1. The purpose of this project is to predict whether an individual is a person of interest. The dataset includes 18 POI and 128 non-POI, for a total number of 146 data points. For each individual, the dataset provides these 21 features:
    a. financial features: ['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees'] (all units are in US dollars)
    b. email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (units are generally number of emails messages; notable exception is 'email_address', which is a text string)
    c. POI label: ['poi'] (boolean, represented as integer)

    POI is actually the target variable, so there are technically only 20 features. I will train a model using a subset of these features to identify whether an individual is a person of interest. From the lessons, we saw that TOTAL was an outlier, and this point was the summation of the points in the dataset, so it is removed. There is also a data point called LOCKHART EUGENE E, which only has NAN values, so it is removed. Finally, the data point THE TRAVEL AGENCY IN THE PARK, is additionally removed since it doesn't represent a person.

2. To select the features for the POI identifier, I used VarianceThreshold and SelectKBest. With VarianceThreshold, I select the features whose variance is greater than 80%. Then with SelectKBest, I keep the six features with the best scores. The features that I keep are: exercised_stock_options (score=24.8), total_stock_value (score=24.18), bonus (20.8), salary (18.2), deferred_income (11.5), and long_term_incentive (9.9). The new features that I created were fraction_from_poi and fraction_to_poi, which are the fraction of received emails that are from a POI, and the fraction of sent emails that are to a POI. The intuition behind these features is that the a POI communicates more with other POIs. However, as shown above, these features do not get included in the kept features. With the six top scoring features, I use MinMaxScalar to normalize them, as the data has a wide range of values, and algorithms like K-Means and SVM don't work properly without normalization.

3. I used GridSearchCV to select the best parameters for Gaussian Naïve Bayes, SVM, and K-Means. Gaussian Naïve Bayes performs the best in terms of accuracy (0.93), precision (0.5), and recall (0.67). KMeans performs very poorly with an accuracy of 0.07. SVM has an accuracy of 0.93, but a precision and recall of 0.

4. When you tune the parameters of an algorithm, you optimize the algorithm's parameters to achieve the best performance of the algorithm. If the tuning of the algorithm isn't done well, the algorithm will perform poorly, such as low accuracy and low precision. With GridSearch, I found that the best performing algorithm is Gaussian Naïve Bayes, which doesn't have any parameters.

However, for the other algorithms that I tried, GridSearchCV trains each algorithm for each combination of parameters in the grid, and then reports which choice of parameters resulted in the best performance.

5. Validation is when you check how your trained model generalizes on an independent data set. If done wrong, a classic mistake you can make is to overfit the model. This is when the model performs very well on the training data set, but does not perform well on unseen data. I validated my analysis by using train_test_split to split the provided data set into a training set and a testing set. The models are trained on the train data, and then the accuracy, precision, and recall are reported for the models evaluated on the testing set.

6. The evaluation metrics that I used are accuracy, precision, and recall. I found that the best performing algorithm was Gaussian Naïve Bayes, with an accuracy of 0.93, precision of 0.5, and recall of 0.67. Accuracy is the fraction of the time that a value is the true value, so my model labels correctly 93% of the POI and non-POI. Precision measures the fraction of the time that an algorithm identifies true positives from all data points that are classified as positives. In this case, the precision of 0.5 means that half of the people who were classified as POIs were actually POIs. Recall measures the fraction of the time that an algorithm identifies true positives out of all data points that are actually positives. In this case, the recall of 0.67 means that 67% of the true POIs in the dataset are correctly identified as POIs.

References:

N/A

I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.