# Identify Fraud from Enron Email project

## Goal of the project

Financial features and email features have been provided for the project. Along with this, some ‘Persons of Interest’ have been identified. My goal was to build upon the correlation between the features and the POI’s, and find additional persons of interest. The first step was to select and algorithm that can correlate the features to POI with sufficient accuracy. That was my goal for this project.

The dataset has a number of financial features which could relate to the to the POI’s provided. Some examples are “salary”, “total\_stock\_value” and “exercised\_stock\_options”. The email features such as “from POI to this person”, “from this person to POI” could also be relevant.

## Data Exploration

The data set has 146 data points, one for each person. There are only 18 identified POIs. The rest 128 are identified as non POIs.

One of the checks that I initially did was to check number of valid data fields for each of the features. On plotting this I identified a number of financial features that had over half the entries populated with ‘NaN’. I decided to remove these features, since these ‘NaN’ entries will also impact the other features which have been populated with real values. This resulted in removal of the following financial features - 'restricted\_stock\_deferred’, 'deferred\_income', 'loan\_advances' and 'long\_term\_incentive'.

## Outliers

Outliers were identified by plotting the data. It seemed that one of the data points had high values for all of its features. The name for this ‘employee’ was found to be ‘TOTAL’. So I removed this employee data from the data set. On rechecking the data plots, I could not identify any more outliers.

## Feature selection

I used the following steps to select features

1. Check for features with less number of valid data points, and remove them – There were a bunch of financial features mentioned above in the ‘Data Exploration’ section that were removed, since less than half of the entries were populated with valid values.

2. Separate financial and email features into separate datasets – I found that using this strategy resulted in more complete data points in each data set. I decided to evaluate the financial data set and email data set separately, and then compare their performance.

3. I removed all entries with any ‘NaN’ entries from both datasets. This also removed any outliers with all NaN values.

## New features

I created a set of new features by taking ratios of the financial and email features and included them in the relevant data sets. I expect some employees may have received a higher ratio of other benefits as compared to their salary, and this may be relevant to identifying POIs.

Similarly for email features, I created new features as ratios of the total messages the individual had sent or received.

Financial features

1. Payment ratio = Total\_payments / salary
2. Bonus Ratio = bonus / salary
3. total stock ratio = total\_stock\_value / salary
4. exercised stock ratio = exercised\_stock\_options / total\_stock\_value

Email Features

1. poi\_from\_ratio = from\_poi\_to\_this\_person / from\_messages
2. poi\_to\_ratio = from\_this\_person\_to\_poi / to\_messages
3. shared\_poi\_ratio = shared\_receipt\_with\_poi / from\_messages

## Feature Scaling

I scaled all the features provided using MinMaxScalar. However, since the algorithms I planned to use did not require scaled features, I did not utilize the scaled features.

## Selecting features

I decided to experiment with SelectKBest for the financial and email features separately. This was done after combining the old and new features for each data set – Financial and Email.

When I used SelectKBest, I obtained the following scores and pvalues for the financial features and the Email Features.

Financial features

('salary', 597564.72199881822, 0.0)

('total\_payments', 193020140.83775568, 0.0)

('bonus', 11480784.700453833, 0.0)

('total\_stock\_value', 97203311.089513928, 0.0)

('expenses', 474.0226565712552, 4.2698938808686188e-105)

('exercised\_stock\_options', 88202079.713050172, 0.0)

('other', 11410779.045417028, 0.0)

('restricted\_stock', 12021234.519666618, 0.0)

('total\_payment\_ratio', 63.827317511160032, 1.3581714392341234e-15)

('total\_stock\_ratio', 21.997873288900312, 2.7295274556805415e-06)

('exercised\_stock\_ratio', 0.051213934759051463, 0.82096424121142997)

('bonus\_ratio', 8.9196567574129997, 0.0028211752924206415)

Email features

('to\_messages', 597564.72199881822, 0.0)

('from\_poi\_to\_this\_person', 193020140.83775568, 0.0)

('from\_messages', 11480784.700453833, 0.0)

('from\_this\_person\_to\_poi', 97203311.089513928, 0.0)

('shared\_receipt\_with\_poi', 474.0226565712552, 4.2698938808686188e-105)

('poi\_from\_ratio', 88202079.713050172, 0.0)

('poi\_to\_ratio', 11410779.045417028, 0.0)

('shared\_poi\_ratio', 12021234.519666618, 0.0)

I used the following values on K on financial features and checked the results using tester.py

K = 3,5,7,8 and 11

The precision and recall values for each of these K values were

K = 3 - Precision: 0.47234 Recall: 0.22200

K = 5 Precision: 0.34157 Recall: 0.23500

K = 7 Precision: 0.25469 Recall: 0.32600

K = 8 Precision: 0.31746 Recall: 0.22000

K = 11 Precision: 0.28694 Recall: 0.23500

The values of precision and recall did not vary to a large extent as the value of K was increased. I decided to stop at the value of K where the scores were high and the pvalue was displayed as 0. This value of K selected was 7.

## Algorithms

I compared the performance of two algorithms, Naïve Bayes and Decision Tree using KFold testing. Over multiple tries I found the Decision Tree classifier provided better results. At this point I also reverted on my decision to separate the Financial and Email data. When the top 7 features were selected using SelectKBest, only the financial features showed up.

## Parameter tuning

I attempted to manually tune the ‘max\_depth’ parameter for the DecisionTreeClassifier. I didn’t want the Decision tree algorithm to overfit the data provided. But I wanted it to have sufficient depth to be able to use all relevant parameters.

I would have had either of these issues if I didn’t tune the max depth parameter. Then the comparison with Naïve Bayes algorithm would not have reliable results.

I expect to tune all relevant parameters for any algorithm I use, so I get optimal results for data set.

For the ‘max\_depth’ parameter, I used values ‘3’, ‘5’, ‘7’, ‘9’. The value of 7 provided the best result for precision and recall.

## Validation

I run validation tests to check if the algorithm that has been fitted to training data, can predict accurately on test data. It helps me identify the performance of the algorithm.

In this case, I wanted to confirm if the Naïve Bayes algorithm can recall POIs in the test data sets. That would help me rely on its predictions. I did need to be careful that the test data set was not totally different from the training data set. That would be a classic error, and my evaluation metrics would really low.

It does not make sense fitting the algorithm on the entire dataset. There would be no test set to validate the fitted algorithm. Also, splitting the data set into a single training set and a test set will not provide large enough data sets for training and testing. The performance values on the test set may not be reliable with few POI data points.

So I used Stratified Shuffle cross validation. This shuffles the data multiple times, and provides a training and testing data set for each shuffle. We can train and test the algorithm on each of these training and testing data sets to obtain a clearer estimate of the performance.

Since the number of POI’s over the entire data set is low, the recall values varied drastically. It was only after changing the max\_depth parameter and testing the results, I was able to achieve the required value of 0.3 for the evaluation metrics.

## Evaluation metrics

The two metrics I used to evaluate the performance of the algorithm are ‘Precision’ and ‘Recall’.

In this case, I’m trying to identify POIs. ‘Precision’ indicates how many of the POIs in the algorithm prediction are specified in the result. For example, when the predicted values have 3 POIs identified and only 1 of them is in the provided result set, the precision is 0.3.

‘Recall’ specifies, of the number of POIs in the result, how many are correctly predicted. If the result set has 4 POIs and only 2 of those show in the prediction, the recall value is 0.5.

When I ran tester.py using the DecisionTreeClassifier, the resulting precision and recall value were

Precision: 0.29613 Recall: 0.32900