Who is the POI?

Jae Hee Lee

Background

The **Enron scandal**, revealed in October 2001, eventually led to the bankruptcy of the Enron Corporation, an American energy company based in Houston, Texas, and the *de facto* dissolution of Arthur Andersen, which was one of the five largest audit and accountancy partnerships in the world. In addition to being the largest bankruptcy reorganization in American history at that time, Enron was cited as the biggest audit failure.

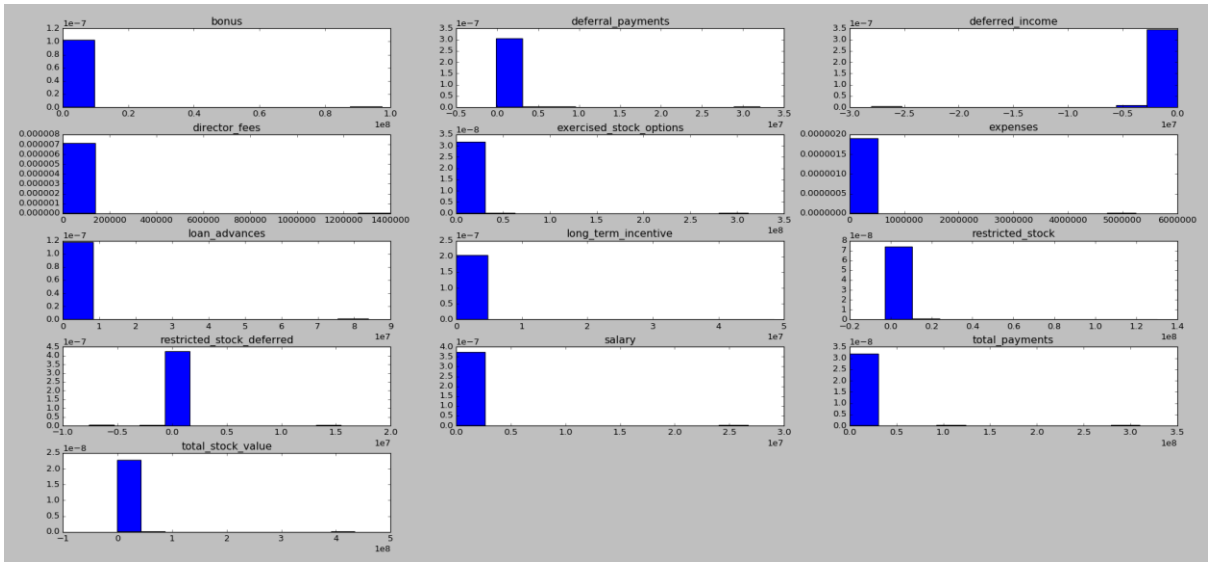
1. Data Exploration

My mission, as a detective, is to predict the people who committed the fraud, namely, the POIs. Using the financial and email data from Enron, this project will build a model that can 'detect' whether an individual is a POI (person of interest) or not. To do this, it is important to explore data first. According to data_exploration.py in tools, we have:

- There are 146 people (or records) in the dataset.
- There are 18 people (or records) labelled as a POI in the dataset.
- There are in total 21 features for each person (or record) in the dataset.
  - ✓ Feature **salary** has 51 missing values.
  - ✓ Feature **to_messages** has 60 missing values.
  - ✓ Feature **deferral_payments** has 107 missing values.
  - ✓ Feature **total_payments** has 21 missing values.
  - ✓ Feature **loan_advances** has 142 missing values.
  - ✓ Feature **bonus** has 64 missing values.
  - ✓ Feature **email_address** has 35 missing values.
  - ✓ Feature **restricted_stock_deferred** has 128 missing values.
  - ✓ Feature **total_stock_value** has 20 missing values.
  - ✓ Feature **shared_receipt_with_poi** has 60 missing values.
  - ✓ Feature **long_term_incentive** has 80 missing values.
  - ✓ Feature **exercised_stock_options** has 44 missing values.
  - ✓ Feature **from_messages** has 60 missing values.
  - ✓ Feature **other** has 53 missing values.
  - ✓ Feature **from_poi_to_this_person** has 60 missing values.
  - ✓ Feature **from_this_person_to_poi** has 60 missing values.
  - ✓ Feature **poi** has 0 missing values.
  - ✓ Feature **deferred_income** has 97 missing values.
  - ✓ Feature **expenses** has 51 missing values.
  - ✓ Feature **restricted_stock** has 36 missing values.
  - ✓ Feature **director_fees** has 129 missing values.

2. Outlier Detection

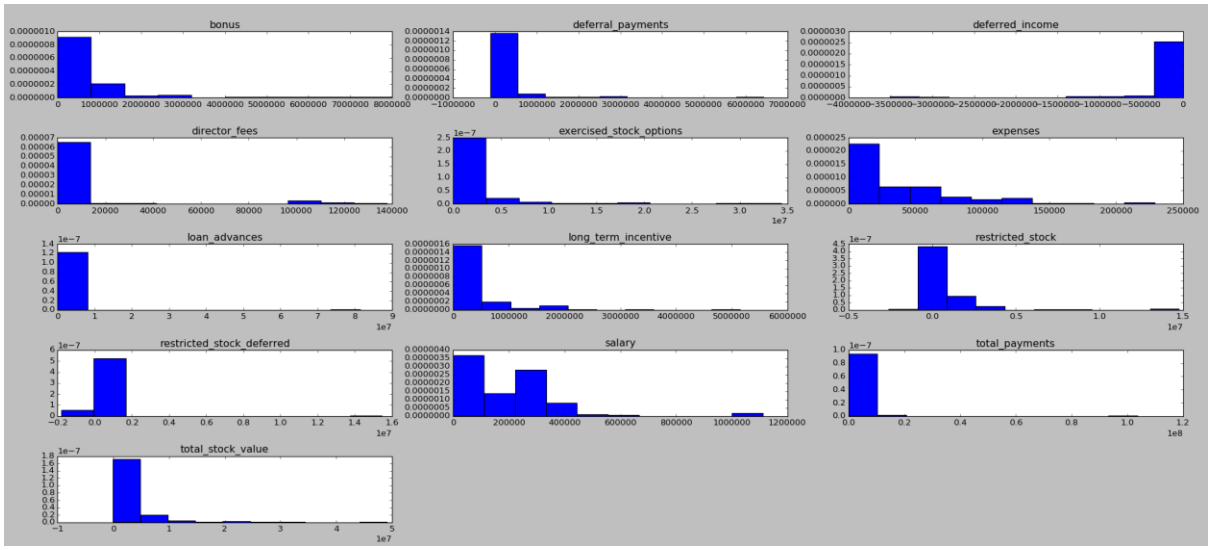
Now the question is, is there an outlier in this dataset. If so, will it be useful for our purpose? I have conducted outlier detection analysis (which can be found in tools/outlier_detection.py) and first I plotted the distribution of data for each feature and not surprisingly, there were outliers for features.



I wanted to know who the outliers are so I used 'bonus' feature to see who they are and the result was:

- **TOTAL**
- LAVORATO JOHN J
- LAY KENNETH L
- SKILLING JEFFREY K
- BELDEN TIMOTHY N

It is obvious from the name that TOTAL cannot be a person so this is certainly not useful for our analysis so I removed it. Other entities that I removed are: 'THE TRAVEL AGENCY IN THE PARK' and 'LOCKHART EUGENE E' (by analyzing the PDF file). After the removal, the result was:



3. Feature Selection

While browsing features, I thought that it would be a good idea to add new features from the existing features.

- One new feature would be poi_interaction_ratio, that is, total number of messages sent and received with POI / total number of messages sent and received **(email)**.
- Another new feature would be restricted_stock_ratio, that is, restricted stock value / total stock value **(financial)**.
- Another new feature would be cash_received, which is the sum of salary, bonus, loan_advances and exercised_stock_options **(financial)**.

To select the most relevant features, I have decided to use the scikit-learn's SelectKBest module to sort features in score order. The results are:

```
[('exercised_stock_options', 24.815079733218212),
 ('total_stock_value', 24.182898678566886),
 ('bonus', 20.792252047181531),
 ('cash_received', 19.587427711132936),
 ('salary', 18.289684043404527),
 ('deferred_income', 11.458476579279765),
 ('long_term_incentive', 9.9221860131898243),
 ('restricted_stock', 9.2128106219770771),
 ('total_payments', 8.7727777300916738),
 ('shared_receipt_with_poi', 8.5894207316823774),
 ('loan_advances', 7.1840556582887247),
 ('expenses', 6.0941733106389337),
 ('poi_interaction_ratio', 5.3993702880944179),
 ('from_poi_to_this_person', 5.2434497133749636),
 ('other', 4.1874775069953794),
 ('from_this_person_to_poi', 2.3826121082276743),
 ('director_fees', 2.126327802007705),
 ('to_messages', 1.6463411294419978),
 ('restricted_stock_ratio', 1.0901571696328465),
 ('deferral_payments', 0.22461127473600523),
 ('from_messages', 0.16970094762175433),
 ('restricted_stock_deferred', 0.06549965290989139)]    (run tools/feature_selection.py)
```

As the feature added really explains the top 10 features (incorporates), I decided to use just 'poi', 'cash_received' and 'poi_interaction_ratio' as features.

4. Feature Scaling

Before running classifiers, I decided to apply scaling because features and have different units and it is important to have consistency (i.e. give equal weight to every feature). I used MinMaxScaler from scikit-learn preprocessing module.

5. Classification, Validation and Evaluation Metrics

- I tried several algorithms including GaussianNB, kmeans, decision tree and k neighbor.

- Parameters had to be tuned manually including adding and removing features, scaling and for each classifier's parameters I manually changed it (for example, for k-means, the n_clusters had to be set to 2 as we want the result to be POI or non-POI).
- Validation ensures that the algorithm (in this case classifiers) generalizes well. A classic mistake would be over-fitting, meaning that the model is trained and performs really well for the training dataset but does bad on cross validation and test datasets. There are many ways to validate the result including using StratifiedKFold (for our purpose, StratifiedShuffleSplit is okay due to the small number of dataset and this is done in final_project/tester.py) and K-folds.
- The evaluation metrics used were precision and recall. First, accuracy cannot be a good metrics as the data is pretty skewed (few POIs). Precision measures the ratio of true positives to the records that are actually POIs (i.e. pois judged as pois). Recall measures the ratio of true positives to the records marked as POIs. Our objective is to ensure that the result gives 0.3 or above for both recall and precision.

Validation (Stratified Shuffle Split)

The **decision tree classifier** was the best classifier that gave more than 0.3 for both recall and precision for two validations (k-fold (K=3) and StratifiedShuffleSplit) The results are:

1. Decision Tree Classifier

Validation 1 (K-fold)

precision	0.333333
recall	0.4
accuracy	0.837209

Validation 2 (StratifiedShuffleSplit)

precision	0.44094
recall	0.30050
accuracy	0.83377

2. K-means

Validation 1 (K-fold)

precision	0.196667
recall	0.2
accuracy	0.790698

Validation 2 (StratifiedShuffleSplit)

precision	0.25756
recall	0.11500
accuracy	0.81285

3. Naïve Bayes (GaussianNB)

Validation 1 (K-fold)

precision	0.666667
recall	0.4
accuracy	0.906977

Validation 2 (StratifiedShuffleSplit)

precision	0.58172
recall	0.22600
accuracy	0.85592

4. K-Neighbors

Validation 1 (K-fold)

precision	0.25
recall	0.2
accuracy	0.837209

Validation 2 (StratifiedShuffleSplit)

precision	0.18122
recall	0.04150
accuracy	0.82369

Conclusion

Given the sparse nature of dataset, many algorithms did not perform well (although most of them performed pretty well given the situation). It was interesting to see that new features that used the top 10 existing features gave good results.