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 investigate the data provided to determine persons of interest (POIs) from the Enron Data Set. Machine learning is useful for accomplishing this task because it can be trained to correctly label a person as either a POI or a Non-POI. Our dataset has 146 total people in it, 18 being known POIs gathered from legal documentation, and the other 128 are Non-POIs. The goal is to train an algorithm to determine if a person would classify as a POI or Non-POI. Each person in the dataset has 21 features that can help train the algorithm. I chose to not use the feature 'email_address' because it is a unique identifier for a person, and not relevant for analysis. It is also string data where the rest is numeric, which can be scaled/normalized to be equal in weight.

There were a few outliers I came across in my investigation, the users 'TOTAL', 'LOCKHART EUGENE E', and 'THE TRAVEL AGENCY IN THE PARK' are all outliers. The user 'TOTAL' is just an accumulation of all values for each feature and can be excluded from the dataset. The user 'LOCKHART EUGENE E' has nothing but null values for every possible feature (except 'poi') and therefore will be excluded. The user 'THE TRAVEL AGENCY IN THE PARK' also appears to be an outlier in that all but two categories are null values. The 'total_payments' and 'other' features for this user have the same numeric value and it really is not clear if this is a person or a business. Due to this uncertainty and lack of information, I removed this person from the dataset.

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 importance's 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"]

My initial investigation, using out-of-the-box learning algorithms, involved using a min/max scaler to normalize all the numeric data to carry the same weight. I then used these normalized features in the SelectKBest algorithm and manually adjusted my values of k to see how each algorithm performed when the only modifications on the dataset where normalizing and adjusting how many input features were used. The initial analysis showed the GaussianNB algorithm performed the best through 100 simulations with differing training/test sets and

using 6 features. I then used a sklearn Pipeline to streamline the process. I again scaled the data, found its most important features using SelectKBest, then ran each learning algorithm. I integrated the GridSearchCV algorithm to test a variety of parameters for each learning algorithm, and a range of values for k (number of features) that were used with SelectKBest.

I created two additional features that I thought could be useful in my analysis. I created a 'to_poi_ratio' and 'from_poi_ratio' feature that calculates how many emails a person sent/received from persons of interest. I expected a high ratio of communication with a POI could be a good indicator of another POI. Ultimately SelectKBest did not find a strong enough correlation to consider using either of these features. The best resulting algorithm using a Pipeline with GridSearchCV still resulted in the GaussianNB algorithm scoring the best, this time utilizing the best 4 features. It makes sense that these 4 features would be selected since the top 4 all have a score above 18, while the 5th highest score drops to a score of 11, which is significantly lower. These features and their scores are:
'exercised_stock_options': 24.815079733218194,
'total_stock_value': 24.182898678566872,
'bonus': 20.792252047181538,
'salary': 18.289684043404513

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"

My final algorithm ended up being a GaussianNB learning algorithm. It scored the best when I manually modified parameters, and when I streamlined the process using GridSearchCV and a variety of k values for SelectKBest.

I also attempted an SVM algorithm, Decision Tree algorithm, and the Adaboost algorithm. They varied in how well the performed, some even scoring higher that the Naïve Bayes algorithm for accuracy. The problem was precision, recall, and f1 score for these other algorithms were poor and could not predict the same level of success that the Naïve Bayes classifier did.

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"]

Model tuning is the process of adjusting the hyperparameters of a learning algorithm to have better efficiency, faster convergence, and improve its performance. I did both manual and

automatic adjustment of parameters when trying to find the best performing algorithm. When doing my initial investigation of learning algorithms, I modified the number of features that SelectKBest was utilizing. Later I applied GridSearchCV to search a grid of several different parameters to find the one that was the best fit. Failure to properly tune our model would mean our performance would suffer and our model would not fit the data as well as it could. The model would not be trained to make accurate predictions when testing with new data and this makes parameter tuning an essential part of building a well-fitting model.

I did not have any parameters to fine tune in my final learning algorithm (Naïve Bayes), just strictly what features were used from SelectKBest (which I did tune the k value to test different feature numbers). One algorithm I did try fine tuning, was scalar vector machines (SVM). I tried different processing kernels and different values of both C and gamma. These parameters control how accurate the algorithm is and how influential a single training point can be. I used GridSearchCV to try a range of options for each parameter and found the best fitting set of parameters.

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"]

Model validation is the process of evaluating how a model performs and reacts to new data. A common mistake is to use the same data to train and test an algorithm. Without a separation between testing and training sets, it is hard to know how the algorithm would perform on new data. The algorithm would do well for data it was trained on, but then perform differently (most likely poorly) on new, unseen data; the algorithm would overfit the data. Properly splitting the data into training and testing sets is crucial in validating the results and being able to accurate predict/model new data.

Cross validation techniques exist that will divide the data into specified testing and training sets which help prevent overfitting (example: train_test_split). I used train_test_split to specify my test set will be 30% of the total dataset. I also used GridSearchCV which has its own cross validation parameter. I specified this parameter to utilize a StratifiedShuffleSplit so that each training/testing set would have an appropriate ratio of each class since we are working with unequal class distributions.

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 the evaluation metrics precision, recall, and f1 score. My Naïve Bayes classifier had and average precision score of .333, recall score of .4, and f1 score of .363. These metrics are better suited than accuracy since we are dealing with highly skewed classes, with more Non-POIs than POIs.

- Precision is the ratio of how many true positives are classified as positive out of all the labels classified as positive.
    - In terms of this investigation, a higher score would mean we are correctly labelling a POI as a POI and would not have many Non-POI incorrectly classified as a POI. This means our number of false positives, people who are a Non-POI but got classified as a POI, is low - we have a low number of false positives.
- Recall is the ratio of how many true positives we identified out of the total number of positives we should have in the dataset.
    - In terms of this investigation, it is how many true POIs were identified from the total number of POI. To have a high recall score we would want to identify a high percent of the total POIs as actual POIs. This would mean the number of POI classified as a Non-POI would be low - a low number of False Negatives.
- F1 score is the balancing point between precision and recall.

With this dataset, I think recall would be of higher importance. We would want to limit the number of false negatives that we have; we would want a POI to be classified as a POI and criminally investigated, not to be incorrectly classified as a Non-POI. A higher recall score would mean there are less false negatives and we have less POIs being classified as a Non-POI. We want to investigate any POI and incorrectly classifying a true POI is what we want to minimize. My Naïve Bayes classifier did this the best out of the ones I tested with a recall score of 0.4, and this was the final model used in my investigation.