Enron Submission Free-Response Questions

by Jamie Brand

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 'Enron corpus' is a collection of hundreds of thousands of emails from enron employees. Enron is a company that went bankrupt as a result of massive amounts of fraud from some of its employees. We are interested in seeing if we can identify those employees that were involved in the fraud from their emails. The dataset we have for this project is derived from the Enron corpus. The dataset has entrys for each employee with emails in the corpus. For each employee there is the number of emails sent, the number received and the number received or sent to POIs (persons of interest). It also has additional features regarding the financial affairs of the employees.

In total there are 145 people in the dataset of which 18 are POIs. For each person there are 21 features. All of the features apart from POI are missing values. In the most extreme case (load advances) only three people had values that were not "NaN". Directors fees and Restricted Stock transferred also had a very high number of "NaN" values. In cleaning the set I removed the 'TOTAL' value which appeared as an extreme outlier and was obviously not a useful data point. I also removed two employees details (BHATNAGER SANJAY and BELFER ROBERT) as they both repeatedly had negative values for financial data when it should have been positive (and vice versa). While we could simply change the negative values to positive and the positive to negative, it is clear that an error has occurred when entering the data and it would be unwise to make presumptions about what that error is.

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"]

While checking for outliers I went through each feature showing the top and bottom 5 values with the name and whether they were a poi. From this I manually selected several features which appeared to be useful for classification. These were 'exercised_stock_options', 'total_stock_value', 'bonus' and 'deferred income'. This gave me reasonable results but I wanted to refine my features using selectKbest. I used the 'f_classif' score function and experimented with different numbers of features (using the testing.py script). The best results I found were using 5 features. These were, with their relevant scores: 'salary' (18),'exercised_stock_options'(24.7),'bonus'(20.5),'total_stock_value'(23.9), and 'deferred_income' (11.3). The code for the selectkbest process is in the 'choosefeatures.py' script.

I engineered additional features for the fraction of email to and from POIs as well as the fraction of emails received that shared a receipt with another POI. These were added to the features. Feature scaling was not done initially as it is not needed by the Naive Bayes classifier. I added Feature scaling using a custom function. It is worth noting that scaling the features this way is more effective than

adding a scaler in a pipeline as it scales the amounts by the minimum and maximum of the whole set, not just the training or testing set. I assume this is acceptable as it is no different than adding a new feature to the dataset.

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 first tried the NaiveBayes algorithm. This gave me quite good results but I wanted to try additional algorithms. Next I tried a decision tree. This also gave me acceptable results but was not as good as the Naive Bayes. Using the decision tree with Adaboost improved the scores but still not to the level of the Naive Bayes. I also tried SVM with RBF kernel and K means clustering. Both of these required that I performed feature scaling on the features. K means clustering had the best recall value but a very low precision. I found my best result using an SVM with the RBF kernel.

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"]

Some algorithms come with defaults parameters that you can expect to give a good result. Others do not show their effectiveness until the parameters are changed. I used trial and error to find the optimal, or close to optimal parameters for each algorithm. For the Scalar Vector Machine I eventually chose to use, I chose to use the 'rbf' kernel and a very high 'C' value of 11,000. The other important parameter with the 'rbf' kernel is the 'gamma' but I found that the default value provided optimal results for my classifier.

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"]

For validation I used the test_classifier function in the 'testing.py' file. I reduced the number of folds to 200 to increase the speed of testing. One classic mistake to make in validation is to overfit the classifier to the data. This can easily happen if the same data is used for training as testing. It can still happen if the training and testing data are different as the user is likely to adjust the parameters to optimise the scores from the testing data. The 'test_classifier' function uses the sklearn 'StratifiedShuffleSplit' object to try a number of different training and testing sets and make an average of the results.
It should be noted that using 'StratifiedShuffleSplit' does not alleviate all overfitting problems. I was experimenting with the 'RandomForestClassifier' and found I was getting exceptionally good results. After further analysis I discovered that the 'Warm_Start=True' parameter was allowing the classifier to remember training from previous tests. This meant that the classifier was training on all the data, not just the training data for each test.

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 'Accuracy' metric describes the total number of predictions the classifier got correct. The average

accuracy for my algorithm was 0.86393. While this seems like an important metric it is not very useful for our data. As there are only 18 POIs out of 145 people in the dataset, if the algorithm predicted that everyone was not a POI it would already have a very high accuracy of about .88. Much more important for us are the precision and the recall. The precision amount of true positives divided by the total number of positives. In our case that is the number of actual POIs divided by the number of people that were predicted to be POIs. In my model the average precision was 0.52905. The recall is the number of correctly predicted POIs divided by the total number of POIs.

In an example like this were we are trying to predict individuals who may be guilty of Fraud, the precision and the recall are important. The recall is important because we want to find as many of people committing fraud as possible and the precision is important because we do not want to waste resources investigating those who are not committing fraud.