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. 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 a POI communicates more with other POIs. 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. To select the features for the POI identifier I used SelectKBest with a Pipeline that also uses GridSearchCV over the classifiers GaussianNB, SVC, and LogisticRegression. The Pipeline along with GridSearchCV tries each classifier for all combinations of parameters in the grid and features.

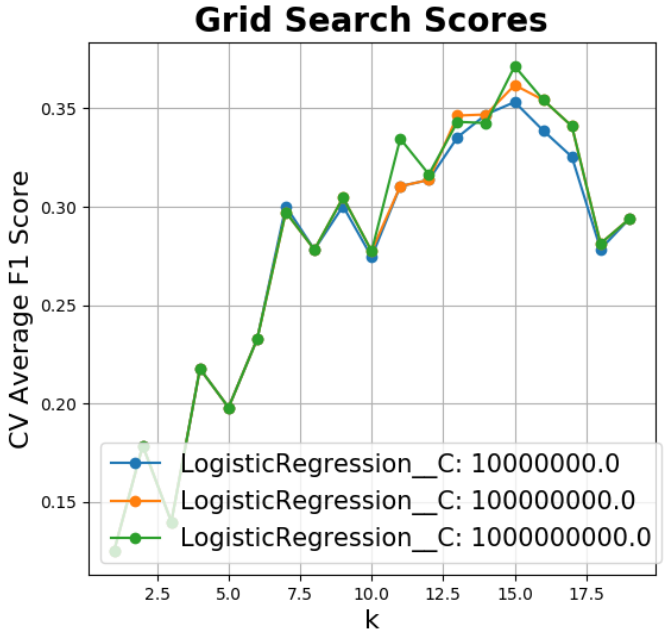
    The best performing logistic regression is found to have 15 features, which are (score, feature):
    (24.815079733218194, 'exercised_stock_options')
    (24.182898678566872, 'total_stock_value')
    (20.792252047181538, 'bonus')
    (18.289684043404513, 'salary')
    (11.458476579280697, 'deferred_income')
    (9.9221860131898385, 'long_term_incentive')
    (9.212810621977086, 'restricted_stock')
    (8.7727777300916809, 'total_payments')
    (8.5894207316823774, 'shared_receipt_with_poi')
    (7.1840556582887247, 'loan_advances')
    (6.0941733106389666, 'expenses')
    (5.2434497133749574, 'from_poi_to_this_person')
    (4.1874775069953785, 'other')

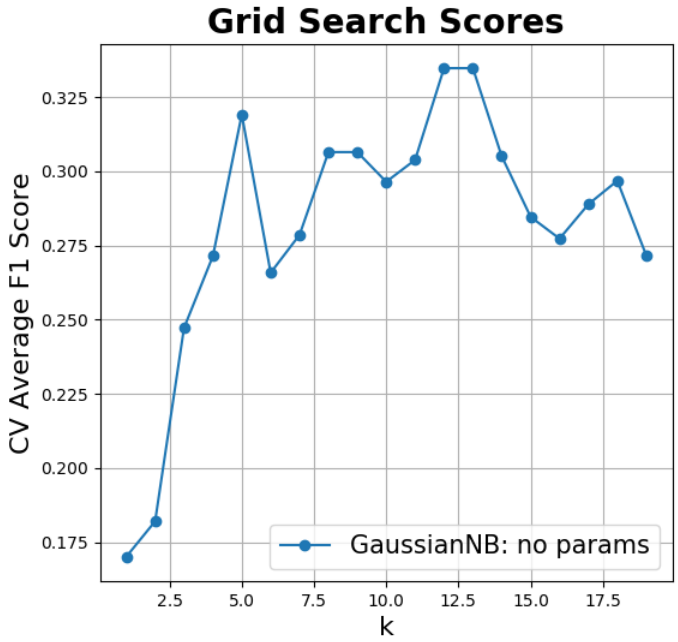
(2.3826121082276743, 'from_this_person_to_poi')
(2.126327802007705, 'director_fees')

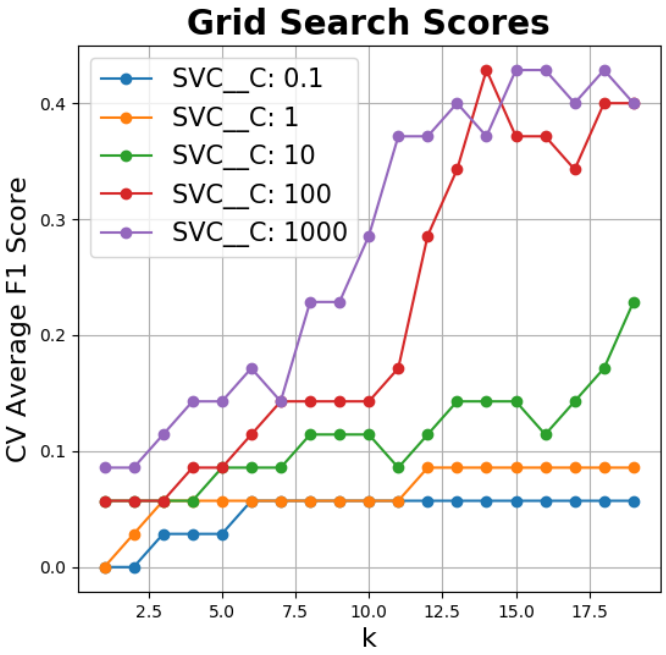
The best performing SVC is found to have 15 features, which are (score, feature):
(24.815079733218194, 'exercised_stock_options')
(24.182898678566872, 'total_stock_value')
(20.792252047181538, 'bonus')
(18.289684043404513, 'salary')
(11.458476579280697, 'deferred_income')
(9.9221860131898385, 'long_term_incentive')
(9.212810621977086, 'restricted_stock')
(8.7727777300916809, 'total_payments')
(8.5894207316823774, 'shared_receipt_with_poi')
(7.1840556582887247, 'loan_advances')
(6.0941733106389666, 'expenses')
(5.2434497133749574, 'from_poi_to_this_person')
(4.1874775069953785, 'other')
(2.3826121082276743, 'from_this_person_to_poi')
(2.126327802007705, 'director_fees')

The best performing GaussianNB is found to have 12 features, which are (score, feature):
(24.815079733218194, 'exercised_stock_options')
(24.182898678566872, 'total_stock_value')
(20.792252047181538, 'bonus')
(18.289684043404513, 'salary')
(11.458476579280697, 'deferred_income')
(9.9221860131898385, 'long_term_incentive')
(9.212810621977086, 'restricted_stock')
(8.7727777300916809, 'total_payments')
(8.5894207316823774, 'shared_receipt_with_poi')
(7.1840556582887247, 'loan_advances')
(6.0941733106389666, 'expenses')
(5.2434497133749574, 'from_poi_to_this_person')

3. I used GridSearchCV to select the best parameters for Gaussian Naïve Bayes, SVM, and LogisticRegression. The following plots show how the cross validated average F1 score changes for different parameters for each of the models, when StratifiedShuffleSplit with five splits is used as cross validation.


Grid Search Scores
CV Average F1 Score
k
LogisticRegression__C: 10000000.0
LogisticRegression__C: 100000000.0
LogisticRegression__C: 1000000000.0


Grid Search Scores
CV Average F1 Score
k
GaussianNB: no params



For each classifier, I set its parameters to those that result in the best F1 score, and then perform a StratifiedShuffleSplit cross validation with three splits to get following reported average accuracy, precision, and recall values.
Gaussian Naïve Bayes: accuracy (0.75), precision (0.38), and recall (0.75).
LogisticRegression: accuracy (0.82), precision (0.51), and recall (0.42)
SVM: accuracy (0.79), precision (0.33), recall (0.08).
For question 6, I chose the Gaussian Naïve Bayes as the final model as it results in the largest recall score as we have unbalanced classes. The Gaussian Naïve Bayes model is fit using the 12 features in question 2.

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. Additionally, parameter tuning can help to prevent overfitting. With GridSearchCV and SelectKBest, 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. I tuned the parameter 'C' for SVC and logistic regression. For example, the above plots show that using k=15 features, the cross-validated F1 score using SVC is best when C=1000, and decreases as C decreases.

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 the cross validation method of StratifiedShuffleSplit.

6. The evaluation metrics that I used are accuracy, precision, and recall. Since there are unbalanced classes, there are more non-POI than POI, I choose the classifier that results in the best average cross-validated recall score. I found that the best performing algorithm was Gaussian Naïve Bayes, with an accuracy of 0.75, precision of 0.38, and recall of 0.75. Accuracy is the fraction of the time that a value is the true value, so my model labels correctly 75% 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 0.38 means that 38% 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 0.75 means that 75% of the true POIs in the dataset are correctly identified as POIs.

References:

Code for making plots: https://stackoverflow.com/questions/37161563/how-to-graph-grid-scores-from-gridsearchcv

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.