Enron Machine Learning Algorithm

By Will Roberts

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?*

The aim of the project is to build an algorithm that identifies which Enron employees may have committed fraud leading to the company's collapse in 2002. Machine learning is an excellent tool for the task, for it can neatly streamline the process of combing a large dataset of features to determine which are significant and how patterns in the data can predict an attribute for a subject. In this project, that prediction is whether an employee was a person of interest (POI) in the Enron scandal, and the features were financial data and email records. There are 14 financial features ranging from salaries and bonuses to various stock options, and the 6 email features include both emails sent and received and whether persons of interest were on the other end of the messages. Finally, a person of interest is defined as an Enron employee who was indicted, settled without admitting guilt, or testified in exchange for immunity.

Before choosing an algorithm, the dataset was first scrutinized for outliers. Immediately two obvious ones became apparent. The first was exposed when plotting the financial features salary and bonus. A feature named "TOTAL" appeared with extremely large numbers for each feature, and it was determined this outlier was a result of the scraping technique used to gather the dataset. With this is mind, the names of the rest of the employees needed to be checked, and one other non-person, "The Travel Agency In the Park", was also removed. In an effort to gauge the completeness of the dataset, a list of each employee and the number of filled out features was created. Because of the relatively small sample size of employees and even smaller group of persons of interest, it was determined that although many entries were missing plenty of features, they must be treated with care when considering removal. In the end, only one more employee was deemed an outlier: "LOCKHART EUGENE E", who had zero financial or email features available. After outlier removal, there are 144 employees in the dataset with 18 of them being POIs.

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?*

In preprocessing to prepare the dataset for use in the algorithm, SelectKBest was used in feature selection and passed through a grid search to determine the ideal number of features for the given algorithm. The five features selected for the final algorithm are as follows with accompanying feature scores: exercised stock options (25.10), total stock value (24.47), bonus (21.06), salary (18.58), and deferred income (11.60). A standard scaler was included in the preprocessing stage as well and proved especially beneficial for the K Nearest Neighbors algorithm. This step precedes the feature selection stage and was included because many of the features had extremely large ranges, and both monetary and email count features were available in the dataset.

Three extra features were engineered in the preprocessing stage. The first, a proportion of exercised stock options to total stock value, was made to look at the value of the options used by employees. As Enron neared its demise, many of the insiders at the company converted their stock in the company to cash, so the hope was to help determine POIs with this information in mind. The other two features pertained to emails records: the ratio of emails sent to other POIs against all emails sent and the ratio of emails received from other POIs to all emails received. In theory, a fraudulent employee at the company would be in constant contact with other like individuals, and thus have a higher proportion of contact with POIs. Before parameter tuning, the dataset before and after introduction of the new variables was run through each of the three algorithms explored. Both the K nearest neighbors (precision: 0.65278, recall: 0.21150, F1: 0.31949) and Naïve Bayes (precision: 0.23578, recall: 0.40000, F1: 0.29668) classifiers returned the exact same F1 score, recall, and precision for both sets, meaning the new variables added no value to the classification. The decision tree classifier (precision: 0.22455, recall: 0.21400, F1: 0.21915) increased performance slightly (precision: 0.23308, recall: 0.21700, F1: 0.22475). In the end, the final algorithm did not include these created features.

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

The final algorithm chosen was the Gaussian Naïve Bayes algorithm with SelectKBest. Out of the box, the Naïve Bayes showed the most promise and after adding the shown feature selection consistently produced the best classifier. In the end, it returned the highest F-1 score of the three algorithms created (F1: 0.38368). The two others formed were a Decision Tree (F1: 0.34219, 12 features) and K Nearest Neighbors (F1: 0.37086, 5 features). Parameter tuning and feature selection were performed for both of these other algorithms in an attempt to improve performance. These included standard scaling, SelectKBest, and PCA. Feature Union was deployed when both SelectKBest and PCA were used.

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?*

Parameter tuning is the process of selecting the best option for each input for an algorithm to maximize its performance given the specifics of a given dataset. It's important because algorithms have many different parameters, and the "out of the box" default parameters generally are not always best for various datasets. The choice of the features selected for each algorithm explored was done through GridSearchCV. This method was deployed because it took into account a wide range for input parameters and streamlined the process of finding the optimal combination. The only tuning needed for the Naïve Bayes algorithm was for feature selection with SelectKBest. To start, a wide range of features (1, 5, 10, 15) were searched with 5 providing the best F1 score (0.38368). As a follow up, the surrounding range of features from 3 to 7 were searched, though 5 still was returned by the GridSearchCV as optimal. The final attempt to improve the algorithm added a range of 1 to 3 PCA components combined with the 3 to 7 range of features, though the resulting optimal algorithm (4 features, 2 PCA components) resulted in a slightly weaker performance (F1 score: 0.38263).

The K-Nearest Neighbors algorithm takes its own set of parameters which again were optimized with GridSearchCV. To start, the addition of a standard scaler, a wide range of n_neighbors (2, 6, 10, 15), a wide range of features (1, 5, 10, 15), and all possible algorithm inputs ('ball_tree', 'kd_tree', 'brute', 'auto') were passed through the GridSearch to return a weak F1 score (0.10). Since 5 features and 2 n_neighbors were chosen here, the next variation passed to the grid search was a range of 3 to 6 features and a range of 1 to 4 neighbors, producing a much better F1 score of 0.34905 (4 features, 1 n_neighbor). Already having accomplished a recall and precision both above 0.3 with these parameters, out of curiosity the n_neighbors range was set at 5 to 8, producing some very interesting results. The F1 score actually increased to 0.37086 along with a large spike in precision, but the recall plummeted to 0.25450. The importance of these metrics will be explained a little later, but this algorithm did not meet a satisfactory recall metric and was deemed obsolete. As with the Naïve Bayes algorithm, PCA component ranges of 1 to 3 were explored before completion, and again it did not add value (F1: 0.23564).

The decision tree classifier underwent the same range of parameter tuning in GridSearchCV for feature selection, criterion, minimum samples split, and minimum samples per leaf. Though the algorithms final optimized metrics (precision: 0.40, recall: 0.30, F1: 0.34) did not outperform the Naïve Bayes classifier, it is worth noting that it took into account many more features than the previous two (12 total), including the “from_poi_to_this_person” feature created in the preprocessing stages.

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 a method of determining whether the algorithm created is predicting what it was built to predict. Generally, in validation an estimate of the performance of the algorithm with be given on an independent dataset to the one that was used to train the algorithm. A major mistake that can be made without proper validation is overfitting to the training data. An overfit algorithm is one that contains too much bias towards the set it was trained on and gives a poor predictive performance on independent data.

In the final algorithm, the Stratified ShuffleSplit cross validator with 100 iterations is used to split the data into training and testing sets. As opposed to a simple train and test split, the Stratified ShuffleSplit method was necessary because of the make-up of the dataset: small and heavily skewed towards more non-POI. Algorithms can be biased towards too heavily or lightly represented classes, and for the algorithm to truly learn the attributes of a sparse class it needs to read as many instances of it as possible. Since the probability of randomly splitting the skewed dataset into a partition representative of the population and full enough to properly train is fairly low, this validation method was the best option.

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.*

The goal of the algorithm is to predict employees that are POIs in the Enron fraud scandal. Therefore, it is more useful to evaluate and potentially include false positives that further investigation could find their true culpability than have many false negatives that are guilty but fall through the cracks to freedom. This is one reason the Gaussian Naïve Bayes algorithm was chosen, for it had the highest recall statistic of the three (0.34550). This means it allowed the fewest employees to be branded clean when they were actually POIs.

The precision metric was also measured in the final evaluation. Precision is defined as the ratio of true positives to the sum of true positives and false positives. With a score of 0.43134, the algorithm predicted that 43% of the POIs deemed by algorithm actually were POIs in the true Enron fraud scandal.

Finally, the main metric used throughout the analysis to test and measure effectiveness was the F1 score. The F1 score takes into account the above precision averaged with the recall. Taking more information into account, it’s a fuller metric to help determine the degree of success the algorithm has. In the end, a F1 score of 0.38368 was achieved.