## Enron Fraud Detectors

## By: Sarah Alofaysan

## Dec 2017

1. **Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those?**

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 use financial and email data from [Enron corpus](https://www.cs.cmu.edu/~./enron/) - publicly made by US Federal Energy Regulatory Commission during its investigation of Enron, which comprised email and financial data of 146 people most of which are senior management of Enron, to come up with a predictive model that could spot an individual as a “Person of Interest (POI)”.

The provided dataset contains:

* 146 records
* 1 labeled target feature (POI)
* 14 financial related features
* 6 email related feature
* 18 persons were labeled as a "Person Of Interest" (POI)
* A lot of NAN values, the top features with NAN values are
  + Loan Advances (142 NaN)
  + Director Fees(129 NaN)
  + Restricted stock deferred (128 NaN)
  + Deferred payment (107 NaN)

In addition, through my exploratory analysis I found out 3 outliers need to be eliminated which are:

* **LOCKHART EUGENE E:** This record contained only NaN data.
* **TOTAL:** This does not represent an individual
* **THE TRAVEL AGENCY IN THE PARK:** This does not represent an individual

1. **What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values.**

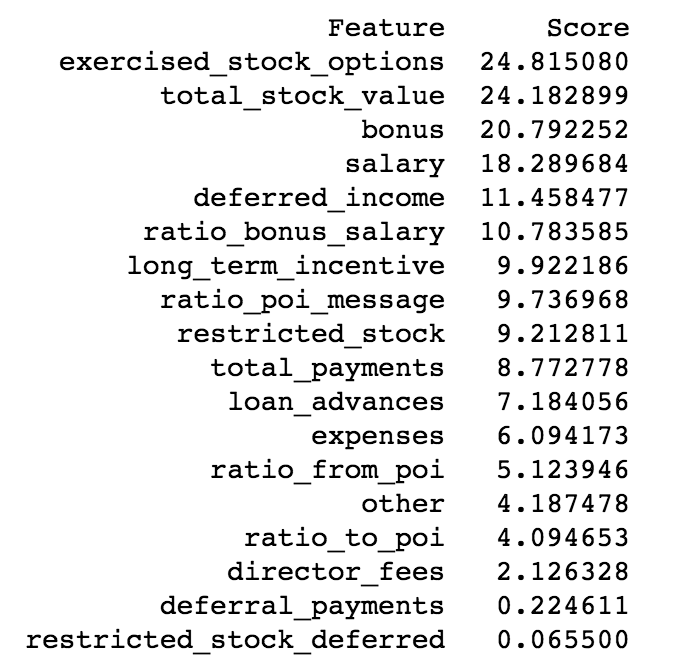
From the initial features, I engineered and added four more features such as: Bonus and Salary Ratio, From this Person to POI Ratio, From POI to this Person Ratio, POI messages Ratio.

* POI messages Ratio= Shared Receipt with POI / (To Messages + From Message)
* From this Person to POI Ratio= From This Person to POI / to messages
* From POI to this Person Ratio= From POI to This Person/ from Messages
* Bonus and Salary Ratio = Bonus/Salary

I also change the NAN values to Zero for better calculation, and I also change POI value from True and False to 0 and 1.

I removed some features, as it will provide any information anymore after I engineered new features, such as: to messages, from messages, shared receipt with poi, from this person to poi, from poi to this person, email address.

After that, I used SelectKBest on the remaining features to figure out the highly influential features. Values below 1 in score were removed in later stage from feature list.



1. **What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?**  
     
   I have tried six different algorithms and each algorithm response with different Accuracy, Precision and Recall rate, as illustrated below:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** |
| **Gaussian**  **Naive Bayes** | .86 | .40 | .40 |
| **SVM** | .88 | .00 | .00 |
| **Decision Tree** | .88 | .00 | .00 |
| **K-means** | .83 | .00 | 0.7 |
| **Random Forest** | .88 | .50 | .20 |
| **AdaBoost** | .83 | .25 | .20 |

I ended up selecting Gaussian Naïve Bayes as the best classifier for this problem as it shows better precision, recall and accuracy in comparison with the remaining classifiers.

1. **What does it mean to tune the parameters of an algorithm, and what can happen if you don’t do this well?  How did you tune the parameters of your particular algorithm? What parameters did you tune? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier).**

Parameters tuning means the adjustment of the algorithm in order to improve the fit on the test set. Parameters can influence the outcome of the learning process, the more tuned the parameters, the more biased the algorithm will be to the training data. If the tuning was not perfect we can have over-fitting or under-fitting. For the algorithm I picked I did not do any parameter tuning. However, for the remaining algorithms I referenced sklearn documentation to help me tuning the model.

1. **What is validation, and what’s a classic mistake you can make if you do it wrong? How did you validate your analysis?**

The classical mistake that could happen is the model cannot generalize the to data outside the training set due to over-fitting. I validated my algorithm by cross validation, where I trained the algorithm on training data and measure the algorithm performance on the testing set.

1. **Give at least 2 evaluation metrics and your average performance for each of them.  Explain an interpretation of your metrics that says something human-understandable about your algorithm’s performance.**

I used Accuracy, Precision and Recall as main evaluation metrics. The best performance belongs to Gaussian Naïve Bayes (Accuracy: 0.86 Precision: 0.40 Recall: 0.40).

By definition: Precision refers to the ratio of true positive (predicted as POI) to the records that are actually POI while recall described ratio of true positives to people flagged as POI. So with a precision score of 0.40, it tells us that if this model predicts 100 POIs, there would be 40 people are actually POIs and the rest 60 are innocent. With recall score of 0.40, this model finds 40% of all real POIs in prediction. On the other hand, Accuracy is not a good measurement alone to decide the efficiency of the model.

**References:**

1. SKlearn Documentation:[**http://scikit-learn.org/stable/index.html**](http://scikit-learn.org/stable/index.html)
2. [**http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/**](http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/)
3. **Intro to Machine Learning Course at Udacity**
4. [**https://github.com/sebasibarguen/udacity-nanodegree-machinelearning/**](https://github.com/sebasibarguen/udacity-nanodegree-machinelearning/)