Project 5: Predicting Fraud at Enron using Machine Learning

Mike Cassell

Mike.Cassell@rogers.com

Objectives

The goal of the project was to utilize financial and e-mail data from the collapse of Enron which was gathered for the course to determine the best way to predict which employees were most likely involved in the fraud perpetrated there. The dataset was made up of publicly filed financial information, information gathered from the news and the Enron e-mail corpus that has subsequently been published.

The dataset consists primarily of the financial data from 146 employees and stakeholders across 20 different metrics (features). The features are primarily numeric, with the exceptions of the e-mail addresses and names (both text) and the provided person of interest (POI) identifier which was a Boolean. There was also a corpus of e-mails released in the subsequent investigation that provides individual e-mails from 150 (although in some cases not the same as the above) employees.

There are a total of 18 persons of interest in the sample which is a small sample. The dataset is fairly heavy in non-persons of interest which must be kept in mind when implementing validation scoring (since there are more non-POIs than POIs, the model could simply flag everyone as not a POI and have a better than 50% accuracy.)

There were a number of possible outliers in the dataset that were scrubbed including a Totals row and a negative valued compensation figure that were not deferred. While there were several other questionable figures, without access to the originals the majority were left in the analysis at this time. Finally there were a large number of cases when individuals had null values across the numeric features that had to be included since the size of the sample was small.

Feature Creation and Selection

Creation

My analysis first began using Pandas in an IPython notebook to visually hunt for features that appeared to be related to the POI indicator. None were readily apparent but I did gain a better insight into the underlying data and the values within.

I created a total of four features, three of which corresponded to an increase in accuracy in identifying persons of interest during early teting. The first two were suggested within the course and in the example sheet and illustrate the frequency of senders mailing POI vs. others and the reverse, the frequency of receiving mails from POI vs. Non-suspects. This raises a concern that we must have identified persons of interest prior to creating the metric and may be of limited use in a production environment.

I also created a feature to measure the proportion of a persons vested or exercised stock options vs. their total with the thinking that a person committing fraud may be more likely to access their funds as quickly as possible. This metric was discarded as it did not add any value to the precision or recall scores.

Lastly I approached parsing a larger portion of the corpus to see if using ML, we could identify words that most frequently corresponded to persons of interest. I was able to parse the e-mails from the senders in the dataset with e-mail addresses (I used all of the POIs plus 10 times as many non POIs to ensure the e-mail sample was roughly proportional to the main financial dataset.) The resulting dataset was processed with a TLIDF function and then processed with a Decision Tree Classifier which provided a model that identified key words which are associated more frequently with a person of interest.

Interestingly the words that initially were most closely associated with fraud tended to be numbers or dates which potentially pointing to events that the senders were involved with that either directly or indirectly involved committing fraud. Due to the very high level of accuracy in the resulting test analysis, these potentially self-identifying dates and a number of words were excluded from the model (typos and names which may also have identified specific e-mails or chains).

I also analyzed for phrases (ngrams of up to 4 words in length) but there were none identified with any predictive power and due to the much greater computational load, this was removed from the final algorithm.

The final list of keywords and their corresponding scores were:

Keyword	Score
origin	0.262
let	0.179
know	0.155
like	0.151
2001	0.105
meet	0.081
need	0.067

The resulting predictive power remained surprisingly high which could point to the keywords still being self-identifying in some way. Given the list of retained words appears to be made up of only common words, short of additional context behind those e-mails (and a lot of reading), I am accepting that they have genuine predictive power in this case for the purpose of this report.

An alternative approach of excluding a certain number of the POIs from the training set entirely was looked at but due to the very small sample size (14 pois had e-mail data), the idea was discarded for the time being since it would be difficult to ensure a well represented sample. The decision tree did not have the users as a labels, only whether the e-mail was from a person of interest or not which ensured the model was not trying to identify a specific person.

Introducing the TLIDF data from the email corpus increased prediction and recall by a huge margin (from 30% to over 92%).

Selection

After manually searching for appropriate features for a while, I took an automated approach and used the SelectKBest approach. Manual iterative testing proved the text analysis component alone drove the majority of the predicting power with very small increases from adding a second. Additional features did not add any value and were therefore removed.

Scores from various K-Best Tests:

K	Accuracy	Precision	Recall	F1	F2
1	0.96	0.95	0.92	0.9	0.93
2	0.967	0.942	0.935	0.9	0.94
3	0.967	0.942	0.935	0.9	0.94

The final features selected were all e-mail related with the Flagged Ratio (indicating the ratio of e-mails scored as fraud related, score 260) and the From Ratio (ratio of e-mails received from a POI, score 15). The precision of the model actual fell marginally but the second feature did add to the recall and so was seen as beneficial.

It could be argued that a simple value could be found where if a sender had a ratio of identified e-mails over that value, they would be considered persons of interest given the hugely influential score of that value.

Algorithm Selection

Originally a number of algorithms were evaluated and the most effective was a Decision Tree Classifier. The preliminary performance numbers for some of the iterations are available at the end of this report. It was surprising that when I attempted to boost the performance with AdaBoost, the performance was actually slightly worse (although not substantially).

Algorithm Tuning

The tuning of an algorithm can lead to marked improvements or drastic error increases if done incorrectly since it can change the underlying setup of the algorithm. This is also the most likely stage to introduce overfitting of a model since it can be made too specific (e.g. a decision tree that is so specifically tuned that it knows individual cases for every branch).

To tune my decision tree, I made use of a grid search configured for the maximum features and leaf depth, the minimum samples per split and samples per leaf and the criterion engine. Since the text based parameter was such a strong predictor, the grid search indicated that there wasn't a need for a large number of splits in the data.

Once the flagged text indicator was added from the TLIDF, the model became somewhat overfit with a Recall score of 1 which is likely too high to be the result of a good model. Adjusting the min_leaf_size and the maximum number of splits corrected for this to ensure that the tree was not creating individual leafs for single persons etc.

The final feature importance numbers from the decision tree were surprising, while there is a small increase in adding the second feature in terms of recall, the importance was scored as 1.0 for the FlaggedRatio feature. This finding emphasized the earlier suspicion that the DCT in the main project could be foregone in lieu of a simple if statement. While the second feature was weighted zero, it did have a marginal impact so it is suspected the 1.0 was rounded.

The opportunity identified by this would be to try and find alternate features that could be injected that did not depend on the e-mail data being available (there had been 4 users identified as persons of interest who had no e-mail data) but upon reviewing their financial information, there were no values for several in any feature which implies there would be little to be gained from further modifying the algorithm since they seem to be lacking data from multiple sources. In this case it appears using an ML algorithm based on another ML algorithm may be over-complicating the project (although the tuning and feature selection validated the original measure).

To try and avoid the over-reliance on the text feature I also tried implementing an SVM, a KNN and a Gaussian algorithm but in all cases, similar results were observed with the precision remaining very high (as high as 1 in many cases). The recall did fall quickly as additional features were added in some algorithms but due to the precision remaining very high, it is suspected that this is indicative of the additional measures detracting from the model instead of helping correct for the remaining missed POIs.

Validation

Validation is the process of reserving some of the data away from the training of the model to ensure that once trained, the model's accuracy can be tested against novel cases. This is useful in preventing the overfitting of a particular model since if the algorithm is over-tuned to the training data, the testing scores on new data should show a marked change in performance. A common potential issue in building ML algorithms is forgetting to exclude a portion as testing data which makes it impossible to test for overfitting or other issues.

Originally I had implemented a KFoldsValidation strategy but encountered problems due to the skewed nature of the data in the relative number of POIs present (there are many more non-POI samples which resulted in very inconsistent folds). I instead ended up using the tester module which implements stratified shuffle split validation. This addresses the issue of not having a large number of persons of interest by ensuring that each fold of the process has proportionally the same number of POIs as non-POI data points.

The tester module is already well engineered and performs 1000 folds by default which I believe to be a very thorough test of the model and in the interest of not reinventing the wheel, I made use of it after ensuring I had a thorough understanding of its methods

The primary metrics that I focused on were the accuracy, precision and recall. Accuracy is the number of correct classifications divided by the total number of classifications completed. This takes into account both the number of correct positive and negative assignments. The final accuracy of my model was 96.7%.

Precision can be the thought of as the accuracy of the positive assertions of the model. The mathematical formula is the number of the True positives the model generated divided by the total number of positive responses from the model. The metric is important since it gives an idea as to how often the model is correct when it indicates someone is a person of interest in this case. The final precision I was able to achieve was 94.2% which indicates that while the model is correct in identifying a person of interest more often than not.

Recall measures how effective the model is at identifying all of the true positives by comparing the number correctly identified to the total number in the population. In the case of my model, I was able to achieve 93.5% which indicates that the model is fairly accurate in this respect.

Conclusion

Overall the exercise appears to have met its goals in creating a model to predict which employees at Enron were related to the fraudulent activities (although in some ways, almost too well). Additional outside information about others involvement in the fraud or around the initial definition of who was a person of interest could be used to further validate the accuracy of the combined TLIDF/Decision Tree which initially scores the senders e-mails to determine where overfitting may be coming into play.

Initial Algorithm Testing Results

Algorithm	Accuracy:	Precision:	Recall:	F1:	F2:	Total predictions:	True positives:	False positives:	False negatives:	True negatives:
GaussianNB	0.6560	0.3050	0.6975	0.4244	0.5547	11000	1395	3179	605	5821
DecisionTreeClassifier	0.7612	0.3339	0.3150	0.3242	0.3186	11000	630	1257	1370	7743
SVC	Divide by zero error									
KNN	0.8203	0.5119	0.2480	0.3341	0.2765	11000	496	473	1504	8527
AdaBoost w DTC (SAMME)	0.7552	0.3179	0.3025	0.3100	0.3055	11000	605	1298	1395	7702
AdaBoost w DTC (SAMME.R)	0.7568	0.3204	0.3010	0.3104	0.3047	11000	602	1277	1398	7723

Additional Resources

I made use of the SKLearn website very frequently as well as the course website and the example code from github for the course.

I also went to the internet for:

Trying to get a better understanding of how GridSearch finds the ideal score:

http://stackoverflow.com/questions/24096146/how-is-scikit-learn-gridsearchcv-best-score-calculated

Extracting KBest features:

http://stackoverflow.com/questions/14133348/show-feature-names-after-feature-selection

GGPlot for Python:

https://github.com/yhat/ggplot

Practical Text Mining Example:

http://www.markhneedham.com/blog/2015/02/15/pythonscikit-learn-calculating-tfidf-on-how-i-met-your-mother-transcripts/

I'd also like to give mention to those who have reviewed the previous iterations of this report and code for the significant contributions they made to help me find and correct its shortcomings.