**Enron POI Identifier Report**

**Introduction:**

In December 2001, Enron declared bankruptcy in what is one of the largest cases of corporate fraud in American history. As a result of the fraud, thousands of documents including emails and compensation history from Enron executives have been released to the public. The Enron corpus includes data for 141 Enron executives, 18 of whom have been identified as persons of interest. Through careful analysis of these documents, the aim is to create a machine learning classifier that can help identify persons of interest (POI), those who were indicted for fraud, settled with the government, or testified in exchange for immunity. Features used in this analysis will include measures of compensation, measures of connectivity among persons of interest, and expenses. This report documents the techniques used in building a POI identifier. Please note the steps taken in this analysis, while reported sequentially, did not follow an entirely linear procession.

**Load Data and Remove Outliers:**

The first step in building the POI classifier is loading the Enron data and verifying its accuracy. Once the Enron data was loaded, features were visualized to ensure there were no erroneous outlier data points. One data point became immediately obvious as it did not represent an Enron executive, but rather the total for all executives. This data point was subsequently removed and the data was visualized once again. Some outliers still remained, however a careful look at the data revealed that these points represented legitimate data.

**Create Features:**

Once the outliers were removed, two new features were added to the dataset. The new features, fraction\_from\_poi and fraction\_to\_poi, were created because it was reasoned that persons of interest may have a higher degree of connectivity among themselves. As a result these new features may have some predictive power when trying to identify persons of interest. After visualizing the new features, it seemed they may indeed have some predictive power. In subsequent steps, these features’ predictive power was verified by looking at feature imporances.

**Scale Features:**

After new features were added to the data set, the features were rescaled so that certain features with a large dispersion (ie bonus compensation), did not contribute a disproportionate amount of weight in subsequent classifiers. Though a random forest classifier was ultimately chosen, scaling features afforded greater flexibility when testing several classifiers. The features were rescaled using the MinMaxScaler from SKLearn.

**Split into Training and Testing:**

After scaling the features, the features and labels were split into training and testing sets using the train\_test\_split function in sklearn.cross validation. The test size was set at 10% of the data, and a random seed was set at 0. In subsequent iterations, the test size and random seed were altered.

**Train Algorithm and Tune Parameters:**

For this step, several algorithms were tested including a support vector machine and decision tree classifier. Some basic evaluations were used to get some high level insight into the performance of each classifier. A random forest classifier was ultimately chosen because it had a higher accuracy score, and there was no immediate evidence of overfitting. The parameters for the classifier were tuned using GridSearchCV.

**Remove Features:**

After training and testing the algorithm, the features were evaluated using the feature\_importances\_ function in SKlearn. Features that had an importance of less than 5% were removed and the classifier was run once again. These steps were repeated until all feature had an importance greater than 5%. It is also important to note that feature importances were inspected to ensure that no feature contributed a disproportionate amount to the classifier. If a feature had contributed too much, it would be cause for further inspection to ensure that the data was not self-affirming when searching for POIs.

**Do PCA:**

Principal component analysis was then used to combine features and reduce the dimensionality of the data. The number of principal components was varied (in subsequent steps) in order to provide the best data to the classifier. While an increase in PC’s did improve the performance of the classifier, many PC’s seemed to be the result of spurious correlations. The components\_ feature from SKlearn was valuable in determining which principal components were most valuable. When the number of principal components was low, the eigenfeatures made intuitive sense (ie one PC was strongly correlated to salary and bonus). As the number of PC’s increased there were some correlations that did not make intuitive sense. The number of PCs was ultimately reduced in order to prevent overfitting by removing those features that did not make intuitive sense.

**Re-Split Training and Testing Data:**

After conducting principal component analysis, the principal components and labels were split into training and testing sets, this time using StrattifiedKFold with 3 folds. This would help me get the same testing results as those that are used by the project evaluators.

**Re-Train Algorithm and Tune Parameters:**

Using the new training and testing data sets, the random forest algorithm was retrained and the parameters were tuned using GridSearchCV. The number of features was tuned manually by observing their importances, and removing those that did not contribute meaningfully to the classifier.

**Evaluation and Discussion:**

The final step in constructing the POI Identifier is evaluating the robustness of the classifier using several evaluation metrics: accuracy, recall, precision and F1 score. It is important to note that many of the preceding steps were revisited at this point in order to optimize the classifier’s performance while also ensuring the classifier did not overfit the data. After running the algorithm several times and tuning the inputs, the accuracy (measure of correct labels divided by total number of labels) ranged between 0.80 and 0.93. The mean accuracy was approximately 0.84. This suggests the algorithm performed reasonably well when trying to identify persons of interest.

The precision and recall scores, however, were cause for concern. Both recall and precision produced vastly different results when the algorithm was run multiple times. In some cases, the precision was 1.0, which suggests persons identified as a POI were almost certainly a POI. Other times the recall was 1.0 which suggests very few persons of interest were not identified as such. These numbers would be encouraging if not for the fact that they would drop as low as 0.0 when the algorithm was re-run.

Perhaps the reason for the wide range of precision and recall scores is the small number of people labeled persons of interest. The dataset is heavily skewed, and only 18 of 141 Enron executives are labeled persons of interest. By increasing the number of persons of interest in the dataset, the classifier’s performance would likely improve and we would feel more confident in the results.

**Amendments for Second Submission:**

In my prior submission, I had failed to get precision and recall above 0.3. This was due to the fact that I used a random forest classifier which, when not given enough training data, can have a high variance. I therefore tried a few less robust algorithms. A support vector machine still did not yield the desired results, but a random forest classifier yielded an average accuracy of 0.823, average precision of 0.345, and average recall of 0.444. This exercise was very helpful in gaining a deeper understanding of the bias-variance tradeoff, and will provide me with some good heuristics for future machine learning problems!

For this project I consulted the SKLearn documentation (frequently), stackoverflow.com (sometimes), and the Udacity discussion forum (very frequently).

“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..