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”]

A. The goal of this project is to build a person of interest identifier based on financial and email data made public as result of the Enron scandal. Since a hand-written list of persons of interest is available, we can use them to build a model which can act as guide to predict further persons of interest. The dataset has 146 records. Of them, 18 are POI’s and 128 are non-POI’s.
There were a few outliers in the dataset. The first outlier was a spreadsheet aggregated value in the dataset – a dictionary key value called 'TOTAL'. It had to be removed from the dataset.
Further analysis of the histograms of variables like ‘SALARY’ and ‘DEFERRAL PAYMENTS’ reveal that there are outliers whose salary is above 1 million and deferral payments are above 6 million. These are kept in the dataset as they might be potential POI’s.
There are a lot of NaN values in the dataset which can be viewed by running my code. They are not removed though. They are converted to 0’s instead. The reason being that they are indicators of whether someone is a ‘poi’ or not. For eg., if the bonus value is NaN, it’s possible that the person is not a ‘poi’. It’s possible that it’s missing data but it’s unlikely as they are financial or message data in most cases whose data capturing is not be prone to human error.

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

A. I noticed that ‘total_stock_value’ was the sum of ‘exercised_stock_options’, ‘restricted_stock_deferred' and 'restricted_stock'. Hence I eliminated the other three.
I engineered two features. The first one is monetary_incentive which is created as the sum of salary, bonus and deferral_payments. It gives an idea of the monetary incentives which a person gets and hence the higher chances of him to be a POI. The second one is poi_contact which is the sum of shared_receipt_with_poi, from_this_person_to_poi and from_poi_to_this_person. This gives an indication of the communication between a person of interest and some other person. The more this number, the likelier the person is to be a person of interest. The effect of the features has been tested. A PCA with Naïve Bayes model was built with and without the engineered features. The model performance with the engineered features was better than without it. It is documented in the table in the next section.
Scaling of features is needed as the range of values is so different for different features that it might cause issues in the model. The performance
K-best feature selection is used to select the 7 best features. The following 7 features are selected:

'salary', 'total_payments','bonus', 'total_stock_value', 'total_payments', 'monetary_incentive', 'poi_contact'. Their corresponding scores are: 1.58060901e+01, 8.96271550e+00, 3.06522823e+01, 1.08146349e+01, 8.49349703e+00, 1.58060901e+01 and 1.06697374e+01. I could have picked lower number of features like removing salary and total_payments. But for the purpose of experimentation, I wanted to apply PCA further and remove such dependencies.

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"]

A. A comparison of model performance is given below:

Model	Accuracy	Precision	Recall	F1 score
PCA with Naïve Bayes (no engineered features)	0.82293	0.34557	0.36700	0.35
PCA with Naïve Bayes (with engineered features)	0.83687	0.37563	0.37750	0.35554
Decision Tree Classifier	0.82233	0.18303	0.09600	0.12
Naive Bayes	0.81920	0.29517	0.25650	0.27448
SelectKBest (K=5) with Naïve Bayes	0.83873	0.36423	0.28100	0.31725
SelectKBest (K=3) with Naïve Bayes	0.83873	0.36423	0.28100	0.31725
SelectKBest (K=3) with Decision Tree Classifier	0.81047	0.28440	0.27800	0.28116
SelectKBest (K=7) with Decision Tree Classifier	0.82173	0.31110	0.27750	0.29334
SelectKBest (K=6) with PCA and Naive Bayes	0.84860	0.41803	0.34550	0.37832

I ended up using the PCA with Naïve Bayes algorithm (with the engineered features) as it gave the required precision and recall scores along with good accuracy.

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). [relevant rubric item: "tune the algorithm"]

A. Tuning the parameters of the algorithm mean trying out different values of the parameters to cater to varying requirements like speed, accuracy, complexity, etc. If not done well, then the algorithm could be slow or inaccurate. Since Naïve Bayes can't be used, I was not able to tune it. Had I used decision tree, I would have optimized the algorithm parameters such as max_depth and max_features by defining a list of values for it and passing it to GridSearchCV parameter param_grid. The best_params_ attribute of the output will give me the list of the best combination of parameters.

Instead of tuning the parameters, I used GridSearchCV to tune the number of components of PCA. The number of components tried out were 5, 6 and 7. It turns out that a 5 component PCA works best in terms of accuracy after applying it to the reduced set of 7 features picked after feature reduction.

The model performance for different component values is documented below:

Model	Accuracy	Precision	Recall	F1 score
5 component 7 Best Tuned PCA with Naïve Bayes	0.85307	0.42735	0.30000	0.31901
6 component 7 Best Tuned PCA with Naïve Bayes	0.84860	0.41803	0.34550	0.37832
7 component 7 Best Tuned PCA with Naïve Bayes	0.84487	0.40274	0.33850	0.36783

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 item: "validation strategy"]

A. Validation refers to testing our algorithm to a test data set and check its performance. Overfitting is a classic mistake that occurs if validation is done incorrectly. I split the dataset into a 70-30 training-test sets split. The model was fit on the training data set and the performance was checked by predicting values using the test data set and comparing it with the actual test set labels.

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"]

A. The evaluation metrics used for the model are shown below:

Model	Accuracy	Precision	Recall
5 component 7 Best Tuned PCA with Naïve Bayes	0.85307	0.42735	0.30000

Accuracy refers to the ratio of correct predictions to the total number of observations. In this case, it means the proportion of the testing set labels correctly predicted. Precision reflects the proportion of POI's correctly predicted from the collection of POI's predicted. Precision gives an indication of whether the algorithm does not overcommit in labelling someone as a POI. Recall refers to the proportion of POI's correctly identified with the algorithm. Recall tells us if the algorithm has the ability to correctly identify a POI.