Enron Submission Free-Response Questions

A critical part of machine learning is making sense of your analysis process and communicating it to others. The questions below will help us understand your decision-making process and allow us to give feedback on your project. Please answer each question; your answers should be about 1-2 paragraphs per question. If you find yourself writing much more than that, take a step back and see if you can simplify your response! When your evaluator looks at your responses, he or she will use a specific list of rubric items to assess your answers. Here is the link to that rubric: [**Link**] Each question has one or more specific rubric items associated with it, so before you submit an answer, take a look at that part of the rubric. If your response does not meet expectations for all rubric points, you will be asked to revise and resubmit your project. Make sure that your responses are detailed enough that the evaluator will be able to understand the steps you took and your thought processes as you went through the data analysis.
Once you've submitted your responses, your coach will take a look and may ask a few more focused follow-up questions on one or more of your answers.
We can't wait to see what you've put together for this project!

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"]

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

This project is aimed to build a person of interest identifier based on financial and email data made public as a result of the Enron scandal. A person of interest (POI) is someone who was indicted for fraud, settled with the government, or testified in exchange for immunity. The dataset has records for 146 Enron executives, including their email and financial information characterized by 20 features. The data are provided in the form of a Python dictionary with each individual as a key and the information about the individual as the values. I converted it to pandas dataframe for easier data manipulation.

All the data are numerical types, except email addresses. They are generally normally distributed. I removed two outliers "TOTAL" and "THE TRAVEL AGENCY IN THE PARK", which should not be included in the analysis. The dataset is very sparse, with many data being "NaN". Some fields have more than half of the data being "NaN," such as loan_advances, director_fees, and deferral_payments, etc. I replaced "NaN" in the financial data with zero as they are known to be zero. I then replaced "NaN" in the email data with the corresponding average values for POI and non-POI. Doing this clean-up can largely improve the model performance, as will be shown later.

When I take a closer look at the lable "poi", I noticed that there are only 18 POIs out of the total 144 records. This is a small size and unbalanced label field and leads to the usage of StratifiedShuffleSplit to performance the cross validation.

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”, “intelligently select features”, “properly scale features”]

I ended up using the following features in my POI identifier:

[‘expenses’, ‘other’, ‘exercised_stock_options’, ‘to_poi_ratio’, ‘shared_receipt_with_poi’].

I first tried several algorithms with the all the cleaned datasets, and only the decision tree classifier just pasted the evaluation standards with default settings, with a precision of 0.32, and a recall of 0.33. Then I added 5 new features to see if the performance would be improved:

[‘from_poi_ratio’, ‘to_poi_ratio’, ‘shared_poi_ratio’, ‘bonus_salary_ratio’, ‘bonus_ratio’]

Adding the first three is because the fraction of email exchanges with a POI might be a better identifier of POI. Bonus-to-salary and bonus-to-total payment ratios could also be strong identifiers of POI as bonus is a more hidden way than salary for the fraud to happen.

The final features list was determined by a mix of applying feature selection analysis and human intuition. I first looked at the feature importances from the decision tree algorithm. The top 10 features are listed as below:

```
1.to_poi_ratio:0.3469
2.shared_receipt_with_poi:0.2675
3.expenses:0.1680
4.other:0.1563
5.from_poi_ratio:0.0612
6.bonus:0.0000
7.salary:0.0000
8.deferral_payments:0.0000
9.deferred_income:0.0000
10.director_fees:0.0000
```

It appears that the first five features are dominant as the importance drops dramatically at the sixth feature. Then I applied SelectKBest to find the optimal number of features, which turned out to be 20. The score ranking is listed below:

```
1.from_poi_ratio:0.6771
2.to_poi_ratio:0.6771
3.total_payments:0.6644
4.bonus_ratio:0.6644
```

```
5.bonus_salary_ratio:0.6470
6.restricted_stock:0.6042
7.total_stock_value:0.5949
8.shared_poi_ratio:0.5938
9.to_messages:0.5729
10.restricted_stock_deferred:0.5671
11.shared_receipt_with_poi:0.5451
12.other:0.4873
13.long_term_incentive:0.4097
14.loan_advances:0.3819
15.director_fees:0.3127
16.deferral_payments:0.2824
17.expenses:0.2812
18.from_messages:0.2049
19.from_this_person_to_poi:0.1933
20.from_poi_to_this_person:0.1910
```

Based on the two rankings, I tested the following combinations of features:

Group 1	All 25 features
Group 2	Top 20 features scored by SelectKBest
Group 3	Top 5 features in the ranking of decision tree feature importances: to_poi_ratio shared_receipt_with_poi expenses other from_poi_ratio
Group 4	5 features by human intuition: expenses other exercised_stock_options to_poi_ratio shared_receipt_with_poi
Group 5	5 features by human intuition: deferred_income exercised_stock_options director_fees to_poi_ratio bonus_salary_ratio
Group 6	Top 20 features scored by SelectKBest with decision tree classifier tuned by GridSearchCV

The model performance metrics for the six groups of features are summarized in the table below:

Features	Precision	Recall	F1 Score	Accuracy
Group 1	0.61446	0.61600	0.61523	0.89727
Group 2	0.61406	0.62450	0.61924	0.89760
Group 3	0.67582	0.68900	0.68235	0.91447
Group 4	0.69572	0.67450	0.68495	0.91727
Group 5	0.45065	0.36300	0.40210	0.85607

Group 6	0.63930	0.67350	0.65595	0.90580

When all the original and additional features are included, the F1 score is 0.615, and precision and recall are both above 0.61, which already meet the 0.3 standard. Keeping only the number of features given by SelectKBest does not improve the performance a lot (Group 2). Keeping the five dominant features given by the decision tree importance ranking shows a conceivable increase in the performance (Group 3). Based on Group 3, I manually changed one or two features, and found one combination that gives the highest F1 score across all the testing scenarios (Group 4). This combination gives an F1 score of 0.685, a precision of 0.696, and a recall of 0.675. I tried the decision tree classifier tuned by GridSearchCV with the 20 features selected by SelectKBest (Group 6). It gives a good performance but not as good as Group 4. So I decided to use Group 4 as my final feature list.

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 ended up using the Decision Tree Classifier because it gives the best performance. Before feature selection, I tried Naïve Bayes, K Means Clusters, and Decision Tree. With all the original features included, the tester gives results as follows:

Classifier	Precision	Recall	F1 Score	Accuracy
GaussianNB	0.23642	0.39400	0.29552	0.74953
KMeans	0.22108	0.10800	0.14511	0.83033
DecisionTree	0.32452	0.33150	0.32797	0.81887

After feature scaling, surprisingly, the performance of Naïve Bayes and KMeans dropped sharply. This is probably due to the sparse data structure and the feature scaling is not appropriate here.

Classifier	Precision	Recall	F1 Score	Accuracy
GaussianNB	0.16160	0.82050	0.27001	0.40847
KMeans	0.16333	0.10200	0.12558	0.81060
DecisionTree	0.32725	0.33200	0.32961	0.81993

Due to the poor performance of KMeans, I droppted it at this step. Then I compared the performance of Naïve Bayes and DecisionTree after adding the five new features.

Classifier	Precision	Recall	F1 Score	Accuracy
GaussianNB	0.23642	0.39400	0.29552	0.74953
DecisionTree	0.61446	0.61600	0.61523	0.89727

The performance of Naïve Bayes didn't change much, but that of the DecisionTree is very sensitive to the features and has improved largely with the additional features,

which implies that DecisionTree has more potential to be tuned for better model performance for the given dataset, and I decided to use it as the final classifier.

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 items: "discuss parameter tuning", "tune the algorithm"]

Parameter tuning is the process of optimizing the "settings" of a machine-learning algorithm to achieve the maximum model performance on a given dataset. If this is not done well, the model performance might drop, or you may miss the chance to achieve a better model performance. I used GridSearchCV to tune the decision tree classifier. The parameters I tuned are criterion, max_features, splitter, min_samples_split, and max_depth.

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 items: "discuss validation", "validation strategy"]

Validation performs multiple splits on the dataset and in each split, forms a different training and testing datasets. In each iteration, the classifier is fit on a training set and tested on a testing set. In the next iteration, the classifier is again trained and tested, but on a different set of training and testing data. Cross-validation prevents one from making the classic mistake of training an algorithm on the same data used for testing. If this happens, the test results may show high accuracy of the classifier, but that is only because the classifier has seen the test data before. When the classifier is deployed on a new dataset, the performance may be poor because the classifier was fit and tuned on a very specific set of data. In my analysis, I used the StratifiedShuffleSplit method to do cross-validation as the dataset is small and unbalanced. There are only 18 POI in the 143 people in the 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. [relevant rubric item: "usage of evaluation metrics"]

The final performance of my algorithm is shown below:

Classifier	Precision	Recall	F1 Score	Accuracy
DecisionTree	0.67582	0.68900	0.68235	0.91447

In this project, precision and recall are the ideal performance metrics, giving the unbalanced dataset. Precision in this project means, in all of the persons that are identified as POI by the algorithm, what the percentage is that they are indeed a POI. In the final results, there are 1378 true positives, and 661 false positives, so the precision is 1378 / (1378 + 661) = 67%.

Recall means, out of all the persons who are POIs, how many of them are identified by the algorithm. In the final results, there are 1378 true positives, and 622 false negatives, so the overall recall is 1378 / (1378 + 622) = 69%.