Simple_Predictive_Modeling_with_SWAT

```r
# Loading the required SWAT package and other R libraries necessary
library(swat)
```

```
## NOTE: The extension module for binary protocol support is not available.
```

```
##       Only the CAS REST interface can be used.
```

```
## SWAT 1.4.0
```

```r
library(ggplot2)
library(reshape2)
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 3.4.4
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
##
## Attaching package: 'dplyr'
```

```
## 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
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'

## The following object is masked from 'package:swat':
##
##     cov

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.4.4
```

```r
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 3.4.4

## Loading required package: gplots

## Warning: package 'gplots' was built under R version 3.4.4

##
## Attaching package: 'gplots'

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

```r
library(Rcpp)
```

```
## Warning: package 'Rcpp' was built under R version 3.4.4
```

```r
# Connect to CAS server using appropriate credentials

s = CAS()
```

```
## NOTE: Connecting to CAS and generating CAS action functions for loaded

##       action sets...

## NOTE: To generate the functions with signatures (for tab completion), set

##       options(cas.gen.function.sig=TRUE).
```

```r
# Create a CAS library called lg pointing to the defined directory
# Need to specify the srctype as path, otherwise it defaults to HDFS

cas.table.addCaslib(s,
                    name = "lg",
                    description = "Looking glass data",
                    dataSource = list(srcType="path"),
                    path = "/viyafiles/tmp"
                   )
```

```
## NOTE: 'lg' is now the active caslib.

## NOTE: Cloud Analytic Services added the caslib 'lg'.

## $CASLibInfo
##   Name Type        Description            Path Definition Subdirs Local
## 1   lg PATH Looking glass data /viyafiles/tmp/                  0     1
##   Active Personal Hidden Transient
## 1      1        0      0         0
```

```r
# Load the data into the in-memory CAS server

data = cas.read.csv(s,
                     "C:/Users/Looking_glass.csv",
                     casOut=list(name="castbl", caslib="lg", replace=TRUE)
                     )
```

```
## NOTE: Cloud Analytic Services made the uploaded file available as table CASTBL in caslib lg.
```

```r
# Invoke the overloaded R functions to view the head and summary of the input table

print(head(data))
```

```
##   lifetime_value calls_in_offpk mou_onnet_pct_MOM mb_data_usg_m01
## 1         9616.9         604.38                 0        1388.947
## 2         7619.3         793.57                 0        2930.470
## 3         2765.7         529.50                 0          69.000
## 4         6426.5         333.39                 1        1739.512
## 5         5372.8         -16.42                 0        1075.152
## 6         1746.9         364.10                 0        1191.598
##   mb_data_usg_m02 mb_data_usg_m03       region upsell_xsell
## 1        1243.291        1299.693      Pacific            0
## 2        2856.150        3030.931    Southwest            0
## 3         431.056         412.150 Mid Atlantic            0
## 4        1766.006        1702.673      Midwest            0
## 5         854.023         829.591        South            0
## 6        1222.585        1254.263      Pacific            0
##   ever_days_over_plan ever_times_over_plan avg_days_susp
## 1                   2                    6             6
## 2                  10                    1             5
## 3                   9                    2             0
```

```
## 4                  0                    2            4
## 5                 11                    5            2
## 6                 14                    3           12
##   mou_onnet_6m_normal unsolv_tsupcomplnt wrk_orders days_openwrkorders
## 1                   0                  0          0                 15
## 2                   0                  0          0                  0
## 3                  -3                  0          0                 11
## 4                  -2                  0          0                  0
## 5                   1                  1          0                 16
## 6                   0                  0          0                  6
```

```
print(summary(data))
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4

## Selecting by Frequency

##  lifetime_value   calls_in_offpk    mou_onnet_pct_MOM  mb_data_usg_m01
##  Min.   :-14006   Min.   :-1410.3   Min.   :-45.0000   Min.   :-2425.0
##  1st Qu.:  1587   1st Qu.:  123.9   1st Qu.: -0.5280   1st Qu.:  540.2
##  Median :  3822   Median :  296.1   Median :  0.0000   Median : 1425.0
##  Mean   :  5281   Mean   :  388.6   Mean   : -0.1368   Mean   : 1697.2
##  3rd Qu.:  7435   3rd Qu.:  545.5   3rd Qu.:  0.0000   3rd Qu.: 2417.2
##  Max.   : 60740   Max.   : 4640.2   Max.   :124.7270   Max.   :40568.7
##
##  mb_data_usg_m02   mb_data_usg_m03            region
##  Min.   :-2171.1   Min.   :-1621.0   Great Lakes   :10900
##  1st Qu.:  538.7   1st Qu.:  535.2   South         :10580
##  Median : 1431.1   Median : 1422.9   Mid Atlantic :10357
##  Mean   : 1698.6   Mean   : 1696.2   Pacific       : 9157
##  3rd Qu.: 2418.3   3rd Qu.: 2417.5   Greater Texas: 7236
##  Max.   :40761.3   Max.   :40784.2
##
##   upsell_xsell    ever_days_over_plan ever_times_over_plan
##  Min.   :0.0000   Min.   : 0.00       Min.   : 0.00
##  1st Qu.:0.0000   1st Qu.: 0.00       1st Qu.: 0.00
##  Median :0.0000   Median : 9.00       Median : 2.00
##  Mean   :0.1213   Mean   :13.65       Mean   : 2.53
##  3rd Qu.:0.0000   3rd Qu.:22.00       3rd Qu.: 4.00
##  Max.   :1.0000   Max.   :99.00       Max.   :26.00
##                   NA's   :58.00
##  avg_days_susp    mou_onnet_6m_normal unsolv_tsupcomplnt   wrk_orders
##  Min.   : 0.000   Min.   :-27.1355    Min.   :0.0000     Min.   :0.000
##  1st Qu.: 0.000   1st Qu.: -0.6147    1st Qu.:0.0000     1st Qu.:0.000
##  Median : 2.000   Median :  0.0000    Median :0.0000     Median :0.000
##  Mean   : 3.474   Mean   : -0.1175    Mean   :0.6858     Mean   :0.112
##  3rd Qu.: 6.000   3rd Qu.:  0.0000    3rd Qu.:1.0000     3rd Qu.:0.000
##  Max.   :62.000   Max.   : 72.0113    Max.   :5.0000     Max.   :6.000
##
##  days_openwrkorders
##  Min.   :  0.000
##  1st Qu.:  0.000
##  Median :  0.000
```

```
##  Mean   :  5.332
##  3rd Qu.:  5.000
##  Max.   : 99.000
##  NA's   :155.000
```

```r
# Check for any missingness in the data

dist_tabl = cas.simple.distinct(data)$Distinct[,c('Column','NMiss')]
print(dist_tabl)
```

```
##                  Column NMiss
## 1        lifetime_value     0
## 2        calls_in_offpk     0
## 3     mou_onnet_pct_MOM     0
## 4       mb_data_usg_m01     0
## 5       mb_data_usg_m02     0
## 6       mb_data_usg_m03     0
## 7                region     0
## 8          upsell_xsell     0
## 9    ever_days_over_plan    58
## 10 ever_times_over_plan     0
## 11         avg_days_susp     0
## 12   mou_onnet_6m_normal     0
## 13    unsolv_tsupcomplnt     0
## 14            wrk_orders     0
## 15    days_openwrkorders   155
```

```r
dist_tabl = as.data.frame(dist_tabl)
sub = subset(dist_tabl, dist_tabl$NMiss != 0)
imp_cols = sub$Column

# Print the names of the columns to be imputed
print(imp_cols)
```

```
## [1] "ever_days_over_plan" "days_openwrkorders"
```

```r
# Impute the missing values

cas.dataPreprocess.impute(data,
                          methodContinuous = 'MEDIAN',
                          methodNominal    = 'MODE',
                          inputs           = imp_cols,
                          copyAllVars      = TRUE,
                          casOut           = list(name = 'castbl', replace = TRUE)
                         )
```

```
## $ImputeInfo
##              Variable ImputeTech               ResultVar     N NMiss
## 1 ever_days_over_plan     Median IMP_ever_days_over_plan 56498    58
## 2  days_openwrkorders     Median  IMP_days_openwrkorders 56401   155
##   ImputedValueContinuous
## 1                      9
```

```
## 2                        0
##
## $OutputCasTables
##   casLib   Name  Rows Columns
## 1     lg castbl 56556      17
```

```r
# Split the data into training and validation and view the partitioned table

loadActionSet(s,"sampling")
```

```
## NOTE: Added action set 'sampling'.

## NOTE: Information for action set 'sampling':

## NOTE:    sampling

## NOTE:       srs -  Samples a proportion of data from the input table or partitions the data into no m

## NOTE:       stratified - Samples a proportion of data or partitions the data into no more than three

## NOTE:       oversample - Samples a user-specified proportion of data from the event level and adjusts

## NOTE:       kfold - K-fold partitioning.
```

```r
cas.sampling.srs( s,
                  table   = list(name="castbl", caslib="lg"),
                  samppct = 30,
                  seed = 123456,
                  partind = TRUE,
                  output  = list(casOut = list(name = "sampled_castbl", replace = T, caslib="lg"), copyV
                )
```

```
## NOTE: Using SEED=123456 for sampling.

## $OutputCasTables
##   casLib           Name Label  Rows Columns
## 1     lg sampled_castbl       56556      18
##
## $SRSFreq
##    NObs NSamp
## 1 56556 16967
##
## $outputSize
## $outputSize$outputNObs
## [1] 56556
##
## $outputSize$outputNVars
## [1] 18
```

```r
# Check for frequency distribution of partitioned data

cas.simple.freq(s,table="sampled_castbl", inputs="_PartInd_")
```

```
## $Frequency
##      Column NumVar       FmtVar Level Frequency
## 1 _PartInd_      0            0     1     39589
## 2 _PartInd_      1            1     2     16967
```

```r
# Partition data into train and validation based on _PartInd_

train = defCasTable(s, tablename = "sampled_castbl", where = " _PartInd_ = 0 ")

val   = defCasTable(s, tablename = "sampled_castbl", where = " _PartInd_ = 1 ")
```

```r
# Create the appropriate input and target variables

info = cas.table.columnInfo(s, table = train)

colinfo = info$ColumnInfo

## nominal variables are: region, upsell_xsell

nominals = colinfo$Column[c(7,8)]

intervals = colinfo$Column[c(-7,-8,-9,-15,-18)]

target = colinfo$Column[8]

inputs = colinfo$Column[c(-8,-9,-15,-18)]
```

```r
# Build a GB model for predictive classification

loadActionSet(s, "decisionTree")
```

```
## NOTE: Added action set 'decisionTree'.

## NOTE: Information for action set 'decisionTree':

## NOTE:    decisionTree

## NOTE:       dtreeTrain - Trains a decision tree

## NOTE:       dtreeScore - Scores a table using a decision tree model

## NOTE:       dtreeSplit - Splits decision tree nodes

## NOTE:       dtreePrune - Prune a decision tree

## NOTE:       dtreeMerge - Merges decision tree nodes
```

```
## NOTE:       dtreeCode - Generates DATA step scoring code from a decision tree model

## NOTE:       forestTrain - Trains a forest

## NOTE:       forestScore - Scores a table using a forest model

## NOTE:       forestCode - Generates DATA step scoring code from a forest model

## NOTE:       gbtreeTrain - Trains a gradient boosting tree

## NOTE:       gbtreeScore - Scores a table using a gradient boosting tree model

## NOTE:       gbtreeCode - Generates DATA step scoring code from a gradient boosting tree model
```

```
model = cas.decisionTree.gbtreeTrain(
                                      s,
                                      casOut=list(caslib="lg",name="gb_model",replace=T),
                                      inputs = inputs,
                                      nominals = nominals,
                                      target = target,
                                      table = train
                                    )

# View the model info

print(model)
```

```
## $ModelInfo
##                                  Descr   Value
## 1                      Number of Trees    50.0
## 2                         Distribution     2.0
## 3                        Learning Rate     0.1
## 4                     Subsampling Rate     0.5
## 5  Number of Selected Variables (M)    14.0
## 6                       Number of Bins    20.0
## 7                  Number of Variables    14.0
## 8             Max Number of Tree Nodes    63.0
## 9             Min Number of Tree Nodes    27.0
## 10             Max Number of Branches     2.0
## 11             Min Number of Branches     2.0
## 12               Max Number of Levels     6.0
## 13               Min Number of Levels     6.0
## 14               Max Number of Leaves    32.0
## 15               Min Number of Leaves    14.0
## 16             Maximum Size of Leaves 19533.0
## 17             Minimum Size of Leaves     5.0
## 18                 Random Number Seed     0.0
##
## $OutputCasTables
##   casLib     Name Rows Columns
## 1     lg gb_model 2716      35
```

```r
# Score the model on test data

out = cas.decisionTree.gbtreeScore (
                                    s,
                                    modelTable = list(name="gb_model", caslib="lg"),
                                    table = val,
                                    encodeName = TRUE,
                                    assessonerow = TRUE,
                                    casOut = list(name="scored_data", caslib="lg", replace=T),
                                    copyVars = target
                                  )

# View the scored results

cas.table.fetch(s,table="scored_data")
```

```
## $Fetch
##    _Index_ upsell_xsell I_upsell_xsell _MissIt_ P_upsell_xsell1
## 1        1            0              0        0      0.05252862
## 2        2            0              0        0      0.15759640
## 3        3            0              0        0      0.11971763
## 4        4            0              0        0      0.03808747
## 5        5            0              0        0      0.11915922
## 6        6            0              0        0      0.04344497
## 7        7            0              0        0      0.05660835
## 8        8            0              0        0      0.06187560
## 9        9            0              0        0      0.05995694
## 10      10            0              0        0      0.04539507
## 11      11            1              0        1      0.44533606
## 12      12            0              0        0      0.05862872
## 13      13            0              0        0      0.05363998
## 14      14            0              0        0      0.06669830
## 15      15            0              0        0      0.25437690
## 16      16            1              0        1      0.23457944
## 17      17            0              1        1      0.64007086
## 18      18            0              0        0      0.04863400
## 19      19            0              0        0      0.05738928
## 20      20            0              0        0      0.04363434
##    P_upsell_xsell0
## 1        0.9474714
## 2        0.8424036
## 3        0.8802824
## 4        0.9619125
## 5        0.8808408
## 6        0.9565550
## 7        0.9433917
## 8        0.9381244
## 9        0.9400431
## 10       0.9546049
## 11       0.5546639
## 12       0.9413713
## 13       0.9463600
## 14       0.9333017
```

```
## 15       0.7456231
## 16       0.7654206
## 17       0.3599291
## 18       0.9513660
## 19       0.9426107
## 20       0.9563657
```

```r
# Train an R Extreme Gradient Boosting model

# First, convert the train and test CAS tables to R data frames for training the R-XGB model
train_cas_df = to.casDataFrame(train)
train_df = to.data.frame(train_cas_df)

val_cas_df = to.casDataFrame(val)
val_df = to.data.frame(val_cas_df)

# In R, we need to do the data pre-processing explicitly. Hence, convert the "char" region variable to
train_df$region = as.factor(train_df$region)
val_df$region = as.factor(val_df$region)

# For XGB model, it requires the input to be numeric. Hence, convert cateogrical variables into numeric
train_dmy = dummyVars(" ~ .", data = train_df,fullRank = T)
val_dmy = dummyVars(" ~ .", data = val_df,fullRank = T)

prep_train = data.frame(predict(train_dmy, newdata = train_df))
prep_val = data.frame(predict(val_dmy, newdata = val_df))

print(head(prep_train))
```

```
##   lifetime_value calls_in_offpk mou_onnet_pct_MOM mb_data_usg_m01
## 1         9616.9         604.38                 0        1388.947
## 2         7619.3         793.57                 0        2930.470
## 3         2765.7         529.50                 0          69.000
## 4         6426.5         333.39                 1        1739.512
## 5         5372.8         -16.42                 0        1075.152
## 6         1746.9         364.10                 0        1191.598
##   mb_data_usg_m02 mb_data_usg_m03 region.Greater.Texas region.Mid.Atlantic
## 1        1243.291        1299.693                    0                   0
## 2        2856.150        3030.931                    0                   0
## 3         431.056         412.150                    0                   1
## 4        1766.006        1702.673                    0                   0
## 5         854.023         829.591                    0                   0
## 6        1222.585        1254.263                    0                   0
##   region.Midwest region.Mtn.West region.New.England region.Pacific
## 1              0               0                  0              1
## 2              0               0                  0              0
## 3              0               0                  0              0
## 4              1               0                  0              0
## 5              0               0                  0              0
## 6              0               0                  0              1
##   region.South region.Southwest upsell_xsell ever_days_over_plan
## 1            0                0            0                   2
## 2            0                1            0                  10
## 3            0                0            0                   9
```

```
## 4           0                0            0                   0
## 5           1                0            0                  11
## 6           0                0            0                  14
##   ever_times_over_plan avg_days_susp mou_onnet_6m_normal
## 1                    6             6                   0
## 2                    1             5                   0
## 3                    2             0                  -3
## 4                    2             4                  -2
## 5                    5             2                   1
## 6                    3            12                   0
##   unsolv_tsupcomplnt wrk_orders days_openwrkorders IMP_days_openwrkorders
## 1                  0          0                 15                     15
## 2                  0          0                  0                      0
## 3                  0          0                 11                     11
## 4                  0          0                  0                      0
## 5                  1          0                 16                     16
## 6                  0          0                  6                      6
##   IMP_ever_days_over_plan X._PartInd_.
## 1                       2            0
## 2                      10            0
## 3                       9            0
## 4                       0            0
## 5                      11            0
## 6                      14            0
```

```
print(head(prep_val))
```

```
##   lifetime_value calls_in_offpk mou_onnet_pct_MOM mb_data_usg_m01
## 1         9165.1        1320.14                 0        6813.458
## 2         1892.9           8.70                 5        1584.943
## 3         9672.0         192.61                 0        2924.855
## 4         5704.2         120.76                 0        1353.099
## 5         5472.7         389.50                 0        1864.386
## 6         6576.7         259.17                -1        1396.245
##   mb_data_usg_m02 mb_data_usg_m03 region.Greater.Texas region.Mid.Atlantic
## 1        6826.472        6992.660                    0                   0
## 2        1695.293        1581.966                    0                   0
## 3        2821.905        2764.459                    0                   0
## 4        1308.704        1413.062                    0                   1
## 5        1799.559        1947.918                    0                   0
## 6        1273.867        1536.013                    0                   0
##   region.Midwest region.Mtn.West region.New.England region.Pacific
## 1              0               0                  0              1
## 2              0               0                  0              0
## 3              0               0                  0              0
## 4              0               0                  0              0
## 5              0               0                  0              1
## 6              0               0                  0              0
##   region.South region.Southwest upsell_xsell ever_days_over_plan
## 1            0                0            0                   0
## 2            1                0            0                  26
## 3            1                0            0                  15
## 4            0                0            0                  22
## 5            0                0            0                   1
```

```
## 6           1                0            0                    31
##   ever_times_over_plan avg_days_susp mou_onnet_6m_normal
## 1                    7             1                   0
## 2                    0             3                  -8
## 3                    6             4                   0
## 4                    0             3                   0
## 5                    7            12                  -1
## 6                    0             0                   0
##   unsolv_tsupcomplnt wrk_orders days_openwrkorders IMP_days_openwrkorders
## 1                  3          0                  0                      0
## 2                  1          0                  3                      3
## 3                  1          0                  0                      0
## 4                  2          0                 18                     18
## 5                  1          0                 11                     11
## 6                  2          0                  0                      0
##   IMP_ever_days_over_plan X._PartInd_.
## 1                       0            1
## 2                      26            1
## 3                      15            1
## 4                      22            1
## 5                       1            1
## 6                      31            1
```

```
# Convert target variable to numeric since XGB expects numeric values. Also, create a new target labels

train_labels = as.numeric(as.factor(prep_train$upsell_xsell)) - 1
test_labels = as.numeric(as.factor(prep_val$upsell_xsell)) - 1

# Remove target and missing value columns from the dataset

prep_train$upsell_xsell = NULL
prep_val$upsell_xsell = NULL

prep_train$days_openwrkorders = NULL
prep_train$ever_days_over_plan = NULL

prep_val$days_openwrkorders = NULL
prep_val$ever_days_over_plan = NULL
```

```
# Train a XGBoost model on the data

xgb = xgboost(data = data.matrix(prep_train[,-1]),
               label = train_labels,
               nround=2,
               objective = "binary:logistic"
               )
```

```
## [1]  train-error:0.095459
## [2]  train-error:0.093536
```

```
# Make predictions on test data

pred = predict(xgb, newdata= data.matrix(prep_val[,-1]))
```

```r
# Evaluate the performance of SAS and R models

## Assessing the performance metric of SAS-GB model

loadActionSet(s,"percentile")
```

```
## NOTE: Added action set 'percentile'.

## NOTE: Information for action set 'percentile':

## NOTE:    percentile

## NOTE:       percentile - Calculate quantiles and percentiles

## NOTE:       boxPlot - Calculate quantiles, high and low whiskers, and outliers

## NOTE:       assess - Assess and compare models
```

```r
tmp = cas.percentile.assess(
                             s,
                             cutStep = 0.05,
                             event = "1",
                             inputs = "P_upsell_xsell1",
                             nBins = 20,
                             response = target,
                             table = "scored_data"

                            )$ROCInfo

roc_df = data.frame(tmp)
print(head(roc_df))
```

```
##          Variable Event CutOff   TP    FP   FN    TN Sensitivity
## 1 P_upsell_xsell1     1   0.00 2010 14957    0     0   1.0000000
## 2 P_upsell_xsell1     1   0.05 1794 11151  216  3806   0.8925373
## 3 P_upsell_xsell1     1   0.10 1258  2961  752 11996   0.6258706
## 4 P_upsell_xsell1     1   0.15 1097  1505  913 13452   0.5457711
## 5 P_upsell_xsell1     1   0.20 1011   987  999 13970   0.5029851
## 6 P_upsell_xsell1     1   0.25  920   697 1090 14260   0.4577114
##   Specificity KS       KS2     F_HALF        FPR       ACC       FDR
## 1   0.0000000  0 0.0000000 0.1438221 1.00000000 0.1184653 0.8815347
## 2   0.2544628  0 0.1470001 0.1667596 0.74553721 0.3300525 0.8614137
## 3   0.8020325  0 0.4279031 0.3330509 0.19796751 0.7811634 0.7018251
## 4   0.8993782  1 0.4451494 0.4416975 0.10062178 0.8574881 0.5784012
## 5   0.9340108  0 0.4369959 0.5053989 0.06598917 0.8829493 0.4939940
## 6   0.9533997  0 0.4111112 0.5425808 0.04660025 0.8946779 0.4310451
##          F1         C      Gini     Gamma       Tau MISCEVENT
## 1 0.2118354 0.7577628 0.5155257 0.6323749 0.1076803 0.8815347
## 2 0.2399198 0.7577628 0.5155257 0.6323749 0.1076803 0.6699475
## 3 0.4039172 0.7577628 0.5155257 0.6323749 0.1076803 0.2188366
## 4 0.4757155 0.7577628 0.5155257 0.6323749 0.1076803 0.1425119
## 5 0.5044910 0.7577628 0.5155257 0.6323749 0.1076803 0.1170507
## 6 0.5073063 0.7577628 0.5155257 0.6323749 0.1076803 0.1053221
```

```
# Display the confusion matrix for cutoff threshold at 0.5

cutoff = subset(roc_df, CutOff == 0.5)

tn = cutoff$TN
fn = cutoff$FN
tp = cutoff$TP
fp = cutoff$FP
a = c(tn,fn)
p = c(fp,tp)
mat = data.frame(a,p)
colnames(mat) = c("Pred:0","Pred:1")
rownames(mat) = c("Actual:0","Actual:1")
mat = as.matrix(mat)
print(mat)
```

```
##          Pred:0 Pred:1
## Actual:0  14753    204
## Actual:1   1350    660
```

```
# Print the accuracy and misclassification rates for the model

accuracy = cutoff$ACC
mis = cutoff$MISCEVENT

print(paste("Misclassification rate is",mis))
```

```
## [1] "Misclassification rate is 0.09158955619732"
```

```
print(paste("Accuracy is",accuracy))
```

```
## [1] "Accuracy is 0.90841044380267"
```

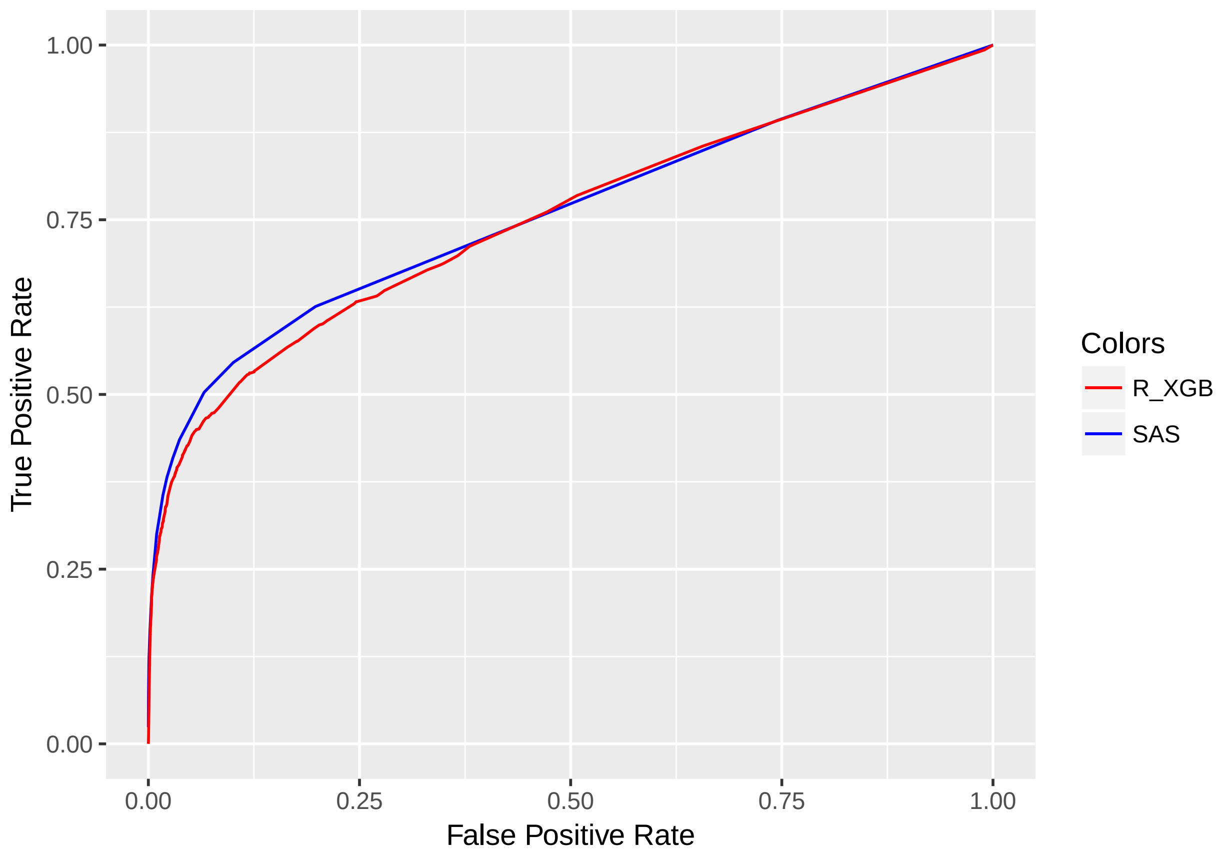
```
## Assessing the performance metric of R-XGB model

# Create a confusion matrix for cutoff threshold at 0.5

conf.matrix = table(test_labels, as.numeric(pred>0.5))
rownames(conf.matrix) = paste("Actual", rownames(conf.matrix), sep = ":")
colnames(conf.matrix) = paste("Pred", colnames(conf.matrix), sep = ":")

# Print the accuracy and misclassification rates for the model

err = mean(as.numeric(pred > 0.5) != test_labels)

print(paste("Misclassification rate is",err))
```

```
## [1] "Misclassification rate is 0.0954794601284847"
```

```r
print(paste("Accuracy is",1-err))
```

```
## [1] "Accuracy is 0.904520539871515"
```

```r
# Plot ROC curves for both the models using standard R plotting functions

FPR_SAS = roc_df['FPR']
TPR_SAS = roc_df['Sensitivity']

pred1 = prediction(pred, test_labels)
perf1 = performance( pred1, "tpr", "fpr" )

FPR_R = perf1@x.values[[1]]
TPR_R = perf1@y.values[[1]]

roc_df2 = data.frame(FPR = FPR_R, TPR = TPR_R)

ggplot() +

geom_line(
           data = roc_df[c('FPR', 'Sensitivity')],
           aes(x = as.numeric(FPR), y = as.numeric(Sensitivity),color = "SAS"),
          ) +

geom_line(
           data = roc_df2,
           aes(x = as.numeric(FPR_R), y = as.numeric(TPR_R),color = "R_XGB"),
         ) +

scale_color_manual(
                    name = "Colors",
                    values = c("SAS" = "blue", "R_XGB" = "red")
                   ) +

xlab('False Positive Rate') + ylab('True Positive Rate')
```




```
# Terminate the CAS session
cas.session.endSession(s)
```

```
## list()
```