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")  
# 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

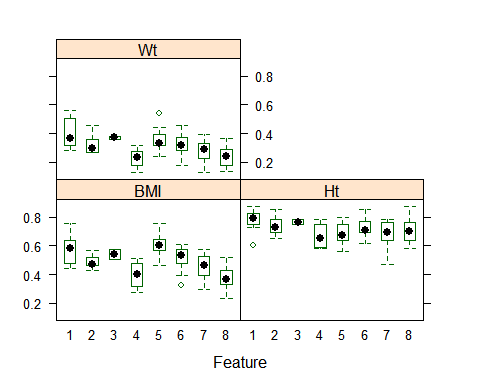
library(Metrics)

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

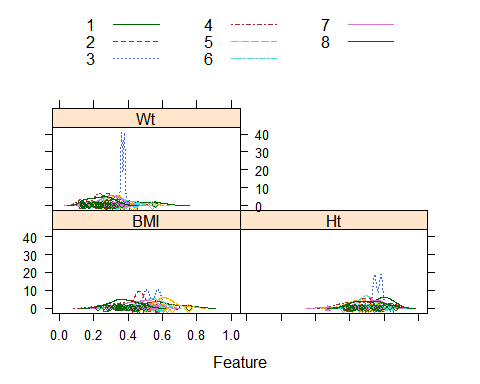
##   
## Attaching package: 'Metrics'

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

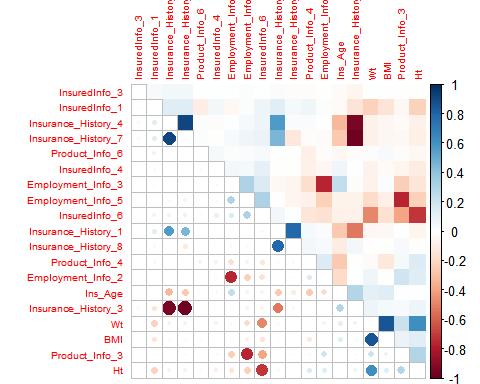
########################################################################################################################  
# Importing the data  
########################################################################################################################  
  
train1<-read.csv(file = "C:/Users/puj83/OneDrive/Portfolio/Prudential\_Life\_Insurance\_Assessment/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))



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"

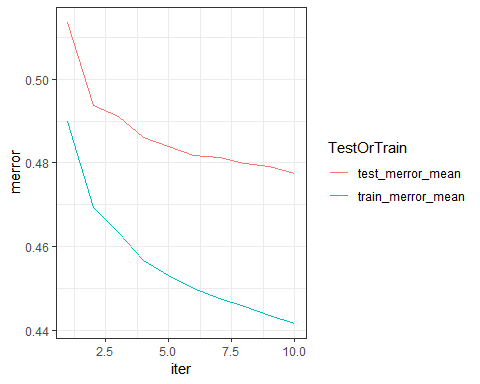
## Warning in base::as.numeric(x): NAs introduced by coercion

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.merror.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  
  
# 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))

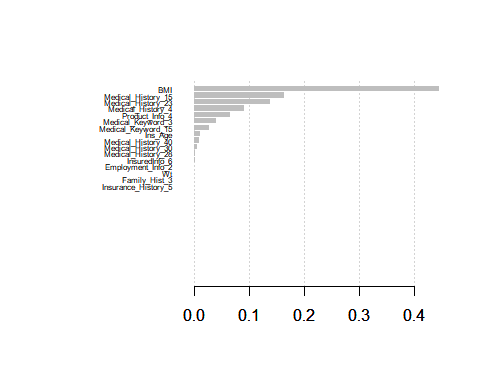
## response  
## 1 2 3 4 5 6   
## 0.10452838 0.11033832 0.01705933 0.02404810 0.09147707 0.18916825   
## 7 8   
## 0.13517792 0.32820262

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))

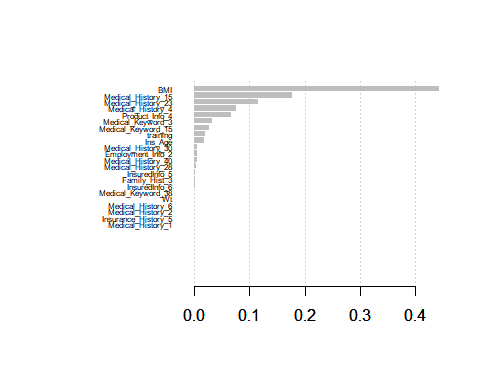
## [1] train-merror:0.465601   
## Will train until train\_merror hasn't improved in 10 rounds.  
##   
## [2] train-merror:0.464411   
## [3] train-merror:0.465466   
## [4] train-merror:0.465578   
## [5] train-merror:0.464703   
## [6] train-merror:0.465511   
## [7] train-merror:0.464456   
## [8] 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)

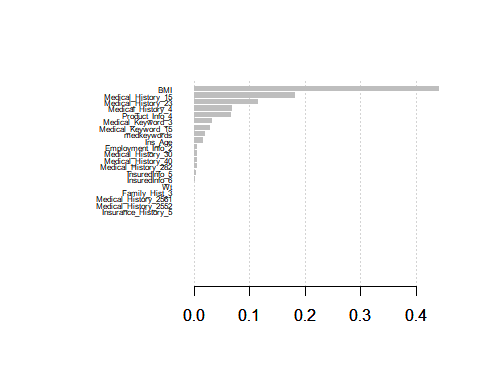
## 'data.frame': 79146 obs. of 677 variables:  
## $ Medical\_History\_231 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...  
## $ Medical\_History\_232 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Medical\_History\_233 : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 1 ...  
## $ 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)  
 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