**Introduction**

In this project, we were given the ENRON Dataset containing the official details of their employees, along with the emails they sent and received and their details. The goal of the project was to classify the Person of Interest (POI) i.e. the person who was a potential criminal.

The dataset contained the details of the users with respect to their finances, loans, emails and whether they were the POI or not. The POI feature which was either true or false indicated if the person was a potential criminal or not. The features related to finances as well as the features of their emails were quite useful to identify the POI.

The Dataset required some cleaning as it contained a lot of NaN values. For this I dropped the columns which contained more than 50% of NaN values. The Dataset also contained some outliers. I detected and removed these outliers using the Z-score method, by computing Z-scores for all the data points and removing the data points which deviated from the mean by more than 3 standard deviations, which were the outliers.

**Feature Selection**

I dropped the columns which contained more than 50% of NaN values and end up using the following features in my project:

1. Salary
2. to\_messages
3. total\_payments
4. bonus
5. total\_stock\_value
6. expenses
7. from\_poi\_to\_this\_person
8. exercised\_stock\_options
9. from\_messages
10. other
11. from\_this\_person\_to\_poi
12. shared\_receipt\_with\_poi
13. restricted\_stock

I did not scale the features as the features mainly contained float values for different financial features such as salary etc and didn’t need scaling. Also, scaling them didn’t improve my classifier’s performance as I selected the AdaBoost Classifier which is based upon the Decision Tree and isn’t affected by scaling. I used the “entropy” as my splitting criteria, and scaling the features doesn’t affect it as well.

**Classifier Selection**

I tried a lot of classifiers including (but not only) the Gaussian Naïve Bayes, Logistic Regression, SVM, Decision Tree, Random Forrest, AdaBoost and Gradient Boosting Classifier. I computed the accuracy, precision, recall and f1-scores for each of the classifiers. I noted that the scores for SVM, Decision Tree, Random Forrest, AdaBoost, Gradient Boosting Classifier were all quite good, so I chose the classifier with the best results of all of these classifiers, which were given by the AdaBoost Classifier. I tried different parameters for different classifiers when I was to choose the classifier and chose the best classifier out of these.

**Parameter Tuning**

Almost every Machine Learning algorithm consists of hyper-parameters, that have to be tuned according to the nature of the problem and the dataset, in order to get optimal results. Therefore, it is an important step and might result in significantly worse results if the parameters are not tuned.

The AdaBoost Classifier consists of the “n\_estimators” and “learning\_rate” hyper-parameters that have to be tuned. I tuned the AdaBoost hyper-parameters using the brute force method. I looped over a set of n\_estimators (1 to 31, with step size of 2) and learning\_rates (0.1 to 2.0 with step size of 0.1), and each time set these different parameters to build my classifier. I computed accuracy, cross-validation score, precision, recall and f1-score for the built classifier, and added all of these to get a score, and stored these scores in an array. At the end of the loop, I found the n\_estimator and the learning\_rate values at which my classifier performed the with the best score, and took these values to train my final classifier.

**Validation Strategy**

In terms of Machine Learning, validation is sort of an evaluation of your trained classifier, that tells about the performance of your classifier i.e. how well it does in predicting for unseen dataset. If you don’t validate your classifier, then it might give unpredictable or unreasonable results while predicting.

I used 5 validation or evaluation metrics in my project namely the accuracy, cross-validation score, precision, recall and f1-score. The accuracy and the cross-validation accuracy can’t give a good idea about the performance of our classifier in this case, as our dataset is very unbalanced. Therefore, I used the precision, recall and f1-score to get a better idea about my classifier’s performance.

**Evaluation**

I am considering cross-validation accuracy and the f1-score for our evaluation. My average cross-validation score is 0.8557 and my average f1-score is 0.8590. Considering the size of the dataset, this is a good evaluation score, keeping in mind that in most of the case 85% is a good score.