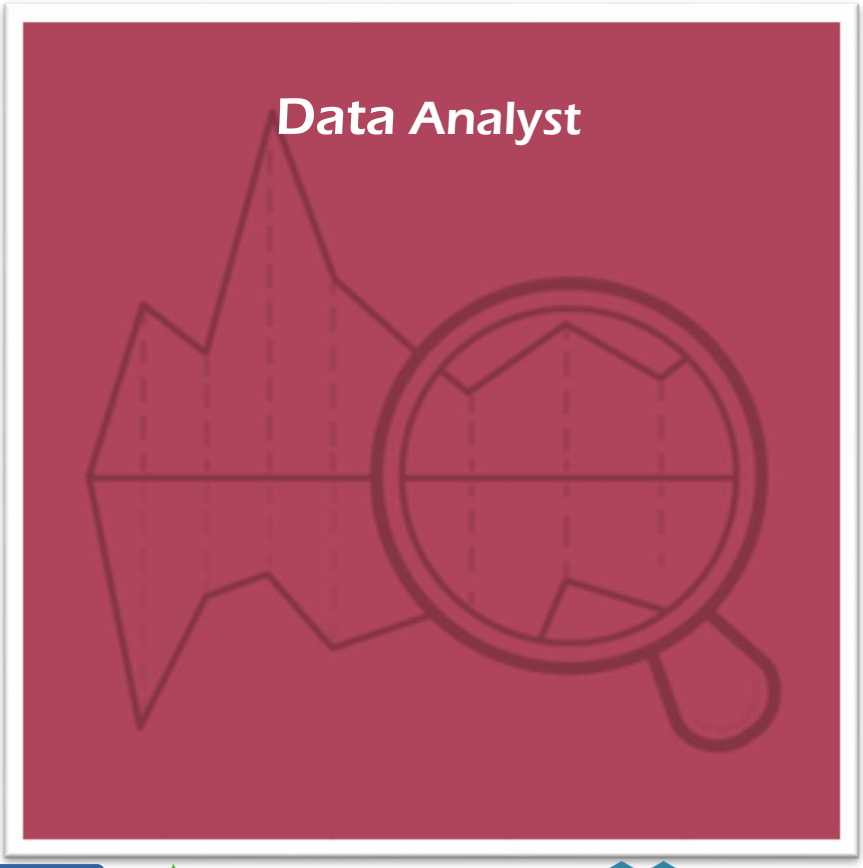


Data Analyst Nano Degree

P5 – IDENTIFYING FRAUD FROM ENRON EMAIL

Harshit Trivedi | Data Analyst Nano Degree | July 22, 2016

Background Information

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, there was a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for to executives.

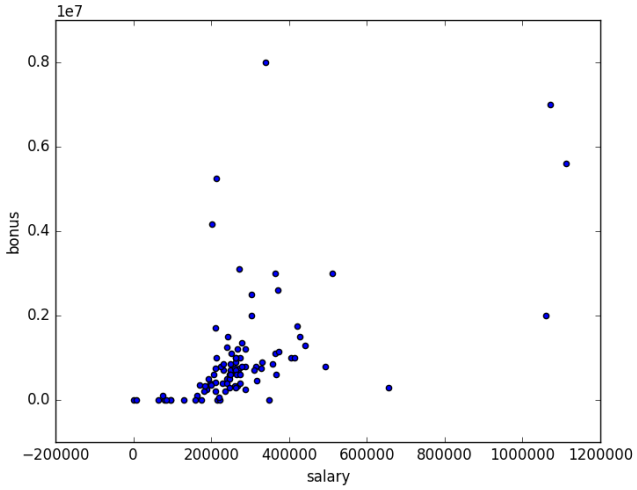
Utilizing ***scikit-learn*** and machine learning methodologies, I built a "person of interest" (POI) identifier to detect and predict culpable persons, using features from financial data, email data, and labelled data - POIs who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

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 was to utilize the financial and email data from Enron to build a predictive, analytic model that could identify whether an individual could be considered a "person of interest" (POI). Since the dataset contained labeled data--culpable persons were already listed as POIs--the value of this model on the existing dataset is limited. Rather, the potential value such a model may provide is in application to other datasets from other companies, to potentially identify suspects worth investigating further. The dataset contained 146 records with 14 financial features, 6 email features, and 1 labeled feature (POI). Of the 146 records, 18 were labeled, a priori, as persons of interest. Through exploratory data analysis and cursory spreadsheet/CSV review, I was able to identify 3 candidate records for removal:

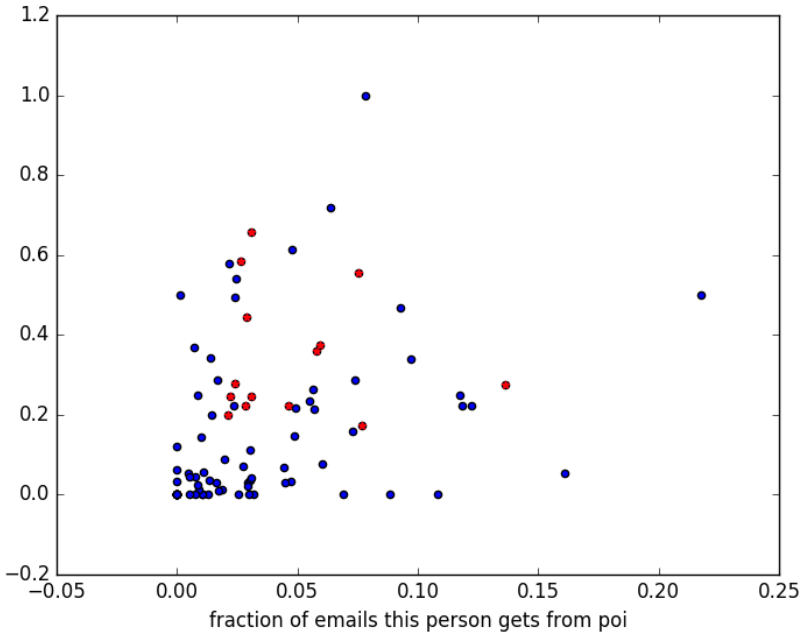
- *Total:* This was an extreme outlier for most numerical features, as it was likely a spreadsheet artifact.
- *The Travel Agency In The Park:* This record did not represent an individual.
- *Lockhart Eugene E:* This record contained no useful data.

The *Total* record become most apparent after visualizing the financial data on scatter-plots. Following data cleaning, 143 records remained.



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 importance 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.

After cleaning the data from outliers I had to pick the most sensible features to use. First I picked 'from_poi_to_this_person' and 'from_this_person_to_poi' but there was no strong pattern when I plotted the data so I used fractions for both features of 'from/to poi messages' and 'total from/to messages'.



2 new features were created and tested for this project. These were:

- Fraction of all emails to a person that were sent from a person of interest
- Fraction of all emails that a person sent to a person of interest

My hypothesis was that there is stronger connection between POIs via email that between POIs via email than between POIs and non-POIs. When we look at scatterplot we can agree that the data pattern confirms said above, i.e there is no POI below 0.2 on the y-axis.

In order to find the most effective features for classification, feature selection using Decision Tree was deployed to rank the features. Selection of features was half manual iterative process. Firstly, I

put all the possible features into features_list and then started deleting them one by one using score value and human intuition.

I picked 10 features which are:

["salary", "bonus", "fraction_from_poi_email", "fraction_to_poi_email", "deferral_payments", "total_payments", "loan_advances", "restricted_stock_deferred", "deferred_income", "total_stock_value"]

Accuracy for this feature set is around 0.8.

Approximate feature ranking:

Feature	Ranking
salary	**0.21170**
bonus	**0.14622**
fraction_from_poi_email	**0.12090**
fraction_to_poi_email	**0.11833**
deferral_payments	**0.09551**
total_payments	**0.08797**
loan_advances	**0.07478**
restricted_stock_deferred	**0.05341**
deferred_income	**0.05341**
total_stock_value	**0.03771**

But with these features my precision and recall were too low (less than 0.3) so I had to change my strategy and manually pick features which gave me precision and recall values over 0.3. In this dataset I cannot use accuracy for evaluating my algorithm because there are a few POIs in dataset and the best evaluators are precision and recall. There were only 18 examples of POIs in the dataset. There were 35 people who were POIs in 'real life', but for various reasons, half of those are not present in this dataset.

Finally, I picked the following features:

["fraction_from_poi_email", "fraction_to_poi_email", "shared_receipt_with_poi"]

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

Firstly, I tried Naïve Bayes but its accuracy was lower than with Decision Tree Algorithm (0.83 as compared to 0.9). I finally settled down with Decision Tree Algorithm because the feature set does not suit the distributional and interactive assumptions of Naïve Bayes well.

Decision Tree Algorithm gave me an accuracy of 0.9 before tuning the parameters. No feature selection was necessary for this algorithm.

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?

Tuning the parameters means to pull some of the 'levers' to influence the results of the model. If we don't tune our parameters appropriately, we will end up using the defaults, which will likely not result in an optimized model. This means the accuracy, precision, recall or other performance measures are not as good as they could be because the model wasn't customized to the particular dataset's features.

After selecting the features and the algorithm, I finally tuned the parameter **min_samples_split**.

min_samples_split	Precision	Recall
2	0.67	0.8
3	0.57	0.8
4	0.57	0.8
5	0.8	0.8
6	0.8	0.8
7	0.67	0.8
Average:	**0.68**	**0.8**

It turned out that the best values for min_samples_split are 5 and 6.

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

This process was validated using 3-fold cross-validation, precision and recall scores. First I used accuracy to evaluate my algorithm. It was a mistake because in this case we have a class imbalance problem – the number of POIs is small compared to the total number of examples in the dataset. So I had to use precision and recall for these activities instead. I was able to reach average value of precision = 0.68 and recall = 0.8.

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 evaluation metrics utilized were ***precision*** and ***recall***. Precision captures the ratio of true positives to the records that are actually POIs, essentially describing how often 'false alarms' are (not) raised. Recall captures the ratio of true positives to the records flagged as POIs, which describes sensitivity. Due to the unbalanced nature of the dataset (few POIs), ***accuracy*** is certainly not a good metric, i.e. if 'non-POI' had been predicted for all records, an accuracy of **0.9** would have been achieved. The average performance achieved for each of these metrics has been recorded above, i.e an average value of 0.68 for precision and 0.8 for recall.

Conclusion

The most challenging aspect of this project was the sparse nature of the dataset, with very few (18) POIs. Most of the algorithms employed perform much better in balanced datasets. One of the possible paths to improvement is digging in to the emails data more. The email features in the starter dataset were aggregated over all the messages for a given person. By digging into the text of each individual's messages, it's possible that more detailed patterns might emerge. Since we live in a world in which more POI finance data might not be easy to find, the next realistic thing to try might be to extract more data from the emails.

Resources

i. Coursera Machine Learning – https://www.coursera.org/learn/machine-learning
ii. Scikit-Learn Documentation For Logistic Regression - http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
iii. Scikit-Learn Documentation For Preprocessing - http://scikit-learn.org/stable/modules/preprocessing.html
iv. Background Study on Enron - http://en.wikipedia.org/wiki/Enron:_The_Smartest_Guys_in_the_Room