1. The purpose of this project is to determine “persons of interest” (POI) in the infamous corporate scandal that led to Enron’s demise in the early 2000’s. I used publically available email metadata and financial data of 145 Enron employees to train a machine learning algorithm to detect POI. Of the 145 employees in the dataset, 18 are listed as POI. There is a glaring outlier in this dataset with the name “TOTAL” that includes the sum of the financial statistics of all 145 employees. Since this observation is not relevant to my analysis, I removed it from the dataset. Machine learning is useful in this application because it learns the patterns in the data and is able to associate them with a label (POI or non-POI). These processes are known as “training” and “classifying”, respectively. Once a classifier is trained, it has the capability to predict labels for which there is no label already assigned.

2. In the first step of the feature selection process, I determined the Pearson correlation coefficient for all features in the dataset, pairwise. For any pairs that had a coefficient value larger than 0.75, I removed one from the pair. This was done to reduce the possibility of over-fitting. Next, I used the “Select K Best” algorithm to reduce features that didn’t contribute to maximizing the f1 score. I did not scale the features because feature distance is not important in the algorithm that I used (AdaBoost with decision tree learning). I engineered and tested three additional features – percentage of incoming emails from POI’s, percentage of outgoing emails to POI’s, and percentage of incoming emails that share receipt with a POI. I created these features because I felt that an email sum was susceptible to bias toward those who generally sent more emails or had a longer tenure with the company. I thought that an employee’s percentage of emails with POI’s, rather than the count, would reveal more about their affiliation with POI’s. In the end, I did not include these additional features because after testing, they did not improve the performance of my classifier. These are the features that I ended up using, along with their scores in the “Select K Best” algorithm:

    - `salary: 18.575703268`
    - `deferral_payments: 0.21705893034`
    - `total_payments: 8.86672153711`
    - `bonus: 21.0600017075`
    - `deferred_income: 11.5955476597`
    - `total_stock_value: 24.4676540475`
    - `expenses: 6.23420114051`
    - `long_term_incentive: 10.0724545294`
    - `director_fees: 2.10765594328`
    - `from_poi_to_this_person: 5.34494152315`
    - `from_this_person_to_poi: 2.42650812724`
    - `shared_receipt_with_poi: 8.74648553213`

3. I used the AdaBoost algorithm with Decision Tree Learning as my weak learner. Using Grid Search CV as a validator, it got a “best score” of 0.346. I also tried the Naïve Bayes

Classifier, which gave me a “best score” of 0.309. Since AdaBoost had a better score, I decided to use it for the final algorithm.

4. Parameter tuning is the process by which parameter inputs to a classifier are altered to improve the performance of an algorithm. If this process isn’t done properly, it could lead to bias, over-fitting, and ultimately a poor classifier. I tuned my parameters using “Grid Search Cross Validation”. I tuned both the “n_estimators” and “learning_rate” parameters of the AdaBoost Classifier, and scored using an f1 score. Grid Search found that the best parameters were `n_estimators = 50` and `learning_rate = 0.9`. I also tuned the “k” value of Select K Best, and found that k = 12 to be best.

5. Validation is the process of measuring the performance of a classifier. It consists of splitting data into training and testing sets, and determining how well a classifier is able to predict the labels of the test set, after being fit on a training set. A mistake in validation occurs when the same data is used to train and test a classifier. This leads to bias when evaluating its performance because the classifier is already really good at classifying the data it was trained on. I evaluated my analysis twice, first using a “Grid Search CV” to decide between two algorithms, and later using a “Stratified Shuffle Split” to measure more specific scores such as precision and recall. This algorithm splits the dataset into several folds, while preserving the percentage of samples in each class. Therefore, each fold will have the same proportion of POI’s to non-POI’s. Note: The “Stratified Shuffle Split” was provided in tester.py.

6. Using “Stratified Shuffle Split”, I got a precision value of 0.44768 and a recall value of 0.32300. Precision is defined as the quotient of correctly identified true values and the total predicted true values. Out of 1500 predictions, the classifier predicted 646 correct true values (true positives). Recall is defined as the quotient of correctly identified true values and the total amount of true values. In this case, on average roughly 8 of 18 POI’s were predicted correctly.