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 for the project is to identify the Enron employees that committed fraud using the dataset provided. Some elements of the dataset involve the number of emails employees have sent, their salaries and bonuses, total payments, etc. With any *outlier investigation*, you want to ensure that the outliers are insignificant, so that we can remove them. In this dataset, for NaN, I designated those values to 0 to help fit with the format of the dataset.

Specifically, “total” was removed, as it reflected total values and would not have contributed to solving the issue of finding any POI’s.

There was a total of 146 data points that consist of 18 POI and 128 non-POI. I used a total of 3 features: decision tree, cross validation, and KFold.

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”, “intelligently select features”, “properly scale features”]

I ended up using the following features, based on my thoughts of key features that can help identify POIs: poi, salary, bonus, expenses, shared_receipt_with_poi, from_messages, and to_messages. I felt that, if I was to locate specific POIs, these features would help locate any outliers, discrepancies, and inconsistencies. For example, a POI can potentially have a higher salary, an abnormal bonus, or different expenses.

Feature scaling was not applied in my coding.

I created two new features: fraction_to_poi_emai and fraction_from_poi_email.The fraction_to_poi_email feature indicates the fraction of all emails send to a poi to all emails between a person and a poi. The fraction_from_poi indicates the fraction of all emails received from a poi to all emails between a person and a poi. I created these two features to explore how often a POI contacted someone or was contacted by the same person. I found that about 27-28 emails were sent to POIs while only 5 emails were received from POIs. I found that there was a minimal impact that the new features would have in my algorithms. In fact, it had a negative impact with some of the algorithms, so I elected NOT to use the new features.

To select the features, I used the attribute feature_importances_ to determine which can be selected. I decided to use the same list as all_features to test which features would be most important.

ExtraTreesClassifier was used to help sort the features and rank by importance. The features that are the most important, in order, are: exercised_stock_options, total_stock_value, long_term_incentive, from_messages, deferral_payments, poi, total_payments, director_fees, and salary. I did this by replacing the features_list parameter with all_features on line 142 of the poi_id.py file.

For my POI identifier, I used cross validation, KFold, and decision trees to help predict values from the data features. The decision tree features are important as decision trees utilize very little memory without any change as the features get scaled.

KFold was used to split the data into 3 consecutive folds, or equal subsets. We can train the cross validation model we used earlier as a training set, and get the accuracy compared to each fold. Then, get the average of the accuracies to better understand how your model works. I then used decision trees to compare testing and training features to get the accuracy before and after tuning. I used 3 folds as it was the default: 3 folds are used as validation, while the k-1 method (so, 2 folds) form the training set.

I compared the accuracy before and after tuning. Before, the accuracy was 0.229. After tuning, however, the accuracy was 1!

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 following algorithms: decision trees, naive bayes, and support vector machines.

I picked DecisionTreeClassifier to help visualize the final data while using very little memory and data preparation. From what I understand, the Decision Tree does not change with feature scaling, but it is still recommended that the data gets standardized.

I also tried GaussianNB, and noticed that the accuracy was actually 0.3125 before tuning, but was 1 after tuning! GaussianNB in this instance is difficult to get too excited, as I know naive bayes assumes that the drawn vectors are the complete representation of the population.

Support vector machines provided an interesting take. As they are effective in high dimensional spaces, they reported an accuracy of 0.3125 before tuning. However, it took longer to run the algorithm when compared to decision trees.

The information from the table below was found my modifying the test_classifier function within the tester.py file. I changed the clf to claf and clasf, based on my code in the poi_id.py file. I also commented out the new features and re-ran the function. Here are the results.

	W/ New Features			W/O New Features		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
DecisionTree	0.79940	0.23093	0.21650	0.79940	0.23036	0.21550
GaussianNB	0.79893	0.22834	0.21350	0.79687	0.22635	0.21650
SVM	0.79607	0.22350	0.21400	0.79867	0.22959	0.21650

As you can see from the table above, the new features provide increases in all aspects of the DecisionTree model, but does reduce some factors in the other algorithms.

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? What parameters did you tune? (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 items: "discuss parameter tuning", "tune the algorithm"]

I tuned my parameters and fit each algorithm with training data to use against the testing data. I then compared the results of the score between the testing and training data, along with the time it took the run the algorithm. I did this for each of my classifiers: decision tree, GaussianNB, and SVM. Tuning the parameters of an algorithm can help frame the data and information a lot more effectively. If there are any visuals involved, it can help tell the story in a more clear and concise manner. It can also optimize said parameters before using any models.

Underfitting, which results in a very simple model, makes the algorithm very rigid and difficult to learn from the dataset. On the other hand, overfitting will create too many results, and it will make it difficult to generalize from any learnings from the dataset.

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 items: "discuss validation", "validation strategy"]

Validation, in the world of machine learning, is where a trained model is evaluated with a testing data set, with the main purpose of using the testing data set to check the ability of a model. For example, we can use prior data to predict that would've happened yesterday, and actually compare that with the events from yesterday. This is why we split between training and testing subsets. It can help with issues such as overfitting!

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"]

	W/ New Features			W/O New Features		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
DecisionTree	0.79940	0.23093	0.21650	0.79940	0.23036	0.21550
GaussianNB	0.79893	0.22834	0.21350	0.79687	0.22635	0.21650
SVM	0.79607	0.22350	0.21400	0.79867	0.22959	0.21650

As precision refers to the false positive rate, and recall refers to the false negative rate, the chart above shows that the new features do not help improve the precision and recall, when compared to the original dataset.

When applied to this project, the precision rate helps identify how well we can determine the POI we were wrong about (but originally believed to be POI), while the recall rate shows how well we were wrong about a certain employee NOT being a POI.