Project Overview:

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives.

Questions:

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 use email and financial data from Enron to build a model that can identify whether a person is a "Person of Interest (POI)" or not. The dataset has 146 data points and 21 features that includes both financial as well as email related fields. "POI" is a labeled feature. Dataset contains 18 records that are labeled as "POI".

Each field has some missing values. Below is the count of missing values.

salary: 51
to_messages: 60
deferral_payments: 107
total_payments: 21
loan_advances: 142
bonus': 64
email_address: 35
restricted_stock_deferred: 128
total_stock_value: 20
shared_receipt_with_poi: 60
long_term_incentive: 80
exercised_stock_options: 44
from_messages: 60
other: 53
from_poi_to_this_person: 60
from_this_person_to_poi: 60
poi: 0
deferred_income: 97
expenses: 51
restricted_stock: 36
director_fees: 129

While visualizing salary and bonus using scatter plot I was able to identify an extreme outlier. On investigating further, I noticed that the data point belonged to a name field 'TOTAL'.
'TOTAL' represents nothing but the sum of all data points and thus was removed from the dataset.

Another outlier was 'THE TRAVEL AGENCY IN THE PARK'. This was not any person's name and had missing value in almost all features.

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", "intelligently select features", "properly scale features"]

First I tested my final classifier with all features and it gave me precision score of only 0.18 which was very less. Next I tried with best 10 features(didn't include new features) and I got relatively much better scores i.e greater than 0.3 for both recall and precision. I further reduced my features to top 5 and got precision score of 0.46 and recall of 0.35. To select best features I used scikit-learn's SelectKBest.

I also created two new features namely 'fraction_from_poi' which is fraction of number of messages that the person received from POI and 'fraction_to_poi' which is the fraction of messages that the person sent to POI. Before adding these features to my feature list, I first removed other email related features except 'shared_receipt_with_poi'. I did this because new features were created using 'to_messages', 'from_messages' , 'from_poi_to_this_person', 'from_this_person_to_poi' features and considering them again would be a repetition.

I tested my classifier again with new features in the list. During my analysis I didn't found a very major difference between the scores that I got when I didn't include them. For example when I selected best 5 features I got precision score of 0.45 and recall of 0.32. At last I ended up using 6 because final scores that I got were comparatively better than with 5 with precision score of 0.47 and recall of 0.35.

New features that I created didn't improve my model much but I found the new fields to be more relevant with respect to POI and hence included them in my final feature list. These new fields showed the ratio of messages that were exchanged between POI and a person.

Below are top 6 features that were used and their scores.

exercised_stock_options:	25.1
total_stock_value:	24.47
bonus:	21.06
salary:	18.58
fraction_to_poi:	16.64
deferred_income:	11.56

Lastly, I scaled all the selected features using MinMaxScaler so that all features are measured on same units.

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”]

I tested four differnet algorithms namely GaussianNB,DecisionTreeClassifier,SVM,KMeans. And I found GaussianNB to be the best performer.

On testing GaussianNB using tester.py I got Precision score: 0.51572 and Recall score: 0.38550.

Precision and recall score for DecisionTreeClassifier was also greater than 0.3 but it wasn't better than GaussianNB. I also found SVC and KMeans to be slower than the other two algorithms I tested.

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”]

Tuning an algorithm plays an important role on its performance. Setting right value of parameters can give unexpected results. If set not properly they might overfit the model which reduces the model's accuracy.

For this project final algorithm I selected is GaussianNB and I didn't had to tune this algorithm as it gave me best results as compared to other algorithms even after tuning them.

In Decision tree classifier, I changed splitter parameter and set it to "random" as it gave better results than with its default parameter. I tried to add other parameters but I didn't see much improvement so I removed them.

Similarly for K means I changed tol and n_clusters and for SVC I changed C and gamma parameter.

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”]

Validation is helpful in assessing how well our algorithm performs besides the data used to train it. That is how well it performs on testing data as compared to training data. The mistake that we can make is by overfitting the model which performs well on training data but its performance on testing data is not that good. So validation serves as check on overfitting.

For my project I saved 30% data for my testing and rest for training purpose. I used cross-validation for splitting the data. I also ran 1000 trials and took average of precision and recall to calculate scores.

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"]

The two evaluation metrics that I have used are precision and recall. Precision tells what proportion of people labeled as POI are actually POIs. And Recall tells proportion of people labeled as correct POI to people flagged as POI.
The best scores were given by GaussianNB which helped me in selecting this algorithm as final one.