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?

The data that was used for this project was pre-processed email summary and financial data from the Enron Emails dataset and Persons of Interest (POI) in this dataset are individuals that were involved in the Enron scandal. The goal of this project was to produce an algorithm that could determine, based on a dataset with a list of input features and labels (either POI (1) or non-POI (0)), whether a given individual with input features belongs to either the POI or non-POI group and to produce this conclusion with metric values of 0.3 for both recall and precision. This project used machine learning as a means to train the algorithm on a portion of the data so that it could produce predictions that would be compared against the correct labels of the other portion of the dataset.

In the dataset, there were 145 total points of which 18 are POI's. There were a considerable amount of outliers (greater than 1.5 times the interquartile range from the 3^{rd} or 1^{st} quartiles) for any given feature. As removing all of these outliers would reduce the small dataset significantly, outliers were only examined for values that were not within the 90^{th} percentile for any feature that did not belong to POI labels. 57 of the 145 datapoints fit into this criteria.This was performed algorithmically using the 'rescale_remove_outliers' function. As the data was preprocessed previously, there are no missing (NaN) values present in the dataset.After examining the data further, I opted to only remove the 35 outliers that had unrealistic values such as fractions greater than 1, a value of 0 for to or from messages,$0 salaries, and the one line item that was a Total. Other values that were present in the dataset appeared to be legitimate even though they had outlier values and so were manually kept in the dataset. The total list of outliers can be seen in the attached 'outliers.html'

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.

In the POI identifier, I used the features in Table 1 using the SelectKBest function to determine the 13 features with the best ANOVA f scores. The ANOVA f score was used to determine the best values since it is the most appropriate test to use for classifiers with non-boolean or frequency-based data. 13 was used as the number of features to retain since it resulted in the best recall and precision scores for the algorithm. The features were scaled to values between 0 and 1 so that algorithms that are sensitive to scaling (i.e SVM's), could be tested if desired.

Two features were created for this project: frac_from_poi and frac_to_poi representing the fraction of total emails received from POI's and fraction of total emails sent to POI's respectively. The rationale behind using these values was that the fraction received from or sent to POI's might be more useful than the total number of messages sent or received (which ended up being confirmed by the ANOVA f score for the frac_from_poi, but not the frac_to_poi value). Without these features, the precision and recall drop to approximately 0.318 and 0.302 respectively, demonstrating the usefulness of these features in achieving the final metric values. Principal Component Analysis was also used to transform and reduce the number of features, but no gain was realized for the additional time to run the algorithm and so this was not used in the final analysis.

Feature	ANOVA F Score	Include / Exclude
salary	0.0242342406	Exclude
deferral_payments	0.25269391	Include
total_payments	0.1556272693	Include
loan_advances	1.4515647538	Include
bonus	0.0164937532	Exclude
restricted_stock_deferred	0.3940907249	Include
deferred_income	0.0495366889	Exclude
total_stock_value	0.0449240533	Exclude
expenses	0.0578289283	Exclude
exercised_stock_options	0.0675558363	Include
other	0.0086863558	Exclude
long_term_incentive	0.0005933473	Exclude
restricted_stock	0.0021339823	Exclude
director_fees	0.3106827117	Include
to_messages	1.9980861192	Include
from_poi_to_this_person	10.2489652995	Include
from_messages	0.2188574574	Include
from_this_person_to_poi	2.3780681459	Include
shared_receipt_with_poi	12.3721915427	Include
frac_from_poi	2.9066173468	Include
frac_to_poi	3.1471432272	Include

Table 1: Features and ANOVA F Scores

3.What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

The algorithm that was used for the final project was a VotingClassifier estimator that used an optimized AdaBoosted DecisionTreeClassifier and an optimized GaussianNB classifier as input. Other algorithms that were used were an optimized DecisionTreeClassifier, optimized AdaBoosted DecisionTreeClassifier and GaussianNB classifiers, but all were unable to attain the required recall and precision values of 0.3. For the Adaboosted DecisionTreeClassifier, it was able to achieve a value of precision greater than 0.3, but resulted in poor recall values (~0.23). The optimized DecisionTreeClassifier received similar results, but with worse performance in each metric. For the optimized GaussianNB classifier, it was able to achieve very large recall values (~0.8), but resulted in very poor precision (~0.1). Because of the individual flaws in both of the preceding algorithms, they were combined using a VotingClassifier with equal soft weights to achieve recall and precision values of 0.383 and 0.386 respectively.

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).

Tuning the parameters of an algorithm means that you are optimizing the input parameters of the algorithm to produce the best possible classifier. If this is not done correctly, the algorithm can overfit or over-generalize to your data and may not result in the best possible predictions. For my algorithm, the performance tuning was done using GridSearchCV to optimize the 'criterion','min_samples_split',and 'max_features' values of the DecisionTreeClassifier. GridSearchCV was also used along with the StratifiedShuffleSplit function to perform 1000 folds of the labels for testing.

5.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 when the data is split into both training and testing datasets so that you can fit the algorithm to a training dataset and validate the algorithm's fit using the testing dataset. Classic mistakes that can occur if done wrong are using too small a training dataset (resulting in a poor fit), and having the data split in ways that can incorrectly bias the fit (such as all 0 labels in the test split, all 1 labels in the train split). In order to validate my analysis, I used the StratifiedShuffleSplit function with 1000 folds and default splitting fraction so that a plethora of train and test split configurations would be tested to yield the best results for the individual algorithms feeding into the VotingClassifier.

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.

The two primary metrics that were used to evaluate the performance were the precision and recall values of the algorithm. These two metrics are especially important as they are good measures of evaluation when there is significantly less of one label than the other (as is the case for POI's and non-POI's). In two test runs, the algorithm achieved an average precision value of 0.383 and an average recall value of 0.386. This precision value of the metric can be interpreted as the algorithm correctly identifying a fraction of 0.383 of correct POI's divided by the total number of POI predictions that were returned. A better precision value indicates that the algorithm tends to be reluctant to guess a value unless it is certain that the value is correct. The recall score can be interpreted as the algorithm correctly identifying a fraction of 0.386 of the actual POI's. A larger recall value indicates that the algorithm is likely to determine that an individual is a POI if they are actually a POI.