Practical 1: Predicting the Efficiency of Organic Photovoltaics

Team 37: CP007
Xinyuan Wang wang_xinyuan@g.harvard.edu
Rui Zhao ruz466@g.harvard.edu
Yijun Zhou yijun_zhou@g.harvard.edu

1 Technical Approach

1.1 Model Selection:

During the process of model exploration, we tried several methods, including Ridge, Lasso, Elastic Net, Neural Network, Decision Tree, and Gradient Boosting (Ensemble method).

For comparison, we used RMSE, the sample standard deviation of the differences between predicted values and observed value, and R^2 score, a measure of how close the data are to the fitted model prediction. We randomly split the original training set into 60/40; 60% is the new training set, and the remaining 40% is the new testing set. In the following, all models are trained on the new training set, while the R^2 or RMSE score are calculated based on the new test set.

- Ridge/Lasso Regression:

  Ridge regression and Lasso regression both apply regularization to the simple linear regression with L2-norm and L1-norm respectively. Regularization improves the conditioning of the problem and reduces the variance of the estimates. We tuned the hyperparameter with 5-fold cross-validation and α from 10^{-10} to 10^{10} to determine an appropriate regularization penalty. At α = 0.1, Ridge has the best R^2 score 0.46159 and RMSE 0.29927, while Lasso has the best R^2 0.46159 and RMSE 0.40730. For this part, we used the Ridge/Lasso package in sklearn.linear_model.

- Elastic Net:

  Elastic Net is a linear regression combining both L1 and L2 priors as a regularizer. We tuned the parameter l1-ratio from 10^{-10} to 10^{-1} with 5-fold cross validation on the training set. l1-ratio is a float between 0 and 1, which is the scaling between L1 and L2 penalties. At l1-ratio=10^{-10}, Elastic Net model has the best R^2 score 0.45271 and RSME score 0.30168. For this part, we used sklearn.linear_model.ElasticNetCV.

- Neural Network:

  Neural Network consists of multiple layers with several nodes on each layer. For each node or unit it has a threshold function to determine if an input data could pass or not. It is a supervised learning algorithm that learns a function $f(\cdot) : R^m \rightarrow R^o$ by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features $X = x_1, x_2, ..., x_m$ and a target y, it can learn a non-linear

function approximator for regression. Here we set the parameter solver to be lbfgs, which is an optimizer in the family of quasi-Newton methods. The R^2 score is 0.53928 and RMSE is 0.27646. For this part we used the package sklearn.neural_network.MLPRegressor.

- Decision Tree:

  The basic idea of Decision Tree is to segment the predictor space into sub-regions and to use the mean (or mode, or median) of the respond variable of the training examples in the segment. To build a basic regression tree, we divided the predictor space into J distinct but not overlapping regions R_1, R_2, R_3,...,R_J. We needed to find boxes R_1, R_2, R_3,...,R_J that minimize the $RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}R_j)^2$ where $\hat{y}R_j$ is the mean response value of all training observations in the R_j region.

  However, this method is high computationally expensive. We might need to use Recursive Binary Splitting algorithm and Pruning algorithm to reduce the computation cost. Here we used the DecisionTreeRegressor package in sklearn.tree. We tuned the parameter max-depth in order to prevent underfitting and overfitting. We found out that at max-depth=44, the Decision Tree has the best R^2 score 0.55155 and RSME score 0.27261.

- Ensemble Methods (Gradient Boosting):

  Gradient Boosting(GB) produces a prediction model in the form of an ensemble of comparatively weak prediction models, typically Decision Trees. GB builds the model in a forward stage-wise fashion, and allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function. The learning rate (which shrinks the contribution of each tree) and *n_estimators* (the number of boosting stages to perform) are set to be 0.1 and 50. We tuned the *max_depth* (maximum depth of the individual regression estimators) from 1 to 9 to guarantee the effectiveness and efficiency. As the result shows, the highest R^2 appears at the point when *max_depth* equals 8, which gives the best R^2 of 0.54978, and RSME of 0.27464.

1.2 Feature Engineering:

After the exploration on regression technicals, we started trying feature engineering to see whether it helps to improve the model performance.

The original number of features is 256; if all the training data have the same value on a certain feature, this feature is useless. We eliminated all the features that contain the same numbers, with only 31 useful features left. This helps decrease the running time of our model.

After filtering existing features, we decided to add more features using RDKit package. We began with looking at the real meaning of y value - the difference in energy between the highest occupied molecular orbital (HOMO) and the lowest unoccupied molecular orbital (LUMO). We found a TPSA function in RDKit package which creates the Topological Polar surface area of a molecular structure and thus differentiates the energy difference between two orbitals. Similarly, we generated new features using ValenceElectron, MaxPartialCharge and MolLogP from Discriptors package in RDKit. Those features are all numerical and good indicators for molecule's chemical properties. We also considered bond types and Radical Electron numbers, but those two are not good since they do not provide any distinctions among different entries in the training data set.

Model	RMSE
BASELINE 1 - LINEAR REGRESSION	0.29957
BASELINE 2 - RANDOM FOREST	0.27242
RIDGE	0.29927
LASSO	0.40730
ELASTIC NET	0.30168
NEURAL NETWORK	0.27646
DECISION TREE	0.27206
ENSEMBLE	0.27464

Table 1: Regression models with corresponding RMSE scores.

Decision Tree Model	RMSE
Existing features	0.27206
Existing features + TPSA	0.25684
Existing features + ValenceElectron	0.19456
Existing features + MaxPartialCharge	0.22652
Existing features + MologP	0.18457
Existing features + TPSA + ValenceElectron + MaxPartialCharge + MologP	0.16821

Table 2: Decision Tree Models with modified feature lists and the corresponding RMSE scores.

After getting the potential candidates of new features, we did some experiments to see their effect on the current Decision Tree model. We started by adding one new feature at a time to eliminate the interference of other new features. As we can see from Table 2, adding single feature has already improved RMSE score from 0.27206 to about 0.19. Combining all the four features together, the RMSE score was down to 0.16821. This is already a huge improvement from before, so we decided that this would be our final model.

2 Results

We have created and submitted a set of predictions on kaggle. After trying both model selection and feature engineering, our final method gives reasonably good performance. The tunning plots of max-depth w.r.t R^2 and RMSE for Decision Tree are shown in Figure 1, with the most appropriate max-depth happening at 44. The summary of our results can be seen in Table 1 and Table 2.

3 Discussion

We started with Ridge Regression. Ridge regression is similar to simple linear regression with a tuning parameter (λ) to control the regression coefficient. The added term λ is shrinkage penalty with the power to shrink coefficients. We believe this would beat simple linear regression, for by choosing λ we are actually doing a trade-off between bias and variance, and cross validation could help us find the best λ. The RMSE on our test set is 0.29927. Next, we considered trying Lasso regression. Ridge regression has one disadvantage that none of coefficients will be set to zero. Some issues would arise when the original number of features is big. Lasso overcomes this disadvantage by changing Ridges L2 penalty to L1 penalty. The RMSE of Lasso is 0.40730. It seems that both Ridge and Lasso do not have much improvement on the RMSE score. So we tried Elastic

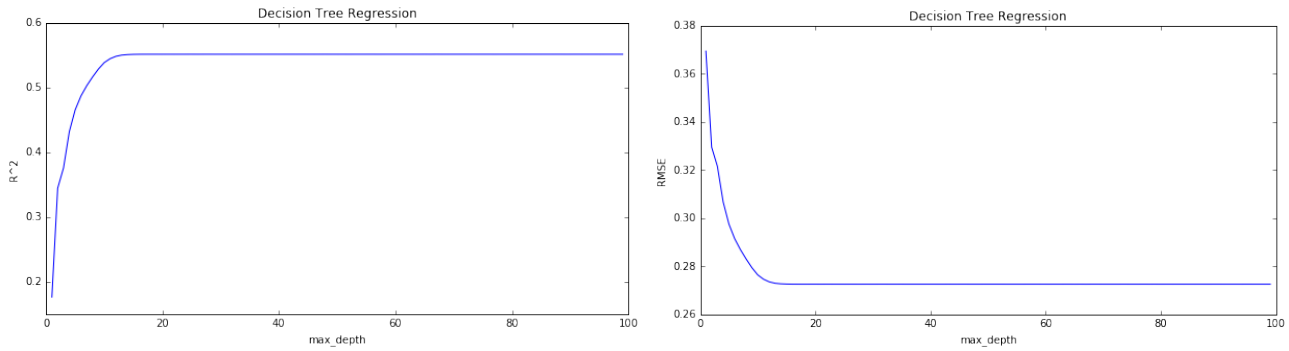

Figure 1: Tuning of Decision Tree with best RMSE and R^2 score at max_depth = 44

Net afterwards, which uses both L1-norm and L2-norm term and might improve the performance by overcoming the limitation of both regressions. The RMSE score for Elastic Net model is 0.30168.

All the methods we tried above are not very helpful in reducing RMSE score. This might due to the non-linear relationship among existing features, so we started trying out other complicated models.

We started with Neural Network model. This model is usually a good choice when the model is non-linear or when it is difficult to find the pattern between inputs and outputs. However, Neural Network is hard to interpret, and did not get our expected result in the experiment, so we decided to try Decision Tree model with parameter tuning on the tree depth. We implemented Decision Tree model, and got a RMSE score of 0.27206, which was the lowest RMSE score until then. We did a further exploration with ensemble method, hoped to see a better improvement, but the RMSE score was down back to 0.30168.

Among all the regression models, the Decision Tree model has the best performance and is also easy to interpret. By cross validation we got the best max-depth at 44 and started heading to another direction - feature engineering, to see if it will largely improve the final result.

We first checked to see if we could throw away redundant features; after filtering, only 31 features left. We used RDKit package with our chemical knowledge on the purpose of adding more useful features. Since we need to predict the energy difference between HOMO and LUMO of a molecule, we focused on finding descriptors that can well represent molecular activity. We finally selected four new features that could be computed by RDKit: the topological polar surface area(TPSA), the valence electron, the max partial charge, and the Wildman-Crippen LogP value(MolLogP) of a molecule. See result in Table 2. We ran the final decision model with total 35 features, and got the RMSE score as 0.16821. Not surprisingly, the modified feature list achieved the lowest RMSE among all of the experiments, and successfully beat the simple linear regression and random forest model.

In short, we explored several regression models at first, and found out that the Decision Tree model with a depth of 44 got the best result. Then we tried feature engineering. Filtering existing features and adding some new features improved our model in a great extent. The combination of feature engineering and regression technique engineering helped us finally achieve our goal. We got an RMSE score of 0.15462 on the actual test data set after a submission to Kaggle.