Identifying Fraud at Enron Using Emails and Financial Data

Project 4 - Udacity Nanodegree - Swaroop Oggu

Introduction

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 labeled data--POIs who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

Short Questions

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?

Goal: 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).

Data Set Details: Dataset contained labeled data- persons of interest were already listed .Therefor value of the model on the existing dataset is limited. But the pattern in identifying a POI can be used as a reference template in other companies.

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 enroninsiderpayreviw.pdf 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 data

The `TOTAL` record became 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 doesn't come ready-made in the dataset--explain what feature you tried to make, and the rationale behind it. If you used an algorithm like a decision tree, please also give the feature importances of the features that you use.

In order to optimize and select the most relevant features, I leveraged the use of the *scikit-learn's SelectKBest* module to select the 10 most influential features. Their associated scores and number of valid (non-NaN) records are listed in the table below:

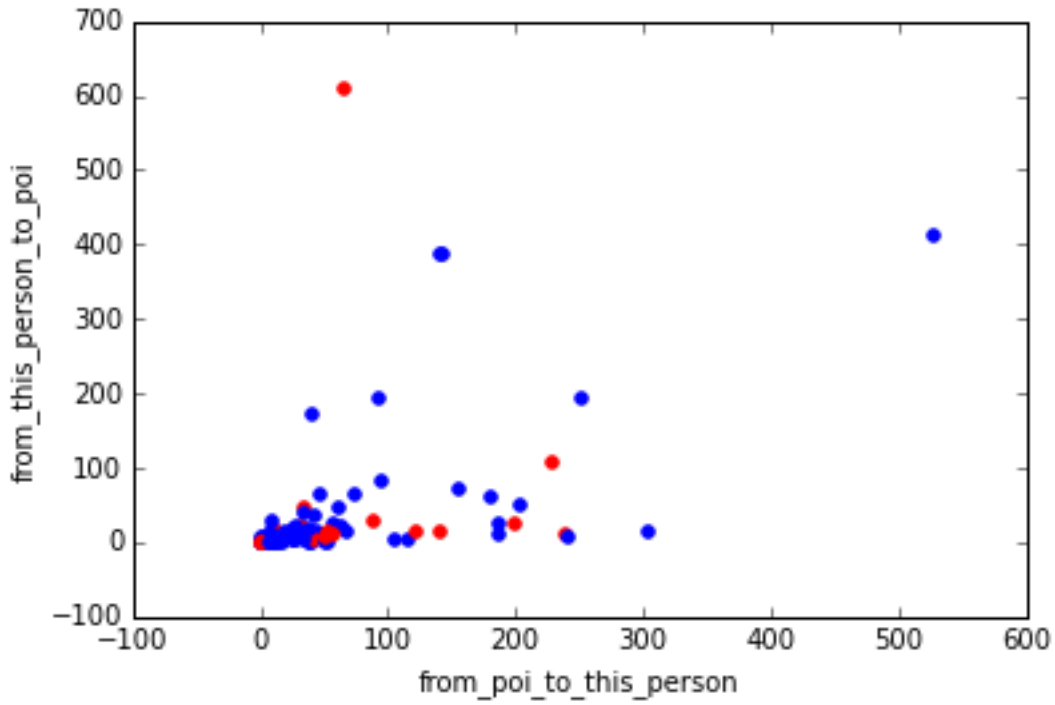
Feature	**Score**
exercised_stock_options	24.815
total_stock_value	20.792
bonus	20.792
salary	18.290
deferred_income	11.458
long_term_incentive	9.922
restricted_stock	9.212

total_payments	8.772
shared_receipt_with_poi	8.589
loan_advances	7.184

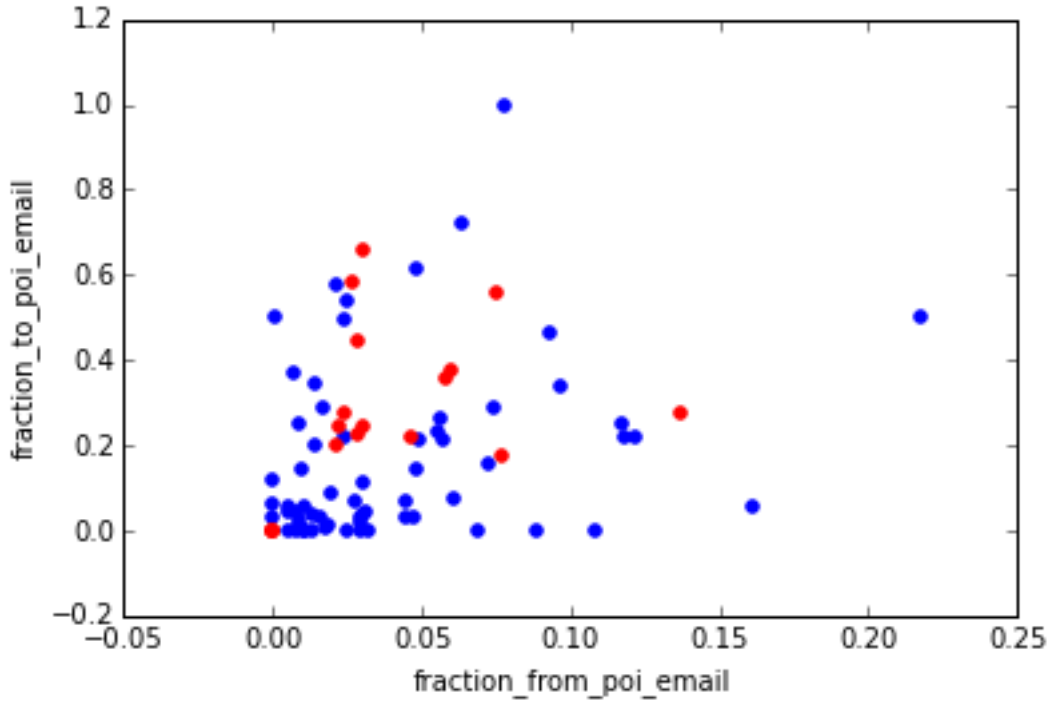
New Features Added:

- Networth :
  - Networth = sum of (exercised_stock_options,salary,total_stock_value). This way the consolidation of the financial feature helped fine tune the classifier
- fraction_from_poi_email
  - The fraction of all emails to a person that were sent from POI.
- fraction_to_poi_email
  - The fraction of all emails that a person sent addressed to POI

The above email features were created as there was no strong correlation between existing feature “from_poi_to_this_person“ and “from_this_person_to_poi”



But when fractions were plotted against each other its evident that there is some correlation with being POI and not. As the plot suggests all of the POI are above 0.2



All features were implemented with a min-max scaler. This was vitally important, as the features had different units (e.g. # of email messages and USD) and varied significantly by several orders of magnitude. Feature-scaling ensured that for the applicable classifiers, the features would be weighted evenly.

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

The final algorithm that I ended up using was a logistic regression. primary reason and motivation for using this was because the prediction outcomes were of binary classification (POI or non-POI).

I tried several algorithms, with a K-means clustering algorithm performing reasonably sufficient. Unsurprisingly, K-means clustering performed well, as the objective was to segregate the dataset into POI/non-POI groups.

I also tested, with marginal success, an AdaBoost classifier, a support vector machine, a random forest classifier, K means neighbor, Decision Tree Classifier and stochastic gradient descent (using logistic regression).

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? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier).

Algorithms may perform differently using different parameters depending on the structure of data. If you don't do this well, you can over-tune an algorithm to predict your training data extremely well, but fail miserably on unseen data

I tuned the parameters of the logistic regression and K-means clustering classifiers using exhaustive searches with the following parameters:

- Logistic regression: `C` (inverse regularization parameter), `tol` (tolerance), and `class_weight` (over/undersampling)
- K-means clustering: `tol`

K-means clustering was initialized with K (`n_clusters`) of 2 to represent POI and non-POI clusters. Additionally, K-means performed better by including only the 8 best features and the 2 additional features.

The other algorithms were tuned experimentally, with unremarkable improvement.

What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric item: "validation strategy"]

The main evaluation metrics utilized were `precision`, `recall and f1 score`.

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.

F1 score can be interpreted as a weighted average of the precision and recall,

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 87.4% would have been achieved. The results are as follows:

KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=2, n_init=10,
    n_jobs=1, precompute_distances='auto', random_state=None, tol=0.001,
    verbose=0)

precision: 0.586373670421

recall: 0.346

F1Score: 0.244471182153

LogisticRegression(C=1000000000000000000L, class_weight=None, dual=False,
        fit_intercept=True, intercept_scaling=1, max_iter=100,
        multi_class='ovr', penalty='l2', random_state=None,
        solver='liblinear', tol=1e-21, verbose=0)

precision: 0.4

recall: 0.4

F1Score: 0.4

Considering the above scores I would rank **precision** secondary to **recall**. Simply put, with the objective of 'flagging' individuals for further human-led investigation, it is more important that suspect individuals are included than innocent individuals be excluded. A high recall value would ensure that truly culpable individuals were flagged as POIs and would be investigated more thoroughly.

In optimizing for higher recall value, including auto-weighting in logistic regression proved to substantially increasing recall:

LogisticRegression(C=1000000000000000000L, class_weight='auto', dual=False,
        fit_intercept=True, intercept_scaling=1, max_iter=100,

multi_class='ovr', penalty='l2', random_state=None,

solver='liblinear', tol=1e-21, verbose=0)

precision: 0.25

recall: 0.6

F1Score: 0.352941176471

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. An interesting further study would be to employ anomaly detection algorithms (as used in other cases of fraud, like credit card checking) to identify persons of interest.

Resources and References

- Introduction to Machine Learning (Udacity)
- Machine Learning (Stanford/Coursera)
- scikit-learn Documentation