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.

Utilizing the classifiers and techniques taught in Into to Machine Learning Class, I built a classifier to detect if a person is culpable or not.

Short Questions

> Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those?

The features in the data fall into three major types, namely financial features, email features and POI labels.

- financial features: salary, deferral_payments, total_payments, loan_advances, bonus, restricted_stock_deferred, deferred_income, total_stock_value, expenses, exercised_stock_options, other, long_term_incentive, restricted_stock, director_fees
- email features: to_messages, email_address, from_poi_to_this_person, from_messages, from_this_person_to_poi, shared_receipt_with_poi
- poi label(A total of 18 entries was labelled as POI)

The goal of this project was simply to leverage the features above in order to mark an individual as a person of interest.

In order to detect outlier and explore the data, I first built a code to convert the dictionay to csv file and is available with the submission. Later, I explored all the values to find out which features had a lot of values and which did not. Immediately looking at the csv, I noticed that the number of data is only 146 and so individually scanning each name - I found the following data points redundant

- Total: Does not convey information pertaining to any individual, hence marked as outlier.
- THE TRAVEL AGENCY IN THE PARK: It does not represent any individual and hence was removed.
- LOCKHART EUGENE E: This record contained no useful data.

After cleaning the data only 143 records remained.

> What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that doesn't come ready-made in the dataset–explain what feature you tried to make, and the rationale behind it. If you used an algorithm like a decision tree, please also give the feature importances of the features that you use.

To pick best features, I used the select k best features of scikit learn. After trying different K values, I decided to go with K value as 8 as it gave the best performance. Folling is the list of top 8 features I selected. The K-best approach is an automated univariate feature selection algorithm, and in using it.

```
'exercised_stock_options' : 24.81
'total_stock_value' : 24.18
'bonus' : 20.79
'salary' - 18.28
'deferred_income' : 11.45
'long_term_incentive' : 9.92
'restricted_stock' : 9.21
'total_payments' - 8.77
```

These features did not include any email features. Therefore I decided to go with my own created feature i.e email interaction

In order to include one more features that would represent an aggregate calue for both financial, I decided to go ahead with the two features above

- email_interaction_ratio: which was a ratio of the total number of emails to and from a POI to the total emails sent or received.

- financial_aggregate: This was the combined sum of exercised_stock_options, salary, and total_stock_value. This captured both liquid and semi solid wealth an individual has.

After testing the k value for financial aggregate I got the following: 15.971747347749965 Thus it is one of the significant features contributing to precision and recall.

Email interaction however did not crop up in either the top 10 or top 8 features. However, on trials with the algorithm, there was slight increase in both gaussian NB and Logistic Regression

I scaled all features using a min-max scaler. This ensures features are evenly balanced and overcomes the disparity due to the units of financial and email features. For classifiers such as Gaussian NB and rf there was no improvement noted post using feature scalling. However it played a pivotal role in Logistic and K-means classifier which forms the central theme for the report.

I used a total of 12 features for Logistic Regression, 8 for K-means clustering.

> What algorithm did you end up using? What other one(s) did you try?

After having performed various ml related projects during my undergraduate studies, a two class problem usually is best for logistic regression and K-means clustering. K-means clustering with PCA and mahalanobis distance provides a very fortified technique for two class detection. However, I went with logistic regression as my final algorithm.

I tried several algorithms, with a K-means clustering algorithm performing reasonably sufficient and gaussian NB performed event better than expected. I also tested a support vector machine, a random forest classifier, and stochastic gradient descent. The best reults I got were from logistic regressor and Gaussian NB.

I looked at famous algorithms for two classifier problems and NB was one of the top ones.

following were the parameters I tuned: - Logistic regression: C (inverse regularization parameter), tol (tolerance), and class_weight (over/undersampling) - K-means clustering: tol

I will explain why parameter tuning is important: Simply put most ml algorithms have a learning rate that is used to fit the dataset. Using such parameter we can either learn fat and properly or never get to the answer in some cases of gradient descent. Parameter tuning is used for this puprose. It defines the learning rate, tolerance to outliers and other parameters that are specific to algorithms used.

K-means clustering was initialized with K (n_clusters) of 2 to represent POI and non-POI clusters. It performed well with the 11 set of features.

Auto-weighting in the case of logistic regression caused a dip in precision and hence I decided to use evenly balanced features.

The other algorithms were tuned experimentally, with unremarkable improvement.

performance before and after adding new features Usual Features(feature selection was done till the best precision and recall values were obtained):

Classifier	Precision	Recall	Features
Logistic Regression	0.367	0.224	12
K-means Clustering, K=2	0.427	0.307	8

New Features:

Classifier	Precision	Recall	Features
Logistic Regression	0.422	0.281	12
K-means Clustering, K=2	0.343	0.346	8

There is a high increase in precision and recall in Logistic Regression and a good rise in recall in K-means. Hence, using the new features is justifiable.

> What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation is performed to ensure that a machine learning algorithm generalizes well. The classic problem that can occur is over-fitting. This happens when we overfit the training data and perform really well in it due to which there is a considerable dip in the performance in the other two datsets(cross validation and testing dataset).

I validated my result using two techniques - bootstrapping (cleaner.py) - k fold method (mytester.py)

> Give at least 2 evaluation metrics, and your average performance for each of them. Explain an interpretation of your metrics that says something human-understandable about your algorithm's performance.

I am using precision and recall as the evaluation metric. - Precision: The ratio of true positives to the records that are actually POIs. This describes the occurrence of false alarms - Recall: captures the ratio of true positives to the records flagged as POIs, which describes sensitivity.

I did not choose accuracy because with such a small dataset using it would mean that a simple heurestic with all data marked false would give an accuracy of more than 85%

Validation 1 randomised sampling(n=1000)

Classifier	Precision	Recall	Features
Logistic Regression	0.422	0.281	12
K-means Clustering, K=2	0.343	0.346	8

Validation 2 (Stratified K-folds, K=3) (highest value taken after many runs)

Classifier	Precision	Recall	Features
Logistic Regression	0.583	0.333	12
K-means Clustering, K=2	0.431	0.500	8

Both algorithms do well inspite of the dataset being noisy.

Let us understand what the values are actually speaking to us. Now, in terms of precision and recall you would want a high recall in such a case. Simply, because you want to be suspecting people. Having a high precision would mean that we are looking for too strict of a conditions to flag an individual. Hence, recall plays a high role in such a case.

Conclusion

The dataset was sparse and most algorithms will perform well only if given a decent number of data to learn. Had there been a large dataset random forest would also work very well. However, I am more interested in what a classic anomaly detection system would do in such a case. A simple combination of - K-means - PCA - Mahalanobis distance measure would result in a very robust outlier detection algorithm and probably would have both a high recall and precision value.

Resources and References

- Introduction to Machine Learning (Udacity)
- Machine Learning (Stanford/Coursera)
- scikit-learn Documentation