Sumudu Katugampola
Introduction to Machine Learning
December 4, 2018

Identifying Persons of Interest in Enron Scandal from Enron dataset

1. The dataset provides financial information (payments and stock value) of most of the employees at Enron Corp. The objective of this project is to identify “persons of interest” (POI) who are responsible and took part in the Enron scandal which led to the bankruptcy of the company in 2001.
   Machine learning can be a useful tool in this type of situations as it (an algorithm) can look for patterns in the data and train itself to recognize those patterns, so that once trained it can be used to identify any unknown POIs in the company.
   There was one outlier that I came across when analyzing the financial data. That was, the values in the row “Total” (which had the sum of values for each feature (column)) were included in the every financial feature namely salary, bonus, expenses etc. It was identified by looking at scatter plots of different combinations of features that were selected. And the “Total” was removed from the data dictionary by using the line “data_dict.pop(‘TOTAL’,0)”.

2. Initially I started using the following features;
   ‘salary’,'bonus','shared_receipt_with_poi','from_this_person_to_poi','from_poi_to_this_person','expenses'.
   These features were chosen under the assumptions that most of the POIs would have a connection (relationship) with other POIs and also some pattern would pop out in their salary and bonus data. The “new” feature that I looked at was the sum of the number of emails to and from a POI for each person. That is, ‘from_this_person_to_poi’ + ‘from_poi_to_this_person’. The main reason for choosing this was the possibility of a more prominent connection to a POI with this total number of emails exchanged.
   Feature scaling was implemented in my analysis mainly because the range of values for ‘salary’, ‘bonus’ and ‘expenses’ were very large when compared to those of other features used.
   In the final analysis only the three most important features were used and the feature importances were found by using a Decision Tree (DT) algorithm and following are the importance values that I got for the features that I used.

   Feature = ‘expenses', Importance = 0.46238
   Feature = ‘shared_receipt_with_poi', Importance = 0.28012
   Feature = ‘from_this_person_to_poi', Importance = 0.21334

3. Three different algorithms were tried in this analysis. They were, Decision Trees (DT), Naive Bayes (NB) and Support Vector Machines (SVM). But the final analysis was done by using Decision Trees because it had the largest Accuracy, Precision and Recall values. Following are the performances of each algorithm.

    DT : Accuracy = 0.833, Precision = 1.00, Recall = 0.2

    NB : Accuracy = 0.792, Precision = 0.00, Recall = 0.00

    SVM : Accuracy = 0.750, Precision = 0.00, Recall = 0.00

    When the classifier, features and dataset were analyzed using the code 'tester.py', for DT, following are the values obtained.

    Accuracy = 0.799, Precision = 0.3805, Recall = 0.3255

4. Some parameters in some algorithms need values as user inputs as those values cannot be obtained or inferred from the dataset itself. These parameter values need to be carefully chosen as they could compromise the performance of the algorithm and could lead to a non-optimized output. Finding these optimum vales for the said parameters is called 'parameter tuning' in machine learning context.
In this analysis parameter tuning has been achieved by the method of Grid Search in which, out of a set of reasonable values for each parameter, a combination of values is chosen at a time and the algorithm is run with those values. This process is repeated until the combination which gives the optimum output from the algorithm is found.
In this analysis, following parameters were tuned using the sklearn package 'GridSearchCV'.

    SVM : 'C', 'gamma'

    DT : 'min_samples_split', 'min_samples_leaf'

5. Validation is a process where a trained algorithm is tested from a test dataset usually obtained as part of the same data set where the training data was obtained. If you use of your entire dataset to train your model and use that same dataset to test the model, your algorithm will be biased only to that particular dataset. It will have an almost perfect accuracy only for that dataset but will perform poorly when introduced with a new data point.

In this analysis, the algorithm was trained by using only 80% of the dataset and was validated by using the remaining 20% of the dataset as the test set. 'train_test_split' of the 'cross_validation' package in sklearn was used to split the data into training and testing sets.

And Accuracy Score, Precision and Recall have been calculated validating each algorithm that was used.

6. Following are the metrics that were obtained for each algorithm tested:

    DT : Accuracy = 0.833, Precision = 1.00, Recall = 0.2

    NB : Accuracy = 0.792, Precision = 0.00, Recall = 0.00

    SVM : Accuracy = 0.750, Precision = 0.00, Recall = 0.00

When you run DT (which was the algorithm used in the final analysis) with “tester.py”, following results were obtained:

    Accuracy = 0.799, Precision = 0.3805, Recall = 0.3255

In this context, having a Precision of 0.38 means, if the algorithm identifies a person as a POI, 38% of the time that person is actually a POI.

Having a Recall of 0.32 means, given a person is a POI, 32% of the time the algorithm predicts that person to be a POI.