Paper 1857-2018 A comparison of Gradient Boosting with Logistic Regression in Practical Cases

Paulo Celio Di Cellio Dias, Serasa Experian; Melissa Forti, Bradesco; Marc Witarsa, Serasa Experian;

ABSTRACT

The usage of boosting methods are increasing every day with the constant evolution of software and hardware infrastructure and their cost reduction. However, we still facing some problems and issues, mainly when we refer to automatic processes, where in most cases you cannot interpret their parameters (the influence of each variable). This paper evaluates one of the main methods of boosting, gradient boosting, and its use in scoring models. We also evaluate the main methodology used today for scoring models, logistic regression, in order to compare the results with the boosting process. We present three real cases of models from the largest credit bureau in Brazil (Serasa Experian), and evaluate the main results, considering simulations of boosting and logistic regression. Finally, we analyze the results and indicate the pros and cons of using new methodology.

Keywords: proc treeboost, proc gradboost, gradient boosting, credit score, collection score, fraud score, Machine Learning

INTRODUCTION

Over the last years, Machine Learning algorithms are among the most studied subjects in the area of Data Mining. Because of the evolution of technology and cost reduction, the storage and processing of large volumes of data is accessible, and with this, it is possible to enable the usage of new knowledge mining techniques.

Machine Learning methods are based on the development of high level data algorithms, considering the types of unsupervised, semi-supervised and supervised learning, proposing algorithms that even take the data to exhaustion for the search of knowledge.

However, as these techniques are developed with high data level and complex algorithms, known as "blackbox". The validation of accuracy, prediction and interpretability is still much discussed and questioned by the market. There is an uncertainty about the real gain and stability of these new algorithms against traditional algorithms such as logistic regression, which is still today the main prediction method for Scoring Score.

The motivation of this work is to analyze in three real cases, the differences of fit and accuracy between logistic regression and gradient boosting methodologies.

The items that will be explored by this work are:

- 1) Gradient Boosting Machines method definition
- 2) Procedure Proc TREEBOOST definition
- 3) Step by STEP: Fitting Gradient Boosting model
- 4) Macro procedure to assist the fit of the method.
- 5) Evaluation of three real cases of comparison of Gradient Boosting with logistic regression

GRADIENT BOOSTING MACHINES – THEORY

The ensemble Gradient Boosting method is a method used to solve classification and regression problems. It consists of a series of combinations of additive models (weak learning), estimated iteratively resulting in a stronger learning model(stronger learning). Usually the gradient boosting method is used of decision tree models, however any model can be used in this process, such as a logistic regression.

Considering the use of decision trees for fitting the gradient boosting, the objective of each fit decision tree is to minimize the loss function, that is, to minimize the objective gradient function of the current model, but for this we can have loss functions with advantages and disadvantages for each type of problem. One of the loss functions used in the algorithm is the Huber Loss function in which it has the advantage of being less sensitive to outliers than the more known loss functions such as the quadratic error and absolute error.

In addition to the choice of loss function, it is necessary to define hyper-parameters at the beginning of the algorithm, such as tree depth and minimum number of individuals at each node, since complex trees can generate overfitting and simple trees can not model all the existing features. (underfitting)

The stopping condition of the algorithm is one of the hyper-parameters that need to be taken into account, since the successive minimization of the residuals even adjusts the random errors. Because of this fact, gradient minimization results in overfitting of training samples. To define the stopping criteria, the number of iterations on the training sample is adjusted by checking the minimum residue also in validation samples, also known as cross validation.

The complete step-by-step gradient boosting algorithm for any loss function is described by Friedman (2001):

1. Number of interactions

$$m = 1 to M$$

2. Loss function definition

$$F_0(x) = \arg\min\sum_{i=1}^n (L(y_i, \gamma)). \qquad (L(y_i, \gamma)) = loss function)$$

3. Minimization of residuals

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x) = F_{m-1}(x)}$$
to $i = 1, \dots, n$.

4. Fit the previous function with the residues found up to the iteration M.

$$F_m(x) = F_{m-1}(x) + \gamma_m r_m(x)$$

PROCEDURE PROC TREEBOOST

For the implementation of Gradient Boosting method in SAS ®, this paper is going to explore the TREEBOOST procedure. This procedure implements the algorithm proposed by Friedman (2001), and follows the structure presented in the topic Gradient Boosting Machines - Theory.

The TREEBOOST procedure is available on the SAS Miner® license and it can be used by SAS Enterprise Miner® or by programming in the SAS Enterprise Guide®. Complete procedure details is available on (SAS Institute Inc. 2016).

The default syntax and the main parameters are described below.

```
PROC TREEBOOST DATA=
      CATEGORICALBINS=
      INTERVALBINS=
      EXHAUSTIVE=
      INTERVALDECIMALS=
      LEAFSIZE=
      MAXBRANCHES=
      ITERATIONS=
      MINCATSIZE=
      MISSING=
      SEED=
      SHRINKAGE=
      TRAINPROPORTION =
      HUBER=
      :
      INPUT
              / LEVEL=;
      TARGET / LEVEL=;
      SUBSERIES ;
      ASSESS VALIDATA=;
      CODE FILE= NOPREDICTION;
      SAVE MODEL= IMPORTANCE=IMPORTANCE;
RUN;
```

QUIT;

Where:

TREEBOOST options:

- 1) DATA = Dataset that will be used for training the algorithm.
- CATEGORICALBINS = Number of preliminary bins for categorical variables. The most frequent categories will be prioritized. The default value is fifteen.
- INTERVALBINS = Number of bins for a continuous variable. In the process of constructing the trees, the algorithm will try to divide a continuous variable in the maximum number of bins specified here.
- 4) EXHAUSTIVE = Number of attempts to search for variable bins. The default value is five thousand.
- 5) INTERVALDECIMALS = Accuracy of continuous variables. The "MAX" parameter uses all available decimal positions without rounding. The indication of k value will use k decimal positions rounding value.
- LEAFSIZE = Minimum number of observations needed to form a new branch. The specified value must be integer and positive. Beware to this parameter, very low values increase the risk of overfitting.

- 7) MAXBRANCHES =Number of possible breaks per node. Parameter two indicates the binary tree.
- 8) ITERATIONS = Maximum number of iterations to test. It is the number of trees that will be developed in the boosting process. For a binary target variable, the number of trees will be the same as specified, for a categorical target, the number of trees is the specified number multiplied by the number of events.
- 9) MINCATSIZE = Minimum number of observations to consider a class of a nominal variable. It follows the same risk of the LEAFSIZE parameter, where low values can generate overfitting.
- 10) Missing: Parameter of how missing records should be treated. The algorithm allows the following options: 1) BIGBRANCH: Allocation of missing's in the largest class (higher number of records); 2) FIRSTBRANCH or LASTBRANCH: Allocation in the first or last class; 3) DISTRIBUTE Allocation of missing's proportionally to each class; 4) SMALLRESIDUAL: Allocation of the missing's in the class that generates the smallest residue; 5) USEINSEARCH: Allocation of missing's as an isolated class.
- 11) SEED: Seed for the training sample. At each iteration, a new sample is generated considering this seed and the training proportion.
- 12) TRAINPROPORTION: Training ratio. At each iteration, a training sample of k percentage is used to train each tree. It is possible to use the option TRAINN, where you can set the number of registers.
- 13) Huber: Parameter for changing the loss function for HUBER. Only valid for continuous target variable.
- 14) SHRINKAGE: Learning algorithm rate. It is the percentage that should be considered for learning. The default value is 0.2. Beware of using this parameter, high values increase the risk of overfitting.
- 15) INPUT: Vector of variables for training. It is possible to declare three different types of variables such as INTERVAL: Continuous variables, NOMINAL: Nominal variables, ORDINAL: Ordinary variables.
- 16) TARGET: Target variable. It is possible to declare four different types, being the same three types of the input, also add the BINARY type.
- 17) SUBSERIES: Number of iterations used by the tree model. (Depth of the tree) parameter options are: 1) LONGEST: Longest series, 2) BEST: Smaller series with the smallest residue, 3) ITERATIONS or NLEAVES: User indicates the number of iterations. Beware: Very long series increase the risk of overfitting.
- 18) ASSESS VALIDATA: Indication of the dataset for the validation of the training, the algorithm will test the decisions based on validation and may or may not change the decision to improve the estimation. It is an important parameter to control overfitting evaluation.
- 19) CODE FILE = Parameter to generate a SAS code for the application model.
- 20) SAVE: Set of parameters to save the outputs of the procedure. The main options are: 1) MODEL: Dataset for application of the model. (Inmodel function); 2) FIT: Dataset with statistics of each iteration; 3) IMPORTANCE: Dataset with the importance of the variables for the development and the validation bases; 4) NODESTATS: Dataset with statistics of each node; 5) RULES: DataSet with the rules used.

The parameters: DECISION, FREQ, MAKEMACRO, PERFORMANCE will not be explored on this paper, but its definition is available at (SAS Institute Inc. 2016).

As you see below there are two examples of using the TREEBOOST procedure. The first example presents a fitting and the second a model application.

1) TREEBOOST fitting example

```
PROC TREEBOOST DATA=DATA TRAINING
     CATEGORICALBINS=20
      INTERVALBINS=20
      EXHAUSTIVE=8000
      INTERVALDECIMALS=MAX
     LEAFSIZE=500
     MAXBRANCHES=20
      ITERATIONS=100
     MINCATSIZE=20
     MISSING=USEINSEARCH
      SEED=13022018
      SHRINKAGE=0.15
     TRAINPROPORTION =0.5
      ;
      INPUT X1 X2 X3 X4 X5 / LEVEL=INTERVAL;
      INPUT X6 X7 X8 X9 / LEVEL=NOMINAL;
      TARGET TARGET / LEVEL=BINARY;
      SUBSERIES BEST;
      ASSESS VALIDATA=DATA_VALIDATION;
      CODE FILE="/SGF/GBM.SAS" NOPREDICTION;
      SAVE MODEL=GBM IMPORTANCE=IMPORTANCE;
```

RUN;

2) TREEBOOST apply example

PROC TREEBOOST INMODEL=GBM; SCORE DATA=DATA_NEW OUT=BKT PREDICTION; RUN;

With the new SAS ® distribution, SAS Viya ®, a new procedure has been made available for fitting Gradient Boosting, called: PROC GRADBOOST, its algorithm and operation are similar to the procedure TREEBOOST, but some improvements have been implemented in it, which we can highlight: 1) CrossValidation k-FOLD: Fitting and automatic test in k-samples; 2) Regularization L1 (LASSO) and L2 (RIDGE); These parameters are available in the AUTOTUNE and CROSSVALIDATION options. Full procedure detail is available on the Data Mining and Machine Learning Procedures documentation (SAS Institute Inc. 2018).

STEP BY STEP: FITTING GRADIENT BOOSTING MODEL

To fit a model using the Gradient Boosting method, some steps and cautions should be observed. Like any other regression or Machine Learning method, it is necessary to evaluate some requirement to obtain success in the estimation. Because it is an exhaustive method where the analyst does not modify the internal steps, it does not mean that the process is entirely automatic; some concerns must be taken to ensure that the model is correctly estimated, just as we deal with the fitting of a logistic regression as an example.

Here are some steps you can take in the process of fitting a model by Gradient Boosting.

SAMPLING

This initial phase is extremely important for the process. Cross-validation (K-Fold) is widely used as best practice sampling with Machine Learning method. The process consists of splitting the dataset into K equal parts. The algorithm will be trained K times on K-1 data subset, where one distinct K subset is used as test sample on each K round or iteration. Another widely used sampling process is 60-20-20, where 60% is used as training sample, and two samples of 20% each, is used for validation and testing.

Another important point is the balance regarding the target variable, performed tests show that the results obtained in databases where the event was balanced in 50% (oversampling or undersampling), have better results than when the actual proportion was used.

For small samples or low default, be careful with the fitting. The Gradient Boosting model for these cases has a high risk of overfitting. Due to the low number of records, the model can become very specific (identification of individuals rather than characteristics) and thus reduce its accuracy on test bases.

TREATMENT OF VARIABLES

Before starting the fitting of a gradient boosting model, it is necessary to evaluate and apply some treatments in the variables to ensure the stability of the model. Because it is an additive tree model, the change in the stability of a variable can cause serious instabilities in the application of the model in the future. The following are the main recommended treatments and evaluation:

1) Missing treatment: The TREEBOOST procedure offers options for the treatment of missing's, but a punctual treatment can yield better results, where the interpretation of the analyst can allocate the missing value in the correct class.

2) Treatment of variables: The application of outliers correction (procedure available in the macro in the next topic) and stability evaluation of the variable in time, helps the decision to keep the variable in the training or if it needs to be treated. For example: Instead of using the variable's continuous form, we categorize (create bins) it and use its bins as a nominal or ordinal variable.

GRADIENT BOOSTING FITTING

The model adjustment process consists in evaluating the behavior of the variables that are composing the model, looking for the best parameterization and stability of the models in different samples. A macro procedure that generates some indicators for easy visualization of the fit, containing the characteristics of the fit and the main indicators for each base will be presented in the next topic. We can summarize the fitting process in the following steps:

- 1) Initial setting, with all available variables and fixed SHRINKAGE, iterations parameters.
- 2) After round one, the variable evaluation process is initiated, removing or treating the variables that have an expressive loss of importance between the training and validation samples.
- 3) After stabilizing the variables, the search of the best parameters is started, where the main five parameters will be changed:
 - a. SHRINKAGE = Learning rate. It is necessary to test the best rate. If it is observed significant differences on KS, ROC and Gini values between the development and test samples, there is a strong indication that the increase in the learning rate is generating an overfitting. Evaluating the difference of these indicators on out-of-time sample is a best practice.
 - b. Iterations: Maximum number of iterations. The TREEBOOST procedure indicates how many iterations were actually used to fit the model (process log). If the number of iteration used is the same one that was set up, more iterations may be required. Beware, too many iterations can also generate an overfitting in the fit.
 - c. SUBSERIES: Depth tree parameter. The recommendation here is to test with the BEST and LONGEST options;
 - d. TRAINPROPORTION: percentage of sample for training. Usually we begin the fitting using a sample of 50% of the training database, but changes in this proportion can generate better results.
 - e. LEAFSIZE: Minimum number of individuals in a node. Too small values can lead to overfitting, but too high values can result underfitting. It is important to understand the number of individuals and perform some tests to observe an ideal value.
- 4) Application of the model generated in test and out-of-time samples to evaluate stability.

MACRO PROCEDURE TO FIT GRADIENT BOOSTING USING PROC TREEBOOST

To assist the model estimation process using the Gradient Boosting methodology, some routines were built to facilitate and standardize the process. As described in STEP BY STEP: Fitting Gradient boosting model, some controls and concerns are needed to get the best of the methodology. The three macro procedures that will be described below, refer to the support of the fitting and performance test of the models.

DATA PREPARATION

Macro procedure responsible for a basic data treatment, where the main indicators such as percentiles, range, minimum and maximum values of each variable are observed. For this evaluation, the univariate procedure was used and the following treatments are performed:

- 1) Identification of variables that did not participate in the fitting.
- 2) Evaluation of variability and completeness of variables.

3) Capping application P1 / P99 for outlier's treatment. The application of capping consists of replacing all values less than percentile 1 (P1) as P1 value, and replacing all values greater than percentile 99 (P99) as P99 value.

Note that the procedure provides fast attributes evaluation and outliers treatment, but it doesn't treat missing values and monotonicity constraints. It should be evaluated manually by the analyst, as described in the item step by step chapter.

FITTING GRADIENT BOOSTING

Macro procedure responsible for assisting the fitting of the Gradient Boosting process. The procedure fits the model based on the main hyper-parameters and presents the results in a simple way. Characteristics of this macro are:

1) As a prerequisite, the macro requests that the training database contains a variable that indicates the split between development sample (DEV) and validation sample (OOV) must exist. It is possible to include other samples for the stability test, but it will not be used in the training.

2) Calculation of LEAFSIZE and MINCATSIZE parameters based on the proportion of training sample records.

3) Outputs to facilitate fit evaluation. List of variables sorted by importance in training and validation samples.

4) Stability evaluation of fit in the samples provided with Kolmogorov-Smirnov (KS), AUROUC and Gini metrics.

5) Creation of backtest code, including capping treatments for application of the model in other databases.



The figure 01 is an example of procedure output.

Figure 1 - Procedure Output

METRICS

It is a macro procedure responsible for calculations of kolmogorov-smirnov (KS), AUROUC and Gini metrics for each given sample. The procedure generates comparison charts to make it easier to see results.

The macro procedures are available in Appendix 1.

REAL CASES

For the evaluation and comparison between logistic regression and gradient boosting methods, three real cases were selected. All cases already have a logistic regression solution developed and the models were fitted using the gradient boosting methodology following the steps suggested in the previous chapters, and also ensuring that the data source was the same available when fitting regression model.

CASE 01 - FRAUD

Goal: Fraud identification based on information from TextMining, using set of attributes of behavior variables (contract data) and TextMining in the description.

Effort: Logistic Regression: Three days / Gradient Boosting: One day;

Gradient Boosting fitting process: In order to fit this scenario, it was considered the following parameters in the steps mentioned in the previous chapters: 1) Sample of 60-20-20 + Out of Time; 2) Evaluation of variables, with correction of missing and capping; 3) Stress of parameters: Shrinkage: 0.11, Trainproportion: 0.6, Leafsize: 6%, SUBSEERIES: longest, Iterations: 200. During the fitting of GBM, it was necessary to treat two variables to correct their relationship of linearity, and thus to obtain better results.

Main results:

			Pasa	Gradient Boosting			Logistic Regression		
Variables		Dase	KS	GINI	ROC	KS	GINI	ROC	
Tumo (Close Mariahlas	Logistic	Gradient	All DataSet	22.1%	30.5%	65.2%	21.6%	30.1%	65.0%
Type/class valiables	Regression	Boosting	Trainning	22.6%	31.5%	65.7%	22.1%	30.6%	65.3%
Event/ Client Identification	2	2	Out of Sample	21.5%	29.6%	64.8%	22.2%	30.0%	65.0%
Text Mining - Description of Client	17	17	Out of Time	22.1%	29.7%	64.8%	21.5%	29.3%	64.6%
·			Out of Validation	21.2%	30.0%	65.0%	22.0%	30.4%	65.2%



Table 1 - Case 01 – Variables and Results

Figure 2 - Main Results

As we can observe the fit considering the logistic model and gradient boosting for this scenario, resulted in a very similar performance. The set of attributes selected has high correlation and the same number of predictor variables (19). In Fig. 02, it could be observed that the Gradient Boosting method is stable in most part of the vintages and samples. It turns out that the models are basically the same for the ordering of the event, depending on the decision of cut-off. For example, considering 70% of best scores, we have 24.8% frauds considering the logistic regression and 24.4% considering the gradient boosting, with a reduction of 1.5% of fraud events, in the out-of-time period we observed the same cut-off point, 23.0% for logistic and 22.8% for gradient, with a reduction of 1.0% of the events.

CASE 02 – CREDIT SCORE - BUSINESSES

Goal: Default identification, using set of attributes of: Negative information, inquiry, client behavior and partners.

Effort: Logistic Regression: Four days / Gradient Boosting: two hours

Gradient Boosting fitting process: In order to fit this scenario, it was considered the following parameters in the steps mentioned in the previous chapters: 1) Sample of 70-30 + Out of Time; 2) Evaluation of variables, with correction of missing and capping; 3) Stress of parameters: Shrinkage: 0.12, Trainproportion: 0.6, Leafsize: 5%, SUBSEERIES: longest. Iterations: 100. During the fitting of GBM, did not present instabilities in the variables, which reduced development time, since it was not necessary to treat the variables.

Main results:

Variables		Deve		Gradient Boosting			Logistic Regression			
Type/Class Variables	Logistic Regression	Gradient Boosting		Base	KS	GINI	ROC	KS	GINI	ROC
Client Identification	3	1		All DataSet	54.3%	70.0%	85.0%	51.8%	66.8%	83.4%
Partners	5	17		Trainning	55.6%	71.9%	86.0%	51.7%	66.8%	83.4%
Negative information	5	17		Out of Time	53.1%	68.1%	84.1%	52.0%	66.6%	83.3%
Inquiry	3	11		Out of Sample	54.2%	69.3%	84.7%	52.7%	67.3%	83.7%



Table 2 - Case 02 – Variables and Results

Figure 3 - Main Results

As we can see, there is a large effort decrease in this scenario when using Gradient Boosting method. For the regression model, it was studied several iterations among the variables to search for the best performance. In the Gradient Boosting process, this manual study was not needed. When we observe the results in the development and Out-Of-Time samples, it is possible to evaluate that the gradient boosting method performs a little better than the regression method. Considering a cut-off point at 70% of approval, we would have a default of 2.8% using the logistic regression and 2.6% using Gradient Boosting, generating a reduction of 8.7% in default. Considering the out-of-time sample when using same cut-off point, we have 3.0% for logistic and 2.8% for gradient, generating a 5% reduction in default. When we observe the stability of the models, the methodology of Gradient Boosting was stable and with better accuracy in all vintage observed. Evaluating the implementation effort, we have 16 variables that make up the regression model against 46 in Gradient, but in this case, we must consider that the implementation of the gradient model must be automatized to be feasible.

CASE 03 – COLLECTION SCORE - CONSUMER

Goal: Debt collection, using set of attributes of: Negative information, inquiry and client behavior.

Effort: Logistic Regression: Six days (two equations) / Gradient Boosting: 2 hours (One model)

Gradient Boosting fitting process: In order to fit this scenario, it was considered the following parameters in the steps mentioned in the previous chapters: 1) Sample of 60-20-20 (Event Balanced – Undersampling – 50%) + Out of Time; 2) Evaluation of variables, with correction of missing and capping; 3) Stress of parameters: Shrinkage: 0.10, Trainproportion: 0.6, Leafsize: 5%, SUBSEERIES: best. Iterations: 100. During the fitting of GBM, did not present instabilities in the variables, which reduced development time, since it was not necessary to treat the variables. The fitting of GBM does not consider a segmentation. In the logistic regression solution, this scenario has two segmentations.

Main results:

Variables					
Type/Class Variables	Logistic R	Gradient			
	Segment 01 Segment 02		Boosting		
Client Identification	4	6	4		
Scores	2	2	2		
Negative information	23	17	34		
Inquiry	4 1		4		
Total Distinct variables	4				

		Gradient Boosting								
Pasa		Total			Segment 01			Segment 02		
Dase	KS	GINI	ROC	KS	GINI	ROC	KS	GINI	ROC	
All DataSet	52.9%	70.5%	85.2%	42.8%	56.6%	78.3%	44.4%	59.2%	79.6%	
Trainning	55.6%	73.1%	86.6%	47.5%	62.3%	81.1%	48.3%	63.2%	81.6%	
Out of Sample	53.2%	70.6%	85.3%	42.7%	56.7%	78.3%	44.4%	59.3%	79.6%	
Out of Time	50.1%	68.0%	84.0%	41.1%	53.8%	76.9%	43.1%	57.3%	78.6%	
Out of Validation	53.8%	71.0%	85.5%	41.6%	55.1%	77.5%	45.6%	60.1%	80.0%	
				Lo	gistic Regress	ion				
Pasa		Total		Segment 01			Segment 02			
Dase	KS	GINI	ROC	KS	GINI	ROC	KS	GINI	ROC	
All DataSet	51.8%	69.3%	84.7%	42.7%	57.0%	78.5%	40.7%	54.4%	77.2%	
Trainning	52.1%	69.9%	85.0%	44.6%	58.8%	79.4%	42.5%	55.3%	77.7%	
Out of Sample	52.1%	69.5%	84.8%	43.0%	57.2%	78.6%	40.9%	54.5%	77.3%	
Out of Time	50.5%	67.7%	83.9%	41.6%	55.6%	77.8%	39.6%	53.4%	76.7%	
Out of Validation	51.4%	69.4%	84.7%	40.4%	54.9%	77.4%	43.6%	55.2%	77.6%	

Table 3 - Case 02 – Variables and Results – Total



Figure 4 - Main Results - Total



Figure 4 - Main Results – Segment 01



Figure 4 - Main Results – Segment 02

In this scenario, we have a different characteristic from the others; the logistic solution is composed of two equations with 33 variables and 26 variables each, with 45 distinct variables. For the comparison of results with the Gradient Boosting methodology, we opted for the development of a non-segmentation model, to evaluate the gradient process ability to handle multiple characteristics on same sample. When we observe the results applied in whole population, we can evaluate that the GBM methodology reached the same results of the regression, maintaining the same performance and sorting, but with a much smaller development effort.

Observing the performance of the model separated by the initial proposed segmentations in the logistics solution, it can be observed that for segment 01 we have similar results and for the Out-of-time sample we have a loss of 0.5 points of KS in relation to the logistic model. In SWAP analysis, there is no difference for cut-off points above 20%.

For segment 02, there is a gain of 5.8 points of KS for the training sample and 3.6 points of KS for the Out-Of-Time sample for the Gradient Boosting methodology. When we observe the SWAP analysis, we observe performance superiority in all range of cut-off points using Gradient Boosting model.

CONCLUSION

As we can see, the gradient boosting method is an alternative to the traditional method of logistic regression, bringing speed to the development of scoring solutions. It is a very robust method that has high flexibility for solving classification problems. Besides we only presented cases with binary target variable in this work, gradient boosting methodology is easily applied to problems with multinomial or interval target variables.

The main factors that Gradient Boosting methodology generates greater or equal results if compared with logistic regression, with less development effort, can be explained by two characteristics:

1) Interaction between variables: boosting process build interactions between variables, providing multiple combinations to further event explanation.

2) Interpretation of parameters: gradient boosting is a process of multiple models, and the interpretation of the variables is not straight, so it is unusual to remove a variable because it presents inverse interpretation of the business sense. In another hand, it is very common exclude variables when fitting logistic regression because this reason, even when it decrease overall model accuracy and prediction power.

The main points of attention that we should have in mind when applying Gradient boosting methodology are:

1) Implementation environment: The methodology can be extremely complex for implementation, because it has multiple rules. In order to enable its use it is necessary to have an infrastructure where an automatic model deployment is possible, because in the same proportion as the methodology reduces development time, it exponentially increases the implementation time if manual coding is needed, which in most cases ends up making the model unfeasible to be used.

2) Stability / Lifetime: The GBM methodology, because of its rapid development, aims to be re-estimated more often than logistic regression. Ensuring stability and better accuracy.

The table below is an adaptation of the comparison made by (Di Cellio Dias, Paulo, Witarsa, Marc, 2017), and presents a brief comparison between the methodologies separated by phase of the model fitting process.

	0.11.1	La statia Da sus astau	Out diaut Departies			
Subject		Logistic Regression	Gradient Boosting			
Outliers treatments		Needs treatment	Needs treatment			
	Multicollinearity - Analysis and removal	Needs treatment	It is not necessary			
Variable Transformation Fit / Parameters Interpretation		Most part of variables needs transformation (functions or bins)	The Process performs transformations automatically (Decision trees) / Some variables can be transform			
		Needs evaluation of significance and interpretation of the parameter	Interactive process. It is not possible to evaluate each parameter			
	Sampling Test	Need to verify performance on hold-out and out of time data	Need to verify performance on hold- out and out of time data. Higher risk of overfitting as used many interactions			
Apply models Effort		Easy deployment	Deployment can be difficult because of the large number of interactions and variables.			
		From 3 to 5 days considering the use of support procedures for the removal of multicollinearity and categorization of the variables	Some hours. The process is interactive, it is only necessary to perform some simulations to identify the best parameters (Number of interactions, learning rate and variables)			

Table 4 - Comparison

Observing the studies and results obtained during the development of this work, we can summarize some pros and cons of each methodology:

	Logistic Regression	Gradient Boosting		
	Direct Parameters Interpretation	Fast Development		
	Easy deployment	Stress of parameters		
Proc	Creater stability over time	Flexibility (Target : Binary / Multinomial		
FIUS	Greater stability over time	/ Interval)		
	Business view			
	Market Confidence			
	-			
	Development effort	Needs implementation environment		
Conc	Manual Fit (Iterations)	New Methodology for the Market		
COIIS		Low variable interpretation		

Table 5 – Pros and Cons

REFERENCES

SAS Institute Inc 2016. SAS Enterprise Miner and SAS Text Miner Procedures Reference for SAS 9.4®. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2018. "Data Mining and Machine Learning Procedures." Accessed February 13, 2018. http://documentation.sas.com/?docsetId=casml&docsetVersion=8.1&docsetTarget=viyaml_gradboost_ overview.htm&locale=pt-BR

Friedman, Jerome H. Greedy function approximation: A gradient boosting machine. Ann. Statist. 29 (2001), no. 5, 1189--1232. doi:10.1214/aos/1013203451.

Di Cellio Dias, Paulo; Witarsa Marc. 04/2017. Revenue Score - Forecasting Credit Card Products with Zero Inflated Beta Regression and Gradient Boosting. Orlando. SAS Publisher. Available at http://support.sas.com/resources/papers/proceedings17/0847-2017.pdf

APPENDIX 1 – PROCEDURES SOURCE CODES

```
*** SAS GLOBAL FORUM 2018
                                           ***;
*** 1857-2018-SGF - A Comparison of Gradient Boosting with Logistic
                                           *** :
Regression in
                                           ***:
*** Practical Cases
*** PROCEDURE: DATA PREPARATION
                                           ***:
***______
***+-----------------+***;
*** | PAULO DI CELLIO | MAR/2018
                                           ***;
*** MELISSA FORTI MAR/2018
                                          ***;
*** MARC WITARSA MAR/2018
                                          ***:
%MACRO DATA_PREPARATION(BASE,TARGET,SAMPLE,DIR,NAME);
   OPTIONS NONOTES NOMPRINT NOMLOGIC;
   ODS NORESULTS;
   %PUT >> DATA PREPARATION;
   %PUT;
   DATA AJUSTE GBM;
       SET &BASE.;
   RUN;
   *****
   * CONTROL VARIABLES
   PROC CONTENTS DATA=AJUSTE_GBM NOPRINT
      OUT=BASE GBS MC(KEEP=NAME TYPE FORMAT);
   RUN;
   *SPECIAL VARIABLES IDENTIFICATION;
   DATA BASE_GBS_MC;
       SET BASE GBS MC;
       LENGTH COD_STATUS 8.
                  $500.;
             STATUS
       IF FORMAT IN ('DDMMYY' 'YYMMDD') THEN DO;
               COD_STATUS = 0;
               STATUS = "00 - DATE";
           END;
           ELSE IF UPCASE(NAME) EQ UPCASE("&TARGET.") THEN DO;
                   COD STATUS = 0;
                   STATUS = "00 - TARGET";
               END;
               ELSE IF NAME IN (& SPECIAL_VARS.) OR NAME IN
               ("&SAMPLE.") THEN DO;
                   COD STATUS = 0;
                   STATUS = "00 - PROCESS VARIABLES";
               END;
   RUN;
       *****
```

* DESCRIPTIVE ANALYSIS AND CAPPING *

```
PROC UNIVARIATE DATA=AJUSTE_GBM(WHERE=(&SAMPLE. EQ 'DEV'))
OUTTABLE=DADOS_DESC NOPRINT;
QUIT;
DATA DADOS_DESC;
     SET DADOS DESC;
     *COMPLETENESS;
     COMPL = NOBS / ( NMISS + NOBS );
     RENAME VAR = NAME;
RUN;
*UPDATE DATASET;
PROC SORT DATA=DADOS_DESC ;
     BY NAME;
RUN;
PROC SORT DATA=BASE_GBS_MC ;
     BY NAME;
RUN;
DATA BASE GBS MC;
     MERGE BASE_GBS_MC (IN=A) DADOS_DESC;
     LENGTH CAP_MIN CAP_MAX $200.;
     BY NAME;
     IF A;
     *CAPPING P1/ P99;
     IF COD_STATUS EQ . AND TYPE EQ 1 THEN DO;
           CAP_MIN = COMPBL("IF NOT MISSING("|NAME||") THEN
     "||NAME||" = MIN("||NAME||","||_P99_||");");
           CAP_MAX = COMPBL("IF NOT MISSING("||NAME||") THEN
     " | NAME | | " = MAX(" | NAME | | ", " | P1 | | ");");
     END;
RUN;
*REPORT PRINT;
PROC SORT DATA=DADOS_DESC ;
     BY NAME;
RUN;
ods TAGSETS.EXCELXP file="&DIR.\01-DESCRIPTIVE ANALYSIS &NAME..XLS"
   style=SASWEB;
ODS TAGSETS.EXCELXP OPTIONS(SHEET_NAME="DESCRITIVA"
SHEET INTERVAL='NONE' ZOOM='85' AUTOFILTER='ALL' AUTOFIT HEIGHT = 'YES'
);
PROC PRINT DATA=BASE GBS MC LABEL NOOBS;
     VAR _ALL_;
RUN;
ODS TAGSETS.EXCELXP CLOSE;
*CAPPING PRINT;
%PUT >> CAPPING;
%PUT;
DATA _NULL_;
     SET BASE_GBS_MC;
     WHERE COD_STATUS EQ .;
     FILE "&DIR.\01_CAPPING_&NAME..SAS";
     IF _N_ EQ 1 THEN DO;
           PUT "*CAPPING (P1-P99);";
     END;
     PUT CAP MIN CAP MAX ;
RUN;
DATA AJUSTE GBM;
```

```
SET AJUSTE_GBM;
%INCLUDE "&DIR.\01_CAPPING_&NAME..SAS";
RUN;
PROC DATASETS LIB=WORK NOLIST;
DELETE LST_SELECAO VETOR_INTERGER VETOR_NOMINAL VETOR_VARS;
QUIT;
OPTIONS NOTES MPRINT MLOGIC;
ODS RESULTS;
%MEND DATA_PREPARATION;
```

```
***;
*** SAS GLOBAL FORUM 2018
*** 1857-2018-SGF - A Comparison of Gradient Boosting with Logistic
                                     * * * :
Regression in
*** Practical Cases
                                     ***;
*** PROCEDURE: GRADIENT BOOSTING
                                     ***:
***______
*** | PAULO DI CELLIO | MAR/2018 | |***;
***+-----+---+----+----+----+-----++**;
*** | MELISSA FORTI | MAR/2018 |
                                    ***;
***+-------+-----+-----+-----+***;
*** MARC WITARSA | MAR/2018 |
                                    ***;
***+-------+----+-----+-----+-------+***;
```

```
%MACRO FIT GBM(BASE, TARGET, SAMPLE, NUM INTS, CAPPING, DIR, NAME);
     OPTIONS NOMPRINT NOMLOGIC NOSYMBOLGEN NOSOURCE;
     *NUMBER OF VARIABLES;
     %LET NUM VARS = %SYSFUNC(COUNTW((&VARS INTERVAL.||&VARS NOMINAL.)));
     *LEAFSIZE CALCULATION;
     PROC SQL NOPRINT;
           SELECT INT(COUNT(*)*&PCT REG NO.*&TRAINPROPORTION.) INTO:
                                                               LEAFSIZE
           FROM &BASE.(WHERE=(&SAMPLE. EQ 'DEV'))
           ;
     OUIT;
     TITLE ;
     %PUT >> GRADIENT BOOSTING FITTING / &NUM_VARS. VARIABLES - &NUM_INTS.
            INTERACTIONS;
     *****
     * MODEL FITTING
                                                   *
     PROC TREEBOOST
           DATA=&BASE.(WHERE=(&SAMPLE. EQ 'DEV'))
           CATEGORICALBINS=&CATEGORICALBINS.
           INTERVALBINS=&INTERVALBINS.
           EXHAUSTIVE=&EXHAUSTIVE.
           INTERVALDECIMALS=&INTERVALDECIMALS.
           LEAFSIZE=&LEAFSIZE.
           MAXBRANCHES=&MAXBRANCHES.
           ITERATIONS=&NUM INTS.
           MINCATSIZE=&LEAFSIZE.
```

```
MISSING=&MISSING.
     SEED=&SEED.
     %IF %SYSFUNC(UPCASE(&TLEVEL.)) EQ INTERVAL %THEN %DO;
           HUBER=&HUBER.
     %END;
     SHRINKAGE=&SHRINKAGE.
     TRAINPROPORTION = & TRAINPROPORTION.;
     INPUT &VARS_INTERVAL. / LEVEL=INTERVAL;
     INPUT &VARS NOMINAL.
                          / LEVEL=NOMINAL;
     TARGET & TARGET. / LEVEL=& TLEVEL.;
     SUBSERIES & TYPE_AJUSTE.;
     ASSESS VALIDATA=&BASE.(WHERE=(&SAMPLE. EQ 'OOV'));
     CODE FILE="&DIR./GBM_&NAME._NINT_&NUM_INTS._NVARS_&NUM_VARS..SAS"
              NOPREDICTION;
     SAVE MODEL=GBM_&NUM_INTS._&NUM_VARS. IMPORTANCE=IMPORTANCE;
RUN;
QUIT;
TITLE "GRADIENT BOOSTING FITTING ";
ods TAGSETS.EXCELXP file="&DIR.\03-GRADIENT BOOSTING FITTING
                                         &NAME..XLS" style=SASWEB;
ODS TAGSETS.EXCELXP OPTIONS(SHEET NAME="MODEL VARIABLES"
SHEET_INTERVAL='NONE' ZOOM='85' AUTOFILTER='ALL' AUTOFIT_HEIGHT =
'YES');
PROC PRINT DATA=IMPORTANCE NOOBS ;
     VAR NAME NRULES IMPORTANCE VIMPORTANCE RATIO;
RUN;
TITLE;
* APPLY
PROC TREEBOOST INMODEL=GBM_&NUM_INTS._&NUM_VARS.;
     SCORE DATA=&BASE. OUT=SCORE_GBM PREDICTION;
RUN;
%IF %SYSFUNC(UPCASE(&TLEVEL.)) EQ BINARY %THEN %DO;
     DATA SCORE GBM;
           SET SCORE_GBM;
           SCORE GBM=INT(P &TARGET.0*1000);
     RUN;
%END;
DATA _NULL_;
     FILE
     "&DIR./GBM_&NAME._NINT_&NUM_INTS._NVARS_&NUM_VARS._BACKTEST.SAS";
     *";
           "* GRADIENT BOOSTING
     PUT
           "* SGF - 2018
     PUT
                                     *";
           PUT
     PUT;
     PUT;
RIIN;
%IF & CAPPING. EQ 1 %THEN %DO;
     DATA BASE_CAP;
           INFILE "&DIR.\01_CAPPING_&NAME..SAS" TRUNCOVER DSD
      LRECL=32767;
          LENGTH STR $500.;
           TNPUT
                @1 STR $500.;
           LENGTH NAME $32.;
```

```
NAME = SCAN(STR, 4);
     RUN;
     PROC SORT DATA=IMPORTANCE OUT=VARS(KEEP=NAME);
          BY NAME;
     RUN;
     PROC SORT DATA=BASE CAP ;
          BY NAME;
     RUN;
     DATA BASE CAP;
          MERGE BASE_CAP (IN=A) VARS (IN=B);
          BY NAME;
          IF B;
          FILE "&DIR./GBM_&NAME._NINT_&NUM_INTS._NVARS_&NUM_VARS.
                                           _BACKTEST.SAS" MOD;
          IF _N_ EQ 1 THEN DO;
               PUT "*CAPPING (P1-P99);";
          END;
          PUT STR ;
     RUN;
%END;
DATA _NULL_;
     FILE "&DIR./GBM_&NAME._NINT_&NUM_INTS._NVARS_&NUM_VARS.
                                      _BACKTEST.SAS" MOD;
     INFILE "&DIR./GBM &NAME. NINT &NUM INTS. NVARS &NUM VARS..SAS";
     INPUT;
     IF _N_ EQ 1 THEN DO;
          PUT;
          PUT "* MODEL
                                   * "
          PUT;
     END;
     PUT;
     PUT @4 _INFILE_;
RUN;
%IF %SYSFUNC(UPCASE(&TLEVEL.)) EQ BINARY %THEN %DO;
     DATA NULL ;
          FILE "&DIR./GBM_&NAME._NINT_&NUM_INTS._NVARS_&NUM_VARS.
                                          _BACKTEST.SAS" MOD;
          PUT "*SCORE - GBM;";
          PUT 'SCORE GBM = INT(P '"&TARGET."'0*1000);';
     RUN;
%END;
* EVALUATION
                                   *
PROC FREQ DATA=SCORE_GBM NOPRINT;
     TABLES & SAMPLE. / OUT=TESTE_AMOSTRAS;
OUIT;
DATA TESTE_AMOSTRAS;
     SET TESTE_AMOSTRAS END=EOF;
     LN=_N_;
     IF EOF THEN CALL SYMPUT("TOTA", LN);
RUN;
PROC DATASETS LIB=WORK NOLIST;
     DELETE METRICAS GBS;
QUIT;
```

```
%DO I=1 %TO &TOTA.;
      DATA _NULL_;
            SET TESTE_AMOSTRAS;
            WHERE LN EQ &I.;
            CALL SYMPUT("VBASE", & SAMPLE.);
      RUN;
      %IF %SYSFUNC(UPCASE(&TLEVEL.)) EQ BINARY %THEN %DO;
            %KS_ROC_GINI(
            SCORE_GBM(WHERE=(&SAMPLE. EQ "&VBASE.")),
            &TARGET.,
            SCORE_GBM
            );
            DATA TMETRICAS;
                  LENGTH KS GINI ROC N_REG GOODS BADS 8. MODEL $100.
                                                            SAMPLE $3.;
                  KS = \& KS2.;
                  GINI=&GINI.;
                  ROC=&ROC.;
                  EVENT=&P.;
                  N_REG=&N_REG.;
                  GOODS = \& BONS.;
                  BADS = \&MAUS.;
                  SAMPLE = "&VBASE.";
                  MODEL = "GBM_&NAME._NINT_&NUM_INTS.
                                     _NVARS_&NUM_VARS.";
            RUN;
            PROC APPEND DATA=TMETRICAS BASE=METRICAS GBS;
            OUIT;
      %END;
      %ELSE %DO;
      *INTERVAL TARGET;
      PROC CORR DATA=SCORE_GBM(WHERE=(&SAMPLE. EQ "&VBASE.")) OUTS=CORS
                                                                 NOPRINT;
            VAR P & TARGET.;
            WITH &TARGET.;
      RUN;
      DATA NULL ;
            SET CORS;
            IF _TYPE_ EQ "MEAN" THEN CALL SYMPUT("MEAN", P_&TARGET.);
            ELSE IF _TYPE_ EQ "STD" THEN CALL SYMPUT("STD", P_&TARGET.);
            ELSE IF TYPE EO "N" THEN CALL SYMPUT("N", P & TARGET.);
            ELSE IF _TYPE_ EQ "CORR" THEN
            CALL SYMPUT("CORR", P_&TARGET.);
      RUN;
      DATA TMETRICAS;
            LENGTH SPEARMAN MEAN STD N_REG 8. MODELO $100. SAMPLE $3.;
            SPEARMAN=&CORR.;
            MEAN=&MEAN.;
            STD=&STD.;
            N_REG=&N.;
            SAMPLE = "&VBASE.";
            MODEL = "GBM_&NAME._NINT_&NUM_INTS._NVARS_&NUM_VARS.";
      RUN;
      PROC APPEND DATA=TMETRICAS BASE=METRICAS GBS;
      QUIT;
      %END;
%END;
```

```
TITLE "RESULTS";
ODS TAGSETS.EXCELXP OPTIONS(SHEET_NAME="RESULTS" SHEET_INTERVAL='NONE'
ZOOM='85' AUTOFILTER='ALL' AUTOFIT_HEIGHT = 'YES');
PROC PRINT DATA=METRICAS_GBS NOOBS ;
      VAR _ALL_;
RUN;
ODS TAGSETS.EXCELXP CLOSE;
TITLE "";
*GRÁFICOS DE COMPARAÇÃO;
Axis1
STYLE=1
WIDTH=1
MINOR=
(NUMBER=1
)
;
Axis2
      STYLE=1
      WIDTH=1
;
TITLE;
%IF %SYSFUNC(UPCASE(&TLEVEL.)) EQ BINARY %THEN %DO;
      TITLE1 "KS";
      PROC GCHART DATA=METRICAS GBS;
            VBAR SAMPLE/
            SUMVAR=KS
            GROUP=MODEL
            CLIPREF
      FRAME TYPE=SUM
      COUTLINE=BLACK
      RAXIS=AXIS1
      MAXIS=AXIS2
      ;
      RUN;
      TITLE1 "AUROC";
      PROC GCHART DATA=METRICAS_GBS;
            VBAR SAMPLE/
            SUMVAR=ROC
            GROUP=MODEL
            CLIPREF
      FRAME TYPE=SUM
      COUTLINE=BLACK
      RAXIS=AXIS1
      MAXIS=AXIS2
      ;
      RUN;
      TITLE "";
%END;
%IF %SYSFUNC(UPCASE(&TLEVEL.)) EQ INTERVAL %THEN %DO;
      TITLE1 "CORRELATION";
      PROC GCHART DATA=METRICAS_GBS;
            VBAR SAMPLE/
            SUMVAR=SPEARMAN
            GROUP=MODEL
            CLIPREF
      FRAME TYPE=SUM
      COUTLINE=BLACK
```

```
RAXIS=AXIS1
         MAXIS=AXIS2
         ;
         RUN;
         TITLE "";
    %END;
    OPTIONS MPRINT ;
%MEND FIT GBM;
*** SAS GLOBAL FORUM 2018
                                                     ***;
*** 1857-2018-SGF - A Comparison of Gradient Boosting with Logistic
                                                     ***;
Regression in
*** Practical Cases
                                                     ***;
*** PROCEDURE: KS/AUROC/GINI
                                                     ***:
***______
***+-----------------+***;
*** | MELISSA FORTI | MAR/2018 | |***;
***+-----+---+----+----+----+-----++**;
*** | MARC WITARSA | MAR/2018 | |***;
***+-----+---+----+----+----+-----++**;
%MACRO KS_ROC_GINI(TABELA, TARGET, SCORE);
    PROC FREQ DATA = & TABELA. NOPRINT;
         TABLES &SCORE.*&TARGET./MISSING OUT = P_KS;
    RUN;
    PROC TRANSPOSE DATA=P KS OUT=P KS(DROP= NAME LABEL ) PREFIX=RESP ;
         BY &SCORE.;
         ID &TARGET.;
         VAR COUNT;
    RUN;
    DATA P_KS;
         SET P_KS;
         IF RESP_0 EQ . THEN RESP_0=0;
         IF RESP_1 EQ . THEN RESP_1=0;
    RUN;
    PROC SUMMARY DATA=P KS NWAY MISSING;
         VAR
             RESP: ;
         OUTPUT OUT=TOT_COL(DROP= _TYPE_ _FREQ_
                               RENAME=RESP_0=RESPC_0
                               RENAME=RESP_1=RESPC_1) SUM=;
    RUN;
    DATA _NULL_;
         SET TOT COL;
         CALL SYMPUT ("RESPC_0", RESPC_0);
         CALL SYMPUT ("RESPC 1", RESPC 1);
    RUN;
    DATA P KS;
         SET P_KS;
         ACM_0=ACM_0+RESP_0;
         ACM_1=ACM_1+RESP_1;
         POR_0=ACM_0/&RESPC_0.;
```

```
POR_1=ACM_1/&RESPC_1.;
            DIFKS=ABS(POR_0-POR_1);
            RETAIN ACM_0 0 ACM_1 0;
      RUN;
      * KS / ROC / GINI;
      DATA MEDIDAS ;
            SET P KS END=EOF;
            RETAIN KS 0 AUC 0 POR_0 0 POR_1 0 ;
            LENGTH ID_BASE $32.;
            IF LAG(POR_1) NE . THEN AUC + ((POR_1 + LAG(POR_1)) * (POR_0 -
                                                          LAG(POR_0)) / 2 ) ;
            KS = MAX(KS,DIFKS);
            GINI = 100*(AUC - 0.5)/0.5;
            ROC = (GINI + 100) / 2;
            IF EOF THEN DO;
                  X=ROUND(KS*100,0.01);
                  Y=ROUND(GINI,0.01);
                  Z=ROUND(ROC,0.01);
                  P=ROUND((&RESPC_1./(&RESPC_1.+&RESPC_0.)),0.0001);
                  N=&RESPC_0.+&RESPC_1.;
                  BONS=&RESPC 0.;
                  MAUS=&RESPC_1.;
                  %GLOBAL KS2 GINI ROC P N_REG BONS MAUS;
                  CALL SYMPUT ("KS2",X);
                  CALL SYMPUT ("GINI",Y);
                  CALL SYMPUT ("ROC",Z);
                  CALL SYMPUT ("P",P);
                  CALL SYMPUT ("N_REG",N);
                  CALL SYMPUT ("BONS", BONS);
                  CALL SYMPUT ("MAUS", MAUS);
            END;
      RUN;
      PROC DATASETS NOLIST;
            DELETE P_KS TOT_COL MEDIDAS;
      QUIT;
%MEND KS_ROC_GINI;
```

RECOMMENDED READING

- SAS® 9.4 Macro Language: Reference, Fifth Edition
- SAS Enterprise Miner® and SAS Text Miner® Procedures Reference for SAS 9.4, Fourth Edition

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Paulo Celio Di Cellio Dias Serasa Experian paulo.dicellio@gmail.com

Melissa Forti Bradesco melissaforti@gmail.com

Marc Witarsa Serasa Experian <u>marc.witarsa@gmail.com</u>