A Comparison of Machine Learning Techniques for Credit Card Fraud Detection

Lusis

April 20, 2017

1 Introduction

Fraud is a billion-dollar business and it is increasing every year. The PwC global economic crime survey of 2016 suggests that more than one in three (36%) of organizations experienced economic crime [1]. Those results reveal clearly that, despite the millions of dollars being spent to tackle it, economic crime remains a persistent and serious issue.

In the last years, many studies have been performed using data mining to investigate new techniques to detect fraud on the basis of the fraudulent paths [2] and different algorithms have been developed to block fraudulent transactions before they are filled. However, new fraud behaviors born every time, above all in the Internet world, and for this reason we need a continuous improvement of those algorithms.

In the next sections, we use a simulated sample of $\sim$ 200k credit card transactions to test two machine learning algorithms for fraud detection: Logistic Regression (LR) and Random Forest (RF) [3]. We also compare our results with a previous study of Supelec students that used a Support Vector Machine (SVM) algorithm [4] for the same aim.

2 The Data

In this study, for legals reasons, are not used real banking transactions but data simulated using R. The data contain the 5% of fraudulent transactions. Anyway, simulation is a good reproduction of the reality, basing on reals cases and interviews of specialized people in the fraud sector.

2.1 Initial Data

We use a set of $\sim$ 200k transactions (exactly 193679) realized with 5000 different cards. For each transaction, the recorded quantities are:

- *datetime*: the date and the time of the transaction in a format: $yyyy - mm - dd\ HH : MM : SS$;
- *ref*: unique ID for transaction;
- *card*: Number of credit card used for transaction. In total we have 5000 cards.
- *localamount*: amount of transaction in the local currency;
- *currency*: code ISO of the currency of the country in which transaction has been realized;
- *terminal*: number of the terminal of the transaction;
- *type*: type of the terminal (ATM, POS,...);
- *mcc*: type of merchant;
- *country*: country in which transaction has been realized;

- *riskmerchant*: boolean variable indicating if the place of the transaction is flagged as high-risk fraud country;
- *riskcountry*: boolean variable indicating if the country of the transaction is flagged as high-risk fraud country;
- *riskmcc*: boolean variable indicating if the type of merchant of the transaction is flagged as high-risk fraud country;
- *travel*: boolean variable indicating if the client owner of the card is flagged as traveling;
- *rtfraud*: boolean variable indicating if the transaction is fraudulent and it is detectable as fraudulent when it has been realized. When it is flagged to 1, the card is blocked and all the following transactions are forbidden.
- *fraud*: boolean variable indicating if the transaction is fraudulent (detection a posteriori). It is flagged after a transaction is flagged as fraudulent (rtfraud = 1), and it goes back to all the last transaction with a fraudulent behavior. Fraud is the variable to predict with the machine learning algorithm.

Listing 1: Initial Data

	datetime	ref	card	localamount	currency	terminal	type	mcc
1	2016-01-01 01:13:31	52524	2168	160	GBP	13753	ATM	0
2	2016-01-01 01:26:45	102718	305	110	SEK	10646	ATM	0
3	2016-01-01 01:50:19	11113	1518	60	EUR	7801	ATM	0
4	2016-01-01 02:15:00	83082	4573	90	DKK	7385	ATM	0
5	2016-01-01 02:33:16	103508	1864	70	CAD	13206	ATM	0
6	2016-01-01 02:41:34	15489	2746	110	DKK	8335	ATM	0

	country	riskmerchant	riskcountry	riskmcc	travel	fraud	rtfraud
1	United Kingdom	0	1	0	0	0	0
2	Sweden	0	0	0	0	0	0
3	France	0	1	0	0	0	0
4	Denmark	0	0	0	0	0	0
5	Canada	0	1	0	0	0	0
6	Denmark	0	0	0	0	0	0

2.2 Derivate Data

The initial attributes are not really useful for the machine learning, because they are not discriminating factors for the fraud detection. For this reason, starting from them, we compute new derivate variables taking in account the history of each credit card. For each transaction, the new quantities are:

- *USDamount*: the *localamount* converted to USD;
- *Diff_time_trans*: time difference in minutes between two following transactions;
- *NFreq_daily*: total number of transactions in a day till this transaction;
- *NCountry_daily*: total number of Country in which transactions are realized in a day till this transaction;
- *tot_daily_amount*: total amount in USD of all transactions in the same day till this transaction;
- *NFreq_weekly*: average of number of transactions for day over 7 days before this transaction. We calculate the total number of transactions with a credit card during the past 7 days prior to transaction and dividet it by 7.
- *NCountry_weekly*: total number of Country in which transactions are realized over 7 days before this transaction;

- *tot_weekly_amount*: total amount of transactions over 7 days before this transaction divided by 7 to compute a daily average;
- *NFreq_monthly*: average of number of transactions for day over 30 days before this transaction. We calculate the total number of transactions with a credit card during the past 30 days prior to transaction and dividet it by 30.
- *NCountry_monthly*: total number of Country in which transactions are realized over 30 days before this transaction;
- *tot_monthly_amount*: total amount of transactions over 30 days before this transaction divided by 7 to compute a daily average;
- *day_night*: flag to identify if transaction has been realized during the night ($day\_night = 1$) after 22 PM and before 4 AM or during the day ($day\_night = 0$);
- *buss_weekend*: flag to identify if transaction has been realized during the weekend ($buss\_weekend = 1$) or during a business day ($buss\_weekend = 0$);
- *fraud*: as the initial data.

Listing 2: Derivate Data

	USDamount	Diff_**time**_trans	NFreq_daily	NCountry_daily	tot_daily_amount
2	162.305	4258.900	1	1	162.305
3	199.760	2694.250	1	1	199.760
4	237.215	1710.383	1	1	237.215
5	49.940	6842.683	1	1	49.940
6	137.335	6760.083	1	1	137.335
7	112.365	7969.650	1	1	112.365

	NFreq_weekly	NCountry_weekly	tot_weekly_amount	NFreq_monthly
2	0.0000000	0	0.00000	0
3	0.0000000	0	0.00000	0
4	0.0000000	0	0.00000	0
5	0.4285714	1	69.55929	0
6	0.2857143	1	26.75357	0
7	0.2857143	1	35.67143	0

	NCountry_monthly	tot_monthly_amount	day_night	buss_weekend	fraud
2	0	0	1	0	0
3	0	0	0	0	0
4	0	0	1	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	1	1	0

For weekly and monthly values, if a transaction has not enough previous days to quantify the attribute, we chose to set it to 0.0.

Since this moment, we will refer to the derivate data for our study.

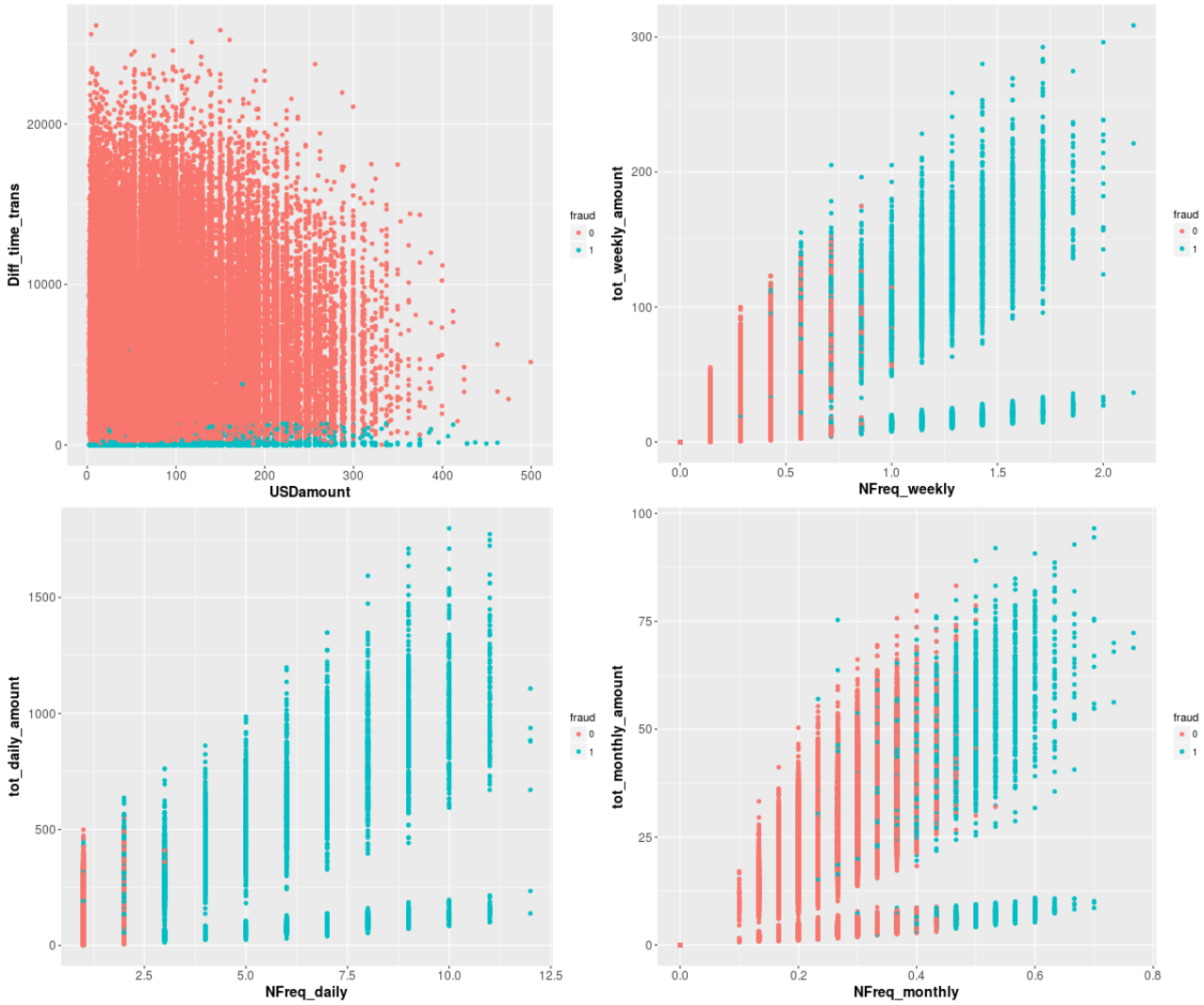
3 Exploring Data

3.1 Zero covariates variables

Before to apply the machine learning algorithms, we explore our data. First af all, we check if there is any "zero covariates" variable. They are variables with low variability and we remove them because they will not be good predictors. In our dataset *NCountry_daily* and *NCountry_weekly* are "zero covariates" variables so, we neglect them from our analysis. For coherence, we chose to remove *NCountry_monthly* too.

3.2 Relations between variables

We explore our data to have a first idea about correlations between the variables. We chose to plot *USDamount* versus *Diff_time_trans*, *NFreq_daily* versus *tot_daily_amount*, *NFreq_weekly* versus *tot_weekly_amount*, *NFreq_monthly* versus *tot_monthly_amount*. Different colors are associated to a different fraud flag (pink =0, blue=1)



Looking at the plots, we can see that, for example, fraud transactions happened in a really short time. Furthermore, higher frequency of transaction are in general associated to fraudulent transactions for all intervals of reference (daily, weekly and monthly).

3.3 Summary properties

Before to apply our machine learning algorithms, is very useful to take a look at the summary properties of our data:

Listing 3: Summary Properties

```
USDamount         Diff_time_trans      NFreq_daily      NCountry_daily
Min.   :   2.25   Min.   :    2.05   Min.   : 1.000   Min.   :1
1st Qu.:  30.00   1st Qu.: 2809.85   1st Qu.: 1.000   1st Qu.:1
Median :  74.86   Median : 5071.22   Median : 1.000   Median :1
Mean   :  82.84   Mean   : 5696.37   Mean   : 1.214   Mean   :1
```

```
 3rd Qu.:120.00    3rd Qu.: 7964.14    3rd Qu.: 1.000    3rd Qu.:1
 Max.   :499.40    Max.   :41800.37    Max.   :12.000    Max.   :1
 tot_daily_amount    NFreq_weekly     NCountry_weekly   tot_weekly_amount
 Min.   :   2.25    Min.   :0.0000    Min.   :0.0000    Min.   :  0.000
 1st Qu.:  32.08    1st Qu.:0.2857    1st Qu.:1.0000    1st Qu.:  7.811
 Median :  76.24    Median :0.2857    Median :1.0000    Median : 26.754
 Mean   : 100.21    Mean   :0.3679    Mean   :0.9651    Mean   : 30.315
 3rd Qu.: 128.34    3rd Qu.:0.4286    3rd Qu.:1.0000    3rd Qu.: 42.857
 Max.   :1797.84    Max.   :2.1429    Max.   :1.0000    Max.   :308.558
 NFreq_monthly     NCountry_monthly  tot_monthly_amount    day_night
 Min.   :0.0000    Min.   :0.0000    Min.   :  0.000      Min.   :0.0000
 1st Qu.:0.2000    1st Qu.:1.0000    1st Qu.:  3.075      1st Qu.:0.0000
 Median :0.2333    Median :1.0000    Median : 19.976      Median :0.0000
 Mean   :0.2217    Mean   :0.8185    Mean   : 18.238      Mean   :0.2582
 3rd Qu.:0.3000    3rd Qu.:1.0000    3rd Qu.: 29.132      3rd Qu.:1.0000
 Max.   :0.7667    Max.   :1.0000    Max.   : 96.551      Max.   :1.0000
  buss_weekend     fraud
 Min.   :0.0000    0:178938
 1st Qu.:0.0000    1:  9741
 Median :0.0000
 Mean   :0.2853
 3rd Qu.:1.0000
 Max.   :1.0000
```

We can see that in general our variables cover different and, often, very large ranges. In this case, above all to perform a logistic regression, it can be useful to preprocess the data and apply transformation to make them homogeneous. We will see in the next sections how to treat this issue.

Finally we remove transactions having NA value: in general they come from the first element of *Diff_time_trans*.

4 Experimental Setup

4.1 Training and Data set

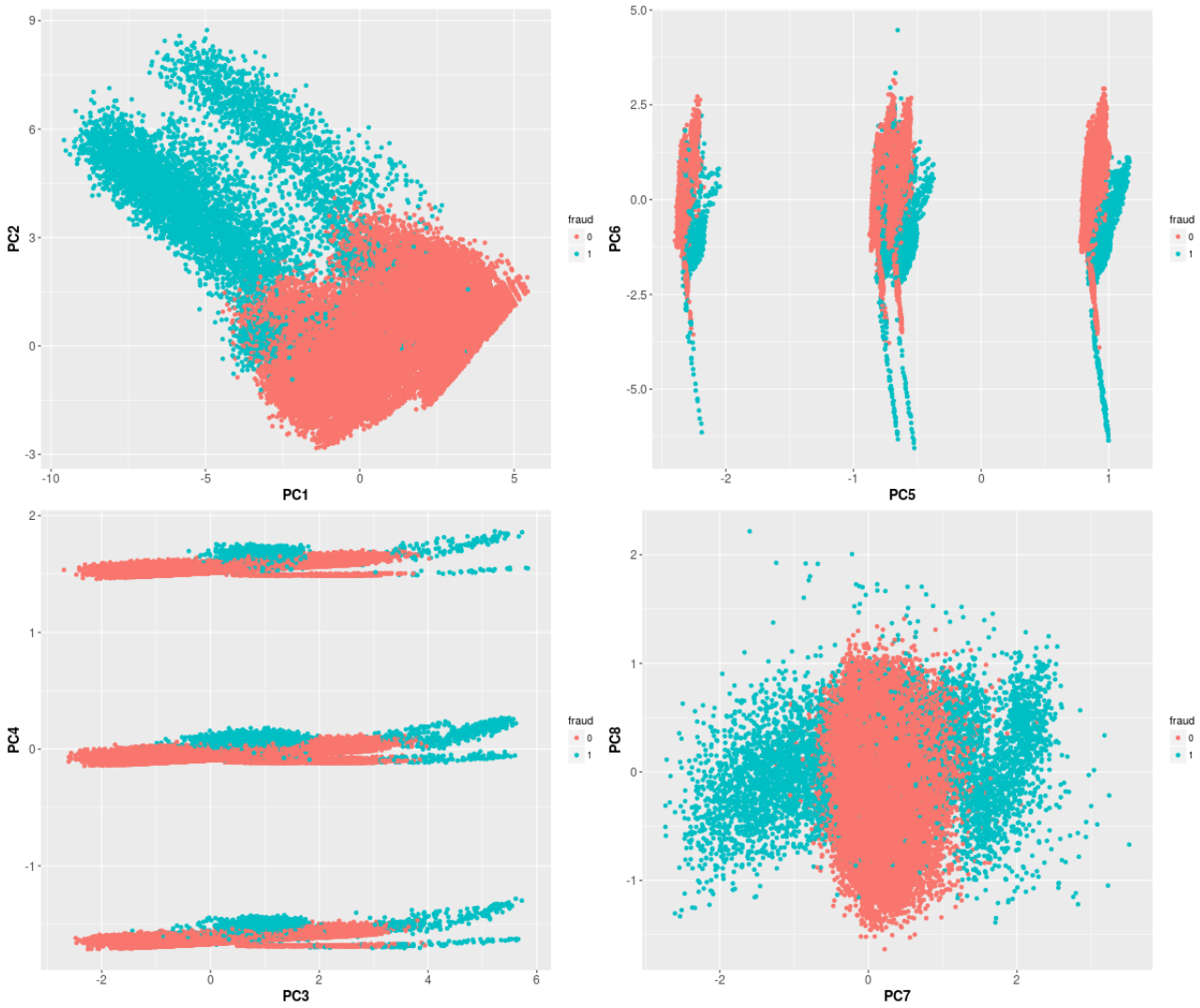
We split our starting sample in training and test data. We use the R function **createDataPartition** of *caret* package [5], using a probability of 0.7. The training data contains 132076 transactions. The set data contains 56603 transactions. Both subsamples have $\sim 5\%$ of fraud transactions.

4.2 PCA Preprocess

When data have a standard deviation too much larger than mean, it could be useful to apply a transformation to reduce it. Furthermore, when there are many variables, we can built a new set of uncorrelated quantities that explain as better as possible the variance. This is can be done, applying a **Principal Component Analysis** (PCA) [3]. The result of this analysis is that the number of principal components is less than or equal to the smaller of the number of original variables or the number of observations. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set.

This transformation is applied to the training set. The same parameters of transformations computed over training set, are then applied to the test set.

We chose to apply to our data a logarithmic transformation and a PCA analysis with a number of component = 10 equal to the starting ones. We use the function **preProcess** of *caret* pakcage. In the following plot, we show the relations betweens the first 8 components. Different colors are still associated to a different fraud flag (pink =0, blue=1):



We can see, above all in the first plot, a good distinction between fraud and no fraud transactions.

We will apply the machine learning algorithms to both data with and without PCA, to check the impact of preprocessing on the results.

5 Machine learning algorithms and Results

5.1 Logistic Regression

The first machine learning algorithm we want to apply to our data is the logistic regression. In statistics, logistic regression, or logit regression, or logit model is a regression model where the dependent variable (DV) is categorical. The typical use of this model is predicting y given a set of predictors x. The predictors can be continuous, categorical or a mix of both.

The categorical variable y, in general, can assume different values. In the simplest case scenario y is binary meaning that it can assume either the value 1 or 0.

For a detailed mathematical description of this model, we refer to the section 4.4 of [3].

In R, to perform a logistic regression we use the **glm** function.

5.2 Random Forest

The second machine learning to test is the Random Forest.

Random Forests are one way to improve the performance of decision trees. The algorithm starts by building out trees similar to the way a normal decision tree algorithm works. However, every time a split has to made, it uses only a small random subset of features to make the split instead of the full set of features (usually $\sqrt{p}$, where p is the number of predictors). It builds multiple trees using the same process, and then takes the average of all the trees to arrive at the final model. This works by reducing the amount of correlation between trees, and thus helping reduce the variance of the final tree.

For a detailed mathematical description of this model, we refer to the chapter 15 of [3].

In R, to apply Random Forest we use the **train** function with method = 'rf', using a K-fold cross validation with k = 3 and k = 6 (for details about cross validation, see section 7.10 of [3]).

5.3 Results

We apply the Logistic Regression and Random Forest (with K=3 and K=6) to both training dataset with and without PCA Processing. After building the model, we apply it to the respective test data set and we compute the Confusion Matrix, containing starting from up-left cell in clockwise the values: True Positive TP, false Positive FP,True Negative TN and and False Negative FN, and the Performance Parameters like:

- Accuracy: (TP+TN)/(TP+FP+TN+FN)
- Sensitivity: TP/(TP+FN)
- Specificity: TN / (FN+TN)
- Positive Prediction value: TP/(TP+FP)
- Negative Prediction Value: TN/(FN+TN).

We show below the Confusion Matrix for our 6 models. We also make comparison with the results obtained by the Supelec students that used a Support Vector Machine algorithm on the same data of us to detect fraud. We have to specify that, even if the starting data are the same, the procedure executed by Supelec is different from us for the computation of the derivate variables and the preprocessing. In particular, their method brings to use a smaller final sample of data where fraud transactions are *sim* 30% of the total sample.

Table 1: Confusion Matrix

Pred \ Ref	0	1
0	55513	158
1	205	2717

(a) LR without PCA

Pred \ Ref	0	1
0	53587	94
1	106	2816

(b) LR with PCA

Pred \ Ref	0	1
0	53605	76
1	52	2870

(c) RF without PCA and K=3

Pred \ Ref	0	1
0	53603	78
1	70	2852

(d) RF with PCA and K=3

Pred \ Ref	0	1
0	53604	77
1	57	2865

(e) RF without PCA and K=6

Pred \ Ref	0	1
0	53606	75
1	68	2854

(f) RF with PCA and K=6

Pred \ Ref	0	1
0	7353	198
1	81	3191

(g) SVM Supelec Study

We summarize below the Performance Parameters of the different algorithms:

Table 2: Performance Parameters sorted by Accuracy

Algorithm	Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value
RF without PCA and K=3	0.9977	0.9990	0.9742	0.9986	0.9822
RF without PCA and K=6	0.9976	0.9989	0.9739	0.9986	0.9805
RF with PCA and K=6	0.9975	0.9987	0.9744	0.9986	0.9767
RF with PCA and K=3	0.9974	0.9987	0.9734	0.9985	0.9760
LR with PCA	0.9965	0.9980	0.9677	0.9982	0.9637
LR without PCA	0.9934	0.9962	0.9418	0.9969	0.9298
SVM Supelec Study	0.9742	0.9891	0.9416	0.9738	0.9752

In the next section we discuss our results.

6 Discussion and Conclusion

Given the Confusion Matrix and Performance Parameters, we can investigate which is the best Algorithm to predict fraud for our data.

First of all we focus on Logistic regression without PCA. We can see that Accuracy of 0.9934 and all others parameters show well that this algorithm is better than Support Vector Machine that has an accuracy of 0.9742. Furthermore, we notice that to apply a Preprocessing to our data still improve the results, reaching an accuracy of 0.9965.

However, the Random Forest algorithm has a better performance than both SVM and LR. In general, we see that we do not need to Preprocess the data to apply Random Forest: indeed the RF algorithm does not really suffer from high number of predictors since it only takes a random subset of them to build each tree.

We can also notice that the number of fold for cross validation it does not really affect results: this method in fact, has in general more importance when the sample is small.

In conclusion, after a comparison of three different algorithms, Random Forest, Logistic Regression and Support Vector Machine, in different conditions (with or without Principal Component Analysis, K for cross validation equal to 3 or 6), for a total of 7 machine learning procedures, we can affirm that for our data for the detection of fraud transactions the Random Forest without PCA and K = 3 has the best performance with an accuracy of 0.9977, sensitivity of 0.9990 and specificity of 0.9742.

We must remind that the algorithms are applied to simulated data and so it could be useful to evaluate performance on real data. Furthermore, it's very significant the number and the type of derivate variables used for machine learning: increasing it and/or choosing different attributes could improve results. However a larger number of variables can affect critically the time of execution of the scripts. In our case, a full test (computation of derivate variables, PCA and building of model) for a single algorithm takes more than 3 hours, so also an optimization of the functions could help to gain computation time and make faster test over larger sample with more predictors.

References

[1] PricewaterhouseCoopers LLP (2016). "2016 Global Economic Crime Survey". Retrieved March 6, 2017.

[2] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J. Christopher Westland. Data mining for credit card fraud: A comparative study. Decision Support Systems, 50(3):602-613, 2011. On quantitative methods for detection of financial fraud.

[3] Hastie, Trevor, Trevor Hastie, Robert Tibshirani, and J. H. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York: Springer, 2001.

[4] Hlne Boudon, Thomas Fromont, Jean-Sbastien Renaud. Rapport CEI: Dtection de fraudes bancaires, 2016

[5] Max Kuhn, R Package caret: https://cran.r-project.org/web/packages/caret/caret.pdf