Prudential_Life_Insurance_Assessment.R

puj83

Sun Jun 28 20:42:30 2020

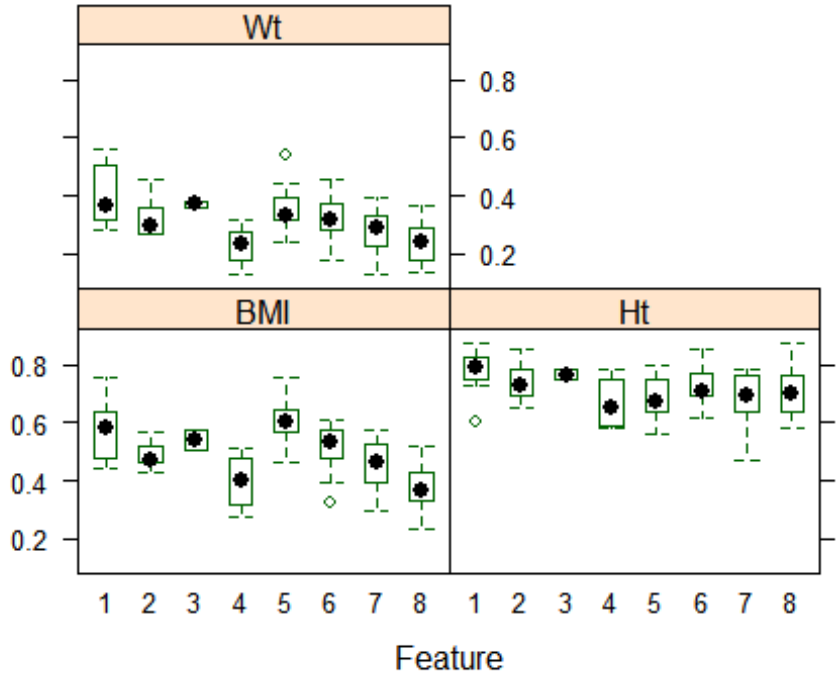
```
# Prudential Life Insurance Assessment

# Business Problem

# Picture this. You are a data scientist in a start-up culture with the
potential to have a very large impact on the business.
# Oh, and you are backed up by a company with 140 years' business experience.

# Curious? Great! You are the kind of person we are looking for.
#
# Prudential, one of the largest issuers of life insurance in the USA, is
hiring passionate data scientists to join a newly-formed Data Science group
solving complex challenges and identifying opportunities. The results have
been impressive so far but we want more.
#
# The Challenge
# In a one-click shopping world with on-demand everything, the life insurance
application process is antiquated.
# Customers provide extensive information to identify risk classification and
eligibility, including scheduling medical exams, a process that takes an
average of 30 days.
#
# The result? People are turned off. That's why only 40% of U.S. households
own individual life insurance.
# Prudential wants to make it quicker and less labor intensive for new and
existing customers to get a quote while maintaining privacy boundaries.
#
# By developing a predictive model that accurately classifies risk using a
more automated approach, you can greatly impact public perception of the
industry.
#
# The results will help Prudential better understand the predictive power of
the data points in the existing assessment, enabling us to significantly
streamline the process.

# install.packages("pillar")
# install.packages("dplyr")
# install.packages("tibble")
# install.packages("pdflatex")
# install.packages("ggpubr")
# install.packages("neuralnet")
# install.packages("ada")
```

```r
# install.packages("zoo")
# install.packages("ade4")
# install.packages("gtools")
# install.packages("xgboost")
# install.packages("forecast")
# install.packages("mlbench")
# install.packages("caret")
# install.packages("mlr")
# install.packages("data.table")
# install.packages("Metrics")

library(caret)

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

## Loading required package: ggplot2

library(corrplot)

## Warning: package 'corrplot' was built under R version 3.5.3

## corrplot 0.84 loaded

library(xgboost)

## Warning: package 'xgboost' was built under R version 3.5.3

library(stats)
library(knitr)
library(ggplot2)
library(Matrix)
library(plotly)

## Warning: package 'plotly' was built under R version 3.5.3

##
## Attaching package: 'plotly'

## The following object is masked from 'package:xgboost':
##
##     slice

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter
```

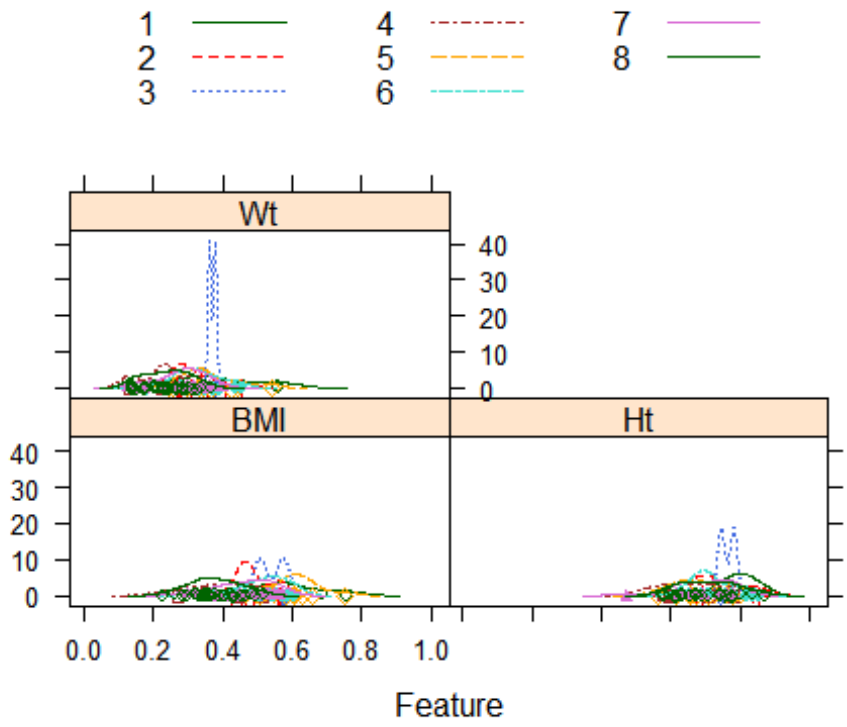
```
## The following object is masked from 'package:graphics':
##
##     layout

library(htmlwidgets)

## Warning: package 'htmlwidgets' was built under R version 3.5.3

library(readr)
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.5.3

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

library(data.table)

## Warning: package 'data.table' was built under R version 3.5.3

library(h2o)

## Warning: package 'h2o' was built under R version 3.5.3

##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit http://docs.h2o.ai
##
## ----------------------------------------------------------------------

##
## Attaching package: 'h2o'

## The following objects are masked from 'package:data.table':
##
##     hour, month, week, year
```

```
## The following objects are masked from 'package:stats':
##
##     cor, sd, var

## The following objects are masked from 'package:base':
##
##     %*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.5.3

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following object is masked from 'package:randomForest':
##
##     combine

## The following object is masked from 'package:xgboost':
##
##     slice

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(tidyr)

## Warning: package 'tidyr' was built under R version 3.5.3

##
## Attaching package: 'tidyr'

## The following objects are masked from 'package:Matrix':
##
##     expand, pack, unpack

library(Metrics)

## Warning: package 'Metrics' was built under R version 3.5.3
```

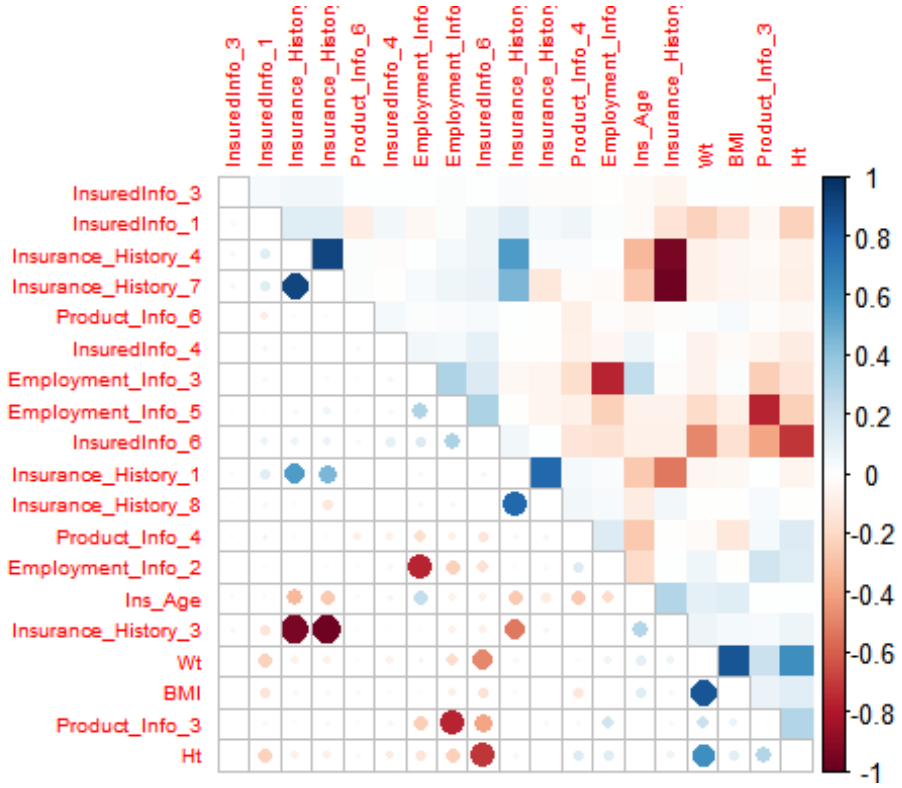
```
##
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':
##
##     precision, recall

################################################################################
##############################################
# Importing the data
################################################################################
##############################################

train1<-read.csv(file =
"C:/Users/puj83/OneDrive/Portfolio/Prudential_Life_Insurance_Assessment/train
.txt", header = T, sep = ",")
test1<-read.csv(file =
"C:/Users/puj83/OneDrive/Portfolio/Prudential_Life_Insurance_Assessment/test.
txt", header = T, sep = ",")

train<-train1
test<-test1

##### Remove id
train$Id<-NULL
test$Id<-NULL
# identify number of classes
num.class = length(levels(factor(unlist(train[,"Response"]))))
y = as.matrix(as.integer(unlist(train[,"Response"]))-1)

#####  Remove columns with NA, use test data as referal for NA
cols.without.na = colSums(is.na(train)) == 0
train = train[, cols.without.na]
cols.without.na = colSums(is.na(test)) == 0
test = test[, cols.without.na]
##### Check for zero variance
zero.var = nearZeroVar(train, saveMetrics=F)

train<-train[,-zero.var]
test<-test[, -zero.var]

##### Simple visualization
#x<-
as.data.frame(head(train[,c("BMI","Ht","Wt","Ins_Age","Product_Info_3")],100)
)
x<-as.data.frame(head(train[,c("BMI","Ht","Wt")],100))
y1<-factor(unlist(head(train[,"Response"],100)))
trellis.par.set(theme = col.whitebg(), warn = FALSE)
featurePlot(x, y1, "box",auto.key = list(columns = 3))
```




```r
featurePlot(x, y1, "density",
            #      scales = list(x = list(relation="free"),
            #                    y = list(relation="free")),
            #      adjust = 1.5,
            #     pch = "|",
            #      layout = c(4, 2),
            auto.key = list(columns = 3))
```

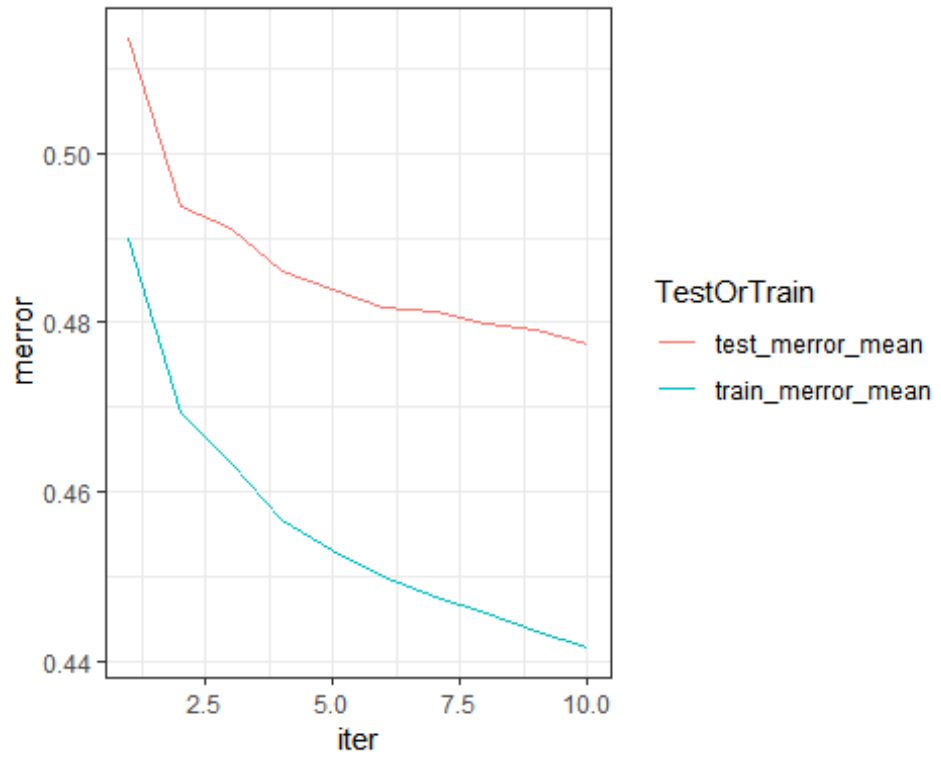



```
corrplot.mixed(cor(train[,c(2:20)]), lower="circle", upper="color",
               tl.pos="lt", tl.cex=0.6, diag="n", order="hclust",
hclust.method="complete")
```



```
##### convert data to matrix
train$Response = NULL
train.matrix = as.matrix(train)
mode(train.matrix) = "numeric"

## Warning in base::as.numeric(x): NAs introduced by coercion

test.matrix = as.matrix(test)
mode(test.matrix) = "numeric"

## Warning in base::as.numeric(x): NAs introduced by coercion

param <- list("objective" = "multi:softprob",    # multiclass classification
              "num_class" = num.class,    # number of classes
              "eval_metric" = "merror",
              "nthread" = 8,   # number of threads to be used
              "max_depth" = 8,    # maximum depth of tree
              "eta" = 0.1,    # step size shrinkage
              "gamma" = 0,    # minimum loss reduction
              "subsample"           = 0.7,
              "colsample_bytree"    = 0.7,
              "min_child_weight"    = 3
)

set.seed(789)

nround.cv = 10
system.time( bst.cv <- xgb.cv(param=param, data=train.matrix, label=y,
                              nfold=10, nrounds=nround.cv, prediction=TRUE,
verbose=T
                              #    callbacks = list(cb.cv.predict(save_models
= FALSE))
))

## [1]  train-merror:0.489973+0.010684  test-merror:0.513583+0.011333
## [2]  train-merror:0.469332+0.007503  test-merror:0.493829+0.008080
## [3]  train-merror:0.463322+0.007149  test-merror:0.491050+0.008741
## [4]  train-merror:0.456736+0.004953  test-merror:0.486132+0.007141
## [5]  train-merror:0.453151+0.003743  test-merror:0.483926+0.007937
## [6]  train-merror:0.449955+0.003766  test-merror:0.481872+0.006926
## [7]  train-merror:0.447616+0.003567  test-merror:0.481333+0.007061
## [8]  train-merror:0.445659+0.003383  test-merror:0.479801+0.007519
## [9]  train-merror:0.443589+0.002394  test-merror:0.479076+0.006445
## [10] train-merror:0.441643+0.002318  test-merror:0.477561+0.006434

##    user  system elapsed
##  229.32   11.02   44.08

bst.cv$evaluation_log %>%
  select(-contains("std")) %>%
  gather(TestOrTrain, merror,-iter) %>%
```

```
  ggplot(aes(x = iter, y = merror, group = TestOrTrain, color = TestOrTrain))
+
  geom_line() +
  theme_bw()
```




```
col.names<-colnames(bst.cv$evaluation_log)
setnames(bst.cv$evaluation_log, old = col.names, new =
c("iter","train.merror.mean","train.merror.std","test.merror.mean","test.merr
or.std" ))

min.merror.idx = which.min(bst.cv$evaluation_log[, test.merror.mean])

bst.cv$dt=bst.cv$evaluation_log
bst.cv$dt[min.merror.idx,]

##    iter train.merror.mean train.merror.std test.merror.mean
## 1:   10         0.4416434      0.002317548        0.4775606
##    test.merror.std
## 1:     0.006434291

pred.cv = matrix(bst.cv$pred, nrow=length(bst.cv$pred)/num.class,
ncol=num.class)
pred.cv = max.col(pred.cv, "last")

y<-factor(y+1)
pred.cv<-factor(pred.cv)
```

```
confusionMatrix(y, pred.cv)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2     3     4     5     6     7     8
##          1  1013   687     8     2   568  1497   688  1744
##          2   517  1335     4     1   845  1563   613  1674
##          3    51    37    14     2   302   427    36   144
##          4    25     7     0     8     2   583    73   730
##          5   158   422    17     0  2875  1099   282   579
##          6   367   317     0     4   534  5407  1338  3266
##          7   188    66     1     1    31  1787  2766  3187
##          8    71    46     0     5    26  1176   560 17605
##
## Overall Statistics
##
##                Accuracy : 0.5224
##                  95% CI : (0.5184, 0.5265)
##     No Information Rate : 0.4872
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3756
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2  Class: 3  Class: 4 Class: 5
## Sensitivity           0.42385  0.45766 0.3181818 0.3478261  0.55470
## Specificity           0.90886  0.90760 0.9831640 0.9760774  0.95282
## Pos Pred Value        0.16320  0.20375 0.0138203 0.0056022  0.52927
## Neg Pred Value        0.97410  0.97005 0.9994860 0.9997412  0.95722
## Prevalence            0.04025  0.04912 0.0007410 0.0003873  0.08728
## Detection Rate        0.01706  0.02248 0.0002358 0.0001347  0.04842
## Detection Prevalence  0.10453  0.11034 0.0170593 0.0240481  0.09148
## Balanced Accuracy     0.66636  0.68263 0.6506729 0.6619517  0.75376
##                      Class: 6 Class: 7 Class: 8
## Sensitivity           0.39936  0.43518   0.6086
## Specificity           0.87291  0.90078   0.9381
## Pos Pred Value        0.48135  0.34459   0.9033
## Neg Pred Value        0.83110  0.93009   0.7161
## Prevalence            0.22800  0.10704   0.4872
## Detection Rate        0.09106  0.04658   0.2965
## Detection Prevalence  0.18917  0.13518   0.3282
## Balanced Accuracy     0.63614  0.66798   0.7733
```

```
train<-train1
test<-test1
```

```r
# All features shared, making feature transformations simultaneously.
response <- train$Response
train$training <- 1
test$training  <- 0

data <- rbind(train[-c(1,128)], test[-1])
colnames(data)
```

```
##   [1] "Product_Info_1"      "Product_Info_2"      "Product_Info_3"
##   [4] "Product_Info_4"      "Product_Info_5"      "Product_Info_6"
##   [7] "Product_Info_7"      "Ins_Age"             "Ht"
##  [10] "Wt"                  "BMI"                 "Employment_Info_1"
##  [13] "Employment_Info_2"   "Employment_Info_3"   "Employment_Info_4"
##  [16] "Employment_Info_5"   "Employment_Info_6"   "InsuredInfo_1"
##  [19] "InsuredInfo_2"       "InsuredInfo_3"       "InsuredInfo_4"
##  [22] "InsuredInfo_5"       "InsuredInfo_6"       "InsuredInfo_7"
##  [25] "Insurance_History_1" "Insurance_History_2" "Insurance_History_3"
##  [28] "Insurance_History_4" "Insurance_History_5" "Insurance_History_7"
##  [31] "Insurance_History_8" "Insurance_History_9" "Family_Hist_1"
##  [34] "Family_Hist_2"       "Family_Hist_3"       "Family_Hist_4"
##  [37] "Family_Hist_5"       "Medical_History_1"   "Medical_History_2"
##  [40] "Medical_History_3"   "Medical_History_4"   "Medical_History_5"
##  [43] "Medical_History_6"   "Medical_History_7"   "Medical_History_8"
##  [46] "Medical_History_9"   "Medical_History_10"  "Medical_History_11"
##  [49] "Medical_History_12"  "Medical_History_13"  "Medical_History_14"
##  [52] "Medical_History_15"  "Medical_History_16"  "Medical_History_17"
##  [55] "Medical_History_18"  "Medical_History_19"  "Medical_History_20"
##  [58] "Medical_History_21"  "Medical_History_22"  "Medical_History_23"
##  [61] "Medical_History_24"  "Medical_History_25"  "Medical_History_26"
##  [64] "Medical_History_27"  "Medical_History_28"  "Medical_History_29"
##  [67] "Medical_History_30"  "Medical_History_31"  "Medical_History_32"
##  [70] "Medical_History_33"  "Medical_History_34"  "Medical_History_35"
##  [73] "Medical_History_36"  "Medical_History_37"  "Medical_History_38"
##  [76] "Medical_History_39"  "Medical_History_40"  "Medical_History_41"
##  [79] "Medical_Keyword_1"   "Medical_Keyword_2"   "Medical_Keyword_3"
##  [82] "Medical_Keyword_4"   "Medical_Keyword_5"   "Medical_Keyword_6"
##  [85] "Medical_Keyword_7"   "Medical_Keyword_8"   "Medical_Keyword_9"
##  [88] "Medical_Keyword_10"  "Medical_Keyword_11"  "Medical_Keyword_12"
##  [91] "Medical_Keyword_13"  "Medical_Keyword_14"  "Medical_Keyword_15"
##  [94] "Medical_Keyword_16"  "Medical_Keyword_17"  "Medical_Keyword_18"
##  [97] "Medical_Keyword_19"  "Medical_Keyword_20"  "Medical_Keyword_21"
## [100] "Medical_Keyword_22"  "Medical_Keyword_23"  "Medical_Keyword_24"
## [103] "Medical_Keyword_25"  "Medical_Keyword_26"  "Medical_Keyword_27"
## [106] "Medical_Keyword_28"  "Medical_Keyword_29"  "Medical_Keyword_30"
## [109] "Medical_Keyword_31"  "Medical_Keyword_32"  "Medical_Keyword_33"
## [112] "Medical_Keyword_34"  "Medical_Keyword_35"  "Medical_Keyword_36"
## [115] "Medical_Keyword_37"  "Medical_Keyword_38"  "Medical_Keyword_39"
## [118] "Medical_Keyword_40"  "Medical_Keyword_41"  "Medical_Keyword_42"
## [121] "Medical_Keyword_43"  "Medical_Keyword_44"  "Medical_Keyword_45"
```

```
## [124] "Medical_Keyword_46"  "Medical_Keyword_47"  "Medical_Keyword_48"
## [127] "training"
```

```
prop.table(table(response))
```

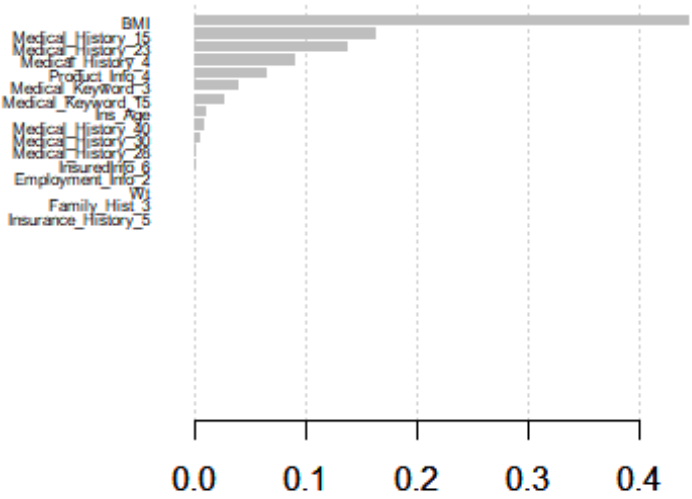
```
## response
##          1          2          3          4          5          6
## 0.10452838 0.11033832 0.01705933 0.02404810 0.09147707 0.18916825
##          7          8
## 0.13517792 0.32820262
```

```
feature.names <- names(data[-127])
for( f in feature.names ){
  if(class(data[[f]]) == "character"){
    print(class(data[[f]]))
    levels <- unique(c(train[[f]],test[[f]]))
    train[[f]] <- as.integer(factor(train[[f]]), levels = levels)
    test[[f]] <- as.integer(factor(test[[f]]), levels = levels)
    data[[f]] <- as.integer(factor(data[[f]]), levels = levels)

  }
}
```

```
data.roughfix <- na.roughfix(data)
y = as.matrix(as.integer(unlist(response))-1)
# Using training data to identify most important features with xgboost.
system.time(model_xgboost <- xgboost(data =
data.matrix(data.roughfix[data.roughfix$training==1,]),
                                     label  = y,
                                     nround  = 10,
                                     objective = "multi:softprob",
                                     eval_metric = "merror",
                                     num_class=8,
                                     eta = 0.01,  # learning rate
                                     max.depth = 3,
                                     missing = NaN,
                                     verbose = TRUE,
                                     print_every_n = 1,
                                     early_stopping_rounds = 10 ))
```

```
## [1]  train-merror:0.472912
## Will train until train_merror hasn't improved in 10 rounds.
##
## [2]  train-merror:0.472912
## [3]  train-merror:0.472895
## [4]  train-merror:0.472575
## [5]  train-merror:0.472777
## [6]  train-merror:0.465974
## [7]  train-merror:0.465974
## [8]  train-merror:0.465907
```

```
## [9]  train-merror:0.466159
## [10] train-merror:0.464980

##    user  system elapsed
##   23.14    0.27    3.37

model_dump <- xgb.dump(model_xgboost, with_stats = T)
importance.matrix <- xgb.importance(names(data.roughfix), model_xgboost)
xgb.plot.importance(importance.matrix[1:30])
```




```
medkeywords <- apply(data.roughfix[,79:126], 1, sum)
data.roughfix$medkeywords <- as.integer(medkeywords)
partition <- createDataPartition(response, times = 1, p = 0.75)
training <- data.roughfix[data.roughfix$training==1,]

y_train <- y[partition$Resample1,]
y_test <- y[-partition$Resample1,]

training_train <- training[partition$Resample1,-127]
training_test <- training[-partition$Resample1,-127]
system.time(model_xgboost <- xgboost(data = data.matrix(training_train),
                                     label  = y_train,
                                     nround  = 100,
                                     objective = "multi:softprob",
                                     eval_metric = "merror",
                                     num_class=8,
                                     eta = 0.01,
```

```
                                      max.depth = 3,
                                      missing = NaN,
                                      verbose = TRUE,
                                      print_every_n = 1,
                                      early_stopping_rounds = 10))
```

```
## [1]  train-merror:0.465601
## Will train until train_merror hasn't improved in 10 rounds.
##
## [2]  train-merror:0.464411
## [3]  train-merror:0.465466
## [4]  train-merror:0.465578
## [5]  train-merror:0.464703
## [6]  train-merror:0.465511
## [7]  train-merror:0.464456
## [8]  train-merror:0.457495
## [9]  train-merror:0.457271
## [10] train-merror:0.457293
## [11] train-merror:0.456934
## [12] train-merror:0.457181
## [13] train-merror:0.457383
## [14] train-merror:0.468699
## [15] train-merror:0.468991
## [16] train-merror:0.468834
## [17] train-merror:0.459224
## [18] train-merror:0.469238
## [19] train-merror:0.471035
## [20] train-merror:0.471910
## [21] train-merror:0.474335
## Stopping. Best iteration:
## [11] train-merror:0.456934
```

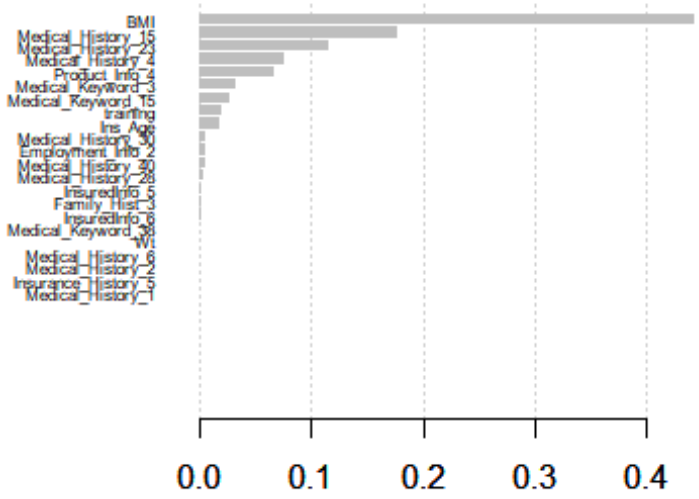
```
##    user  system elapsed
##   36.29    0.67    5.22
```

```
pred <- predict(model_xgboost, data.matrix(training_test), missing=NaN)
pred_m<- matrix(pred, nrow=length(pred)/num.class, ncol=num.class)
pred_m = max.col(pred_m, "last")
confusionMatrix(factor(y_test+1), factor(pred_m))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4   5   6   7   8
##          1 200 176 193 190 198 205 205 209
##          2 201 210 194 200 209 191 207 187
##          3  26  37  28  34  35  34  36  28
##          4  44  56  54  53  37  37  40  46
##          5 153 179 164 176 182 174 182 159
##          6 362 379 341 366 337 353 345 314
##          7 245 266 297 244 244 237 236 282
```

```
##          8 608 612 578 583 635 629 598 585
##
## Overall Statistics
##
##                Accuracy : 0.1244
##                  95% CI : (0.1191, 0.1298)
##     No Information Rate : 0.129
##     P-Value [Acc > NIR] : 0.9538
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity           0.10875  0.10966 0.015143  0.02871  0.09696  0.18978
## Specificity           0.89420  0.89258 0.982302  0.97584  0.90847  0.81178
## Pos Pred Value        0.12690  0.13133 0.108527  0.14441  0.13294  0.12621
## Neg Pred Value        0.87648  0.87128 0.875163  0.87616  0.87422  0.87492
## Prevalence            0.12388  0.12900 0.124554  0.12435  0.12644  0.12529
## Detection Rate        0.01347  0.01415 0.001886  0.00357  0.01226  0.02378
## Detection Prevalence  0.10616  0.10771 0.017380  0.02472  0.09222  0.18841
## Balanced Accuracy     0.50148  0.50112 0.498723  0.50228  0.50272  0.50078
##                      Class: 7 Class: 8
## Sensitivity            0.1276  0.32320
## Specificity            0.8603  0.67449
## Pos Pred Value         0.1151  0.12117
## Neg Pred Value         0.8739  0.87771
## Prevalence             0.1246  0.12193
## Detection Rate         0.0159  0.03941
## Detection Prevalence   0.1382  0.32523
## Balanced Accuracy      0.4940  0.49885
```

```
model_dump <- xgb.dump(model_xgboost, with_stats = T)
importance.matrix <- xgb.importance(names(data.roughfix), model_xgboost)
xgb.plot.importance(importance.matrix[1:30])
```




```
categorical_string <- as.character("Product_Info_1, Product_Info_2,
Product_Info_3, Product_Info_5, Product_Info_6, Product_Info_7,
Employment_Info_2, Employment_Info_3, Employment_Info_5, InsuredInfo_1,
InsuredInfo_2, InsuredInfo_3, InsuredInfo_4, InsuredInfo_5, InsuredInfo_6,
InsuredInfo_7, Insurance_History_1, Insurance_History_2, Insurance_History_3,
Insurance_History_4, Insurance_History_7, Insurance_History_8,
Insurance_History_9, Family_Hist_1, Medical_History_2, Medical_History_3,
Medical_History_4, Medical_History_5, Medical_History_6, Medical_History_7,
Medical_History_8, Medical_History_9, Medical_History_11, Medical_History_12,
Medical_History_13, Medical_History_14, Medical_History_16,
Medical_History_17, Medical_History_18, Medical_History_19,
Medical_History_20, Medical_History_21, Medical_History_22,
Medical_History_23, Medical_History_25, Medical_History_26,
Medical_History_27, Medical_History_28, Medical_History_29,
Medical_History_30, Medical_History_31, Medical_History_33,
Medical_History_34, Medical_History_35, Medical_History_36,
Medical_History_37, Medical_History_38, Medical_History_39,
Medical_History_40, Medical_History_41")
categorical_names <- unlist(strsplit(categorical_string, split = ", "))
top30features <- importance.matrix$Feature[1:30]
which(top30features %in% categorical_names)

##  [1]  3  4 10 11 12 13 14 16 19 20

top30categorical_names <- top30features[which(top30features %in%
categorical_names)]
# One-hot encoding top 15 categorical variables
```

```
top30categorical_factor <-
as.data.frame(apply(data.roughfix[,top30categorical_names],2,as.factor))
categorical_one_hot <- as.data.frame(model.matrix(~.-1,
top30categorical_factor[-8])) # Except Medical_History_2 which has too many
levels.
categorical_one_hot2 <- as.data.frame(sapply(categorical_one_hot,as.factor))
str(categorical_one_hot2)
```

```
## 'data.frame':    79146 obs. of  677 variables:
##  $ Medical_History_231 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2
...
##  $ Medical_History_232 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_233 : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 1
...
##  $ Medical_History_42  : Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 2 2 2
...
##  $ Medical_History_302 : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2
...
##  $ Medical_History_303 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_210 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_211 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_212 : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 2 1 1
...
##  $ Employment_Info_213 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_214 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_215 : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1
...
##  $ Employment_Info_216 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_217 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_218 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_219 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_22  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_220 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_221 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_222 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_223 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
```

```
...
##  $ Employment_Info_224 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_225 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_226 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_227 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_228 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_229 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_23  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_230 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_231 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_232 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_233 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_234 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_235 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_236 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_237 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_238 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_24  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_25  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_26  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_27  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_28  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Employment_Info_29  : Factor w/ 2 levels "0","1": 1 1 2 2 2 1 1 1 2 1
...
##  $ Medical_History_402 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_403 : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2
...
##  $ Medical_History_282 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
```

```
...
##  $ Medical_History_283 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ InsuredInfo_53      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_62  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_63  : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2
...
##  $ Medical_History_210 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2100: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2101: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2102: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2103: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2104: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2105: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2106: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2107: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2108: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2109: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_211 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2110: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2111: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2112: Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2113: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2114: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2115: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2116: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2117: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2119: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
```

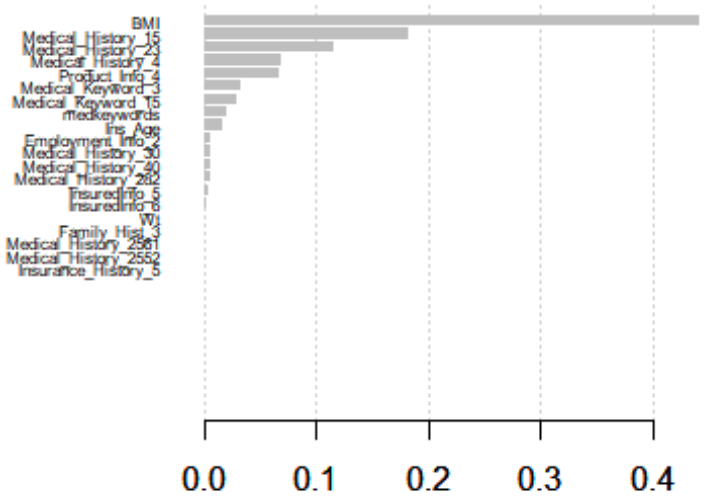
```
...
##  $ Medical_History_212 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2120: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2121: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2122: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2123: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2124: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2125: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2126: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2127: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2128: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2129: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_213 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2131: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2132: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2133: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2134: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2135: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2136: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2137: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2138: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2139: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_214 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2140: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2141: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2142: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
```

```
...
##  $ Medical_History_2143: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2144: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Medical_History_2145: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1
...
##   [list output truncated]
```

```
data.roughfix2 <- cbind(data.roughfix, categorical_one_hot2)

system.time(model2 <- xgboost(data =
data.matrix(data.roughfix2[data.roughfix2$training==1,]),
                              label  = y,
                              nround  = 100,
                              objective = "multi:softprob",
                              eval_metric = "merror",
                              num_class=8,
                              eta = 0.01,
                              max.depth = 3,
                              missing = NaN,
                              verbose = TRUE,
                              print_every_n = 1,
                              early_stopping_rounds = 10 ))
```

```
## [1]  train-merror:0.471615
## Will train until train_merror hasn't improved in 10 rounds.
##
## [2]  train-merror:0.471615
## [3]  train-merror:0.471599
## [4]  train-merror:0.471531
## [5]  train-merror:0.471531
## [6]  train-merror:0.464610
## [7]  train-merror:0.464745
## [8]  train-merror:0.464677
## [9]  train-merror:0.464492
## [10] train-merror:0.474007
## [11] train-merror:0.463330
## [12] train-merror:0.473266
## [13] train-merror:0.473721
## [14] train-merror:0.473721
## [15] train-merror:0.473603
## [16] train-merror:0.473721
## [17] train-merror:0.473518
## [18] train-merror:0.473266
## [19] train-merror:0.473215
## [20] train-merror:0.473266
## [21] train-merror:0.473182
## Stopping. Best iteration:
## [11] train-merror:0.463330
```

```
##    user  system elapsed
##  285.91    1.20   38.83

model_dump <- xgb.dump(model2, with_stats = T)
importance.matrix <- xgb.importance(names(data.roughfix2), model2)
xgb.plot.importance(importance.matrix[1:30])
```




```
folds <- createFolds(response, 2)
training <- data.roughfix[data.roughfix$training == 1,]
cv_results <- lapply(folds, function(x){
  train <- data.matrix(training[-x,])
  test <- data.matrix(training[x,])
  model <- xgboost(data = train,
                   label = y[-x],
                   nround  = 100,
                   objective = "multi:softprob",
                   eval_metric = "merror",
                   num.class=8,
                   eta = 0.01,
                   max.depth = 3,
                   missing = NaN,
                   verbose = TRUE,
                   print_every_n = 1,
                   early_stopping_rounds = 10
  )

  model_pred <- predict(model, test, missing=NaN)
```

```r
  pred_m<- matrix(model_pred, nrow=length(model_pred)/num.class,
ncol=num.class)
  pred_m = max.col(pred_m, "last")
  actual <- response[x]
  qwkappa <- Metrics::ScoreQuadraticWeightedKappa(actual, pred_m)
  print(qwkappa)
  return(qwkappa)
})
```

```
## [1]  train-merror:0.480112
## Will train until train_merror hasn't improved in 10 rounds.
##
## [2]  train-merror:0.480112
## [3]  train-merror:0.479977
## [4]  train-merror:0.479977
## [5]  train-merror:0.478933
## [6]  train-merror:0.478866
## [7]  train-merror:0.472163
## [8]  train-merror:0.472264
## [9]  train-merror:0.472332
## [10] train-merror:0.472163
## [11] train-merror:0.472332
## [12] train-merror:0.472399
## [13] train-merror:0.471995
## [14] train-merror:0.482267
## [15] train-merror:0.482301
## [16] train-merror:0.480044
## [17] train-merror:0.481223
## [18] train-merror:0.481089
## [19] train-merror:0.479741
## [20] train-merror:0.479741
## [21] train-merror:0.477586
## [22] train-merror:0.477417
## [23] train-merror:0.477451
## Stopping. Best iteration:
## [13] train-merror:0.471995
##
## [1] -0.009338625
## [1]  train-merror:0.468609
## Will train until train_merror hasn't improved in 10 rounds.
##
## [2]  train-merror:0.467632
## [3]  train-merror:0.467733
## [4]  train-merror:0.461570
## [5]  train-merror:0.461367
## [6]  train-merror:0.470731
## [7]  train-merror:0.471169
## [8]  train-merror:0.471101
## [9]  train-merror:0.471135
## [10] train-merror:0.471034
```

```
## [11] train-merror:0.471303
## [12] train-merror:0.471101
## [13] train-merror:0.471303
## [14] train-merror:0.471472
## [15] train-merror:0.471405
## Stopping. Best iteration:
## [5]  train-merror:0.461367
##
## [1] 0.006995061

cv_results

## $Fold1
## [1] -0.009338625
##
## $Fold2
## [1] 0.006995061
```