Introduction

> [In addition to being the largest bankruptcy reorganization in American history at that time, Enron was cited as the biggest audit failure]
(http://en.wikipedia.org/wiki/Enron_scandal)

From a $90 price per share, to a $1 value represents the huge value loss and scam that happened in Enron. This case has been
a point of interest for machine learning analysis because of the huge real-world impact that ML could help out and try to figure out what went wrong and how to avoid it in the future. It would be of great value to find a model that could potentially predict these types of events before much damage is done, so as to permit preventive action. Corporate governance, the stock market, and even the Government would be quite interested in a machine learning model that could signal potential fraud detections before- hand.

Enron Data

The interesting and hard part of the dataset is that the distribution of the non-POI's to POI's is very skewed, given that from the 146 there are only 11 people or data points labelled as POI's or guilty of fraud. We are interested in labelling every person in the dataset
into either a POI or a non-POI (POI stands for *Person of Interest*). More than that, if we can assign a probability to each person to see what is the chance she is POI, it would be a much more reasonable model given that there is always some uncertainty.

Data Processing

All features in the dataset are either financial data or features extracted from emails. Financial data includes features like salary and bonus while the email features include number of messages written/received and to whom/form.

There are 2 clear outliers in the data, **TOTAL** and **THE TRAVEL AGENCY IN THE PARK**. The first one seems to be the sum total of all the other data points, while the second outlier is quite bizarre. Both these outliers are removed from the dataset for all the analysis. Also all features are scaled using the **MinMaxScaler()** (although it is not included in the final model).

New features

From the initial dataset, 5 new features where added, you can find more details in the area below:

Ratio of POI messages |
POI related messages divided over the total messages from the person |

Log of financials (multiple) |
Financial variables with logarithmic transformation |

Square of financials (multiple) |
Financial variables with squared transformation |

The reasoning behind the **ratio of POI messages** is that we expect that POI's contact each other relatively more often

than with non-POI's and the relationship might be non-linear. We also can expect that the financial gains for POI's is actually non-linear, that is why applying a logarithmic and a square transformation to the original features, and it should improve many algorithms.

PCA

It's quite reasonable to think that all the **email** features we have, 5 initial features plus 1 computed feature, really represent 1 underlying feature or principal component, something like increased amount of communication between POI's versus between POI's and non-POI's. The same goes for the financial features, which we could think are really measuring the POI's corruption via big money gains. In other words, we expect that a POI has a higher money gain compared to a non-POI, and that all the financial features are really trying to measure this underlying one. By tuning the parameters, we get the best classification results with 2 email components and 3 financial components. From the **29** features in total, they are reduced to **20** principal components.

Algorithms selection and tuning

For the analysis of the data, a total of 10 classifiers were tried out, which include:

- Logistic Regression
- Linear Discriminant Analysis
- Decision Tree Classifier
- Gaussian Naive Bayes
- Linear Support Vector Classifier (LinearSVC)

- AdaBoost
- Random Forrest Tree Classifier
- K Nearest Neighbor
- KMeans
- Bernoulli RBM (together with Logistic Regression)

The object of the algorithm is to classify and find out which people are more likely to be POI's. There are clearly
2 categories we are looking to label the data.

To tune the overall performance, both automated and manual tuning of parameters was involved. The automated tuned parameters where done using the **GridSearchCV** from SkLearn. The manual tuning occurred in the following ways:
1. Including the PCA features
2. Adding/removing features
3. Scaling features

For the most part, PCA made a huge improvement when the new features where added. PCA is kind of getting the best parts of the 29 features and cramming them up into 20. The new features really made the difference to push recall and precision up.

Optimization

All the machine-learning algorithms where optimized using **GridSearchCV()**. The general
process was:

> build list of classifier with parameters > optimize each classifier with training data > evaluate all the classifiers > compare f1, recall and precision scores > choose the best classifier

Validation and Performance

To validate the performance of each algorithm, `recall`, `precision` and `F1 scores` where calculated for each one. You can find below a summary of the scores of the top algorithm.

Logistic Regression
F1 Score: 0.43820 | Recall: 0.55400 | Precision: 0.36245 |

The best classifier was actually a *Logistic Regression* using PCA beforehand. This was achieved by using `sklearn Pipeline`. The logistic regression achieved a consistent score above 0.30 for both precision and recall. The final parameters that were used are detailed below:

```{Python}
Pipeline(steps=[('pca', PCA(copy=True, n_components=20, whiten=False)), ('logistic', LogisticRegression(C=100000000000000000000L, class_weight='auto', dual=False, fit_intercept=True, intercept_scaling=1, penalty='l2', random_state=42, tol=1e-10))])
```

It seems that the most important parameter to tune was to set the `class_weight` to `auto`. I suspect this is due to the skewed nature of the dataset, because class weight assigns the importance of each class (POI or non-POI) depending on the inverse appearance of the class. So it set a much higher importance to POI's class which is exactly what we want in this case.

Discussion and Conclusions

This was just a starting point analysis for classifying Enron employees. The results should not be taken too seriously and more advanced models should be used. Possibilities for future research could be to include more complex pipelines for the data, or even Neural Networks. Here we tried a basic neural network, but the SkLearn library is very limited in what it has to offer in this regard.

References

- [Udacity - Intro to Machine Learning course](https://www.udacity.com/course/ud120)
- [Sklearn documentation](http://scikit-learn.org/stable/documentation.html)

```
```