Bicocca University, Milan, Italy

Course of Machine Learning

A.Y. 2017-2018 February Session

Project Work

MACHINE LEARNING MODELING OF CREDIT CARD FRAUD DETECTION

Group nr. 27

Guido Gennaro (834130)

Riccardo Piani (833144)

Project developed with Knime v. 3.5.1

Requirements:

- Extension: Scripting R
- R libraries: ROCR, pracma, randomForest

Dataset from https://www.kaggle.com/mlg-ulb/creditcardfraud

Abstract

A dataset of 280k credit card transactions with non-labeled predictors (PCA variables) and only 500 observations classified as frauds: that is the challenge for this project work.

The project is focused on counteracting the negative effects of imbalanced data and it is supposed to be managed in a business context where model performances are evaluated in terms of costs with the respect to the quality of the anti-fraud services provided to the credit card holder.

The final result of the project shows that indeed it's not the definitive one and that, probably, business objectives and constraints will require to rework looping among the several decisions taken during the model development process.

Table of Content

Introduction

For sure, among the business areas where fraud detection is being relevant – e.g. insurance claims, insider trading, …- credit card transactions are undoubtedly one of them. Especially during the last decades, the usage of credit cards has been increased and, currently, the fraudulent activity is a growing problem causing billions of dollars of loss every year.

Nowadays, the IT and hardware state of the art allows to face the problem with sophisticated machine learning algorithms that are able to evaluate large quantity of data, learn, detect fraudulent patterns and make predictions in real time.

This project work is focused on the development of supervised predictive models applied to a credit card dataset downloadable from *Kaggle.com* [1]: it is related to European credit card transactions made in September 2013 over a period of two days.

A first glance to the dataset puts in evidence:

- the data preparation activity will be short due to 28 PCA unlabeled predictors and two labeled columns: *Time* and *Amount*;

- the transactions are "stand-alone": they are not linked to the credit card holder (customer ID);

- the 284k classified transactions are highly imbalanced with only 492 actual frauds (0,172%).

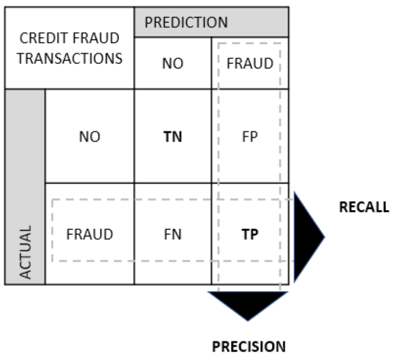
Class is the binary target variable (fraud=1) and the question to be answered is: **is that credit card transaction a fraud?**

The imbalanced aspect is the first point to be tackled.

1. Validation Metrics

In case of an imbalanced dataset the traditional metric as *Accuracy* is misleading because even a *ZeroR* model would highly perform: in our case there would be classified all non-frauds with a baseline *Accuracy* of 99,83%.

So, instead of the *Accuracy* there will be also considered the key metrics: *Precision*, *Recall* and *F1*.



Precision

TP/(TP+ FP)

Recall

TP/(TP+FN)

Fig. Confusion Matrix

F1 = 2*Precision*Recall / (Precision + Recall)

These metrics allow to focus on the performance of the less representative class.

It follows that also for the *Evaluation Phase* the best metrics to be used are based on *Precision* and *Recall*, in fact, different models may be more effectively compared by examining the *PR AUC* values (Area Under the Curve Precision-Recall) and *PR* curves.

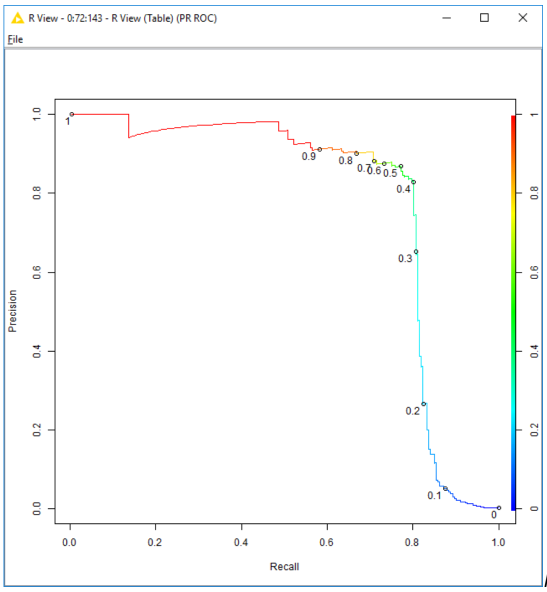


Fig. Example of a PR curve with the cut-off probabilities

2. Business Objective

The project work is supposed to be contextualized in a business environment where *Precision* and *Recall* are to be negotiated among the Business Units. As an example, below are the descriptions of the anti-fraud services provided by USA Visa to the end customers [2]:

1) ***You are notified of unusual activity***: *Your financial institution will notify you to verify the legitimacy of questionable charges.*
2) ***Charges can be put on hold***: *To safeguard your security, your financial institution may temporarily put suspicious charges on hold.*
3) ***Shop worry-free***: *With Visa's Zero Liability Policy you won't be held responsible for unauthorized charges.*

Let's suppose that the main objective of the project is the reduction of the service costs, that means:

- the notification is an operational cost;
- moreover, the detection of *FPs* (false positives) could impact on the customer satisfaction (counterclaims costs) and, in the worst cases, on the reputational risk (customers annoyed by recurrent false notifications or by blocked transactions);
- conversely, missing fraud detections (*FNs*, false negatives) imply a loss cost that is worse than the false positive cost – loss of customer trust as well - and, not secondarily, it does not prevent other probable subsequent and recurrent frauds over the same customer.

Incorporation of specific costs with the **cost sensitive matrix** (applying different weights to the *FPs* and *FNs*) during model training may bias the model towards the fraudulent class.

We we'll suppose that the **business objective** is the cost reduction of the *Shop worry-free service* (minimum FN costs $\rightarrow$ maximum recall) capping as much possible the costs of the *notification* and "*on hold charges*" services (FP costs $\rightarrow$ reasonable precision)

Finally, once the model has been optimized and selected, the last but not the least step before the deployment is the acceptance based on the business context, where the involved Business Units (Operations, Marketing, Line of Business, …) play their roles into the inevitable precision & recall trade-off and the consequent **alternative cut-off** tuning.

3. Data Preprocessing

Specifically for that dataset, the data preparation activity is focused on two variables (*Time* and *Amount*).

Then the dataset is partitioned in *Training*, *Validation* and *Test* set and some techniques as outlier removal, over/under sampling and feature selection are applied in order to evaluate impacts on the performance.

3.1 Data Preparation

For the ***28 PCA*** preprocessed variables, which distributions look quite similar (standardized and centered around zero with several outliers), we do not proceed further with data preparation activities.

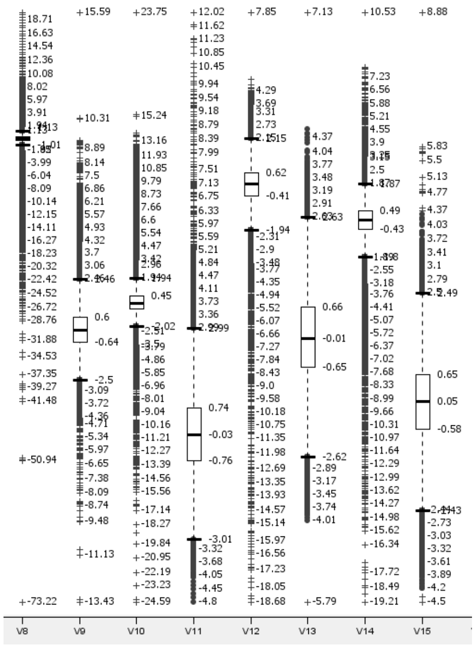


Fig. A frame of PCA variable boxplots – normalized view

The variable ***Time***, an integer sequence of seconds from zero to 48 hours, is unusable if taken as is: the distribution reveals the possibility to overlap the two days into a binned variable of 24-hour intervals.

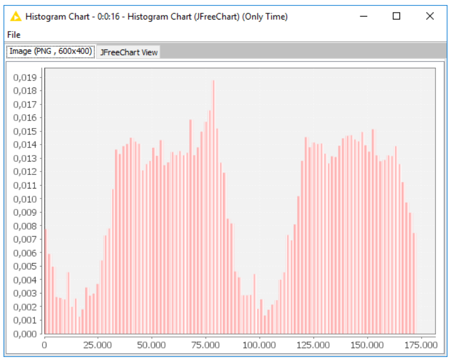


Fig. Variable Time histogram chart over 48 hours

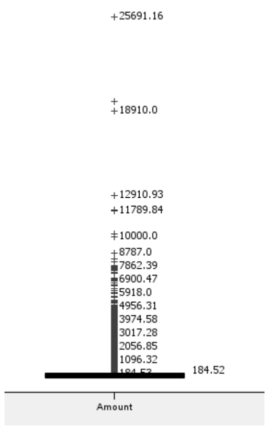


Fig. Amount boxplot

As the variable ***Amount*** is strongly skewed, it is transformed with the log function.

Moreover, 1.825 entire rows are also deleted because considered as **missing values** with zero amount value.

None of the other variables has missing values.

Besides the *PCA* variables where we don't expect relationship, the **collinearity** between *PCAs*, *Time* and *Amount* is examined with *Pearson's* measure.

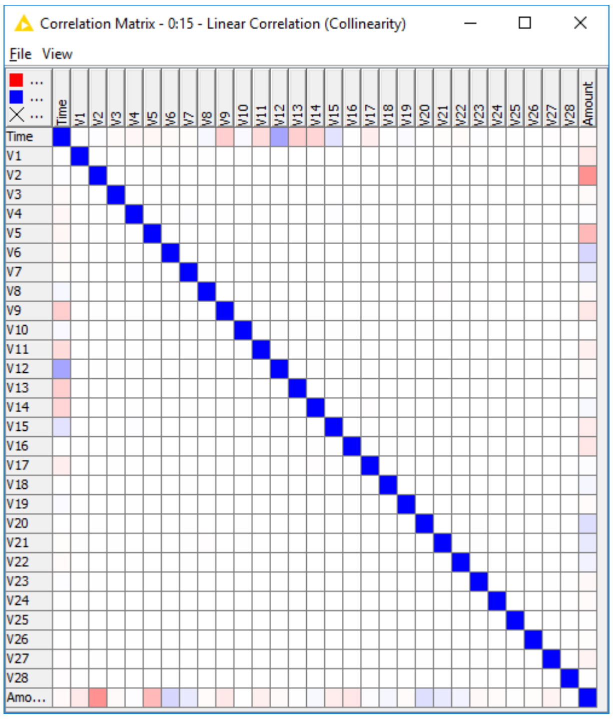


Fig. Linear Correlation between predictors

Because the strongest correlation is -0,43 (*Amount* vs *V2*) the correlations are considered negligible and all predictors are kept.

3.2 Data Partitioning

Data partitioning could be critical for a such imbalanced dataset with about only 500 frauds: we pay attention to distribute homogeneously the frauds between the partitioned datasets (keeping at least 100 frauds per partition).

A first 50% stratified sampling on the *Class* variable generates the *Training* set, then the remaining is further partitioned into two equal partitions generating the *Validation* and *Test* set.

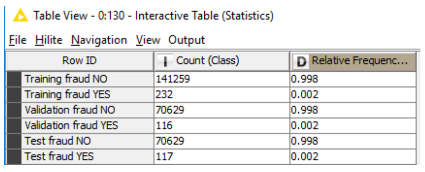


Table View - 0:130 - Interactive Table (Statistics)

File Hilite Navigation View Output

Row ID	Count (Class)	Relative Frequenc...
Training fraud NO	141259	0.998
Training fraud YES	232	0.002
Validation fraud NO	70629	0.998
Validation fraud YES	116	0.002
Test fraud NO	70629	0.998
Test fraud YES	117	0.002

Fig. Breakdown of 284k transaction into Training, Validation and Test set

The *Test* set will be used only in the end for verifying if occurs overfitting.

3.3 Outlier removal, Over/Under sampling and Feature selection

Now that we have the prepared dataset we decide to have a first idea on the performances with the application of the ***Random Forest*** model. It has been selected because:

- It works well on large dataset and on nominal target.
- It does not require feature engineering (scaling and normalization);
- it is robust to outliers;

A default *Random Forest* parametrization set (*weka 3.7: maxDepth=10, numFeatures=5, numTrees=100*) applied to the *BASIC* **training/validation** dataset (just the output of the data preparation and partitioning phase) is used as a benchmark for comparing the impact on performances of the "outlier removal", "over/under sampling" and "feature selection" preprocessing techniques:

- **outlier removal**, entire row deletion if one of the variable is a severe outlier - exceeds 3 x IQR threshold (about 10k out of 141k deleted);
- **over sampling**, by using SMOTE, the training set size gets doubled (from 141k to 282k and equal class size);
- **under sampling,** the non-frauds are reduced to the number of frauds (extreme sampling with a training set of 500 rows);
- **feature selection**, by applying the *Random Forest* wrapper, the first 10 – out of 30 - predictors have been selected (V17, V12, V14, V16, V10, V11, V9, V4, V18, V7). A point of attention on changing the *numFeature* according to the new dataset dimensionality *numFeature = sqrt(# predictors).*

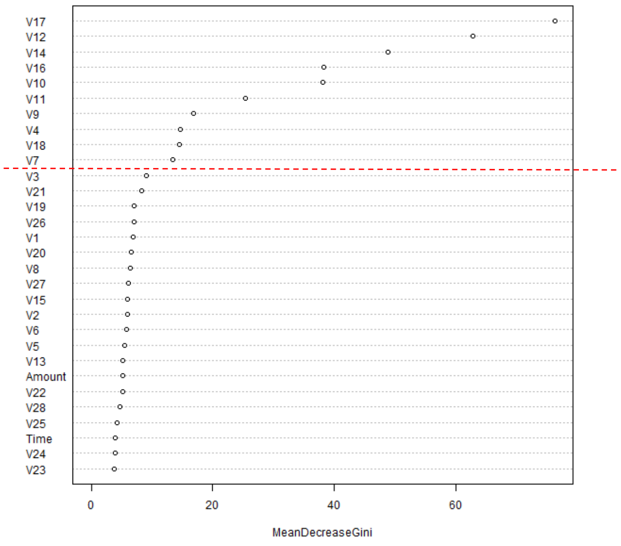
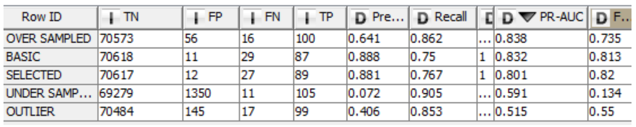


Fig. Mean decrease Gini ranking, output from Random Forest wrapper

Then, making the comparison of the performances by using the *PR AUC* as driver, the "under sampled" and "outlier reduction" techniques are immediately abandoned:

- the "outlier removal", for this kind of imbalanced dataset, is risky because the frauds could be themselves outliers by nature;
- the "under sampling" reduced the model capability of non-fraud detection, hence penalizing significantly the *Precision*.



Row ID	TN	FP	FN	TP	Pre...	Recall		PR-AUC	F...
OVER SAMPLED	70573	56	16	100	0.641	0.862	...	0.838	0.735
BASIC	70618	11	29	87	0.888	0.75	1	0.832	0.813
SELECTED	70617	12	27	89	0.881	0.767	1	0.801	0.82
UNDER SAMP...	69279	1350	11	105	0.072	0.905	...	0.591	0.134
OUTLIER	70484	145	17	99	0.406	0.853	...	0.515	0.55

Fig. Confusion Matrix and metrics comparison on different preprocessing techniques

The remaining options are comparable:

- even if the "over sampled" dataset shows the best performances in terms of *PR AUC* value, it has not been selected because the larger dataset size (training of 282k rows) increases dramatically the computational time, practically preventing model tuning and k-folder cross validation;
- the "**feature selection**" technique has been preferred because the dataset, though its size has been reduced by 2/3, the negative impact on the performance is considered acceptable and the computational time is highly benefitted.

4. Modeling

From a business point of view, for that dataset, there is no interpretability need about model type choice (since *PCA* variables are not interpretable), otherwise models as *Decision Tree* or *Logistic Regression* should have been preferred.

The main objective of modeling will be to find the best model trying to cope the business target (maximum *Recall* with acceptable *Precision*), counteracting as much as possible the negative effects of class imbalance. Starting from the chosen preprocessed training dataset - the feature selected one – we proceed following two steps:

1) perform **Model Tuning** trying to maximize the *PR AUC value.*
2) perform **Cost Sensitive** training for giving more weight - in costs - on *FNs*

For all the steps the validation set only has been used.

Five different type of models have been trained and compared in performances: **DT** *Decision Tree C4.5*, **RF** *Random Forest*, **LR** *Logistic Regression*, **SVM** *Support Vector Machine* and **NB** *Naïve Bayes*.

4.1 Model Tuning

In order to have a starting point we decide, at first, to assess the performances of the models keeping the default parametrization set (all *weka 3.7* nodes)

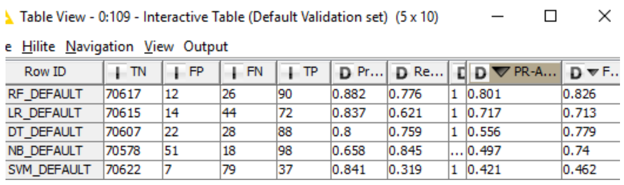


Table View - 0:109 - Interactive Table (Default Validation set) (5 x 10)
e Hilite Navigation View Output

Row ID	TN	FP	FN	TP	Pr...	Re...		PR-A...	F..
RF_DEFAULT	70617	12	26	90	0.882	0.776	1	0.801	0.826
LR_DEFAULT	70615	14	44	72	0.837	0.621	1	0.717	0.713
DT_DEFAULT	70607	22	28	88	0.8	0.759	1	0.556	0.779
NB_DEFAULT	70578	51	18	98	0.658	0.845	...	0.497	0.74
SVM_DEFAULT	70622	7	79	37	0.841	0.319	1	0.421	0.462

Fig. Confusion Matrix and metrics comparison on different model with default parametrization set

The model tuning is performed for finding the parametrization set that maximizes the *PR AUC* value: the larger is the *PR Area Under the Curve* the higher are the *Precision* and *Recall* values that can be obtained once the probability cut-off value has been fixed.

For simplicity, one parameter per model has been chosen for tuning (except for Naïve Bayes that has not parameters)

Decision Tree J48: *minNumObj* sets the minimum instances per node applying the online pruning (tuned from 1 to 100, step 10 → optimal=21)

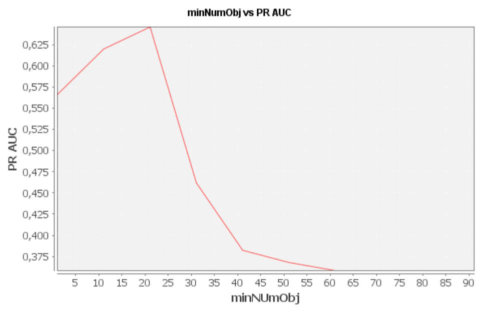


Fig. PR AUC values vs. minNumObj parameter

Random Forest: *maxDepth* defines the maximum depth of the trees (tuned from 1 to 15 → optimal=5)

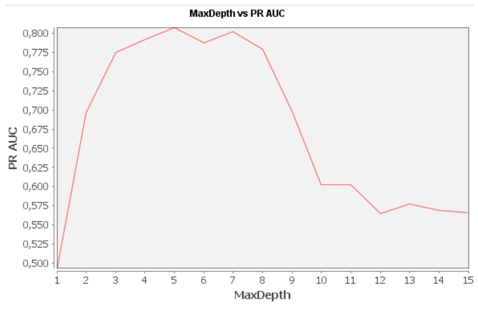


Fig. PR AUC values vs. maxDepth parameter

Logistic: *ridge* in the log-likelihood (tuned from 0 to 3.000, step 150 → optimal = 1950)

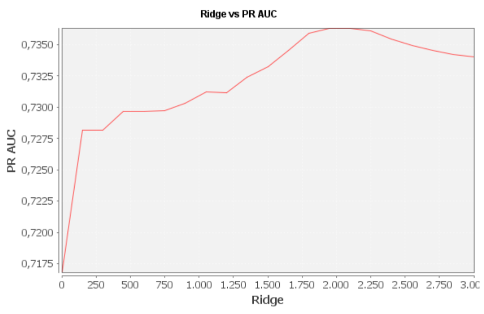


Fig. PR AUC values vs. ridge parameter

SMO (SVM): *complexity*, controls how soft the class margins are (tuned from 0,1 to 1000, step 100 → optimal=200)

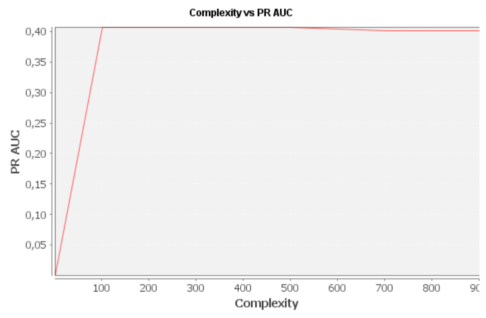


Fig. PR AUC values vs. Complexity parameter

The new *PR AUC* values are showed in the table below. In general, compared to the default parametrization, all models do not show meaningful improvements in *PR AUC* across the tuning parameters.

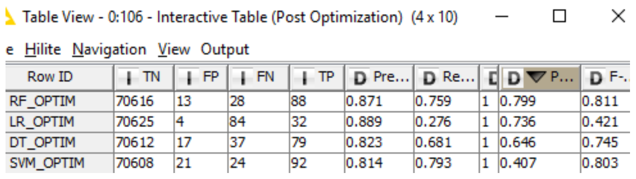


Table View - 0:106 - Interactive Table (Post Optimization) (4 x 10)

Row ID	TN	FP	FN	TP	Pre...	Re...		P...	F-..
RF_OPTIM	70616	13	28	88	0.871	0.759	1	0.799	0.811
LR_OPTIM	70625	4	84	32	0.889	0.276	1	0.736	0.421
DT_OPTIM	70612	17	37	79	0.823	0.681	1	0.646	0.745
SVM_OPTIM	70608	21	24	92	0.814	0.793	1	0.407	0.803

Fig. Performances of the models with the optimized parameters

As a comment, the *Random Forest PR AUC* is little bit decreased because the *numTree* parameter has been set to 10 instead of 100 - of the default – due to high computational time.

4.2 Cost Sensitive

As stated before in the "Business Objective" chapter the misclassification of true events (false negatives) is *N* times much costly than incorrectly predicting nonevents (false positives). Because of the definition of *N* from a business point of view could be very complex, our target will be to balance the *Precision* and *Recall* values tuning the "cost matrix" with different values of *N* (e.g. *FNs* 10 or 100 times *FPs*)

The cost-sensitivity method used is the default one of the *weka* node (*CostSensitiveClassifier*): reweighting training instances according to the total cost assigned to each class [3]

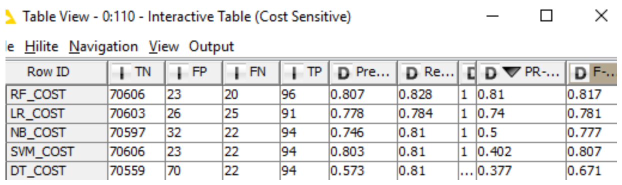


Table View - 0:110 - Interactive Table (Cost Sensitive)

Row ID	TN	FP	FN	TP	Pre...	Re...		PR-...	F-...
RF_COST	70606	23	20	96	0.807	0.828	1	0.81	0.817
LR_COST	70603	26	25	91	0.778	0.784	1	0.74	0.781
NB_COST	70597	32	22	94	0.746	0.81	1	0.5	0.777
SVM_COST	70606	23	22	94	0.803	0.81	1	0.402	0.807
DT_COST	70559	70	22	94	0.573	0.81	...	0.377	0.671

Fig. Performances of the models with the cost matrix set

The comparison of the performances leads us to make the analysis easier abandoning the last two models, because:

- SVM training takes a lot of time and the *PR AUC* seems hopelessly low;
- The Decision Tree family is better represented by the ensemble model (random forest)

5. Models Evaluation

Among the remaining models (*Random Forest*, *Logistic Regression* and *Naïve Bayes*) we want to select the best one that will be subject to discussion for setting the more appropriate cut-off probability according to the business objective.

We'll follow two steps:

1) the models are compared through the ***PR curve*** and ***AUC*** after a k-folder cross validation;
2) the **cut-off probability** is then fixed and the final validation is done on the **Test set** - used for the first time.

5.1 Precision-Recall curve

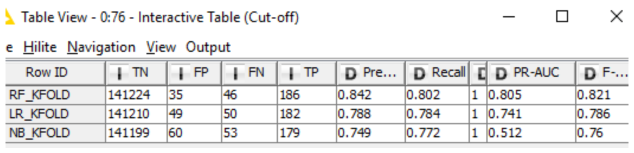


Table View - 0:76 - Interactive Table (Cut-off)

Row ID	TN	FP	FN	TP	Pre...	Recall		PR-AUC	F-...
RF_KFOLD	141224	35	46	186	0.842	0.802	1	0.805	0.821
LR_KFOLD	141210	49	50	182	0.788	0.784	1	0.741	0.786
NB_KFOLD	141199	60	53	179	0.749	0.772	1	0.512	0.76

Performances of the models with k-folder cross validation

Each model (as configured after the cost sensitive training) is trained through a 10-folder cross validation (stratified sampling) on the training set and the predicted values are quantified with the *PR curves.*

From the comparison table It's clear that **Random Forest** is the best performing model and that's evident also looking at the PR curves.

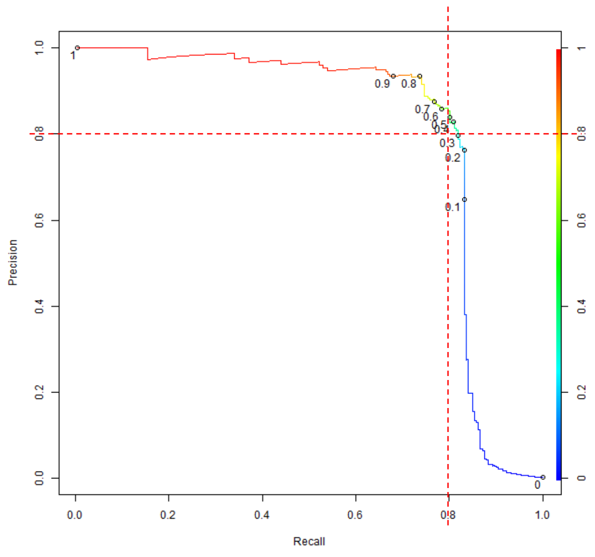


Fig. Random Forest PR curve

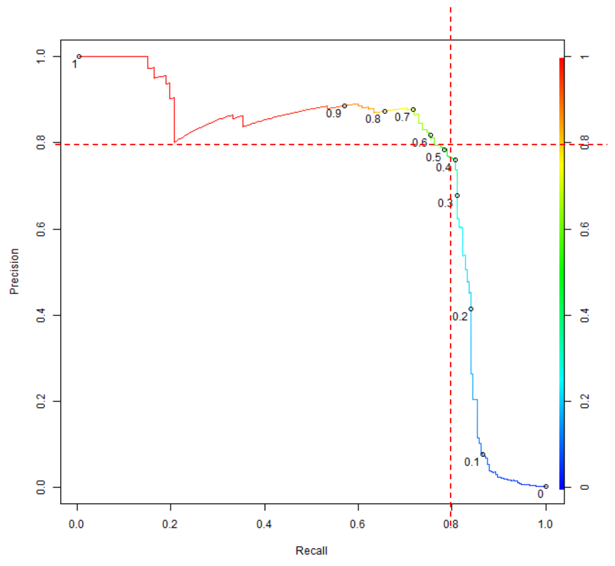


Fig. Logistic Regression PR curve

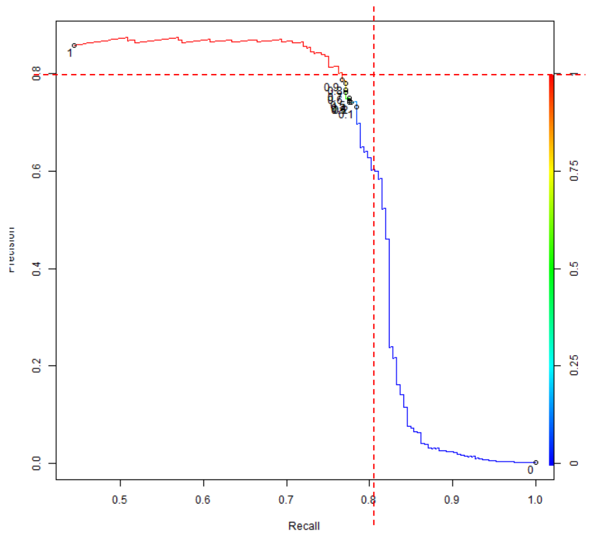


Fig. Naïve Bayes PR curve

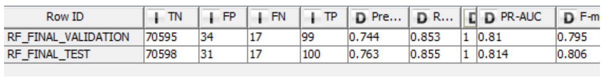
Random Forest is the only model that allows, applying a cut-off between about 0,3 and 0,5, to stay over 80% with both *Precision* and *Recall* (see the cross red lines)

5.2 Cut-off Probability and Test set validation

If we leave the *weka* node (*ThresholdSelector*) to optimize the cut-off maximizing the *F1* measure, the calculated value is 0,83, that means high *Precision* and low *Recall* (the top-left side of the *PR* curve): it's not the solution we are searching for, we want to stay as much as possible on the rightmost quadrant without falling in precision.

Analyzing the *PR* curve, it seems that it's a nonsense for the cut-off going under the value of 0,2 where there's no gain in *Recall* and high loss in *Precision*.

Now that we have fixed the cut-off value, the last step is the validation on the *Test* set that had been put aside at the very beginning of the data analysis and model development.



Row ID	TN	FP	FN	TP	Pre...	R...		PR-AUC	F-m
RF_FINAL_VALIDATION	70595	34	17	99	0.744	0.853	1	0.81	0.795
RF_FINAL_TEST	70598	31	17	100	0.763	0.855	1	0.814	0.806

Fig. Random Forest performances with cut-off = 0,2 on validation and test set

That final test shows that the performances are practically the same for both *Validation* and *Test* set, therefore the model generalizes. On the contrary, in case of overfitting, a complete check of the analysis is required (especially the model parametrization).

Finally, the selected model is:

RANDOM FOREST	WEKA 3.7

PARAMETERS	maxDepth = 5
	numFeatures = 3
	numTrees = 100

COST MATRIX	FN = 10 x FP
CUT-OFF	0,2

6. Conclusions

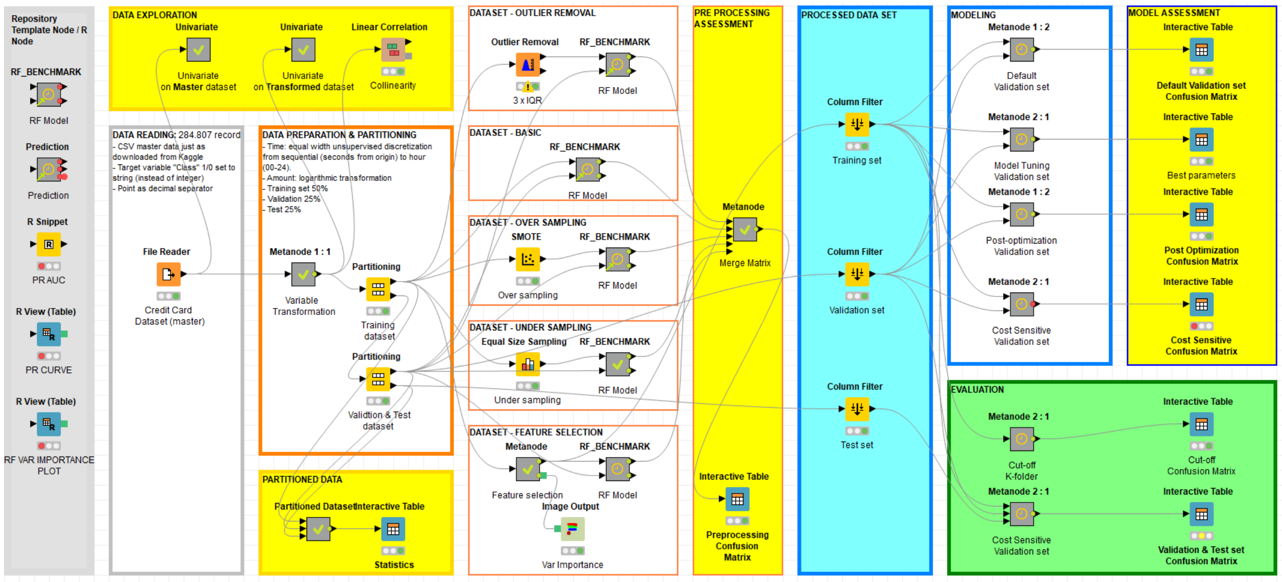
The final model declares performances that imply daily about 65 cases of false notifications (FP) and about 35 (out of 230) frauds not detected (FN). If these predictions are not considered acceptable in a business context, all the decisions taken along the model development process could be reviewed, for example:

- An *undersampling* method less strong (the one applied reduced the nonevent size equal to the frauds)
- No *feature selection* or usage of other wrappers (e.g. logistic regression)
- Other libraries (e.g. *Knime* library) instead of *Weka*
- Application of other models as *Neural Network*, *Gradient Boosting*, *Ensamble of different models, …*
- Tuning of other parameters also in combination

Characteristics of the models, data, computational time and business contexts all force to make several choices that can lead to different solutions: a continuous attention is required for avoiding mistakes but also curiosity and patience are needed to discover chances for improvements. It's a continuous search to find the best model for that particular problem taking always into account the trade-offs between speed, model performances and complexity, and remembering that there isn't a perfect model that is the best for every problem (no free lunch theorem).

7. Knime Workflow

This project has been developed on *Knime v.3.5.1* and needs the scripting *R extension* with the *ROCR, pracma and randomForest* libraries. Dataset is available from https://www.kaggle.com/mlg-ulb/creditcardfraud



DATA READING: 284.807 record
- CSV master data just as downloaded from Kaggle
- Target variable "Class" 1/0 set to string (instead of integer)
- Point as decimal separator

DATA PREPARATION & PARTITIONING
- Time: equal width unsupervised discretization from sequential (seconds from origin) to hour (00-24).
- Amount: logarithmic transformation
- Training set 50%
- Validation 25%
- Test 25%

Fig. Knime workflow screenshot

The workflow follows the sequence of the phases as explained in the project work.

Apart from the Meta nodes that include other nodes, on the leftmost panel there are listed the **customized** ones:

RF_BENCHMARK: a Template Meta node that has been used for comparing preprocessing techniques

PREDICTION: a Template Meta node that collects in a table data about the confusion matrix and performance metrics

PR AUC: an R node that calculates the Area Under the Precision-Recall curve

PR CURVE: an R node that outputs the PR curve as an image

RF VAR IMPORTANCE PLOT: an R node that outputs the RandomForest varImportance plot as an image

The weka nodes have been selected because they are more standardized and flexible than the native Knime nodes. Besides the traditional model nodes:

- "*Parameter Optimization Loop*" allows to scan the chosen parameters into a defined interval;
- "*CostSensitiveClassifier*" gives the possibility to set discretional weights to the confusion matrix;
- "*ThresholdSelector*" calculates the best cut-off value setting the target parameter (e.g. maximizing the F1 measure) and allows to set manually the cut-off value.

8. References

[01] KAGGLE

https://www.kaggle.com/dalpozz/creditcardfraud/data

[02] VISA USA

https://usa.visa.com/support/consumer/security.html

[03] Knime node description