Enron Final Project Report

1. The goal of this project is to find the best machine learning algorithm and the features that most accurately identify persons of interest in the Enron case. The provided dataset includes 146 rows (people) and 21 columns (features). However, of the features, I eliminated 'email_address' from the beginning as someone's email address will clearly not be a good indicator of whether that person was involved in a criminal action. Furthermore, 'poi' is one of the column titles but it is not really a feature; it is a label. Therefore, I used 19 features in my project. Also, it should be noted that there were 18 persons of interest and 128 non persons of interest. There were many features for which many individuals did not have any values entered in the data set. In fact, all columns except for 'poi' had some NaNs where 'total stock value' had minimum number of NaNs of 20 and 'loan_advances' had the maximum number of NaNs at 142. However, NaNs in this dataset simply means there was no value assigned to it in the "Payments to Insiders" document. In other words, NaNs really mean 0 in most cases. This explains the large number of NaNs in this dataset. Therefore, I decided to replace NaNs with 0 to best reflect the actual meaning of the data.

To look for outliers in the dataset, I drew scatter plots and bar graphs for visual check. The first thing that stood out was 'Total' which distorted the whole scatter plot. Therefore, I removed 'Total' first. I drew bar graphs dividing up persons of interest and non persons of interest. Two things were noticeable from the graphs of non POIs. First, they generally had lower payments and lower stock values. However, the second noticeable aspect of the non POIs was that there were some huge peaks, namely for Lou Pai, also known as the luckiest person in Houston. He was deeply involved in the Enron Operation, transferring EES losses into the wholesale division and etc., but luckily for him, he took off early due to his personal problem and never stood in court. It may be reasonably argued that he should be removed from the data set because he is someone who should have been labeled as a POI. However, I wanted to challenge my machine to even distinguish the difference between a bad guy who was put in jail and a bad guy who got away. So, I included all the points for Lou Pai and other controversial people. In general, I was conservative about removing outliers because the data set was small in size and looking into the details, the unusual points were more of useful information less of an error. Thus, I excluded only one more row from the data which was "The Travel Agency In the Park." The data for travel agency revealed no meaningful information.

2. Intuitively, I first thought that salary and bonus must be good choices for features; bad guys do bad things for money! However, running univariate analysis made me realize that there were some lucky employees who got off with no charges but huge sums of money. Then I realized that other features such as messages from poi and message to poi may also be strong indicators of whether someone is a poi or not. Running univariate analysis revealed that messages from and to poi should be represented as a fraction because absolute numbers could be misleading. Therefore, I created new features, “fraction to poi,” “fraction from poi,” and “fraction shared with poi.” In the end, I used two of the three features I made: “fraction to poi” and “fraction shared with poi.” Creating these new features was very helpful; the f score I got running only the given features was a mere 0.27186 (precision: 0.27173, recall: 0.27200) but my final f score using the features I made was 0.39232 (precision: 0.36505, recall: 0.42400). In fact, I would not have passed the project requirement if I had not created new features.

Still, scaling the features was absolutely necessary because many features had values that cannot be compared directly. Thus, I scaled the features using Min/Max scaler and selected the most powerful features using SelectKBest. After using SelectKBest, I used Grid Search CV to find the optimal number of features. Ultimately, I used the following twelve features and the scores and p values for those features are as follows:

```
('exercised_stock_options', 24.815079733218194, 1.8182048777865317e-06)
('total_stock_value', 24.182898678566879, 2.4043152760436886e-06)
('bonus', 20.792252047181535, 1.10129873239521e-05)
('salary', 18.289684043404513, 3.4782737683651706e-05)
('fraction_to_poi', 16.409712548035792, 8.3889533567042162e-05)
('deferred_income', 11.458476579280369, 0.00092203670846723062)
('long_term_incentive', 9.9221860131898225, 0.0019941812453537077)
('restricted_stock', 9.2128106219771002, 0.002862802957909168)
('fraction_shared_with_poi', 9.1012687391935039, 0.0030312943103536103)
('total_payments', 8.7727777300916756, 0.0035893261725153044)
```

3. I ended up using Decision Trees Algorithm. I tried Gaussian NB, Decision Tree Classifier, Support Vector Machine, K Neighbors, and Random Forest. I compared F1 values for all of them and Gaussian NB topped all of them with a F1 score of 0.29856. The second best algorithm was decision tree with a F1 score of 0.23905 and the worst was support vector machine with an F1 score of 0.00100. After the preliminary round, I used the two best algorithms for a final. I used GridSearchCV

and compared the best of both. After tuning the parameters, it turned out that decision tree was the best with an F1 score of 0.39232

4. Tuning the parameters means finding the best parameter values for an algorithm in a specific context. In other words, by tuning the algorithm, I made my algorithm the best it can be for the Enron dataset. If I don't choose the right parameter values, my result will be worse than what could be. When I first ran Decision Tree with default parameters, my F1 score was 0.23905. After tuning my classifier for the optimal number of features, my f1 score increased to 0.39232. I was able to tune my parameters using Grid Search CV. It returned that the best parameters were 'entropy' for criterion which measures the quality of a split, 25 for minimum samples split which is a minimum number of samples required to split an internal node and 12 as the number of features.

5. Validation is a process of estimating an algorithm's performance using an independent data set. This could check whether the algorithm that has been constructed is overfitting the training data or not. A classic mistake in validation would be not using the independent set of data during validation process. In other words, if algorithm is trained on a certain set and then tested on the same set, the algorithm's performance will be inaccurately assessed and overfitting will not be detected. Therefore, dividing up the data set clearly is absolutely necessary. However, then evaluations may depend on the particular way the data is divided up. Cross validation solves this problem by ensuring that every example from the original dataset appears in the testing and training set by equal amount.

For this, I used Stratified Shuffle Split. I implemented SSS while I was running Grid Search CV. Stratified Shuffle Split returns stratified randomized folds but the folds are made by preserving the percentage of samples for each class. In other words, while SSS splits the data randomly multiple times, (in this case 100 times), it assures that each training and testing set contain the same proportion of class distributions. Since the class distribution is highly skewed in our case, where most of the data are non-poi's, using SSS was essential.

6. The most important evaluation metric for this research was f1 metric. It is a weighted average of precision and recall. Thus, it may be more meaningful to talk about Precision and Recall. Mathematically, precision is true positive/(true positive + false positive). Therefore, a high precision means that the among the people the algorithm predicts as a person of interest, a great number of

them are actually persons of interest. Recall is true positive/(true positive + false negative). High recall means that among all the actual persons of interest, the algorithm managed to find many. Because the number of poi's was small, accuracy was not a good measure to gauge the power of the algorithm. In fact, if the algorithm guessed everyone was not a poi, the accuracy would have been as high as 87.67%. SVM did something very similarly. And it achieved a recall of 0.00050. Among 15000 predictions, it only flagged three as poi's. SVM would make a terrible prosecutor. However, on the contrary, K nearest neighbor had the lowest precision (0.06228). This means that K nearest neighbor flags someone as a poi most easily. No lawyer would want KNN as a judge. My final algorithm achieved precision score and recall score of 0.36505 and 0.42400

* Sources : Udacity Discussion Forum, https://discussions.udacity.com/t/selectkbest-and-stratifiedshufflesplit/196653/7