Stephen D. Wells
Udacity Nanodegree: Data Analyst
August 15, 2015

Identifying Fraud in Enron

Machine Learning Project

GOAL

In 2002, the Harvard's Ig Nobel Prize for the 'Most Creative Use of Imaginary Numbers' went to a company that two years prior had received it's sixth, 'America's Most Innovative Company' award from Fortune magazine[1]. In 2000, Enron was the seventh-largest company in America with over $100 Billion in revenue.[2] The next year as the result of fraudulent accounting practices, they were bankrupt.[3] During the Federal Energy Regulatory Commission's investigation, internal company emails were released to the general public.[4] Combining this dataset with the Securities and Exchange Commission financial data, along with Persons of Interest derived from newspaper articles we will answer the question, '**Can we identify Persons Of Interest (POI) solely from this dataset?**'

DATA EXPLORATION	
146	Total No. of Data Points
18/144	Allocation Across Class (POI/non-POI)
35	Total Gathered POIs
14.38%	NaN for Total Payments
0 of 18	POIs with NaN for Total Payments
21	Number of Features
95	Qualified Salary
111	Known Email Address

Features with Many Missing Values
Loan Advances
Director Fees
Restricted Stock Deferred

[1] https://en.wikipedia.org/wiki/Enron

[2] http://www.forbes.com/2002/01/15/0115enron.html

[3] https://en.wikipedia.org/wiki/Enron_scandal

[4] https://www.cs.cmu.edu/~./enron/

OUTLIER INVESTIGATION

Using machine learning algorithms we make predictions and calculate the accuracy of those predictions. The worst case of outliers we uncovered was due to a TOTAL in the financial statement which was read in as a line item. This was removed by hand. Four other outliers were left in as they were considered valid data points. A technique to programmatically remove outliers by discarding 10% of the points with the largest errors was introduced and used to help identify additional potential outliers, however it was determined that those points were valid. A careful reading of the field names revealed, 'THE TRAVEL AGENCY IN THE PARK' which, not being an individual was removed.

LEADERSHIP	
CEO	Jeffrey Skilling
Chairman	Kenneth Lay
CFO	Andrew Fastow

The financial dataset does not include all POIs hence some are missing.

Kenneth Lay took home the most at **$103,559,793** during this time but his conviction was vacated due to his untimely death before he could appeal.

OPTIMIZE FEATURE SELECTION/ENGINEERING

Feature Importance

with sample algorithm performance

- **.28** Salary
- **.19** Bonus
- **.30** Other
- **.12** Fraction of Deferred Income to Total Payments
- **.11** Milk (defined below)

These types of supervised ML algorithms require re-training at periodic intervals. The "best" algorithm for the Netflix Prize was presented with $1 Million though it wasn't used in production because of the engineering effort required for it's minimal gains.[5] The challenge in this case was to achieve recall and precision values of 0.3+. In an effort to simulate a real-world scenario, I opted to achieve this with a minimum of features and utilize as few resources as possible.

Salary, Bonus and Other (severance, consulting, relocation, tax advances, housing etc.) were chosen in combination with sklearn's univariate feature selection, **selectPercentile**, I painstakingly added and removed features by

[5] http://www.forbes.com/sites/ryanholiday/2012/04/16/what-the-failed-1m-netflix-prize-tells-us-about-business-advice/

hand until I had narrowed down only the most relevant factors. New scaled features were selected according to the following rules.

New Features

Though each iteration changes the impact of the new features, these two impact the final algorithm performance by about 10% each.

fraction of deferred income to total payments

This feature assumes that non-POIs would believe that the company would continue to grow and be in a better position to make those payments. POIs would know that the foundations were crumbling and want to be paid before it was unable.

milk (as in MILKing the company)

The value "milk" was scaled using the following ratio.

$$\frac{\text{(Expenses + Deferral Payments)}}{\text{1 + (Loan Advances + Long Term Incentive + Deferred Income)}}$$

The belief that POIs have an incentive to get as much out of the company as quickly as possible since they don't see a long-term future in the company drove the decision to create a label "milk" as in 'Milking the Company'. Expenses include consulting, reimbursements from the company and deferral payments are distributions from deferred compensation. The company must pay these now whereas loan advances, long term incentives and deferred income are future payments. If employees believe that the company won't have the money to pay them in the future, they will want to collect what they can now.

PICK AN ALGORITHM

The goal is to classify individuals as either a POI or non-POI. Since this is a classification problem, I was able to eliminate regression and clustering algorithms. Further to this, it's not a "Big Data" issue as I don't have more than 100,000 samples, so I didn't need to go with an SGD Classifier or kernel approximation. This left me testing the following five algorithms:

- Naive Bayes (GaussianNB)
- Support Vector Machines
- LinearSVC
- Ensemble: Random Forest
- Decision Trees

GaussianNB was simple to implement but even with excellent recall numbers, no amount of tuning afforded respectable precision results. Support Vector Machines took too long to run, however LinearSVC was much faster and both my precision and recall numbers looked more even but but didn't yet yield the results I was looking for. RandomForest resulted in higher precision than the recall number and it took a long time for my computer to learn. Decision Trees gave back very fast results and the default was very close to the results we were looking for.

Algorithm	Precision	Recall
GaussianNB	0.15	0.75
SVC	n/a	n/a
LinearSVC	0.20	0.30
RandomForestClassifier	0.45	0.24
tree	0.33	0.34

TUNE AN ALGORITHM

Algorithm tuning is vital to improving performance as the attributes for a given method help to refine how that algorithm is processed. After tuning the algorithms above (apart from SVC as it took too long to run), the DecisionTreeClassifier was chosen and tuned using the following parameters:

GridCV Trials of Tuning Settings on DecisionTreeClassifier

several parameters were tuned and included the following…

criterion gini, entropy

splitter best, random

max_features 1, 2, 3, 5, 9, 0.1, 0.2, 0.25, 0.5, 0.75, 0.8, 0.9, 0.99, auto, sqrt, log2, None

max_features 0.1, 0.2, 0.25, 0.5, 0.75, 0.8, 0.9, 0.95

class_weight None, auto

max_leaf_nodes None, 2, 3, 4, 5, 6, 7, 8, 9, 10

After starting with a grid search (GridCV), I manually tuned the parameters of my decision tree. Interestingly, I received worse results when adding *max_leaf_nodes* as a parameter to **GridSearchCV**( ). In this case, the defaults gave the best results, with the exception of changing the **splitter** attribute from *best* to *random*. The best algorithm-tune combination from these results were selected for the final analysis.

CLASSIFIER

A decision tree classifier was used in the final product. This decision is based on a balance between speed and accuracy. For instance, though the SVC algorithm with proper tuning may have ultimately given a better result, the algorithm was discarded as it hadn't completed in overnight tests while on the other hand GaussianNB and LinearSVC were both very fast, but didn't yield the required results. Other algorithms such as Random Forest were tested and removed on similar grounds

EVALUATION MATRICS

Recall that our goal was to achieve higher than 0.3 for both precision and recall. In this case our average precision value was 0.34. which translates to mean that 34% of the time we don't label a POI as a Non-POI. Recall looks at the other side of the equation and in plain English means that 35% of the time we correctly classify POIs.

VALIDATION STRATEGY

To validate our data we split it into two groups which we label as, training and testing. The algorithm is then used to train our data and validated using our testing data. A naive mistake would be to use our testing data to train, because even though it will generate great results in our simulated environment, those results won't translate to a real-world environment. Since we have a small dataset the decision was made to use a **StratifiedShuffleSplit** as part of the cross_validation package from sklearn. It will return stratified randomized folds (1,000 in this case) which are produced by preserving the percentage of samples for each class.

Precision and Recall Values		
	Precision	**Recall**
mean	**0.34**	**0.35**
median	0.34	0.35
max	0.37	0.38
min	0.32	0.33
first quartile	0.34	0.34
third quartile	0.35	0.36

The DecisionTreeClassifier was run 1,000 times where it recorded the precision and recall values each time. These statistics were generated using this data.

ALGORITHM PERFORMANCE

I created a script to run the algorithm with our tuning parameters 1,000 times and recorded the results into a csv file. From here I was able to build out the table on left which clearly shows that we are able to achieve precision and recall results above 0.3 for each iteration.

PERSONAL INSIGHTS

Enron's upper management didn't write their own emails. In fact, according to this dataset, Ken Lay never wrote a single email as he had a personal assistant write and send emails on his behalf. This means we shouldn't use word choice as part of our analysis as it would only be analyzing his assistants vocabulary. The metadata, from/to information is slightly better as we assume that management played a role in who received emails. Though limited information could be garnered from the emails under the assumption that non-POIs emailed each other more than they emailed POIs, it proved possible to manipulate the financial dataset with enough insight to make that analysis obsolete. This is why I dropped the email metadata from my analysis.

I did run across a video[6] the other day that claimed that you can tell who a director is at Enron by comparing the number of the emails they sent to the number of nodes. In other words, those that emailed people who didn't email each other were more likely to be a director in the company.

[6] https://www.youtube.com/watch?v=GBzoNgqF-gQ&t=38m17s