Project 5 Identify Fraud from Enron

Final Project Documentation

I. Project Overview

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, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives.

In this project Machine Learning techniques are used to find Persons of Interest (POI) by analysing the information (email and financial data).

In the Beginning a poi had to be defined. This was done by the Katie (class instructor). The main criteria was whether the person got a trial or received a penalty. Katie created a list of 35 people. According to our definitions of POIs these persons were very likely to be involved in the Enron fraud.

The data set that was analysed had 146 data points, only 18 POIs (data points).

The project was separated in 4 main parts:

- Feature Selection
- Machine Learning Algorithm
- Final Algorithm/Validation
- Conclusion

II. Feature Selection

Before using a machine learning algorithm one had to find out which features had the most significance concerning the classification of the data.
There were overall 20 features in the data set excluding the label POI.

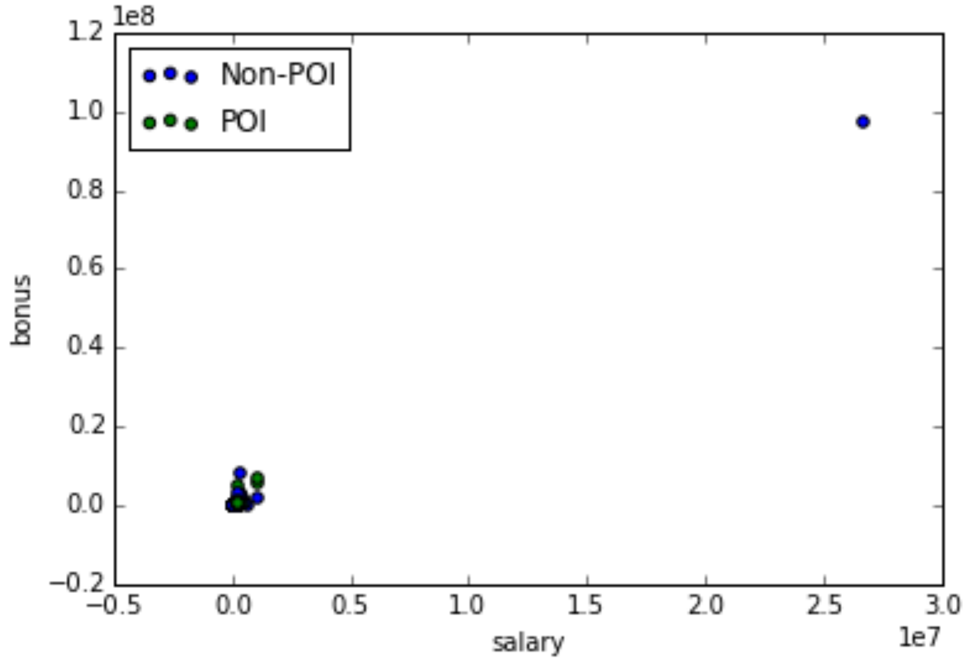
email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'poi', ‘shared_receipt_with_poi']

financial features: ['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees']

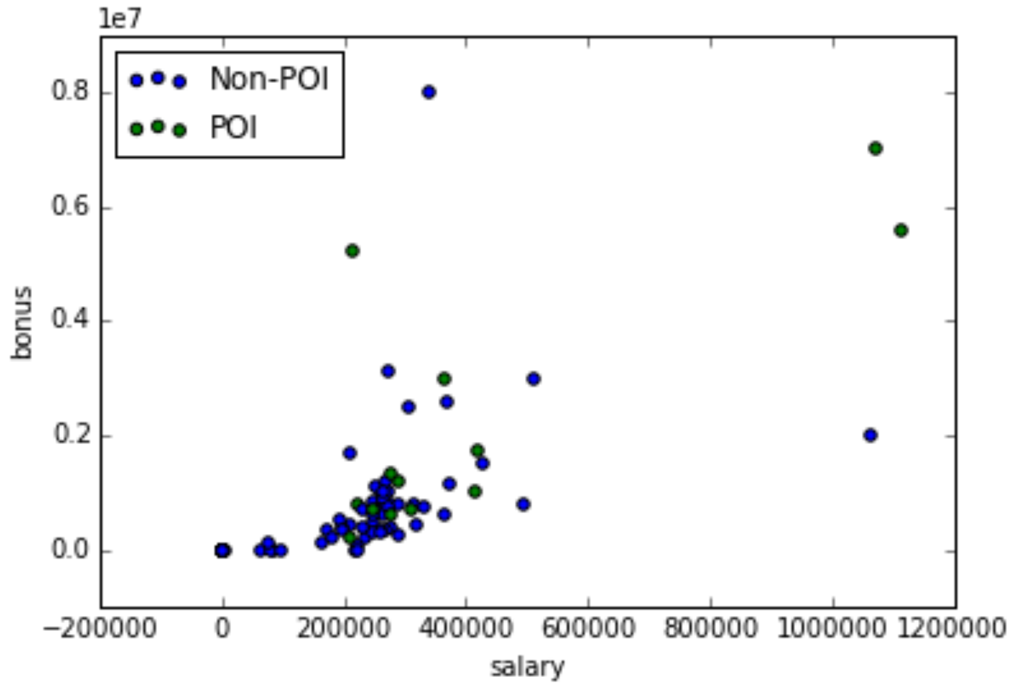
My first approach to find significant features was the drawing of different scatterplots. It is important to use the right features with the most variance (referring to PCA and feature reduction). The best way for Machine Learning algorithms is to have less features with more information. It is important to keep in mind that features do not mean more information. The goal is to find a middle way of features and performance of the algorithm.

The following scatterplots show some features combinations.
Only the Training Data is shown in the scatterplots. The data was divided in 70% training data and 30% test data.

1. Scatterplot Salary/Bonus (values in US Dollar) (was also discussed in the mini-projects)



After removing the outlier *Total* which calculated the sum in the underlying spreadsheet of the data (which was also done in the mini project before).

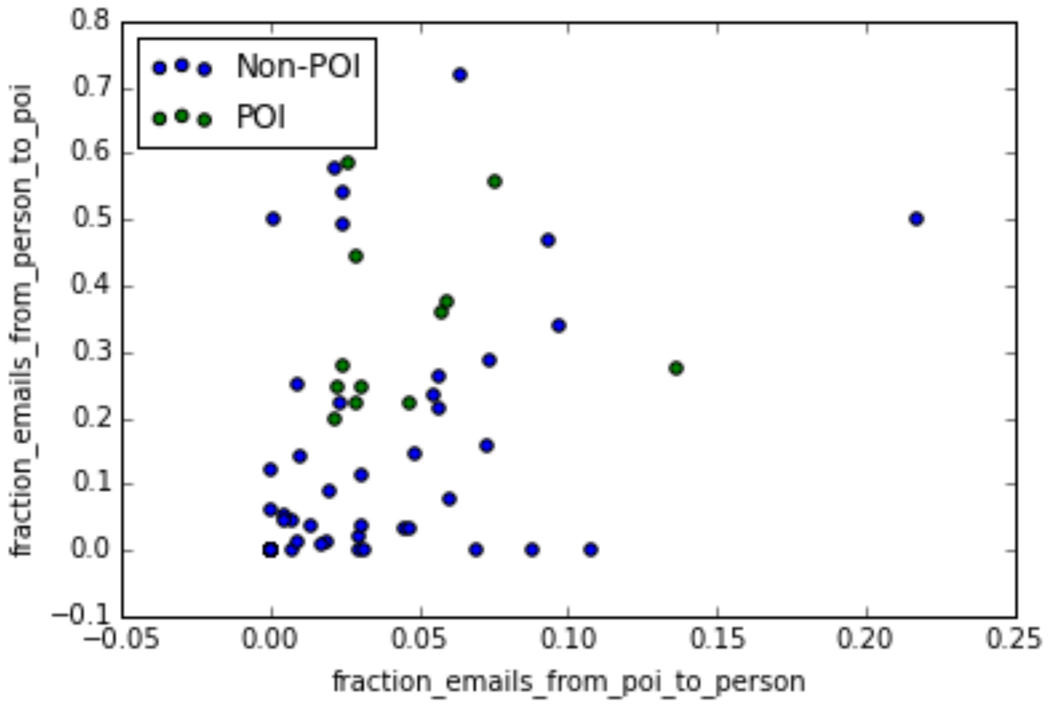


Analysing the scatter plot it is difficult to recognise a decision surface. There is no clear separation. Besides the 'Total outlier' there are two other outliers which had a very high bonus. Both are POIs and famous people in the Enron fraud (real outliers) : Lay and Skilling. There were real data points and were not removed (they are very important POIs)

2. Self created features(these features were also covered in the mini-project):

- The feature "Fraction_emails_from_poi_to_person" describes the fraction of emails a person received from a POI.
- The feature "Fraction_emails_from_person_to_poi" describes the fraction of emails a person sent to a poi.

One can see that all POIs sent at least 20% of their emails to other POIs. But there are also Non-POIs who sent more than 20% of their mails to other POIs. It does not look like a clean decision surface but in the next step machine algorithms are used and compared.



III. Machine Learning Algorithm

Before implementing the algorithm a feature selection had to be done. So I used my intuition and the results of the scatter plot and picked the following 9 features for my algorithm:

Feature_List = [salary,exercised_stock_options,bonus,shared_receipt_with_poi,
fraction_emails_from_poi_to_person, fraction_emails_from_person_to_poi,
total_stock_value,expenses,long_term_incentive]

Different Algorithms:

DecisionTree without GridSearch

Decision Tree without GridSearch and with 9 features mentioned above	
Score	0,88
recall	0,34
precision	0,25
f1 score	0,286

Feature Importance DecisionTree without GridSearch

Feature Importance	
salary	0,000
exercised_stock_options	0,100
bonus	0,238
shared_receipt_with_poi	0,249
fraction_emails_from_poi_to_person	0,053
fraction_emails_from_person_to_poi	0,219
total_stock_value	0,000
expenses	0,145
long_term_incentive	0,000

Both the recall and precision were low so I tried to optimise the Decision Tree algorithm by using GridSearch.

The parameters which were variable:

parameters = {‘min_samples_split':(2,10,100),'max_features':('sqrt','log2',None),}

The accuracy was better, but recall and precision deteriorated.

Result Decision Tree with GridSearch

Decision Tree with GridSearch and 9 features	
Score	0,929
recall	0,0
precision	0,0
f1 score	0,0

- The Second algorithm I used was Gaussian Naive Bayes :

The results were better than above, so the features correlated with the normal distribution which is the fundamental of the Gaussian Naive Bayes Classifier.

Gaussian Naive Bayes with 9 features

Gaussian Naive Bayes with 9 features	
Score	0,905
recall	0,67
precision	0,4
f1 score	0,5

To improve the classifier and to avoid an overfitting with to many features, I used SelectKBest.

Score for the features using SelectKBest

Score SelectKBest	
salary	17,15
exercised_stock_options	23,69
bonus	19,72
shared_receipt_with_poi	7,96
fraction_emails_from_poi_to_person	2,80
fraction_emails_from_person_to_poi	15,85
total_stock_value	23,05

Score SelectKBest	
expenses	5,54
long_term_incentive	9,32

Comparing different result for different K's, the NB performance was nearly constant, the DecsionTree with GridSearch however had the best results for k=7 feature (F1_Score = 0,67).

Results for the different algorithms using SelectKBest:

k in SelectKBest	2	3	4	5	6	7	8
DTwithGrid_Score	0,93	0,93	0,93	0,93	0,93	0,83	0,93
DTwithGrid_Recall	0	0	0	0	0	0,34	0
DTwithGrid_Precision	0	0	0	0	0	0,17	0
DTwithGrid_F1Score	0	0	0	0	0	0,23	0
DTwithoutGrid_Score	0,71	0,83	0,87	0,81	0,83	0,95	0,86
DTwithoutGrid_Recall	0	0	0	0,67	0,67	0,67	0,34
DTwithoutGrid_Precision	0	0	0	0,23	0,25	0,67	0,2
DTwithoutGrid_F1Score	0	0	0	0,34	0,36	0,67	0,25
NB_Score	0,93	0,90	0,93	0,93	0,93	0,93	0,93
NB_Recall	0,67	0,67	0,67	0,67	0,67	0,67	0,67
NB_Precision	0,5	0,4	0,5	0,5	0,5	0,5	0,5
NB_F1Score	0,57	0,5	0,57	0,57	0,57	0,57	0,57
SVM_Score	0,93	0,93	0,93	0,93	0,93	0,93	0,93
SVM_Recall	0	0	0	0	0	0	0
SVM_Precision	0	0	0	0	0	0	0
SVM_F1Score	0	0	0	0	0	0	0

Abbreviations for the table above:

DTwithGrid	:= DecisionTree with GridSearch
DTwithoutGrid	:= DecisionTree without GridSearch
GNB	:= Gaussian Naive Bayes
SVM	:= Support Vector Machine

Feature Scaling:

Feature Scaling is an important part in machine learning. Most of the features have different units and some algorithms use distance calculations for classification. Most common example is the SVM classifier which tries to maximise the perpendicular distance between decision surface and nearest data point of every class. So it is important all features have the same dimension otherwise the distances would be not trustworthy. Considering a non-supervised learning algorithm (regression), feature scaling is very important because the algorithm tries to predict values by drawing a linear line (Linear Regression) so the normalisation effects are significant.

The Gaussian NB and Decision Tree classifiers were not influenced by feature scaling, because every feature is considered independently.

After applying the MinMaxScaler to the features I improved the evaluation metrics of the SVM classifier significantly.

SVM after Feature_Scaling, SelectKbest (K=7)

Score	0,95
recall	0,33
precision	1,0
f1 score	0,5

SVM after Feature_Scaling, SelectKBest (K=9), StratifiedShuffleSplit(1000)

Score	0,85
recall	0,13
precision	0,21
f1 score	0,16

After applying StratifiedShuffleSplit(1000) the evaluation metrics worsen significantly. That is why I choose Naive Bayes for the final classification.

Parameter Tuning:

Machine Learning algorithms have a lot of parameter which can be tuned to influence and improve the outcome. For example SVM offers parameters like C (penalty parameter), kernel, gamma. To find the best combination of these parameters one can use common method such as GridSearch, Bayesian Optimisation or Random Search. When using GridSearch all parameter values are tried out and the best solution (combination of the specified parameters) is used for the classification. The cost for this operation is very high regarding big data sets, so one can use Random Search which tries parameter values randomly. But parameter tuning does not ensure a better result as seen above.

IV. Final Algorithm/Validation

Validation is very important before using the ML classifier for further problems/data sets. One need to know how the classifier performs. Moreover a separation in training and test data is mandatory. The training data is used to fit the classifier, the test data instead validates the classifier. At the beginning you want to collect information as for example slope and bumpiness data to predict the speed for an autonomous car in a specific terrain. Then you separate the data in training and test sets. After the training of the classifier independent test data is used for validation, because otherwise the result would be misguided. Using training data for validation will always show very high results compared to test data evaluation (KFold also splits the data in training and testing but uses the entire data for both which is more accurate).
Validation is also important regarding statistical type 3 errors where you reject the hypotheses using the wrong reason. When we don't split the data in training and testing set we try to prove a hypothesis with the same data which lead to this analysis.

Evaluation:

I used recall,f1_score and precision for the evaluation of the algorithms. The problem with this data set was the huge difference in numbers between POIs and Non-POIs. So the accuracy is not a good parameter for evaluation. In our case if our Classifier would predict no POI at all, our accuracy would be 87,7%. We see the accuracy is still high.
Moreover the data was split in Training and Test data.

After deciding to use the Gaussian NB classifier I split the data set with the StratifiedShuffleSplit algorithm (Folds=1000). KFolds would be not a decent option because of the imbalance of the data.

Gaussian Naive Bayes with SelectKBest(K=9), StratifiedShuffleSplit(1000)

Score(mean)	0,83
recall(mean)	0,32
precision(mean)	0,39
f1 score(mean)	0,35

So after applying the StratifiedShuffleSplit the results deteriorated. But because of the imbalance of the data this split is necessary to get proofed results.

It was necessary to use precision and recall for the evaluation of the classifier.
A high precision means that when we indicate a POI it is very likely that it is actually a POI. So when the precision is high we avoid label innocent people as POIs *(True Positives/(True Positives + False Positives))*.
The recall predicts how good the classifier finds all the existing POIs in the data set. *(True Positives/(True Positives + False Negatives))*.So a high recall means the classifier identifies a lot of POIs in the data set. So it depends what is more important for the data analyst. In our case of the enron data set, is it more important to find all POIs and accept that we label innocent people as POIs (high recall, small precision) or does he wants to be sure that no innocent people are labeled as POIs but we will not identify that much POIs (high precision, lower recall)
The results of the Gaussian NB classifier were 39% precision and 32% recall. So it seems to be a trade-off between finding all POIs in the dataset (recall) and being sure that we identified the right person (precision). The values are in the same area.
In addition to the values above the f1 score was calculated and used as an evaluation metric. This score is a combination of precision and recall and takes both evenly into account. The f1 score was 35%.

V. Conclusion

The Enron data set had a huge gap between the number of POIs and Non POIs, so it was necessary to use evaluation metrics like recall and precision to choose the right classifier. moreover after apply the StratifiedShuffleSplit the results deteriorated. So the first results, where I used the normal split function, were not so informative because I split function separated the data just in the right way. Nevertheless the results were above 30% which is ok.
Several techniques were used to improve the classification (SelectKBest,GridSearch,Outlier_detection). None of them had a massive improvement. To get better result one have to dig deeper in the emails, especially the content of the mails, because until know we only looked at the recipient and forwarder.

VI. Reference

- Udacity Intro Machine Learning
- Udacity Forum
- Sklearn Documentation
- wikipedia (Recall/Precesion, SVM, cross-validation (statistic), hyperparameter-optimization)
- Programming libraries Python and stackoverflow

VII. Personal Statement

I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.