Aimee Rose
Udacity/WGU

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”]**

   The initial dataset contained 146 items. One of these was the total, which was removed, leaving 145. Of the 145, 18 were POIs. With the features I selected, the final dataset had 132 points. No POIs were dropped, so the data before the train/test split had 18 POIs.

   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 looked at the financial metrics first. Considering that insider trading involves the trading of stock, my first choice for financial metrics were stock related: exercised stock options, restricted stock, and total stock value. I was initially most interested in the exercised stock options metric, and began with that one by itself – but I continued to get very low precision and recall scores (often 0 each). On the opposite side (by running a classification report on the test dataset), I saw that the precision and recall listed for the 0 (non-POI) was very high, near 1.

I wondered why this might be and my first thought was then to wonder about how many 0s and/or NaNs there might be in the data set. From the total data set of 146, 145 after the removal of ‘total’, only 101 remained after the feature format function was used. Examining the contents of the field showed that there were 44 NaNs, of the 145 data points. Because the field was the only field chosen, this mean that when using the defaults of the feature format function, these values, which constituted 30% of the dataset, were being converted to zeros by the function, then those were being removed as the function was then removing any instances where all chosen features were 0.

I did not like this result, so my choice then was to either change the feature I was using or add features. I looked at the restricted stock feature and had similar results. Only slightly fewer (36) had NaN values and one of those was for a person of interest, so I did not want to see that dropped from the data set. Looking at the other available stock features, I then had restricted stock deferred, and total stock value. Since the restricted stock deferred was a sub-set of the restricted stock measures, I hypothesized that this field would be as likely or more to have NaNs. This proved to be true – as a full 128 of the values in this field had NaN values, and 18 of those were for POIs. This left the total stock value field.

I guessed that the total stock value field, which would be a roll up of several types of stock holdings, would be less likely to have NaN values. This proved to be true with only 20 NaN values – which was 14% of the total – but none of the NaN values were POIs. I settled on using this as my stock measure. However, because using it alone would cause records to be dropped, I decided I would still want to use additional financial fields – hoping that some of these would have values for the records that had NaN values and would also provide addition predictive value.

In deciding the other financial features to use, I graphed the data in these fields, not only looking for value of these fields as inputs for the algorithm, but also for outliers that could distort them. In the process, I found that the bonus and salary features had some outliers that were really interesting, and it seemed clear that the higher values in these columns seemed to correspond to POIs. I tried adding each of these individually to the features, along with total stock value – adding just bonus raised the recall, while adding salary lowered the recall but increased the precision. Adding both was the best of both worlds, increasing both values.

That left the “birds of a feather” measures. As mentioned previously, these were calculated features I formed by dividing the amounts of emails sent, received, and those on which the recipient was copied along with a POI by each corresponding total number of emails (sent or received, depending on the context. 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 a scaler would have made the adjustment.

I tested each of these features singly along with the financial metrics and tested them together. Ultimately, including all three improved the scores from those given by using the financial features alone; and also improved upon the results if each were included singly with the financial metrics, or alone. In fact, running the feature importance command showed that these three were given significant feature importances – two at .2 and one at .4. Interestingly, the feature importances for salary and total stock value were below both of these, at 0. This made me question including these features in my algorithm, but despite their importances, removing them resulted in a decrease accuracy, precision and recall when run on the test data. So, I chose to leave them in, along with bonus – which had a feature importance of .2.

Tester results with the financial metrics only:

```
Accuracy: 0.67585     Precision: 0.24160     Recall: 0.51750        F1: 0.32941    F2: 0.42128
Total predictions: 13000
True positives: 1035   False positives: 3249  False negatives:  965 True negatives: 7751
```

Tester results with both financial metrics and “birds of a feather” metrics.

```
Accuracy: 0.72350     Precision: 0.30881     Recall: 0.75550        F1: 0.43842    F2: 0.58598
Total predictions: 14000
True positives: 1511   False positives: 3382  False negatives:  489 True negatives: 8618
```

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.

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 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 .756, with an accompanying precision score of .31, and an accuracy of .723. 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.

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"]**
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.
In addition, I did implement a Grid Search using GridSearchCV, but ultimately switched to manual tuning as the suggestions from the grid search continued to prioritize increased accuracy at the expense of increasing recall, which was what I was trying to target.

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 .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.

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"]**

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.

Initially, I followed the training/testing model of validation in my analysis – as running the accuracy score, precision score, confusion matrix and classification report gave me a very quick way to get a sense of how my tunings were impacting my results – as the tester script took a longer time to run, especially with the more resource-heavy algorithms such as the SVM. I knew, however, that I wanted to do my final tests via the tester since it used the StratifiedShuffleSplit to split the (admittedly small dataset) into multiple randomized sets to better simulate algorithm's performance on a larger dataset.

In each phase of my testing, both when I was looking at different algorithms, choosing the features to use in my algorithm, and tuning the algorithm, I initially tested the result of my adjustments by running the above scores on the test set in the poid script, and if I thought I was close – I ran the tester and saved the results for comparison. The results from the tester were generally lower than those run on the smaller data set, which is to be expected as in getting the results from a dataset that's been randomized and split multiple ways. We are better seeing the algorithm's ability to generalize across larger datasets, and the algorithm's weaknesses (and potential overfittings) are more evident.

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"]**
Ultimately, the best results I got from the random forest model (using the tester) was .31 for precision and .756 for recall. The .756 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, and I was not as concerned with false positives (the algorithm falsely identify POIs). I would rather have false positives than miss a POI. The recall score, I ultimately got resulted in 1511 true positive identifications out of 2000 total positives in the dataset. The .756 score there was above .3 as per the project assignment, but still meant that 24% of those that should have been predicted as positives were not identified. In other words, 24% of the POIs that existed in the data were incorrectly identified as being non-POIs. Meanwhile, the precision was at .31 – meaning that when it did detect a POI, it did so correctly 31% of the time, which means 69% of those identified were falsely identified as POIs when they were not. Were this being used "in the wild" – the results would definitely need some addition due diligence to verify if any identified positives were true positives (i.e., if the individual had been truly engaged in some kind of fraud).

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

Geeks for Geeks – Sklearn.StratifiedShuffleSplit() function in Python
https://www.geeksforgeeks.org/sklearn-stratifiedshufflesplit-function-in-python/

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

Stack Overview – How to count the occurrence of certain item in an ndarray?
https://stackoverflow.com/questions/28663856/how-to-count-the-occurrence-of-certain-item-in-an-ndarray

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