1. Today, machine learning is pervasive. It can be applied to different fields to come up with solutions to difficult problems such as in the case of Enron Fraud. The high-profile investigation into the bankruptcy leading to fraud being a can be used as an example to find out the possible trait which are shown by a person of interest. This can allow us to form a classification model that will can aid us in investigating other such organizations in future. As always, raw data has some outliers which cause the predicting model to be swayed by it. One such was the ‘TOTAL’ data point which recoded the total of every other feature. Removing it allowed me to correctly classify the “Person of Interest” (POI) with mush higher accuracy.
2. To understand the effects of all the relevant features on the prediction model, PCA and a feature list was used:

['salary','total\_stock\_value','exercised\_stock\_options','bonus','long\_term\_incentive']

The reason behind used PCA was to find the eigenface values of all the features in the list. This allowed to me to see what effect these features had on the dataset provided. Also, it was observed the features towards the end of the list were far less aiding in predictions. From my above analysis, I inferred that ‘salary’ and ‘total\_stock\_option’ had the largest contribution. Henceforth, they can be used in predicting.

1. Selecting an algorithm is always difficult. We don’t want to an algorithm that is either too bias or has high variance. Therefore, an exhaustive search over algorithms such as Naïve Bayes (GaussianNB), Support Vector Machines, Decision Trees, AdaBoost and Random Forest was performed. To pick one from this list of algorithms, I used two metrics to begin with i.e. Computation time and prediction accuracy. This allowed be to sieve modelling techniques which were complicated and time consuming for the scope of the project. It was observed that SVM and Decision trees were better compared to the rest of the modelling algorithms.

1. There are two ways to go about tuning the algorithm that I selected from my previous analysis. Either I do a manual tuning of the parameter one-by-one or I use a GridSearch over all the parameters that I presume will affect the performance of the classifier. I went with an automatic GridSearchCV on the data. Just to briefly explain what GridSearchCV does, its purpose is to exhaustively perform ‘fit’ and ‘predict’ using certain parameters and then compare it with the other previously calculated. Finally, it will spew out an estimator with best parameters to work with. This function also performs Cross validation which allows the remove any skewed order of the dataset.

Since, my previous analysis led me to 2 classifiers namely, SVM and DT, I did GridSearchCV on both. First, I shall list down the parameters that I tuned for SVM and then for DT.

SVM:

C = [1e3, 5e3, 1e4, 5e4, 1e5], # Decision Surface (DS) is smoother

gamma=[0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1], # Data-points farther from DS to have effect

kernel=['linear','rbf','sigmoid','poly'], # Kernel Trick to classify

class\_weight=['balanced'] # “balanced” mode uses the values of “labels” to automatically adjust weights inversely proportional to class frequencies in the input data

DT:

criterion=['gini','entropy'], # Function to measure the quality of split

min\_samples\_split=[x for x in range(2,16)] # Min. number of sample req. to split node

1. Validation is a technique to check the performance of the used classifier. Accuracy which has been used so far is a misleading metric to evaluate the performance of the model. Instead we need a more robust way to be certain that the model will perform in a certain way no matter the type of data used. If validation is not done properly, we will get a model that works very well with our training and testing data, but will cease to perform when real world data is supplied to it, making it useless. To avoid such a scenario, multiple performance metrics such as “Recall”, “Precision”, “F1-Score”, etc. should be used. To evaluate my model, I used a testing script separate from out modelling. This script performed a “StratifiedKFold” cross validation technique to split the data. Then, I used the split data to make predictions and record the number of “True Positives”,” False Positives”,” False Negatives” and “ True Negatives”. on the data. This allowed me to build my own “Recall”, “Precision” and “F1-score” metrics. Based on the evaluation of these metrics, I made it certain that my classifier will work and is not skewed by the data that is provided
2. The performance of these metrics was averaged over the various folds of the data. Both Recall and Precision were recorded to give a score greater than 0.30 with an average accuracy 81 %.