**Abstract**

According to Nilson Report (https://nilsonreport.com) the credit card frauds cost business and card issuers around 28 billion dollars in 2018. As we are moving into a cashless society this number is going to grow steadily. When talking about fraudulent transactions, most people think about the values defrauded. However, the total cost of the frauds has to be understood in its whole extension:

1) There is the loss due to the fraud itself (what most people see)

2) There is the cost associated with managing the losses – cancelling orders and refunding

charges (exceeds the cost of the frauds 300%)

3) There is the cost of mistakenly rejecting orders

4) There is the cost of developing and applying mechanisms to avoid fraud.

Therefore, the problem is not only what you lose due the frauds themselves, but the cost to mitigate the negative impact and manage the process (update security measures, block and reissue cards, reimburse customers etc), the cost of building and maintaining mechanisms to prevent the frauds and of course the cost of losing money when you unduly block a sales assuming wrongly it is a fraud.

Therefore, many financial organizations are facing the challenge of building a successful fraud detection model which is easy to maintain and highly effective in spotting frauds and at the same time doesn´t have a too high false-positive rate. Certain ML algorithms seem to be well suited to address all these issues. They can help automatize the process of adjusting for identifying new types of fraud (the big problem with the current processes) and it can archive an effective identification rate without too many false-negative.

**Problem statement /Business problem:**

Our objective is to spot possible frauds in credit card operations. This identification will be based on the client´s profile, the seller´s profile and the data of the transaction itself. The objective is to flag transactions with the high possibility of fraud.

Today frauds spin around 4% of all sales made with credit cards in Brazil. Our client already has mechanisms in place that detects potential fraudulent transactions, however, these mechanisms have two problems:

1) They deploy semi-static rules that generate a scenario where people in the background have to keep looking for new types of fraud to keep adjusting the current model.

2) The process suffers a paradoxical problem: if it is too stringent it creates problems for the clients blocking legit sales if it is too lose it allows a too high level of fraud. A middle term is difficult to archive – even more so when you have to keep adjusting the rules.

To address these two issues, the idea would be to create a model that would not only identify (or at least flag) the suspect transactions but also would identify changes in the patterns and adapt automatically to new fraud patterns.

In order to do that we managed to get a database merging the three data sources (client, seller and transaction) and the idea is to develop a machine learning model which not only assertive (Assertive meaning identify a high percentage of the actual frauds without blocking too many legit ones) but adaptable.

We were verbally informed that the current mechanism spots around 50% (2% of the total) of potential frauds and flags around 2% of legit ones (false positive). If that is true (We have no way to verify this information), it means that the current process gets it right at around 2% of all transactions and wrong at 2%. In summary, it can stop 50% of all fraudulent transactions and has a false positive rate of 2%.

It is interesting to notice that although frauds correspond to just 4% of all transactions, they answer for 8% of the total value of the transactions. It means each 1% of fraud elimination corresponds to approximately R$ 16.000.000,00 /month (CAD 5.330.000,00).

**Datasets/Getting the data**

We used a real anonymized and sanitized dataset representing a subset of the transactions that occurred during one day of September of 2019 in Brazil, totalling 290.398 transactions.

**Data exploration/Evaluating the data/ Cleaning the data**

Accuracy and completeness: A visual inspection showed that the data was basically correct

**Stan Insert here**

The conclusion of our analysis showed that the factors affecting the chance of fraud have much more to do with the type of the transaction, how far from the average the value of the transaction is and the income of the buyer (wealth people tend to have fewer frauds – better financial education?). And only after that, we have the score of the seller.

Note that the only personal parameter which seems to affect the level of fraud is the income (Age and sex doesn´t seem to have any significant impact). Now with or database adjusted we tested three possible strategies:

**Selecting the training data and the test data**

The idea is to use 95% of the measurements as our training data and 5% as our test data. We had a database with 290.398 clients and we separated it into training data 275.878 and test data 14.520.

**The meaning of the columns (data dictionary)**

The database contains three sources:

* Clients data
* Sellers data
* And transactional data.

Information regarding the clients:

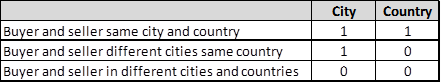
* Age
* Sex
* Income
* Average expenditure  in recurrent  payments  per transaction
* Average expenditure  buying goods (Using the card directly) per transaction
* Average expenditure buying services (using the card directly) per transaction
* Average expenditure buying goods on-line per transaction
* Average expenditure buying services on-line per transaction

Information about the seller:

* Score of the seller (Number 0 to 100) indicating the ranking of the seller regards frequency of frauds.

Information about the transactions:

* Type: which one of the five categories it belongs (0-recurrent, 1- goods, 2-services, 3-online goods or 4-online services).
* Value of the transaction
* If the addresses of the seller and the buyer  are in the same city
* If the addresses of the seller and the buyer  are in the same country
* If the transaction was fraudulent or not (Y or N)



**Approach /Analytical problem**

Initially, we realized that the variable value only has a meaning as long it is compared with the average expenditure with this particular type of transaction. Given this fact, we realized that it would be necessary to create a synthetic variable called “Dispersion”. Dispersion is created comparing the value of the transaction and its type with the average expenditure with this particular type. Example: If the value is equal R$ 20,00 and type equal 0. That means this is a recurrent transaction. The average expenditure with recurrent transactions is R$ 25,00. In that case “Dispersion” will be: absolute (25-20)/25 -> 0.20. This variable gives us a view of how far from the average the value of the transaction.

In sequence, we evaluated if we could deploy the method decision-tree directly to identify the frauds. The results were a bit disappointing, although the accuracy seems to be good it happens just because the data is imbalanced, and the frauds are 4% of the total.

The objective is to actually say if the transaction is a fraud or not (not just calculate the chance it to be). Note it identified correctly only 26% of the frauds (157 out of 588), although with a very low false positive (0,2%).

Confusion Matrix and Statistics

          Reference

Prediction     N     Y

         N 13898   431

         Y    34   157

               Accuracy : 0.968

                 95% CI : (0.965, 0.9708)

    No Information Rate : 0.9595

    P-Value [Acc > NIR] : 4.669e-08

                  Kappa : 0.391

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9976

            Specificity : 0.2670

         Pos Pred Value : 0.9699

         Neg Pred Value : 0.8220

             Prevalence : 0.9595

         Detection Rate : 0.9572

   Detection Prevalence : 0.9868

      Balanced Accuracy : 0.6323

However, the method indicated how many clusters would be ideal (74) and also gave us an idea about how to proceed. The tree also discovered an interesting pattern: The personal data has almost nothing to do with the results. The relevant factors identified were:

* Score of the seller – Indicates the seller is often target for frauds (probably due type of service or lack of internal controls).
* The type of transaction – Internet services have a way more frauds than everybody else
* City – Transactions where the buyer and the seller are in different cities have much more frauds (they tend to overlap with internet services sales).
* Dispersion (How far from the average spent with this kind of item the value of the operation is)

Then, we decided to create a second synthetic variable (V11). To get to the V11 we grouped the transactions using k-means and checked the % of the frauds in each cluster. In sequence, we verified to which cluster the transaction belongs and loaded this % into V11.

In the sequence when we grouped the transactions by clusters we noticed that there were direct and inverse correlations between some of the variables and the frequency of occurrence of frauds. Analyzing it further we realized that we could measure this correlation in a variable. Therefore we created a third synthetic variable which we called “Points”

The correlations identified were as follows:

fraud total percentage pontos incomemean city count seller disper type V11 V12

1 0 0 0.00000000 0 0.0000000 0.00000000 0.0000000 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000

2 11129 257284 0.04325570 1 0.2224684 0.78694773 0.8936676 0.4836477 0.03171155 0.5155814 0.04148007 0.9585199

3 10665 176858 0.06030262 2 0.2193214 0.69504541 0.8478000 0.3839804 0.04185849 0.6262858 0.05724616 0.9427538

4 9028 84225 0.10718908 3 0.2374615 0.38884540 0.6939401 0.3978975 0.07532368 0.8855479 0.09684283 0.9031572

5 8539 68613 0.12445163 4 0.2299642 0.28858617 0.6347340 0.3975054 0.08782032 0.9205853 0.11061024 0.8893898

6 7797 55553 0.14035246 5 0.2170008 0.26569850 0.6040410 0.3427865 0.09242719 0.9258642 0.12386626 0.8761337

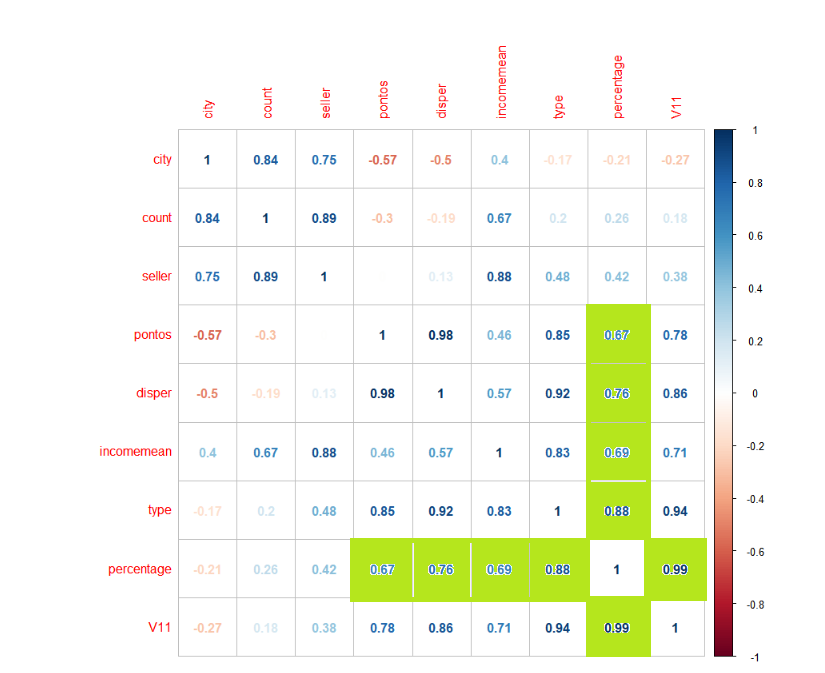
7 4907 31702 0.15478519 6 0.2181868 0.04520746 0.5522413 0.3000573 0.11571376 0.9813297 0.13721372 0.8627863

8 2056 22233 0.09247515 7 0.2097575 0.00000000 0.3972168 0.2496612 0.12394320 0.9885648 0.09891327 0.9010867

9 787 8812 0.08931003 8 0.2000000 0.00000000 0.0000000 0.2306427 0.13803058 0.9929159 0.09794667 0.9020533



That understanding allowed us to create an algorithm that gives the transaction a score between 1 and 8 (the closer you get to 8 the bigger the chances to have a fraud). That variable becomes another of our indicators. Therefore we created three synthetic variables “Dispersion”, “V11” and “Points”.



As we can see, the percentage of frauds has strong correlation with five variables:

**V11** => Synthetic variable V11 (come from the clustering process - % of frauds associated with the profile) is also very effective predictor. (0.99)

**Type** => type of the transaction (Recurrent, in person or on-line). (0.88)

**Disper**=> Synthetic variable which indicates how far from the average expenditures of this specific client with that specific kind of item the transaction is. (0.76)

**Income** => Income of the buyer (0.69)

**Pontos** => synthetic variable merging eight parameters (0.67)

Once we finished creating the synthetic variables we evaluated that we need to treat the data regards three aspects:

* Encoding
* Cross-validation (to test each model)

The k-fold cross-validation procedure provides a good general estimate of model performance that is not too optimistically biased, at least compared to a single train-test split.

* Balancing using SMOTE (The database is very imbalanced – frauds are just 4% of the samples)

At 4%, this is clear we have a skewed and imbalanced dataset. Imbalanced data pose classification problem for predictive modelling as most of the machine learning algorithms are used for classification were designed around the assumption of an equal number of examples for each class.

As a result, models train on imbalanced data have poor predictive performance specifically on minority class. Specifically, for fraud detection where predicting the minority class in the most important aspect of the model.

Synthetic Minority Oversampling Technique makes data balanced by synthesizing new data from the existing dataset using KNN. The approach is effective because new synthetic examples from the minority class are created that are plausible and relatively close in feature space to existing examples from the minority class.

Once we got the data encoded and balanced we tested five methods to spot frauds. We also tested these five methods without treating the data just to check if balancing the data was, in fact, improving the quality of the identification process. In a way, it was a research process.

* Decision-tree (Baseline Model)
* Random-Forest
* Random Forest + Bagging
* Support Vector Machine Classifier
* XGBoost

In logical terms we created the following analysis scenarios:



This strategy of testing several possible techniques was important because we needed to explore alternatives and try through these techniques to improve the success rate of the identification process.

**Summary of findings /Evaluating the results**

We tested several models training to identify which ones yield the best results. It was a systematic process, where we run the algorithm and evaluate the confusion Matrix. The objective was to identify the % of frauds identified correctly and the percentage of false positive.

**Using Decision-Tree model (not treated)**

The composition of these variables applied in a Decision tree-method generated the following result:

In a test set composed by 5% of the total database (5% of 290.398 -> 14.520)

* The model identified as fraud  and they were in fact frauds 231 ( 1,57%)
* The model identified as fraud but they were not 444 ( 3,05%)
* The model identified as not being frauds but they were 357 ( 2,45%)
* The model identified as not being frauds and they in fact were not 13.488 (93,78%)

**It spotted 60,71% of the frauds with a false positive rate (444 ->3,05%)**

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 13488 231

1 444 357

Accuracy : 0.9535

95% CI : (0.95, 0.9569)

No Information Rate : 0.9595

P-Value [Acc > NIR] : 0.9998

Kappa : 0.4902

Mcnemar's Test P-Value : 3.353e-16

Sensitivity : 0.9681

Specificity : 0.6071

Pos Pred Value : 0.9832

Neg Pred Value : 0.4457

Prevalence : 0.9595

Detection Rate : 0.9289

Detection Prevalence : 0.9448

Balanced Accuracy : 0.7876

'Positive' Class : 0

**Using Random Forest model(not treated)**

If instead using decision tree we use random forest the results would be like:

In a test set composed by 5% of the total database (5% of 290.398 -> 14.520)

* The model identified as fraud  and they were in fact frauds 364 (2,5%)
* The model identified as fraud but they were not 315 ( 2,1%)
* The model identified as not being frauds but they were 224 (1,5%)
* The model identified as not being frauds and they in fact were not 13.617 (93,9%)

**It spotted 62% of the frauds with false positive rate (False positive ->2,1%)**

**Therefore using random forest would be more effective in identifying frauds but would triple the percentage of false positive.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | Confusion Matrix and Statistics  Reference  Prediction 0 1  0 13617 224  1 315 364    Accuracy : 0.9629  95% CI : (0.9597, 0.9659)  No Information Rate : 0.9595  P-Value [Acc > NIR] : 0.0195472    Kappa : 0.5553    Mcnemar's Test P-Value : 0.0001059    Sensitivity : 0.9774  Specificity : 0.6190  Pos Pred Value : 0.9838  Neg Pred Value : 0.5361  Prevalence : 0.9595  Detection Rate : 0.9378  Detection Prevalence : 0.9532  Balanced Accuracy : 0.7982    'Positive' Class : 0 | |  | | |  | | --- | |  | |     Note the relative importance of the synthetic variables.   1. Pontos – V12 – Synthetic variable 2. V11 - Kmeans % - Synthetic variable 3. Seller\_scor – Original variable (seller info) 4. Type of transaction - Original variable 5. Dispersion – synthetic variable   Note that this ranking doesn´t match exactly the ranking identified when using correlation. |
|  |
| |  | | --- | |  | |

**Using the** **Random Forest + Bagging model (not treated)**

In a test set composed by 5% of the total database (5% of 290.398 -> 14.284)

* The model identified as fraud  and they were in fact frauds 457 ( 3,14%)
* The model identified as fraud but they were not 677 ( 4,6%)
* The model identified as not being frauds but they were 131 (0,9%)
* The model identified as not being frauds and they in fact were not 13.255 (91,28%)

**It spotted 77% of the frauds with false positive rate (False positive ->4,6%)**

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 13255 131

1 677 457

Accuracy : 0.9444

95% CI : (0.9405, 0.948)

No Information Rate : 0.9595

P-Value [Acc > NIR] : 1

Kappa : 0.5043

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.9514

Specificity : 0.7772

Pos Pred Value : 0.9902

Neg Pred Value : 0.4030

Prevalence : 0.9595

Detection Rate : 0.9129

Detection Prevalence : 0.9219

Balanced Accuracy : 0.8643

'Positive' Class : 0

**Using the Support Vector Machine Classifier model (not treated)**

In a test set composed by 5% of the total database (5% of 290.398 -> 14.520)

* The model identified as fraud  and they were in fact frauds 229 ( 1,5%)
* The model identified as fraud but they were not 162 ( 1,1%)
* The model identified as not being frauds but they were 359 (2,4%)
* The model identified as not being frauds and they in fact were not 13.770 (94,83%)

**It spotted 38,94% of the frauds with false positive rate (False positive ->1,1%)**

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 13770 359

1 162 229

Accuracy : 0.9641

95% CI : (0.961, 0.9671)

No Information Rate : 0.9595

P-Value [Acc > NIR] : 0.002212

Kappa : 0.45

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.9884

Specificity : 0.3895

Pos Pred Value : 0.9746

Neg Pred Value : 0.5857

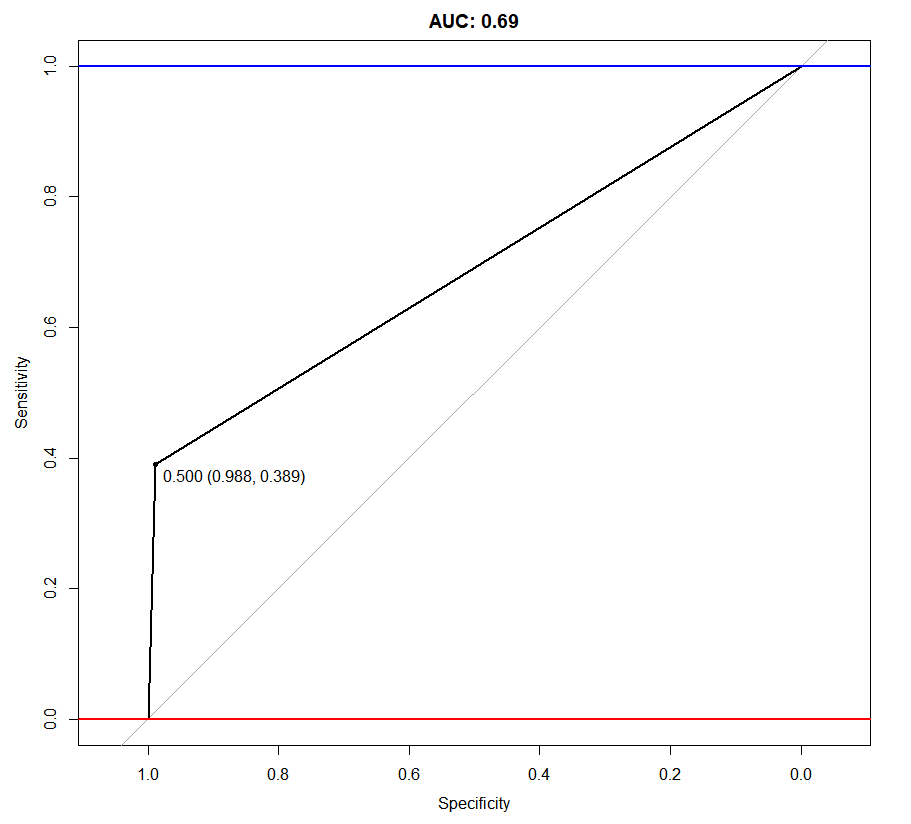
Prevalence : 0.9595

Detection Rate : 0.9483

Detection Prevalence : 0.9731

Balanced Accuracy : 0.6889

'Positive' Class : 0



**Using the XGBoost model(Not treated)**

In a test set composed by 5% of the total database (5% of 290.398 -> 14.520)

* The model identified as fraud  and they were in fact frauds 537 ( 3,6%)
* The model identified as fraud but they were not 780 ( 5,3%)
* The model identified as not being frauds but they were 51 (0,3%)
* The model identified as not being frauds and they in fact were not 13.152 (90,57%)

**It spotted 91,33% of the frauds with false positive rate (False positive ->5,30%)**

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 13152 51

1 780 537

Accuracy : 0.9428

95% CI : (0.9389, 0.9465)

No Information Rate : 0.9595

P-Value [Acc > NIR] : 1

Kappa : 0.5379

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.9440

Specificity : 0.9133

Pos Pred Value : 0.9961

Neg Pred Value : 0.4077

Prevalence : 0.9595

Detection Rate : 0.9058

Detection Prevalence : 0.9093

Balanced Accuracy : 0.9286

'Positive' Class : 0

**Using Decision-Tree model (treated)**

The composition of these variables applied in a Decision tree-method generated the following result:

In a test set composed by 5% of the total database (5% of 290.398 -> 14.520)

* The model identified as fraud  and they were in fact frauds 314 ( 2,1%)
* The model identified as fraud but they were not 1.036 ( 7,13%)
* The model identified as not being frauds but they were 274 ( 1,88%)
* The model identified as not being frauds and they in fact were not 12.896 (88,81%)

**It spotted 53,40% of the frauds with a false positive rate (False positive ->7,13%)**

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 12896 274

1 1036 314

Accuracy : 0.9098

95% CI : (0.905, 0.9144)

No Information Rate : 0.9595

P-Value [Acc > NIR] : 1

Kappa : 0.2836

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.9256

Specificity : 0.5340

Pos Pred Value : 0.9792

Neg Pred Value : 0.2326

Prevalence : 0.9595

Detection Rate : 0.8882

Detection Prevalence : 0.9070

Balanced Accuracy : 0.7298

'Positive' Class : 0

**Using Random Forest (treated)**

The composition of these variables applied in a Decision tree-method generated the following result:

In a test set composed by 5% of the total database (5% of 290.398 -> 14.520)

* The model identified as fraud  and they were in fact frauds 463 ( 0,8%)
* The model identified as fraud but they were not 1.163 ( 8,0%)
* The model identified as not being frauds but they were 125 ( 3,1%)
* The model identified as not being frauds and they in fact were not 12.769 (87,94%)

**It spotted 78,40% of the frauds with a false positive rate (False positive ->8,0%)**

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 12769 125

1 1163 463

Accuracy : 0.9113

95% CI : (0.9066, 0.9159)

No Information Rate : 0.9595

P-Value [Acc > NIR] : 1

Kappa : 0.3815

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.9165

Specificity : 0.7874

Pos Pred Value : 0.9903

Neg Pred Value : 0.2847

Prevalence : 0.9595

Detection Rate : 0.8794

Detection Prevalence : 0.8880

Balanced Accuracy : 0.8520

'Positive' Class : 0

**Using the Random Forest + Bagging (treated)**

In a test set composed by 5% of the total database (5% of 290.398 -> 14.520)

* The model identified as fraud  and they were in fact frauds 487 ( 3,3%)
* The model identified as fraud but they were not 905 ( 6,23%)
* The model identified as not being frauds but they were 101 (0,6%)
* The model identified as not being frauds and they in fact were not 13.027 (89,71%)

**It spotted 82,82% of the frauds with false positive rate (False positive ->9,9%)**

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 13027 101

1 905 487

Accuracy : 0.9307

95% CI : (0.9265, 0.9348)

No Information Rate : 0.9595

P-Value [Acc > NIR] : 1

Kappa : 0.4612

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.9350

Specificity : 0.8282

Pos Pred Value : 0.9923

Neg Pred Value : 0.3499

Prevalence : 0.9595

Detection Rate : 0.8972

Detection Prevalence : 0.9041

Balanced Accuracy : 0.8816

'Positive' Class : 0

**Using the Support Vector Machine Classifier model (treated)**

In a test set composed by 5% of the total database (5% of 290.398 -> 14.520)

* The model identified as fraud  and they were in fact frauds 291 ( 2,0%)
* The model identified as fraud but they were not 1450 ( 9,9%)
* The model identified as not being frauds but they were 297 (2,0%)
* The model identified as not being frauds and they in fact were not 12.482 (85,96%)

**It spotted 49,49% of the frauds with false positive rate (False positive ->9,9%)**

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 12482 297

1 1450 291

Accuracy : 0.8797

95% CI : (0.8743, 0.8849)

No Information Rate : 0.9595

P-Value [Acc > NIR] : 1

Kappa : 0.2016

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.8959

Specificity : 0.4949

Pos Pred Value : 0.9768

Neg Pred Value : 0.1671

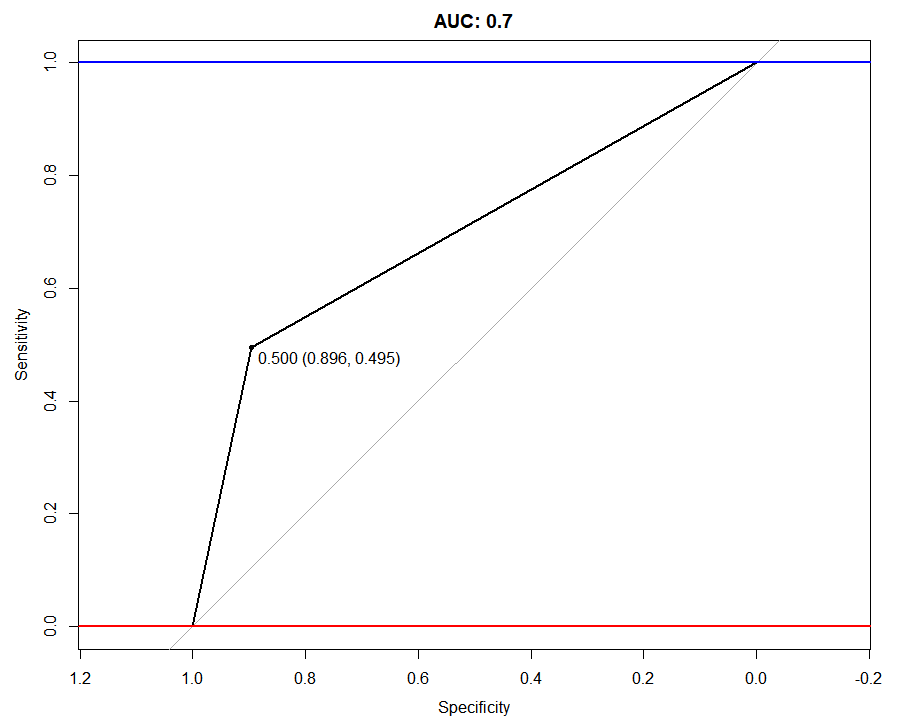
Prevalence : 0.9595

Detection Rate : 0.8596

Detection Prevalence : 0.8801

Balanced Accuracy : 0.6954

'Positive' Class : 0



**Using the XGBoost model(treated)**

In a test set composed by 5% of the total database (5% of 290.398 -> 14.520)

* The model identified as fraud  and they were in fact frauds 558 (3,8%)
* The model identified as fraud but they were not 927 (6,3%)
* The model identified as not being frauds but they were 30 (0,2%)
* The model identified as not being frauds and they in fact were not 13.005 (89,56%)

**It spotted 94,90% of the frauds with false positive rate (False positive ->5,30%)**

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 13005 30

1 927 558

Accuracy : 0.9341

95% CI : (0.9299, 0.9381)

No Information Rate : 0.9595

P-Value [Acc > NIR] : 1

Kappa : 0.5099

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.9335

Specificity : 0.9490

Pos Pred Value : 0.9977

Neg Pred Value : 0.3758

Prevalence : 0.9595

Detection Rate : 0.8957

Detection Prevalence : 0.8977

Balanced Accuracy : 0.9412

'Positive' Class : 0

**Summarizing the methods and conclusion:**



The table above represents a potential improvement of 44,89 % over the current model. That means in practical terms that every day you would be able to spot something around 5.936 additional frauds, totalizing a value around CAD 8.444.000 a month in terms of avoided frauds. Of course, there is also the issue of the false positive that can lead to alienating clients, which due to our lack of understanding of the business is hard for us to define what would be acceptable.

However, the client can improve the most accurate model by accepting higher losses, especially from high-net-worth cardholders, in order to prevent alienating those clients and losing business due to the increase in false-positive detection.

Another important additional advantage would be having the dynamic process as it “learns” new patterns as they appear and therefore will demand less effort to maintain.

We understand that the actual implementation of the model needs to be analyzed with care and skepticism, tests need to be made and eventual performance issues evaluated. (We need to remember that the whole verification has to occur in few seconds just after the user made the transaction – although in the online services we may have the option of letting an order “pending approval”).

In summary, although we recognize that there are several issues that need to be evaluated it seems clear that deploying the machine learning model to this process would add a lot of value and surely should be considered.

