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 the project was to use financial and email data from the Enron data corpus to identify persons of interest. Enron was a global petroleum company whose executives hid losses and exaggerated profits, as well as traded with insider information. This was uncovered in 2001, and a massive criminal investigation went underway. As a result of this, all of Enron's emails and financial data was released publicly. That is this dataset. It contains 20 features and 146 people (18 of which are poi). There are a lot of variables with missing values, mostly in the financial data.
When I first got the dataset, there were a few outliers. Mostly these were executives with huge salaries, bonuses, etc. However, when I selected my features the only feature from the financial datasets that I used was loan_advances. This feature in particular had a lot of NaN values. I assumed, however, that NaN with this feature most likely mean that there were actually no loan advances, and so I left this to zero.
For the other features that I used, there were a lot of outliers with many of their features set to NaN. They showed up as outliers because by default NaN values are set to zero in testing. For these features, it doesn't make sense to set them to zero. For example, it's highly unlikely that an individual is sending and receiving zero emails. I tried to change NaN values from NaN to the average of the respective feature. I did this for two reasons. The first is that I would minimize error best between the actual values and the values in the data if I set the values to the mean. Additionally, this would ensure than when my DecisionTree made splits, the NaN values would end up in the bigger branch rather than arbitrarily being put in the branch with the smaller values.
However, this strategy didn't work, and decreased the recall of my algorithm substantially. So I decided to just drop all of the NaN values. This got my algorithm's performance to what I needed it at.
As discussed in the course, there was also one row of completely bad data: TOTAL. However, because it had NaN values for to_messages and from_messages and so it was removed in the previous step, I did not have to manually remove it.

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 engineered a simple feature which was the function of to_messages/from_messages, if these variables were available. My rationale was that this could serve as a measure of power or

seniority: if a person was receiving a lot more messages than they were sending, maybe they had more seniority. This simple feature turned out to be very effective.
To select features, I used SelectKBest. When I looked at the feature scores, I saw that there was a large dropoff after the sixth feature, which was to_messages. Because the feature that I engineered, in_over_out, had a higher score than and was derived partially from to_messages, I decided not to use to_messages and instead only use in_over_out and the rest of the top five features.
The features that I ended up using (with SelectKBest scores):
shared_receipt_with_poi: 8.9
in_over_out: 5.6
from_poi_to_this_person: 5.4
loan_advances: 2.5
from_this_person_to_poi: 2.5
I used robust_scaler when testing with an SVM, because SVMs require scaled features. However, because I didn't use an SVM for my final algorithm, I removed the scaling step.

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 ended up using an AdaBoostClassifier with a DecisionTreeClassifier as the base classifier.
I also tested other classifiers:

Name	Precision	Recall	F1 Score
GaussianNB	0	0	0
SVM.SVC	0	0	0
Decision Tree	0.316	0.367	0.337
Random Forest	0.296	0.298	0.297
Adaboost (with decision tree)	0.324	0.374	0.345
Gradient Boosting	0.183	0.183	0.183

I found it interesting that the Random Forest and Gradient Boosting algorithms actually decreased the performance of the decision tree that they were based on, and that only Adaboost made any improvements.

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”]

When using an algorithm with parameters, you should tune the parameters to optimize the algorithm for the particular dataset that you are using. This can substantially increase performance, and if not done correctly could make the model much worse. For example, my decision tree was initially performing at around a 0.28 F1 score, but when I changed the

class_weight parameter to “weighted”, which brought the score up to 0.337. To tune my parameters, I used GridSearchCV and tried out a lot of different parameter combinations for each algroithm.

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 the process of evaluating the performance of your algorithm. If you do it wrong you can substantially overestimate your algorithm's performance. For example, if I had used accuracy to measure my performance, the accuracy would have been very high simply because almost all of the people in the dataset were not persons of interest. To validate my analysis, I used the F1 score, which is a combination of precision and recall. To get this F1 score, I used Udacity's provided testing function. This function utilizes StratifiedShuffleSplit. StratifiedShuffleSplit. This function shuffles all of the data and then splits it randomly into training and testing data. It does this many times (1000 by default) and so uses 1000 different combinations of training and testing data so as to get more accurate results. It also keeps the proportion of POIs and non-POIs equal in the training and testing sets as they are in the whole of the data, which ensures that no training data produces a classifier skewed towards one label or another simply because that label has more representation in the training data than it should.
Additionally, to optimize parameters, I used the F1 score in GridSearchCV.

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”]

The precision for my model was 0.32. This means that if the model identifies a person as a person of interest, there is a 32% chance that the person really is a person of interest. The recall of my model was 0.37, which means that if a person is a person of interest, there is a 37% chance that they will be identified by the model.