# P5: Identify Fraud from Enron Email Questions

***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? [relevant rubric items: "data exploration", "outlier investigation"]***

The goal is to build an algorithm based on Enron employees' financial compensation and email statistics that can identify those employees who committed fraud. We have been provided with a dataset of top Enron executives and a Person of Interest (POI) identifier that flags certain individuals who were found in court to have committed fraud.

The dataset contains 146 records (rows) including:

- 126 employees who are NOT considered to have committed fraud (POI flag = false)
- 18 employees who ARE considered to have committed fraud (POI flag = true)

In addition to each employee's name, email address, and POI flag, there are 19 features in this dataset including:

- Email statistics: from/to emails count, mails exchanged with POIs
- Non-Equity compensation: salary, bonuses, expenses
- Equity compensation: restricted_*stock*_deferred, total_*stock*_value, restricted_*stock, excercised*_stock_options

There were several types of outliers including:

1. Rows not associated with individuals (TOTAL and THE TRAVEL AGENCY IN THE PARK). I removed these records.
2. Lou Pai was a high level executive at Enron's energy trading division who was eventually tried by the SEC and settled out of court for over $30 million. He is marked with a POI flag of '0' but I feel that he should be a person of interest.
3. Many employees appear with no salaries. I am not sure why this is the case.

I left the dataset as-is except for removing the two records indicated in item #1 above.

2. 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. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

I selected three features to build my classifier: Exercised stock options, # of To Messages, and the ratio of shared receipt POI mails to all received mails.

I created 6 scaled features including:

- From POI Ratio - Percentage of received mails that were sent from POIs
- To POI Ratio - Percentage of sent mails sent to POIs

- Shared Receipt POI Ratio - Percentage of received mails where POIs were included in the recipient list
- Exercised Stock % - Percentage of stock compensation that was executed
- Bonus:Salary Ratio - Fraction (or multiple) of salary received as bonus
- Bonus:Total Ratio - Percentage of total compensation that was bonus

To select the features, I tried to use sklearn's SelectKBest function but was not impressed with the results. The 'best' fields had the highest accuracy scores but not necessarily high precision and recall.

Instead of testing a single 'best' combination, I wanted to run a comprehensive test on 2-feature and 3-feature classifiers. I wrote a couple of test scripts (bulk_*tester*_2features.py and bulk_*tester*_3features.py) that iterate over different classifiers and combinations of features. I hand selected the features included in the bulk tester by finding features that exist for more than 50% of the records (and for both POI and non-POI employees).

I output the results into .csv files stored in the reports/ folder. Using a spreadsheet tool to sort by Recall, I was able to quickly identify features on Decision Tree classifiers that gave high accuracy scores (80%+) as well as high precision and recall. This classifier scored 83.2% accuracy based on my tests with a 49.6% precision and 47% recall.

For my classifier, I confirmed the following average feature importance scores (for 1000 sets of randomized training data):

- Exercised stock options: 0.39
- To Messages: 0.28
- Shared Receipt POI Ratio: 0.33

### 3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: "pick an algorithm"]

I used a Decision Tree algorithm. In my bulk tester I also tried using Naïve Bayes, Support Vector Machines, and Random Forest classifiers. I had problems with the Support Vector Machines due to many 0s in the dataset.

For other classifiers I was able to train my classifiers and test them on the test data. Although many Naïve Bayes algorithms scored high on accuracy, they were not as high on the recall and precision scores. I felt that the Decision Tree classifiers scored better overall with >45% precision and score.

Many of the top (accuracy) scoring classifiers were Naïve Bayes. However the recall scores were quite low. I found that the most balanced classifiers were decision trees.

### 4. 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? (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). [relevant rubric item: "tune the algorithm"]**

A decision tree classifier offers several configurable parameters including:

max_features – Number of features to consider when making a split
max_depth – Maximum depth of the tree
min_samples_split – Minimum number of samples required to split a node

If the decision tree is not deep enough (not enough branch conditions), then the classifier may not be tuned sufficiently for diverse kinds of data. On the other extreme, if there are too many branches, the classifier may be overfitting to training data and therefore not flexible enough to consider other test data.

Similar to the bulk tester, I wrote a script to try iterate across different combinations of the above tuning parameters. Using the non-default '2' setting for maximum features and '6' setting for min samples split, I was able to increase the performance of the classifier.

| | Default Parameters | Max Features = 1 |
|---|---|---|
| Accuracy | 83.03% | 84.77% |
| Precision | 49.04% | 54.94% |
| Recall | 46.05% | 47.80% |

**5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric item: "validation strategy"]**

Validation is the process of testing a machine learning algorithm and verifying performance. If I train and test a machine learning algorithm on the same set of data, the results may not be meaningful.

An important method of testing machine learning classifiers is to separate a dataset into separate training and testing data. This is called Cross-Validation. However, even with cross validation, it is a common mistake to train on one kind of data and test on another.

For example: If our POI employees were at the bottom of the list, and we used the top 80 rows (all non-POI employees) to train our algorithm, then our training data would not include any persons of interest and our algorithm may just assume that all employees are not persons of interest.

To validate my analysis, I used the sklearn StratifiedShuffleSplit class that creates multiple, diverse sets of training and testing data. For my tests, I use the default setting of 90% training data and 10% testing data. By specifying 1000 'folds', the classifier runs for 1000 'random' sets of training and testing data each time I want to calculate the precision, recall, and accuracy scores of my classifier. Since the results cover a wide range of test cases, I reduce the risk of any one scenario (too many or not enough POIs in the training set) impacting my scores.

**6. 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. [relevant rubric item: "usage of evaluation metrics"]**

My tuned classifier performed with ~85% accuracy, ~55% precision, and ~48% recall.

The accuracy score means that each time the classifier attempted to predict whether a test data Enron employee was a POI (or not), it was correct 85% of the time.

The precision score means that each time the classifier predicted a test data employee was a person of interest, it was correct 55% of the time.

The recall score meant that of the persons of interest in the testing data, the classifier was able to identify 48% of them.

My goal with this exercise was to identify a classifier with a high recall score which I thought to be the most important. When scanning data for fraud, I believe it is important to catch a high percentage of fraud cases even if the accuracy is lower than a different algorithm. I believe this particular classifier is strongly balanced and demonstrates high scores for all three evaluation metrics.

Sources:

Lou Pai - https://en.wikipedia.org/wiki/Lou_Pai