CS181 - Practical 1

[Team "J.Q.G"] Jing Wen, Qing Zhao, Jeewon (Grace) Hwang

1 Technical Approach

In this practical, we are trying to predict a continuous variable 'gap' with a series of binary predictors. To tackle this problem, we focus on linear regression methods.

1) Features Engineering

The feature 'SMILES' is the representation of chemical formulae. Since HOMO-LUMO stand for Molecular Orbitals, we conjectured that the 'gap' value should be strongly correlated with the type of molecules appeared on the 'SMILES' string. Therefore, we embarked to add new features based on 'SMILES' string. Followings are the list of features we add in addition to existed 256 features:

1. *carbon*: number of **c** or **C** in 'SMILES'
2. *oxygen*: number of **o** in 'SMILES'
3. *nitrogen*: number of **n** in 'SMILES'
4. *sulfur*: number of **s** in 'SMILES'
5. *hydrogen*: number of **H** in 'SMILES'
6. *[]*: number of **[]** in 'SMILES'
7. *()*: number of **()** in 'SMILES'
8. *onelink*: number of in 'SMILES'
9. *doublelink*: number of **=** in 'SMILES'
10. *1 : number of chemical-bonds that is on 1st Valence electron of its atom*
11. *2 : number of chemical-bonds that is on 2nd Valence electron of its atom*
12. *3 : number of chemical-bonds that is on 3rd Valence electron of its atom*
13. *4 : number of chemical-bonds that is on 4th Valence electron of its atom*
14. *5 : number of chemical-bonds that is on 5th Valence electron of its atom*
15. *6 : number of chemical-bonds that is on 6th Valence electron of its atom*
16. *p_XXX : percentage of atom 'XXX' among the SMILES string*

2) Feature Reduction / Regularization

A row of molecular data is now associated with more than 256 different features. However, not all the features are important to predict the HOMO-LUMO gap. In addition, having too many features make the data sparse, and hence we need to tackle the curse of dimensionality problem. More importantly, we may encounter over-fitting problems by including too many features. Therefore we want to reduce the number of features in our regression models.

There are two approaches we applied here to achieve feature reduction: PCA and Pearsonr. In terms of PCA, it uses Singular Value Decomposition of the data and keeps only the most significant singular vectors to project the data to a lower dimensional space. As for Pearsonr, we select features based on their correlation with the HOMO-LUMO gap.

Another way to prevent our models from overfitting, we leverage the inherent properties of regression models, such as Ridge Regression and Lasso Regression. Both methods impose a penalty on the size of the coefficients and likely set the coefficient of unimportant features to be zero. Therefore, we had tried both PCA and Pearsonr, and to regularize our regression, we had tried both Ridge, and LASSO regression.

In addition, after we reduced the number of features, we also tried to use interaction terms as new features. The result will be discussed in the Result part.

We have plotted the distribution of the HOMO-LUMO gap. The distribution looks quite symmetric, which meets the normality assumption of linear regression model. Thus, we do not need to transform our predicted values.

3) Model Selection

To address the over-fitting problem, we divided the given training dataset additionally into training and validation datasets. Then, for each model, we apply **k-fold (we usually tried 5-fold) cross-validation** using the new training data. In addition, for models that require extra tuning, we iterated cross-validations through different values of parameters and select the parameters that gives the best score. **For example, for Ridge Regression, we iterated different values of α, the size of the penalty, i.e. $"alpha" = [0.001, 0.01, 0.1, 1.0, 10.0, 100.0]$, and we then chose the alpha that gives us the smallest mean squared error.** So our procedure is as follows:

1. Create a list of parameter values.
2. Apply cross-validation to models using different parameter values one by one.
3. Select the parameter that gives the best result, fit the model with the whole training data-set, and compute the accuracy score of the validation data-set.
4. Compare the accuracy score of the training data set and validation data set to rule out over-fitting or underfitting problems.

To compare different models (In this paper we mainly discussed Ridge, Lasso, Random Forest, and AdaBoost), we looked at the accuracy score of the validation data-set.

2 Results

After we added more features, we observed a substantial boost in the accuracy score as summarized in Table 1, where we added the first five new features listed in part 1.

PCA

Table 1: Baseline Results

	256 Features		256Features+ New Features	
Score	Validation	Kaggle	Validation	Kaggle
Linear Regression	0.460285	0.29846	0.6093	
Random Forest	0.548676	0.27207	0.75536	0.18965

We used PCA to keep only the most significant singular vectors, and projected the data to a lower dimensional space. By examining the percentage of variance explained by each of the selected components, we were able to decide how many features we want to keep for the following training procedure. Below are top 40 components with highest explanation ratio.

[5.68581186e-01 1.77050806e-01 5.76816007e-02 4.13911982e-02
2.82745347e-02 1.83094374e-02 1.65082420e-02 1.19341573e-02
1.13207224e-02 9.48347042e-03 7.85560842e-03 7.32021954e-03
5.41441169e-03 5.11061951e-03 4.84577398e-03 4.12546966e-03
3.92736580e-03 2.95616996e-03 2.69813713e-03 2.30524904e-03
2.10304439e-03 1.56127987e-03 1.50222682e-03 1.30601320e-03
1.26986745e-03 1.15425030e-03 8.92212762e-04 8.86720271e-04
7.38812293e-04 4.62408936e-04 3.69421124e-04 2.40237153e-04
1.79098233e-04 1.34223552e-04 5.87486332e-05 4.57689377e-05
7.92476188e-07 4.93998764e-07 5.71305393e-30 1.12132867e-31]

As we can see from above, the top 36 components contribute to over 99.9999% of variances. It is reasonable to perform the **fit_transform** operation on training set and reduce the number of features to 36.

Pearsonr

As discussed in 'Technical Approach' Part, we also reduced the number of features by looking at the correlation of features with the predicted values. Ranking features by the absolute value of their correlation coefficients, we selected features that have coefficient values greater than 0.02. Therefore, 28 features are selected.

Then, we tried out different models with reduced features. Results are summarized in Table 2&3.

Ridge Regression and Lasso Regression

In addition to apply regression models to all features, we also applied these regression models to selected features and their interaction terms. The first set of selected features is the top 28 features that has the highest correlation with predicted values (high correlation). The second set is the features that have non-zero coefficients based on the best Ridge or Lasso Regression model(Non-Zero). For each set of selected features, we create interaction terms for each pair of features and then rerun Ridge Regression and Lasso Regression model. Results are summarized in Table 2&3.

Note that since for each parameter we reran our models five times, 'Std' is the standard deviation of the MSE of the model. 'Score' is the accuracy score of the model predicting validation data set.

Random Forest

After we tried Ridge and LASSO regressor, they did not show us satisfying results (Refer the Table 2&3), which made us to explore not only generalized linear models but also ensemble models. More specifically, we adopted two methods: averaging methods, represented by **Random Forest** and boosting methods, represented by **AdaBoost**. We start with Random Forest, and Table 4 shows the result after combining newly created features.

Table 2: Ridge Regression

	Ridge (All Features)		Ridge (High Correlation)		Ridge (Non Zero)	
α	MSE	Std	MSE	Std	MSE	Std
0.001	-0.06463	0.00037	-0.4958	0.00017	-0.04529	0.00026
0.01	-0.06463	0.00037	-0.4958	0.00017	-0.04529	0.00026
0.1	-0.06463	0.00037	-0.4958	0.00017	-0.04529	0.00026
1	-0.06463	0.00037	-0.4958	0.00017	-0.04529	0.00026
10	-0.06463	0.00037	-0.4959	0.00017	-0.0453	0.00026
100	-0.06465	0.00035	-0.4963	0.00017	-0.04546	0.00026
Best α		1		1		1
Test Score		0.61037		0.70126		0.72788
Validation Score		0.60893		0.70147		0.72550

Table 3: Lasso Regression

	Lasso (All Features)		Lasso (High Correlation)		Lasso (Non Zero)	
α	MSE	Std	MSE	Std	MSE	Std
0.001	-0.06578	0.00057	-0.0540	0.00025	-0.05248	0.00036
0.01	-0.07558	0.00046	-0.0657	0.00043	-0.06291	0.00049
0.1	-0.15281	0.00084	-0.0751	0.00044	-0.07372	0.00046
1	-0.1658	0.00083	-0.1181	0.00055	-0.11975	0.00056
10	-0.16581	0.00083	-0.16581	0.00083	-0.16581	0.00083
100	-0.16581	0.00083	-0.16581	0.00083	-0.16581	0.00083
Best α		0.01		0.001		0.001
Train Score		0.60339		0.67409		0.68370
Validation Score		0.60170		0.67330		0.68263

AdaBoost

As we can see, Random Forest yields pretty satisfying results with an accuracy score staying around 0.933390, which in a way proves that ensemble models could be a potential boost to our result; meanwhile, there's still room for improvement (at least in terms of our ranking on Kaggle!). We want to further capture those trends which are difficult to predict in random forest by using boosting algorithm. More specifically, we try to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, and here we use small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. The data modifications at each so-called boosting iteration consist of applying weights w_1, w_2, ..., w_N to each of the training samples. Initially, those weights are all set to $w_i = 1/N$, so that the first step simply trains a weak learner on the original data. For each successive iteration, the sample weights are individually modified and the learning algorithm is reapplied to the re-weighted data. At a given step, those training examples that were incorrectly predicted by the boosted model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly. As iterations proceed, examples that are difficult to predict receive ever-increasing influence. Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence.

In our practice, we specified the base_estimator parameter to be Decision Tree Regressor. Furthermore, we used GridSearchCV provided by sklearn to implement cross-validation, and tune n_estimators (the number of weak learners) to get better result. Results are summarized as below.

Table 4: Random Forest + Feature Engineering

	256 Features + New Features	
Score	Validation	Kaggle
Random Forest	0.933390	0.13922

Table 5: AdaBoost Regression

	AdaBoost (All features)		AdaBoost (PCA)	
n_estimators	MSE	Std	MSE	Std
10	-0.04421	0.00263	-0.05415	0.00244
20	-0.04295	0.00293	-0.05156	0.00177
50	-0.04219	0.00290	-0.05019	0.00277
100	-0.04192	0.00298	-0.04989	0.00221
Best	**n_estimators**	Score	**n_estimators**	Score
	100	0.98129	100	0.97925

3 Discussion

Feature Engineering

At the beginning of this practical assignment, we believed that it is critical to have feature reduction to remove those unsignificant features, especially in cases where the dataset is sparse. So we tried feature reduction and applied several kinds of models. However, as we can tell from the results, reducing features using either PCA nor Pearsonr did not improve our models' performance as we expected. The reasons behind this might be we have enough number of dataset, thus keeping all these features did not lead to dimensionality curse.

Since it was hard to make significant improvement by just adjusting given binary features, we decided to create new features. We looked into the string variable 'SMILES' and created new features by counting number of atoms so that we can have more closer prediction to Molecular Orbital value. For example, we counted the number of the 'C' in the strings or the number of 'double links' in the strings.

After adding new few features, we observed an immediate increase in the accuracy score of Random Forest and Linear Regression models. The accuracy score boosted from 0.548678 to 0.933390, whereas previously no matter what models we tried to fit, the increase of accuracy score never exceeded 0.1.

Model Selection

Table 6: Model Comparisons

Models	Accuracy Score	Kaggle Score
Sample (Baseline)	0.46100	0.27207
Poly Linear Regression	0.52639	0.27962
Ridge Regression	0.60239	0.27941
Random Forest	0.93339	0.13922
Adaboost	0.97925	0.16685
Best	**Random Forest**	0.13922

Followed by the feature engineering, we tried out different regression models. Since Ridge Regression model or Lasso Regression model did not improve the accuracy score much from Linear Regression, we concluded that the relationship between predictors and predicted values may not be linear, hence it is safe to say that the Ridge and Lasso models are **under-fitting** as they missed almost 40% of the dataset. This

led us to explore other approaches like ensemble models which combine the predictions of several base estimators built with a given learning algorithm to improve generalizability and robustness over a single estimator.

Judging by the consistency in accuracy scores between training and testing data, we conclude that random forest generally produce **good fitting**. Our Kaggle score also clearly proved that, as it beats all of out other models.

AdaBoost was able to predict with very high accuracy, which is 0.97925, however, it failed to give us best solution among the several models. As we could see in Table 6, it has higher accuracy on training set, but failed to generate satisfying result with testing set on Kaggle, which made us to conclude that this is due to **over-fitting**.

To sum up, among all the models we tried out, random forest showed the best performance with regard to accuracy in both training and testing data. Meanwhile, we found that Ridge and LASSO regressions failed to capture all the trends within the dataset and were, thus, under-fitting, whereas Adaboost fits well on training set only, indicating it is over-fitting. In addition, one of our takeaways from this practical is that it is very important to include essential features in models to make better predictions.