This dataset is emails from the employees of the infamous Enron Corporation. Enron was known for its accounting fraud known as the Enron scandal. The goal of this project was to see if we could build an algorithm to see if we could detect a “poi” or someone involved in the fraud based on emails and financial data. The data can be used by the algorithm to make predictions. There were two outliers in the data, “Total” and “The Travel Agency in the Park”. Obviously these are not employees at Enron. I initially discovered about them by reading forum posts. I had initially overlooked them and was getting 14% accuracy in my Naive Bayes algorithm with errors in the dataset. After removing those two errors the accuracy jumped to 87%.

The features I ended up choosing, were simply just picked by trial and error. I literally added all of the columns to the features list and ran the test classifier. I achieved a precision of around 20% and a recall of around 70%. I first tried to be logical and take out features that might lead to an inaccurate precision, but that didn’t work because feature that I thought would increase precision turned out to actually decrease. This process was sort of a black box, because I didn’t fully understand what was happening or how it worked. I just started randomly picking features to remove to improve the precision because my recall was already above 30%. I thought there would be a more systematic way to do this but after reading the Udacity forum post “confused-about-feature-selection-and-outliers” (resources), I realized this was the way to do it. The feature I came up with was “unusually high bonuses.” Looking at the Enroninsiderpay PDF, I noticed that most of the names highlighted had an unusually high bonus. I did not use an algorithm, but came up with 8,000,000 because that seemed to be the cut off for the names highlighted.

I did not have to tune the parameters for the algorithm for this project. Using the GausianNB algorithm was very straight forward and easy, I just entered the training features and training labels, which were already given in the starter code in the fit function and was able to get around 80% accuracy. However, after going through the Deep Learning Nano Degree I’m very familiar with the importance of tuning hyper parameters. In that Nano Degree I would spend around 20-30% of my time just tuning the hyper parameters, which included Epochs, Keep Probability, Batch Sizes, Convolutional Layers and Pooling Layers. By not tuning hyper parameters the accuracy of the model will be low.

Validation is making sure your model can generalize well to the test data. A classic mistake is to use testing data for training. This is a huge error and will ensure that your model can’t be generalized to other datasets.

For this project I did not have to use feature scaling. Both the Naïve Bayes and Decision Tree classifiers work without feature scaling.

The evaluation metrics used in this project were accuracy, precision and recall. These already came with the starter code in the tester.py file. As already mentioned the accuracy I got when running the GausianNB method was around 80%. After running the test_classifier function on my features list I got a recall of about 30% and a precision of 42%. From what I understand Precision is the proportion of true positives to true positives and false positives. Recall is the proportion of true positives to true positives and false negatives. In other words, precision measured how correctly the algorithm predicted someone was a POI, when they were actually a POI. From our algorithm, this was done only 30% of the

time. Recall measured of the total number of POIs how many did our algorithm mark as POI. This was less than half for both our algorithms.

This algorithm was validated using stratified shuffle split with 1000 folds in the tester.py file. Comparing the two classifiers I found the Naïve Bayes classifier had a significantly larger recall score and only a slightly less precision score. Therefore it would be better to use the Naïve Bayes classifier on this dataset.