Paper 847-2017

Revenue Score - Forecasting Credit Card Products with Zero Inflated Beta Regression and Gradient Boosting

Paulo Celio Di Cellio Dias, Serasa Experian; Marc Witarsa, Serasa Experian;

ABSTRACT

Using the zero inflated beta regression and gradient boosting techniques (proc nlmixed and proc treeboost), they have developed a solution for gross revenue forecast of credit card products. In this paper will be explored the solution of revenue score forecasting, divided in three sources of revenue: revolving, interchange, and products. Also will be explored the methods zero inflated beta regression and gradient boosting.

Keywords: proc nlmixed, proc treeboost, zero inflated beta regression, gradient boosting, credit cards, invoice variables, revenue.

INTRODUCTION

For gross revenue forecasting of credit card products, the solution that will be explored in this paper considers the use of two methods combined to a process to divide the sources of revenue and create a set of attributes based on invoice information. The aim is to explore the solution, methods and results. The items that will be explored are:

1) Revenue Score: Concept, type of Information, segmentation of the type of revenue, target variables and set of attributes.
2) Zero inflated beta regression: Basic concepts and implementation of regression (proc nlmixed).
3) Gradient Boosting: Basic concepts and implementation of technique (proc treeboost).
4) Case: Main steps used on the project and the main results.
5) Compare Methods: Main differences between methods.

REVENUE SCORE - CONCEPT

The Revenue Score is an important tool to support the business strategies decision. Its function is to classify customers in relation to their gross revenue. With this, it is possible to offer better products, build strategies for growth of utilization of revenue and apply more assertive rates (applying of discounts on services). In this project, we consider the gross revenue estimation and won't estimate the net revenue that is the target of a profit score solution.

Seeing the sources of possible credit card revenues, in order to obtain a better assertiveness in the revenue score solution, we recommend observing the customer's revenue divided into three pillars, as below:

1) Revolving: Fees and interest earned, when the amount payed was less then total credit card amount (invoice), is considered as revolving revenue.
2) Interchange: Percentage of the amount of each credit card transaction earned by the credit card

issuer. The calculation is different for national and international transactions by type of card issuer and type of card product (Gold, Platinum, Black, Infinite, among others). The interchange is also different when the purchase is made by one installment or multiple installments and installments with retail shops (without interest charges) and installments with bank (with interest charges)

3) Products: Credit card contracts can have multiple product options that can be hired. Examples: annual fee, insurance, bill payment and installment invoices. We recommend to create one target variable to each product.

The total gross revenue of the customer can be written summarizing the three pillars:

$$Gross\ revenue = revenue_{revolving} + revenue_{interchange} + revenue_{products}.$$

The solution can be used to observe gross revenue in each pillar or the total revenue of client. The solution development, characteristics of the target variable and predictor variables (set of attributes) are explored in the items below.

TARGET VARIABLE

The target variable is calculated to each source of revenue, where we observe the customer revenue in each of the pillars for a period of 6 or 12 months ahead from reference date. It is calculated the ratio of the total value by the credit limit value at the reference date, transforming target variable in a percentage domain. The revenue observed in each of the pillars (revolving, interchange and products) has similar characteristics, that when we observe the distribution of the target variable. It is possible to identify a zero inflated beta distribution and since we have a concentration in zero due to clients that did not use the service. We have a beta distribution for customers that consumed some value in the observed period.

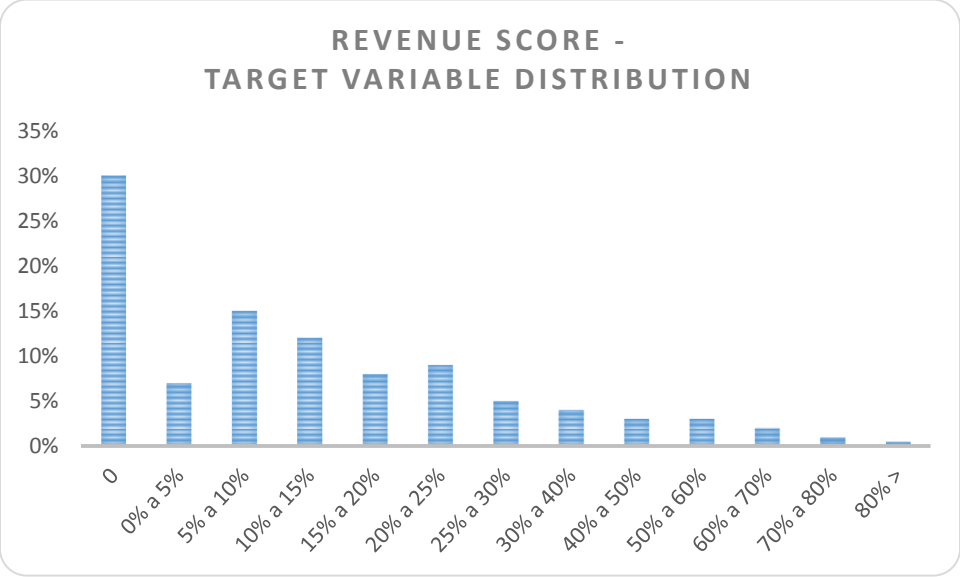

Figure 1. Revenue Score – Target variable distribution

SET OF ATTRIBUITES

For development of these models, it is indicated the creation of a set of attributes based on invoice information. It is recommended to use a history of at least 6 months and below we list some examples of variables which can be created:

A) Balances

1. Ratio of the balances by credit limit in the observed month;
2. Creation of basic behavior variables: sum, mean, coefficient of variation, standard deviation, minimum, median and maximum;
3. Creation of evolution of variables: Trends and comparison in relation to the portfolio;

B) Frequencies

1. Creation of basic behavior variables: sum, mean, coefficient of variation, standard deviation, minimum, median and maximum;
2. Creation of ratio between events from a given period to another.
3. Creation frequencies of payment: Number of total payments, minimum and partial.

C) Other behavior attributes

1. Creation of variables related to period of time and relationship time, number of cards, dependents, location, and credit limit use.

ZERO INFLATED BETA REGRESSION – PROC NLMIXED

One of the options to adjust the revenue score is the use of the zero inflated beta regression. The beta regression is a very versatile regression for the adjustment of many situations due to the flexibility of the beta distribution, but its limits must be ranging between 0 and 1 not included. The zero inflated regression is an upgrade that incorporates the possibility of adjusting the value 0 and allowing the modeling of the entire distribution. The inflated beta regression methodology allows us to work with inflation at zero, one or both tails. In this paper, we will explore only inflation at zero. The zero inflated beta distribution and regression are presented below:

Zero inflated beta distribution:

$$f(y;\pi_0;\mu;\phi) = \begin{cases} \pi_0 = 1/\left(1+\exp\left(-x_i^{'}\delta\right)\right), y = 0 \\ (1-\pi_0) * h(y), \quad 0 < y < 1 \end{cases}$$

Zero inflated beta regression

$$E(Y_i) = \pi_0 \times \left(1 + exp(-x_i'\beta)\right)^{-1}$$

When adjusting a zero inflated beta regression it is necessary to consider the following steps:

1. Descriptive analysis of predictor variables: Removal of variables with low variability and correction outliers;
2. Analysis and multicollinearity removal;
3. Transformation of the predictor variables (transformation or categorization (bins));

4. At the time of regression it is necessary to verify the significance of the parameters and if the adjusted parameter is interpretable.(Market sense);
5. Test performance rates in hold out sample and out-of-time sample.

To evaluate the quality of the fitting we can calculate the correlation of spearman between predicted and target variable and we can build groups of predicted values and compare the mean values between predicted and target variable.

To fit the zero inflated beta regression in SAS® we use the procedure PROC NLMIXED, procedure responsible for the adjustment of nonlinear models and where it is possible to customize the likelihood function. Below we indicate the main parameters, likelihood function for the zero inflated beta regression, after indicating the macro constructed to fit the model, indicate the calculated betas and the spearman correction results and residual graphic analysis.

PROC NLMIXED – ZERO INFLATED BETA REGRESSION MAIN PARAMETERS

```
PROC NLMIXED DATA=&DATASET. METHOD=&METHOD.;
      P0 = EXP(&XB_P0.)/(1+EXP(&XB_P0.));
      MU = EXP(&XB_MU.)/(1+EXP(&XB_MU.));
      IF (&TARGET. = 0) THEN LL = LOG(P0);
      ELSE
      LL = LGAMMA((MU*PHI)+(PHI - MU*PHI)) - LGAMMA((MU*PHI)) - LGAMMA((PHI -
      MU*PHI)) + (((MU*PHI)-1)*LOG(&TARGET.)) + (((PHI - MU*PHI)-1)*LOG(1-
      &TARGET.)) + LOG (1-P0);
      MODEL &TARGET ~ GENERAL(LL);
      PREDICT MU OUT=MU_RESULTS;
      PREDICT P0 OUT=P0_RESULTS;
RUN;
```

Where:

METHOD = Method that will be used for the adjustment of the likelihood function, has as options: FIRO, GAUSS, HARDY and ISAMP. For the project, we use the HARDY method.

Functions Parameters = P0 / Mu. Inflated beta regression functions.

LL = Likelihood function for zero inflated beta regression.

Predict = Output dataset with the adjustments of each part of the equation (zero and beta)

Macro variables: &Dataset = Dataset with predict variables and target variable

&XB_P0. = Vector with variables predicting a response of zero

&XB_mu. = Vector with variables predicting a response of beta

&target. = Target variable

IMPLEMETATION OF ZERO INFLATED BETA REGRESSION

Call macro:

```
%ZERO_BETA_REG(DATASET,   /*DATASET*/
               METHOD,    /*METHOD USE TO FIT MODEL: FIRO, GAUSS, HARDY and
                                                                    ISAMP.*/
               MU_VARS,   /*VECTOR WITH VARIABLES PREDICTING A RESPONSE OF
                                                                    MU*/
               P0_VARS,   /*VECTOR WITH VARIABLES PREDICTING A RESPONSE OF
                                                                    P0*/
               TARGET    /*TARGET VARIABLE*/
            );
```

Macro Code:

```
%MACRO LOAD_VARS(VARS,
                 B0,
                 XB2,
                 B
                 );
      DATA _NULL_;
            %GLOBAL &XB2;
            LENGTH &XB2 $30000.;
            &XB2=&B0;
            %IF &VARS NE '' %THEN %DO;
                  %LET N=1;
                  VAR&N="%SCAN(&VARS,&N,' ')";
                  %DO %WHILE( %SCAN(&VARS,&N,' ') NE );
                        %LET N=%EVAL(&N+1);
                        VAR&N="%SCAN(&VARS,&N,' ')";
                  %END;
                  %LET N_1=%EVAL(&N-1);
                        ARRAY XBV {*} $ VAR1--VAR&N_1;
                  %DO J=1 %TO &N_1;
                        &B&J= "&B&J";
                  %END;
                  %LET ONE=1;
                  ARRAY &B{*} $8 &B&ONE--&B&N_1 ;
                  ARRAY S{1} $ 8 ('+');
                  ARRAY T{1} $ 8 ('*');
                  DO I=1 TO DIM(XBV) WHILE (XBV{I} NE '');
                        &XB2       =  CATS(OF &XB2 S{1} &B{I} T{1} XBV{I});
                  END;
            %END;
            CALL SYMPUT("&XB2",&XB2);
      RUN;
%MEND LOAD_VARS;
```

Macro LOAD_VARS is a part of code : Swearingen, Christopher J; Melguizo Castro, Maria S; Bursac, Zoran. 2012

```
%MACRO ZERO_BETA_REG(DATASET,METHOD,MU_VARS,P0_VARS,TARGET);
    %PUT  ********************************;
    %PUT  * ZERO INFLATED BETA REGRESSION  *;
    %PUT  * SERASA EXPERIAN - BRAZIL       *;
    %PUT  * ANALYTICS - 2017               *;
    %PUT  * PAULO DI CELLIO / MARC WITARSA *;
    %PUT  ********************************;
    %PUT;
      *FIT ZERO INFLATED BETA REGRESSION;
      DATA MODELO;
            SET &DATASET.;
            *CHECK TARGET VARIABLE STRUCTURE AND CHANGE IF NECESSARY;
            IF &TARGET. LT 0 THEN DO;
               %PUT CAUTION : TARGET VARIABLE HAS VALUES < A 0, VALUES
                                                          CHANGED TO 0;
               &TARGET.=0;
            END;
            ELSE IF &TARGET. GE 1 THEN DO;
               %PUT CAUTION : TARGET VARIABLE HAS VALUES >= A 1, VALUES
                                                        CHANGED TO 0.99;
               &TARGET.=0.99;
            END;
      RUN;
      *LOAD VECTORS;
      %LOAD_VARS(&MU_VARS.,'M0',XB_MU,M0);
      %LOAD_VARS(&P0_VARS.,'P00',XB_P0,P0);
      %PUT: FIT VARIABLES;
      %PUT MU = &XB_MU.;
      %PUT P0 = &XB_P0.;
      *FIT;
      ODS OUTPUT FitStatistics=AIC ParameterEstimates=ESTIMADORES
                            ConvergenceStatus=ConvergenceStatus;
      PROC NLMIXED DATA=&DATASET. METHOD=&METHOD.;
            P0 = EXP(&XB_P0.)/(1+EXP(&XB_P0.));
            MU = EXP(&XB_MU.)/(1+EXP(&XB_MU.));
            IF (&TARGET. = 0) THEN LL = LOG(P0);
            ELSE LL = LGAMMA((MU*PHI)+(PHI - MU*PHI)) - LGAMMA((MU*PHI)) -
            LGAMMA((PHI - MU*PHI)) + (((MU*PHI)-1)*LOG(&TARGET.)) + (((
            PHI - MU*PHI)-1)*LOG(1-&TARGET.)) + LOG (1-P0);
            MODEL &TARGET ~ GENERAL(LL);
            PREDICT MU OUT=MU_RESULTS ;
            PREDICT P0 OUT=P0_RESULTS ;
      RUN;
      ODS OUTPUT CLOSE;
      *RESULTS;
      *PREDICT VALUE;
      DATA MU_RESULTS;
            SET MU_RESULTS;
            LINHA = _N_;
            RENAME Pred=MU;
      RUN;
      DATA P0_RESULTS;
            SET P0_RESULTS;
            LINHA = _N_;
            RENAME Pred=P0;
      RUN;
```

```
DATA PREDITO;
      MERGE MU_RESULTS P0_RESULTS ;
      BY LINHA;
      ESP = P0 + (1-P0)*MU;
      RESID=&TARGET.-ESP;
RUN;
*EASY VISUALIZATION;
DATA ESTIMADORES;
      SET ESTIMADORES;
      LENGTH VAR $32.;
      CONTROL = SUBSTR(Parameter,1,2);
RUN;
PROC SORT DATA=ESTIMADORES;
      BY CONTROL;
RUN;
DATA ESTIMADORES;
      RETAIN PARAMETER VAR;
      SET ESTIMADORES;
      BY CONTROL;
      IF FIRST.CONTROL THEN ID=0;
      ELSE ID=ID+1;
      RETAIN ID;
      IF CONTROL EQ "M0" AND ID EQ 0 THEN VAR = "MU - INTERCEPT";
      ELSE IF CONTROL EQ "M0" AND ID GT 0 THEN VAR =
                                          SCAN("&MU_VARS.",ID);
      ELSE IF CONTROL EQ "P0" AND ID EQ 0 THEN VAR = "P0 - INTERCEPT";
      ELSE IF CONTROL EQ "P0" AND ID GT 0 THEN VAR =
                                             SCAN("&P0_VARS.",ID);
RUN;
PROC FORMAT;
      VALUE COR
      LOW - 0.001   = 'GREEN'
            0.001 <- 0.05 = 'GRAY'
            0.05 <- HIGH  = 'RED'
      . = 'BLACK'
;
QUIT;
TITLE "ZERO INFLATED BETA REGRESSION";
PROC PRINT DATA=ESTIMADORES NOOBS;
      VAR Parameter VAR ESTIMATE;
      VAR Probt / style(column)={foreground=COR.};
RUN;
*BASIC EVALUATION;
*SPEARMAN CORRELATION;
PROC CORR DATA=PREDITO OUTS=CORR_PRED NOPRINT;
      VAR ESP;
      WITH &TARGET.;
RUN;
*AVERAGE BY GROUP;
PROC RANK DATA=PREDITO GROUPS=10 OUT=PREDITO;
      VAR &TARGET.;
      RANKS DECIL_RESP;
RUN;
PROC SQL;
      CREATE TABLE ANALISE_RESP AS
            SELECT
                  DECIL_RESP,
```

```
                        AVG(&TARGET.) AS AVG_RESP,
                        AVG(ESP)      AS AVG_ESP
                  FROM PREDITO
                  GROUP BY DECIL_RESP
            ;
      QUIT;
      PROC SGPLOT DATA=ANALISE_RESP NOCYCLEATTRS;
             SERIES X=DECIL_RESP Y=AVG_RESP / LINEATTRS=(COLOR=BLUE);
             SERIES X=DECIL_RESP Y=AVG_ESP / LINEATTRS=(COLOR=RED);
      RUN;
      PROC PRINT DATA=CORR_PRED NOOBS LABEL ;
            WHERE _TYPE_ EQ "CORR";
            VAR ESP / style={tagattr='format:PERCENT'};
            LABEL
                  ESP = "SPEARMAN CORRELATION - PREDICT X TARGET "
                  ;
      QUIT;
      TITLE "RESIDUAL ANALYSIS";
      PROC SGPLOT DATA=PREDITO;
            SCATTER X=ESP Y=RESID;
      RUN;
%MEND ZERO_BETA_REG;
```

GRADIENT BOOSTING – PROC TREEBOOST

Gradient Boosting is a machine learning algorithm that can be used to solve regression or classification problems. It is an additive combination algorithm of several week models (week learning), estimated interactively resulting in a stronger model (stronger learning). The idea behind the Gradient boosting algorithm is to improve each interaction of the previous model, observing the model's deficiencies through gradients. Considering the development of decision tree models and after the adjustment of the model, we can observe that its residual (quadratic loss function) is equal to a negative gradient function. If our goal is to minimize residual, this is the same as approaching the gradient to zero. In addition to the quadratic loss function other functions can also be applied is : absolute loss or Huber loss.

The Gradient Boosting Algorithm Steps are:

1) Fit the first model

2) Fit of the correction model (Second model) using the residuals of the first model as target variable. The role of this second model is to minimize the error of the first model, to compensate for the parameters that have not been adjusted so well in the first model.

3) If after the fit of the second model, still have residuals (errors), the process repeats step two considering the last fit until it minimizes the residue (reduce the gradient to the nearest zero).

The gradient boosting algorithm can be applied to a variety of problems, considering nominal, continuous or binary target variables and has an advantage that does not require that the predictor variables be treated for multicollinearity. But it is not possible to interpret the parameters of the variables due to multiple interactions.

Because it is a machine learning and interactive algorithm, we recommend the validation of the model in hold out and out of time samples to avoid problems of over fitting. But the gradient boosting proves to be quite efficient against over fitting when compared to other interactive tree algorithms (Eg. Random Forest).

In this paper, we explore an implementation of Gradient Boosting using the SAS® PROC TREEBOOST, where a procedure was created to test the gradient boosting algorithm considering different numbers of interactions and variables (selected the most significant k variables after the result considering all the vector of variables). Finally, the results of the fit in each of the samples and the codes for the application of the models are generated.

PROC TREEBOOST – GRADIENT BOOSTING MAIN PARAMETERS

```
PROC TREEBOOST DATA=&DEV_DATA.
      CATEGORICALBINS=10
      INTERVALBINS=400
      EXHAUSTIVE=500
      INTERVALDECIMALS=MAX
      LEAFSIZE=100
      MAXBRANCHES=6
      ITERATIONS=100
      MINCATSIZE=100
      MISSING=USEINSEARCH
      SEED=05042017
      SHRINKAGE=0.1
      SPLITSIZE=100
      TRAINPROPORTION =0.5;
      INPUT &VECTOR_1. / LEVEL=INTERVAL;
      INPUT &VECTOR_2. / LEVEL=BINARY;
      INPUT &VECTOR_3. / LEVEL=NOMINAL;
      TARGET &TARGET.  / LEVEL INTERVAL;
      IMPORTANCE NVARS=50 OUTFIT=BASE_VARS;
      SUBSERIES BEST;
      CODE FILE="&DIR./GBS_&NAME._&P1._&NUM_VARS..SAS" NOPREDICTION;
      SAVE MODEL=MODELGBS.GBS_TEST;
RUN;
```

Where:

CATEGORICALBINS	= Maximum number of bins for nominal variables
INTERVALBINS	= Number of bins for continuous variables
EXHAUSTIVE	= Maximum number of splits
INTERVALDECIMALS	= Precision, in decimals
ITERATIONS	= Number of boosting.
LEAFSIZE	= Minimum number of observations that is necessary to form a new branch.
MAXBRANCHES	= Number of leaves - 2 produces a binary tree
MINCATSIZE	= Minimum number of observations for one category node
MISSING	= How to use the missing's. useinsearch - considers missing in a separate class

SEED	= seed
SHRINKAGE	= learning rate
SPLITSIZE	= minimum number of observations that is necessary to form a new slipt
TRAINPROPORTION	= % training sample
SUBSERIES	= Number of interactions that will be used. Best uses the smallest interactions with the best performance.
CODE FILE	= Export SAS Code to apply algorithm.
SAVE MODEL	= Export inmodel dataset to apply gradient boosting tree.
Input	=Vector of variables will be testing
IMPORTANCE NVARS	= Select the k most important variables.

IMPLEMETATION OF GRADIENT BOOSTING

Call macro:

```
/*PARAMETERS*/
%LET CATEGORICALBINS=10;
%LET INTERVALBINS=400
%LET EXHAUSTIVE=500;
%LET INTERVALDECIMALS=MAX
%LET LEAFSIZE=100;
%LET MAXBRANCHES=6;
%LET MINCATSIZE=100;
%LET MISSING=USEINSEARCH;
%LET SEED=05042017;
%LET SHRINKAGE = 0.1;
%LET SPLITSIZE=100;
%LET TRAINPROPORTION=0.5;
/*Variables*/
%LET VARS_INTERVAL = x1 x2;
%LET VARS_BINARY   = x3 x4;
%LET VARS_Nominal  = x5;


%GRADIENT_BOOSTING(DEV,              /*Training Sample */
                   OOS,              /*Hold out sample*/
                   OOT,              /*Out of time sample*/
                   100 50 20,        /*Interactions to test */
                   50 20 10,         /*Number of best variables*/
                   Target,          /*Target variable */
                   INTERVAL,        /*Type of target variable - BINARY /
                                  INTERVAL NOMINAL*/
                   \\sgf\      /*Patch*/
                   PROD_1      /*Name*/
      );
```

Macro Code:

```
%MACRO
GRADIENT_BOOSTING(DEV_DATA,OOS_DATA,OOT_DATA,NGRADIENT,NVARS,TARGET,LEVELT,
DIR,NAME);
    OPTIONS NOMPRINT NONOTES NOSYMBOLGEN NOMLOGIC;
    %PUT  **********************************;
    %PUT  * GRADIENT BOOSTING              *;
    %PUT  * SERASA EXPERIAN - BRAZIL       *;
    %PUT  * ANALYTICS - 2017               *;
    %PUT  * PAULO DI CELLIO / MARC WITARSA *;
    %PUT  **********************************;
    %PUT;
      *GRADIENT BOOSTING STRESS;
      * TRY TO FIND A BETTER MODEL BASED ON PARAMETERS OF INTERATIONS AND
NUMBER OF VARIABLES;
      %LET NTEST = %SYSFUNC(COUNTW(&NGRADIENT.));
      %LET TVARS = %SYSFUNC(COUNTW(&NVARS.));
      %LET TTEST = %EVAL(((&NTEST.*&TVARS.)+&NTEST.));
      LIBNAME MODELGBS "&DIR.";
      %PUT NUMBER OF TESTS : &TTEST.;
      %DO I=1 %TO &NTEST.;
            %LET P1 = %SYSFUNC(SCAN(&NGRADIENT.,&I.));
            *LOAD VARIABLE VECTORS;
            %LST_VARS(1,0);
            %PUT INTERATIONS : &I. - NUMBER OF VARIABLES : &NUM_VARS.;
            %DO J=1 %TO &TVARS.;
                  %LET P2 = %SYSFUNC(SCAN(&NVARS.,&J.));
                  %PUT VARIABLES : &VECTOR_1. &VECTOR_2. &VECTOR_3.;
                  *GRADIENT WITH ALL VARIABLES;
                  PROC TREEBOOST DATA=&DEV_DATA.
                    CATEGORICALBINS=&CATEGORICALBINS.
                    INTERVALBINS=&INTERVALBINS.
                    EXHAUSTIVE=&EXHAUSTIVE.
                    INTERVALDECIMALS= &INTERVALDECIMALS.
                    LEAFSIZE=&LEAFSIZE.
                    MAXBRANCHES=&MAXBRANCHES.
                    ITERATIONS=&P1.
                    MINCATSIZE=&MINCATSIZE.
                    MISSING=&MISSING.
                    SEED=&SEED.
                    SHRINKAGE=&SHRINKAGE.
                    SPLITSIZE=&SPLITSIZE.
                  TRAINPROPORTION = &TRAINPROPORTION.;
                  INPUT &VECTOR_1. / LEVEL=INTERVAL;
                  INPUT &VECTOR_2.   / LEVEL=BINARY;
                  INPUT &VECTOR_3.  / LEVEL=NOMINAL;
                    TARGET &TARGET. / LEVEL=&LEVELT.;
                  IMPORTANCE NVARS=&P2. OUTFIT=BASE_VARS(KEEP=_INPUT1_);
                  SUBSERIES BEST;
                  CODE FILE="&DIR./GBS_&NAME._&P1._&NUM_VARS..SAS"
                                                    NOPREDICTION;
                    SAVE MODEL=MODELGBS.GBS_&NAME._&P1._&NUM_VARS;
                  RUN;
                  QUIT;
                  %APPLY_GBS(MODELGBS.GBS_&NAME._&P1._&NUM_VARS);
                  %LST_VARS(2,&P2.);
```

```
            %PUT INTERATIONS : &I. - NUMBER OF VARIABLES : &P2.;
            %PUT VARIABLES : &VECTOR_1. &VECTOR_2. &VECTOR_3.;
            PROC TREEBOOST data=&DEV_DATA.
                  CATEGORICALBINS=&CATEGORICALBINS.
                INTERVALBINS=&INTERVALBINS.
                EXHAUSTIVE=&EXHAUSTIVE.
                INTERVALDECIMALS= &INTERVALDECIMALS.
                LEAFSIZE=&LEAFSIZE.
                MAXBRANCHES=&MAXBRANCHES.
                ITERATIONS=&P1.
                MINCATSIZE=&MINCATSIZE.
                MISSING=&MISSING.
                SEED=&SEED.
                SHRINKAGE=&SHRINKAGE.
                SPLITSIZE=&SPLITSIZE.
                TRAINPROPORTION = &TRAINPROPORTION.;
                INPUT &VECTOR_1. / LEVEL=INTERVAL;
                INPUT &VECTOR_2.   / LEVEL=BINARY;
                INPUT &VECTOR_3.  / LEVEL=NOMINAL;
                TARGET &TARGET. / LEVEL=&LEVELT.;
                SUBSERIES BEST;
                CODE file="&DIR./GBS_&NAME._&P1._&P2..SAS"
                                              NOPREDICTION;
                SAVE MODEL=MODELGBS.GBS_&NAME._&P1._&P2;
            RUN;
            QUIT;
            %APPLY_GBS(MODELGBS.GBS_&NAME._&P1._&P2);
      %END;
%END;
*RESULTS;
PROC SORT DATA=MODELGBS.RESULTS NODUPKEY;
      BY MODEL SAMPLE ;
RUN;
PROC TRANSPOSE DATA=MODELGBS.RESULTS(KEEP=P_&TARGET. SAMPLE MODEL)
                                                      OUT=RESULTS ;
      BY   MODEL ;
      VAR P_&TARGET.;
      ID SAMPLE;
RUN;
PROC SORT DATA=MODELGBS.RESULTS OUT=MODELGBS.RESULTS(KEEP=MODEL V_:)
                                                            NODUPKEY;
      BY MODEL;
RUN;
PROC SORT DATA=RESULTS;
      BY MODEL;
RUN;
DATA MODELGBS.RESULTS;
      MERGE MODELGBS.RESULTS (IN=A) RESULTS;
      BY MODEL;
      IF A;
      DIF_OOS = 1-(OOS/DEV);
      DIF_OOT = 1-(OOT/DEV);
      DROP _NAME_ _LABEL_;
RUN;
PROC SORT DATA=MODELGBS.RESULTS;
      BY DESCENDING DIF_OOS;
RUN;
```

```
      PROC PRINT DATA=MODELGBS.RESULTS NOOBS;
            VAR _ALL_;
      RUN;
      *CLEAN;
      PROC DATASETS LIB=WORK NOLIST;
            DELETE _NAMEDAT BASE_VARS BASE_VARS_T BKT CORR KS LST_VARS
                                                            LST_VARS_T ;
      QUIT;
      LIBNAME MODELGBS CLEAR;
      OPTIONS MPRINT NOTES NOSYMBOLGEN NOMLOGIC;
%MEND GRADIENT_BOOSTING;


%MACRO LST_VARS(TYPE,J);
      %GLOBAL VECTOR_1 VECTOR_2 VECTOR_3 NUM_VARS;
      %IF &TYPE. EQ 1 %THEN %DO;
            * VARIABLE VETOR;
            DATA LST_VARS;
                  LENGTH VAR $32 TYPE 8.;
                  %LET COUNT_VARS_INTERVAL  =
            %EVAL(%SYSFUNC(COUNT(%CMPRES(&VARS_INTERVAL.),%STR( ))));
                  %LET COUNT_VARS_BINARY    =
            %EVAL(%SYSFUNC(COUNT(%CMPRES(&VARS_BINARY.),%STR( ))));
                  %LET COUNT_VARS_NOMINAL   =
            %EVAL(%SYSFUNC(COUNT(%CMPRES(&VARS_NOMINAL.),%STR( ))));

                  %IF &COUNT_VARS_INTERVAL. GE 1 %THEN %DO;
                        %DO K=1 %TO %SYSFUNC(COUNTW(&VARS_INTERVAL.));
                              VAR =
                              SCAN("&VARS_INTERVAL.",&K.);TYPE=1;OUTPUT;
                        %END;
                  %END;
                  %IF &COUNT_VARS_BINARY. GE 1 %THEN %DO;
                        %DO K=1 %TO %SYSFUNC(COUNTW(&VARS_BINARY.));
                              VAR = SCAN("&VARS_BINARY.",&K.);TYPE=2;OUTPUT;
                        %END;
                  %END;
                  %IF &COUNT_VARS_NOMINAL. GE 1 %THEN %DO;
                        %DO K=1 %TO %SYSFUNC(COUNTW(&VARS_NOMINAL.));
                              VAR = SCAN("&VARS_NOMINAL.",&K.);TYPE=3;OUTPUT;
                        %END;
                  %END;
            RUN;
            DATA _NULL_;
                  SET LST_VARS END=EOF;
                  LENGTH VECTOR_1 VECTOR_2 VECTOR_3 $32000.;
                  RETAIN VECTOR:;
                  IF _N_ EQ 1 THEN DO;
                        VECTOR_1 = '';
                        VECTOR_2 = '';
                        VECTOR_3 = '';
                  END;
                  IF TYPE EQ 1 THEN VECTOR_1 = COMPBL(VECTOR_1||VAR);
                  ELSE IF TYPE EQ 2 THEN VECTOR_2 = COMPBL(VECTOR_2||VAR);
                  ELSE IF TYPE EQ 3 THEN VECTOR_3 = COMPBL(VECTOR_3||VAR);
                  IF EOF THEN DO;
                        CALL SYMPUT("VECTOR_1",COMPBL(VECTOR_1));
```

```
                        CALL SYMPUT("VECTOR_2",COMPBL(VECTOR_2));
                        CALL SYMPUT("VECTOR_3",COMPBL(VECTOR_3));
                        CALL SYMPUT("NUM_VARS",COMPRESS(_N_));
                  END;
            RUN;
      %END;
      %IF &TYPE. EQ 2 %THEN %DO;
            * SELECT  PARTIAL VARIABLES;
            DATA BASE_VARS_T;
                  SET BASE_VARS;
                  IF _INPUT1_ EQ '' THEN DELETE;
                  IF _N_ LE &J.+1;
                  VAR=_INPUT1_;
                  KEEP
                        VAR ;
            RUN;
            PROC SORT DATA=BASE_VARS_T OUT=BASE_VARS_T;
                  BY VAR;
            RUN;
            PROC SORT DATA=LST_VARS;
                  BY VAR;
            RUN;
            DATA LST_VARS_T;
                  MERGE LST_VARS (IN=A) BASE_VARS_T (IN=B);
                  BY VAR;
                  IF A AND B;
            RUN;
            DATA _NULL_;
                  SET LST_VARS_T END=EOF;
                  LENGTH VECTOR_1 VECTOR_2 VECTOR_3 $32000.;
                  RETAIN VECTOR:;
                  IF _N_ EQ 1 THEN DO;
                        VECTOR_1 = '';
                        VECTOR_2 = '';
                        VECTOR_3 = '';
                  END;
                  IF TYPE EQ 1 THEN VECTOR_1 = COMPBL(VECTOR_1||VAR);
                  ELSE IF TYPE EQ 2 THEN VECTOR_2 = COMPBL(VECTOR_2||VAR);
                  ELSE IF TYPE EQ 3 THEN VECTOR_3 = COMPBL(VECTOR_3||VAR);
                  IF EOF THEN DO;
                        CALL SYMPUT("VECTOR_1",COMPBL(VECTOR_1));
                        CALL SYMPUT("VECTOR_2",COMPBL(VECTOR_2));
                        CALL SYMPUT("VECTOR_3",COMPBL(VECTOR_3));
                  END;
            RUN;
      %END;
%MEND LST_VARS;


%MACRO APPLY_GBS(MODEL);
      *CHECK MODEL;
      %DO P=1 %TO 3;
            %IF &P. EQ 1 %THEN %DO;          %LET DATATEST=&DEV_DATA.; %END;
            %ELSE %IF &P. EQ 2 %THEN %DO;    %LET DATATEST=&OOS_DATA.; %END;
            %ELSE %IF &P. EQ 3 %THEN %DO;    %LET DATATEST=&OOT_DATA.; %END;
            *APPLY MODEL;
            PROC TREEBOOST INMODEL=&MODEL.;
```

```
                SCORE DATA=&DATATEST. OUT=BKT PREDICTION;
            RUN;
            PROC CORR DATA=BKT OUTS=CORR(WHERE=(_TYPE_ EQ 'CORR')) NOPRINT;
                VAR P_&TARGET.;
                WITH &TARGET.;
            RUN;
            DATA CORR;
                SET CORR;
                LENGTH MODEL $50. SAMPLE $3. V_INTERVAL V_BINARY V_NOMINAL
$32000.;
                MODEL = ("&MODEL.");
                V_INTERVAL = "&VECTOR_1.";
                V_BINARY = "&VECTOR_2.";
                V_NOMINAL = "&VECTOR_3.";
                %IF &P. EQ 1 %THEN %DO; SAMPLE='DEV'; %END;
                %ELSE %IF &P. EQ 2 %THEN %DO; SAMPLE='OOS'; %END;
                %ELSE %IF &P. EQ 3 %THEN %DO; SAMPLE='OOT'; %END;
            RUN;
            PROC APPEND DATA=CORR BASE=MODELGBS.RESULTS;
            QUIT;
        %END;
%MEND APPLY_GBS;
```

REVENUE SCORE CASE

For the application of the Revenue Score solution, the following steps were used:

1) Business and data understanding: Evaluation of information sources (invoice Information, available vintages and forecast period (6 or 12 months).
2) Data Preparation:
    a. Creation of set of attributes
    b. Creation of target variable, divided into 3 revenue sources, observing 6 months of forecast. (Target variable item). In this case were built target variables for: revolving, interchange and four distinct card products. On total, six target variables.
3) Model Development

    For the development of this case, we applied the two methodologies for all cases of target variable (six possible), so that in the end we could compare the results of each case.

    a. Method Zero inflated beta regression:

        For the development of this methodology the following steps were performed:

        i. Descriptive analysis of predictor variables: Removal of variables with low variability and outliers detection and treatment.
        ii. Analysis and multicollinearity removal. Removal of predictor variables with multicollinearity. Parameter used VIF[1] greater than 10.
        iii. Transformation of the predictor variables in bins

[1] VIF = Variance Inflation factor

iv. Regression fit by observing the significance of the parameters and their meaning with the business.

v. Stability evaluation of the model throughout the vintages and in relation to the hold out and out of time samples.

b. Method Gradient Boosting

For the development of this methodology the following steps were performed:

i. Descriptive analysis of predictor variables: Removal of variables with low variability and outliers detection and treatment.

ii. Fit of the gradient boosting models, observing tests with the combination of interactions and number of variables, looking for the best result with the least number of interactions and variables.

iii. Stability evaluation of the model throughout the vintages and in relation to the hold out and out of time samples.

4) Models results:

The table below shows the results obtained with each of the models. The metric we use here is the spearman correlation.

Model	Target Variable	Number of predictor variables	Spearman Correlation		
			Dev Sample	Hold out sample	Out of time Sample
			jan/16 to mar/16	jan/16 to mar/16	Apr/16 to Jun/16
Zero Inflated beta regression	Revolving	8	0.55	0.53	0.54
Zero Inflated beta regression	Interchange	6	0.64	0.62	0.62
Zero Inflated beta regression	Product 1	4	0.85	0.82	0.83
Zero Inflated beta regression	Product 2	8	0.42	0.4	0.4
Zero Inflated beta regression	Product 3	7	0.74	0.73	0.72
Zero Inflated beta regression	Product 4	5	0.69	0.71	0.7
Gradient Boosting	Revolving	50 vars / 30 Interactions	0.56	0.54	0.54
Gradient Boosting	Interchange	30 vars / 20 Interactions	0.63	0.62	0.63
Gradient Boosting	Product 1	10 vars / 25 Interactions	0.85	0.84	0.83
Gradient Boosting	Product 2	40 vars / 40 Interactions	0.45	0.44	0.43
Gradient Boosting	Product 3	15 vars / 40 Interactions	0.75	0.72	0.71
Gradient Boosting	Product 4	12 vars / 25 Interactions	0.68	0.7	0.7

Table 1. Models Results

We can observe that the results of fit between the methodologies is quite similar. For the gradient boosting models, the results presented in table 1 are of the selected model, in all cases we did not use the complete vector of tested variables (about 800 variables) and opted for the use of the most important variables, to avoid over fitting.

In this case, the models selected for the use were: Revolving: Zero inflated beta regression, Interchange and all products: Gradient Boosting; the option for the inflated beta regression at zero for the revolving was due to the need to interpret each parameter. For the other models, it was not necessary to detail the behavior of each variable, which allowed the use of the gradient boosting.

5) Converting the results from models to application

After developing the models, it was defined revenue forecasting bins to be compared with the observed revenue. We evaluated the percentiles 25th, 50th and 75th, in addition of average and sum of the observed revenue on each group of forcasted revenue. It is possible to note on next table, the percentage of customers against the total revenue in each estimated group.

Revenue Score	Dev Sample							Out-of-time Sample						
	P25	Median	P75	Mean	Sum	% total Clients	% Total revenue	P25	Median	P75	Mean	Sum	% total Clients	% Total revenue
1 - Low	R$ -	R$ -	R$ 4	R$ 2	R$ 643,938	15%	2%	R$ -	R$ -	R$ 3	R$ 2	R$ 620,473	14%	2%
2 - Low	R$ -	R$ 2	R$ 10	R$ 5	R$ 1,508,864	20%	5%	R$ -	R$ 2	R$ 9	R$ 4	R$ 1,465,335	19%	5%
3 - Neutral	R$ 10	R$ 15	R$ 80	R$ 20	R$ 6,789,326	40%	25%	R$ 11	R$ 16	R$ 82	R$ 21	R$ 7,239,383	44%	27%
4- Neutral	R$ 100	R$ 40	R$ 180	R$ 75	R$ 5,783,938	15%	21%	R$ 96	R$ 38	R$ 160	R$ 70	R$ 5,304,214	15%	20%
5 - High	R$ 195	R$ 120	R$ 400	R$ 180	R$ 12,973,472	10%	47%	R$ 199	R$ 125	R$ 420	R$ 185	R$ 12,455,433	8%	46%

Table 2. Revenue Score Bins

As we can see in table 2, the gross revenue forecast is very efficient, since we managed to concentrate in the last range a large part of all revenues regarding a small group of clients (eg Table 2: with 10% of clients we have about 47% of all revenue). This information is very important to support the strategy or set a go market action, because it is possible to reach only 10% of customers that generate almost 50% of all revenue. In this table we present the results obtained in the client view (sum of all the revenues), but it is possible to have the same evaluation for each type of revenue individually.

Finally, it is possible to work with a cross credit risk x revenue score matrix, to observe areas of interest such as: 1) Clients with high risk and high revenue: Clients that have good gross revenue, but probably this revenue will not be converted into net revenue 2) Customers with high revenue and low risk: Cost-effective and low-risk clients, bonus or incentive opportunities.

Revenue Matrix / Credit Score (Averages of Revenue)		Revenue Score									
		Low				Neutral				High	
		1		2		3		4		5	
Credit Risk	High	1	R$ 1	R$ 4		R$ 18		R$ 60		R$ 160	
		2	R$ 1	R$ 5		R$ 20		R$ 65		R$ 165	
	Neutral	3	R$ 2	R$ 5		R$ 19		R$ 70		R$ 170	
		4	R$ 3	R$ 6		R$ 21		R$ 72		R$ 180	
	Low	5	R$ 3	R$ 7		R$ 22		R$ 70		R$ 180	

Table 3. Revenue Score Matrix

METHOD COMPARISON

The two methods for gross revenue forecasting were very efficient and achieved very similar results which indicates that we can use either method to fit. Three main characteristics decide which method to use:

1) Model parameters interpretation required or desired: In some situations, the monitoring of each variable and interpretation of the parameter is indispensable which can't be done with the gradient boosting. In these cases, it is necessary to use the methodology of zero inflated beta regression.

2) Practical for developmenting: The gradient boosting method does not require many previous treatments in the variables, and because it is an interactive and automatic process, it ends up being very fast to reach models. Gradient boosting is the most suitable method if there are several models to be developed in a short time.

3) Model Application: The gradient boosting method has a more complex implementation because it considers numerous variables and several interactions. Which makes implementation work more complicated if it is not applied in an appropriate environment.

In summary with an environment prepared for large implementations that uses SAS coding for the application of the models, the Gradient boosting method is very efficient due to its practicality and speed for model fitting. In environments where a conversion from SAS code to C code is required for example, you need to automate the implementation process or the cost of implementing gradient boosting will be very high.

Below is a table with the main differences between the two methodologies:

Subject	Zero inflated Beta Regression	Gradient Boosting
Outliers treatments	Needs treatment	It depends on the loss function. Square loss = Need Huber loss = more robust to out-liers
Multicollinearity - Analysis and removal	Needs treatment	It is not necessary
Variable Transformation	Needs transformation of variables (linear or in bins)	The Process performs transformations automatically (Decision trees)
Fit / Parameters Interpretation	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	Easy deployment	Deployment can be difficult because of the large number of interactions and variables.
Effort	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 and variables)

Table 4. Compare methods

CONCLUSION

Gross revenue models will increasingly be part of the day-to-day analysis of credit, allowing a view not only on the credit risks but also on the revenue that the client is able to generate for the company. The solution proposed in this paper presents a division of gross revenue into 3 main pillars (revolving, interchange and products) and their estimation considering two different methodologies (zero inflated beta regression and gradient boosting), was also explored how to convert the results of the models through revenue score ranges and their combination with credit risk generating interest groups (eg low risk and high profitability = opportunities for bonus or incentives for the clients).

Both methodologies were efficient and with similar results, which gives us the option to choose which methodology is best for each situation:

- Zero inflated beta regression, is traditional methodology that needs some treatments in predictive variables, and requiring a longer time for development, but it is possible to evaluate each parameter and its implementation is relatively simple (low number of variables).
- Gradient boosting, a machine learning algorithm where it is not necessary to treat the predictor variables and the development of the model is very fast (interactive and automatic process), but it is not possible to interpret the parameters and their implementation can be complex if there is not an appropriate environment (many variables and interactions).

The solution proposed here is not limited to the world of credit cards and can be applied for the evaluation of gross revenue of other product lines.

REFERENCES

Li, first Cheng. "A Gentle Introduction to Gradient Boosting." College of Computer and Information Science Northeastern University. Available at: http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf . 20/02/2017

SAS Institute Inc. 2015. SAS/STAT® 14.1 User's Guide. Cary, NC: SAS Institute Inc.

SAS Institute Inc 2012. SAS Enterprise Miner and SAS Text Miner Procedures Reference for SAS 9.3®. Cary, NC: SAS Institute Inc.

Swearingen, Christopher J; Melguizo Castro, Maria S; Bursac, Zoran. 2012. "Inflated Beta Regression: Zero, One, and Everything in Between." Orlando, FL, <SAS Global Forum - 2012>. Available at: http://support.sas.com/resources/papers/proceedings12/325-2012.pdf 15/02/2017

RECOMMENDED READING

- *Base SAS® Procedures Guide*
- SAS® 9.4 Macro Language: Reference, Fifth Edition
- SAS/STAT® 14.1 User's Guide The NLMIXED Procedure
- *SAS Enterprise Miner® and SAS Text Miner® Procedures Reference for SAS 9.3*

CONTACT INFORMATION

Your comments and questions would be valued and encouraged.

Contact the author at:

Paulo Celio Di Cellio Dias
Serasa Experian
paulo.dias@br.experian.com

Marc Witarsa
Serasa Experian
marc.witarsa@br.experian.com