Enron POI Identifier Report

Martin Welss

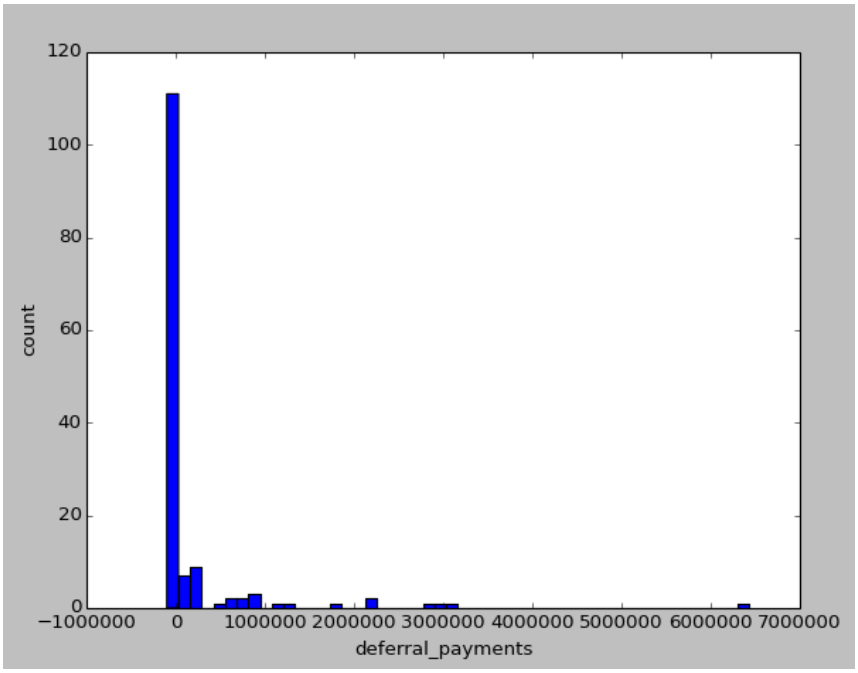
Introduction

The Enron case was one of the biggest cases of company fraud in the history of the USA. In the last years before the collapse, the managers created more and more complex accounting strategies to pretend fabulous profits but in reality produced only losses. These strategies where backed by the accountants of Andersen Consulting which disintegrated in the follow up turmoil and in the course of the trial. There is an excellent book which describes the rise and fall of Enron in great detail: “The Smartest Guys in the Room” by Bethany McLean.

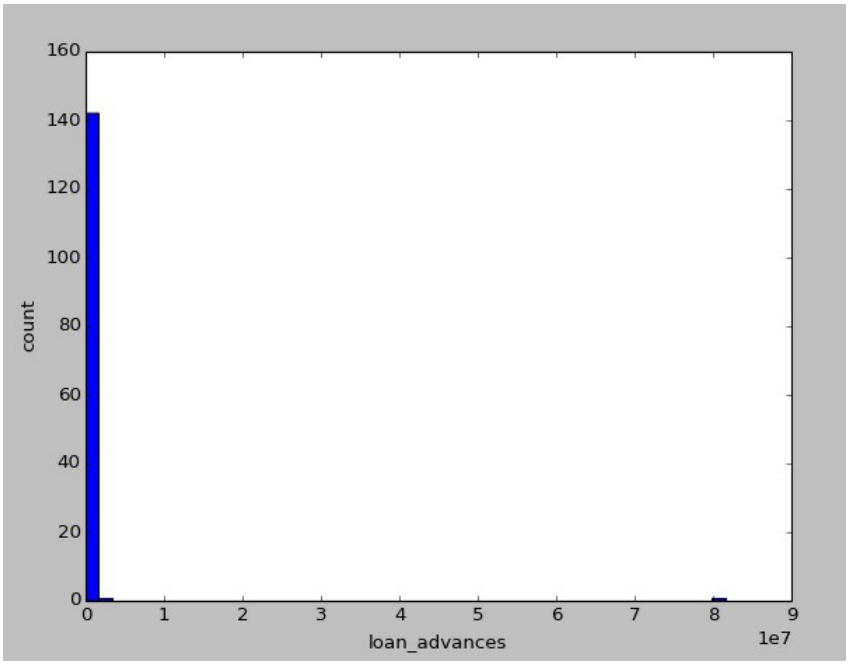
The Enron Dataset

The poi_names.txt file contains a list of POIs and preceding each POI is a flag, which signals if the inbox of that person is available. There are 35 POIs in this list and we have the inbox of only of four of them. In the actual dataset there are records of 146 persons that are potential POIs. The goal of the project is to write a machine learning program that identifies the POIs out of the complete dataset.

The first step is to scrutinize the dataset for irregularities and outliers. One outlier can be found in the dataset which is a list of totals from a spreadsheet export. This outlier is removed from the dataset at the beginning of the program. Looking at the data manually by plotting the distributions, one can see that most of features have many missing values. I included two diagrams as examples:



The feature “deferral payments” has more than 100 missing data points and the following distribution of “loan advances” has more than 140 missing data points (that's nearly all!)



Feature Selection

Inspired by the analysis above and the fact of many missing values, I ranked the features by counting the NaN values and the “Hit count”. A data point is a “hit”, if its value is not NaN and the person is a POI. So the first line in the table means that for the feature “total_stock_value” there are only 20 NaNs and 18 POIs have a not NaN as value.

I ranked them from most promising (total_stock_value) to the least meaningful feature (restricted_stock_deferred).

Feature	NaN count	Hit count
total_stock_value	20	18
total_payments	21	18
expenses	51	18
other	53	18
restricted_stock	36	17
salary	51	17

bonus	64	16
to_messages	59	14
from_poi_to_this_person	59	14
from_messages	59	14
from_this_person_to_poi	59	14
shared_receipt_with_poi	59	14
exercised_stock_options	44	12
long_term_incentive	80	12
deferred_income	97	11
deferral_payments	107	5
loan_advances	142	1
director_fees	129	0
restricted_stock_deferred	128	0

Furthermore I created two more features:

- bonus_by_salary which is the bonus divided by salary which should give a good impression of the relation between bonus and salary for that particular person.
- email_fraction_from_poi which is the fraction of emails the person received that was sent by a POI. The idea is that POIs are likely to sent more messages to other POIs

Selecting a Classifier

The first classifier that I tried was a GaussianNB. But with several variations with up 6 of the highest ranking features from the table above, the classifier did not reach the required precision of 0.4: the top precision was 0.38. Since GaussianNB has no parameters to tune I switched to SVM. I tried three different kernels: linear, rbf and poly but none of them gave satisfying results.

I finally went for a DecisionTreeClassifier which I tuned with GridSearchCV. I tried several combinations and finally went with this parameter grid:

```
param_grid = {
   'min_samples_split': [2, 3, 4],
   'max_depth': [3,4,5],
   'criterion': ['gini', 'entropy'],
   'splitter': ['best', 'random']
}
```

The result is as follows:

Accuracy: 0.85240 Precision: 0.31741 Recall: 0.09300 F1: 0.14

Discussion and Conclusion

A precision of 32% is not really satisfying because it means there are more misses than hits. I was really wondering why Naive Bayes effectively gave better results than the fine tuned DecisionTreeClassifier. I think one of the main problems with this dataset are the many missing values. Maybe it would be one way to improve the results to go back to the data collection and try to complete the data for the features.