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"]**

The goal of the UD120 final project is to provide a machine learning solution that allows us to achieve precision and recall rates of at least 0.3 in predicting whether or not a person is a so-called POI in the Enron case (i.e. person of interest); simply put, if a person is very likely to have committed fraud. The provided dataset contains 146 data points (employees), and 21 features to begin with. The features are a selection of financial and email data (provided as a Pickle file), including also the one feature that labels a person as POI or Non-POI. Since we have that label-feature available, we can rely on supervised-learning techniques to approach the problem, i.e. use that feature to train our model. By examining the dataset, I could spot two distinct outliers (TOTAL and 'THE TRAVEL AGENCY IN THE PARK') which I removed from the dictionary. One further entry ('LOCKHART EUGENE E') has no valid values for none of the features and so I decided to remove that person as well. Finally, my intuition told me to remove features that most probably wouldn't prove of benefit to my further investigation; as a result I removes the features: 'EMAIL ADDRESS', 'TOTAL PAYMENT', 'TOTAL STOCK VALUE', 'OTHER' - since we have discrete financial data available for payments and stocks, we probably don't need their summed values.

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 used SelectKBest to find the best-scoring feature(s) for the winning algorithm that finally performed the job. Before applying SelectKBest I trimmed the features list, removing features with more than 50% NaN values. Additionally, I engineered two features which calculate the proportion of emails sent-to/received-from POIs, to the total number of emails sent/received for any given data point in the dictionary (employees). I engineered those features based on the rationale that people who sent/received mail to/from POIs are more likely to be POI themselves. That resulted in 14 features, before applying SelectKBest, including the label-feature (POI, Non-POI). With feature scaling, I applied the MinMaxScaler to scale the final features before going into the KNN and SVM algorithms - this was necessary because these algorithms use Euclidean distance between two data points in their computations, and therefore we need to bring all features to the same level of magnitude. The other two algorithms I decided to try out were Gaussian NB and Decision Tree Classifier - these do not require feature scaling. The features that were used in the end are ‘exercised_stock-options’ and ‘bonus’, depending on the preferred algorithm.

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 winning algorithm(s) based on the resulting F1 score were GaussianNB and Decision Tree - **both return the exact same results**. However, as you can see below, based on the Test Classifier results, Decision Tree yields a better combination of precision/recall than GaussianNB. In addition to the aforementioned, I tried out K Nearest Neighbors, and Support Vector Machine. To test/measure all four algorithms, I engineered a function that loops through them, while increasing by one (up to six) the number of features to use at each cycle (SelectKBest). Why did I have my function pick according to F1? Since we know that Non-POIs account for 87% of the dataset, the different classes (POI, Non-POI) result in significantly heterogeneous weightings. If we were to rely on the classifier’s accuracy alone, a very likely value of 0.87 (based on Non-POIs in our dataset) would be misleading. F1, on the other hand, provides a better and more reliable metric for measuring an algorithm’s performance, since it takes into account both, precision and recall.

===
Classifiers used (including Classification Report):

GaussianNB(priors=None, var_smoothing=1e-09)
Number of features used: 1
The features (incl. scores) used are: [['exercised_stock_options', '24.25047235452619']]

and ...

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=6,
min_weight_fraction_leaf=0.0, presort=False, random_state=42,
splitter='best')
Number of features used: 2
The features (incl. scores) used are: [['exercised_stock_options', '24.25047235452619'], ['bonus', '20.25718499812395']]

	f1-score	precision	recall	support
macro avg	0.81	0.96	0.75	43
micro avg	0.93	0.93	0.93	43
non_poi	0.96	0.93	1.00	37
poi	0.67	1.00	0.50	6
weighted avg	0.92	0.94	0.93	43

Gaussian NB - Test Classifier Result:
Accuracy: 0.90409 Precision: 0.46055 Recall: 0.32100 F1: 0.37831 F2: 0.34171
Total predictions: 11000 True positives: 321 False positives: 376 False negatives: 679
True negatives: 9624

DecisionTree - Test Classifier Result:
Accuracy: 0.83208 Precision: 0.45490 Recall: 0.46150 F1: 0.45818 F2: 0.46017
Total predictions: 13000 True positives: 923 False positives: 1106 False negatives: 1077
True negatives: 9894
===

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”]

Tuning the parameters of an algorithm means finding ‘the’ combination of parameters that yields the best results for the chosen algorithm. If not tuned properly, an algorithm might miss patterns in our dataset, and therefore perform poorly. I chose GridSearchCV to tune the parameters at each of my functions cycles, and for each one of the tested algorithms. Interestingly enough, GaussianNB which does not support parameter tuning, was one of the two winning algorithms - according to Test Classifier results, it was the second best. Here as well, the best set of parameters for each algorithm, at each cycle, was chosen based on F1 scoring. Below is an example of how this was performed on the Decision Tree Classifier.

```
tree = DecisionTreeClassifier()
parameters = {'criterion': ['gini', 'entropy'],
        'max_features': ['auto', 'sqrt', 'log2'],
        'min_samples_split': [2,3,4,5,6,7],
        'random_state': [0,10,42]}
clf_tree = GridSearchCV(tree, parameters, scoring='f1')
```

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 is the process in which one validates the efficiency of an algorithm’s performance by splitting the available data into a training set (typically 70% of the data), and a test set (the remaining 30%). If not done properly, this can lead to overfitting - according to Wikipedia, overfitting is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may, therefore, fail to fit additional data or predict future observations reliably". In my solution, I used the train_test_split utility from SKlearn’s model_selection module (former cross_validation) to fit/train my algorithm on 70% of the dataset, and then used the remaining 30% to make predictions and compared those with the provided labels (POI, Non-POI).

6. Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that say something human-understandable about your algorithm’s performance. [relevant rubric item: “usage of evaluation metrics”]

The metrics that I used were precision, recall and their combined F1 score. With Decision Tree, for example, and the results returned by tester.py, the precision score (which is an average score) stands at 0.45 (not bad at all) - what that means is the 45% of all people were correctly identified as POIs and Non-POIs, whereas the remaining 55% were incorrectly identified as POIs and Non-POIs. **Recall**, on the other hand, reads 0.46 which means that 46% out of all POIs and Non-POIs were correctly identified, whereas the remaining 54% were incorrectly

identified as their respective counterparts (instead they should have been identified as POIs and Non-POIs respectively). Finally, F1 is a combination of the two, and the better the score, the better the algorithm has performed - in this case, the F1 score stands at 0.45.