1 **The goal of this project** was to using machine learning in order to develop a classifier for detecting persons of interest ("POI") among Enron employees in the 2001 criminal investigation for fraud. The POI have already been labeled (based on historical knowledge), so the job of the classifier amounts to classification (POI or not-POI) using supervised learning. The data set was not large, containing data for just 146 individuals, and it was imbalanced in the sense that just 18 individuals were labeled POIs, which is less than 15% of the individuals in the data set. Machine learning is useful for finding statistical patterns in classification problems like this, and this is being done in increasingly sophisticated ways, such as classifying an image as a cat in an online video.

The data contained financial information, such as salary or bonuses, and email data, such as the number of emails received from a POI. The email texts were available for processing and text mining into potential usable features, but this was not given in the dataset and the email data consisted solely of metadata based on the email senders and recipients. Looking over the top 1 or 5% of the values for outliers in the financial data, we did see not see any obviously invalid data points (aside from one bogus individual "TOTAL") but we did notice that the top 1 or 5% did have some possibly significant overlap with the POI individuals. Since these data points were not in error in any obvious way and they apparently had positive predictive value, they were retained (*see section on outliers and top 1/5% in the reference document TestRunsTuning.ipynb included in the project submission for more details*). In the processing of the data, we also did some cleaning by turning "NaN" values into 0s, out of a lack of any better alternative.

2 **Feature Selection and Engineering:** I tried a variety of feature selection techniques in the pipeline – Select K best, Principle Component Analysis, along with scaling (min/max) – but none were effective in increasing recall or precision. With some research, I found that "L1 based feature selection" (see section in http://scikit-learn.org/stable/modules/feature_selection.html) could work by using logistic regression with L1 penalty as the model for sklearn's *SelectFromModel* API, and sure enough, this did help boost the precision and recall scores to over 0.30. I did some final checks to verify that using *SelectFromModel* along with the classifier in the pipeline worked better than using the other feature selection techniques, or using it in addition to feature scaling, Select K best features, principal component analysis. It's possible that using logistic regression with "balanced" feature weights worked well with the unbalanced data that we had. *(See the test runs in TestRunsTuning.ipynb included as part of the project submission for further details.)*
We created a feature for having an email address (vs. N/A or NaN) to try to find a benefit from the fact that no one without an email address was a POI. Surprisingly, this did not provide any benefit in the classification tests, as can be seen in the test runs; also, this feature did not make the list of selected features in the recursive feature elimination. Additionally, for further attempts at feature engineering I tried to do some text mining for the code words of some of the secret Enron activities – "fat boy" and "death star" were code words for some fraud schemes. The hope was that I could create a feature for individuals who used these key words, as well as those who emailed these individuals, as an alternative to the feature for having emails to/from POIs. Unfortunately, despite a (surprisingly quick) search through every email in every inbox, there were no mentions of these keywords aside from forwarded news clips on Enron and some mentions using the keywords outside of the Enron context (i.e. Fat Boy Slim). It may be that the bad guys were too smart to leave behind an incriminating email trail like this. *(See the search results in EnronEmailSearches.txt, included in the project submission for further details.)*

3 **The algorithm I chose was AdaBoost**, with Random Forest being a close second. I chose these based on the recall and precision scores, keeping an eye out for the lowest scores. In addition, I tried out support vector machines and naïve bayes, but chose not to use them. During the initial trail runs of algorithms, if I saw precision and recall scores both at or above 0.2, I kept investigating and trying to improve performance; otherwise, I dropped them from consideration. *(See the test runs in TestRunsTuning.ipynb included as part of the project submission for further details.)*

4 **Tuning the parameters of an algorithm** means to make various tweaks to change parameter values (or sometimes code) in order to get improved performance. Without tuning, you will often get poor performance. Improved performance can mean better running time or better output. For machine learning, better output usually means better classification or regression results, which in turn involve better scores for accuracy, precision, and other metrics. For this project, we were concerned with the recall and precision metrics, so I tuned these algorithms for improved precision and recall in the grid search along with the feature selection by logistic regression, as mentioned in question #2. In addition, I did some manual tuning to try to make small improvements on the suggested parameters from the grid search. The grid searches searched for optimal parameter values for number of estimators and learning rate (AdaBoost), as well as maximum decision tree depth (both Random Forest and AdaBoost using decision trees). I also searched for optimal tolerance levels of the feature selection's logistic regression model in the grid search. *(See the test runs in TestRunsTuning.ipynb included as part of the project submission for further details.)*

5 **Validation** is the process of showing that the analysis gives a valid model of the data; in other words, the model matches up with reality or the true state of affairs. A model or statistical analysis that is invalid would be making assumptions that don't correspond to the true state of affairs and so would be giving bad predictions. Since the classification models were constructed by fitting or mapping input features to output categories, we wanted to validate the model by making sure that it works well with *new* input data not part of the training data. Failure to do this kind of validation will often result in overfitting, which happens when the model can predict very well with the input data it has already seen, but perform poorly with new data is given to it. This means the model cannot truly generalize.

Typically, with large enough data sets we can split the data into training data for developing the model and test data that the model hasn't seen yet for validating the model. If the model predicts the new test data well, then that is one classic way of validation. However, in our case we did not have a lot of Enron data (less than 150 individuals that could be a POI) and the data was imbalanced (less than 20% of individuals were POIs), so we used different techniques to do *cross validation*, which performs multiple different test/train splits and averages the performance. This is computationally expensive, but it does not waste any data. The particular method used in the validation was stratified shuffle split, which considers the data imbalances when making the test and training splits. The grid searches also used (stratified shuffling) cross validation to find the optimal parameters.

6 **The two performance metrics of interest in this project were recall and precision**. Precision measures the probability that the individuals identified as targets are actually targets. If the identified individual is not a target, then this is a case of a false positive, and perfect precision means there are no false positives. Recall is the probability of correctly identifying a target among the sample; if an individual is not correctly identified as a target, then this is a case of a false negative, and perfect recall means there are no false negatives.

For the Enron data we worked with, most of the individuals were not POIs, and a “naïve” algorithm might do OK in terms of recall by simply guessing that most individuals are the majority condition, not POIs. Indeed, Naïve Bayes scored 87% recall (perhaps by almost always classifying not-POI) but it had a low precision of just 16%, so it did not do well to identify the actual POIs as POIs (bad false positive rate). This might be good if someone was more interested in not sending a guilty person to jail than in making sure the guilty do go to jail. In my final tuned algorithm, I was getting precision scores averaging in the 40s percent-wise, with the top, final score of .523; this means that I was able to find about half (52.3%) of the POIs that existed in a sample of POIs and non-POIs. I got recall scores usually in the 30s percent rage, with the final, top score of .397. This means that the selected algorithm had at best a 39.7% chance that the individuals identified as POIs were actually POIs.