# Identifying Enron Fraud

Cameron Roberts

## Question 1

The purpose of this project is to try and identify persons of interest (POI) in a criminal investigation into corporate fraud which took place at Enron. In order to identify POIs we will utilise a public data set containing financial and email data for the companies executives.

Machine learning is a useful tool in this exercise because it can help us uncover patterns in the data that may not be observable through manual inspection.

The dataset provided for this project contains financial data and email meta data for 146 Enron executives. Of these executives 18 have been acknowledged as persons of interest. It is our task to try and build a classifier which accurately predicts these POIs from the available data. The data contains 21 features/variables including the employee’s salary, bonus who they’ve sent emails to and who they’ve received them from.

The data is not perfect with most employees having incomplete data for the 21 features. For example, 51 employees don’t have a salary recorded and only 4 have data for loan advances. We will have to be mindful of this fact as we use the features in any classifiers.

When looking at the financial features, salary for instance, there is an obvious outlier which presents itself. There is a record for the TOTAL of all employee’s salary, bonuses etc. This record has simply been removed from the data set before continuing with any further analysis.

On further inspection of the data we see that there are 3 people (Ronnie Chan, Eugene Lockhart, William Powers) which have zero/null total payments and total stock value. These people have also been removed.

## Question 2

To begin, I excluded a number of features from further consideration. These included email address as this is simply another identifier for a person. I also removed the total payments and total stock value features since these two are merely the sum of a number of other features so contain redundant information. Finally, I removed features which had less than 50% coverage for all records.

Next, I utilised principle component analysis (PCA) to construct two new features – one each for the financial and email groups of features. I figured these would be two good features to add since we are dealing with two basic types of information: 1) the quantity/type of money being handled; and, 2) the volume/type of communication in emails.

The features in these two sets were first scaled using *preprocessing.scale* from *sklearn* as PCA requires the features to be on the same scale. Otherwise, certain features may dominate the principle component. These two features were added to the original list so that a choice of the best could be made from amongst all.

From here, I used *SelectKBest* to select the best *k* features. This selection was done in a pipeline as part of a grid search such that we tested *k* for values 1 to the number of features. Since I used a decision tree (see Q3) no further scaling was needed.

The result of this search was that the following nine features were selected (with scores from SelectKBest):

* Salary (18.00)
* Exercised stock options (24.53)
* Bonus (20.52)
* Restricted stock (9.08)
* Shared receipt with POI (8.43)
* Expenses (5.95)
* Other (4.09)
* From POI to this person (5.14)
* Financial PC - my constructed feature from PCA (24.97)

## Question 3

I ended up using a decision tree supplemented by *AdaBoost*. I originally started with a plain decision tree and also tested a random forest. The F1, precision and recall scores (returned from *tester.py*) for the three algorithms were:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Precision | Recall | F1 |
| Decision Tree | 0.417 | 0.409 | 0.413 |
| Random Forest | 0.275 | 0.291 | 0.283 |
| AdaBoost | 0.351 | 0.846 | 0.496 |

While the precision of AdaBoost was not as good as the decision tree, its F1 score and recall were superior.

## Question 4

Tuning the parameters of an algorithm refers to adjusting the different arguments provided to a class/method such that the result of the fit is optimised. For example, a decision tree has several different parameters which can take different values including (but not limited to):

* criterion
* max\_depth
* class\_weight

In fact, I tuned these three parameters in my solution by incorporating them in a GridSearchCV along with paramters for AdaBoost and SelectKBest. The grid search tries all combinations of the parameters and selects the estimator with the best score from among them.

If parameters aren’t tuned, you may not be getting the best performance from your algorithm for your particular problem since defaults will be used.

## Question 5

Validation refers to testing or confirming the performance of your algorithm on a separate independent data set to the one used to train the classifier. You generally use some type of metric such as accuracy to determine the performance.

A classic mistake is to not use an independent partition for validation and instead test the performance of the classifier on the same set used for training. This will result in inflated performance due to overfitting.

Since the Enron dataset we used in this project was relatively small, I used a stratified shuffle split method of cross validation on the entire data set in the grid search. I utilised stratified shuffle split as this is what is also used in the provided tester.py. This method returns indices to create training/test partitions over several iterations (folds). By doing this over the entire data set, I was able to maximise the data available to validate. This cross validator is provided to GridSearchCV so that it can be incorporated in its search.

## Question 6

The two evaluation metrics we are most concerned with in this exercise are *precision* and *recall*. The estimators tested in the grid search were evaluated using the *F1 score* so that we could determine the best. The F1 score is the harmonic mean of precision and recall and so suited this purpose.

The average values of precision and recall returned from *tester.py* for the performance of the final algorithm were 0.351 and 0.846 respectively.

In terms of our particular application these values can be explained as follows:

* *Precision*  
  35.1% of the persons we identified as a POI were indeed a POI. Meaning 64.9% of the persons we identified as a POI were not POIs.
* *Recall*  
  84.6% of all POIs were identified as such. As a result, 15.4% of POIs were not identified.

## References

1. <http://matplotlib.org/users/pyplot_tutorial.html>
2. <http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-scaler>
3. <http://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection>
4. <http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>
5. <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics>
6. <http://scikit-learn.org/stable/auto_examples/model_selection/grid_search_digits.html#example-model-selection-grid-search-digits-py>
7. <http://scikit-learn.org/stable/modules/pipeline.html>
8. <http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedShuffleSplit.html>
9. <http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html#sklearn.grid_search.GridSearchCV>
10. <http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation>
11. <http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier>
12. <http://stackoverflow.com/questions/32889929/gridsearchcv-scoring-parameter-using-scoring-f1-or-scoring-none-by-default-u>
13. <http://stackoverflow.com/questions/30805192/scikit-learn-random-forest-class-weight-and-sample-weight-parameters>
14. <https://en.wikipedia.org/wiki/Precision_and_recall>
15. <https://discussions.udacity.com/t/understanding-gridsearchcv/26168/3>
16. <https://discussions.udacity.com/t/different-results-by-using-gridsearchcv-vs-direct-applying-algorithm/35550/14>
17. <https://discussions.udacity.com/t/can-the-scoring-function-in-tester-py-be-used-in-gridsearch/24236/3>
18. <https://discussions.udacity.com/t/gridsearchcv-and-testingtraining-data/36107/19>
19. <https://discussions.udacity.com/t/gridsearchcv-and-testingtraining-data/36107/4>