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?

The goal of this product is to leverage obtained knowledge around machine learning to investigate an existing dataset, the Enron emails, in order to connect the dots from what was learned academically and how to use it in practice. Given the size of data sets such as these it is essential to be able to leverage technologies to analyze the data from trends and then be able to make predictions based on those learnings. There were some outliers in the data. The most glaring was the "TOTAL" key, which was an aggregate of all users data. This was clearly not a person. Jeff Skilling also represented an outlier based on his substantially larger salary as opposed to others.

The data itself contained 21 features, with 121 potential pois provided. Much of this data is not available and unavailable parameters appear to be set as 'NaN'. Stratified shuffle split was leveraged in the testing script in order to ensure any unbalanced data did not skew test and training groups.

Because of the relatively low volume of data, stratified shuffle split is adequate as a means of evaluating data.

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”]
What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?
I used 5 features: 'poi','total_payments', 'from_this_person_to_poi', 'total_stock_value', 'expenses', 'long_term_incentive'

4 of them are pretty straightforward, all about financial incentive ('total_payments', total_stock_value', 'expenses', 'long_term_incentive')
It stands to reason if someone is receiving substantial financial benefit they may well be a POI.

My last one was determined after quite a bit of tweaking. 'from_this_person_to_poi'. I presumed the opposite feature 'from_poi_to_this_person' would yield more favorable results, but this parameter actually drastically reduces performance. It's possile these folks were doing work on behalf of other POIs and thus communicating with them often.

I also created a custom feature that netted all of the aggregate dollar paramters. This feature does not have a huge impact on the data itself but it is similar to the "total_payments" parameter and does have a net positive effect on the f1 and f2 tracking of the data. It stands to reason people that walked away with more pure dollars had a higher likelihood of being POIs

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).

Tuning the parameters of an algorithm essentially means tweaking calculation parameters from their default setting. For a novice it can certainly mean just trying different things to see if it improves your outcomes. In my instance, using decision trees, I altered the max_features parameter to account for all the features in my example (in this case 5)

Here are some of my examples of how I attempted manually tuning the algorithm and the outcome:

```
clf_tree = tree.DecisionTreeClassifier(max_features=5, max_depth=None, splitter="best")
F1 Performance of decision tree:  0.2
```

```
clf_tree = tree.DecisionTreeClassifier(max_features=None, max_depth=3, splitter="best")
F1 Performance of decision tree:  0.285714285714
```

```
clf_tree = tree.DecisionTreeClassifier(max_features=None, max_depth=None,
splitter="random")
F1 Performance of decision tree:  0.285714285714
```

Overall the performance was superior when I used the deafaults, so I stuck with those

```
clf_tree = tree.DecisionTreeClassifier(max_features=None, max_depth=None, splitter="best")
Sample F1 Performance of decision tree:  0.333333333333
```

feature importance weights of my decision tree are as follows: [0.27652874 0.
0.39267945 0.06999533 0.06040718 0.2003893]

What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation is the process of testing your algorithm against actual data. Breaking the data into training and testing groups and then examining performance. One thing to be careful of is not breaking ordered data into uneven test groups. i.e. getting a training test group that is not an accurately representative subset of the actual dataset.

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.

recall - How good is my algorithm at identifying those who are POIs as POIs
precision - How good is my algorithm at making sure I don't flag non-POIs as POIs

```
Accuracy: 0.81892      Precision: 0.40000      Recall: 0.35400          F1: 0.37560     F2: 0.36233
Total predictions: 13000        True positives:  708    False positives: 1062  False negatives: 1292
True negatives: 9938
```

StratifiedShuffleSplit from tester.py was leveraged to validate the results of my test, which works comparitively well in scenarios where there's much fewer targets than there are non targets.