Identifying Fraud from Enron Emails and Financial Data

By Arpit Kanodia

INTRODUCTION

In 2000, Enron was one of the largest company with total revenue of $111 billion. Fortune named "America's Most Innovative Company" for six consecutive years.

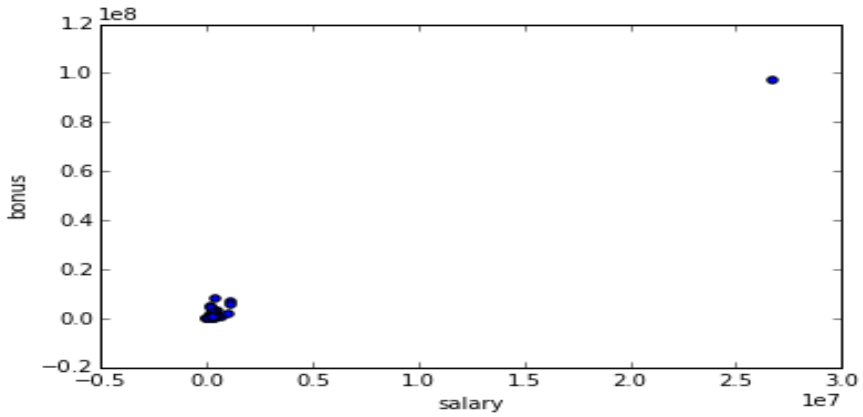
The Enron Scandal revealed in October 2001, that led the company to bankruptcy.

Using scikit-learn and many machine learning methodologies, I created a People of Interest(POI) identifier to predict the culpable persons, using features from financial data, email data and labeled data.

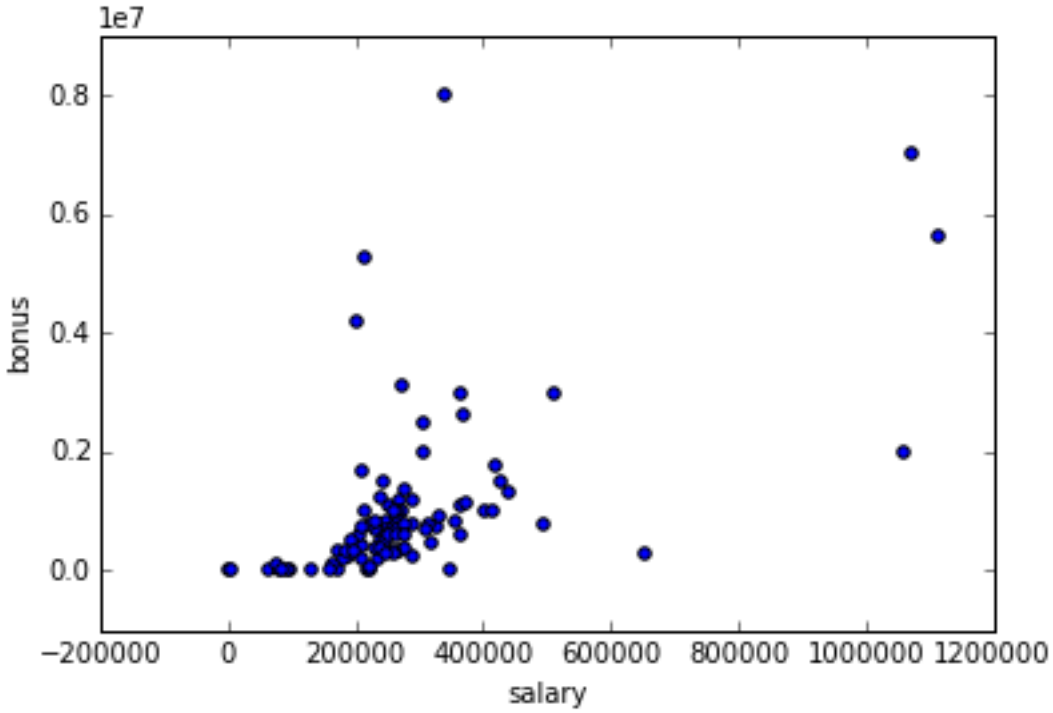
Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those?

The goal of the project is to develop a predictive and analytic model by choosing a combination of features of Former Employees and choose an algorithm that able to predict that a person should be considered as POI or not. The model may provide an application to identify potential suspects for further investigation, then finding proofs against them and ultimately filing charges on them.

The dataset contains 146 records with 14 financial features, 6 email features and 1labeled feature that is POI. For further analysis I created a CSV file and scatter plot.



Clearly TOTAL is a outlier, and need to be removed.



By further analysis of CSV file, I able to find 2 more outliers "THE TRAVEL AGENCY IN THE PARK" and "LOCKHART EUGENE E"

So, the three outliers are removed because of this reasons.

TOTAL : It's the most extreme outlier with high numerical value, and it is like a lone point in the scatter plot.

THE TRAVEL AGENCY IN THE PARK : This record doesn't represent the person.

LOCKHART EUGENE E : All the datasets for this data is filled with 'NAN'.

After removing this outliers dataset contains 143 records.

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 choose the best feature I used the SelectKBest module from scikit-learn. (http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html)

The list of features by their KBest Score.

[('exercised_stock_options', 24.815079733218194),

('total_stock_value', 24.182898678566879),

('bonus', 20.792252047181535),

('salary', 18.289684043404513),

('deferred_income', 11.458476579280369),

('long_term_incentive', 9.9221860131898225),

('restricted_stock', 9.2128106219771002),

('total_payments', 8.7727777300916792),

('shared_receipt_with_poi', 8.589420731682381),

('loan_advances', 7.1840556582887247),

('expenses', 6.0941733106389453),

('from_poi_to_this_person', 5.2434497133749582),

('other', 4.1874775069953749),

('from_this_person_to_poi', 2.3826121082276739),

('director_fees', 2.1263278020077054),

('to_messages', 1.6463411294420076),

('deferral_payments', 0.22461127473600989),

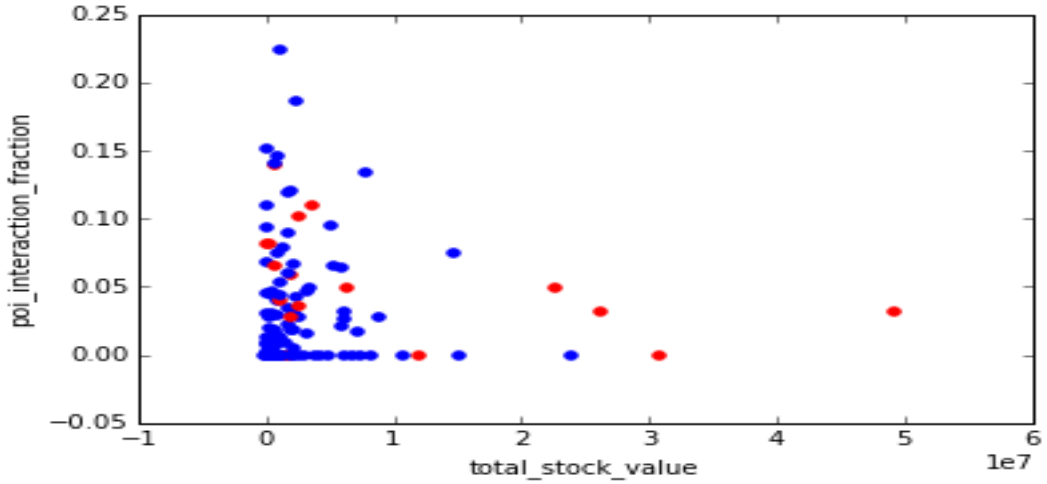
('from_messages', 0.16970094762175533),

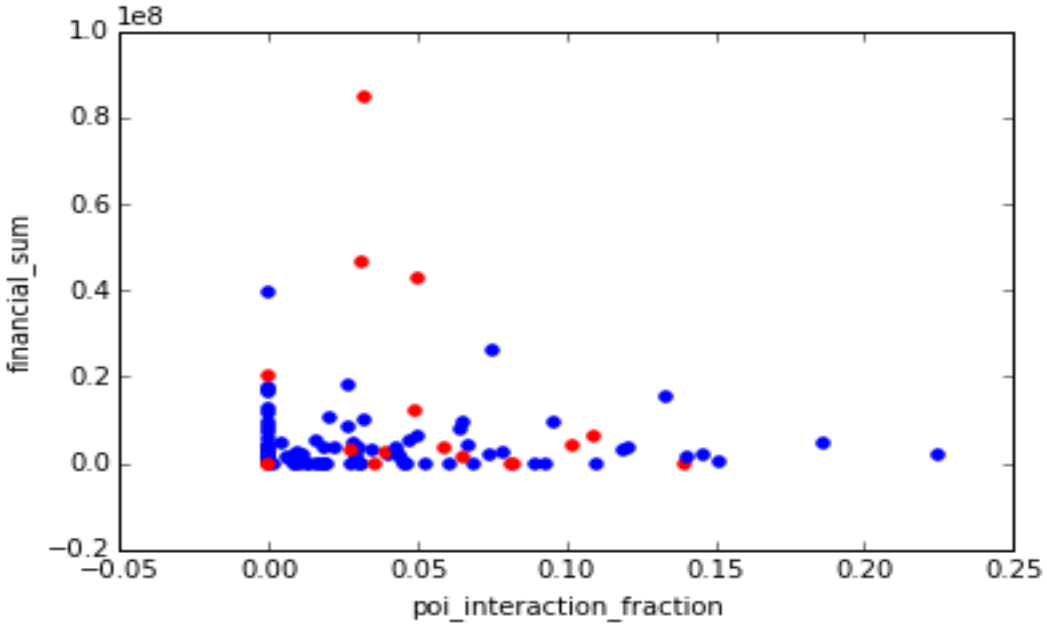
('restricted_stock_deferred', 0.065499652909942141)]

The top 10 most influential features are:-

10 most influential features: ['salary', 'total_payments', 'loan_advances', 'bonus', 'total_stock_value', 'shared_receipt_with_poi', 'exercised_stock_options', 'deferred_income', 'restricted_stock', 'long_term_incentive']

The K best approach is best in automated univariate feature selection, but it lack the email features. For this I created a feature named poi_interaction_fraction which is fraction of total number of emails to and from a POI to the total number of emails sent and received. I also created financial_sum, which is sum of 'total_stock_value', 'exercised_stock_options' and 'salary'. This feature created to simplify and to analyze how much wealth an individual have.





Let's check the how the features and 'custom features' affecting the Algorithms.

For K-N (Before Tuning)

Features	Accuracy	Precision	Recall
With 2 Most Influential Features	0.86246	0.68929	0.19300
Adding poi_interaction_fraction	0.86685	0.74410	0.20500
Adding financial_sum	0.82531	0.29250	0.09550
Adding both custom features	0.85557	0.44330	0.04300
With 4 Most Influential Features	0.86615	0.65971	0.26850
Adding poi_interaction_fraction	0.87493	0.64164	0.28200
Adding financial_sum	0.86108	0.65798	0.20200
Adding both custom features	0.87043	0.64856	0.20300
With 6 Most Influential Features	0.86693	0.60029	0.20500
Adding poi_interaction_fraction	0.86793	0.60926	0.21050
Adding financial_sum	0.86836	0.61840	0.20500
Adding both custom features	0.86993	0.63499	0.21050

With this table, it clearly shown the best results received with 4 most influential Features by adding poi_interaction_fraction, while financial_sum decreasing the performance, I am not going to add this feature in final result.

After Tuning the K-N with 4 features with poi_interaction_fraction (Total 5 Features)

	Accuracy	Precision	Recall
Before Tuning	0.87493	0.64164	0.28200
After Tuning	0.88079	0.63293	0.39400

Definitely after tuning Algo is working better

For Logistic Regression

Features	Accuracy	Precision	Recall
With 8 Most Influential Features	0.85760	0.40503	0.14500
Adding poi_interaction_fraction	0.85633	0.39653	0.14850
Adding financial_sum	0.85093	0.36499	0.15950
Adding both custom features	0.84900	0.34926	0.15350
With 10 Most Influential Features	0.86347	0.47015	0.18900
Adding poi_interaction_fraction	0.86447	0.47925	0.19050
Adding financial_sum	0.85920	0.43708	0.19450
Adding both custom features	0.85827	0.42984	0.19300
With 12 Most Influential Features	0.83640	0.28173	0.14650
Adding poi_interaction_fraction	0.83520	0.27524	0.14450
Adding financial_sum	0.83153	0.25892	0.14150
Adding both custom features	0.83167	0.25984	0.14200

The logistic regression is working better with 10 features adding poi_interaction_fraction.

Also before the different machine learning algorithm classifiers, I scaled all features according to max and min.

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

I tried many algorithms KNeighbours, Decision Tree, Gaussian NB etc. I also tried logistic regression (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

I thought to use this because the prediction outcomes are binary based, i.e. POI or Non-POI.

Algorithm Performance with 5 features

	Accuracy	Precision	Recall
LogisticRegression	0.85187	0.40036	0.22300
KNeighbors	0.88079	0.63293	0.39400
DecisionTree	0.81127	0.28371	0.27250

Clearly, the Accuracy, Precision and Recall in KNeighbors are working way better than LogisticRegression, with a huge difference in between Recall.

Algorithm Performance with 11 features

	Accuracy	Precision	Recall
LogisticRegression	0.86447	0.47925	0.19050
KNeighbors	0.86247	0.47249	0.27050
DecisionTree	0.81120	0.28732	0.28100

Still, K-N working better, so I ultimately chooses K-N with 4 most influential feature adding poi_interaction_fraction(Total 5 features).

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).

The machine learning algorithms are parameterized so it behavior and performance can be tuned. It's important to tune the algorithm for getting best results of recall or precision or accuracy or for all of these. For tuning the KN, I used the GridSearchCV.

For KN I tuned this parameter using GridSearchCV

- Number of neighbors
- Weights : Uniform or Distance
- KN Algorithms used to compute
- Leaf size
- Minkhowski Parameter

```
tune_kn_clf_recall = GridSearchCV(KNeighborsClassifier(),
                    { "n_neighbors" : [1, 2, 3, 4,5],
                    "algorithm":('auto','ball_tree','kd_tree','brute'),
                     "p":[1,2,3,4,5],
                     "leaf_size":[10,20,30,40,50],
                     "weights":('distance','uniform')},
                     scoring = 'recall')
```

For this the best parameters I got

{'n_neighbors': 3, 'weights': 'uniform', 'leaf_size': 10, 'algorithm': 'auto', 'p': 3}

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

Validation is performed to ensure that a machine learning algorithm generalize the trends well. This reduces the problem of over fitting the model.

I used cross validation from sklearn to use train test split to do the splitting of the data into training set and test set with a test size of 0.3.The num_iters I used is 1100.

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.

Algorithm Performance

	Accuracy	Precision	Recall
LogisticRegression with 11 Features	0.86447	0.47925	0.19050
KNeighbors with 5 Features	0.88079	0.63293	0.39400

The main evaluation metric I used Precision and Recall (but also keeping in mind about decent Accuracy).

Accuracy: Accuracy is the ratio of correct predictions out of the total predictions are made. It means how many POI the algorithm able to predict right. In this case 88% of predictions are correct.

Precision: The precision is ratio of correct positive predictions made out of the total positive predictions. 1245 total positive predictions are made and the total positive predictions that are correct is 788. This means 63.29% of precision.

Recall: Recall is the ratio of correct positive prediction to the actual that were indeed positive (correct positive predictions + incorrect false negatives). The algorithm able to achieve 39.4% of recall.

Both algorithms gave almost similar results but there is big difference in precision and recall. But in my sense Recall is primary metrics for describing the result in this case, a high recall is needed to ensure that truly culpable individual were flagged as POI and investigated thoroughly.