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 of the project is to use the Enron dataset and machine learning technique to correctly identify the person of interest (POI). I will use features in the dataset to build models that has the highest accuracy in predicting if a person is POI or not.

   The dataset has 146 entries, out of which, 18 of them are POIs. The data also has 20 features (excluding POI). They are:

   - salary
   - to_messages
   - deferral_payments
   - total_payments
   - exercised_stock_options
   - bonus
   - restricted_stock
   - shared_receipt_with_poi
   - restricted_stock_deferred
   - total_stock_value
   - expenses
   - loan_advances
   - from_messages
   - other
   - from_this_person_to_poi
   - director_fees
   - deferred_income
   - long_term_incentive
   - email_address
   - from_poi_to_this_person

   Upon reading into the dataset, I decided to explore the data by looking at "salary" and "bonus" features, because intuitively the Enron scandal may be related to the salary and/or bonus received by the POI.

   Plotting out the salary and bonus data, there is an outlier that separate out from all other entries, and this data point is not even a POI. I then examine the data by looking at sorted salary, and I found out the outlier is the "total" entry, which should not be included in the data for us to use. I then removed it.

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.

I used R to conduct feature analysis.

When exploring the feature, I found some features have many missing values ("NaN"), and may not be a good identifier to build the classification model. The features with more than 50% of data are NaNs are:

- deferral_payments
- restricted_stock_deferred
- loan_advances
- director_fees
- deferred_income
- long_term_incentive

I decided to only include features that has less than 50% of missing value, and this helped me to reduce the features from 20 down to 14 (salary, to_messages, total_payments, exercised_stock_options, bonus, restricted_stock, shared_receipt_with_poi, total_stock_value, expenses, from_messages, other, from_this_person_to_poi, email_address, from_poi_to_this_person)

Since "email_address" and "other" features do not give us additional useful information, I also decided to exclude them, which brings down the features to 12.

Also when looking at the email features with GGAlly, I found out that among POIs their to_message has 0.97 correlation with shared_receipt_with_poi, and from_message also has 0.99 correlation with from_this_person_to_poi. This may be due to that shared_receipt_with_poi and from_this_person_to_poi will increment for POIs since they are POIs themselves. A better way to include these data is to look at the fraction of total from and from message that are shared with or sent to POIs. I created "fraction_from_poi" and "fraction_to_poi" which looks at what percentage of their sent are sent to POIs and what percentages are received from POIs.

Finally, we shrink down our features to 9 to use in our data modeling:

- salary
- total_payments
- exercised_stock_options
- bonus
- restricted_stock
- total_stock_value
- expenses
- fraction_from_poi
- fraction_to_poi

Before continuing, I noticed the financial data (salary, payments, bonus, etc) all have wide range, especially comparing to the email fraction data. So in order not to let finance data to skew the feature importance, I decide to scale all features to “standardize” them.

While we still have 9 features, it is still relatively high dimensional, so I also decide to perform PCA to reduce dimensionality and leverage SelectKBest to automatically select the best number of features.

I did these two steps in parallel with featureUnion function in Sklearn, and feed into a pipline with an Random Forest Classifier algorithm, and it turns out the combination of features and PCAs are 1 PCAs with 3 features: 'salary', 'bonus', and 'total_stock_value', as they have the highest scores with SelectKBest algorithm (salary: 16.85, bonus: 34.13, and total_stock_value: 11.84). The combination of 1 PCA with these 3 features yielded total score of 0.85.

Note that the two features we created (‘fraction_to_poi’ and ‘fraction_from_poi’) have low sc ores: fraction scored 0.12, and ‘fraction_from_poi scored 1.99.

3. **What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?**

Initially I chose Gaussian Naïve Bayes(GNB) as provided in the code. I also tried out Random Forest(RF) as well as SVM. After running all three models, I found Gaussian Naïve Bayes model not only had the fastest training speed, and the accuracy is also much higher than Random Forest and SVM: GNB finished training in less than 1 second, while SVM lapsed for 14 seconds and Random Forest took up to 6 minutes!

This may be due to the parameter fine tunes that affects the number of fits happening for each model: 72 fits for GNB, 960 fits for SVM and 864 fits for RF.

GNB not only require less time to train, but the accuracy score is also higher (as elaborated in the later section), I end up using GNB.

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).**

   Tuning the parameter is like an art – to find the best configuration that lets your algorithm find the best classification for the dataset. I used SVM and tuned the “kernel”(linear and rbf) and “C” (1, 10, 100, 1000, 10000) parameters. The kernel parameter determines which “kernel trick” the model will be using – linear kernel will be able to compute a linear boundary and rbf can compute a non-linear boundary; the C parameter is the tradeoff between having a smooth boundary (may not have the best classification precision) or a more “ragged” boundary (may have higher precision in classification, but prone to over fitting).

I used a pipeline to feed the parameters to the model and find the best combination, and it turns out the best parameter combination for SVM are linear kernel with C value 10. However, the model estimator score is lower than GNB so I ended up using GNB for my final algorithm.

Because GNB does not take any parameters, I didn't fine tune any parameter for the GNB model. But, I also used a pipeline and feed feature selection and PCA to GNB model like I did for SVM, and find the combination with the best score for the training data. And it turns out the model has the highest score with 1 PCA from 3 selected features (salary, bonus and total_stock_value)

Having the best model score does not necessarily mean it's the best model, as over-fitting could be an issue, therefore I ran cross validation to validate if the model has a good accuracy score.

5. **What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?**

Cross Validation is to partition datasets into training and testing datasets to train model on training data and validate accuracy on test data. It gives estimate of performance on independent dataset (training vs testing), and because the data set are split to two independent sets, it can also help to determine if the model is over fitting on training set, and poor on making classification (or prediction on testing set).

The K-fold validation splits data in to K number of folds, with comparable amount of data in each fold, and one of the fold of data will be the training set. We then run the machine learning algorithm K times on each of the fold as training data, and average the test results. This helps to increase accuracy as this method essentially uses all the data to train the model.

I use this with Sklearn's cross validation test and training set split feature, and the GridSearchCV, running the model training with 3-fold cross validation with each combination of PCA and SelectKBest and with 72 fits.

I then validate the analysis by looking at the best estimator score as well as the accuracy of the model on testing dataset.

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.**

I looked at the precision and recall score. The average performance for precision and recall are both 0.86.

Looking into the details, non-POIs have great precision and recall scores, both at 0.92, which means when the model is great at identify non-POIs: If we are given a non-POI, we have 92% chance to correctly identify him/her as non-POI; similarly, if the model classifies someone as non-POI, the accuracy is 92%.

For POIs, both prediction and recall rate is at 0.5, which is essentially similar to coin toss: when given a POI, we have 50/50 chance to identify him/her correctly; and if the model predicts someone as POI, the accuracy is also 50/50. Therefore, the model does not have very good accuracy in identifying a POI. However, I also compare the same metrics to SVM and RF, POIs precision and recall rate are much lower (some are even 0), hence I decided to use GNB instead as it has a better balance of precision and recall score for both non-POIs and POIs.

I also used the accuracy score from Sklearn to get the accuracy score with labels predicted on test set vs the real test set label. The accuracy score (which we correctly identified the persons as POIs/non-POIs) is at 0.86, which means the model we correctly identified almost 86% of the testing data set as POIs or non-POIs.