Identify Fraud from Enron Email

Nov. 10 2015

Ren Zhang, zhang_ren@bentley.edu

1 Introduction

Enron Corporation is an American company which bankrupted on December 2, 2001 due to fraud and corruption. The Enron Email Dataset was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation. We are working on a processed version of the dataset provided by Udacity trying to build a classifier that can distinguish person of interest (POI for short) that is the person possibly involved in the fraud and corruption.

There are 146 persons in the dataset, 18 of them are identified as POI and the rest 128 are non POIs. For each person in the dataset we have 19 features. Some features are financial related, like a person's salary. Some features are related to the emails of that person, like to whom the person has sent an email. There are many missing values in the dataset, 128 out of 146 persons in the dataset don't have a value for the feature *restricted_stock_deferred*, and 129 out of 146 persons don't have a value for the feature *director_fees*. For a detailed list of missing values, please see Appendix 1.

One obvious outlier in this dataset is named *TOTAL*, which is rather a summary of all persons than an individual person. We can verify this by calculating the sum of salaries of all other persons in the dataset, and the result is exactly the same as the salary for *TOTAL*. So we removed *TOTAL* from our dataset, since otherwise it will be an outlier and affect the performance of our classifier. After removing *TOTAL* from our dataset, we have 145 persons left.

2 Feature Engineering

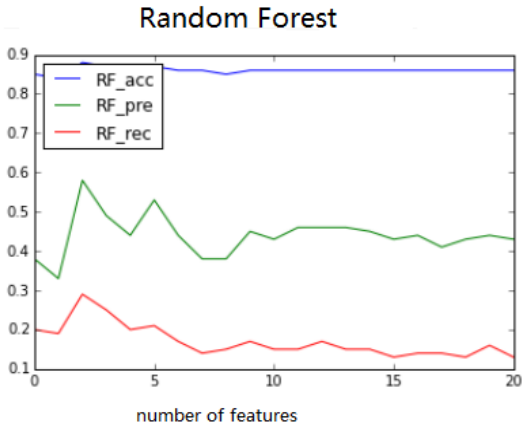
We created two new features in this project. One is called *to_poi_message_ratio*, which measures how frequently one person sent emails to POIs. And the other one is called *from_poi_message_ratio*, which measures how frequently one person received emails from POIs. After these two features are created, we have 21 features in the dataset.

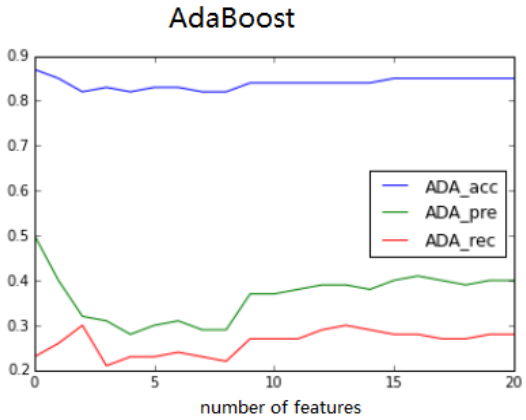
Later on we performed feature selection, and unfortunately none of these two features remained after that process.

3 Feature Selection

In this part, we performed feature selection using cross validation and to select the best number of features we should keep. We used the *f_classif* function as the scoring function for *SelectKBest* method in scikit-learn, and get the best k features for k = 1,2…21. And in each case, we built random forest

classifiers as well as adaboost classifiers and tested their performances through cross validation. The performance of the classifiers with different best k features are shown in the following plots.





Based on the plots we can see, the accuracy of both classifiers are roughly the same at different number of features used. Random Forest classifiers tend to have a higher precision score than AdaBoost classifiers. The recall scores are relatively low for both classifiers no matter how many features used. The highest recall scores are obtained for both classifiers when we use the best 3 features from *SelectKBest*.

The three features selected are *bonus*, *total_stock_value* and *exercised_stock_options*. We also obtained the importance of these three features from the random forest model fitted using these three features, and the importance scores are 0.33, 0.36 and 0.30. So there three remaining features have roughly the same importance score. The features scores of these three features using *SelectKBest* are 21.06, 24.47 and 25.10 respectively.

We didn't perform any feature scaling since the classifiers we are using are tree based classifiers that don't require feature scaling.

4 Tune an Algorithm

Since in the previous part, we noticed that the recall scores for both classifiers are relatively low. In this part we used *GridSearchCV* to find the best hyper-parameters to obtain a higher recall score for both random forest classifier and adaboost classifier. These hyper-parameters determines the complexity of the classifiers and we want get the right values in order for our classifier to perform better on future data.

For random forest classifier, we tuned the values for *max_features*, *min_sample_splits* as well as *n_estimators*. The higher values of *max_features* and *n_estimators* the higher complexity our classifier will be. And for *min_sample_splits* the higher value it is, the smaller complexity our classifier will be.

The optimal values we find with respect to recall score are: 2 for *max_features*, 2 for *min_samples_split* and 100 for *n_estimators*.

For adaboost classifier, we tuned the values for *n_estimators* and *learning_rate* and found the optimal values for them with respect to recall score are: 200 and 0.6 respectively.

5 Validate and Evaluate

Cross validation is a resampling method that we use to extract additional information about the performance of our classifiers. We used this method in feature selection part as well as in the algorithm parameter tuning part. The goal of cross validation is to measure how well the algorithm would work on out of sample cases. And it is more reliable than if we simply measure the algorithm with the sample data we have in hand. Since the number of observations in our dataset is rather small and the proportion of POIs in the dataset is relatively low, we used the *StratifiedShuffleSplit* method to do cross validation in this project. Below is a summary table of some performance measurements of our two tuned classifiers generated via cross validation.

Classifier	*Precision*	*Recall*	*Accuracy*
Random Forest	0.56201	0.39200	0.85946
AdaBoost	0.33204	0.29950	0.79954

Precision in this project measures the proportion of persons who are classified as POIs by the classifier that is indeed POIs. That is: $Precision = \frac{number\ of\ persons\ classified\ as\ POIs\ and\ are\ indeed\ POIs}{number\ of\ persons\ classified\ as\ POIs}$

Recall in this project measures the proportion of persons that are indeed POIs that is correctly identified by the classifier. That is: $Recall = \frac{number\ of\ persons\ classified\ as\ POIs\ and\ are\ indeed\ POIs}{number\ of\ persons\ that\ are\ indeed\ POIs}$

Notice that we got higher scores for both precision and recall from the tuned random forest model, thus we choose it to be the final classifier.

6 Final Remark

We originally thought this project is a bit ill proposed since the task is to predict who is a POI and some of the features given to us, like the number of emails the person sent to POIs, are the information only available when POIs are correctly identified. We thought that using these features are like cheating. However the final features selected using cross validation doesn't contain features of these types. And thus we got some confidence that the classifiers should have some usage in real life.

Appendix 1 Missing Values in the Dataset

Feature Name	Number of Missing Values
poi	0
salary	51
deferral_payments	107
total_payments	21
loan_advances	142
bonus	64
restricted_stock_deferred	128
deferred_income	97
total_stock_value	20
expenses	51
exercised_stock_options	44
other	53
long_term_incentive	80
restricted_stock	36
director_fees	129
to_messages	60
from_poi_to_this_person	60
from_messages	60
from_this_person_to_poi	60
shared_receipt_with_poi	60