Senay Goitom
Udacity Machine Learning Final Project
14 October 2017

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 of this project is to predict persons of interest in the Enron Fraud case based on email correspondence and detailed financial data that was made public for top executives. More broadly the data provided allows us to use machine learning techniques to detect patterns that are predictive of behavior associated with persons of interest. The financial features provide information about the incentives and financial rewards that motivated potentially criminal behavior, and the email features provide information about communications within the network of executives that may have included persons of interest.

The dataset has a total of 146 observations (determined by the length of the dictionary form of the dataset). There are eighteen Persons of Interest (POIs) and 128 non-POIs. Each potential POI has 21 features (fourteen financial features, six email features and the POI indicator).

Plotting salary and total stock value, there was a clear outlier in the financial features. Consulting the Enron insider pay document it became clear this was "TOTAL" row, which is not an actual potential POI. This outlier was removed from the dictionary prior to feature formatting.

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", "intelligently select features", "properly scale features"]**

Using the SelectKBest module, the following features were selected:

['salary', 'total_payments', 'loan_advances', 'bonus', 'deferred_income', 'total_stock_value', 'exercised_stock_options', 'long_term_incentive', 'restricted_stock', 'shared_receipt_with_poi']

The performance of the classifier based on this selection method (using the GaussianNB() classifier) was:

Accuracy: 0.77256 Precision: 0.47555 Recall: 0.22850

In an attempt to improve upon the performance above, I added the following generated features: shares of emails from an individual to a POI, and vice versa (including shared receipt). I used shares, rather than numbers of emails to normalize the intensity of communication across individuals. Since these features varied considerably in absolute value, I used a minmax scaler. Again using SelectKBest, the following features were selected:

['salary', 'total_payments', 'loan_advances', 'bonus', 'deferred_income', 'total_stock_value', 'exercised_stock_options', 'restricted_stock', 'pct_email_shared_reciept', 'pct_email_to_poi']

Using the GaussianNB() classifier, the features above performed as follows:

Accuracy: 0.77578 Precision: 0.49026 Recall: 0.22650

This performance did not meet the minimum requirements for precision and recall. I pared down the selection of features manually in an attempt to achieve better performance. I based my selection of the financial features on the fact that the particulars of Enron executive compensation likely motivated illegal behavior[1]. Enron executives placed a great deal of importance in the company's stock price, and compensation for senior executives in particular was tied to the stock price in the form of options. In the end I settled on the following features:

['salary', 'bonus', 'total_stock_value', 'pct_email_to_poi', 'pct_email_shared_reciept']

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 began with a Naïve Bayes classifier. Using the final features, the GaussianNB() classifier performed as follows:

Accuracy: 0.80944 Precision: 0.68015 Recall: 0.26900

Using the Support Vector Machine Classifier on the final feature selection yielded the following results:

Accuracy: 0.78322 Precision: 0.51548 Recall: 0.40800

[1] https://en.wikipedia.org/wiki/Enron_scandal#Executive_compensation

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? What parameters did you tune? (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 items: "discuss parameter tuning", "tune the algorithm"]**

Tuning the parameters of an algorithm simply means adjusting the parameters that determine the shape of the decision boundary. Poor, or no, tuning can lead to overfitting, or suboptimal classification of the training datasets. I tuned the parameters following parameters of the SVM: kernel, gamma, and C. In particular I used a grid search over the following parameter space:

```
param_grid =    [       {'SVC__C': [1, 10, 100, 1000, 10000],
                         'SVC__kernel': ['linear']
                        },
                        {'SVC__C': [1, 10, 100, 1000, 10000],
                         'SVC__gamma': [1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0],
                         'SVC__kernel': ['rbf']
                        }
                ]
```

The parameter combination that performed the best was the following:

{'SVC__gamma': 1.6, 'SVC__kernel': 'rbf', 'SVC__C': 1000}

These were the parameters used in testing.

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 items: "discuss validation", "validation strategy"]**

Validation is a way of estimating the performance of the algorithm on an independent dataset. It also serves as a check on overfitting. A classic mistake is overfitting. If the training dataset is too large relative to the testing dataset, then the algorithm may perform poorly on the testing dataset. I used train_test_split function from sklearn's cross_validated package. In this case, cross validation was done using the StratifiedShuffleSplit package. This package is used to insure that each fold has a similar percentage of POIs as the entire sample. Without stratification, the imbalanced nature of the POI/non-POI distinction (few POIs) makes it likely that a simple random shuffle split will contain folds in which there are no POIs.

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"]**

The evaluation metrics I used are Precision and Recall. The performance for precision was 0.51548, and for recall it was 0.408. Recall is the probability that the algorithm correctly identified a POI, given that the person was, in fact a POI. Precision is the probability that, given that the algorithm identified a POI, that the person is actually is a POI.