Question 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 take a dataset of Enron data points and determine if someone is a person of interest (P.O.I.). Machine Learning algorithms will be tested and used to determine which algorithms(s) are better at predicting the data points. There are a total of 146 data points (144 after cleaning). The data set contains 21 features with a total of 35 P.O.I.s with 18 of them being Enron employees. The dataset is incomplete containing many features that have a 0 or NaN value. A visual inspection of the dataset shows features like director_fees and loan_advances are missing values for many individuals. I removed three outliers from the dataset: 1) Total, 2) The Travel Agency In The Park, and 3) LOCKHART EUGENE E. The first two points are definitely not individuals and based on feedback the third has no relevant data. The course suggested removing 10% of the top data points as outliers if there are enough data points in the dataset but in my opinion there are not enough data points to do this.

Question 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 doesn't 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.) If you used an algorithm like a decision tree, please also give the feature importances of the features that you use. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

I decided to use only financial features for my testing. I utilized scikitlearn's selectKBest to then narrow that list of 14 features down to a list of top 4 features. I did not need to do any feature scaling as all data points were financial in nature and carried intrinsic values. The new feature I created is named "total_compensation" and is a combination of each person's salary and bonus. I felt this feature would give more insight into how much fraud was taking place as several individuals had a low or average salary (comparatively speaking for the overall company) but had a bonus that was significantly higher. For example Allen Phillip K had a salary of $201,955 but had a bonus of $4,175,000. If we were to only view the salary it would not stand out but adding the two features gives a total compensation of $4,376,955.00 which is exorbitant. The selectKBest scores were:

Feature	**Score**
`exercised_stock_options`	24.815079733218194
`total_stock_value`	24.18289867856688
`bonus`	20.792252047181535

salary	18.289684043404513
deferred_income	11.458476579280369
long_term_incentive	9.922186013189823
restricted_stock	9.2128106219771
total_payments	8.772777730091676
loan_advances	7.184055658288725
expenses	6.094173310638945
other	4.187477506995375
director_fees	2.1263278020077054
deferral_payments	0.2246112747360099
restricted_stock_deferred	0.06549965290994214

I decided early on to set k = 'all' so I could choose the best financial features based on scores. I settled on using the top 4 feature scores as their combined scores were more than the remaining 12 feature scores combined.

Question 3: What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: "pick an algorithm"]

I tested the data using the Naïve Bayes (NB), Support Vector Machines (SVM) and Decision Tree (DT) algorithms. I ended up using the Decision Tree algorithm. The course material had suggested that the Naïve Bayes algorithm was a stronger candidate for text evaluation but in my testing of the data it performed best.

Algorithm	Accuracy	Precision	Recall
Naïve Bayes	0.906976744	0.406511628	0.6
Support Vector Machines	0.837209302	0.203100775	0.4
Decision Tree	0.790697674	0.126356589	0.2

The Decision Tree classifier appeared to be random, giving accuracy scores anywhere from 78% to 90%. The precision and recall scores were just as random.

Question 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 don't 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 if you used, say, a decision tree classifier). [relevant rubric item: "tune the algorithm"]

Tuning an algorithm parameters means that you can change the default parameters of the algorithm to (hopefully) achieve better results. If the parameters are over tuned than a high variance may result with overfitting taking place. If the parameters are under tuned then a high bias might result with new data being ignored. I did performance tuning on both the Support

Vector Machine and the Decision Tree algorithms. No matter what I tried I could not get the tester.py script to come up with acceptable scores for the SVM so decided to focus on the DT algorithm instead. Below are the results from tuning the DT algorithm:

Tuning Parameters criterion = 'gini'	**Accuracy**	**Precision**	**Recall**
min_samples_split = 100	0.83454	0.31358	0.064
min_samples_split = 25	0.81077	0.23502	0.102
min_samples_split = 2	0.78815	0.31875	0.332

Tuning Parameters criterion = 'entropy'	**Accuracy**	**Precision**	**Recall**
min_samples_split = 100	0.82623	0.18025	0.037
min_samples_split = 25	0.79623	0.16373	0.079
min_samples_split = 2	0.79262	0.3178	0.304

Question 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 would be the process where the algorithm is evaluated with split data. This process uses training data to tune and then testing data, all from the same dataset, to determine results. A classic mistake would be to use the same data for both training and testing. This would cause overfitting and result in inflated results. In this project the dataset was split between a training subset and a testing subset of the dataset.

Question 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"]

I used several evaluation metrics in this project. Accuracy, Precision and Recall with Precision and Recall being the two most heavily weighted. Using the tester.py script I was able to use a 3rd metric with was the stratified shuffle split and it also showed the F1 and F2 scores. The results are below:

Naïve Bayes: Accuracy (0.907), Precision (.41), Recall (0.6)

SVM: Accuracy (0.837), Precision (0.203), Recall (0.4)

DT: Accuracy (0.814), Precision (0.116), Recall (0.0)

DT Tester.py Results: Accuracy (0.788), Precision (0.319), Recall (0.332), F1 (0.332), F2 (0.336)

After all is said and done I still feel that the Naïve Bayes was the better algorithm but I was unable to find much information on tuning it. In the results from the tester.py for the DT algorithm the recall score was higher than the precision score which means more individuals may be added to the P.O.I. list including some that may be innocent.