RMD XGboost

Benjamin GUIGON

12/17/2020

Pour que les programme fonctionne il faut faudra a minima ces library

Dans ces 3 problèmes je vais vous montrer différentes utilisations de XGboost. Il existe 2 façons d'utiliser XGboost : 1) la Regression 2) la Classification

L'une sert a faire de la prédiction direct un data set après un entrainement. L'autre sert à trouver les caractères d'importances majoritaires dans un data set suivant le résultat souhaité.

La 1er exemple utilise la regression. Nous allons recherche dans un data set, la Shap value.

Shap Value : "Grâce à la valeur de Shap, on peut déterminer l'effet des différentes variables d'une prédiction pour un modèle qui explique l'écart de cette prédiction par rapport à la valeur de base."

Dans ce premier exemple je vais chercher la shap value d'un data set tiré de stackoverflow pour savoir ce qui rendrait suseptible une personne de travailler en télé travail ou non.

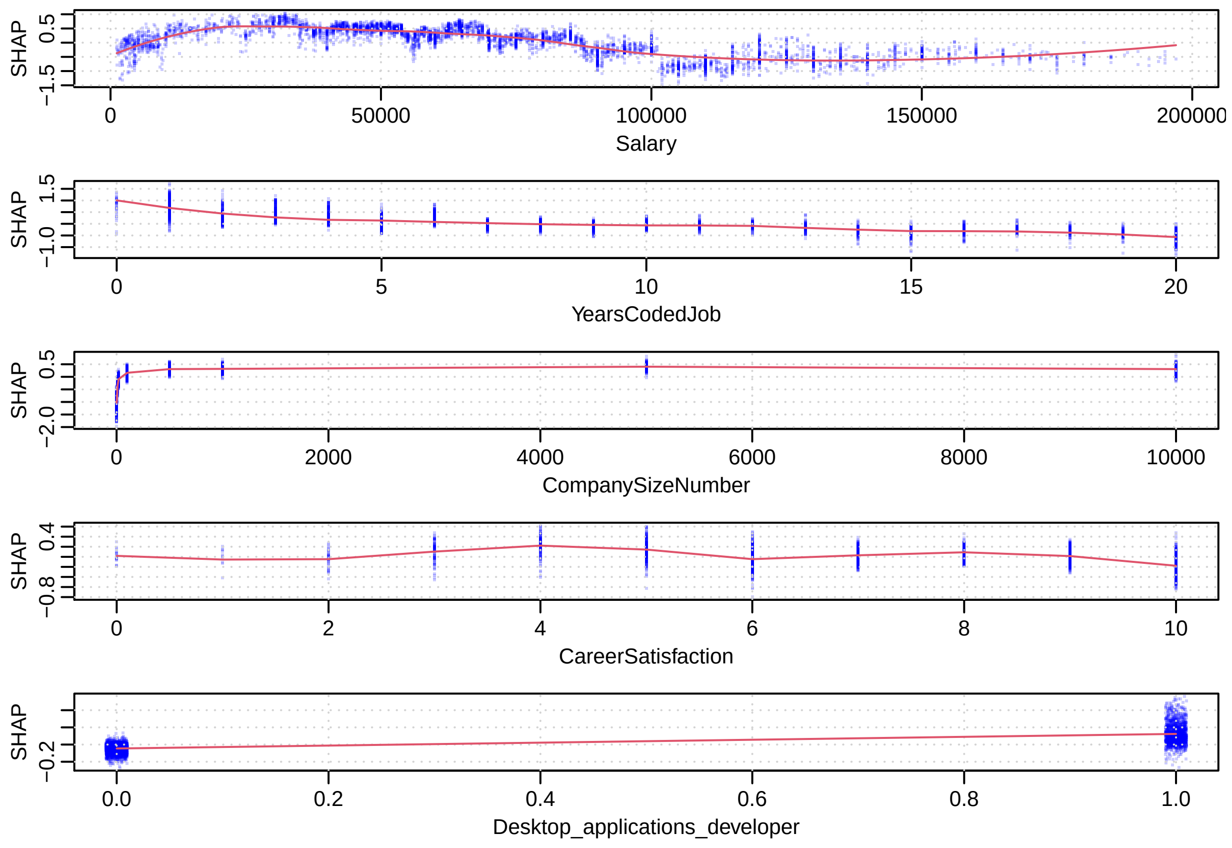
```r
data("stackoverflow")

# isolate X and Y
y = as.numeric(stackoverflow$Remote) - 1
X = stackoverflow %>% select(-Remote)
str(X)
```

```
## tibble [5,594 x 20] (S3: tbl_df/tbl/data.frame)
##  $ Country                             : Factor w/ 5 levels "Canada","Germany",..: 4 5 5 2 3 5 5 2 5
##  $ Salary                              : num [1:5594] 100000 130000 175000 64516 6636 ...
##  $ YearsCodedJob                       : int [1:5594] 20 20 16 4 1 1 13 4 7 17 ...
##  $ OpenSource                          : num [1:5594] 0 1 0 0 0 0 0 1 1 1 ...
##  $ Hobby                               : num [1:5594] 1 1 1 0 1 1 1 0 1 1 ...
##  $ CompanySizeNumber                   : num [1:5594] 5000 1000 10000 1000 5000 20 20 5000 20 20 ...
##  $ CareerSatisfaction                  : int [1:5594] 8 9 7 9 5 8 7 7 8 9 ...
##  $ Data_scientist                      : num [1:5594] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Database_administrator              : num [1:5594] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Desktop_applications_developer      : num [1:5594] 0 0 0 0 0 0 0 0 1 1 ...
##  $ Developer_with_stats_math_background: num [1:5594] 0 0 0 0 0 0 0 0 0 1 ...
##  $ DevOps                              : num [1:5594] 0 1 0 0 0 0 0 0 0 0 ...
##  $ Embedded_developer                  : num [1:5594] 1 1 0 0 0 0 0 0 0 0 ...
##  $ Graphic_designer                    : num [1:5594] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Graphics_programming                : num [1:5594] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Machine_learning_specialist         : num [1:5594] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Mobile_developer                    : num [1:5594] 0 0 0 0 0 0 0 0 1 0 ...
##  $ Quality_assurance_engineer          : num [1:5594] 0 1 0 0 0 0 0 0 0 0 ...
##  $ Systems_administrator               : num [1:5594] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Web_developer                       : num [1:5594] 0 1 1 1 1 1 1 1 0 0 ...
```

```r
# Transform factor into dummy variable

X = dummy_cols(X,
               remove_first_dummy = TRUE)
X = X %>% select(-Country)


# Setting the parameters
params = list(set.seed = 1997,
              eval_metric = "auc",
              objective = "binary:logistic")

# Running xgboost

model = xgboost(data = as.matrix(X),
                label = y,
                params = params,
                nrounds = 20,
                verbose = 1)
```

```
## [20:25:25] WARNING: amalgamation/../src/learner.cc:516:
## Parameters: { 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:0.708575
## [2]  train-auc:0.737982
## [3]  train-auc:0.767017
## [4]  train-auc:0.785276
## [5]  train-auc:0.800835
## [6]  train-auc:0.810605
## [7]  train-auc:0.819928
## [8]  train-auc:0.828242
## [9]  train-auc:0.838329
## [10] train-auc:0.847708
## [11] train-auc:0.856806
## [12] train-auc:0.865083
## [13] train-auc:0.870668
## [14] train-auc:0.873610
## [15] train-auc:0.879881
## [16] train-auc:0.886794
## [17] train-auc:0.890748
## [18] train-auc:0.898051
## [19] train-auc:0.899678
## [20] train-auc:0.903651
```

```r
# Shap value
xgb.plot.shap(data = as.matrix(X),
              model = model,
              top_n = 5)
```



On retrouve donc ici le graphe qui affiche la shap value. Nous voyons donc que les 5 paramètres qui ont le plus d'influances sont : le salaire, l'expérience, la taille de l'entreprise, la satisfaction de la carrière et le type d'application utilisée. Auriez-vous pariés sur ces paramètres avant de faire tourner le programme ? Personnellement, je n'aurais jamais mis la satisfaction de la carrière. Cette exemple nous montre bel et bien qu'il faut se mefier des a priori en data science.

Le 2ème exemple utilise un data set tiré des resultats d'un Sonar qui detecte les rochers aux matériaux plus précieux. Dans ce cas nous allons essayer de prédire a l'aide d'un découpage en entrainement et en teste si les pierres que nous scanons auraient bien été detectées par le sonar.

```
Sonar = read.csv(file = '/Users/benjamin.guigon/Desktop/PSB/Maths - R/R/Xgboost/sonar_csv.csv')

DataFrame = Sonar
dim(DataFrame)
```

```
## [1] 208  61
```

```
head(DataFrame,3)
```

```
##   attribute_1 attribute_2 attribute_3 attribute_4 attribute_5 attribute_6
## 1      0.0200      0.0371      0.0428      0.0207      0.0954      0.0986
## 2      0.0453      0.0523      0.0843      0.0689      0.1183      0.2583
## 3      0.0262      0.0582      0.1099      0.1083      0.0974      0.2280
##   attribute_7 attribute_8 attribute_9 attribute_10 attribute_11 attribute_12
## 1      0.1539      0.1601      0.3109       0.2111       0.1609       0.1582
## 2      0.2156      0.3481      0.3337       0.2872       0.4918       0.6552
## 3      0.2431      0.3771      0.5598       0.6194       0.6333       0.7060
##   attribute_13 attribute_14 attribute_15 attribute_16 attribute_17 attribute_18
## 1       0.2238       0.0645       0.0660       0.2273       0.3100       0.2999
## 2       0.6919       0.7797       0.7464       0.9444       1.0000       0.8874
## 3       0.5544       0.5320       0.6479       0.6931       0.6759       0.7551
##   attribute_19 attribute_20 attribute_21 attribute_22 attribute_23 attribute_24
## 1       0.5078       0.4797       0.5783       0.5071       0.4328       0.5550
## 2       0.8024       0.7818       0.5212       0.4052       0.3957       0.3914
## 3       0.8929       0.8619       0.7974       0.6737       0.4293       0.3648
##   attribute_25 attribute_26 attribute_27 attribute_28 attribute_29 attribute_30
## 1       0.6711       0.6415       0.7104       0.8080       0.6791       0.3857
## 2       0.3250       0.3200       0.3271       0.2767       0.4423       0.2028
## 3       0.5331       0.2413       0.5070       0.8533       0.6036       0.8514
##   attribute_31 attribute_32 attribute_33 attribute_34 attribute_35 attribute_36
## 1       0.1307       0.2604       0.5121       0.7547       0.8537       0.8507
## 2       0.3788       0.2947       0.1984       0.2341       0.1306       0.4182
## 3       0.8512       0.5045       0.1862       0.2709       0.4232       0.3043
##   attribute_37 attribute_38 attribute_39 attribute_40 attribute_41 attribute_42
## 1       0.6692       0.6097       0.4943       0.2744       0.0510       0.2834
## 2       0.3835       0.1057       0.1840       0.1970       0.1674       0.0583
## 3       0.6116       0.6756       0.5375       0.4719       0.4647       0.2587
##   attribute_43 attribute_44 attribute_45 attribute_46 attribute_47 attribute_48
## 1       0.2825       0.4256       0.2641       0.1386       0.1051       0.1343
## 2       0.1401       0.1628       0.0621       0.0203       0.0530       0.0742
## 3       0.2129       0.2222       0.2111       0.0176       0.1348       0.0744
##   attribute_49 attribute_50 attribute_51 attribute_52 attribute_53 attribute_54
## 1       0.0383       0.0324       0.0232       0.0027       0.0065       0.0159
## 2       0.0409       0.0061       0.0125       0.0084       0.0089       0.0048
## 3       0.0130       0.0106       0.0033       0.0232       0.0166       0.0095
##   attribute_55 attribute_56 attribute_57 attribute_58 attribute_59 attribute_60
## 1       0.0072       0.0167       0.0180       0.0084       0.0090       0.0032
## 2       0.0094       0.0191       0.0140       0.0049       0.0052       0.0044
## 3       0.0180       0.0244       0.0316       0.0164       0.0095       0.0078
##   Class
## 1  Rock
## 2  Rock
## 3  Rock
```

```
ind = createDataPartition(DataFrame$Class, p = 2/3, list = FALSE)

trainDF = DataFrame[ind,]
testDF = DataFrame[-ind,]

ControlParametres = trainControl(method = 'cv',
                                  number = 5,
                                  classProbs = TRUE)

parametersGrid = expand.grid(eta = 0.1,
                             colsample_bytree = c(0.5,0.7),
                             max_depth = c(3,6),
                             nrounds = 100,
                             gamma = 1,
                             min_child_weight = 2,
                             subsample = c(0,1,2))
```

```
parametersGrid
```

```
##    eta colsample_bytree max_depth nrounds gamma min_child_weight subsample
## 1  0.1              0.5         3     100     1                2         0
## 2  0.1              0.7         3     100     1                2         0
## 3  0.1              0.5         6     100     1                2         0
## 4  0.1              0.7         6     100     1                2         0
## 5  0.1              0.5         3     100     1                2         1
## 6  0.1              0.7         3     100     1                2         1
## 7  0.1              0.5         6     100     1                2         1
## 8  0.1              0.7         6     100     1                2         1
## 9  0.1              0.5         3     100     1                2         2
## 10 0.1              0.7         3     100     1                2         2
## 11 0.1              0.5         6     100     1                2         2
## 12 0.1              0.7         6     100     1                2         2
```

```
modelxgboost = train(Class~.,
                     data = trainDF,
                     method ="xgbTree",
                     trControl = ControlParametres,
                     tuneGrid = parametersGrid)
```

```
## Warning: model fit failed for Fold1: eta=0.1, max_depth=3, gamma=1, colsample_bytree=0.5, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold1: eta=0.1, max_depth=3, gamma=1, colsample_bytree=0.7, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold1: eta=0.1, max_depth=6, gamma=1, colsample_bytree=0.5, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold1: eta=0.1, max_depth=6, gamma=1, colsample_bytree=0.7, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold2: eta=0.1, max_depth=3, gamma=1, colsample_bytree=0.5, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold2: eta=0.1, max_depth=3, gamma=1, colsample_bytree=0.7, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold2: eta=0.1, max_depth=6, gamma=1, colsample_bytree=0.5, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.
## Warning: model fit failed for Fold2: eta=0.1, max_depth=6, gamma=1, colsample_bytree=0.7, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold3: eta=0.1, max_depth=3, gamma=1, colsample_bytree=0.5, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold3: eta=0.1, max_depth=3, gamma=1, colsample_bytree=0.7, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold3: eta=0.1, max_depth=6, gamma=1, colsample_bytree=0.5, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold3: eta=0.1, max_depth=6, gamma=1, colsample_bytree=0.7, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold4: eta=0.1, max_depth=3, gamma=1, colsample_bytree=0.5, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold4: eta=0.1, max_depth=3, gamma=1, colsample_bytree=0.7, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold4: eta=0.1, max_depth=6, gamma=1, colsample_bytree=0.5, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold4: eta=0.1, max_depth=6, gamma=1, colsample_bytree=0.7, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold5: eta=0.1, max_depth=3, gamma=1, colsample_bytree=0.5, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold5: eta=0.1, max_depth=3, gamma=1, colsample_bytree=0.7, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold5: eta=0.1, max_depth=6, gamma=1, colsample_bytree=0.5, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.

## Warning: model fit failed for Fold5: eta=0.1, max_depth=6, gamma=1, colsample_bytree=0.7, min_child_
##   value 2 for Parameter subsample exceed bound [0,1]
## subsample: Row subsample ratio of training instance.
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.

## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results
```

```
modelxgboost
```

```
## eXtreme Gradient Boosting
##
## 139 samples
##  60 predictor
##   2 classes: 'Mine', 'Rock'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 111, 112, 111, 111, 111
## Resampling results across tuning parameters:
##
##   max_depth  colsample_bytree  subsample  Accuracy   Kappa
##   3          0.5               0          0.5322751  0.0000000
##   3          0.5               1          0.8055556  0.6074665
##   3          0.5               2                NaN        NaN
##   3          0.7               0          0.5322751  0.0000000
##   3          0.7               1          0.7838624  0.5647355
##   3          0.7               2                NaN        NaN
##   6          0.5               0          0.5322751  0.0000000
##   6          0.5               1          0.7986772  0.5941347
##   6          0.5               2                NaN        NaN
##   6          0.7               0          0.5322751  0.0000000
##   6          0.7               1          0.7841270  0.5662015
##   6          0.7               2                NaN        NaN
##
## Tuning parameter 'nrounds' was held constant at a value of 100
## Tuning
##  'gamma' was held constant at a value of 1
## Tuning parameter
##  'min_child_weight' was held constant at a value of 2
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were nrounds = 100, max_depth = 3, eta
##  = 0.1, gamma = 1, colsample_bytree = 0.5, min_child_weight = 2 and subsample
##  = 1.
```

```
prediction = predict(modelxgboost,testDF)

t = table(predictions = prediction, actual = testDF$Class)
t
```

```
##            actual
## predictions Mine Rock
##        Mine   32    8
##        Rock    5   24
```

Grace la table de prediction, nous voyons les resultats de notre algorithme. Il a classé 58 réponses justes et 11 réponses fausses soit une précision de 84%

Enfin dans ce dernier cas nous allons essayer de predire si un élève sera admit ou non. Dans ce cas nous utiliserons les 2 premiers exemples, la regression et la classification. Nous allons à la fois trouver les paramètres d'influences essayer de prédire si des élèves serons admit ou non et voir la précision de notre algorithme.

```
# Data
#data = read.csv(file.choose(), header = T)
data = read.csv(file = '/Users/benjamin.guigon/Desktop/PSB/Maths - R/R/Xgboost/binary.csv', header = T)
str(data)
```

```
## 'data.frame':    400 obs. of  4 variables:
##  $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
```

```
data$rank = as.factor(data$rank)

# Partition data
set.seed(1234)
ind = sample(2, nrow(data), replace = T, prob = c(0.8, 0.2))
train = data[ind == 1,]
test = data[ind == 2,]


# Create matrix - One-Hot Encoding for Factor variables
trainm = sparse.model.matrix(admit~.-1, data = train)
train_label = train[,"admit"]
length(train_label)
```

```
## [1] 325
```

```
dim(trainm)
```

```
## [1] 325   6
```

```
train_matrix = xgb.DMatrix(data = as.matrix(trainm), label = train_label)

head(train_label)
```

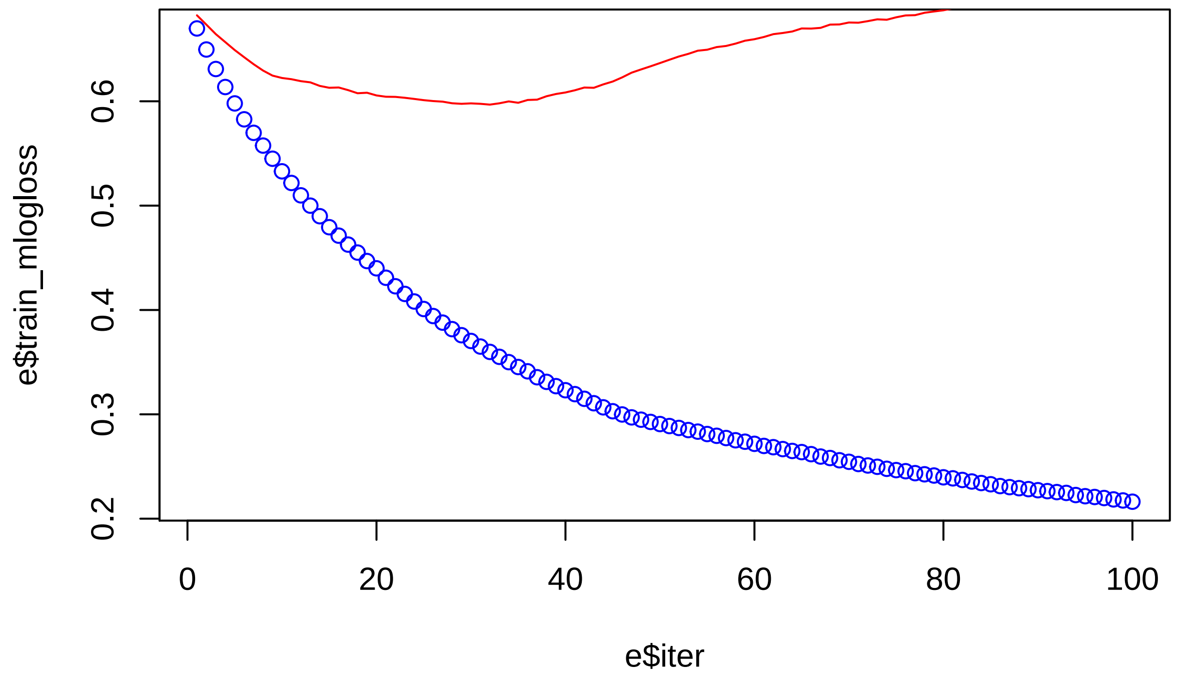
```
## [1] 0 1 1 1 1 1
```

```
testm = sparse.model.matrix(admit~.-1, data = test)
test_label = test[,"admit"]
test_matrix = xgb.DMatrix(data = as.matrix(testm), label = test_label)

#Parameters

nc = length(unique(train_label))
```

```r
xgb_params = list("objective" = "multi:softprob",
                  "eval_metric" = "mlogloss",
                  "num_class" = nc)

watchlist = list(train = train_matrix, test = test_matrix)

#eXtrem Gradient Boosting Model

bst_model = xgb.train(params = xgb_params,
                      data = train_matrix,
                      nrounds = 100,
                      watchlist = watchlist,
                      eta = 0.05,
                      max.depth = 8,
                      gamma = 0,
                      subsample = 1,
                      colsample_bytree = 1,
                      missing = NA,
                      set.seed = 333)
```

```
## [20:25:28] WARNING: amalgamation/../src/learner.cc:516:
## Parameters: { missing, 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-mlogloss:0.669793 test-mlogloss:0.682422
## [2]  train-mlogloss:0.649612 test-mlogloss:0.673427
## [3]  train-mlogloss:0.630930 test-mlogloss:0.664365
## [4]  train-mlogloss:0.613669 test-mlogloss:0.656742
## [5]  train-mlogloss:0.597961 test-mlogloss:0.649082
## [6]  train-mlogloss:0.582724 test-mlogloss:0.642229
## [7]  train-mlogloss:0.569788 test-mlogloss:0.635541
## [8]  train-mlogloss:0.557552 test-mlogloss:0.629436
## [9]  train-mlogloss:0.544959 test-mlogloss:0.624606
## [10] train-mlogloss:0.532880 test-mlogloss:0.622339
## [11] train-mlogloss:0.521725 test-mlogloss:0.621159
## [12] train-mlogloss:0.509861 test-mlogloss:0.619294
## [13] train-mlogloss:0.499991 test-mlogloss:0.618153
## [14] train-mlogloss:0.489870 test-mlogloss:0.614727
## [15] train-mlogloss:0.479323 test-mlogloss:0.612952
## [16] train-mlogloss:0.471251 test-mlogloss:0.613227
## [17] train-mlogloss:0.462669 test-mlogloss:0.610690
## [18] train-mlogloss:0.455030 test-mlogloss:0.607695
## [19] train-mlogloss:0.446884 test-mlogloss:0.608180
## [20] train-mlogloss:0.439936 test-mlogloss:0.605611
## [21] train-mlogloss:0.430883 test-mlogloss:0.604356
## [22] train-mlogloss:0.422730 test-mlogloss:0.604233
## [23] train-mlogloss:0.415391 test-mlogloss:0.603347
## [24] train-mlogloss:0.408160 test-mlogloss:0.602252
## [25] train-mlogloss:0.400941 test-mlogloss:0.601087
## [26] train-mlogloss:0.394185 test-mlogloss:0.600211
## [27] train-mlogloss:0.387901 test-mlogloss:0.599662
## [28] train-mlogloss:0.381644 test-mlogloss:0.598120
## [29] train-mlogloss:0.375820 test-mlogloss:0.597586
## [30] train-mlogloss:0.370314 test-mlogloss:0.598017
## [31] train-mlogloss:0.364952 test-mlogloss:0.597588
## [32] train-mlogloss:0.359843 test-mlogloss:0.596816
## [33] train-mlogloss:0.355097 test-mlogloss:0.598076
## [34] train-mlogloss:0.350030 test-mlogloss:0.599890
## [35] train-mlogloss:0.345329 test-mlogloss:0.598580
## [36] train-mlogloss:0.341215 test-mlogloss:0.601268
## [37] train-mlogloss:0.335534 test-mlogloss:0.601567
## [38] train-mlogloss:0.331117 test-mlogloss:0.604836
## [39] train-mlogloss:0.326992 test-mlogloss:0.606993
## [40] train-mlogloss:0.323091 test-mlogloss:0.608460
## [41] train-mlogloss:0.319353 test-mlogloss:0.610561
## [42] train-mlogloss:0.314877 test-mlogloss:0.613182
## [43] train-mlogloss:0.310630 test-mlogloss:0.612924
## [44] train-mlogloss:0.306688 test-mlogloss:0.616200
## [45] train-mlogloss:0.302953 test-mlogloss:0.618925
## [46] train-mlogloss:0.299872 test-mlogloss:0.622888
## [47] train-mlogloss:0.297074 test-mlogloss:0.627419
## [48] train-mlogloss:0.294867 test-mlogloss:0.630512
## [49] train-mlogloss:0.292720 test-mlogloss:0.633511
## [50] train-mlogloss:0.290696 test-mlogloss:0.636635
## [51] train-mlogloss:0.288775 test-mlogloss:0.639802
## [52] train-mlogloss:0.286950 test-mlogloss:0.642933
## [53] train-mlogloss:0.284936 test-mlogloss:0.645457
## [54] train-mlogloss:0.283414 test-mlogloss:0.648468
## [55] train-mlogloss:0.281119 test-mlogloss:0.649424
## [56] train-mlogloss:0.279433 test-mlogloss:0.651945
## [57] train-mlogloss:0.277284 test-mlogloss:0.653027
## [58] train-mlogloss:0.275127 test-mlogloss:0.655281
## [59] train-mlogloss:0.273767 test-mlogloss:0.658054
## [60] train-mlogloss:0.271745 test-mlogloss:0.659491
## [61] train-mlogloss:0.269713 test-mlogloss:0.661607
## [62] train-mlogloss:0.268512 test-mlogloss:0.664329
## [63] train-mlogloss:0.266692 test-mlogloss:0.665452
## [64] train-mlogloss:0.264918 test-mlogloss:0.666896
## [65] train-mlogloss:0.263810 test-mlogloss:0.669805
## [66] train-mlogloss:0.261830 test-mlogloss:0.669664
## [67] train-mlogloss:0.259532 test-mlogloss:0.670366
## [68] train-mlogloss:0.258138 test-mlogloss:0.673476
## [69] train-mlogloss:0.255990 test-mlogloss:0.673674
## [70] train-mlogloss:0.254498 test-mlogloss:0.675505
## [71] train-mlogloss:0.252400 test-mlogloss:0.675294
## [72] train-mlogloss:0.250970 test-mlogloss:0.676797
## [73] train-mlogloss:0.249724 test-mlogloss:0.678546
## [74] train-mlogloss:0.247788 test-mlogloss:0.678131
## [75] train-mlogloss:0.246553 test-mlogloss:0.680484
## [76] train-mlogloss:0.245505 test-mlogloss:0.682281
## [77] train-mlogloss:0.243637 test-mlogloss:0.682545
## [78] train-mlogloss:0.242518 test-mlogloss:0.684851
## [79] train-mlogloss:0.241352 test-mlogloss:0.686059
## [80] train-mlogloss:0.239643 test-mlogloss:0.687222
## [81] train-mlogloss:0.238638 test-mlogloss:0.689061
## [82] train-mlogloss:0.237094 test-mlogloss:0.690354
## [83] train-mlogloss:0.235635 test-mlogloss:0.691684
## [84] train-mlogloss:0.234167 test-mlogloss:0.692364
## [85] train-mlogloss:0.232996 test-mlogloss:0.693451
## [86] train-mlogloss:0.231238 test-mlogloss:0.693840
## [87] train-mlogloss:0.230224 test-mlogloss:0.693841
## [88] train-mlogloss:0.229237 test-mlogloss:0.694680
## [89] train-mlogloss:0.228304 test-mlogloss:0.695539
## [90] train-mlogloss:0.227292 test-mlogloss:0.696024
## [91] train-mlogloss:0.226379 test-mlogloss:0.696498
## [92] train-mlogloss:0.225471 test-mlogloss:0.696914
## [93] train-mlogloss:0.224674 test-mlogloss:0.697656
## [94] train-mlogloss:0.222648 test-mlogloss:0.698214
## [95] train-mlogloss:0.221540 test-mlogloss:0.699610
## [96] train-mlogloss:0.220790 test-mlogloss:0.700719
## [97] train-mlogloss:0.219793 test-mlogloss:0.702224
## [98] train-mlogloss:0.218456 test-mlogloss:0.702243
## [99] train-mlogloss:0.217464 test-mlogloss:0.700434
## [100]    train-mlogloss:0.216272 test-mlogloss:0.700554
```

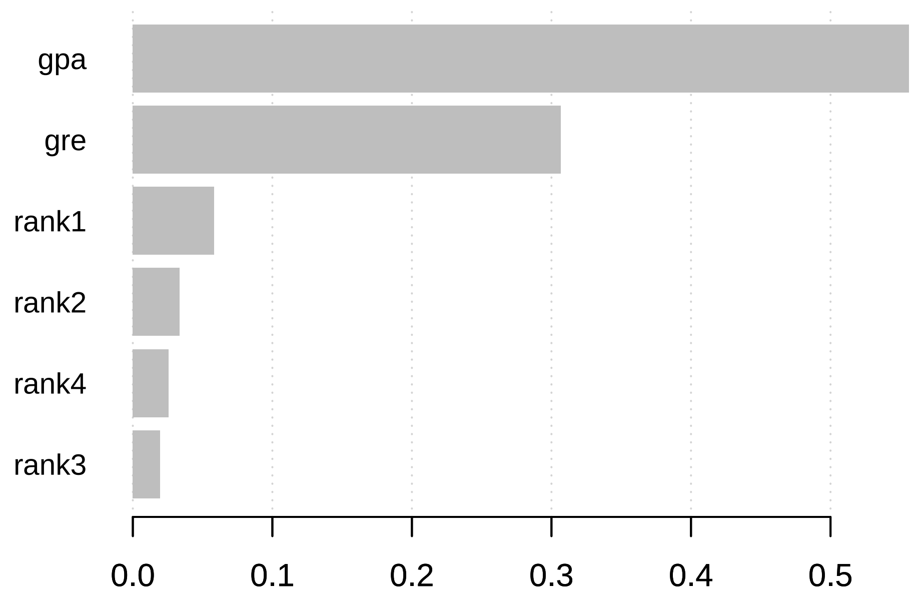
```
e = data.frame(bst_model$evaluation_log)
plot(e$iter,e$train_mlogloss, col = 'blue')
lines(e$iter,e$test_mlogloss, col = 'red')
```



```r
# Feature importance

imp = xgb.importance(colnames(train_matrix), model = bst_model)
print(imp)
```

```
##    Feature       Gain      Cover  Frequency
## 1:     gpa 0.55612893 0.54976202 0.51440329
## 2:     gre 0.30676939 0.29106938 0.34832451
## 3:   rank1 0.05830964 0.04250906 0.03233392
## 4:   rank2 0.03358177 0.03761700 0.03880071
## 5:   rank4 0.02566080 0.06590625 0.03409759
## 6:   rank3 0.01954948 0.01313630 0.03203998
```

```r
xgb.plot.importance(imp)
```



```r
# Prediction & confusion matrix - test data

p = predict(bst_model, newdata = test_matrix)
head(p)
```

```
## [1] 0.98687077 0.01312923 0.88708937 0.11291065 0.93908501 0.06091495
```

```r
pred = matrix(p, nrow = nc, ncol = length(p)/nc) %>%
  t() %>%
  data.frame() %>%
  mutate(label = test_label, max_prob = max.col(.,"last")-1)
head(pred)
```

```
##          X1         X2 label max_prob
## 1 0.9868708 0.01312923     0        0
## 2 0.8870894 0.11291065     0        0
## 3 0.9390850 0.06091495     0        0
## 4 0.2327828 0.76721722     1        1
## 5 0.6745946 0.32540542     1        0
## 6 0.1603446 0.83965546     1        1
```

```r
t = table(Prediction = pred$max_prob, Actual = pred$label)
t
```

```
##           Actual
## Prediction  0  1
##          0 42 16
##          1  8  9
```

En effet, nous avons ici les paramètres classés selon leurs influences sur le modèle. Ainsi dans la table de prédiction, nous avons classé 51 bonnes réponses et et 24 mauvaises réponse, soit une précision de 68%.

Dans un autre document vous pourrez retrouver une tutoriel sur le gradient boosting de manière manuscrite dans lequel j'explique comment fonctionne l'algorithme pour arriver à tous ces resultats.