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"]**

   The goal of this project is to analyze the Enron dataset and make use of machine learning to label the people in the dataset as person of interest (when they happen to be a poi).

   Background:

   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.

   Overview of the dataset:

   Enron email and financial data into a dictionary, where each key-value pair in the dictionary corresponds to one person. The dictionary key is the person's name, and the value is another dictionary, which contains the names of all the features and their values for that person. The features in the data fall into three major types, namely financial features, email features and POI labels.

   1. financial features: ['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees'] (all units are in US dollars)

   2. email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (units are generally number of emails messages; notable exception is 'email_address', which is a text string)

   3. POI label: ['poi'] (boolean, represented as integer)

   The POI variable can be true or false; certain variables are marked as true and certain as false. The analysis and use of machine learning can be exploited to build a poi identifier.

   Aim:

Building a person of interest identifier based on financial and email data made public as a result of the Enron scandal.

Outliers:

Outlier analysis when done manually showed two errors:

1. TOTAL row is the biggest Enron E+F dataset outlier. We should remove it from dataset for reason it's a spreadsheet quirk.
2. The second invalid record was for a travel agency called ‘THE TRAVEL AGENCY IN THE PARK’.

   Outliers have to be removed that is the only way to deal with them. Only then the analysis is possible.

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”]

Here is a list of all features:

financial features: ['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees'] (all units are in US dollars)

email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (units are generally number of emails messages; notable exception is ‘email_address’, which is a text string)

POI label: [‘poi’] (boolean, represented as integer)

At an initial stage the features that affect the fact that if the person is a poi are chosen as per intuition. These features are: ‘salary’, ‘bonus’, ‘exercised_stock_options’.

I engineered two features:

1. fraction_from_poi = from_poi_to_this_person/ to_messages

2. fraction_to_poi = from_this_person_to_poi/from_messages

The rationale behind this:

The higher the interaction a person has with the poi, the higher the chance that the person himsef/ herself is a poi.

I made use of SelectKBest in order to decide which ones to choose. The selected features:

[(True, 'exercised_stock_options', 24.815079733218194), (True, 'total_stock_value', 24.182898678566872), (True, 'bonus', 20.792252047181538), (True, 'salary', 18.289684043404513), (True, 'deferred_income', 11.458476579280697)

Here the parameter used was K=5. This gave the maximum precision for classifiers and hence, was selected.

Scaling was done. We made use of MinMax scaling as the units are different, and ranges of the features are also diverse.

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 tried three algorithms in all.

1. **Decision Tree Classifier**

The output got improved from:

Accuracy: 0.80427 Precision: 0.25523 Recall: 0.24400 F1: 0.24949 F2: 0.24617

Total predictions: 15000 True positives: 488 False positives: 1424 False negatives: 1512 True negatives: 11576

to:

Accuracy: 0.86300 Precision: 0.44399 Recall: 0.10900 F1: 0.17503 F2: 0.12837

Total predictions: 15000 True positives: 218 False positives: 273 False negatives: 1782 True negatives: 12727

2. **K Nearest Neighbors**

Accuracy: 0.86380 Precision: 0.23457 Recall: 0.00950 F1: 0.01826 F2: 0.01176

Total predictions: 15000 True positives: 19 False positives: 62 False negatives: 1981 True negatives: 12938

3. GaussianNB

Accuracy: 0.75587 Precision: 0.25385 Recall: 0.42850 F1: 0.31882 F2: 0.37667
Total predictions: 15000 True positives: 857 False positives: 2519 False negatives: 1143 True negatives: 10481

The comparison for the performance metrics:

	Accuracy	Recall	Precision
Decision Tree Classifier	86.3%	44.399%	10.9%
K Nearest Neighbors	86.38%	0.95%	23.45%
GaussianNB	75.58%	42.85%	25.38%

On comparing the three algorithms the performance of DTC are comparatively better and hence, is chosen as the algorithm.

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"]

Different problems have their own intricacies. Machine learning models are parameterized so that their nature can be tuned for a given problem.

Models can have many parameters and finding the best combination of parameters can be treated as a search problem. There are predefined algorithms in python to enable this.

If this is not done well, the performance metrics get affected and are not optimum.

Grid search is an approach to parameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid.

These are the parameters chosen for Decision tree classifier:

```
parameters = {'min_samples_split':[2,4,6,8,10],
              'splitter': ('best','random'),
              'max_depth':[None,2,4,6,8,10]
                                    }
```

Grid search is this efficient parameter tuning technique that helps us solve the parameter discovery problem without using trial and error.

Hence, we get the modified classifier as:

```
DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=2,
            max_features=None, max_leaf_nodes=None,
min_samples_leaf=1,
            min_samples_split=8, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None,
splitter='random')
```

The output got improved from:

Accuracy: 0.80427 Precision: 0.25523 Recall: 0.24400 F1: 0.24949 F2: 0.24617

Total predictions: 15000 True positives: 488 False positives: 1424 False negatives: 1512 True negatives: 11576

to:

Accuracy: 0.86300 Precision: 0.44399 Recall: 0.10900 F1: 0.17503 F2: 0.12837

Total predictions: 15000 True positives: 218 False positives: 273 False negatives: 1782 True negatives: 12727

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 technique that is employed to section your data into training and testing dataset initially. It allows you to train using a set of data and the other piece can be used to validate your analysis. You train the classifier using 'training set', tune the parameters using 'validation set' and then test the performance of your classifier on unseen 'test set'.

The classic mistake is using the same data to test and train. The flaw with doing this is that it does not inform you on how well the model has generalized to new unseen data.

A model that is trained and tested on the data is not generalized enough to handle unseen data. This leads to overfitting.

In order to validate my analysis I made use of stratified shuffle split cross validation developed by Udacity and defined in tester.py file

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 evaluation metrics I used are as follows:

   1. Accuracy: The classification accuracy is given by the number of correct predictions from all predictions made. **The algorithm predicts 86 percent of the time correctly that the person was a POI or a non-POI when the person was a POI or a non-POI.**

   2. Recall: Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives. **Out of all the POIs in the dataset what proportion could be classified by the identifier, this is what recall gives us.**

   3. Precision: Precision is the number of True Positives divided by the number of True Positives and False Positives. **It gives the proportion of people that were actually POIs from the people that were identified by the algorithm as POIs.**

For DTC:	Accuracy	Recall	Precision
	86.3%	44.399%	10.9%