Please answer each question; your answers should be about 1-2 paragraphs per question. If you find yourself writing much more than that, take a step back and see if you can simplify your response!

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”]

We’d like to use machine learning to be able to create an algorithm capable of figuring out who is a person of interest based on their e-mail activity. We can also supplement this with financial data, such as salaries and bonuses earned by Enron executives (http://news.findlaw.com/hdocs/docs/enron/enron61702insiderpay.pdf). By training an algorithm to use these contextual clues to identify persons of interest, we can uncover more clues in both understanding emails and using machine learning to recognize and identify specific situations.

There are 146 persons in the dataset, 18 of whom are persons of interest. Each person has 22 associated features, although some data points were missing values (e.g. not every person had an associated e-mail address or reported salary). There was one primary set of outliers which was removed from the data: the ‘TOTAL’ data which summed the numerical values (e.g. salary) of each person. These data were reported in the Enron financial data but do not serve a useful purpose here as we are interested in identifying persons of interest from the features of individuals, not the collective.

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 implemented a new feature based on the emails sent by each person in the dataset. I collected all the emails (~5500 emails) and marked each email and marked whether or not each email was sent by a person of interest. I used SnowballStemmer to stem the words, Tfidf to vectorize the data, and a decision tree to train an algorithm to predict whether or not an email was sent by a person of interest based on the words in the email. To make sure that there weren’t signature words that were causing an overfit, I trained on a small dataset (150 emails) to purposely overfit and removed features that reported an importance above 0.2, and then repeated until no features had an importance above 0.2. Then using a much larger training set, I trained the decision tree algorithm which returned a 0.9887 accuracy score. Using this algorithm, I tested the set of emails for each individual person and stored the proportion of emails which were marked as being sent by a person of interest. The create_text_feature.py file can be used to reproduce this.

I experimented with using different sets of features. Using financial data made sense as it provides financial motive for committing the fraud, whereas the email data could show that communicating with persons of interest makes one a person of interest. Using the random forest classifier, with only bonus and salary as features, the recall was 0.143 and the precision was 0.5. Again using the random forest classifier, using just the email features from_this_person_to_poi and from_poi_to_this_person resulted in 0.25 recall and 0.25 precision. Combining these four features with my new feature resulted in different levels of success, but the best result came about from using all five features discussed.

While testing various algorithms, I had to deploy feature scaling when using SVMs to prevent features with large numerical ranges (e.g. salary) from dominating those with smaller ranges (e.g. from_this_person_to_poi). I performed the feature scaling using scikit-learn's MinMaxScaler().

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"]

I used the random forest method. I also tried using naïve Bayes, support vector machines and decision trees. Naïve Bayes had a very low accuracy while the other algorithms were comparable in performance. The decision tree and random forest yielded identical initial results, so I chose to use the random forest algorithm.

| | Random Forest | Decision Tree | SVM | Naïve Bayes |
|---|---|---|---|---|
| Accuracy | 0.94 | 0.94 | 0.85 | 0.86 |
| Recall | 0.6 | 0.6 | 0.55 | 0.2 |
| Precision | 1 | 1 | 1 | 0.5 |

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"]

Once deciding upon using random forests, I used GridSearchCV to tune my algorithm. Parameter tuning is important because many machine learning models are parametrized, and these input parameters can affect the results of the resulting algorithm. If you don't tune these parameters well, your algorithm will be limited in how well it can perform.

The parameters I tuned were the criterion ('gini' and 'entropy') and the number of trees (n_estimators: 10, 100, 250, 500). For these 8 cases, the combination of 'gini' and 10 estimators performed the worst, while the combination of 'entropy' and 10 estimators was the second worst. The other 6 cases performed the best, with the same results. To maximize performance while minimizing time needed, I used criterion = 'entropy' and n_estimators = 100.

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"]

Validation is figuring out if the hypothesized relationships between the features and the labels is an accurate description of the data. A classic mistake with validation is overfitting to the training

data, resulting in a high degree of variance. As a result, the model could be focusing more on the fluctuations in the training data as opposed to capturing the actual relationships between the features and the labels. I validated my analysis by splitting the data into a training set and a testing set. As the names suggest, I trained on the training set and then performed my evaluation on the testing set.

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"]

To evaluate my algorithm, I extracted both the recall score and the precision score. For my random forest algorithm, the recall was 0.75 and the precision was 1. The recall is a proportion of true positives to true positives and false negatives, serving as an indication of how many actual persons of interest could be identified; therefore, my algorithm correctly identified 75% of the persons of interest while resulting in false negatives for the other 25%. The precision is a proportion of true positives to true positive and false positives, indicating how often the algorithm is correct when it identifies somebody as a person of interest. Whenever my algorithm identified a person of interest, it was correct, as a precision of 1 indicates no false positives.