1. Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those? [relevant rubric items: “data exploration”, “outlier investigation”]

The goal of this project is to, using machine learning techniques, create a classifier to predict whether or not a certain individual is a person of interest in the Enron case. The massive fraud case gave origin to many interesting points of reflection, one of them being the question: is it possible to determine whether or not a member of a group or organization bears responsibility for a reprehensible act traced back to said group or organization? If so, how? Obviously this specific case dates back to the 1990’s, but the question itself is a current one.
A significant amount of data, that normally is internal to a company, became public knowledge as consequence of legal proceedings. In this project the data will be used to train and test a machine learning algorithm to determine whether or not a given individual is a Person of Interest, or POI.

2. What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values. [relevant rubric items: “create new features”, “properly scale features”, “intelligently select feature”]

I wanted to use the financial features alone, to see if we can get answers only from financial information. I figured a lot of the features would be redundant, which is why I engineered a smaller amount of features using PCA. Since the number of principal components cannot be included in the grid for cross validation, different component numbers were tested on the out of the box classifier, and then after tuning it was revisited to ensure the best number of PCs.
Scaling was used for two reasons: to be able to feed the same data into other algorithms that were tried and because I used PCA.
In investigating missing data, outliers and inconsistencies, a few points were removed for having negative values that seemed inconsistent with the type of feature.

3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: “pick an algorithm”]

The algorithm used was Random Forest. I also tried Gaussian and Bernoulli Naive Bayes, as well as C -Support Vector Classification.
The first thing I considered was accuracy, where, in conjunction with dimensionality reduction, Random Forests and Naive Bayes perfomed the best. In preliminary further tuning of each classifier, I was able to get Random Forest to respond better to parameter variations trying to get precision and recall to improve.

Part of this perhaps has to do with the fact that this is a small and well known data set, which makes it easy to intuit which Forest parameters to adjust and how, given that Decision Trees is not a black box method.

4. What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well? How did you tune the parameters of your particular algorithm? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier). [relevant rubric item: "tune the algorithm"]

Tuning the parameters of an algorithm is the process of setting the parameters of the instance of the algorithm in use (the class), to best respond to your data and your needs. In this case, with Random Forests, as would have also been the case with Decision Trees, the biggest risk is to overfit, meaning to create an overcomplicated decision boundary, causing the algorithm to make good prediction on the specific training set used, but fail to do so in other subsets of the data. This is a more pronounced issues with Decision Trees, but not absent in Random Forests. When using Forests or Trees, it is fundamental to cross-validate as a way to avoid overfitting.
I tuned the parameters in conjunction with validation using GridSearchCV.

Because of the long time it takes to run even a simple random forest grid search, it is done in iterations. One parameter and 3 values at a time, GridSearchCV is ran and once the best value for a given parameter is determined, it is extracted via best_estimator and hard-coded into the instance of RandomForestClassifier passed as starter classifier to GridSearchCV.
The params dictionary commented out is the full grid used over the many iterations. An empty dictionary was then passed to params with the sole purpose of being able to display the code with
the validation. The same or similar performance should be observed by designating clf as the same classifier being fed into the grid search. Please see comments in the code for clarification on this, it is easier seen than explained.

5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric item: "validation strategy"]

The validation was done using GridSearchCV, with the cross validation parameter set to StratifyShuffleSplit, due to the small size of the dataset. Some pitfalls of validation are to tune to the wrong metric, for example. When using a tool like GridSearchCV, it is important to ensure the metric being used is the same that you actually want to optimize.
A different, more obvious mistake one can make is to fail to keep training and testing data rigorously separate. Testing on the data used for training will artificially produce good metrics.

6. Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that says something human-understandable about your algorithm's performance. [relevant rubric item: "usage of evaluation metrics"

Recall - average 0.34

Recall measures the rate at which the classifier can make a correct positive prediction, that is to say, the amount of correct positive predictions compared with the amount of actual positive-labeled data points. This was the metric that was the hardest to bring up above 0.3. What this means is that my classifier was having a hard time "seeing" the POIs, and yielding a lot of false negatives.

Accuracy - average 82%

Accuracy the rate at which the algorithm correctly classifies a data point. This is not a great metric on which to rely in most cases, this one included. This is so because of the class distribution imbalance, meaning there are much more non-POIs than POIs, an algorithm that simply classifies all points negatively will yield an impressive looking accuracy.