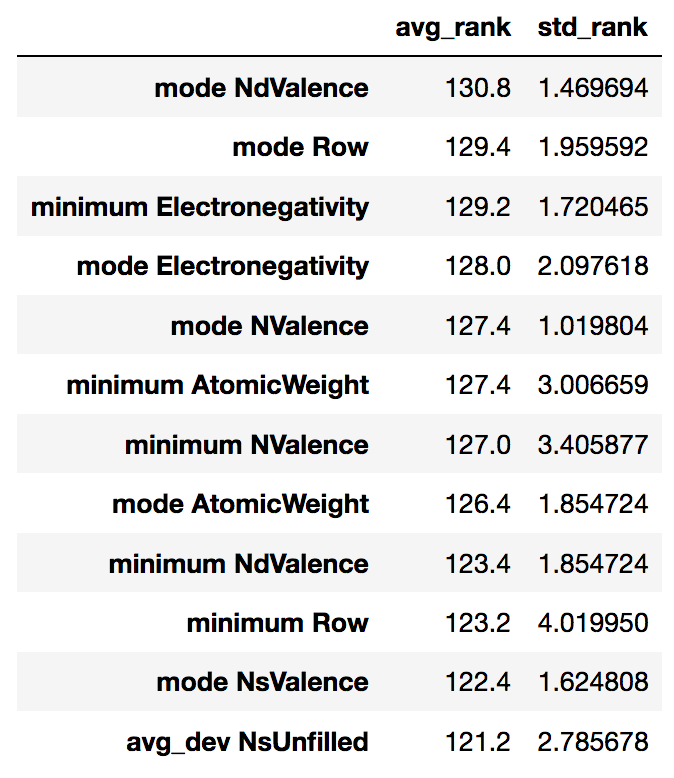
Recursive feature elimination (RFE) offers a balance between computation time and feature set exploration. First, the model is fit using all the features. Then, features are ranked according to the feature importance scores returned by the RF regressor. Even if it’s difficult to identify the *most* important feature because of the large error bars on the importances, we can at least be confident in throwing away the absolute *worst* feature. So, the least important feature is eliminated from the feature set, and the model is re-fit with *n*–1 features. MAE is tracked as features are eliminated, until only one is left.

Just doing the RFE process once may not be very robust; so scikit-learn offers an RFE-CV method that repeats the RFE process over 5 CV splits. I had to re-write their RFECV class to get out the raw results – its default is to try to pick the best number of features for you. From the raw results, we can track MAE as a function of the number of features in the feature set, plotted below.

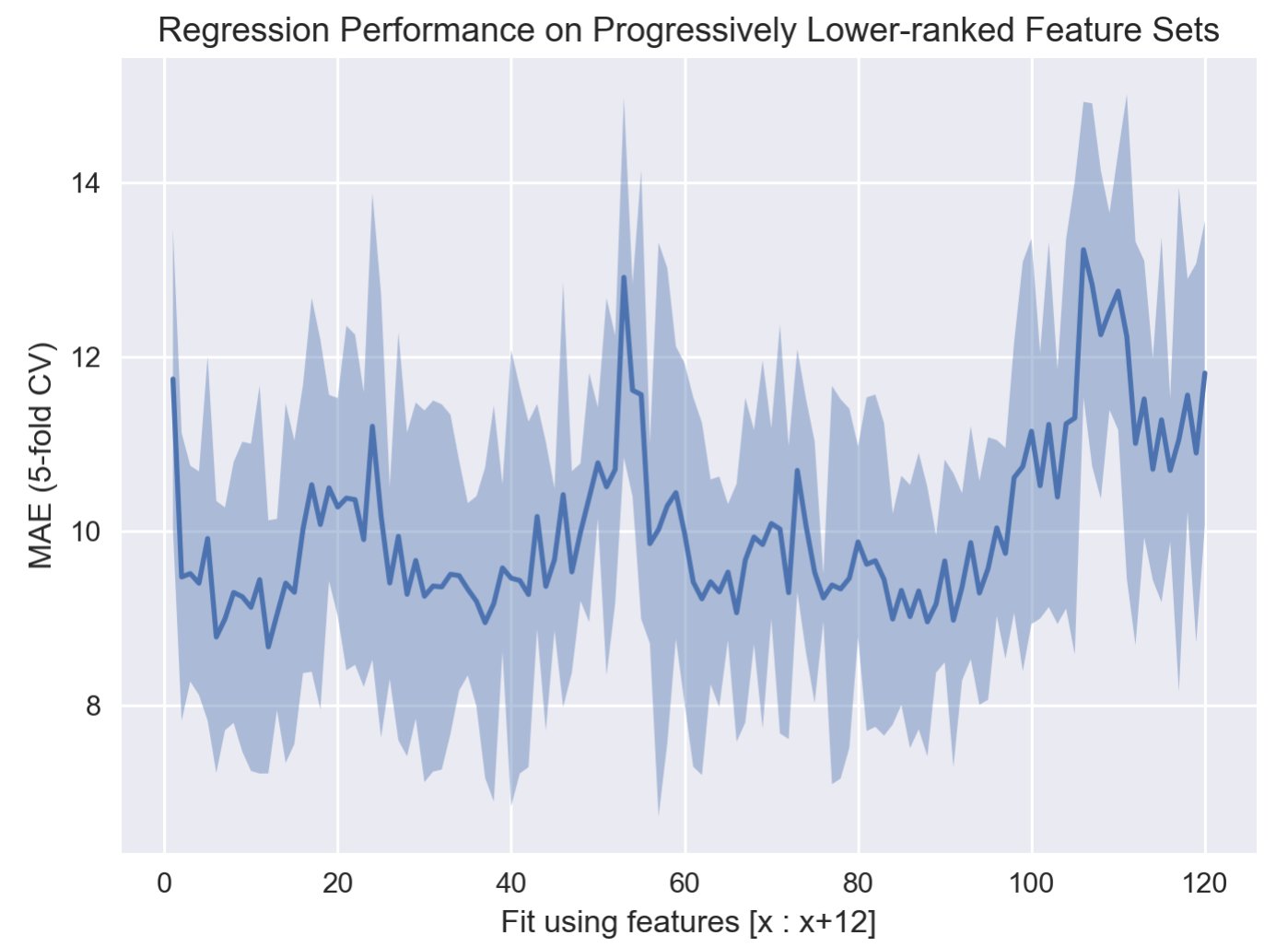


RFE wants a scoring function where higher values are better, so we track the negative MAE. The law of diminishing returns kicks in around 12 features – so we only need 12 features to train a robust RF regressor. But the real question is: which 12 features?

We can rank features by the order they were eliminated from RFE. Since we performed RFE 5 times, we have 5 separate ranking lists, which we can average to get some statistics on the feature ranking. Here is a list of the top 12 features by how long they survived the RFE process (higher is better):



Compared to the raw feature importance scores, these rankings seem to be more robust. However, this feature elimination strategy is a path-dependent process, which probably affects the results. So the next logical question is: does it really matter which 12 features we select? Below is plot of the 5-fold cross-validated MAE from fitting the RF regressor on sets of 12 features, starting with top-ranked features 1:12, then 2:13, then 3:14, and so on.



There are a couple notable features of this plot. First, the top twelve features (the first point) are substantially worse than features **1**:13 (the second point) (lower is better here, we’re back to positive MAE). That’s a bit odd, but the next few feature sets are more stable, and it appears that performance even continues to improve until the set containing features **12**:24. There are some spikes where training gets worse, but even out to features **90**:102, similar performance can be achieved. Perhaps somewhat encouraging is that models fit with feature sets ranked 100 or worse yield definitively higher MAE than the top-ranked features.

To me, this is another point in favor of “random forest is just behaving as a nearest-neighbor classifier”. I’d like to try using the Correlation-based Feature Selection algorithm in WEKA, just to see which features it selects. It also may not be completely infeasible to fit every combination of features on CTCMS over a few days. If we only consider sets of up to 12, it’s probably tractable. Maybe there’s a set of 5 or so that work really well together.