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 using Select K Best and PCA I included all of the financial 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'. Leaving out the email related ones as they would be used later. 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 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 chose K Nearest Neighbours after having tried and SVM and Naive Bayes. The Naive Bays classifier after minimal tuning gave me a recall number near 0.89 but a precision number around 0.16. The SVM again with minimal amounts of tuning provided a recall number of 1 but a precision of 0.13

After settling on the Algorithm, I had several values that I could tune to derive better results from my algorithm. This is an iterative process by which you change the value of a parameter of the algorithm and look at how the results change, intending to find the best setting. I had to fine tune my Select K Best algorithm, the Principal Component Analysis algorithm and the K Nearest Neighbours algorithm.

Below are a series of tables showing my tuning of each algorithm's parameters.

Select K Best:

Select K Best is an algorithm which selects the most powerful features to use in a future algorithm. The parameter to tune is the number of Features that the algorithm returns. Find below the results of my tuning the k parameter.

K features returned	Precision	Recall
1	0.27	0.160
2	0.335	0.167
3	0.482	0.238
4	0.595	0.321
5	0.624	0.339
6	0.576	0.272

Past 6 the precision and recall numbers continued to go down. So I kept the value 5 for k.

Primary Component AnalysisL

PCA is an algorithm which takes the features given to it and groups them for best effect trying to maximize the effectiveness of the groupings. Below is my tuning of the number of components returned

n_components	Precision	Recall
1	0.350	0.186
2	0.502	0.218
3	0.635	0.337
4	0.619	0.332

As above past 4 the precision and recall numbers continued to fall, so I chose n_components = 3

K Nearest Neighbours:

K Nearest Neighbours is a classification aglorithm which uses the class of it's nearest neighbours to determine a datapoint's classification. It has a couple of parameters to tune. N_neighbours which is the number of neighbours to use in the classification, and the p value, which tells the algorithm which metric to use.

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

n_neighbours	**Precision**	**Recall**
1	0.448	0.365
2	0.448	0.356
3	0.578	0.371
4	0.586	0.326
5	0.635	0.337
6	0.714	0.268

I selected 3 for the n_neighbours parameter because it gave me the most balanced precision and recall numbers.

Lastly is the table for p-values

P value	**Precision**	**Recall**
1	0.538	0.313
2	0.578	0.371

As you can see changing the metric used, in this case, negatively impacted precision and recall.

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. In more general terms validation is the process by which you split you data into training and test sets. This allows you to estimate your algorithm's performance, and helps to prevent over-fitting. A classic mistake in cross validation is to use the same data to train the algorithm that you use to test the algorithm. This results in over-fitting/

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 43% 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 31.3% 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 not particularly good. However, they are better than you would expect from a random guess. I would not recommend using this particular algorithm to do this sort of investigation.

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.