Fraud detection with Enron Data

Project Summary:

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives.

The goal of this project is to build a classification algorithm that can identify, based on the Enron dataset, who at Enron is a suspected person of interest in the fraud case that happened when the company went bankrupt.

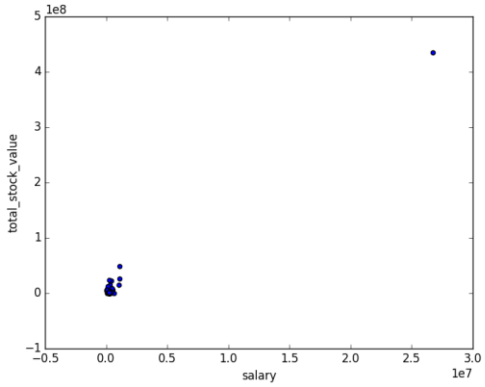
Data Exploration:

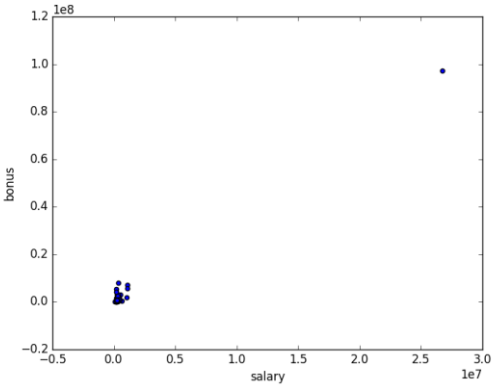
The data used in this project generally have three categories: Financial data, email data and POI which is defined as a person who has been indicted for fraud, settled with the government, or testified in exchange for immunity. 14 financial features (salary, stock ownership, bonus etc…) and 5 email features were collected for each data point.

In this dataset, there are 146 data points available and 18 of them are labeled as POI.

By checking missing values, it was found there are three features with more than 75% of missing data. For instance, local advances feature has 142 missing values and directory fees and deferral payment has 129 and 107 respectively. It's common to see missing value in data studies. However, including features with such large amount of missing values could lead the result of study to be biased. Therefore, those three features will be excluded first in feature selection process.

In order to determines whether there are any outliers or bad data in the dataset, several features including salary, bonus and total stock values were plotted. According to the plot, one data point appears to deviate markedly from other observation and it was determined this particular data point belongs to 'Total'. This outlier seems to be a spreadsheet quirk and will be removed for the rest of the test.





Feature scaling:

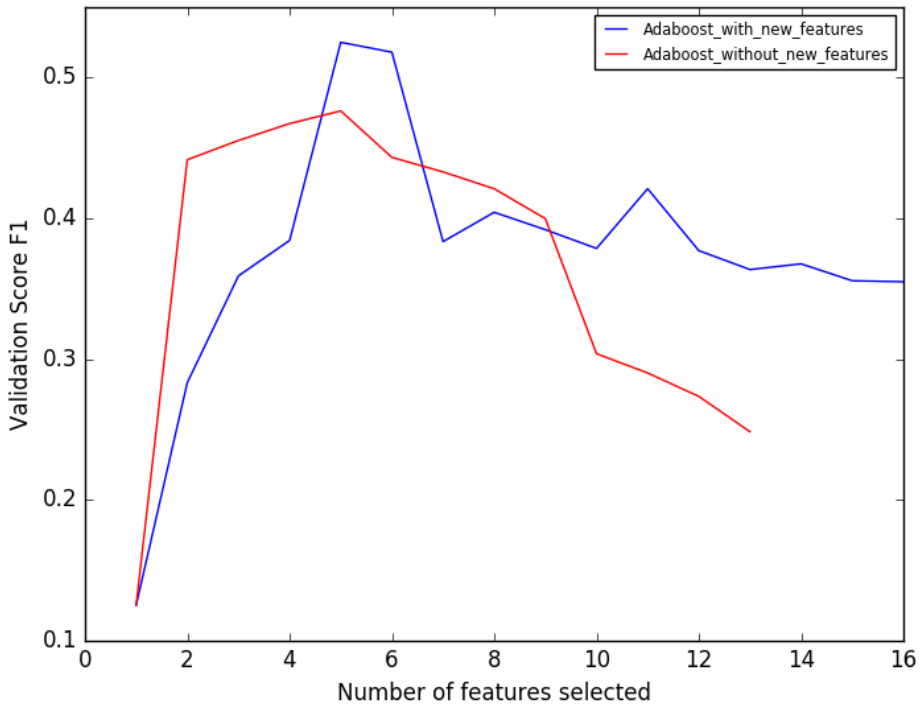
The classifiers which will be tested are Adaboost, ClassificationTree and RandomForest Tree. These three classifiers don't need feature scaling. Per this post, scaling is needed when different features need to be comparable in Euclidean distance, and Decision trees don't employ Euclidean distance during fitting [2].

Feature Selection:

New features:

Three new features were generated based on the email information. In our case, I believe the percent of emails sent to POI and emails received by POI among the total number of emails can better reflect the frequency of interaction between people and POI, considering the possibility that some people may not have too many emails but most of the emails were with POI, which may be potentially indicating that those people are closer to POI than the people who has the same number of emails with POI, but have many more emails. Including the percentage of emails with POI might provide a good picture about the interaction level between people and POI.

With new features included, the highest F1 score of Adaboost increased from 4.76 to 5.25.

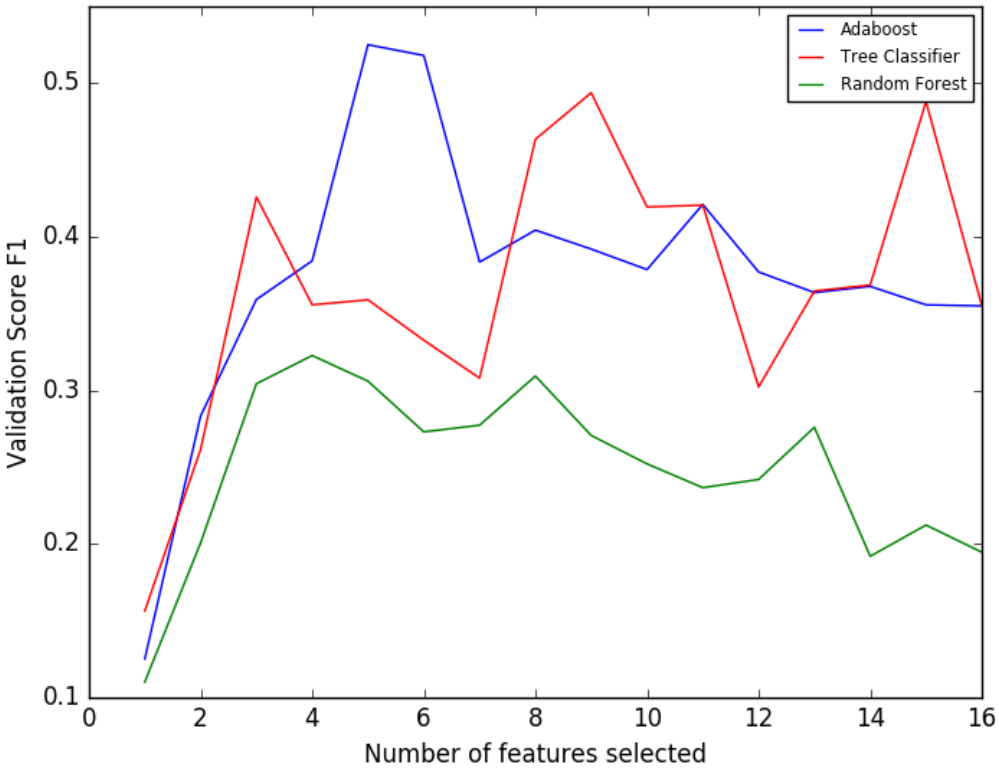


Fraction from poi: percentage of emails sent by POI.
Fraction to poi: percentage of emails sent to POI.
Fraction share with poi: percentage of received emails shared receipt with POI.

Feature selection:

The goal of feature selection is to identify the most useful and most relevant features for the problem we are working on. To select most relevant attributes from the features list, recursive feature elimination (RFE) algorithm was deployed. Since in the rest of the test, three different of classifiers will be examined, we want to select the best feature combination for each single classifier. Compared with SelectKbest which does not interact with classifier, RFE will select the best features conditional on classifier[1].

Different numbers of features were tested against three classifiers. As the below graph shows, Adaboost classifier with 5 most relative features selected by RFE achieves highest F1 score. (Table 1 in attachment list F1 scores for all the combinations of classifiers and different number of features). The feature importance is also documented as below.



Feature	shared_receipt_with_poi	exercised_stock_options	other	expenses	fraction_to_poi
Feature_importance	0.07004	0.21667	0.28149	0.33885	0.09292

Unbalanced Class:

As discussed in Data Exploration section, only 18 out of 146 data points are identified as POI, which unbalanced data distribution should be considered when tuning classification algorithm and choosing evaluation metrics.

Performance Measure:

In this binary classification problem, accuracy can be misleading since one class (in our case, non-POI) is much more common than another (POI) and guessing more common class (non-POI) will yield very high accuracy. For this reason, a different metric that is less sensitive to imbalance will be used to evaluate the predictive performance of classifier.

The goal is to identify the POI, which is the rare class. For this purpose, Precision, Recall and F1 score will be used.

In this case, precision reflects the proportion of predicted POI are truly POI, and recall represent the percentage of true POIs are identified as POI by the model. As the table shows, the 60 percent of predicted POI are correctly classified and 45 percent of true POIs are detected by our model.

Algorithm Tunning:

Most classifier are constructed to minimize the overall error rate, it will tend to focus more on the prediction accuracy of the majority class, which often results in poor accuracy for the minority class. Since the goal of this project is to identify the POI, which is the rare class, I selected F1 score as the performance measure during algorithm tuning process. F1 score conveys the balance between precision and recall and is usually used to evaluate the predictive performance on positive class (in our case POI is the positive class). I gave F1 as scoring parameter to GridSearchCV to select the parameter combination with highest F1 score for each classifier. Table 1 lists parameter tuned for each classifier.

Adaboost	'n_estimators': [10, 20, 30, 40, 60, 100], 'algorithm' : ['SAMME', 'SAMME.R']
Random Forest	n_estimators': [10, 20, 30, 40, 60, 100], 'max_features': ['auto', 'sqrt', 'log2']
Tree Classifier	'max_features': ['auto', 'sqrt', 'log2']

Validation:

Validation is always necessary to evaluate a model's performance. In most cases, after a model is generated we always want to test the model on a different set of related data to get a good estimate on how the model performs in terms of predicting yet-unseen data. To test model on another part of data will help us to avoid model overfitting and get a fair report on model generalization performance.

However, by partitioning the available data into two sets, we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of (train, validation) sets. Especially in this case, the dataset we are working on is fairly small. So cross validation in used in order to take full advantage of the dataset. StratifiedShuffleSplit cross validation is included in GridSearchCV function when finding best parameters for classifier.

Conclusion:

Through testing different combinations of features and classifiers by running RFE and GridSearchCV function, it was found Adaboost classifier gave the best performance in terms of identifying POI. The GridSearchCV result is documented as below:

Classifier	Feature Selected By RFE	F1 score in GridSearchCV
Random Forest Tree	bonus,total_stock_value,other,fraction_from_poi, fraction_shared_poi	0.306
Adaboost	shared_receipt_with_poi, exercised_stock_options,other,expenses,fraction_to_poi	0.525
Decision Tree	shared_receipt_with_poi, exercised_stock_options,other,expenses, fraction_to_poi	0.359

In the final test, Adaboost also performs very well, with F1 of 0.52, precision of 0.624 and recall of 0.45.

AdaBoostClassifier(algorithm='SAMME', base_estimator=None, learning_rate=1.0,n_estimators=60, random_state=0)

Accuracy: 0.88271 Precision: 0.62413 Recall: 0.45000 F1: 0.52295 F2: 0.47659

Total predictions: 14000 True positives: 900 False positives: 542 False negatives: 1100 True negatives: 11458

[1] https://discussions.udacity.com/t/grid-search-with-feature-selection-worse-results-when-adding-new-features/167487

[2] https://discussions.udacity.com/t/stratifiedshufflesplit-in-tester-py-modifying-tester-py/39517/7

F1 score Table for different classifiers and numbers of features selected by RFE:

Number of features	Adaboost F1 score	Tree Classifier Tree F1 score	Random Forest F1 score
1	0.125	0.156	0.110
2	0.283	0.261	0.200
3	0.359	0.425	**0.304**
4	0.384	0.355	0.322
5	**0.525**	0.358	0.306
6	0.518	0.332	0.273
7	0.383	0.308	0.277
8	0.404	0.463	0.309
9	0.392	**0.493**	0.270
10	0.379	0.419	0.252
11	0.378	0.420	0.236
12	0.377	0.302	0.242
13	0.364	0.364	0.276
14	0.368	0.368	0.192
15	0.356	0.488	0.212
16	0.355	0.355	0.194