Udacity Data Analyst Nanodegree

P5 Final Project - Identify Fraud from Enron Mail

Allan MacGowan
March 30, 2016

Section 0: Introduction

The goal of this project is to investigate and identify Enron employees who were possibly involved in the scandal that became public in October 2001. As a direct result of widespread corporate fraud, Enron Corporation would eventually file for bankruptcy in one of the largest reorganizations the United States has ever seen. This project will show how various machine learning techniques can be applied to publicly available financial and email data from Enron in order to build a model that predicts which employees should have been regarded as persons of interest (POI).

Section 1: Understanding the Dataset & Overall Question

The dataset for this project covers 145 Enron employees (and one more entry called TOTAL).

Details for each employee are organized according to 21 features:
['salary', 'to_messages', 'deferral_payments', 'total_payments', 'exercised_stock_options', 'bonus', 'restricted_stock', 'shared_receipt_with_poi', 'restricted_stock_deferred', 'total_stock_value', 'expenses', 'loan_advances', 'from_messages', 'other', 'from_this_person_to_poi', 'poi', 'director_fees', 'deferred_income', 'long_term_incentive', 'email_address', 'from_poi_to_this_person']

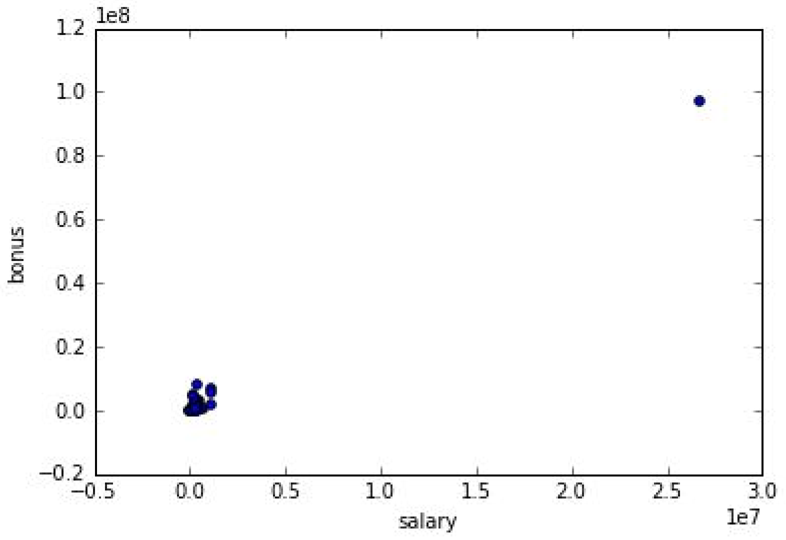
Of the 146 entries, 18 are identified as POIs and 128 are not POIs as indicated by the ‘poi’ field being True or False respectively. There were no ‘NaN’ (not a number) entries for the ‘poi’ field.

However, for other fields, there are a lot of missing entries. In total, there are 1358 ‘NaN’ entries in the dataset broken down by feature as follows:
{'salary': 51, 'to_messages': 60, 'deferral_payments': 107, 'total_payments': 21, 'loan_advances': 142, 'bonus': 64, 'email_address': 35, 'restricted_stock_deferred': 128, 'total_stock_value': 20, 'shared_receipt_with_poi': 60, 'long_term_incentive': 80, 'exercised_stock_options': 44, 'from_messages': 60, 'other': 53, 'from_poi_to_this_person': 60, 'from_this_person_to_poi': 60, 'poi': 0, 'deferred_income': 97, 'expenses': 51, 'restricted_stock': 36, 'director_fees': 129}

The list of POIs is as follows:
['HANNON KEVIN P', 'COLWELL WESLEY', 'RIEKER PAULA H', 'KOPPER MICHAEL J', 'SHELBY REX', 'DELAINEY DAVID W', 'LAY KENNETH L', 'BOWEN JR RAYMOND M', 'BELDEN TIMOTHY N', 'FASTOW ANDREW S', 'CALGER CHRISTOPHER F', 'RICE KENNETH D', 'SKILLING JEFFREY K', 'YEAGER F SCOTT', 'HIRKO JOSEPH', 'KOENIG MARK E', 'CAUSEY RICHARD A', 'GLISAN JR BEN F']

There is an outlier in the data, as the scatterplot in Figure 1 shows:

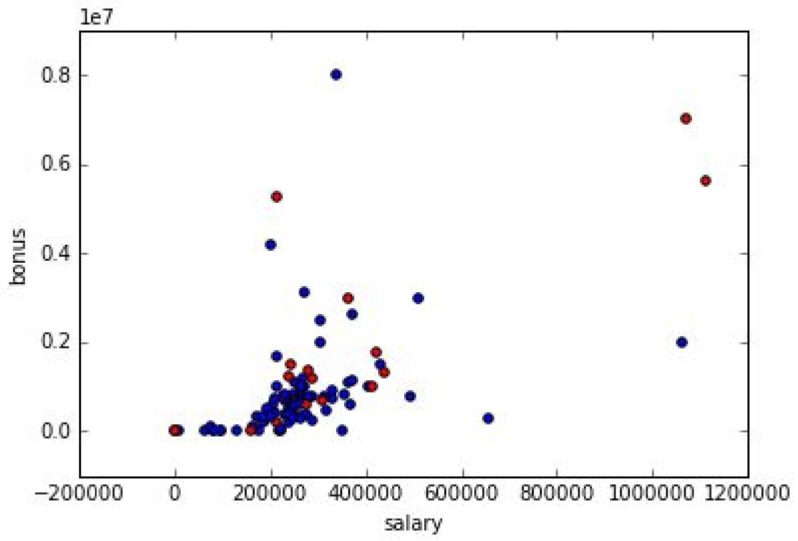
Figure 1 - Scatterplot of Bonus vs. Salary Prior to Outlier Removal



This outlier on the top right corner corresponds to the name ‘TOTAL’, which can safely be removed because, unlike all other data points, it does not correspond to an actual person but rather is an aggregate view of all persons in the dataset.

After removal of this outlier, the scatterplot in Figure 2 becomes:

Figure 2 - Scatterplot of Bonus vs. Salary After Outlier Removal (POIs in Red)



We still see several outliers to the far right and in the top section of Figure 2. However, these outliers may be meaningful, as several of them correspond to key POIs (indicated by red circles) within the company, including SKILLING JEFFREY K and LAY KENNETH L, and may therefore be helpful in building the POI identifier.

Section 2: Feature Selection & Engineering

The following existing features from the dataset were initially selected for the POI identifier:

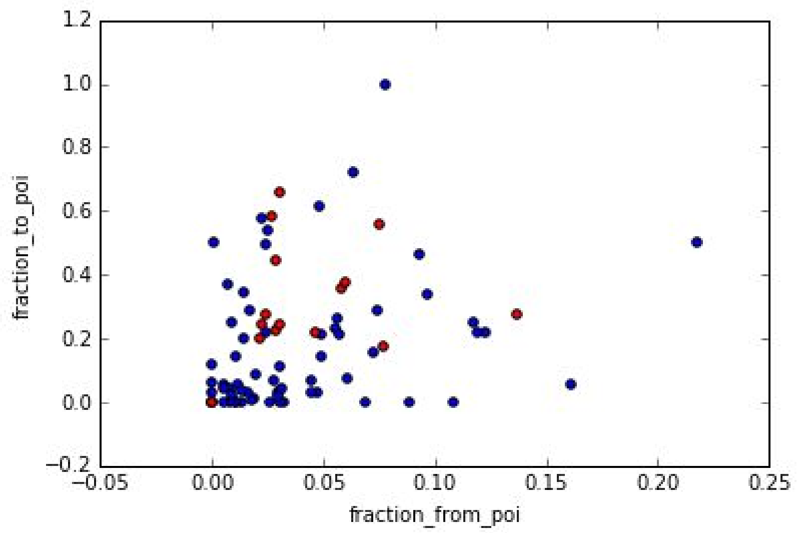
features_list = ["poi','salary', 'bonus', 'total_payments', 'exercised_stock_options', 'restricted_stock', 'total_stock_value"]

This selection was based on the suggested correlation between a person who is a POI and a person having a value significantly above the mean for at least one of these features (as indicated by the Payments to Insiders pdf document).

Two new features were added to the dataset: ‘fraction_from_POI’ and ‘fraction_to_POI’.
‘fraction_from_POI’ = ‘from_poi_to_this_person’ / ‘to_messages’
‘fraction_to_POI’ = “from_this_person_to_poi’ / ‘from_messages’

These features were added under the reasonable assumption that people who communicate frequently with one or more POIs may themselves be POIs. Existing features ‘from_poi_to_this_person’ and ‘from_this_person_to_poi’ already indicate the number of emails to/from a POI, but the two new features go a step further by calculating these totals as a proportion of all to/from email messages. The rationale is that people with a higher percentage of email communication to/from POIs are in turn more likely to be POIs.

Figure 3 - Scatterplot of New Features (POIs are Red)



Other feature combinations were investigated as well to optimize performance of the POI.
Combination #1:
features_list = ['poi', 'bonus', 'exercised_stock_options', 'restricted_stock', 'total_stock_value', 'fraction_from_poi', 'fraction_to_poi']
#Result: recall=0, precision=0, accuracy < 0.86 for all algorithms

Combination #2:
features_list = ['poi','salary', 'to_messages', 'deferral_payments', 'total_payments', 'exercised_stock_options', 'bonus', 'restricted_stock', 'shared_receipt_with_poi', 'restricted_stock_deferred', 'total_stock_value', 'expenses', 'loan_advances', 'from_messages',

'other', 'from_this_person_to_poi', 'director_fees', 'deferred_income', 'long_term_incentive', 'from_poi_to_this_person', 'fraction_from_poi', 'fraction_to_poi']
#Result: accuracy = 0.87 for all algorithms, precision = 0.5 for all three, and recall = 1 for decision tree and random forest, recall = 0.5 for Naive Bayes

Combination #3:
features_list = ['poi','bonus', 'restricted_stock', 'expenses', 'deferred_income', 'fraction_from_poi', 'fraction_to_poi']
# Result: recall, precision = 0 for decision tree and random forest, recall = 0.5, precision = 0.33 for Naive Bayes, accuracy = 0.71 to 0.86

The final feature list yielded the best results for the decision tree algorithm (see Section 4) with the least number of features.

Final feature list:
features_list = ['poi', 'bonus', 'total_payments', 'exercised_stock_options', 'restricted_stock', 'fraction_from_poi', 'fraction_to_poi']

For the decision tree classifier, the feature_importances_ attribute highlighted 'bonus' and 'exercised_stock_options' as most important:
['bonus', 'total_payments', 'exercised_stock_options', 'restricted_stock', 'fraction_from_poi', 'fraction_to_poi']
[0.30550501725616924, 0]
[0.094505494505494475, 1]
[0.33789131260444638, 2]
[0.033190993788819949, 3]
[0.1020408163265306, 4]
[0.1268663655185393, 5]

However, removing some of the less important features impacted performance of the classifier, so they remained in the feature list.

Feature scaling was not implemented, because none of the selected algorithms (decision tree, random forest and Naive Bayes) are affected by feature scaling (unlike SVMs, for example).

Section 3: Algorithm Selection & Tuning

The output of our POI identifier is a discrete set of labels (1 if a POI, 0 if not). Hence, it is appropriate to apply various supervised classification algorithms to the dataset and select the classifier with the best overall results (i.e. recall, precision and accuracy values closest to 1).

Many machine learning algorithms, such as decision trees and random forests, rely on a number of parameters as part of the learning process, such as the maximum number of features to include, the maximum depth of a tree and many more constraints. Varying these parameter settings can have a significant impact on a classifier's performance, depending on the size of the dataset, type of data and a number of other factors. Hence, when selecting an algorithm it is important to tune various parameters, which involves testing various combinations to arrive at a set of values that optimizes the classifier's performance. This process of finding the optimal set of parameters can be done manually (as shown below for the random forest classifier) or it can be automated through techniques such as GridSearchCV (as shown below for the decision tree classifier).

The following algorithms were trained and tested with parameter tuning performed on the latter two classifiers. Results were compared to arrive at a final selection.

#1 Naive Bayes Classifier
Results were as follows:
{'True Neg': 12, 'False Neg': 1, 'False Pos': 1, 'True Pos': 1}
{'Recall': 0.5, 'Precision': 0.5, 'Accuracy': 0.8666666666666667}

#2 Random Forest Classifier
Parameter tuning was performed manually, with results shown in the table below.

Table 1 - Results of Manual Parameter Tuning for Random Forest Classifier

Parameter Name	Combination #1	Combination #2	Combination #3	Combination #4
n_estimators	1,000	10	10,000	1,000
min_samples_split	50	10	100	100
criterion	entropy	entropy	gini	gini
min_samples_leaf	50	10	100	100
bootstrap	False	False	False	True
Results				
Recall	0.5	0.5	0.5	0.5
Precision	1.0	1.0	0.5	1.0
Accuracy	0.93	0.93	0.87	0.93

Combination #2 was selected as the set of parameter tunes, since it matched other combinations in performance but required the fewest number of estimators (trees in the forest), which translates into less training time and processing than other combinations.

#3 Decision Tree Classifier
The final algorithm selected for the POI identifier was a decision tree classifier due to its superior performance. Results are provided in Section 4. Parameters were tuned as follows using GridSearchCV:
parameters = {'criterion':("gini", "entropy"), 'splitter':("best", "random"), 'max_features':("auto", "sqrt", "log2")}

Section 4: Validation & Evaluation

Validation of the POI against test data is an important step because it provides an estimate of the classifier’s performance on an independent data set and helps identify any overfitting. In splitting the data into training and testing sets, the goal is to optimize both in order to ensure the best results from training as well as the best validation.

Two options for splitting data into training and testing tests were attempted using cross validation: train_test_split and StratifiedShuffleSplit. The latter provided better performance and was used for the final validation and evaluation. Using train_test_split, the decision tree POI had inferior performance as shown:

```
Decision Tree Classifier Results with train_test_split cross validation
{'True Neg': 34, 'False Neg': 4, 'False Pos': 3, 'True Pos': 2}
{'Recall': 0.3333333333333333, 'Precision': 0.4, 'Accuracy': 0.8372093023255814}
```

Using StratifiedShuffleSplit, with parameter tuning highlighted in Section 3 and features selected in Section 2, the Decision Tree algorithm provided the best performing POI classifier. To measure performance, the following metrics were used:

Recall measures the number of correctly identified POIs out of the total number of actual POIs.

Recall = # of true positives / (# of true positives + # of false negatives)

A recall value closer to 1 indicates that the POI classifier does a good job of minimizing the number of people who are actually POIs but are identified as non-POIs.

Precision measures the number of correctly identified POIs out of the total number of predicted POIs.

Precision = # of true positives / (# of true positives + # of false positives)

A precision value closer to 1 indicates that the POI classifier does a good job of minimizing the number of people who are falsely identified as being POIs.

Accuracy measures the number of people correctly identified as either POIs or not out of the total number of people.

Accuracy = (# of true positives + # of true negatives) / (# of total predictions)

An accuracy value closer to 1 indicates that the POI classifier does a good job of minimizing the number of people falsely identified either as a POI or non-POI.

Results of the selected algorithm are given below. The test set consists of 15 employees, representing approximately 10% of the total number of employees. In two of the three iterations, precision, recall and accuracy are all 1, so there are no false positives or false negatives.

Decision Tree Classifier Results

```
Iteration #1
{'True Neg': 13, 'False Neg': 0, 'False Pos': 0, 'True Pos': 2}
{'Recall': 1.0, 'Precision': 1.0, 'Accuracy': 1.0}

Iteration #2
{'True Neg': 12, 'False Neg': 1, 'False Pos': 1, 'True Pos': 1}
{'Recall': 0.5, 'Precision': 0.5, 'Accuracy': 0.8666666666666667}

Iteration #3
{'True Neg': 13, 'False Neg': 0, 'False Pos': 0, 'True Pos': 2}
{'Recall': 1.0, 'Precision': 1.0, 'Accuracy': 1.0}
```

Appendix: References

The following resources were cited in this project.

https://en.wikipedia.org/wiki/Enron_scandal
http://stackoverflow.com/questions/402504/how-to-determine-the-variable-type-in-python
http://scikit-learn.org/stable/modules/feature_selection.html
http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedShuffleSplit.html
http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html
http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB
http://stackoverflow.com/questions/22903267/what-is-tuning-in-machine-learning

I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.