Investigating Eron Emails for Fraud

April 27, 2019

1 Investigating Enron Emails for Fraud

Machine Learning project - Josh Maine Enron was one of the largest companies in the US in 2000 and within two years it had sunk into backruptcy and caused a firestorm of legal investigations due to company wide corruption and fraud. A Federal investigation was conducted that resulted in a lot of parties admitting guilt, found guilty, or settled out of court. These events led to an overhaul in laws that govern business conduct, mainly the Sarbanes-Oxley Act but also an unprecidented amount of data was made public. This data includes thousands of intracompany emails and detailed financial information including the information for top executives.

This project is being done to leverage machine learning to answer some questions outlined below.

2 Questions

2.0.1 Question 1:

> 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 goal of this project is to use machine learning algorithms to sort through involved persons and build an means of identifying people of interest. The list that contains the persons of interest (POI) consists of those that were indicted, settled out of court, accepted a plea deal, or testified for immunity form prosecution. Using the POI list and the financial and email data the model can be built to compare trends and make the determination.

There are 146 records in the dataset, with 18 records identified as being poi, and the remaining 128 being classified as non-poi. Each person record has 21 features that may be used in finding trends and building the final model.

There is currently three likely outliers that will be removed form analysis. These outliers are a TOTAL record, which appears to be just a summation of the data features, 'THE TRAVEL AGENCY IN THE PARK' which doesnt appear to represent a person and is assumed to be an error, and lastly the record of Eugene Lockhart, as all the features had no values.

2.0.2 Question 2:

> 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 does not come ready-made in the dataset – explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values.

I calculated the feature scores using SelectKBest to find the 10 highest scoring features to use in the algorithm. After selectKBest returned results, min-max scalers were used on the features. Scaling was used after looking through the data and seeing a order of magnitude variances. The scaling allows for more even consideration of features.

The selected features are:

Financial Sourced: exercised_stock_options, total_stock_value, bonus, salary, deferred_income, long_term_incentive, restricted_stock, total_payments, loan_advances
Email Sourced: shared_receipt_with_poi

Looking at the features above, it seems that possible correspondence with a POI might be a trend or indicator. So I want to create two new features, *messaged_poi_ratio* and *messaged_by_poi_ratio*. These will hold the calculated ratio of how the total communications in regards to a person emailing a POI and how often a POI messages them, respectively. I suspect outside of possible assistants or email delegates that the higher the ratio the more likely the person might have some involvement in the fraud. Adding these features improved preformance of the selected top 10 resulting in the below results.

2.0.3 Question 3:

What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

I attempted two algorithms: Decision Tree, and Logistic Regression. I wanted to explore a familar one from the lesson and one I was unfamilar with, Logistic Regression.

I had better results with Logistic Regression after tuning. I assume that it was due to the categorical natural of POI or non-POI in comparsion to the decision tree results having multiple splits.

The results using all the original features:

Method	Precision	Recall	Accuracy
DT	0.33	0.4	0.842
LR	0.354	0.57	0.804

The results using the top 10:

Method	Precision	Recall	Accuracy
DT	0.356	0.44	0.849
LR	0.205	0.425	0.703

The results using the top 10 and custom made features:

Method	Precision	Recall	Accuracy
DT	0.546	0.541	0.892
LR	0.393	0.728	0.814

As shown in the numbers accuracy overall was higher for the decision tree model but recall and precision was at best around 50%. The addition of the custom features regarding commnication ratios improve our model compared to the results of just using the top 10. Using just the top 10 the performance of the Logistic Regression model dropped while the Decision Tree slightly improved compared to using all features. The results of using the top and the new features gave the best results for the Logisitc Regression while still only leading to a smaller improvement of the decision tree.

2.0.4 Question 4:

> What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well?
> How did you tune the parameters of your particular algorithm? What parameters did you tune?

Tuning a parameter is adjusting the algorithm in order to improve the behaviour using test data.Parameters affect the learning process and behaviour of the algorithm. Overtuning a parameter can cause a highly biased behaviour and over fitting. Failing to tune enough can lead to a swing to the other side of the bias-variance delima and lead to poor or unexpected preformance.

The tuning parameters were obtained using GridSearchCV due to the assist it adds in tuning for a novice in machine learning like myself. The parameters given and used are:

```
>{tol : 0.1, C = 10^9, class_weight: balanced }
in additon for the final parameters I added the below due to sample size mostly
{ criterion: entropy, min_samples_split: 20 }
```

2.0.5 Question 5:

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

Validation the process of checking that the model built behaves similarly on the test data as it did the training data. One pitfall in trying to match results is driving the model into a state of over-fitting the data points, making the model less flexible and accurate. This is usually easy to spot if your training set preforms expontentially better than your test data.

One of the most common techniques is using cross-validation. Validation was carried out with a defined evulation fuction. Using train_test_split from SKLearn to do a 30/70 test:traing split, for over 50 trials. Cross validation was carried out using best_estimator from GridSearchCV using a 1000 fold StratifiedShuffleSplit. Stratified results were used because the the size of the dataset is so small it allows consistent POI and non-POI ratios.

From the entire data set the POI ratio worked out to be 12.6%. From 1000 folds and a random state, the split used a test size of 15 with 2 POI also working out to roughly 12.6% This shows the splits improved the validation effort.

2.0.6 Question 6:

> 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.

The main metrics used in evaluating the proformance was precision, recall, and accuracy. Comparison of the results for the train test split and the results of the testing script as shown here

```
           train_test_split           tester.py
Recall:          0.79                  0.73
Precision:       0.54                  0.39
Accuracy:        0.89                  0.81
```

The stratified results from the test script was lower but I expected it as its more accurate over just splitting the data, keeping some modesty about possible inflated results.

The most important of the three metrics would have to be recall. Recall still being pretty high is a great indicator especially when coupled with a lower precision. In plain english this means that on average the ability to identify a POI is pretty high, recall, while the chance of incorrectly flagging false POIs is about 40%. In most legal and law enforcement matters its preferred to be interested in an innocent that can be absolved vs completely having a guilty party be missed and slip through.

Ultimately it means the model is amirable first forray into the field. The model would probably benefit from having more data points as more data points can help over come tuning issues in general. Datasets that are processed by machine learning are usually a lot larger but it was interesting using the data to get into the field and practices of building a model for this information seeking activity. As I mentioned before additonal informaton and categorization might be helpful based on job title as a start as most companies use lower tier employees for mass communications which may have skewed this data.