Practical 1: Predicting the Efficiency of Organic Photovoltaics

Elbert Gong, Jennifer Hu, & Harrison Li (Team HEJ Fund)

February 10, 2017

1 Technical Approach

We began by creating a function for k-fold cross validation to quantitatively evaluate the performance of the various models and minimize the risk of overfitting. This meant first dividing the training dataset into k equally sized parts (folds) using the KFold tool in the Python scikit-learn package. Then we successively fit each model on the k possible subsets of $k-1$ folds and computed the RMSE on the remaining fold.Models with the lowest mean RMSE averaged over the k folds were deemed most desirable.

1.1 Linear, Ridge, & Lasso Regression

We began by exploring basic variants on simple linear regression, specifically the ridge and lasso regularization methods. The regularization parameter (called α in the scikit-learn packages) was tuned using 5-fold cross validation over the set $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4\}$ where larger values correspond to increased regularization. Regularization tends to decrease model variance at the cost of some bias, to reduce the tendency of ordinary linear regression to diminish the risk of overfitting, which is a concern here due to the large number of features albeit somewhat mitigated by the large quantity of data.

1.2 Random Forest

To improve upon the baseline random forest model, we used 5-fold cross validation to tune the parameter known as "n_estimators" in scikit-learn. This parameter controls the number of trees in the random forest. We also planned to tune the number of parameters used at each decision node, but did not due to time constraints. We also tried bootstrap aggregating (or "bagging") which means that the trees are trained on various bootstrapped versions of the training set. Bootstrapping is essentially just resampling with replacement, and produces new data to give a non-parametric estimate the distribution of a sample.

1.3 Data preprocessing

We tried some basic data preprocessing techniques, such as standardizing the features to all have standard deviation 1, since this is an assumption in the derivation of ridge regression from the minimization of the standard regularized loss function $L(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^\mathsf{T}(\boldsymbol{y} - \boldsymbol{x}\boldsymbol{w}) + \lambda \boldsymbol{w}^\mathsf{T}\boldsymbol{w}$, however this actually worsened the RMSE. We also considered principal component analysis to reduce the dimensionality of the features, but this significantly increased RMSE as well, perhaps because we are working with large number of complex molecules that can only be adequately described using a relatively large number of features.

1.4 Feature Engineering

With all the different types of models above, we were unable to attain any meaningful improvement over the basic untuned random forest model. Therefore, we decided to harness the full power of RDKit in order to generate meaningful features in our machine learning process. After consulting some of our classmates with greater knowledge of chemistry, we decided to test out a variety of molecular fingerprints using RDKit.

The first method we tried was topological fingerprinting, which generates a fingerprint (i.e. bit vector) based on topological paths along bonds in the molecule. Next, we tried generating MAACS keys, which are also represented as bit vectors and create a unique representation of a molecule. Finally, we tried Morgan (circular) fingerprinting, which applies the Morgan algorithm to atom invariants and also produces bit vectors.

We were also interested in fingerprinting methods such as atom pairs and molecular fragmenting, but there were errors in the RDKit code that prevented us from making use of these potentially useful resources.

2 Results

2.1 Feature Selection

The first thing we did was try to find the best parameters for the problem. Using the different methods described in the previous section, we ran several models on the original features given in the problem, Morgan fingerprints, topological fingerprints, MACCS keys, and a combination of the original features and Morgan fingerprints.

Note that none of our models in this first step were tuned; we simply wanted to get a sense of how well different fingerprinting methods in RDKit would work in helping us predict the gap values.

	Linear	**Ridge**	**Lasso**	**Random Forest**	**Bagging**
Original	0.29863	0.29838	0.40823	0.27680	0.27697
Morgan	0.19150	0.18150	0.40823	0.14583	0.14569
Top	$5.6052 * 10^8$	0.35768	0.40763	0.34341	0.34203
MACCS	0.28341	0.28323	0.40763	0.27546	0.27751
Combo	0.18280	0.18280	0.40823	0.14096	0.14054

Table 1: RMSE values for various models run on various molecular fingerprints. Each model was trained on 50,000 data points.

The first thing we noticed was that the topological fingerprints performed terribly under the linear model and consistently poorly for all the others. Out of the three "new" fingerprinting methods we tried (Morgan, Top, and MACCS), Morgan had the best performance under each model. However, we observed the best results on the combination of original and Morgan features, giving us 512 features in total.

With this knowledge in mind, we proceeded to use the combination features in order to select the best model.

2.2 Model Selection

Next, we trained the linear, ridge, lasso, random forest, and bagging models on the combination features generated for the entire training dataset (1,000,000 observations). Note that these models were also initially untuned.

We observed that lasso regression has by far the poorest performance, while random forest and bagging do quite well. Our next step is to determine the best hyperparameters for tuning the random forest, bagging, and ridge models.

2.2.1 Tuning the Models

In the interest of time (and our personal computing power), we only tuned the models using 50,000 training data points. We were not interested in getting precise error values, but rather finding the hyperparameters

Model	**RMSE**
Linear	0.18175
Ridge	0.18175
Lasso	0.40718
Random Forest	0.07903
Bagging	0.07927

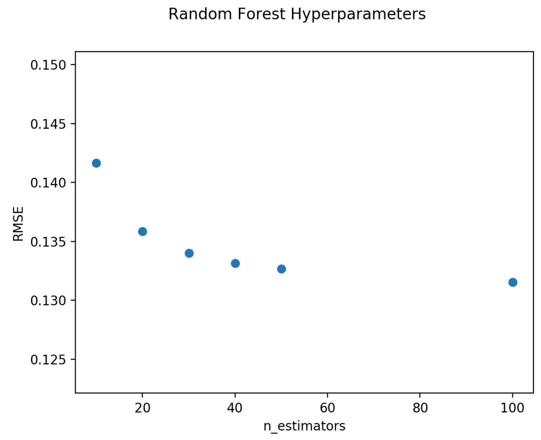
Table 2: RMSE values for various untuned models run on the combination of original and Morgan fingerprints (512 features overall). Each model was trained on the entire dataset, and results.

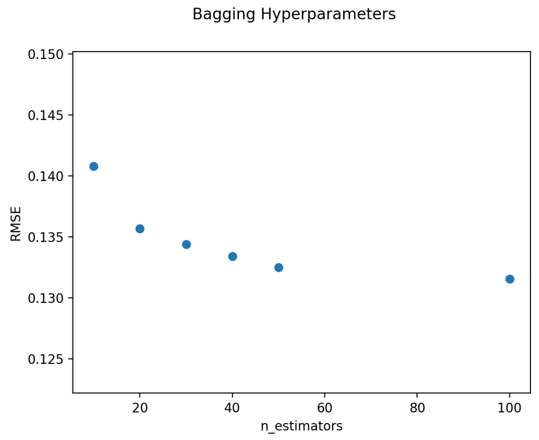
that would give the best relative performance when used on the test data.

To tune random forest and bagging, we computed the RMSEs resulting from hyperparameter

$$n \in \{10, 20, 30, 40, 50, 100\}.$$

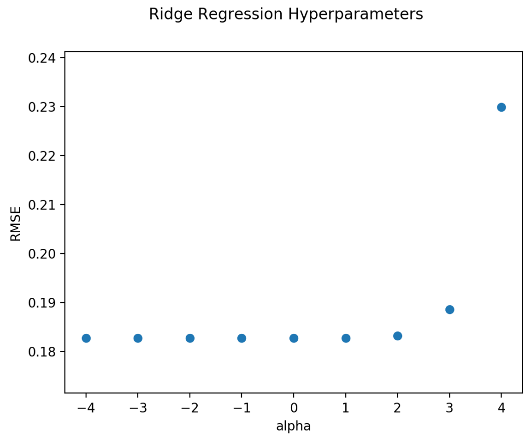
We see a clear negative correlation: as n increases, the error decreases.





To tune ridge regression, we computed the RMSEs resulting from hyperparameter

$$\alpha \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}\}.$$



Here, we see that the error increases as α increases.

2.3 Selecting Final Model

Model	**Hyperparameter**	**RMSE**
Random Forest	$n = 100$	0.13154
Bagging	$n = 100$	0.13156
Ridge	$\alpha = 0$	0.18280

Table 3: RMSE values for various tuned models (using the optimal hyperparameters determined above) run on the combination of original and Morgan fingerprints (512 features overall). Each model was trained on the entire dataset, and results.

Of the three tuned models, random forest has the lowest RMSE, albeit it is essentially indistinguishable from the RMSE for bagging. Since the optimal value of n was not an interior solution in the parameter space tested, (with a monotonically decreasing RMSE with increases in n), we decided to use the random forest model with hyperparameter $n = 150$ for our final model, though it is clear that the sensitivity to this hyperparameter is not substantial. Given more time and/or computing resources, we would also tune the ”max-features” parameter, which controls the number of features considered at each node when determining the desired split.

3 Discussion

Our initial approach was to build more complex and refined models on the data given, since naively it seemed that it would be possible to see significant improvement in model generalized accuracy by iterating upon the very simple baseline linear regression and random forest models. However, when it became apparent that the variants and tuning of these methods described above did not lead to substantial improvement in the k-fold validated RMSE, we decided that exploring different features might be another approach, which led us to explore using the various different types of molecular fingerprints indicated above.

Using the Morgan fingerprints as features led to a marked improvement in the quality of even the baseline models relative to any of the models trained using the original features. The tuning on the random forest model is just figurative ”icing on the cake,” given the modest improvement in model performance from tuning as compared to the improvement from adding the Morgan fingerprints as features. This is in line with the concept of ”garbage in, garbage out”: a model’s effectiveness is limited by the quality of its inputs.

Another interesting result we noticed is that the regularized regression methods offered negligible improvement over standard linear regression. This suggests that linear models are underfitting rather than overfitting the data, since regularization is designed to provide more “flexibility” into the model by decreasing the variance at the expense of introducing some bias.