Free-Response and Report:

Ashish Sahu

Summary

In 2000, Enron was one of the largest companies in the United States. By 2002 , it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, there was a significant amount of typically confidential information entered into public record, including tens of thousands of emails and detailed financial data for top executives.

This project will attempt to identify the employees that were directly involved in the fraud, also known as person of interest (POI). A person of interest (POI) is someone who was indicted for fraud, settled with the government, or testified in exchange for immunity.

The dataset contains email and financial information of 146 employees. Among the information is a label that classifies whether an employee is a POI. These features and labels can train a supervised machine-learning algorithm to identify a POI. This report documents the machine learning techniques used in building a POI identifier.

DataSet observations:

- *total number of information points/employee: 146*
- *For each employee 21 feature is available (poi label + 14 financial + 6 email).*
- *There are 18 POI and 128 non-POI.*
- *There are 3066 data points, which also contains 1358 missing values.*

Looking carefully I found few datapoint/employee contained outliers and had no information, so I removed them-

- `Total` : Just total information of dataset
- `LOCKHART EUGENE E` : No information : All feature are 'NaN'.
- `THE TRAVEL AGENCY IN THE PARK` : Not a person

The cumulative value of the financial data is in the "TOTAL" row. This row doesn't refer to an employee and all of its values are outliers so it was removed from the dataset. 'WHALEY DAVID A', 'WROBEL BRUCE' and 'GRAMM WENDY L' also contained missing values for most of the features, but for now I decided to keep them.

Features

In order to find the most effective features for classification, feature selection using "Decision Tree" was deployed to rank the features. Selection features was half manual iterative process. First I put all the possible features into features_list and then started deleting them one by one using score value and human intuition.

I picked 10 features which are:-

[salary, bonus, fraction_from_poi_email, fraction_to_poi_email, deferral_payments, total_payments, loan_advances, restricted_stock_deferred, deferred_income, total_stock_value]

Accuracy for this feature set is around 0.8. Approximate feature ranking:

feature	**scores**
salary	0.211
bonus	0.146
fraction_from_poi_email	0.120
fraction_to_poi_email	0.118
deferral_payments	0.095
total_payments	0.087
loan_advances	0.074
restricted_stock_deferred	0.053
deferred_income	0.053
total_stock_value	0.037

But with these features my precision and recall were too low (less than 0.3) so I had to change my strategy and manually pick features which gave me precision and recall values over 0.3. In this dataset I cannot use accuracy for evaluating my algorithm because there a few POI’s in dataset and the best evaluator are precision and recall.

Finally I picked the following features:-

[“fraction_from_poi_email”, “fraction_to_poi_email”, “shared_receipt_with_poi”]

Two new features were created and tested for this project. These were:

- the fraction of all emails to a person that were sent from a person of interest.
- the fraction of all emails that a person sent that were addressed to persons of interest.

I plotted a scatterplot to confirm email between POIs and non-POIs, i.e. there is no POI less than 0.2 in “y” axis. (there is stronger connection between POI’s via email that between POI’s and non-POI’s.)

Algorithm

Since Naive Bayes performs well text based classificaation, Firstly I tried *Naive Bayes accuracy was lower than with Decision Tree Algorithm* (0.83 and 0.9 respectively). Possibly cause the feature set I used does not suit the distributional and interactive assumptions of Naive Bayes well.

I selected Decision Tree Algorithm for the POI identifier, which gave me *DT accuracy before tuning parameters = 0.9*.

No feature scaling was deployed, as it’s not necessary when

using a decision tree. After selecting features and algorithm I manually tuned parameter min_samples_split. Here we hace class imbalance problem, so to good measure should be precision and recall, not accuracy.

min_samples_split	precision	recall
2	0.67	0.8
3	0.57	0.8
4	0.57	0.8
5	0.8	0.8
6	0.8	0.8
7	0.67	0.8

The best value for min_samples_split is 5 and 6.

Analysis Validation and Performance

What is validation and why it is important? The validation process ensures that results from the ML algorithm can be trusted, which is performed by splitting data into train and test sets. The testing data set is a separate portion of the same data set from which the training set is derived. The main purpose of using the testing data set is to test the generalization ability of a trained model.

Several performance metrics are:

1. Accuracy – proportion of test data correctly classified
2. Recall – proportion of true positives out of true positives and false negatives
3. Precision – proportion of true positives out of true positives and false positives
4. F1 - score – weighted average of the recall and precision.

This process was validated using 3-fold cross-validation, precision and recall scores. First I used accuracy to evaluate my algorithm. It was a mistake because in this case we have a class imbalance problem. the number of POIs is small compared to the total number of examples in the dataset. So I had to use precision and recall for these activities instead. I was able to reach average value of precision = 0.68, recall = 0.8.

Explaination of precision and recall score in context to the project

Precision

Precision is the probability of correctly identifying a data point as positive when the data point is identified as positive. Mathematically, precision is equal to the number of true positives divided by the number of positives (true and false). In our case, the precision rate represents the rate of a person being a POI actually when a person is identified as a POI by the algorithm, this measures how certain we are that the person really is a POI.

My final algorithm’s precision of 0.68.

Recall

Recall rate is the probability of correctly identifying a data point as true when in fact the data point is in fact true. Mathematically, recall is equal to the number of true positives divided by the sum of true positives and false negatives. In our case, the recall rate represents the rate of correctly identifying a POI when a person is in reality a POI.

My final algorithm’s recall value is 0.8.

Discussion and Conclusions

The fact that this is 0.68 means that using this identifier to flag POI’s would result in 32% of the positive flags being false alarms. Recall measures how likely it is that identifier will flag a POI in the test set. 80% of the time it would catch that person, and 20% of the time it wouldn’t.

These numbers are quite good but we still can improve the strategy. One of the possible paths to improvement is digging in to the emails data moreor or by digging into the text of each individual’s messages.

Reference

- k-Fold Cross Validation
- Scikit-Learn Data Preprocessing