**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.

   The goal of this project is to build an algorithm to identify Enron Employees who may have committed fraud based on the public Enron financial and email dataset. Summary Data as follows (code in "explore_enron_data.py"):
   - Total number of data points: 146
   - Allocation across classes (POI/non-POI): 18 POI, 128 non-POI
   - Number of features used: 21

   Outliers:
   When I graphed salary v.s. bonus (see code "tools/enron_outliers.py), I could see that there was an outlier at the very far top right. I looked up the values in the data, and realized that it is actually for "Total" salary and bonus. This outlier was taken out as it is not a valid salary/bonus for any specific person. After taking out the "Total" value, I re-graphed the data, it looks better, but the data is still a little dispersed, with 4 data points that look like potential outliers. However, upon further investigation, those are actually valid salary and bonus values for some of the highest paid employees, and should be kept. Also, by looking at the enron61702insiderpay pdf, I took out "The Travel Agency in the Park" since it is not a 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.

   First I created two new features, "from_poi_email_fraction" and "to_poi_email_fraction" by transforming the existing features for actual number of to and from emails that are related to poi. Because of the difference in actual number of emails sent/received, the absolute value of poi related emails is hard to compare for meaningful patterns. It makes more sense to use the fraction of emails to/from poi over the total to/from emails, as the values will be more comparable for different people. These two featured did not turn out to be critical to the algorithm. "from_poi_email_fraction" did not make the cut for the top 10 features from SelectKBest, while "to_poi_email_fraction" is 5$^{th}$ most important feature, it did not make it to the final features used in the algorithm, as I only chose the top 3.

   After creating the features, I then ran SelectKBest to look at the top 10 features:

   - 'exercised_stock_options', 25.09
   - 'total_stock_value', 24.46
   - 'bonus', 21.06

- 'salary', 18.57
- 'to_poi_email_fraction', 16.64
- 'deferred_income', 11.59
- 'long_term_incentive', 10.072
- 'restricted_stock', 9.34
- 'total_payments', 8.86
- 'shared_receipt_with_poi', 8.746

I then further tuned the features and used the top 3 in my algorithm ("exercised_stock_options", "total_stock_value" and "salary"). I tried out different numbers of features to include. I started out with the top 10 features from SelectKBest, then I started to take away features one by one from the least important, and see at what point I get the best recall (while also making sure precision is greater than 0.3). I settled on the top 3 features: "exercised_stock_options", "total_stock_value" and "salary" as they yielded the best recall/precision/accuracy/f1 scores.

Feature Scaling: I used minMaxScaler to scale the features I chose.

3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

   I tried both decision tree and Naïve Bayes algorithm and compared their performance. I ended up choosing Decision Tree because it gave me the highest recall value. I think recall is the most important metric for this project. (explained later)

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?

   Parameter tuning means to choose a set of optimal hyperparameters for a machine learning algorithm. It is an important step in machine learning to enable the algorithm to perform the best. I used GridSearchCV to further tune the parameters for the Decision Tree classifier. Grid Search will test out all a set of parameters I defined and choose the best value for each parameter to optimize my target criteria – which I have set to be recall. I looked into "criterion", "min_samples_split", "max_depth", "min_samples_leaf" and "max_leaf_nodes" for tuning.

5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

   Validation is testing out the algorithm on the test data, to evaluate model performance. A classic mistake is to test out the algorithm on the training dataset instead of the test dataset.

For validation, I ran the tester.py code to perform Stratifield Shuffle Split. Stratifield Shuffle Split is a cross validation method that combines StratifiedKFold and ShuffleSplit, which returns stratified randomized folds. This makes sense because there are very few POIs in the dataset so it is important to use to ensure the data is randomized and validated multiple times to obtain the correct metrics.

My results are as follows:

- Accuracy: 0.80092
- Precision: 0.35838
- Recall: 0.37200
- F1: 0.36506
- F2: 0.36919
- Total predictions: 13000
- True positives: 744
- False positives: 1332
- False negatives: 1256
- True negatives: 9668

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.

   Evaluation Metrics: Since the algorithm is trying to correctly identify and flag potential POIs, it is most important to maximize recall. There is no real harm in identifying someone as POI when they are not (false positive). This algorithm should provide a starting point for the whole investigation, so the objective should be to identify as many POIs as possible, which is represented by recall.

   For my algorithm, recall is ~0.38 which means that the algorithm can identify 38 % of all actual POIs. My precision is ~0.37, meaning out of all the POIs identified by my algorithm, 37% are actually POIs.