Enron Person of Interest Identifier

Introduction

The Enron Fraud was one of the largest Corporate Fraud cases in U.S. History. It happening after the advent of the internet and the detail with which the company was investigated afterwards, allows us to look at more of the everyday data that the company produced. In this project, the email and financial data for 146 was used in an attempt to use a machine learning algorithm to discern persons of interest in the case. A person of interest, in this case, is a person who was either indicted for fraud, settled with the government, or testified in exchange for immunity.

The Enron Data

The Enron Data included a few outliers. One of which was found to be a spreadsheet “Total” Line which was removed. The other outliers were found to be people, who made a lot of money from the fraud, and were not removed.

Feature processing

As I knew that I would be trying PCA I included fourteen of the twenty-one features in the data set. These features were: 'poi','salary', 'exercised_stock_options', 'deferral_payments', 'total_payments', 'other', 'bonus', 'restricted_stock', 'deferred_income', 'long_term_incentive','restricted_stock_deferred', 'total_stock_value', 'expenses', 'shared_receipt_with_poi'. I also added 2 features which I named 'percent_from_poi', 'percent_to_poi', which took the number of emails to/from a person of interest and divided it by the total number of emails that they sent/received. The reason for adding that particular set of features was that I suspected that persons of interest would be more likely to send emails to each other than people who were not persons of interest.

I implemented a min/max scaler to scale my feature values, as K-means clustering is sensitive to values which vary widely across the axises.

I intended to include a TFIDF feature as well, but ran into multiple problems using the resulting object, and so ultimately it was left out.

Algorithm Selection and Tuning:

I decided to use K-means clustering, with PCA determining the 10 best components. I decided to used K-Means after playing around with both Naive Bayes and Support Vector Machines, neither of which gave me results that I was happy with.

After settling on the Algorithm, and deciding a number of principle components to use. I manually tuned the algorithm's n_cluster's variable in an attempt to balance the potential for over-fitting, while still maintaining performance metric values that made me happy.

Below is a chart showing the values obtained for each of the values that I set for n_clusters.

n_clusters	Precision	Recall	Accuracy
2	0.259	0.076	0.84
3	0.509	0.444	0.82
4	0.756	0.790	0.845
6	0.833	0.857	0.865
8	0.920	0.921	0.909
10	0.945	0.951	0.930
20	0.978	0.982	0.967

In the end I settled on using 8 cluster centres as I feel that precision and recall values much over 0.92 might be over-fit. Even though the validation methodology (see below) should help to ameliorate that risk.

Validation and Performance

The validation method built in to the test method was used to validate my results. This used a stratified shuffle split cross validation method. This randomly splits and shuffles the data into training and test sets. In this case it was split one-thousand times, and allowed each run of the algorithm to make 15000 predictions, in spite of having a data set with only 145 individuals and 18 persons of interest.

Discussion and Conclusions

First I will define the performance metrics used. Precision, in this case, means the likelihood that the prediction of a person of interest is actually a person of interest. With this precision result 8% of positive results would be false positives. Likewise, recall is the likelihood that in a test set with a person of interest in it, that person would be identified as a person of interest. With the recall score I got with my final model 92.1% of the time a person of interest would be identified given that there was one in the test set.

The numbers that I got are very good. Good enough that I worry that I may be over-fitting the algorithm. Subsequent testing of higher values for the number of cluster centres showed that I could get higher values with more cluster centres.

One of the challenges for me was figuring out how to utilize the TFIDF feature that I tried. Including it among the other features threw an error in numpy that I could not figure out how to fix. I wound up taking the text analysis out of the model in the end. Though, I will probably continue to play with it, to figure out how to make it work in the future.

The only way that I can think of to make the algorithm more accurate (other than including more cluster centres) is to provide the algorithm with more data. For some reason only 18 of the 35 identified Persons of Interest were included in the data set. With more data to draw from you could expect more accuracy from a less finely tuned algorithm, thus risking over-fitting a lot less.