**Identifying Fraud from Enron Email**

Adam MacDonald

Udacity- Intro to Machine Learning Final project

**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 for this project is to identify Enron employees, who may have committed fraud, as persons of interest (POI’s). For the context of this project, a POI is defined as an individual who was indicted, reached a settlement or plea deal with the government, or who testified in exchange for prosecution immunity. Two datasets were made available for this analysis, one being the financial data for each individual (where available) involved in the Enron scandal, and the second consisting of email content and summary data from Enron employees. Machine learning is useful for accomplishing this goal since it is an effective way to analyze and predict real world data utilizing different, available, algorithms and tuning these algorithms to best represent the end goal.

As mentioned, the dataset consists of two different sources, where the financial data may be useful in identifying POI’s by how much they were earning from different sources (salary, bonus, stock, ect.) while employed by Enron and their email data may give clues based on who was in contact/associated with any known POI’s from both an email count or an email body content/keyword point of view. The data set was considerably imbalanced with 18 POI’s out of 145 data points, which made it difficult to eliminate data from any POI’s based on outlier detection. There were a few outliers identified, one being Kenneth Lay, who for a few financial features represented the max value by a large margin; for example his total payments were 46 times larger than the mean. While obvious outliers were identified, my algorithm performed worse when removing them, and with the limited number of POI’s I decided to leave them in my dataset. Aside from the outliers, there was also some data validation necessary. Two individuals (Robert Belfer and Sanjay Bhatnagar) had some of their financial data incorrectly entered and were manually adjusted based on the Enron, FindLaw, financial data. Two entries in the dataset also had to be removed, “Total” and “The Travel Agency in the Park”, since neither of these represented a “person” of interest, which is the goal of this project.

**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 doesn’t 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.) If you used an algorithm like a decision tree, please also give the feature importance of the features that you use. [Relevant rubric items: “create new features”, “properly scale features”, “intelligently select feature”]**

For my POI identifier I chose to use 4 features; total payments, exercised stock options, other, and ratio of emails the individual sent to a known POI compared to total emails sent. I reached these final four features by starting with the complete list of features and using an AdaBoost classifier to identify the least important features and eliminate them. During this process I created a few new features with the data, two of them being ratios which represented the number of emails sent to/from a POI compared to total emails sent/received. This gave a more normalized representation of the to/from POI emails than a general count. The other two features I created occurred further in the feature selection process, when I had the list narrowed down to the final features, as I investigated whether it would be better to lump the three financial features together as a sum or a product compared to having them as separate features. Neither of these new features showed better results with the model, and as such were disregarded. My final algorithm did not require feature scaling as the AdaBoost classifier does not use a distance measure in the model, it uses decision trees to divide the features into categories and therefore feature scaling wouldn’t impact my algorithm results. Each feature selection iteration was documented by comments in the poi\_id.py file to map my thought process, and below is the feature importance’s of my final algorithm using an AdaBoost classifier.

Total Payments = 0.224

Exercised Stock Options = 0.191

Other = 0.365

To POI Ratio = 0.220

**3. What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: “pick an algorithm”]**

I ended up using an AdaBoost classifier as my algorithm, which was tuned to use a decision tree with the criterion equal to “entropy” as a base estimator, “SAMME” as the algorithm, and 50 as the number of estimators. I tried a few different algorithms before landing on this one. The first I tried was a Naïve Bayes classifier, which resulted in a reasonable precision of roughly 0.3, but the recall using this algorithm was really low measuring at under 0.2, based on the final feature selection. I then opted to try a few variations of decision trees, while using sklearn’s GridSearchCV, and the results were promising with a precision around 0.5 and recall around .45. The next classifier tested was RandomForest which showed much worse performance than the decision tree classifier when it came to recall, with a measure of less than 0.2. The final algorithm I looked at, and the one I selected to use was the AdaBoost classifier, where I incorporated the findings from my previous decision tree classifier as the base estimator. This resulted in similar precision and recall to the decision tree algorithm, however, its’ recall was slightly higher on a consistent basis and I was preferential to the more accurate recall as I felt it was the more important metric for the goal of this project, since I wanted to ensure as many known POI’s as possible were identified by the algorithm.

**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 don’t 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 if you used, say, a decision tree classifier). [relevant rubric item: “tune the algorithm”]**

Many machine learning algorithms/classifiers have parameters that can be altered in order to find a better fit for the dataset, or to ensure the data isn’t being over/under fit. I tuned my algorithm at two different stages, by first using GridSearchCV on a decision tree classifier to determine the optimal parameters (based on the ones I investigated) for that algorithm. I then used this tuned decision tree as the base estimator in an AdaBoost classifier, where I tuned a couple more of its parameters by iterating through GridSearchCV. Below is my final algorithm, with parameter tuning.

tree = tree.DecisionTreeClassifier(criterion='entropy')

clf = ensemble.AdaBoostClassifier(base\_estimator=tree, algorithm='SAMME', n\_estimators=50)

**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 a technique used to automate the testing of a machine learning algorithm to new data, as opposed to purely assessing an algorithm on the training data. One classic mistake that could be made by applying this technique incorrectly is when datasets are sorted in a way that groups specific characteristics. For example, this was seen during our Udacity lectures where the email data from Chris and Sara were ordered to group Sara’s emails first, then Chris’. Therefore, if you were to split the data at the midpoint, and use one for testing and one for training, the algorithm would fit to either Chris/Sara’s data, but would be used to predict the other individuals and would prove to be a very poor algorithm.

I validated my analysis using two different methods; the first being the stratified shuffle split cross validation provided in the starter code, and the second being stratified k fold. Both of these validation methods were used in functions to display the accuracy, precision, recall, F1, and F2 metrics, along with the total predictions, true positive, false positive, false negative, and true negative counts of the algorithm being tested. As expected, the stratified shuffle split resulted in much higher accuracy, precision, recall, F1 and F2 metrics as it creates more data points to train and test the dataset on compared to the k-fold cross-validation.

**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 three main metrics I used to assess my algorithm were its accuracy, precision, and recall. The accuracy measures how well the model predicts whether an individual is or isn’t a POI. Precision measures how confident we are that an individual is a POI, when the algorithm predicts the individual is a POI. Finally, recall measures how confident we are that a known POI will be identified by the algorithm. The average metrics of my algorithm are listed below, based on running the algorithm with the tester 10 times.

Accuracy = 0.861

Precision = 0.514

Recall = 0.456