Enron Submission Free-Response Questions

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”]

In this project, I’m working on Enron Dataset. 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. There is a list of “person of interest” (POI), which means individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity. My task in the project is to build up a machine learning algorithm to identify whether a given person is in the list of “person of interest” based on the email and financial data.

Before I built the algorithm, I explored the dataset a little bit. The following is a summary of some basic facts for the dataset.

Number of data points: 146
Number of POI: 18
Number of Non-POI, 127
Percentage POI: 0.124137931034
Percentage non-POI: 0.87586206896
Total features: 20 (14 financial features, 6 email features)

The followings are the table with features and number of missing.

The features that are missing most data are: loan_advances, director_fees, restricted_stock_value

Features	Missing Num
salary	50
deferral_payments	106
total_payments	20
loan_advances	141
bonus	63
restricted_stock_deferred	127
deferred_income	96
total_stock_value	19
expenses	50
exercised_stock_options	43
other	52
long_term_incentive	79
restricted_stock	35
director_fees	128

During the exploration process. There are some outliers in the dataset I removed:

"Total": It's not the data for any specific person
"THE TRAVEL AGENCY IN THE PARK": this has nothing to do with POI
"LOCKHART EUGENE E": no data for this person

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. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

   New features: I created some "ratio" features for both financial and email features, including: The ratio between total emails received and the email received from POI(from_poi_rate); the ratio between total emails sent and the email sent to POI(to_poi_rate); the ratio between bonus to salary(bonus_to_salary) and the ratio between stock to salary(stock_to_salary). I believe those ratios will be a better indicator compared with absolute values, since it's comparable between different people.

   Feature scaling: Before I selected the features, I first used MinMaxScaler for all the features. Because the range of different feature varies a lot, e.g. total_stock_value can be huge but the ratios I created are always between 0-1.

   Feature selection: I used SelectKBest for feature selection process. I tuned the K value from 1-23 (total number of features) and compared the accuracy for the algorithm I selected (I used GaussianNB). K = 5 gave the best results. So I selected 5 features in the final model. They are 'bonus', 'exercised_stock_options', 'salary', 'total_stock_value', 'eferred_income'. The followings are the scores for these 5 features.

   K value is tuned manually and the following table is a summary. The results may vary with the output from tester.py, since they are using train_test_split(features, labels, test_size=0.3, random_state=42)

Value of k	Recall Score	Precision Score
1	0.333333	0.5
2	0	0
3	0.142857	0.5
4	0.125	0.333333
5	0.333333	0.4
6	0.333333	0.4
7	0.333333	0.4
8	0.333333	0.4
9	0.142857	0.333333
10	0.142857	0.333333
11	0.714286	0.121951
12	0.714286	0.121951
13	0.714286	0.121951
14	0.714286	0.121951
15	0.571429	0.1
16	0.571429	0.102564
17	0.571429	0.105263
18	0.571429	0.117647
19	0.571429	0.121212
20	0.571429	0.121212
21	0.571429	0.121212
22	0.571429	0.125
23	0.571429	0.125

The following is a summary

('exercised_stock_options', 24.815079733218212)

('total_stock_value', 24.182898678566886),

('bonus', 20.792252047181531)

('salary', 18.289684043404527)

('deferred_income', 11.458476579279765)

Features Score: The following list the feature scores for all the features, including newly added features (from high to low). We can say the “bonus_to_salary” feature has a very high score among all the other features, ranking the 6th.

('exercised_stock_options', 24.815079733218212),

('total_stock_value', 24.182898678566886),

('bonus', 20.792252047181531),

('salary', 18.289684043404527),

('deferred_income', 11.458476579279765),

('bonus_to_salary', 10.359195709258653),

('long_term_incentive', 9.9221860131898243),

('restricted_stock', 9.2128106219770771),

('total_payments', 8.7727777300916738),

('shared_receipt_with_poi', 8.5894207316823774),

('loan_advances', 7.1840556582887247),

('expenses', 6.0941733106389337),

('from_poi_to_this_person', 5.2434497133749636),

('other', 4.1874775069953794),

('from_poi_rate', 3.1280917481567356),

('from_this_person_to_poi', 2.3826121082276743),

('director_fees', 2.126327802007705),

('to_messages', 1.6463411294419978),

('deferral_payments', 0.22461127473600523),

('from_messages', 0.16970094762175433),

('to_poi_rate', 0.14313106248590121),

('stock_to_salary', 0.11773867028602983),

('restricted_stock_deferred', 0.06549965290989139)

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”]

   I used GaussianNB for the final project. I tried DecisionTreeClissifier as well.

   Here are the accuracy for these two algorithms:

   GaussianNB()

   Accuracy: 0.85464 Precision: 0.48876 Recall: 0.38050 F1: 0.42789 F2: 0.39814

   DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,

   max_features=None, max_leaf_nodes=None, min_samples_leaf=1,

   min_samples_split=2, min_weight_fraction_leaf=0.0,

   random_state=None, splitter='best')

   Accuracy: 0.79000 Precision: 0.32299 Recall: 0.33300 F1: 0.32792 F2: 0.33095

   GaussianNB has higher precision and recall scores.

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? 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 item: “tune the algorithm”]

   Tuning the parameters for an algorithm would improve the precision and recall scores for the prediction. I tuned the parameters for DecisionTreeClassifier. I used GridSearchCV to tune “feature_selection_k”, “criterion”, “max_depth”, “min_sample_split” and “class_weight”.

   The following is the results after tuning

   DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
   max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
   min_samples_split=2, min_weight_fraction_leaf=0.0,
   random_state=None, splitter='best')
   Accuracy: 0.80660 Precision: 0.28120 Recall: 0.28950 F1: 0.28529 F2: 0.28780

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"]

   Validation is the process where a trained model is evaluated with a testing data set. The purpose of validation is to test the generalization ability of the trained model. If we don't do the validation correctly, then we are likely to create over-fitting model. I used "train_test_split" model for validation, 70% percent of data for training and 30% for testing

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"]

   I used precision score and recall score as evaluation metrics.

   Precision score: The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. In this project, for those people that my algorithm marks as a POI, 48.9% of them are real POI, 51.1% are actually non-POI.

   Recall score: The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. In this project, my algorithm recognize 38% of the real POI.