Identify Fraud from Enron Email

1. Overview

Background

Enron Corporation was an American energy, commodities and services company based in Houston, Texas. In 2000, Enron was one of the world's largest companies with approximately 20,000 employees. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, there was a significant amount of typically confidential information entered into public record, including over 500,000 typically confidential emails from 158 employees and detailed financial data for top executives.

Goal

For this project predictive models were build, based on appropriate machine learning algorithms using scikit-learn, numpy, and pandas modules in Python. The goal was to identify whether a person is considered a person of interest (POI). A person of interest (POI) is an individual who was indicted, reached a settlement with the government without admitting guilt, or testified in exchange for prosecution immunity.

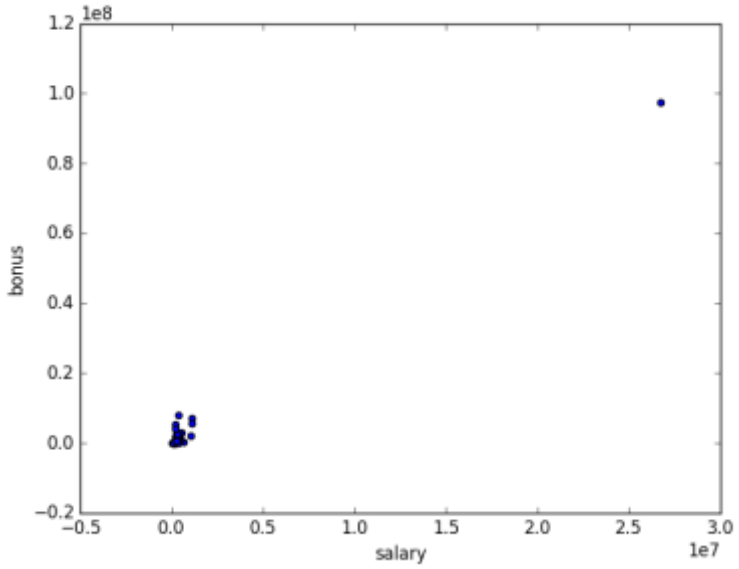
This is a Supervised Classification problem, as we are trying to predict a binary outcome (Non-POI / POI) for which we already know the correct answer. Our target is to achieve the most accurate predictions when we apply our Machine Learning algorithm to test this model. This report documents the machine learning techniques used in building a POI identifier as well as the followed validation process.

The dataset consists from 146 total number of data points, each of which corresponds to an Enron employee. From those 18 are POIs and 128 ono-POIs. After inspecting the list of features for missing values, there were 4 Features containing more than a lot of NaNs:

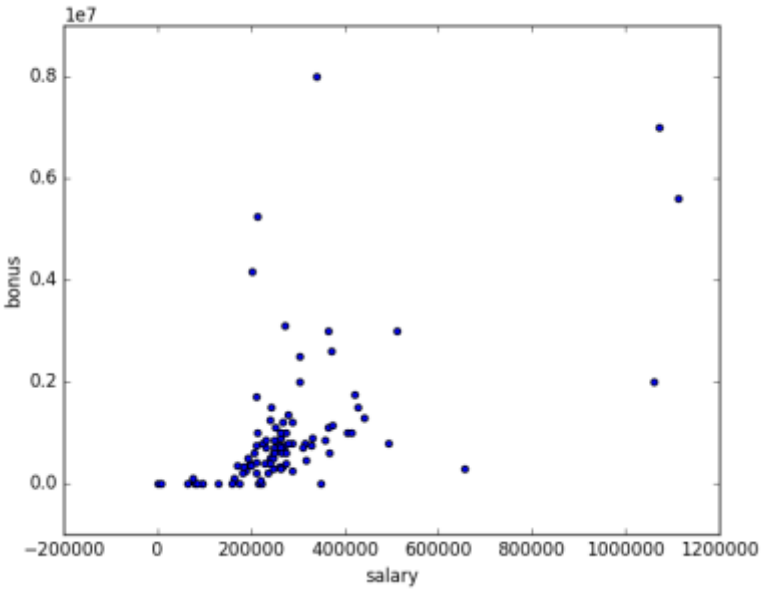
FEATURE	NaNs	NaNs %
deferral payments	107	73.3%
director fees	129	88.4%
loan advances	142	97.3%
restricted stock deferred	128	87.7%

Outliers

Outliers may result from sensor malfunction, data entry errors, or they may be valid extreme values. Next we are plotting bonus vs salary data looking for outliers and one outlier pops out immediately.



After reviewing the original data source (enron61702insiderpay.pdf) we found this outlier to be the “TOTAL” which sums up all bonuses and salaries and got it removed.



After removing the “TOTAL” outlier, the two extreme values shown in the plot are “LAY KENNETH” and “SKILLING JEFFREY”, two of Enron's biggest bosses and definitely persons of interest, thus they are just considered extreme values and definitely staying for further investigation. Then I kept searching for more outliers, using alternative methods.

After visual inspection of Enron employees, I identified a second outlier named “THE TRAVEL AGENCY IN THE PARK”, which I consider an outlier and have it removed. Finally, we are checking for keys that have NaN's in all features and identify a third outlier to remove, "LOCKHART EUGENE E". The final count of records, after removing the abovementioned outliers, is 143.

2. Features

I engineered two new features, “rate_from_POI” & “rate_to_POI”, developed by dividing the “from_poi_to_this_person” and “from_this_person_to_poi” features with "to_messages" and "from_messages" features accordingly. The new features demonstrate each employee’s rate of emails received by and sent to Persons of Interest. This choice was due to the hypothesis that there will be a more frequent interaction through email of POI’s with each other, than between POI’s and non-POI’s. After assessing the correlation of “from_poi_to_this_person” and “from_this_person_to_poi”, we had to come up with something better, so the “rate_from_POI” & “rate_to_POI” were tested, which eventually delivered a stronger connection.

After inserting the two newly constructed features into the dataset, I used the scikit-learn SelectKBest algorithm, in order to assess the most relevant and effective features. Their associated scores are listed below:

k_best.scores

[('exercised_stock_options', 24.815079733218194), ('total_stock_value', 24.182898678566879), ('bonus', 20.792252047181535), ('salary', 18.289684043404513), ('fraction_to_POI', 16.409712548035799), ('deferred_income', 11.458476579280369), ('long_term_incentive', 9.9221860131898225), ('restricted_stock', 9.2128106219771002), ('total_payments', 8.7727777300916792), ('shared_receipt_with_poi', 8.589420731682381), ('loan_advances', 7.1840556582887247), ('expenses', 6.0941733106389453), ('from_poi_to_this_person', 5.2434497133749582), ('other', 4.1874775069953749), ('fraction_from_POI', 3.1280917481567374), ('from_this_person_to_poi', 2.3826121082276739), ('director_fees', 2.1263278020077054), ('to_messages', 1.6463411294420076), ('deferral_payments', 0.22461127473600989), ('from_messages', 0.16970094762175533), ('restricted_stock_deferred', 0.065499652909942141)]

I tried the final classifier with the top 2, 3, 4, 5, 6 & 7 features and the most efficient option proved to be using the following top 4 features, together with 'poi'.

Best selected features

['poi', 'bonus', 'exercised_stock_options', 'salary', 'total_stock_value']

3. Algorithms

After trying 3 different algorithms (Gaussian naive bayes, Decision Trees, K nearest neighbors) and getting the following results, I decided to use and further tune the Decision Trees Classifier, since it proved to be the one with the best overall performance.

Gaussian Naive Bayes scores
accuracy = 0.923
precision = 0.5
recall = 0.33

Decision Trees scores
accuracy = 0.846
precision = 0.286
recall = 0.666

KNeighbors scores
accuracy = 0.948
precision = 1.0
recall = 0.33

4. Algorithm tuning

In order for an algorithm to perform well and provide the best possible results, one should consider tuning the parameters of the algorithm. In the DecisionTreeClassifier I used GridSearchCV and provide the classifier with a range of possible parameters for the GridSearchCV to find the best combination of parameters that should be used in order to score higher in the final results.

Moreover, I decided not to scale my features, since the implemented algorithms are independent of the feature scaling. Specifically, Decision Trees, the final algorithm which was used in my model, is not affected by different scales. Therefore such variable transformation was not required, because the Tree structure would remain unchanged.

After using GridCV I used the best estimator as a parameter to be implemented to DecisionTreeClassifier. Parameters tuning refers to the adjustment of the algorithm when training, in order to improve the fit on the test set. Parameter can influence the outcome of the learning process, the more tuned the parameters, the more biased the algorithm will be to the training data & test harness. The strategy can be effective but it can also lead to more fragile models & overfit the test harness but don't perform well in practice. Manually testing the implemented algorithm, I end up using the following paremeters:

DT_Tuned_clf.best_params_

{'min_samples_split': 1, 'splitter': 'best', 'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 1}

5. Validation

Validation is a method that estimates the performance of the applied model. The way to validate a dataset is to randomly splitting the data into training and testing. Training data is the data that will fit in an algorithm and create a model, which can then be validated based on the testing data.

A classic mistake one can make during the validation process is overfitting, where your model is well tuned to predict your training data but then fails to perform well on the test data. In order to avoid overfitting, you implement a tool called cross-validation.

Due to the imbalance of my relatively small dataset, I implemented a cross-validation method for stratified sampling called Stratified Shuffle Split, which creates splits by preserving the same percentage for each target class as in the complete set. This is important because it makes sure that the ratio of POI and non-POI is the same during training and testing.

6. Evaluation metrics

Accuracy

Accuracy refers to the ratio of correct predictions out of the total predictions made. In our case, Accuracy does not fit as a suitable evaluation metric due to the sparsity of POI's being predicted. For instance, we could get pretty high levels of accuracy for guessing that there is no POI in our dataset, thus realising that just because a model has a very high accuracy doesn't necessarily mean it is a suitable one. In our case where there are much more non-POI than POI, recall and precision are both better measures than accuracy.

Precision

Precision can be thought of as the ratio of correct positive predictions made out of the total positive predictions made. In our case, precision can be thought as the probability a specific person to being a POI, when the model predicts that it is.

Recall

Recall refers to the ratio of correct positive predictions made out of the total actual positive. In our case, recall can be thought more important criteria than precision, since all people involved in the fraud need to be found, even if that means some innocent people need to be also investigated along the way. Thus, an effort was made to optimise the applied model, towards higher recall values.

The metrics of our tuned model:

Accuracy: 0.80231
Precision: 0.36732
Recall: 0.39450
F1: 0.38042
F2: 0.38875

Whereas the metrics of the untuned model:

Accuracy: 0.80215
Precision: 0.36458
Recall: 0.38500
F1: 0.37451
F2: 0.38074