Enron Submission Free-Response Questions

Enron was one of the largest companies within the country in 2000. In 2002, Enron suffered a bankruptcy due to corporate fraud that was found throughout the company. After extensive federal investigation, it was found that there was a significant amount of confidential information that was entered into public record, including tens of thousands of emails and detailed financial data for Top Executives.

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 objective of this project is to develop a Machine Learning algorithm using sklearn (part of Python) to investigate the Enron corpus, publicly made files, that comprise email and financial data of 146 people. The purpose of the algorithm is to try and determine if a person is a "Person of Interest" (POI). The dataset contains 146 records with the following features: 1 labeled feature (POI), 6 E-Mail features, and 14 Financial features. Within the corpus, there are 18 people who are labeled as "Person of Interest". Although the Enron corpus already contains some people labeled as "POI", the purpose of this project is to build an algorithm that utilizes features that are provided within the corpus to build a Person of Interest Identifier. I found within the dataset, there are several NaN values within the 21 features:

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

The features with several NaN values most likely won't be included later on in the analysis for the features that should be considered for the algorithm as those features have several values missing.

My initial review of the people included within the dataset pointed out the outlier of "Total". "Total" appears to be an outlier as it is not the name of a person. I also found an outlier titled "The Travel Agency In the Park" also does not appear to be an actual person. Lastly, I found that there wasn't any information contained in "Lockhart Eugene E", therefore all three of those data points were removed from the dataset. Once those three data points were removed from the dataset, the total data points decreased to 143.

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"]**

I did create three new features for the dataset. I created the three new features to see if there would be an improvement to determine if there was an increase in the precision and recall scores of the algorithm.

Bonus-Salary Ratio for Employees: I created this feature to determine if there was an imbalance for employees with a lower salary, however they have a much higher bonus amount, or the opposite of employees who had very large salaries, however low bonuses.

To POI Message Ratio: I created this feature to identify if there was a higher ratio for emails that were sent to a POI from a POI versus from a non-POI.

From POI Message Ratio: I created this feature to identify if there was a higher ratio of emails that were sent by a POI to another POI versus to a non-POI.

Once I created my three new features and added them into the features_list, I then utilized SelectKBest and also used the min-max scaler on all of the features. It is important to scale the features to ensure that all of the features would be considered evenly for the algorithm. I first chose for the algorithm to select the top 10 features of the dataset so that I could see what features had the most influence on the dataset. I also felt that 10 was an appropriate number of features to select because there is a total of 21 features, therefore selecting the top 10 features would allow for me to understand if there were more financial features or E-Mail features. The Precision and Recall Score that was produced with 10 features is shown below:

```
Accuracy: 0.86500       Precision: 0.49015      Recall: 0.31100 F1: 0.38054     F2: 0.33553
Total predictions: 15000        True positives:  622    False positives:  647   False negatives: 1378   True negatives: 12353
```

Although the 10 features satisfied the requirement of both scores above 0.3, I wanted to see if reducing the number of features to 8, if there would be an improvement. I changed the number of features to 8 and found that the precision and recall score improved:

```
Accuracy: 0.87913       Precision: 0.58095      Recall: 0.33550 F1: 0.42536     F2: 0.36647
Total predictions: 15000        True positives:  671    False positives:  484   False negatives: 1329   True negatives: 12516
```

I then tried 7 features, and found that the precision and recall score did not score as high as the 8 features selection did, and with 9 features selected, I found that the Precision and Recall score was just barley under the scores for 8 features:

```
Accuracy: 0.87913       Precision: 0.58081      Recall: 0.33600 F1: 0.42572     F2: 0.36693
Total predictions: 15000        True positives:  672    False positives:  485   False negatives: 1328   True negatives: 12515
```

When I tried 11 features, the precision and recall score suffered:

```
Accuracy: 0.85367       Precision: 0.41500      Recall: 0.23800 F1: 0.30251     F2: 0.26019
Total predictions: 15000        True positives:  476    False positives:  671   False negatives: 1524   True negatives: 12329
```

I decided to leave the amount of features at 8 as it produced the highest precision and recall scores.

I then added the three features that I created into the features_list and the SelectKBest was then produced, those three new features did not pick up in the top 10 list, therefore they don't have enough of an improvement. I also ran the entire algorithm and the tester.py code, and the recall score dropped to 0.287, therefore the three features I created has an effect on the final scores:

```
Accuracy: 0.85733       Precision: 0.44565      Recall: 0.28700 F1: 0.34915     F2: 0.30900
Total predictions: 15000        True positives:  574    False positives:  714   False negatives: 1426   True negatives: 12286
```

Once I commented out the lines to add the three new features, the precision and recall scores increased:

```
Accuracy: 0.86500       Precision: 0.49015      Recall: 0.31100 F1: 0.38054     F2: 0.33553
Total predictions: 15000        True positives:  622    False positives:  647   False negatives: 1378   True negatives: 12353
```

Below are the 8 features that the SelectKBest produced as well as the scores:

```
8 best features:['salary', 'to_messages', 'total_stock_value', 'expenses', 'exercised_stock_options', 'from_messages', 'from_this_perso
n_to_poi', 'deferred_income']
[ 17.71787358  11.18458025  23.61374045   5.815328    24.25047235
   1.5425809    0.18121501   2.2951832 ]
```

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"]

The algorithm that I selected was the Logistic Regression algorithm. I ultimately selected the Logistic Regression algorithm because we are trying to determine the difference between two classifications, either POI or non-POI. For further comparison to other algorithms, I also utilized KMeans, Random Forest, and Adaboost algorithm to determine which algorithm would produce the highest precision and recall score. After I tuned the algorithms, below are the precision and recall scores that I received:

Classifier	Precision Score	Recall Score
Logistic Regression	0.392	0.252
KMeans	0.482	0.315
Random Forest	0.266	0.435
Adaboost	0.330	0.132

Both the Logistic Regression and KMeans performed well and gave me the closest scores to 0.3 for both precision and recall, whereas the Random Forest and Adaboost were more spread out.

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"]**

   The tuning of parameters means that adjustments are made to the algorithm to improve the fit of the algorithm on the dataset. The parameters can have a direct effect on the precision and recall scores that are processed, meaning the algorithm is trying to consider features that can hinder the performance of the algorithm. The algorithm will try to consider too many features or too much data and won't be able to then make accurate predictions on the dataset.

   On my algorithm for Logistic Regression, I manually tuned the tol (tolerance for stopping criteria) and C (inverse of regularization strength) parameters to various amounts to see what would produce a high enough precision and recall score. I found that increasing C (10**18) and tol (10**-21 values, I was able to achieve the higher precision and recall scores. This was a trial and error process on my side that was very manual, but it offered me insight into the tuning of the parameters how that affected the scores.

   In regards to the KMeans algorithm, I modified the tol(tolerance for stopping criteria) parameter. I found that tuning the parameters specifically on the Logistic Regression and KMeans algorithms, I saw an improvement on both the precision and recall 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? [relevant rubric item: "validation strategy"]**

   Validation is the training and testing of data and provides an estimate of performance on an independent dataset. The validation serves as a check for overfitting of the data by the algorithm. A classic mistake is over-fitting the training data set that performs well, however the test data performance is much lower. I validated my analysis by taking the average precision and recall scores for 1,500 random trials with a ratio of 3:1 training to test data. The tester.py script includes the StratifiedShuffleSplit because the dataset that we are utilizing contains a very small amount of POIs compared to the dataset as a whole so the data needs to be random when it is split so that the POIs don't get grouped up into one of the trials.

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"]**

   2 evaluation metrics that I utilized were the precision and recall scores. The precision score is the ratio of true positives that are actually POIs divided by the sum of true positives plus the false negatives, cases where it was predicted to be positive but are actually incorrect. The recall score is the ratio of true positives divided by the sum of true positives plus false negatives, cases where it was predicted to be negative and was truly positive. In regards to my algorithm, the logistic regression produced a precision score of 0.392 and a recall score of 0.252 before the tester.py was ran. For the kmeans algorithm, the precision score was 0.482 and recall score of 0.315, which is closer to the goal of 0.3 for both scores. To put these scores into context, the precision score is stating that for 100 POIs, both Logistic Regression and KMeans would predict

that 36 – 39 people are truly POIs, and the remaining are non POIs. In regards to the recall score, out of 0.252 and 0.315, the model would find around 25% of true POIs in its prediction before the tester.py script runs. I chose Logistic Regression as my final algorithm to use through the tester.py script and found that the precision score raised to 0.581 and a recall score of 0.336 so there was an increase in both scores.