Project Machine Learning

Enron Fraud

1 Overview

1.1 Machine Learning

In case of the Enron fraud is not clear who are the persons, which was involved in the fraud. There are some employees which are defined as “Person of interest”. Only defining a Person as POI doesn’t means that this employee was really involved at the fraud. To find the real persons of interest, a technique is needed that can decide, by looking at different attributes, if someone is involved at the fraud or not. With machine learning is it possible to train the “machine” which should give an answer for the question: “Who was involved in the fraud from Enron?”.

1.2 Data Set

The Enron Data set is a really small data set which has only 146 persons inside and they are described by 23 attributes, of in this case called feature. In the dataset are 18 employees which have the Flag “poi”.
Features are information about the employees of Enron, for example there salary, bonus, written / send emails with POIs, a POI-Flag and so on. To analyze the involved fraud employees the machine learning algorithm should look at the features like salary or bonus and the communications via mail with POIs. These are attributes, which can indicate that this person is a real POI. For example, an employee gets a very high, not typical bonus.

By looking at the data, there are some datasets that can be defined as “noise”. For example the employee Eugen Lockhard has no informations, all 23 features are empty. (Information can be found in the PDF-File (enron61702insiderpay.pdf). This is a dataset that should be deleted because it has no sense to perform a machine learning algorithm with an empty dataset.

Another test, which I have run, is to look at the keys of the data set. The “key” is the employee itself. By printing out the whole list of employees, there is a name “THE TRAVEL AGENCY IN THE PARK”, which doesn’t looks like an employee of a company. This is a dataset which can influence the machine algorithm by deciding / predicting. For this “employee” are some features filled with information.

Another way to clean up the data is to look only at employees with a salary. By deleting the employees without salary we are losing 51 employees. At the end there are only 94 employees in the data set.

The Dataset includes “Total”, which sums up all the values for every feature, this data should be deleted because the values are extremely high and influence the analysis. This outlier was removed with the “pop” command.

2 Select and create features

2.1 Selecting features

For selecting the most important features for the machine learning algorithm, I used "SelectKBest". SelectKBest returns the features and there "rating"-scores. These rating scores describes how important is a features, if a features as a higher score than the other ones than it means that this features is more important for predicting. For getting the best features I feed the SelectKBest algorithm with all available features, for loosing no important feature. (The features selection can be found in the "select_features.py")

The output was the following:

```
# --- Output of all features and there scores
"""
[('exercised_stock_options', 25.097541528735491),
('total_stock_value', 24.467654047526398),
('bonus', 21.060001707536571),
('salary', 18.575703268041785),
('deferred_income', 11.595547659730601),
('long_term_incentive', 10.072454529369441),
('restricted_stock', 9.3467007910514877),
('total_payments', 8.8667215371077752),
('shared_receipt_with_poi', 8.7464855321290802),
('loan_advances', 7.2427303965360181),
('expenses', 6.2342011405067401),
('from_poi_to_this_person', 5.3449415231473374),
('other', 4.204970858301416),
('from_this_person_to_poi', 2.4265081272428781),
('director_fees', 2.1076559432760908),
('to_messages', 1.6988243485808501),
('deferral_payments', 0.2170589303395084),
('from_messages', 0.16416449823428736),
('restricted_stock_deferred', 0.06498431172371151)]
"""
```

The features are sorted descending by score. I chose the first 12 features, because I thought that the feature "from_this_person_to_poi" can be also an interesting player in the prediction, although if the score is not as high as the others. To get the best Accuracy depending on features, I played a little bit around. With features exercised_stock_options to restricted_stock I get the following Accuracy with Naive Bayes:

```
Count Dataset:  146
Count features:  21
Count POIs:  18
Accuracy Naive Bayes:  0.880952380952
```

Choosing total payments up to the first 12 gives an Accuracy of 0.86 so its worst.

Then I tried only the first for, included salary and get this Accuracy:

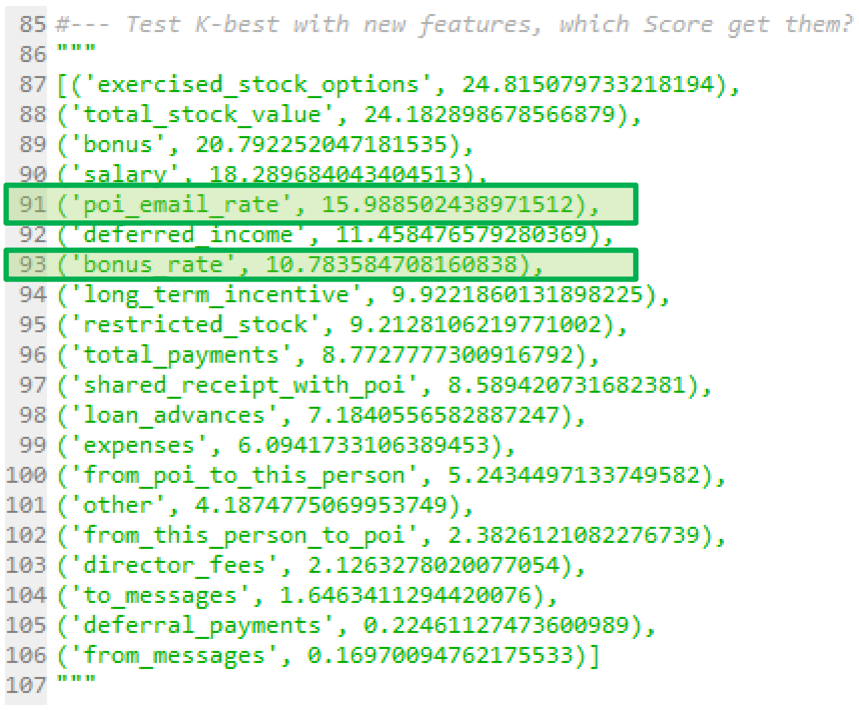
```
Count Dataset:  146
Count features:  21
Count POIs:  18
Accuracy Naive Bayes:  0.825
```

This returns the worst result.

The best result can be found if only is poi and exercised_stock_options chosen.

```
Count Dataset:  146
Count features:  21
Count POIs:  18
Accuracy Naive Bayes:  0.921052631579
```

After feature creation the list changes a little bit:



```
#--- Test K-best with new features, which Score get them?
"""
[('exercised_stock_options', 24.815079733218194),
('total_stock_value', 24.182898678566879),
('bonus', 20.792252047181535),
('salary', 18.289684043404513),
('poi_email_rate', 15.988502438971512),
('deferred_income', 11.458476579280369),
('bonus_rate', 10.783584708160838),
('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)]
"""
```

Looks like that are two new good features, because the score is good.
Explanation of these features follows in the next chapter “create new features”.

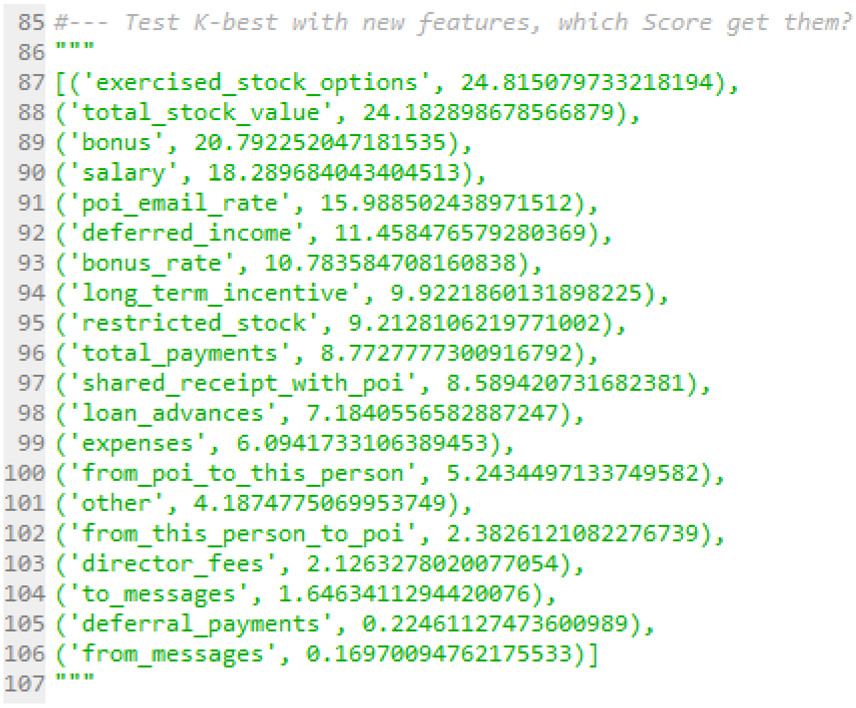
2.1.1 Feature Scaling

In case of this data set features scaling has no influence of the result, this means it is not needed.

2.2 Create new features

In this section I describe two new created features, which I found could be interesting to find the employees which were involved at the Enron fraud. (create_new_features.py)

The first new feature is the poi email rate. There will be calculated how high the rate of mails to or from POI per employee is.
The second new feature is the bonus rate. This rate calculates how high the bonus is depending on salary. With this feature I tried to find out, which employees get an extremely high bonus and who not. An extreme high bonus can be a sign for that this employee was part of the fraud.



```
#--- Test K-best with new features, which Score get them?
"""
[('exercised_stock_options', 24.815079733218194),
('total_stock_value', 24.182898678566879),
('bonus', 20.792252047181535),
('salary', 18.289684043404513),
('poi_email_rate', 15.988502438971512),
('deferred_income', 11.458476579280369),
('bonus_rate', 10.783584708160838),
('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)]
"""
```

Both features get good scores at the K-best test and will be used in the machine learning algorithm.

3 Machine Learning Algorithm

3.1 Which algorithms were tried?

I tried the machine learning algorithms Naïve Bayes, Support Vector Machine (SVM), Decision Tree and Adaboost.

3.2 How did model performance differ between algorithms?

After running the 4

Algorithms	Precision	Recall	F1-score	Accuracy
Naive Bayes	0.6	0.75	0.66	0.921052631579
SVM	0.2	1.0	0.33	0.894736842105
Decision Tree	0.4	0.4	0.4	0.842105263158
Adaboost	0.2	0.25	0.22	0.815789473684

After running the different algorithms without tuning and usage of the given cross-validation, the Naive Bayes algorithm delivers the best accuracy but the worst values for Precision, Recall and F1.

SVM returns the second best accuracy and the best score for recall.
The result of Adaboost is also not so bad.

Adaboost has in all scores the words values it will exclude from the analysis. I will also exclude Decision Treem where the values compared to SVM and Naive Bayes not good enough.

In the next step I generate the best parameters with GridSearchCV and will look again and the scores for Naive Bayes and SVM. The algorithm with the best scores will be chosen. Naive Bayes can't be tuned. (https://stats.stackexchange.com/questions/299842/why-grid-search-is-not-performed-for-naive-bayes-classifier/299843)

For the GridSearchCV I used the example from eigenfaces and changed some of the parameters.

Algorithms	Precision	Recall	F1-score	Accuracy
Naive Bayes	0.4	0.4	0.4	0.921052631579
SVM	0.4	0.5	0.44	0.868421052632

All Scores changed for Precision and f1 we get a better value, for recall a worst value. Also the Accuracy getting more worst with the new “best” classifier.

3.3 Which algorithm was chosen?

Like the scores in the previous chapter, Naive Bayes should be used.

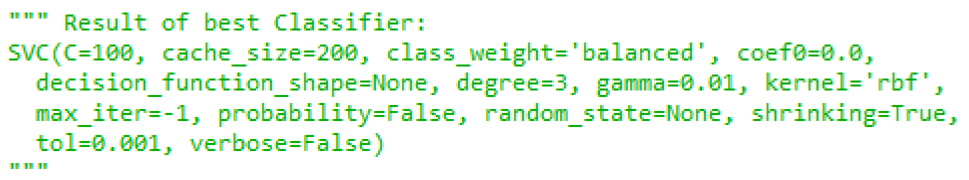
4 Tuning Paramerter for algorithms

4.1 How did model performance differ between algorithms?

Like shown in the previous chapter there are algorithms like SVM, Decision Tree or Adaboost that can be tuned by changing parameters that fits the data best. In case of the Naive Bayes which I have chosen it is not possible to tune them with something like GridSearchCV.

If you have an algorithm like SVM, Decision Tree or something like this and you are not tuning the data it can be that the Accuracy is to low or the Accuracy is close to 1, which can mean that it overfits.

In case of the SVM I tested it first with the standard classifier: SVC(). GridSearchCV throws back the following Classifier which is the best:



```
""" Result of best Classifier:
SVC(C=100, cache_size=200, class_weight='balanced', coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.01, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
"""
```

5 Validation

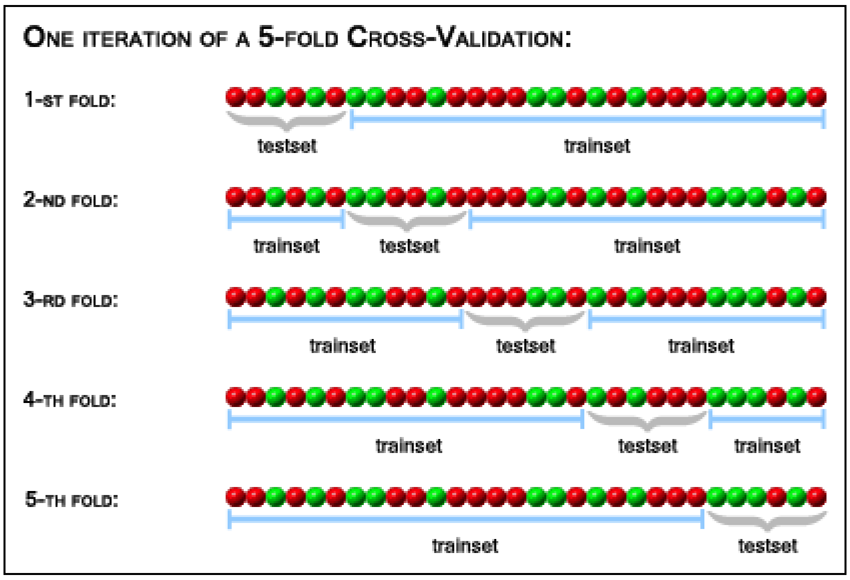
5.1 What is Validation?

Validation describes the process to estimate the performance of the test dataset which will be used for machine learning algorithm. It can also prevent overfitting. This means with a good performed machine learning algorithm the result shouldn't be overfittet. If there is an overfitted dataset and you change the parameters, the result will be better.

Splitting the data into test and train is a typical technique, which finds the optimal splitting for a good result of a machine learning algorithm.

Typicall algorithms are the train test split Cross validation and KFolds. Train-test-split, splits the data in a test and a training set.

K-fold iterates over the whole dataset and run an analysis on each defined fold, and returns an average of all the errors.



https://stats.stackexchange.com/questions/1826/cross-validation-in-plain-english

5.2 Mistakes

A typical mistake can be to **not** drop outlier, like the "Total" in the dataset. It sums up all the features over all persons, so the feature values for total are extremely high.

In case of SVM it can be possible that you overfit when you use a high variance when the number of samples is smaller than the number of features.

Information from:

https://www.kdnuggets.com/2015/03/machine-learning-data-science-common-mistakes.html

5.3 How is the fraud analyses validated?

For validation I used the Stratified Shuffle split, which creates splits by preserving the same percentage for each target class as in the complete set.

http://scikit-learn.org/stable/modules/cross_validation.html

“Stratification works by keeping the balance or ratio between labels/targets of the dataset. So if your whole dataset has two labels (e.g. Positive and Negative) and these have a 30/70 ratio, and you split in 10 subsamples, each stratified subsample should keep the same ratio. Reasoning: Because machine-learned model performance in general are very sensitive to a sample balancing, using this strategy often makes the models more stable for subsamples.”

https://stackoverflow.com/questions/37635460/stratifiedkfold-vs-stratifiedshufflesplit-vs-stratifiedkfold-shuffle

6 Evaluation metrics

6.1 Two evaluation metrics

First I will describe you the evaluation metrics for SVM because here is a change of the different scores visible.

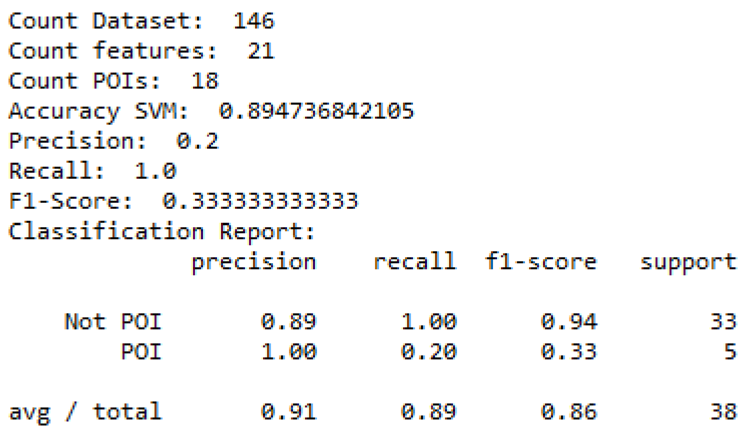
The typical scores which are included is precision, recall, f1-score and support.

The following classification reports show the scores for a POI or not POI. Support gives the count of people. In case of SVM we have 38 non POIs and 5 POIs. The 5 POIs are the employees where we interested in, because after running the learning algorithm these are the employees which were part of the fraud.

Precision and Recall gives information about correct and not correct classified samples. The calculation is:

Recall: True Positive / (True Positive + False Negative).
Recall returns the values of how many POIs are correctly identified.

Precision: True Positive / (True Positive + False Positive)
Precision shows how many of the found POI were correct.



```
Count Dataset:  146
Count features:  21
Count POIs:  18
Accuracy SVM:  0.894736842105
Precision:  0.2
Recall:  1.0
F1-Score:  0.333333333333
Classification Report:
             precision    recall  f1-score   support

    Not POI       0.89      1.00      0.94        33
        POI       1.00      0.20      0.33         5

avg / total       0.91      0.89      0.86        38
```

```
Count Dataset:  146
Count features:  21
Count POIs:  18
Accuracy SVM:  0.868421052632
Precision:  0.4
Recall:  0.5
F1-Score:  0.444444444444
Classification Report:
             precision    recall  f1-score   support

    Not POI       0.91      0.94      0.93        33
        POI       0.50      0.40      0.44         5

avg / total       0.86      0.87      0.86        38
```

As we can see precision, f1 increases after tuning the parameters for the SVM-Classifier. Recall and Accuracy decrease.

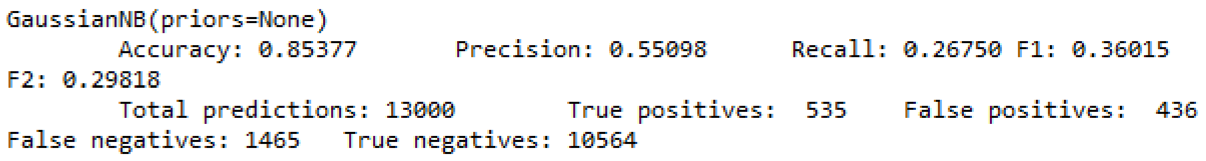
Here is the Classification Report for the chosen algorithm Naive Bays, where the results in all cases are better than in SVM.

```
Count Dataset:  146
Count features:  21
Count POIs:  18
Accuracy Naive Bayes:  0.921052631579
Precision:  0.6
Recall:  0.75
F1-Score:  0.666666666667
Classification Report:
             precision    recall  f1-score   support

    Not POI       0.94      0.97      0.96        33
        POI       0.75      0.60      0.67         5

avg / total       0.92      0.92      0.92        38
```

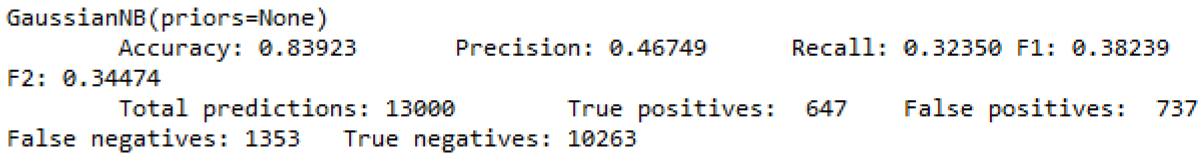
After running test.py I saw that the value for recall is to low:



```
GaussianNB(priors=None)
        Accuracy: 0.85377       Precision: 0.55098      Recall: 0.26750 F1: 0.36015
F2: 0.29818
        Total predictions: 13000        True positives:  535    False positives:  436
False negatives: 1465   True negatives: 10564
```

It should be higher than 0.3.

By adding two new features, total_stock_value and bonus, the recall grow up



```
GaussianNB(priors=None)
        Accuracy: 0.83923       Precision: 0.46749      Recall: 0.32350 F1: 0.38239
F2: 0.34474
        Total predictions: 13000        True positives:  647    False positives:  737
False negatives: 1353   True negatives: 10263
```

: