Identify Fraud Project Report

Sheng Weng

Background

The sudden downfall of Enron is always an interesting reminiscence in American history that people would like to talk about. Thanks to the well-documented investigation, data scientists are able to dive into the big dataset behind the corporate fraud case and find out some great stories.

In this project, I will use email and financial data to predict persons of interest (POI) in the Enron fraud case with the help of machine learning. At the end of this report, a POI identifier will be built to label those who were indicted for fraud, settled with the government, or testified in exchange for immunity.

Dataset and Outlier

The Enron dataset is in a dictionary format. There are 146 keys in this dataset, including 145 executives' names at Enron and a "TOTAL" key. Among them, 18 are POI and 127 are non-POI. The value of each person's name is another dictionary, which contains 22 features in total and the respective values for that person. A lot of features do not have values, which means they are assigned as "NaN". For example, under the name of 'DIMICHELE RICHARD G', 11 out of 22 features have their specific values.

The "TOTAL" data is an outlier because it represents the summary of the data sheet, so it was removed by hand. Another outlier removed by hand was 'THE TRAVEL AGENCY IN THE PARK' because it's not a person's name. All the other data points were kept because although some financial points seemed to be outliers (Fig. 1), they were not noise or mistake but the true amount of money that a POI may have earned in the corporate fraud.

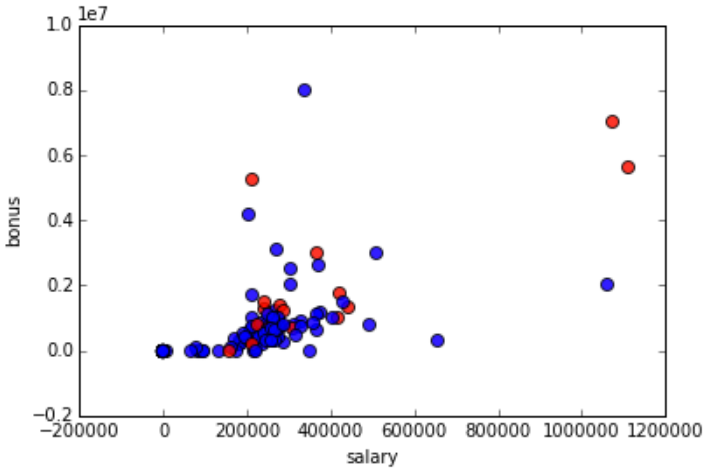

Fig. 1. Red points are POIs. Blue points are non-POIs.

Features

One new feature was added and tested for this project. It was assumed that a POI might have larger number of emails to or from the other POIs. So, the new feature was calculated by dividing the sum of "from_poi_to_this_person" and "from_this_person_to_poi" over all emails that the person had.

I then applied MinMaxScalar feature scaling to the initial feature list as shown below.
features_list = ['poi','salary', 'bonus', 'total_stock_value', 'shared_receipt_with_poi', 'fraction_poi_related_emails', 'loan_advances', 'exercised_stock_options', 'deferral_payments', 'deferred_income', 'director_fees', 'expenses', 'long_term_incentive', 'other', 'restricted_stock', 'restricted_stock_deferred', 'total_payments', 'from_poi_to_this_person', 'from_this_person_to_poi']

In order to select proper features for the algorithm, I first attempted to use recursive feature selection method on support vector machine classifier. However, this method turned out to be too slow to train. I stopped the program after 30 minutes of running without getting any result. I decided not to use SVM as the algorithm for identification. The relevant code is as follows:

```
from sklearn.svm import SVC
clf = SVC(kernel = 'linear')
from sklearn.feature_selection import RFECV
selector = RFECV(clf, step=1)
selector = selector.fit(features, labels)
print selector.support_
print selector.ranking_
```

Then, I decided to use univariate feature selection method to determine what features I would choose. I got the below feature score board:

[18.57570327 21.06000171 24.46765405 8.74648553 5.51850554
7.2427304 25.09754153 0.21705893 11.59554766 2.10765594
6.23420114 10.07245453 4.24615354 9.34670079 0.06498431
8.87383526 5.34494152 2.42650813]

From the above score board we knew that 'salary', 'bonus', 'total_stock_value', and 'exercised_stock_options' are significantly higher than others. So I decided to select these features. I also maintained the feature that I created because I wanted to identify the effect of the new feature on the overall algorithm performance. Then, I used the Naïve Bayes algorithm and the tester code to quickly calculate the precision and recall as I added more features in the list based on their scores.

Selected Features	Precision	Recall
'salary', 'bonus', 'total_stock_value', 'fraction_poi_related_emails', 'exercised_stock_options'	0.461	0.339
'salary', 'bonus', 'total_stock_value', 'fraction_poi_related_emails', 'exercised_stock_options', 'deferred_income'	0.489	0.367
'salary', 'bonus', 'total_stock_value', 'fraction_poi_related_emails', 'exercised_stock_options', 'deferred_income', 'long_term_incentive'	0.476	0.373
'salary', 'bonus', 'total_stock_value', 'fraction_poi_related_emails', 'exercised_stock_options', 'deferred_income', 'long_term_incentive', 'restricted_stock'	0.463	0.379
'salary', 'bonus', 'total_stock_value', 'fraction_poi_related_emails', 'exercised_stock_options',	0.401	0.318

'deferred_income', 'long_term_incentive', 'restricted_stock', 'total_payments'		

Table 1

Table 1 indicates that when the features are ‘poi', 'salary', 'bonus', 'total_stock_value', 'fraction_poi_related_emails', 'exercised_stock_options', and 'deferred_income', the precision is highest, and the recall is also reasonably high.

Algorithm

Tuning the parameters is an important part in setting up the algorithm. The parameters can alter the performance of the algorithm in unexpected ways if they are not handled properly.

1) Naïve Bayes

Naïve Bayes normally doesn’t have parameters to tune.

a) Testing result with the new feature
Accuracy: 0.86447
Precision: 0.48901
Recall: 0.36700
F1: 0.41931
F2: 0.38628
Total predictions: 15000
True positives: 734
False positives: 767
False negatives: 1266
True negatives: 12233

b) Testing result without the new feature
Accuracy: 0.86787
Precision: 0.50577
Recall: 0.39450
F1: 0.44326
F2: 0.41266
Total predictions: 15000
True positives: 789
False positives: 771
False negatives: 1211
True negatives: 12229

With the new feature the Naïve Bayes algorithm would give lower accuracy, precision and recall as compared to the results without using the new feature. This might largely due to the fact that the new feature could not reliably reflect whether a person is POI or not. I decided to abandon the new feature in order to make the performance better.

2) Decision Tree

Decision Tree does not need feature scaling. Two parameters in the Decision Tree algorithm were tuned by hand. Table 2 shows that the best precision and recall were found when min_samples_split = 1. Table 3 indicates that when min_samples_split = 1 and splitter = "random", the precision and recall are near 0.3.

Min_samples_split	Average precision	Average recall
1	0.284	0.276
2	0.281	0.269
3	0.277	0.243

Table 2

splitter	Average precision	Average recall
best	0.284	0.276
random	0.287	0.291

Table 3

Feature importance:
[0.16309031 0.14796238 0.24324947 0.28221187 0.16348598]

Naïve Bayes was selected as a better algorithm over Decision Tree because it gave higher values of precision and recall.

Validation and Evaluation

Validation is used to assess how the results of an analysis will generalize to an independent dataset. It can also help checking if over-fitting happens. A common mistake happens when splitting the training and testing sets improperly. Specifically, in our case, since we only have limited number of valid data points (144), the possibility of getting a skewed training or testing dataset is very high. Therefore, I applied a 3-fold cross validation method to evaluate the performance. The results are as follows:

accuracy is 0.917
precision is 0.600
recall is 0.600
f1 is 0.600

f2 is 0.600

These results are higher than what I got using the tester code because the tester code is using a different approach of stratified shuffle split cross validation.

The precision is the possibility that a person who is identified as a POI is actually a true POI. The precision of 0.506 means that this identifier would result in 49.4% of the positive flags being false alarms. Recall measures the possibility of, given that there's a POI in the test set, this identifier would flag him or her as a POI. Therefore, 39.5% of the time this identifier would recognize that person as a real POI.

Discussion

The current identifier is not a convincing and reliable tool to identify POIs in the Enron data, so it still needs deeper optimization. Only if both precision and recall are close to 1.0 can we safely demonstrate that it is a good identifier.

One possible reason leading to low precision and recall is that the training dataset is too small. Only 18 POIs are included in the dataset, while there were 35 POIs in total. It would be very helpful to have all the POIs in the dataset.

Another way to improve the performance of the identifier is by making fully use of the email content. Finding some "key words" in POIs' emails and adding them into the features may greatly enhance the performance of the analysis.

I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.