Josh Johnson
Udacity Project P5 Report
Intro to Machine Learning Final Project
Identify Fraud from Enron Email/Financial Data

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?

The purpose of this project was to identify individuals who were persons of interest (POI) in the Enron scandal of the early 2000's. POI were either individuals who were indicted for fraud, settled, or were granted immunity in exchange for information. In much the same way that machine learning is used in the banking industry to identify suspicious behavior, it is a good tool to determine if the features at our disposal can identify Enron POI as well. The data set consists of a combination of email statistics as well as financial data of 146 Enron employees. In all, there are 18 individuals in the dataset that are identified as POI and 128 identified as non-POI. There are 14 financial features and 6 email features. These features are likely to be indicative of POI since we would expect that the POI would have been in more regular email communication with each other and their financial activities would have been similar. In addition, we should expect that their email and financial behaviors should have been anomalous with respect to the rest of the population.

This problem is best described as a supervised learning problem, meaning that we have a labeled dataset of features and POI identity that can be used to train and test a supervised algorithm. Once the model is trained, it can be used to identify any new individuals as POI or non-POI. As such, we do not want to train our model on outliers that could skew our model's performance. Therefore, outliers were first identified and removed. The process described in the Outliers lesson works with linear regression and takes an sequential approach to removing the data points that have the largest absolute error between the data point and the regressed line. However, this project is a binary classification problem and the error will be either 1 or 0 and, therefore, a different approach must be taken. I decided to use a univariate approach in which I tallied the mean and standard deviation for each feature and then compared each individual's features to 3 standard deviations above the mean. This method allowed me to detect any unusual entities in the data, such as "TOTAL." I also listed the names of individuals to determine if there were any unusual names and I found 2 names that appeared to be entities that were not humans - "TOTAL" and "THE TRAVEL AGENCY IN THE PARK."

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.

I decided to use the automated feature selection algorithm SelectKBest to determine the best parameters for my classifier. I used the Pipeline operator along with GridSearchCV function to determine the optimal parameters by giving a range over which GridSearchCV could try each parameter. The scores for each feature are:

'Bonus' = 31.1
'Exercised_stock_options' = 23.9
'Total_stock_value' = 23.3
'Salary' = 18.8
'Fraction_to_poi' = 15.5
'Shared_receipt_with_poi' = 14.2
'Deferred_income' = 10.6
'Long_term_incentive' = 10.3

I also decided to create two features, “fraction_from_poi” and “fraction_to_poi”. The former represents the fraction of emails that an individual sent to a POI over the total number of emails that the person sent. The latter represents the fraction of emails that an individual received from a POI over the total number of emails that the person received. These features were created because it seems more characteristic for POIs to have a greater proportion of their individual emails passed to each other than to those that are not POIs. This effectively normalizes the number of emails to and from POIs to the total number of emails and should be able to help differentiate POIs from those that simply send a large quantity of emails in general.

I did use rescaling with my parameters because the scales between the numbers of emails (~100s-1000s) or proportions of emails (0-1) are orders of magnitude different compared to salaries and bonuses. While Naive Bayes does not require re-scaling, this scaling difference makes it numerically challenging to find hyper-parameters which generalise across environments.

3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

I tried several different algorithms from Naive Bayes to AdaBoost to Decision Trees and more. Overall, I got the best performance from Naive Bayes when used with SelectKBest to determine the proper number of features. The goal was to achieve better than 0.3 for both precision and recall. Naive Bayes achieved 0.45 precision and 0.37 recall while the Decision Tree achieved 0.24 precision and 0.19 recall.

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).

Tuning the parameters of a machine learning algorithm allows one to optimize the performance of the model for a particular dataset. Typically there are tradeoffs between parameters such as the min_samples_split which determines the minimum number of data samples required to split an internal node. Making this variable small can lead to overfitting, while making it large may lead to a biased model. While Naive Bayes doesn't have parameters to tune, I did use GridSearchCV along with SelectKBest to determine the best number of features. I also used GridSearchCV and PCA with the Decision Tree and also performed a parameter search to find the optimal classifier.

5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation is a method in which you withhold part of the dataset for the purpose of independently determining the accuracy of the model. For instance, one would train their model on 80% of a dataset and test on the remaining 20%. If this isn't done, then the model is essentially trained and tested against the same data which leads to overfitting and an inability to generalize to new data.

I used StratifiedShuffleSplit in sklearn which essentially indexes the dataset and maintains the same proportion of labels in both the training and testing data. This function will perform a set number of folds in which it randomly shuffles and splits the data as described above. This method ensures that our training set does not use all of the POIs while leaving none for the test set due to the fact that our dataset is small.

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.

I used precision and recall as performance evaluation metrics for this project. Using Naive Bayes I got a precision of 0.45 or 45% and a recall of 0.37 or 37%. The precision results can be interpreted such that for all predictions that were made for the POI class, the model was correct 45% of the time. Additionally, the recall results can be interpreted such that for all actual POI instances in the data, the model correctly identified them 37% of the time.

Reference Material
https://discussions.udacity.com/t/outlier-removal-with-a-classification-algorithm/16573
https://discussions.udacity.com/t/error-using-pipeline/171750/4

https://discussions.udacity.com/t/i-cant-achieve-an-accuracy-score-above-0-3/180098/8