Global ®
Software
Solutions
Group

Extra Gradient Boosting - X BoosT with R

Case study – Banking Sector

March 1 2021

Table of Contents

INTRODUCTION

This is a bank loan case with a binary (yes / no) answer option to accept or reject the loan in a tele marketing campaign.

The detail description of the campaign and its data source is as follows:

This dataset is public available for research. The details are described in [Moro et al., 2011].
Please include this citation if you plan to use this database:

[Moro et al., 2011] S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology.
In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimarães, Portugal, October, 2011. EUROSIS.

Available at: [pdf] http://hdl.handle.net/1822/14838
[bib] http://www3.dsi.uminho.pt/pcortez/bib/2011-esm-1.txt

1. Title: Bank Marketing

2. Sources
Created by: Paulo Cortez (Univ. Minho) and Sérgio Moro (ISCTE-IUL) @ 2012

3. Past Usage:

The full dataset was described and analyzed in:

S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology.
In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimarães,
Portugal, October, 2011. EUROSIS.

4. Relevant Information:

The data is related with direct marketing campaigns of a Portuguese banking institution.
The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required,
in order to access if the product (bank term deposit) would be (or not) subscribed.

There are two datasets:
1) bank-full.csv with all examples, ordered by date (from May 2008 to November 2010).
2) bank.csv with 10% of the examples (4521), randomly selected from bank-full.csv.

The smallest dataset is provided to test more computationally demanding machine learning algorithms (e.g. SVM).

The classification goal is to predict if the client will subscribe a term deposit (variable y).

5. Number of Instances: 45211 for bank-full.csv (4521 for bank.csv)

6. Number of Attributes: 16 + output attribute.

7. Attribute information:

For more information, read [Moro et al., 2011].

Input variables:
bank client data:
1 - age (numeric)
2 - job : type of job (categorical: "admin.","unknown","unemployed","management","housemaid","entrepreneur","student", "blue-collar","self-employed","retired","technician","services")
3 - marital : marital status (categorical: "married","divorced","single"; note: "divorced" means divorced or widowed)
4 - education (categorical: "unknown","secondary","primary","tertiary")
5 - default: has credit in default? (binary: "yes","no")
6 - balance: average yearly balance, in euros (numeric)
7 - housing: has housing loan? (binary: "yes","no")
8 - loan: has personal loan? (binary: "yes","no")
related with the last contact of the current campaign:
9 - contact: contact communication type (categorical: "unknown","telephone","cellular")
10 - day: last contact day of the month (numeric)
11 - month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")
12 - duration: last contact duration, in seconds (numeric)
other attributes:
13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
15 - previous: number of contacts performed before this campaign and for this client (numeric)
16 - poutcome: outcome of the previous marketing campaign (categorical: "unknown","other","failure","success")

Output variable (desired target):
17 - y - has the client subscribed a term deposit? (binary: "yes","no")

8. Missing Attribute Values: None

#1 Upload the Data

/Users/ricardomendez/Documents/GSSG/R_&_MySQ_Demos/XGBoost/XGBoost for Business in Python and R/bank-full.csv

Data <- read.csv("bank-full.csv", sep = ";")

#1.1Check Data structure

str(Data)

```
'data.frame':	45211 obs. of  17 variables:
$ age      : int  58 44 33 47 33 35 28 42 58 43 ...
$ job      : chr  "management" "technician" "entrepreneur" "blue-collar" ...
$ marital  : chr  "married" "single" "married" "married" ...
$ education: chr  "tertiary" "secondary" "secondary" "unknown" ...
$ default  : chr  "no" "no" "no" "no" ...
$ balance  : int  2143 29 2 1506 1 231 447 2 121 593 ... $ housing  : chr  "yes" "yes" "yes" "yes" ...
$ loan     : chr  "no" "no" "yes" "no" ...
$ contact  : chr  "unknown" "unknown" "unknown" "unknown" ...
$ day      : int  5 5 5 5 5 5 5 5 5 5 ...
$ month    : chr  "may" "may" "may" "may" ...
$ duration : int  261 151 76 92 198 139 217 380 50 55 ...
$ campaign : int  1 1 1 1 1 1 1 1 1 1 ... $ pdays    : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
$ previous : int  0 0 0 0 0 0 0 0 0 0 ...
$ poutcome : chr  "unknown" "unknown" "unknown" "unknown" ... $ y        : chr  "no" "no" "no" "no"
...
```

#1.2 Key question

The positive answers rate are 12% in the real scenario; ¿ Which variable combination will reduce the negative answers (reduce the negative predictive value)? Check #17.6 and which is the importance and direction (positive or negative) influence on the decision ? Check #18 and following

To check de initial data values:

head(Data)

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
1	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
2	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
3	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
4	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
5	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no
6	35	management	married	tertiary	no	231	yes	no	unknown	5	may	139	1	-1	0	unknown	no
7	28	management	single	tertiary	no	447	yes	yes	unknown	5	may	217	1	-1	0	unknown	no
8	42	entrepreneur	divorced	tertiary	yes	2	yes	no	unknown	5	may	380	1	-1	0	unknown	no
9	58	retired	married	primary	no	121	yes	no	unknown	5	may	50	1	-1	0	unknown	no
10	43	technician	single	secondary	no	593	yes	no	unknown	5	may	55	1	-1	0	unknown	no
11	41	admin.	divorced	secondary	no	270	yes	no	unknown	5	may	222	1	-1	0	unknown	no
12	29	admin.	single	secondary	no	390	yes	no	unknown	5	may	137	1	-1	0	unknown	no
13	53	technician	married	secondary	no	6	yes	no	unknown	5	may	517	1	-1	0	unknown	no
14	58	technician	married	unknown	no	71	yes	no	unknown	5	may	71	1	-1	0	unknown	no

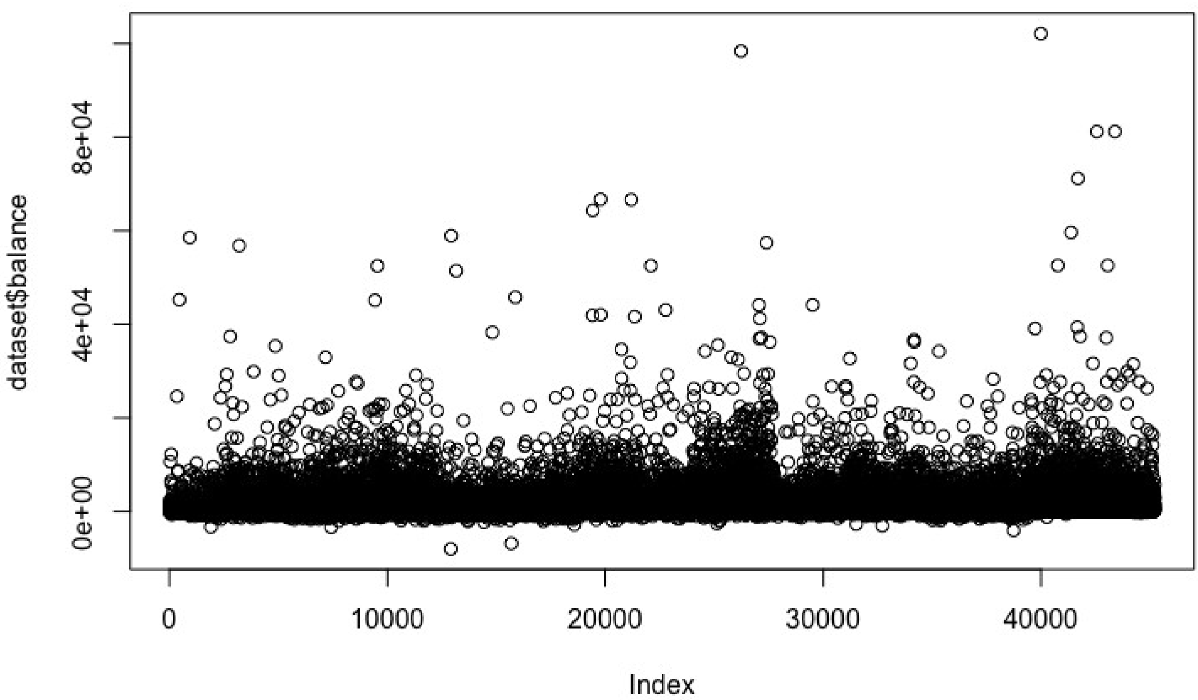
#1.2 Select only numeric values from the Data Structure (Work with dplyr)

library(dplyr)
dataset <- Data %>% select_if(is.numeric)

#1.3 Check summary statistics

summary(dataset)

plot(dataset$balance) #There are some peak values, but XGBoost will mange them corectly; same happens with the Duration variable



# find correlation
cor(dataset)
# there is not correlation between variables, but XGBoost will not be affected.

```
                 age       balance          day      duration
age       1.000000000  0.097782739 -0.009120046 -0.004648428
balance   0.097782739  1.000000000  0.004502585  0.021560380
day      -0.009120046  0.004502585  1.000000000 -0.030206341
duration -0.004648428  0.021560380 -0.030206341  1.000000000
campaign  0.004760312 -0.014578279  0.162490216 -0.084569503
pdays    -0.023758014  0.003435322 -0.093044074 -0.001564770
previous  0.001288319  0.016673637 -0.051710497  0.001203057

             campaign        pdays     previous
age       0.004760312 -0.023758014  0.001288319
balance  -0.014578279  0.003435322  0.016673637
day       0.162490216 -0.093044074 -0.051710497
duration -0.084569503 -0.001564770  0.001203057
campaign  1.000000000 -0.088627668 -0.032855290
```

```
pdays       -0.088627668  1.000000000  0.454819635
previous    -0.032855290  0.454819635  1.000000000
```

#1.4 Include the dependent variable in the data set and rename it

```
dataset <- cbind(Data$y,dataset)
colnames(dataset)[1] <- "yes"
```

	yes	age	balance	day	duration	campaign	pdays	previous
1	no	58	2143	5	261	1	-1	0
2	no	44	29	5	151	1	-1	0
3	no	33	2	5	76	1	-1	0
4	no	47	1506	5	92	1	-1	0
5	no	33	1	5	198	1	-1	0
6	no	35	231	5	139	1	-1	0
7	no	28	447	5	217	1	-1	0
8	no	42	2	5	380	1	-1	0
9	no	58	121	5	50	1	-1	0
10	no	43	593	5	55	1	-1	0
11	no	41	270	5	222	1	-1	0
12	no	29	390	5	137	1	-1	0

2 Split Data set into Trining and Test, use caTools Package

```
install.packages("caTools")
library(caTools)
set.seed(1502)
split <- sample.split(dataset$yes,SplitRatio = 0.8)
```

#Meaning: 80% of data will be TRUE, and 20% will be set to False. The TRUE values will be used to build the training set, and the FALSE values to build the TEST set.

```
training_set <- subset(dataset, split == TRUE)
test_set <- subset(dataset, split == FALSE)
```

#2.1 Isolate the Y variable an convert it to numeric values

```
train.y <- as.numeric(as.factor(training_set$yes)) - 1
test.y <- as.numeric(as.factor(test_set$yes)) - 1
```

R cannot transfor directly form char to numeric, this step is required. The (-1) is a trick to convert Yes and No into "ceros" and "ones"

#2.2 Isolate the X Variables, they have to be transformed to Matrix in R.They are integers which is ok for XGBoost

```
train.x <- as.matrix(training_set[,2:ncol(training_set)])
test.x <- as.matrix(test_set[,2:ncol(test_set)])
```

#3. Set the parameters - Check meaning of each one.

```
Parameters <- list (eta = 0.3,
                    max_depth = 6,
                    subsample = 1,
                    colsample_bytree = 1,
                    minchild_weight = 1,
                    gamma = 0,
                    set.seed = 1502,
                    eval_metric = "auc",
                    objective = "binary:logistic",
                    booster = "gbtree")
```

#4. Set Up parallel running

to increase machine efficiency. Detect cores first (Optional if would like to work in different tasks, so your machine would not be slow!. It wolud be needed for the high parameter tunning step. Check step No.11)

```
install.packages("doParallel")
library(doParallel)
detectCores()
```

#5 Run XGBoost

```
install.packages("xgboost")
library(xgboost)
model1 <- xgboost(data = train.x,
                 label = train.y,
                 sed.seed = 1502,
                 nthread = 3,
                 nround = 100,
                 params = Parameters,
                 print_every_n = 50,
                 early_stopping_rounds = 10)
```

#Looks like the model is overfitted. Check the AUC (Area under the curve) theory; it´s a performance mesure for classification problems.

```
[09:37:49] WARNING: amalgamation/../src/learner.cc:541: Parameters: { minchild_weight, sed_seed, set_seed } might not be used.  This may not be accurate due to some parameters are only used in language bindings but  passed down to XGBoost core.  Or some parameters are not used but slip through this  verification. Please open an issue if you find above cases.

[1]     train-auc: 0.863569

Will train until train_auc hasn't improved in 10 rounds.

[51]    train-auc: 0.926421
[100]   train-auc: 0.944965
```

#6 Predict with xgboost

```
Predictions1 <- predict(model1,newdata = test.x    )
```

#Results are not 0 or 1, so to approximate:

```
Predictions1 <- ifelse(Predictions1 > 0.5,1,0)
```

0.5 is a correct aproximation value, because of the AUC curve

#7 Evaluate the model with the confussion matrix

```
install.packages("caret")
install.packages("e1071")  #only if it shows an error according to your R version
library(caret)
library(e1071)
```

confusionMatrix(table(Predictions1, test.y))

#ConfusionMatrix runs as table; results show a low value for Specificity

```
Confusion Matrix and Statistics

            test.y
Predictions1   0    1
         0    7741  686
         1    243   372

          Accuracy :          0.8973
          95% CI :            (0.8908, 0.9034)
          No Information Rate : 0.883
          P-Value [Acc > NIR] : 9.432e-06

            Kappa : 0.3924

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

        Sensitivity : 0.9696
        Specificity : 0.3516
        Pos Pred Value : 0.9186
        Neg Pred Value : 0.6049
        Prevalence : 0.8830
        Detection Rate : 0.8561
        Detection Prevalence : 0.9320
        Balanced Accuracy : 0.6606

        'Positive' Class : 0
```

#8. Transform the original Character variables (Job, marital, education, etc..) into Dummy variables

The data set will be ready to work. It is easy with this package
install.packages("fastDummies")
library(fastDummies)

dataset_dummy <- dummy_cols(Data, remove_first_dummy = TRUE)
dataset_dummy <- dataset_dummy[,(18:ncol(dataset_dummy))]

#9. Join all columns in the dataset to prepare the final dataset

dataset <- cbind(dataset,dataset_dummy)
dataset <- dataset %>% select (-y_yes) #another way to remove the y column; there are two y columns, and can work only with one.

#1###

10. Run the xgboost again with the final dataset (same previous process

##

10.1 Split Data set into Trining and Test, use caTools Package

install.packages("caTools")
library(caTools)
set.seed(1502)
split <- sample.split(dataset$yes,SplitRatio = 0.8) #Meaning: 80% of data will be TRUE, and 20% will be set to False. The TRUE values will be used to build the training set, and the FALSE values to build the TEST set.

training_set <- subset(dataset, split == TRUE)
test_set <- subset(dataset, split == FALSE)

#10.2 Isolate the Y variable an convert it to to numeric values

train.y <- as.numeric(as.factor(training_set$yes)) - 1
test.y <- as.numeric(as.factor(test_set$yes)) - 1 # R cannot transfor directly form char to numeric, this step is required. The (-1) is a trick to convert Yes and No into "ceros" and "ones"

#10.3.2Isolate the X Variables, they have to be transformed to Matrix in R.They are integers which is ok for XGBoost

train.x <- as.matrix(training_set[,2:ncol(training_set)])
test.x <- as.matrix(test_set[,2:ncol(test_set)])

#10.4 Run XGBoost

install.packages("xgboost")
library(xgboost)
model2 <- xgboost(data = train.x,
                label = train.y,

```
sed.seed = 1502,
nthread = 3,
nround = 100,
params = Parameters,
print_every_n = 50,
early_stopping_rounds = 10)
```

```
[09:56:27] WARNING: amalgamation/../src/learner.cc:541: Parameters: { minchild_weight, sed_seed,
set_seed } might not be used.  This may not be accurate due to some parameters are only used in
language bindings but  passed down to XGBoost core.  Or some parameters are not used but slip
through this  verification. Please open an issue if you find above cases.

[1]	train-auc:0.867103

Will train until train_auc hasn't improved in 10 rounds.

[51]	train-auc:0.965581
[100]	train-auc:0.979705
```

#10.5 Predict again with model 2

```
Predictions2 <- predict(model2,newdata = test.x    )#Results are not 0 or 1,
Predictions2 <- ifelse(Predictions2 > 0.5,1,0)# 0.5 is a correct aproximation value, because of the AUC
curve

confusionMatrix(table(Predictions2, test.y)) #ConfusionMatrix runs as table; results show a better
value for Specificity
```

```
Confusion Matrix and Statistics

            test.y
Predictions2    0    1
           0 7715  568
           1  269  490

               Accuracy : 0.9074
                 95% CI : (0.9013, 0.9133)
    No Information Rate : 0.883
    P-Value [Acc > NIR] : 4.679e-14

                  Kappa : 0.4894

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

```
                    Sensitivity : 0.9663
                    Specificity : 0.4631
                    Pos Pred Value : 0.9314
                      Neg Pred Value : 0.6456
                      Prevalence : 0.8830
                      Detection Rate : 0.8532
                      Detection Prevalence : 0.9161
                      Balanced Accuracy : 0.7147

                      'Positive' Class : 0
```

##

#11. START THE HYPER-PARAMETER TUNNING - Many models will be run by the function itself; It will set up and find the best parameters

```
#Do parallel processing (Check step No.4)

### If This code presents an error in R
#        cpu <- makeCluster(4)
#        registerDoParallel(cpu)
# create the cluster for caret to use
#cl <- makePSOCKcluster(no_cores)

#        May use this code instead:

N_cores <- detectCores() -1
library(doParallel)

cl <- parallel::makeCluster(N_cores, setup_strategy = "sequential")
registerDoParallel(cl)
```

#12. State in parameters

```
Y <- as.factor(as.numeric(as.factor(dataset$yes)) - 1)
X <- as.matrix(dataset[,2:ncol(dataset)])
```

#13. State the crossvalidation parameters

```
tune_control <- trainControl( method = "cv",
```

```
                    allowParallel = TRUE,
                    number = 5)
```

#14 Set the parameters

```
tune_grid <- expand.grid(nrounds = seq(from = 50, to = 600, by = 50),
                    eta = c(0.1,0.2,0.3,0.4),
                    max_depth = seq(2,10, by = 2),
                    subsample = c(0.5, 0.7, 1),
                    colsample_bytree = 1,
                    min_child_weight = 1,
                    gamma = 0)
```

#15 Cross validation and parameter tuning start (It will take some time!! check It !!)

```
start <- Sys.time()
xgb_tune <- train(x = X,
          y = Y,
          method = "xgbTree",
          trControl = tune_control,
          tuneGrid = tune_grid)
end <- Sys.time()
```

#16 Check for the best parameters

```
xgb_tune$bestTune
```

	nrounds	max_depth	eta	gamma	colsample_bytree	min_child_weight	subsample
87	150	6	0.1	0	1	1	0.7

```
View(xgb_tune$results)
```

	eta	max_depth	gamma	colsample_bytree	min_child_weight	subsample	nrounds	Accuracy	Kappa	AccuracySD	KappaSD
87	0.1	6	0	1	1	0.7	150	0.9089602	0.5047514	0.004154456	0.01935584
52	0.1	4	0	1	1	0.7	200	0.9089381	0.4970248	0.002865683	0.01586259
67	0.1	4	0	1	1	1.0	350	0.9088939	0.5010538	0.003337417	0.01869653
68	0.1	4	0	1	1	1.0	400	0.9088496	0.5020510	0.003085834	0.01578564
253	0.2	6	0	1	1	0.5	50	0.9088054	0.5003283	0.002269971	0.01315075
69	0.1	4	0	1	1	1.0	450	0.9085621	0.5019961	0.003219642	0.01648075
59	0.1	4	0	1	1	0.7	550	0.9085179	0.5035858	0.003046242	0.01653167
71	0.1	4	0	1	1	1.0	550	0.9084515	0.5043590	0.003470531	0.01826523
58	0.1	4	0	1	1	0.7	500	0.9084294	0.5035638	0.003324589	0.01797741
66	0.1	4	0	1	1	1.0	300	0.9083851	0.4962801	0.002782914	0.01557831
40	0.1	4	0	1	1	0.5	200	0.9083851	0.4954147	0.003581285	0.01835584
39	0.1	4	0	1	1	0.5	150	0.9083187	0.4914125	0.003570236	0.02072496
51	0.1	4	0	1	1	0.7	150	0.9082745	0.4904896	0.003376732	0.01929546

Showing 1 to 14 of 720 entries, 11 total columns

##

17. HYPER-PARAMETER TUNNING (2 round)

It´s possible to do as many rounds as needed
##

```
#Do parallel processing
N_cores <- detectCores() -1 (# In this example the machine is working with 3 cores out of 4)
cl <- parallel::makeCluster(N_cores, setup_strategy = "sequential")
registerDoParallel(cl)

#17.1 Set the parameters

tune_grid2 <- expand.grid(nrounds = seq(from = 50, to = 600, by = 50),
                          eta = xgb_tune$bestTune$eta,
                          max_depth = xgb_tune$bestTune$max_depth,
                          subsample = xgb_tune$bestTune$subsample,
                          colsample_bytree = c(0.5,0.7,1),
                          min_child_weight = seq(1,6,by = 2),
                          gamma = c(0,0.05,0.1,0.15))
```

#17.2 Cross validation and parameter tuning start (It will take some time!! check It !!)

```
start <- Sys.time()
xgb_tune2 <- train(x = X,
                   y = Y,
                   method = "xgbTree",
                   trControl = tune_control,
                   tuneGrid = tune_grid2)
```

end <- Sys.time()

Check for the best parameters

xgb_tune2$bestTune

```
      nrounds  max_depth   eta    gamma   colsample_by tree   min_child_weight
147    150        6        0.1     0.05          0.7                 1

      subsample
147     0.7
```

View(xgb_tune2$results)

Filter

	eta	max_depth	gamma	colsample_bytree	min_child_weight	subsample	nrounds	Accuracy	Kappa	AccuracySD	KappaSD
147	0.1	6	0.05	0.7	1	0.7	150	0.9091814	0.5012436	0.002457552	0.02138672
422	0.1	6	0.15	1.0	5	0.7	100	0.9091151	0.5007338	0.002212806	0.01870047
281	0.1	6	0.10	0.7	5	0.7	250	0.9088497	0.5043235	0.001895746	0.01587323
172	0.1	6	0.05	0.7	5	0.7	200	0.9088054	0.5032962	0.002583143	0.01985839
280	0.1	6	0.10	0.7	5	0.7	200	0.9087612	0.4991637	0.002284302	0.01916532
254	0.1	6	0.10	0.7	1	0.7	100	0.9087391	0.4925833	0.002030012	0.01494826
196	0.1	6	0.05	1.0	3	0.7	200	0.9086727	0.5043507	0.002770113	0.02190055
423	0.1	6	0.15	1.0	5	0.7	150	0.9086727	0.5018099	0.001578485	0.01431232
146	0.1	6	0.05	0.7	1	0.7	100	0.9086727	0.4925173	0.002168873	0.01609140
232	0.1	6	0.10	0.5	3	0.7	200	0.9086506	0.4995297	0.002976076	0.02214860
255	0.1	6	0.10	0.7	1	0.7	150	0.9086506	0.4985777	0.002188457	0.01679913
52	0.1	6	0.00	0.7	3	0.7	200	0.9085842	0.5018419	0.002661378	0.01947031
183	0.1	6	0.05	1.0	1	0.7	150	0.9085842	0.5053032	0.001885723	0.01628142

Showing 1 to 14 of 432 entries, 11 total columns

#17.3 Thirth round (Run XGBoost for the last time; might do it several times up to your best accuarcy)

#17.4 Set parameters 3

```
Parameters3 <- list (eta = xgb_tune2$bestTune$eta,
                     max_depth = xgb_tune2$bestTune$max_depth,
                     subsample = xgb_tune2$bestTune$subsample,
                     colsample_bytree = xgb_tune2$bestTune$colsample_bytree,
                     minchild_weight = xgb_tune2$bestTune$min_child_weight,
                     gamma = xgb_tune2$bestTune$gamma,
                     set.seed = 1502,
                     eval_metric = "auc",
```

```
                objective = "binary:logistic",
                booster = "gbtree")
```

#17.5 Run XGBoost for the model 3

```
model3 <- xgboost(data = train.x,
               label = train.y,
               sed.seed = 1502,
               nthread = 4,
               nround = xgb_tune2$bestTune$nrounds,
               params = Parameters3,
               print_every_n = 50,
               early_stopping_rounds = 10)
```

```
15:47:05] WARNING: amalgamation/../src/learner.cc:541:
Parameters: { minchild_weight, sed_seed, set_seed } might not be used.

  This may not be accurate due to some parameters are only used in language bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through this
  verification. Please open an issue if you find above cases.


[1]    train-auc:0.702125
Will train until train_auc hasn't improved in 10 rounds.

[51]   train-auc:0.907561
[101]  train-auc:0.918065
[150]  train-auc:0.926064
```

#17.6 Predictions part 3

```
Predictions3 <- predict(model3,newdata = test.x    )
```

#Results are not 0 or 1, the best approximation to improve confusion matrix woild be 1 if > to 0.05 or 0 if different. This comes from cheking several tuning performance for sensitivity vs specificity vs negative predictive value for the present binary situation, and , 0.05 may be a correct aproximation value, because of the AUC curve

Predictions3 <- ifelse(Predictions3 > 0.05,1,0)

#17.7 Cheking Accuracy

confusionMatrix(table(Predictions3, test.y)) # ConfusionMatrix runs as table; results show a better value for Specificity and for negative predictive value (In the real scenario it is 88% ,it was reduced to 25%)

```
Confusion Matrix and Statistics

                 test.y
        Predictions3   0    1
              0       5167   81
              1       2817   977

            Accuracy : 0.6795
           95% CI : (0.6698, 0.6891)
           No Information Rate : 0.883
           P-Value [Acc > NIR] : 1

            Kappa : 0.2689

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

         Sensitivity : 0.6472
         Specificity : 0.9234
         Pos Pred Value : 0.9846
         Neg Pred Value : 0.2575
         Prevalence : 0.8830
      Detection Rate : 0.5714
      Detection Prevalence : 0.5804
      Balanced Accuracy : 0.7853

      'Positive' Class : 0
```

##

#18 Important drivers ### Most important business value conclusion

##

devtools::install_github("liuyanguu/SHAPforxgboost") library("SHAPforxgboost")

#18.1 work with the Shap values (Shapley Additive Explanations) – Check theory ***

#18.2 The ranked features by mean |SHAP|

shap_values$mean_shap_score

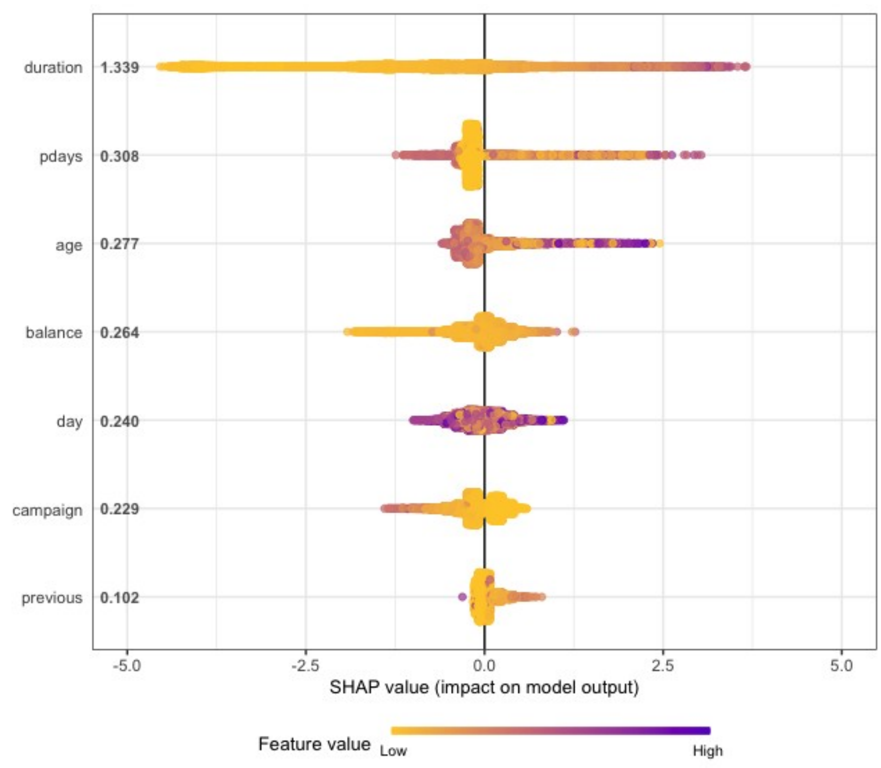
duration	pdays	age	balance	day	campaign	previous
1.3389118	0.3079860	0.2769341	0.2637306	0.2395724	0.2285023	0.1020042

To prepare the long-format data:

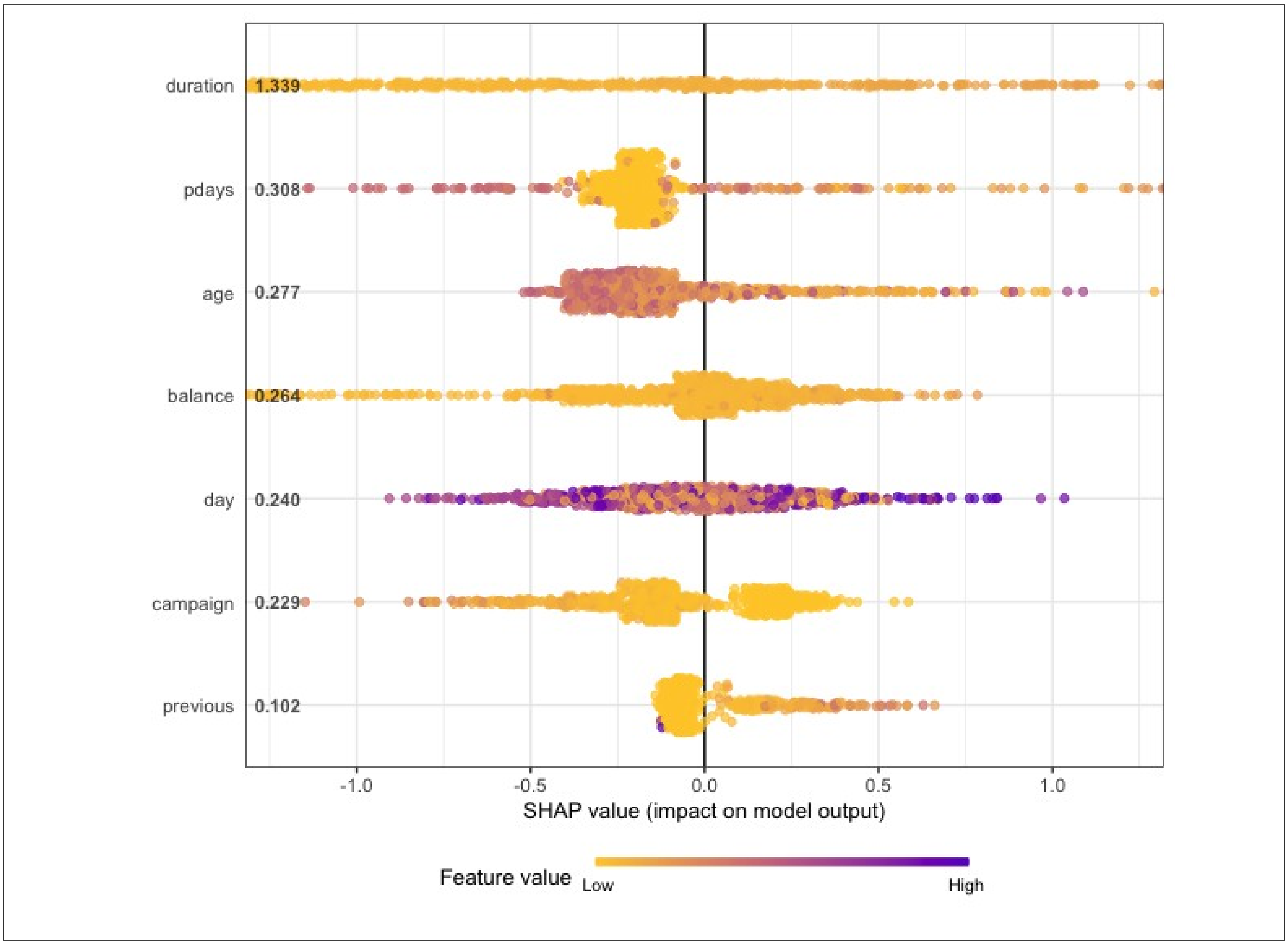
shap_long <- shap.prep(xgb_model = model3, X_train = test.x)

#18.3 **SHAP summary plot**

shap.plot.summary(shap_long)



shap.plot.summary(shap_long, x_bound = 1.2, dilute = 10)



Alternatives options to make the same plot:

option 1: from the xgboost model

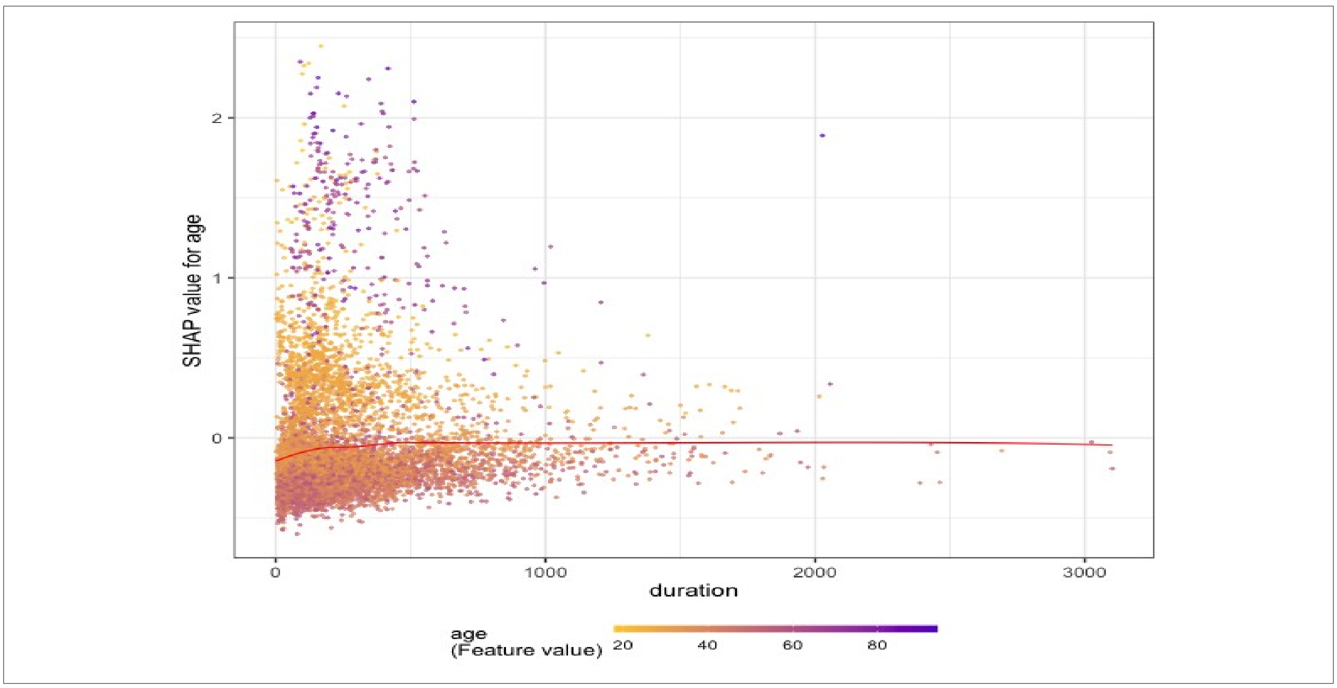
shap.plot.summary.wrap1(model3, X = as.matrix(test.x))

option 2: supply a self-made SHAP values dataset (e.g. sometimes as output from cross-validation)

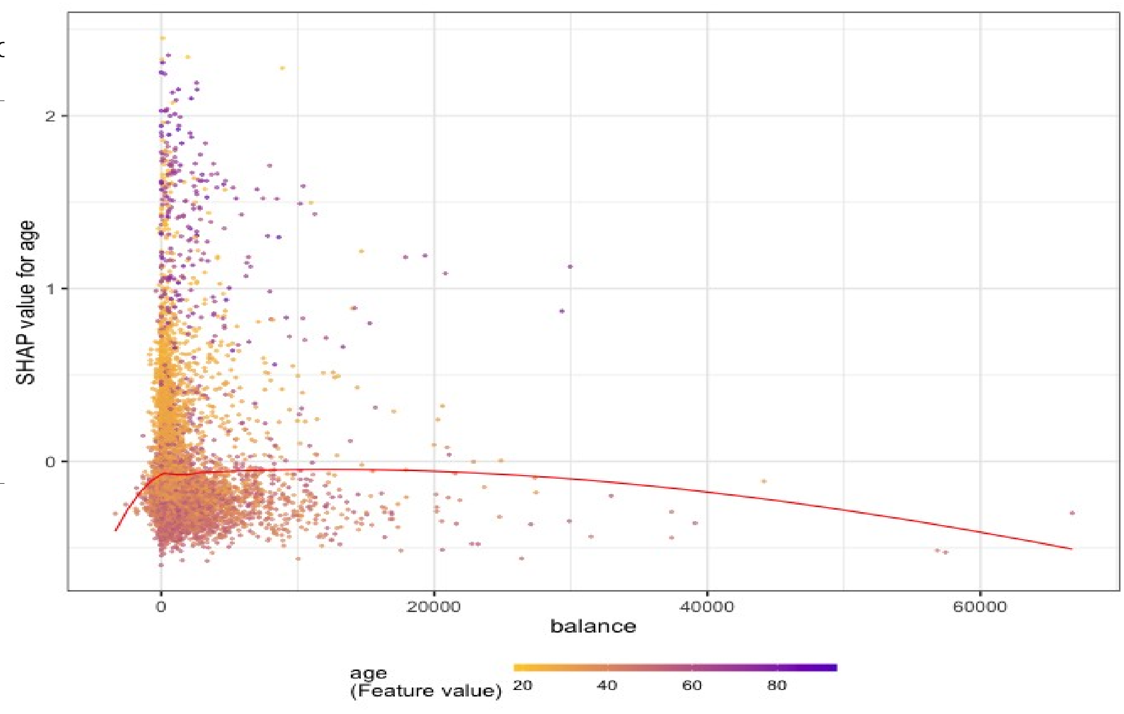
shap.plot.summary.wrap2(shap_values$shap_score, as.matrix(test.x))

#18.4 **SHAP dependence plot**

# prepare the data using either:
# (this step is slow since it calculates all the combinations of features.)
data_int <- shap.prep.interaction(xgb_mod = model3, X_train = as.matrix(test.x))
shap.plot.dependence(data_long = shap_long, x= "duration",y = "age", color_feature = "age")

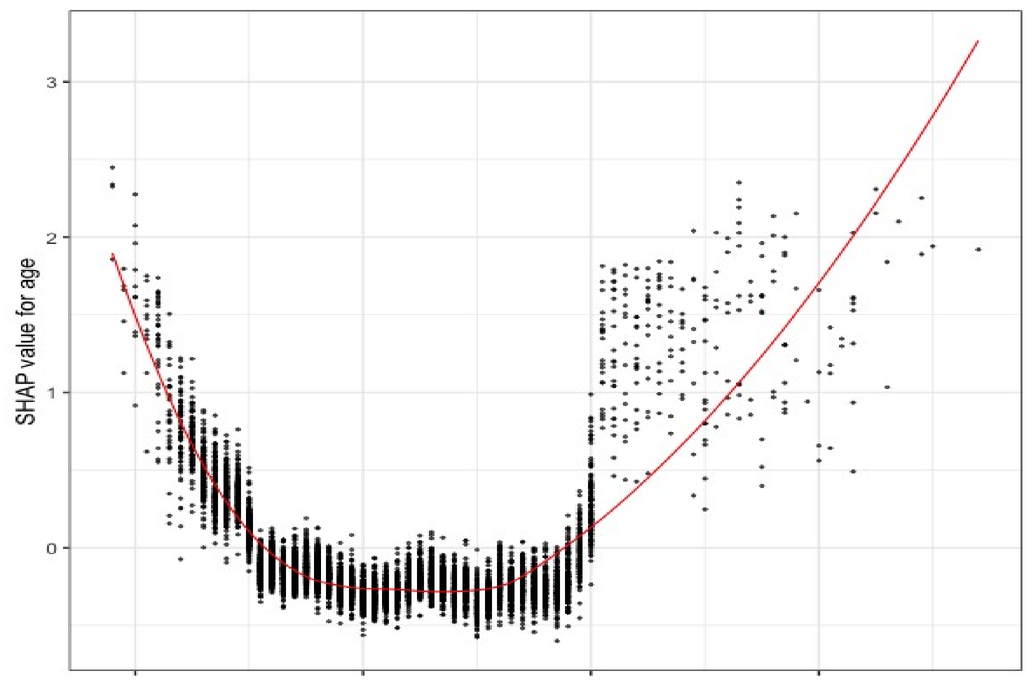


shap.plc

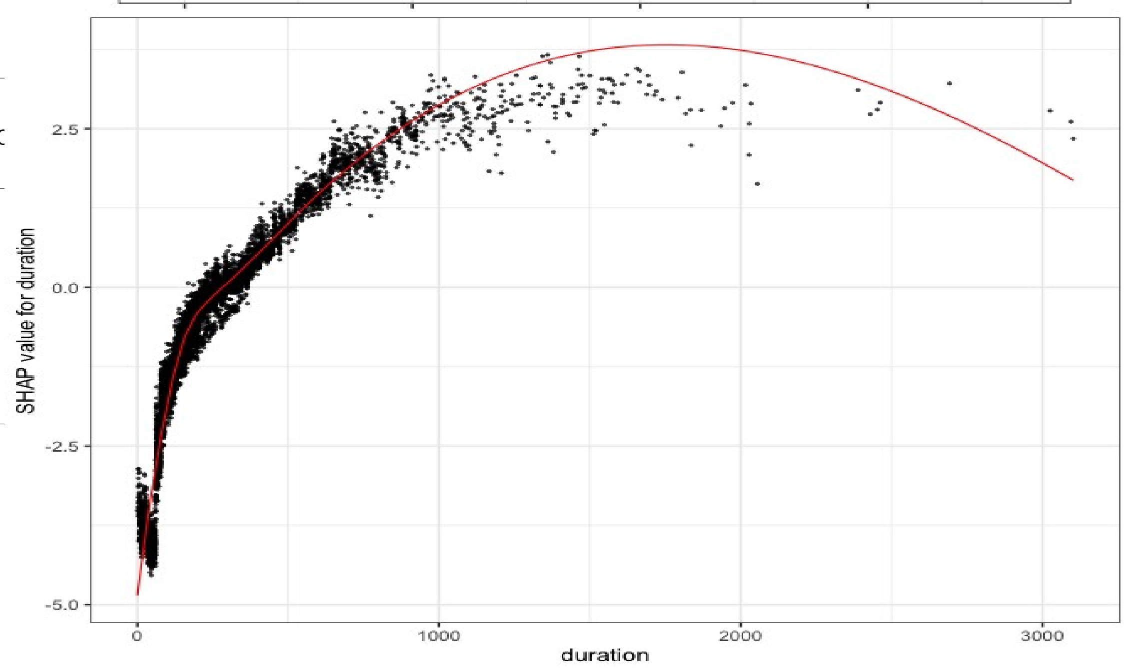


#18.5 without color version but has marginal distribution, just plot SHAP value against feature value

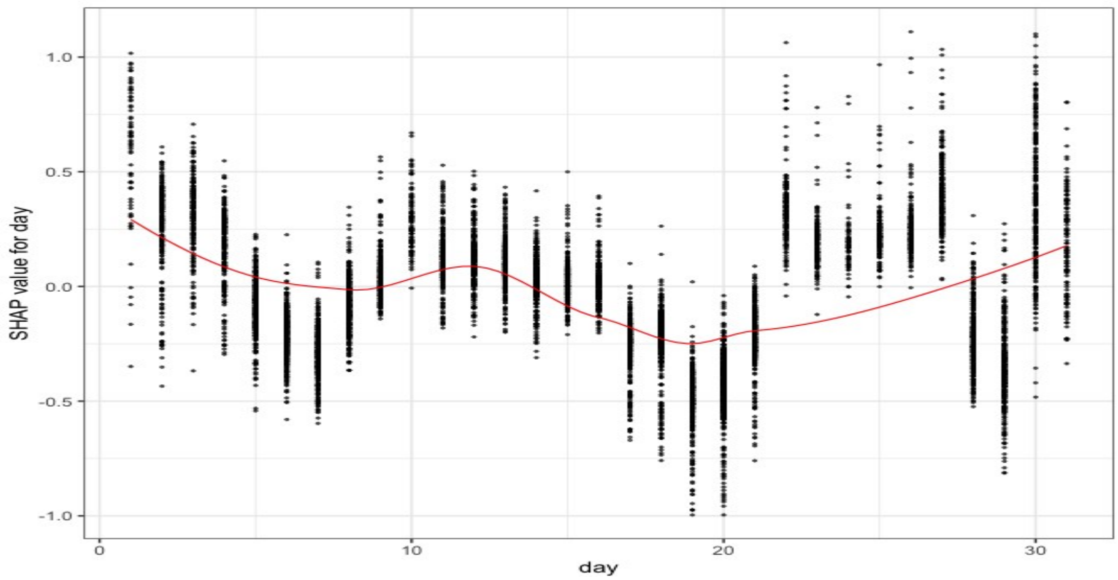
shap.plot.dependence(data_long = shap_long, "age")



shap.plo



shap.plot.dependence(data_long = shap_long, "day")



##18.6*** SHAP FORCE PLOT ***#####

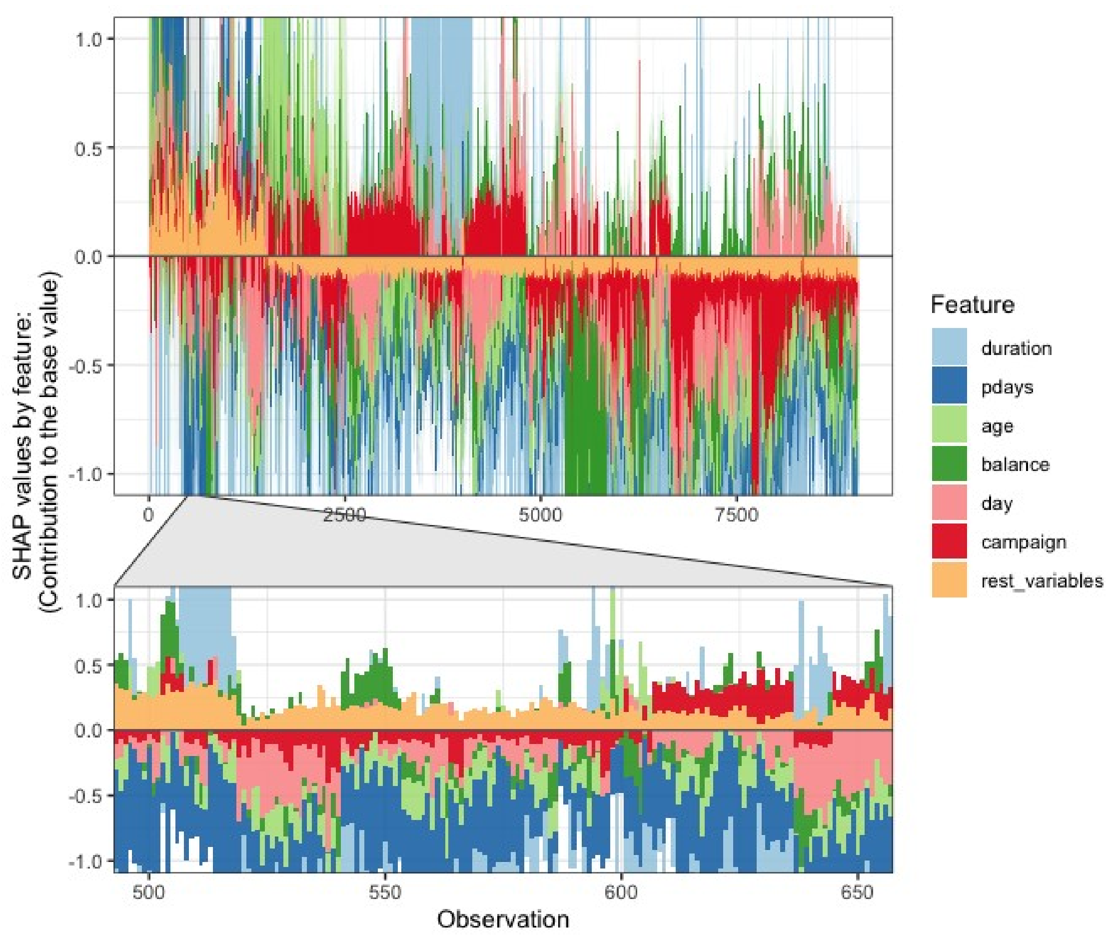
choose to show top 4 features by setting `top_n = 4`, set 6 clustering groups.

plot_data <- shap.prep.stack.data(shap_contrib = shap_values$shap_score, top_n = 6, n_groups = 6)

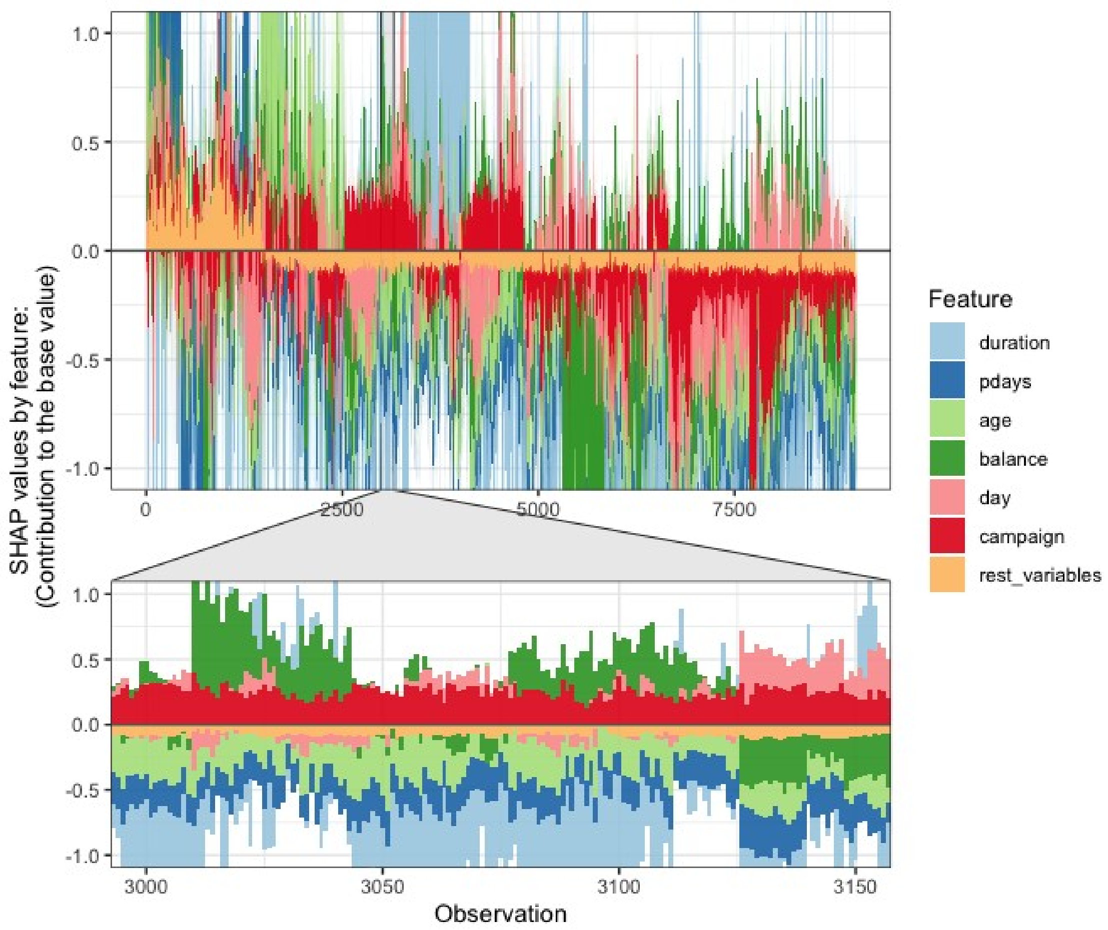
The SHAP values of the Rest 1 features were summed into variable 'rest_variables'.

choose to zoom in at location 500, set y-axis limit using `y_parent_limit`
it is also possible to set y-axis limit for zoom-in part alone using `y_zoomin_limit`

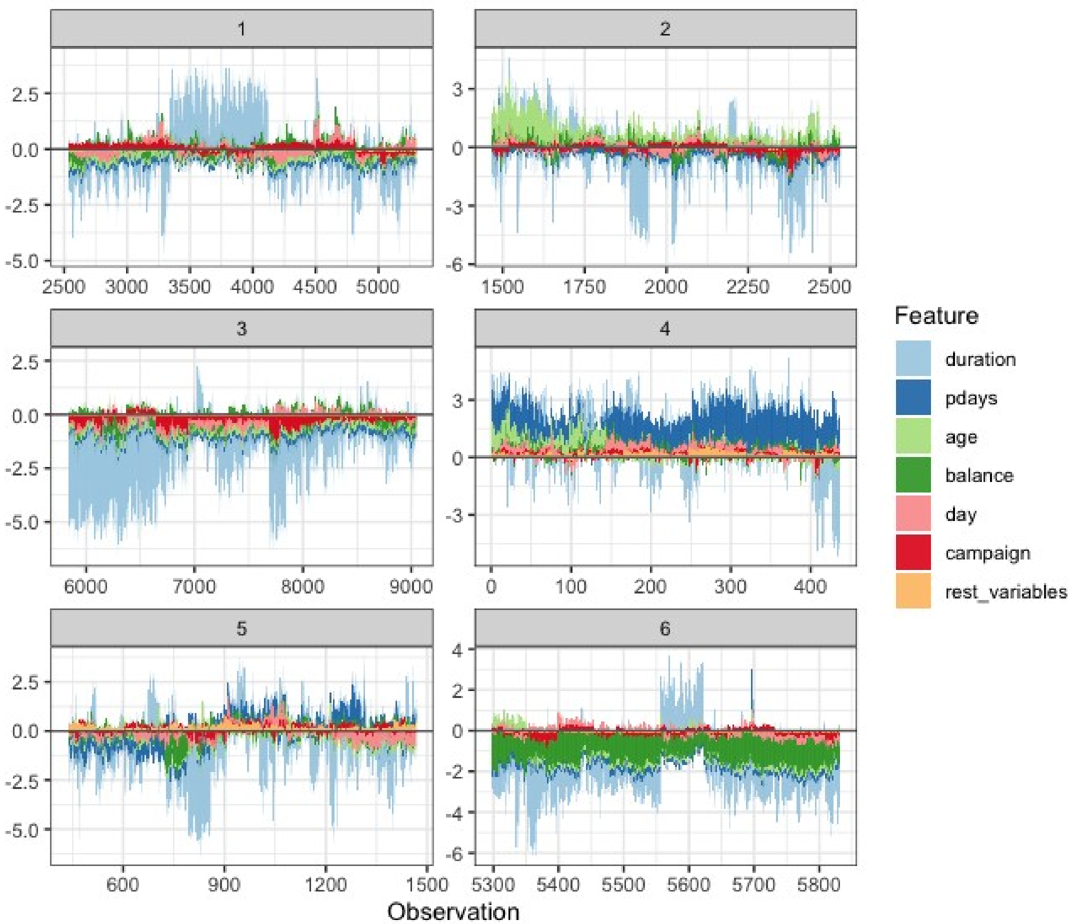
shap.plot.force_plot(plot_data, zoom_in_location = 500, y_parent_limit = c(-1,1))



shap.plot.force_plot(plot_data, zoom_in_location = 3000, y_parent_limit = c(-1,1))



shap.plot.force_plot_bygroup(plot_data)



CONCLUSIONS

- After the best tuning possible with the XGBoost algorithm, the variables that have more probability of influence for a "YES" answer to get a loan, according to the SHAP analysis are duration, past days, age and balance, without a causality effect.

- There are many more specific “possible” actions to take according to the business intuition mixed from the visual analysis, i.e:

- Shap values for age might show three different stages : Negative direction before the age of 40 vs positive direction for olders after 60: in the middle ages it does not has a direction of impact. ¿ Does it means a three segment strategy may be developed ?

- The duration for the first 1,000 days have a positive direction, but after the 2,000 days the diection of the impact is negative.¿ what is going on after the 2,000 days ?

- Balance has a low negative direction impact; with a a high concentration of less than 10,000 USD loans . The older group , older than 60 , seem to have a positive impact. ¿ What drives on a positive impact on the older than 60 segment, would it be possible to extend that input to the youngers, to change the low negative to a positive direction ?

- Day has a strong impact on every direction; this is a neutral variable

- The force plot graphs may help to discover any data pattern if at any circunstance it is necessary to go into detail