Identifying Fraud from Enron Emails and Financial Data

Introduction

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 the public record, including tens of thousands of emails and detailed financial data for to executives.

We will be using machine learning algorithms from scikit-learn to detect the POI (Person of Interest). We will be using labeled financial and the e-mail communication data for training our algorithms.

1. Dataset and Goals

The goal of this project was to utilize the financial and email data from Enron to build a predictive, analytic model that could identify whether an individual could be considered a "person of interest" (POI).

Our dataset consists of 146 data points. Each data point has 21 features. From all the 146 data points, only 18 are labeled as Person of Interest.

From analyzing data in file “enron61702insiderpay.pdf” as well as from visual exploration of raw data, we find that there are two records which need to be removed.

a. THE TRAVEL AGENCY IN THE PARK: As we are trying to model the POI. This record seems like a company name, which is not an individual.
b. LOCKHART EUGENE E: All columns for this record are empty. So it is not a useful data point.

We also drew a scatter plot of salary vs bonus, which gave us outlier

c. TOTAL: This is just spreadsheet total of all. It doesn’t represent any individual.

2. Feature Selection and Scaling

We intuitively selected all the features which made some common sense to be used. We also added two new features “fraction_from_poi”, “fraction_from_poi” which represented the

fraction of communication with the POI. We also accumulated all the income related features like 'salary','total_stock_value','exercised_stock_options','bonus' into another new parameter named “wealth”.

Then we used scikit-learn's SelectKBest module to find out the 10 most relevant features. Below table has the score and the Non NaN number of recrods for each feature. It is surprising to note that even bonus has 81 non-NaN records, still it made to top 3 most relevant features.

Featture	Score	Non-NaN
exercised_stock_options	24.82	101
total_stock_value	24.18	125
bonus	20.79	81
salary	18.29	94
fraction_to_poi	16.41	143
wealth	15.37	143
deferred_income	11.46	48
long_term_incentive	9.92	65
restricted_stock	9.21	109
total_payments	8.77	123

It is interesting to know that two of our newly added features “wealth” and “fraction_to_poi” made to the top 10 of the most relevant features with really high score.

Before feeding the data to our classifiers, we also used MinMaxScaler to scale data evenly on same scale to get fair weightage.

3. Algorithm Selection

I used the following three classification algorithms

a. GaussianNB
b. DecisionTree
c. AdaBoost

Below is summary of findings.

GaussianNB:
Precision: 0.358773809524
Recall: 0.3035

DecisionTree:
Precision: 0.212785714286
Recall: 0.190880952381

criterion='entropy',
max_depth=10,
max_leaf_nodes=10,
min_samples_leaf=1,
min_samples_split=10,

AdaBoost:
Precision: 0.357928571429
Recall: 0.176404761905
algorithm='SAMME',
learning_rate=0.5,
n_estimators=40,

We can see that GaussianNB and AdaBoost have almost same precision but AdaBoost has very low recall value. We are more interested in recall value because a POI can be later verified aloes. So GaussianNB is algorithm of our choice.

4. Algorithm Optimization

We always have possibility of overfitting an algorithm or using wrong number of clusters. We used scikit-learn GridSearchCV during the algorithm selection process to get the optimum parameters. Please see summary results in section 3.

5. Validation

Validation allows us to assess how well the chosen algorithm performs on datasets other than training datasets. The biggest mistake we can make is over-fitting, where the trained model performs very well on the training dataset, but is worse on the cross-validation and test datasets.

As we analyzed the allocation across classes, we see that the data is unbalanced. Given the imbalance in the dataset between POIs and non-POIs, accuracy would not have been an appropriate evaluation metric. We therefore used precision (proportion of POIs who have successfully been identified) and recall (the proportion of individuals identified as POIs, who actually are POIs) instead. In GaussianNB case, we have highest recall value

Recall: 0.3035.

As we know our data is unbalanced, this means we should use cross-validation method like Stratified Shuffle Split, since it makes sure the ratio of POI and non-POI is the same during training and testing. Using StratifiedShuffleSplit gave even better precision and accuracy for GaussianNB

Precision: 0.39195 **Recall**: 0.33100

Although sometimes recall value comes at lower precision value, but in our case we can make this tradeoff because we can validate the POI from other means. Simply, 'flagging' individuals so that further investigation could be done, is more important than innocent individuals excluded. A high recall value ensures that truly POIs were identified and would be investigated more thoroughly.

6. Conclusion

As we already know that more data is always better for training. In our case, we had only 18 POIs from a small dataset of 146 records. We also dropped some records which were not individuals. Then we are also splitting the records for testing. This might result in even less POIs in the training data.

I think we could do some more data digging about the people indicted as well as people who were not officially indicted because of the closed door deals. Then we can also include more POIs from other firms. We can probably also include features like credit score, age, gender and total experience to see if we could improve more. I also think that duration of a person with Enron could be a big factor too. Longer the person worked at Enron, the more he might know about ins and outs and more confident he would feel to cheat.