**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”]**  
     
   In this project, we are trying to create an algorithm that will correctly predict, based on certain features, whether an individual in a dataset is like to be a person of interest in related to the fraudulent activities carried out by former Enron leaders. Machine learning is useful for this as it can be a tool to sort through large volumes of data, identify which features will be most useful for this, and carry out the predictions on large volumes of data.   
     
   Since most of my features were financially related, I elected to use visualization to identify the outliers and inspect them. The challenge with this data set is that within the dataset there are some very large variations. One outlier was clearly invalid data: the TOTAL. I removed that from my dataset. But investigating the next largest outliers, I see that these are not only valid data, but actually are examples of persons of interest, so it was definitely better to keep them in.
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”]**

Since many of the crimes for which the individuals were indicted for were related to financial practices such as insider trading, I felt financial features would probably be most useful in the algorithm. In addition, since “birds of a feather flock together” I felt the ratios of emails to and from POIs for each person would likely be useful as well in order to identify those that had high interaction with others identified as persons of interest.

I felt a ratio was more meaningful than the straight counts provided in the original data because some of the individuals in the data set were much more prolific email writers than others, so computing a ratio of each compared to the total amounts sent and/or received scaled these counts as compared to the other individuals in the dataset. Since the total emails sent/received were different data fields within the feature set, neither the machine learning algorithms I chose nor the scaler would make this adjustment, so I manually created these as new data points within the initial data dictionary. I tried combinations of most of the financial metrics, but ultimately landed on salary, bonus, total payments, and total stock value. The highest salary and bonus numbers, when graphed, easily identified some of the highest-profile POIs in the dataset – so I felt I was on the right track with them. Since many of the different payment types, and/or stock figures were rolled up into the measures for total payments and total stock value, these seemed to be good representative measures to capture those metrics; with a minimum number of NaNs/blank fields. Although, upon inspection, there were still as significant amount of NaN’s in these fields, there were only a few POIs that had NaNs in either of these fields.

I worried that the NaNs might distort the number of non-POIs identified in the data – since while testing the Random Forest algorithm, I kept seeing very low precision and recall scores for POIs, but very high numbers for non-POIs. I tested this by removing the records that had NaNs in the financial data, and found that while accuracy suffered a little, both the precision and recall increased significantly.

For example, with the following algorithm and parameters:

RandomForestClassifier(bootstrap=False, class\_weight='balanced', criterion='entropy', max\_depth=None, max\_features='auto', max\_leaf\_nodes=15, min\_impurity\_split=1e-07, min\_samples\_leaf=2, min\_samples\_split=2, min\_weight\_fraction\_leaf=0.3, n\_estimators=5, n\_jobs=5, oob\_score=False, random\_state=16, verbose=0, warm\_start=False)

The results with the dataset including NaNs were:

Accuracy: 0.79367 Precision: 0.36226 Recall: 0.72000 F1: 0.48201 F2: 0.60125 Total predictions: 15000 True positives: 1440 False positives: 2535 False negatives: 560 True negatives: 10465

The results with the dataset excluding NaNs were:

Accuracy: 0.74050 Precision: 0.48888 Recall: 0.83550 F1: 0.61683 F2: 0.73174 Total predictions: 8000 True positives: 1671 False positives: 1747 False negatives: 329 True negatives: 4253

While coding my algorithm, I did implement a scaler, specifically because of the wide variation in the financial metrics, and their variation in relation to each other. This did, as well, scale my email related metrics, which was not strictly necessary as they were already proportions…however, I didn’t feel that this would be a significant enough impact to want to go through the process of splitting the data and scaling them differently. I ultimately removed the scaler, there was no significant impact to scaling the data when using the random forest algorithm which I ultimately went with.

1. **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 began by using the Naïve Bayes algorithm as suggested by the skeleton code.I wanted to start with this as it was the simplest to get going, and quickest to run – which made it easier to get my initial coding up and running. However, I did expect that my final selection would be something different, as I had a feeling that some of my chosen features would wind up being more important in prediction than others, and I knew that Naïve Bayes tends to weight features equally. The initial accuracy (on test data imported into POId) was 86%, and precision was .36, but the recall was just over .20 (at .209) and I wanted to see if any of the other algorithms came out with initial values that were better.   
     
   Next, chose an SVM. The SVM had a high recall score on the test data, and with tuning was able to get it up to .67. However, this came at the expense of the precision score, which dropped under .3 and the accuracy, which was below .7. Ultimately, through tuning and testing, I was able to get to a trade-off where the accuracy was .88, the precision was .6 and the recall was .5. This algorithm took a very long time to calculate, and did not play nice with the tester code – several runs resulted in a divide by zero error. Of those responses I was able to get from it, I was not completely satisfied because I really wanted to see a higher recall score than precision (if I had to choose).

I tried a Decision Tree classifier next – I was a little leery of this given the smallish data set, and worried about the potential for overfitting. I ran the data a few times with a few different parameters. While accuracy declined a bit, I did see an increase in recall, albeit with an accompanying reduction in precision. I was also interested to view the feature importances that resulted from this algorithm. I had a feeling I might find that some of my features might prove to have a very low impact on the final result, and perhaps I might be able to reduce the number of features I was using. While there was some variance in the importances assigned, I didn’t find any that were so low that I would want to remove the feature altogether.

Since I saw some change in recall with the use of decision trees, I thought it might be worthwhile to try a Random Forest classifier. My hope was the use of the Random Forest would preserve some of the benefits I had seen in using Decision Tree, while addressing my concern about overfitting and reducing error. Ultimately, after testing many different permutations, the highest recall I was able to get with the scorer was .836, with an accompanying precision score of .49, and an accuracy of .741. I would have liked to see all three higher. I was able to get the accuracy above .88 at one point, but the accuracy was in the wrong direction – it was identifying far more zeros than it was correctly identifying true POIs.

1. **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”]**I tried many variations of parameters for each of the algorithms I chose, but did the most testing on the Decision Tree, Random Forest, and SVM. For Decision Tree, I played with the criterion, max\_features, max\_depth, min\_samples\_split, and min\_samples\_leaf parameters. With Random Forest, I tested various settings of all of these as well as the n\_estimators parameter. With SVM, I tested various combinations of C, kernel, gamma, class\_weight, and probability.  
     
   This kind of tuning is important because you can get very different results depending on these settings – for example, just a different of .5 between C values in the SVM made very large differences in the metrics. Sometimes a slight adjustment in one of the parameters could knock both precision and recall to 0. Others could take accuracy from from .69 to .88. So, what you start with as far as the defaults on an algorithm may not ultimately get you the best answer out of the gate; but tuning these different parameters can help you get more refined responses.
2. **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”]**At it’s most basic, the process of validation is the way you can assess how your algorithm is doing. The simplest form is splitting your data set into separate training and testing sets. You then train your algorithm using the training data, then use the testing data to test how well it actually works. If you do not properly separate these sets, or do not use them correctly, you can wind up thinking your algorithm is doing a better job than it is. For example, if you train and then do your predictions on the same data, your scores are bound to be high – but that doesn’t mean that it will work equally well on a different, but similar data set. By training on one portion of the data, then testing on the other, you can see if the algorithm and the specific tunings you’ve applied to it will generalize to a different set of data effectively. I followed the training/testing model of validation in my analysis. Another option would have been to do Kfold cross validation, which performs several different tests, which can give you an increased sense of performance across different subsets of data, however, in this case I did not feel it necessary to carry it out to that level for this particular analysis.
3. **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”]**Ultimately, my best score using the random forest model (using the tester) was .49 for precision and .836 for recall. The .836 was the best recall score of the various algorithms and parameters I tried.

I was specifically focused on recall as I wanted to identify as many positive cases as I could correctly. The .836 score there was above .3 as per the project assignment, but still meant that 16.4% of the positives were not identified. In other words, 16.4% of the POIs that existed in the data were incorrectly identified as being non-POIs. Meanwhile, the precision was at .488 –meaning that when it did detect a POI, it did so correctly 48.8% of the time, which means that 51.2% of those identified were falsely identified as POIs when they were not.

**Resources:**

Analytics Vidya - How to Improve Class Imbalance using Class Weights in Machine Learning

<https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/>

Big Data Made Simple – Dealing with Unbalanced Class, SVM, Random Forest and Decision Tree in Python

<https://bigdata-madesimple.com/dealing-with-unbalanced-class-svm-random-forest-and-decision-tree-in-python/>

Developers.google.com – Classification: Precision and Recall

<https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall>

Scikit Learn – Choosing the right estimator

<https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html>

Stack Abuse -Python: How to Remove a Key from a Dictionary

<https://stackabuse.com/python-how-to-remove-a-key-from-a-dictionary/>

Stack Overflow - Get all keys of a nested dictionary

<https://stackoverflow.com/questions/39233973/get-all-keys-of-a-nested-dictionary>

Toward Data Science – Fine tuning a classifier in scikit-learn

<https://towardsdatascience.com/fine-tuning-a-classifier-in-scikit-learn-66e048c21e65>

Toward Data Science – Pros and cons of various Machine Learning algorithms

<https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6>

TutorialKart – How to set Color for Markers in Scatter Plot in Matplotlib? (tutorialkart.com)

<https://www.tutorialkart.com/matplotlib-tutorial/matplotlib-pyplot-scatter-color/#:~:text=Matplotlib%20Scatter%20Plot%20%E2%80%93%20Markers%E2%80%99%20Color%20To%20set,or%20matplotlib%20inbuilt%20color%20strings%2C%20or%20an%20integer>.

Matplotlib – List of Named Colors

<https://matplotlib.org/stable/gallery/color/named_colors.html>

W3 Schools – Matplotlib Scatter

<https://www.w3schools.com/python/matplotlib_scatter.asp>