# Introduction

This document describes the different steps and thought when performing the P5 project.

You can customize the way the poi\_id.py script behave by settings some Boolean variables.

* showunivariate: will display histogram chart for all features if set to True
* showheatmap: will display the correlation heatmap if set to True
* showcorrelation: will display correlation scatter plot between features if set to True
* performalltunings: will perform the algorithm tuning is set to True

By default, all these parameters are set to True.

# Descriptive statistics, outliers mgt, features selection

We use all features and retrieve associated descriptive statistics

salary

count 95.000000

mean 562194.294737

std 2716369.154553

min 477.000000

25% 211816.000000

50% 259996.000000

75% 312117.000000

max 26704229.000000

If I look to the maximum value of salary, I see it corresponds to the total value of salaries from the enron61702insiderpay.pdf. I will delete the TOTAL values and re-compute descriptive statistics.

salary bonus long\_term\_incentive deferred\_income \

count 94.000000 81.000000 65.000000 48.000000

mean 284087.542553 1201773.074074 746491.200000 -581049.812500

std 177131.115377 1441679.438330 862917.421568 942076.402972

min 477.000000 70000.000000 69223.000000 -3504386.000000

25% 211802.000000 425000.000000 275000.000000 -611209.250000

50% 258741.000000 750000.000000 422158.000000 -151927.000000

75% 308606.500000 1200000.000000 831809.000000 -37926.000000

max 1111258.000000 8000000.000000 5145434.000000 -833.000000

deferral\_payments loan\_advances other expenses \

count 38.000000 3.000000 92.000000 94.000000

mean 841602.526316 27975000.000000 465276.663043 54192.010638

std 1289322.626180 46382560.030684 1389719.064851 46108.377454

min -102500.000000 400000.000000 2.000000 148.000000

25% 79644.500000 1200000.000000 1209.000000 22479.000000

50% 221063.500000 2000000.000000 51984.500000 46547.500000

75% 867211.250000 41762500.000000 357577.250000 78408.500000

max 6426990.000000 81525000.000000 10359729.000000 228763.000000

director\_fees total\_payments exercised\_stock\_options \

count 16.000000 1.240000e+02 101.000000

mean 89822.875000 2.623421e+06 2959559.257426

std 41112.700735 9.488106e+06 5499449.598994

min 3285.000000 1.480000e+02 3285.000000

25% 83674.500000 3.863802e+05 506765.000000

50% 106164.500000 1.100246e+06 1297049.000000

75% 112815.000000 2.084663e+06 2542813.000000

max 137864.000000 1.035598e+08 34348384.000000

restricted\_stock restricted\_stock\_deferred total\_stock\_value \

count 109.000000 17.000000 125.000000

mean 1147424.091743 621892.823529 3352073.024000

std 2249770.356903 3845528.349509 6532883.097201

min -2604490.000000 -1787380.000000 -44093.000000

25% 252055.000000 -329825.000000 494136.000000

50% 441096.000000 -140264.000000 1095040.000000

75% 985032.000000 -72419.000000 2606763.000000

max 14761694.000000 15456290.000000 49110078.000000

to\_messages from\_poi\_to\_this\_person from\_messages \

count 86.000000 86.000000 86.000000

mean 2073.860465 64.895349 608.790698

std 2582.700981 86.979244 1841.033949

min 57.000000 0.000000 12.000000

25% 541.250000 10.000000 22.750000

50% 1211.000000 35.000000 41.000000

75% 2634.750000 72.250000 145.500000

max 15149.000000 528.000000 14368.000000

from\_this\_person\_to\_poi shared\_receipt\_with\_poi poi

count 86.000000 86.000000 145.000000

mean 41.232558 1176.465116 0.124138

std 100.073111 1178.317641 0.330882

min 0.000000 2.000000 0.000000

25% 1.000000 249.750000 0.000000

50% 8.000000 740.500000 0.000000

75% 24.750000 1888.250000 0.000000

max 609.000000 5521.000000 1.000000

I can see that:

* There are 145 peoples in the dataset (there are 18 POIs within this set of 145 peoples).
* Only the POI feature has been fully set (all other feature have missing values)
* Emails data is filled for only 86 peoples.
* The deferral\_payments, restricted\_stock, restricted\_stock\_deferred , total\_stock\_value have positive values and negative values in it.
* There are a very small number of information for features loan\_advances, director\_fees and restricted\_stock\_deferred.

As a consequence, I decide to:

* Remove features loan\_advances, director\_fees and restricted\_stock\_deferred from the dataset and analysis.
* Delete negative values for features: deferral\_payments, restricted\_stock, total\_stock\_value

I decide then to have a look on data distribution using a histogram chart.

I see that a lot of feature distributions are skewed: bonus, long\_term\_incentive, deferred\_income, deferral\_payments, other, expenses, total\_payments, exercised\_stock\_options, restricted\_stock, total\_stock\_value, to\_messages, from\_poi\_to\_this\_person, from\_messages, from\_this\_person\_to\_poi, shared\_receipt\_with\_poi.

In the machine learning algorithm, I will check if applying a log transformation on these feature values helps.

I can also find some outliers values, but I decide to keep these values as they are real values.

I now want to have a look on possible correlated data in my dataset.

We can find some strong correlations, for instance between deferral\_payments and deferred\_income, total\_stock\_value and restricted\_stock, to\_message and from\_this\_person\_to\_poi, to\_message and shared\_receipt\_with\_poi.

On the other hand, I notice there is not strong between poi and all other variables.

When looking to the variables scatter plots, I decided not to perform any data transformation based on these correlations. I prefer to rely on the principal component algorithm with prior Min/Max scaling.

When looking to the correlation matrix, we identify 3 groups of data:

* Salary related information
* Stock related information
* Email related information.

When doing the machine learning algorithm tweak, I will evaluate performance using:

* All selected features
* Only the total\_payments, total\_stock\_value and all emails features.

Missing values will be replaced with 0. I think that other standard strategies (Mean, Median, Most frequent) proposed in sklearn are not relevant in our case.

# Additional features creation

We can reasonably think that email exchange between POIs may be greater than between POIs and non POIs.

Instead of working with absolute values, I decide to create two additional ratios:

* poi\_to\_ratio = from\_poi\_to\_this\_person / to\_messages
* poi\_from\_ratio = from\_this\_person\_to\_poi / from\_messages

If we look to correlation, I cannot see a real correlation between POIs and these 2 new ratios.

poi\_to\_ratio poi\_from\_ratio poi

poi\_to\_ratio 1.000000 0.245350 0.059688

poi\_from\_ratio 0.245350 1.000000 0.312483

poi 0.059688 0.312483 1.000000

# Machine learning algorithm selection and tuning

I will perform a benchmarking analysis using:

* 3 differents supervised classification machine learning algorithm (Naïve Bayes, Support Vector Machine, Decision Tree).
* Each algorithm will be preceded by a scaling activity (Min/Max) and a principal component anlaysis.
* 3 differents datasets: all features except new ratios, all features including ratios, restricted features excluding ratio. Each time, we will evaluate the algorithm with unchanged information and log changed information.
* 2 differents validation strategies: use of a basic sample of the data set, use of a KFold sets of the data set.
* For SVM algorithm and decision tree algorithms, a random selection based tuning will be used to limit computer CPU usage.

For the PCA algorithm, the number of final component is part of the tuning activity.

For the SVM algorithm, the kernel, C and gamma parameters are part of the tuning activity. The possible values will be:

* C=[1,10,100,1000],
* gamma=[0.01,0.001, 0.0001],
* kernel=['rbf','linear','poly']

For the decision tree algorithm, the criterion and max features parameters are part of the tuning activity. The possible values will be:

* criterion=['gini','entropy'],
* max\_features=['sqrt','log2',None])

For each of this test, we will use the f1 scoring function as value to be optimized. This metrics offer a good balance between precision and recall. We will have a look to precision and recall metrics for the selected algorithm.

We use two f1 computation. The one based on estimator.best\_score\_ value (directly provided by sklearn), and a second computed using a modified version of the test\_classifier function from the tester.py.

The results are provided in the following table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data set | Log values | Val strategy | Algo | F1 (computed) | F1 (best\_score\_) |
| Full | Yes | Basic | Naïve Bayes | 0.31 | 0.39 |
| Full | Yes | Basic | SVM | 0.03 | 0.61 |
| Full | Yes | Basic | Decision tree | 0.27 | 0.61 |
| Full | Yes | KFold | Naïve Bayes | 0.43 | 0.47 |
| Full | Yes | KFold | SVM | 0.19 | 0.42 |
| Full | Yes | KFold | Decision tree | 0.28 | 0.38 |
| Full | No | Basic | Naïve Bayes | 0.19 | 0.46 |
| Full | No | Basic | SVM | 0.20 | 0.60 |
| Full | No | Basic | Decision tree | 0.28 | 0.52 |
| Full | No | KFold | Naïve Bayes | 0.43 | 0.47 |
| Full | No | KFold | SVM | 0.20 | 0.42 |
| Full | No | KFold | Decision tree | 0.29 | 0.32 |
| Full with ratio | Yes | Basic | Naïve Bayes | 0.27 | 0.39 |
| Full with ratio | Yes | Basic | SVM | 0.02 | 0.62 |
| Full with ratio | Yes | Basic | Decision tree | 0.28 | 0.46 |
| Full with ratio | Yes | KFold | Naïve Bayes | 0.27 | 0.4 |
| Full with ratio | Yes | KFold | SVM | 0.02 | 0.47 |
| Full with ratio | Yes | KFold | Decision tree | 0.28 | 0.48 |
| Full with ratio | No | Basic | Naïve Bayes | 0.27 | 0.39 |
| Full with ratio | No | Basic | SVM | 0.02 | 0.62 |
| Full with ratio | No | Basic | Decision tree | 0.28 | 0.42 |
| Full with ratio | No | KFold | Naïve Bayes | 0.27 | 0.4 |
| Full with ratio | No | KFold | SVM | 0.02 | 0.47 |
| Full with ratio | No | KFold | Decision tree | 0.28 | 0.5 |
| Limited | Yes | Basic | Naïve Bayes | 0.37 | 0.35 |
| Limited | Yes | Basic | SVM | 0 | 0 |
| Limited | Yes | Basic | Decision tree | 0.22 | 0.28 |
| Limited | Yes | KFold | Naïve Bayes | 0.35 | 0.32 |
| Limited | Yes | KFold | SVM | 0 | 0 |
| Limited | Yes | KFold | Decision tree | 0.22 | 0.26 |
| Limited | No | Basic | Naïve Bayes | 0.38 | 0.35 |
| Limited | No | Basic | SVM | 0 | 0 |
| Limited | No | Basic | Decision tree | 0.22 | 0.28 |
| Limited | No | KFold | Naïve Bayes | 0.35 | 0.32 |
| Limited | No | KFold | SVM | 0 | 0 |
| Limited | No | KFold | Decision tree | 0.21 | 0.22 |

**Interpretation**

**Validation strategy**

We see that when tuned, the SVM algorithm including all features (+ additional ratios) is the better performance using a basic validation strategy.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data set | Log values | Val strategy | Algo | F1 (computed) | F1 (best\_score\_) |
| Full with ratio | Yes | Basic | SVM | 0.02 | 0.62 |
| Full with ratio | No | Basic | SVM | 0.02 | 0.62 |

But we see that when applying the Udacity validation strategy, this algorithm behave badly.

This is clearly important to perform validation in the way the algorithm will be used in operation condition.

At the end, the best algorithm is not the one providing the best tuned result.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data set | Log values | Val strategy | Algo | F1 (computed) | F1 (best\_score\_) |
| Full | Yes | KFold | Naïve Bayes | 0.43 | 0.47 |
| Full | No | KFold | Naïve Bayes | 0.43 | 0.47 |

The final selected algorithm is a pipeline of:

* Min Max scaler
* PCA with 12 components
* Naïve Bayes classifier

Output from the tester.py script are:

Accuracy: 0.83440 Precision: 0.39702 Recall: 0.46650 F1: 0.42897 F2: 0.45072

Total predictions: 15000

True positives: 933 False positives: 1417 False negatives: 1067 True negatives: 11583

POIs were identified in 47% of the cases, and when identified, we were right in 40% of cases.

http://www.h5.com/document-review-accuracy-the-recall-precision-tradeoff/

Globally, I consider this algorithm a bit weak.

When we run a simple test strategy, we can find results looking like the following one:

precision recall f1-score support

0.0 0.89 1.00 0.94 39

1.0 0.00 0.00 0.00 5

avg / total 0.79 0.89 0.83 44

The algorithm behave pretty good, but it is very efficient identifying non POIs. He is totally week for detecting POIs.

This may come from our initial dataset with a lot of missing values.

As a conclusion, I see that the algorithm validation phase is crucial and has to be well thought.

**Log transformation of feature**

|  |  |  |
| --- | --- | --- |
| **Row Labels** | **Average of F1 (computed)** | **Average of F1 (best\_score\_)** |
| No | 0.216111111 | 0.375555556 |
| Yes | 0.211666667 | 0.383888889 |
| **Grand Total** | **0.213888889** | **0.379722222** |

We see that the log transformation of features values has no real impact on the algorithm performances.

**Inclusion of new features**

|  |  |  |
| --- | --- | --- |
| **Row Labels** | **Average of F1 (computed)** | **Average of F1 (best\_score\_)** |
| Full | 0.258333333 | 0.4725 |
| Full with ratio | 0.19 | 0.468333333 |
| Limited | 0.193333333 | 0.198333333 |
| **Grand Total** | **0.213888889** | **0.379722222** |

We see that inclusion of the two new ratios does not provide a significant impact on the performances of the algorithm. It even tends to decrease it performance.

I applied the PCA algorithm prior to the final machine learning algorithm tuning. If the new ratios do not add new information, they will not be considered (or weakly considered) by PCA algorithm.