P5: Identify Fraud from Enron Email

By Ai Tee Kang

1. The goal of this project was to build an identifier to identify persons of interest (POI) through machine learning using the Enron data available. This data includes financial information of top executives and their emails. The initial data consisted of 146 person's name and 21 features that were associated with them. Of the 146 person, only 18 are person of interest. This constitutes about 12% of the dataset.

   Two outliers were deleted from the dataset. The two outliers are namely 'TOTAL' and 'THE TRAVEL AGENCY IN THE PARK'. 'TOTAL' was just the sum total of all the features and does not correspond to an employee of Enron. 'THE TRAVEL AGENCY IN THE PARK' was also not an employee of Enron and was stated in the document 'enron61702insiderpay.pdf' as a debtor where payments were made to by Enron employees on account of business-related travel.

2. Initially all available features were included in the features_list except for 'email_address'. This is because I intended to use SelectKBest to select the best features. Also, in order to avoid selecting features that were highly correlated to each other, I have used PCA as well. In order to get better results with PCA, I have used the StandardScaler on the data. SelectKBest and PCA are used with the help of FeatureUnion. StandardScaler and FeatureUnion are passed through a pipeline before passing through GridSearchCV to get the best features.

   Below is the table of features selected by SelectKBest and its corresponding feature score:

SelectKBest Features	Feature Score
Exercised Stock Options	24.82
Total Stock Value	24.18
Bonus	20.79

   While researching through the Enron case, I realised that most POI committed fraud by inflating the financial standing of the company while making huge amount of money through selling Enron shares. This led me to believe that POIs should typically have high ratio of 'Total Stock Value' against 'Total Payments'. I computed a feature called 'stocks_payment_fraction' which takes the value of 'Total Stock Value' divided by 'Total Payments'. However, it would seem that only 9 of the 18 POIs have above 1 for 'stocks_payment_fraction', with the other 9 POIs having values below 1. This shows that 'stocks_payment_fraction' is not an insightful feature added.

3. The final algorithm I used was that of AdaBoost Classifier. The best parameter determined by GridSearchCV are as below:

   {'features__pca__n_components': 2, 'ada__n_estimators': 30, 'features__univ_select__k': 3}

   The results generated by tester.py are:

   Tester Classification report:

   Accuracy: 0.82727 **Precision: 0.34633** **Recall: 0.33300** **F1: 0.33954** F2: 0.33558 Total predictions: 15000 True positives: 666 False positives: 1257 False negatives: 1334 True negatives: 11743

   The other algorithms I have tested includes Gaussian Naïve Bayes, Decision Tree and Random Forest. However, the best result was achieved with AdaBoost. Below are the results generated from the other algorithms.

   Best parameters for Decision Tree:

   {'features__pca__n_components': 3, 'dt__min_samples_split': 6, 'features__univ_select__k': 1}

   Tester Classification report:

   Accuracy: 0.82553 **Precision: 0.33387** **Recall: 0.31000** **F1: 0.32149** F2: 0.31450 Total predictions: 15000 True positives: 620 False positives: 1237 False negatives: 1380 True negatives: 11763

   Best parameters for Gaussian Naïve Bayes:

   {'features__pca__n_components': 2, 'features__univ_select__k': 3}

   Tester Classification report:

   Accuracy: 0.82900 **Precision: 0.33921** **Recall: 0.29800** **F1: 0.31727** F2: 0.30542 Total predictions: 15000 True positives: 596 False positives: 1161 False negatives: 1404 True negatives: 11839

   Best parameters for Random Forest:

   {'features__pca__n_components': 1, 'rf__min_samples_split': 5, 'rf__n_estimators': 50, 'features__univ_select__k': 3}

Tester Classification report:

Accuracy: 0.86780 **Precision: 0.50740** **Recall: 0.29150** **F1: 0.37028** F2: 0.31861 Total predictions: 15000 True positives: 583 False positives: 566 False negatives: 1417 True negatives: 12434

As can be seen from the above results, decision tree was the only other classifier that was able to achive results above 0.3 for precision, recall and f1 scores. However, adaboost had higher scores for all categories. Decision tree and random forest were not able to achieve 0.3 for recall.

4. Since different algorithms have different parameters, fine tuning the parameters of an algorithm would mean trying out different values for the parameters to get the best result from the algorithm. In so doing, we could reduce over fitting and thus be able to achieve better prediction.

   For my algorithm, I have used GridSearchCV to get the best parameters. GridSearchCV was able to find the best features using PCA and SelectKBest. GridSearchCV was also able to pick the best n_estimator for adaboost at 30 out of the levels that were provided namely [5, 10, 30, 50].

5. Validation is to separate the data into training and test set. Training is done on the training set without the inclusion of the test set. Then testing is done on the test set to see how well the algorithm is able to predict the data. Normally training will be on 70% of the data with 30% kept as the test set. One common mistake is to both train and test on the full set of data. Doing it his way will produce perfect result since the test set is already used for training.

   Since our data is small and there are only 18 POIs in 146 data points, it is not practical to withhold 30% of the data as a test set as this will limit the training set and not produce good results. Therefore, I have used the StratifiedShuffleSplit cross validation method within GridSearchCV in order to perform cross validation on the full set of data. StratifiedShuffleSplit will shuffle and split the data into train-test sets 100 folds as specified. This has produced better results as oppose a simple train-test-split.

6. The two metrics used for evaluation is precision and recall. The AdaBoost algorithm produced a precision of 0.34633 and recall of 0.33300. A precision of 0.34633 means that of all the predictions made by the algorithm, 34.63% is correct or relevant. A recall of 0.33300 means that of all the relevant or correct elements, 33.30% has been predicted as true positives.