Enron Submission Free-Response Questions
Cynthia O'Donnell

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 purpose of this project was to identify which of Enron's employees were likely to have committed fraud and, therefore, warrant further investigation. There are 146 observations (i.e., people) in the dataset and 18 of the people were labeled as a 'poi' or person of interest ("POI"). There were 21 features in the dataset, 15 financial features and 6 email features. Many of the features had a lot of `NaN` values. Several of the features were more than 70% `NaN`. I did not use those features. I also did not use features if all of the POI's had `NaN` values.

There were three outliers that I removed. Two were not people (`TOTAL` and `THE TRAVEL AGENCY IN THE PARK`) and one was a person who had `NaN` values for all features. The dataset is relatively small and the number of POIs in the dataset is also small. I did not remove any of the extreme value outliers because I felt it was likely the the extreme values could be useful in machine learning. This belief is partially based on prior knowledge of the Enron scandal, namely, that the people with the highest salary and bonuses (among other relevant features) were engaged in fraud.

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 doesn't 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.) If you used an algorithm like a decision tree, please also give the feature importances of the features that you use. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]*

Initially, I manually reviewed the data to determine which features might be most useful. I determined that `from_poi_to_this_person` and `from_this_person_to_poi` might not be particularly useful as counts so I created `to_poi_ratio` and `from_poi_ratio` to represent the proportion of emails to to or from a poi. The `from_poi_ratio` was ranked as the 10th best feature by the SelectKBest algorithm, but the other proportion was not selected. I then decided to normalize the ratios by subtracting the mean ratio from these new features, creating `to_poi_ratio_normalized` and `from_poi_ratio_normalized`. Neither of these features were chosen by the SelectKBest algorithm.

After I created the new features, I determined that several features had too few observations to be reliable. For example, `loan_advances` had only 3 observations, one to a POI. I felt that there would be no possible benefit to keeping this feature in the POI Identifier and its inclusion would only obfuscate other more reliable patterns. I removed `director_fees`, `restricted_stock_deferred`, and `deferral_payments` for similar reasons.

After this preprocessing, I selected the following features using SelectKBest, listed with their scores:

Rank	Feature	Score
1	'exercised_stock_options'	24.815079733218194
2	'total_stock_value'	24.182898678566879
3	'bonus'	20.792252047181535
4	'salary'	18.289684043404513
5	'deferred_income',	11.458476579280369
6	'long_term_incentive'	9.9221860131898225
7	'restricted_stock'	9.2128106219771002
8	'total_payments'	8.7727777300916756
9	'expenses'	6.0941733106389453

I tried both the MinMaxScaler() and the StandardScaler() to determine if they improved the result of the various algorithms I tested. The MinMaxScaler() did make a difference with other algorithms, but it had no effect on the scores for my final choice of algorithm, I decided to scale the features anyway because it did not harm the result and if I were to add features that were measured in different units, scaling would likely be necessary.

3. *What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: "pick an algorithm"]*

My final algorithm was K Nearest Neighbors Classification. I also was able to meet the project criteria with Decision Tree and Naive Bayes, but the K Nearest Neighbors gave me the highest accuracy and precision. Unfortunately, the recall is fairly low for K Nearest Neighbors. Both the Decision Tree and Naive Bayes Identifiers had better recall. However, K Nearest Neighbors had significantly higher precision.

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 don't 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 if you used, say, a decision tree classifier). [relevant rubric item: "tune the algorithm"]*

Tuning is the process of choosing appropriate parameters to maximize your desired result given your specific situation (e.g., size of data, available hardware, and time available for processing). If you don't tune your parameters well, the algorithm could take too long to run. On the other hand, some algorithms allow you to more fully process data, which would be appropriate for smaller datasets or where time is less of an issue. Finally, you would tune parameters to give you the desired recall, precision and accuracy for the problem.

In the K Nearest Neighbors classifier, I tuned the `weight` parameter to distance, which weights closer points more heavily than more distant points. I then analyzed the various permutations of the `n_neighbors`, `algorithm` and `p` parameters and found that I got the maximum precision and accuracy with 3 neighbors, the ball tree algorithm and p of 5.

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"]*

There are a couple of ways to validate your results. One way is to split the data into training and testing sets. Using this method, you would train your model on the training data alone and then test the model against the testing data. This method can work well if you are sure that the data is being split in a way that does not bias the results. Also, it is very important to ensure that you keep the testing and training data separate. It would be very easy to accidently test on the training data or train on the testing data through a simple typo. This would cause the model to be overfit to the data.

Another way to validate is by using KFold or ShuffleSplit. These methods do away with the necessity of setting aside data to test on which allows you to use all of the data to train and test the model. Additionally, accidentally mixing up the training and testing data is less likely because, once you set the algorithm, it performs the splitting, training and testing automatically. The testing script for this project uses StratifiedShuffleSplit, a combination of the KFold and ShuffleSplit methods, which splits

the data into new training and testing sets 1,000 times and runs the POI Identifier on each of these sets.

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"]*

For reference in this section, here are the accuracy metrics from my top three algorithms and the baseline model that is introduced below:

	Classifier	**Accuracy**
1	KNeighborsClassifier	0.87060
2	GaussianNB	0.85013
3	DecisionTreeClassifier	0.81800
4	Baseline	0.87671

Accuracy is a common metric which benefits from being very simple to compute. It is simply the proportion of correct predictions to total items in the dataset. For instance, the data in this project has 146 people and 18 are POIs. The Baseline model, an algorithm that always predicts the most frequent outcome, has an accuracy of 0.8767 and this accuracy is higher than that of any of the three other algorithms represented above. However, this basic algorithm would never predict that someone is a POI which is the entire point of this project. Therefore, this algorithm would fail in predicting every important datapoint. This extreme example demonstrates the shortcomings of using accuracy alone as a metric.

	Classifier	**Precision**
1	KNeighborsClassifier	0.52502
2	GaussianNB	0.42584
3	DecisionTreeClassifier	0.31823
4	Baseline	undefined

Precision is the ratio of true positives to the sum of true positives and false positives. In this case, it is the ratio of POIs that the algorithm correctly identified to the

sum of POIs correctly identified and the people that the algorithm predicted to be a POI, but were really innocent. In the algorithms shown above, the Baseline's precision is undefined because it will never predict POIs, either correctly or incorrectly, so both the numerator and denominator will always be 0. The KNeighborsClassifiers algorithm shows a significantly higher precision than the others. This precision explains that when a person is flagged as a POI, there is a 0.52502 probability that the person is actually a POI. While this may not seem great, it is literally infinitely better than the Baseline model which had a slightly better accuracy as explained in the previous section. This shows how precision is a more nuanced validation metric than accuracy.

	Classifier	**Recall**
1	KNeighborsClassifier	0.30950
2	GaussianNB	0.35600
3	DecisionTreeClassifier	0.31950
4	Baseline	0

Recall is the ratio of true positives to the sum of true positives plus false negatives. In this project, it is the ratio of POIs correctly identified to the sum of POIs correctly identified and the POIs that the algorithm failed to identify. It is the proportion of actual POIs who were identified. Considering this, I believe this is the most important metric of the three discussed here. In a fraud investigation, you want to flag every person who committed fraud, even if you end up investigating people who are innocent. Given this, I found it disconcerting that the recall rates were so low. The recall value of the KNeighborsClassifier indicates that this algorithm only flags 30.95% of the POIs. I was tempted to include GaussianNB as my final algorithm based on its recall metric alone which is only 4.65% better in absolute terms, but would result in identifying roughly 15% more POIs than the KNeighborsClassifier.