178 Project Report

Mengchen Xu ID: 61281584
Yang Tang ID: 53979886
Kerong Sun ID:21727683

Models	Training Data	Validation Data	LeaderBoard Data
KNN	0.66313	0.64839	0.68092
RANDOM FOREST	0.73149	0.70157	0.74128
GRADIENT BOOSTING	0.735936	0.72005	0.73291
Ada Boosting	0.75344	0.68432	0.71125

- KNN

For KNeighborsClassifier, we learned from all the features at first, and we were trying to find out a better k value for a lower error rate. We compared the result when the k value is different. However, the accuracy of this method is kind of low, so we write a function to choose features by variance. The ideal is that when a feature doesn't vary much within itself, it generally has very little predictive power. Thus, first of all, we removed the features whose variance is 0. And then we ordered all the features by their variance from big to small and we chose the top 40 features. We trained the model with this new dataset and the performance improves on a small scale.

- Linear model

For the linear model, because we have a large amount of training data, we defined polynomial regression models for good performance. We set up a list of degrees=[1,3,5,7,10,15,18], and we add the all-ones constant features to the model, then we used rescale function to rescale the features. We did the same thing on the testing data to be sure that all the data are applied to the same rescaling and polynomial expansion. Moreover, we applied cross-validation to improve the performance of the model. We created a 6-fold validation test to access the validation performance by averaging them. We calculate the use and accuracy of each degree. And we picked the model with a degree of 10 which has the lowest validation error 0.249 as the best linear model.

- Random Forest

We first divide the training data into two subsets: training and validation data. We decide to try combined 50 classifiers as a begin. We repeatedly bootstrap the training data and use them to create a tree Classifier with a max depth of 15 for 50 times. We average the predictions of those decision tree classifiers and try to evaluate the MSE and AUC of training and validation data. We eventually collect 200 classifiers with a different setup at the end. Since there are only 107 features in total and some features are the same among rows, we decrease the number of the randomly chosen features to 60. We modify the minParent parameter and the maxDepth parameter for complexity control purposes since the greater minParent and smaller maxDepth would avoid overfitting. Although the bootstrapped training MSE increase and AUC decrease at the end -- it is understandable since we enforce the complexity to goes down, the validation and non-bootstrapped training MSE decrease and AUC increase. For the test data, we apply the soft

prediction and increment it through each iteration. We got a test data accuracy score of 0.74 at the end.

- Gradient Descent

For gradient boosting, we used both “Adaboost” and “GradientBoostingClassifier” functions from sklearn.ensemble class. We want to compare the differences between these two models and try to figure out which model is better. (Since our training data contains numeric features, categorical features, and binary features, we need to determine

For the gradient boosting model, we used the function “GradientBoostingClassifier” from sklearn.ensemble class. The parameters we chose for GradientBoostingClassifier are max_deph=5, loss,n_estimators. We trained the model with raw inputs. We used max_depth because from Homework 4 we learned that the max depth of the tree can make a huge difference to our prediction. We choose to use the loss as ‘deviance’ because ‘deviance’ can be used to binary classification and multi-class classification, and more than half of the values of our data are binary-valued and discrete categorical. We control the n_estimators to be 100 and we got an AUC score of 0.7359. We tried to use large n_estimators such as 1000 because large n_estimators could be better at predicting results. However, the running time for 1000 n_estimators is too large and the acc score did not improve dramatically. Thus, we finally choose n_estimators=100 because of this trade-off.

- Ada Boosting:

We used Ada Boosting also, and we test the model with different N_estimators. When N_estimators is 100, our accuracy score for training data is 0.69; when N_estimators is 1000, our accuracy score is 0.748; When N_estimators is 2000, our accuracy score is 0.753. We found that larger N_estimators will make the performance better, although the running time will be much slower.

- Conclusion Paragraph

In the end, the Random Forest method reaches the highest test data accuracy score of 0.7412. But we have tried several technologies to train learners and get good performances as well. When we train the Knn model, we use a single feature performance to select suitable features. We also try to adjust learner parameters according to our past experiences and knowledge to reduce validation error. Those practices lead us to the final result.

We could have done better in other aspects to improve performance. Applying normalization to the training data on selected features is a potential solution. Since there are 41 real-valued features and some of them have extreme outliers, normalization will reduce distances of outliers and improve the performance. But the limitation of our coding abilities shortens the scope we can reach.