Christian Rivera

Udacity Data Analyst Program

Project 5: Identify Fraud from Enron Email

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 was to design a machine learning algorithm that could predict whether or not an Enron employee should be a person of interest in regards to Enron's fraud case. Given a list of Enron employees, their financial data, their email data, and whether or not each employee was a person of interest, the machine learning algorithm should be able to reasonably predict if someone was a person of interest based on indicators from certain financial and/or email features. Considering the machine learning algorithm is trying to predict conspirators in an incredibly large fraud case, such features like extremely high salaries, expenses, and/or stock options associated with specific individuals could be signs that those individuals are persons of interest as they would have a larger incentive to commit fraud in order to gain/retain their current fortunes. Another indicator could be large number of email correspondences between individuals and whether or not those emails contain phrases or words that would indicate fraud.

There were several outliers in the dataset including an employee listed as 'TOTAL' which was just the sum of the data from the rest of the dataset. While it is understandable why this 'TOTAL' row was in the dataset, it isn't an actual employee and has feature values so much greater than any other row entry, that it distorts any predictions that could be made. As such, it was removed from the dataset. Also, a lot of entries had features with the value 'NaN'. Considering that these 'NaN' values were for features that generally associated with only those in senior positions within a company (for example, 'restricted_stock_deferred'), I made the assumption that if the value was 'NaN', it was actually just 0. As such, all 'NaN' values were replaced with zeroes.

Finally, there were two incomplete records in the data set, 1 for "Robert Belfer" and the other for "Sanjay Bhatnagar" that needed to be added in.

Total number of data points: 147

Total number of POI's: 18

Total number of non-POI's: 129

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 created 3 metrics:

a. "true_total_payments" which is the sum of salary, bonus, long term incentive, deferred income, deferral payments, loan advances, other, expenses, and director fees. "total_payments" didn't actually include some of these values.
b. "true_total_stock_value" which is the sum of exercised stock options, restricted stock, and restricted stock preferred. "total_stock_value" didn't include restricted stock preferred which was an oversight so a new value needed to be created.
c. "total_compensation" which is the sum of "true_total_payments" and "true_total_stock_value". This feature was created because the people who were paid the most, regardless of how they were paid, had the most to gain from cooking the books and so this feature reflects the

Since the algorithm with the best raw performance on the total features list, was the AdaBoost classifier, I chose that one for my algorithm. Since AdaBoost is a type of decision tree classifier, I used the "feature_importances_" function to determine which features. Of the 22 features (14 financial features, 5 email features, and 3 created financial features), only five features recorded a non-zero "feature_importances_" score. Those features are:

1. Bonus: 0.095
2. Salary: 0.148
3. Expenses: 0.048
4. Total compensation: 0.047
5. Total stock value: 0.371

I did not have to do any feature scaling as all of the algorithms I chose to try do not use Euclidean distance in their formulas and therefore feature scaling would not be useful.

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"]

The algorithm I ended up using was the AdaBoost Classifier. I also tried a Decision Tree Classifier, Gradient Tree Boosting Classifier, and the Naïve-Bayes algorithm. The reason why I chose the AdaBoost Classifier was that, none of the algorithms originally reach 0.3 or greater recall and precision. I then applied feature tuning to the more complicated classifiers (AdaBoost, Gradient Tree) and found that Adaboost had the highest scores as well as the requisite greater than 0.3 precision and recall scores.

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"]

To "tune the parameters of an algorithm" is to adjust certain aspects of a machine learning algorithm so that it does not overfit or underfit the data and provide misleading results. Overfitting is when an algorithm is tuned so sensitively that it mistakes data noise as important features. Because the noise is random and does not reflect the underlying relationships, trying to make a prediction based on noise would be useless and as such the algorithm would be no good in predicting future data output. In contrast, underfitting the data is when the algorithm is tuned to such that it ignores actual relationships among data points and therefore does not know the real pattern among the points nor can accurately predict any new points. For the algorithm I used, the AdaBoost classifier, I tuned two paramters: n_estimators and learning_rate. N_estimators is the number of classifiers applied to the data. Each subsequent classifier focuses on the false classified instances the previous classifier made. Learning_rate represents the diminishing amount of effect each subsequent classifier or "booster" impacts the overall algorithm. Finding a balance between the two parameters is key to making sure the algorithm does not overfit or underfit the data. To determine what the balance is, I created a gridsearchCV process and inputted a range of values for both n_estimators and learning_rate for the AdaBoost Classifier and directed the function to find the optimal combination for the highest recall score. I decided that recall score should be the desired metric because it was the lowest score regardless of what the algorithm applied to the dataset was.

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 using the maximum amount of test data in order to verify whether or not an algorithm is accurate. There is a problem though because the data needs to be split between testing data for validation and training data in order to actually create the algorithm to test. Training data should also be maximized in order to create a theoretically accurate algorithm before testing. Shirking the size of the testing data for validation would lead to skewed test results, while shirking the size of the training data could mean that the algorithm hasn't properly weighted the features correctly. There are a finite number of data points to split between validating and training so in order to maximize both, I used a train_test_split cross validation which duplicates the test set repeatedly with different segments of data assigned to testing and training for each duplicate. As such every data point is used in both testing the algorithm and training the algorithm.

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"]**

- *Recall*: Recall is the ratio of how many times the algorithm correctly identified someone as a person of interest over the number of actual persons of interest there were in the dataset. (true positive / (true positive + false negatives). My algorithm's recall score is

30.4% which means the algorithm finds 30.4% of all the persons of interest in the dataset which means 69.6% of all persons of interest are still left unidentified.

- *Precision***:** Precision is the ratio of how many times the algorithm correctly identified someone as a person of interest out of all the people it identified as a persons of interest. The algorithm's precision score is 40.7% which means that 59.3% of the people that the algorithm identified as persons of interest were in fact not persons of interest.