xgboost with crossvalidation in R

Arpan

January 4, 2017

Setting the seed so that we get the same results each time we run the model

```
set.seed(123)
```

Importing the library mlbench for sonar dataset

```
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 3.2.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.2.5
```

Storing the data set named "Sonar" into DataFrame named "DataFrame"

```
data("Sonar")
DataFrame <- Sonar
```

Type help("Sonar") to know about the data set

```
help("Sonar")
```

```
## starting httpd help server ...

##  done
```

Check the dimension of this data frame

```
dim(DataFrame)
```

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

Check first 3 rows

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

```
##       V1     V2     V3     V4     V5     V6     V7     V8     V9    V10
## 1 0.0200 0.0371 0.0428 0.0207 0.0954 0.0986 0.1539 0.1601 0.3109 0.2111
## 2 0.0453 0.0523 0.0843 0.0689 0.1183 0.2583 0.2156 0.3481 0.3337 0.2872
## 3 0.0262 0.0582 0.1099 0.1083 0.0974 0.2280 0.2431 0.3771 0.5598 0.6194
##      V11    V12    V13    V14    V15    V16    V17    V18    V19    V20
## 1 0.1609 0.1582 0.2238 0.0645 0.0660 0.2273 0.3100 0.2999 0.5078 0.4797
## 2 0.4918 0.6552 0.6919 0.7797 0.7464 0.9444 1.0000 0.8874 0.8024 0.7818
## 3 0.6333 0.7060 0.5544 0.5320 0.6479 0.6931 0.6759 0.7551 0.8929 0.8619
##      V21    V22    V23    V24    V25    V26    V27    V28    V29    V30
## 1 0.5783 0.5071 0.4328 0.5550 0.6711 0.6415 0.7104 0.8080 0.6791 0.3857
## 2 0.5212 0.4052 0.3957 0.3914 0.3250 0.3200 0.3271 0.2767 0.4423 0.2028
## 3 0.7974 0.6737 0.4293 0.3648 0.5331 0.2413 0.5070 0.8533 0.6036 0.8514
##      V31    V32    V33    V34    V35    V36    V37    V38    V39    V40
## 1 0.1307 0.2604 0.5121 0.7547 0.8537 0.8507 0.6692 0.6097 0.4943 0.2744
## 2 0.3788 0.2947 0.1984 0.2341 0.1306 0.4182 0.3835 0.1057 0.1840 0.1970
## 3 0.8512 0.5045 0.1862 0.2709 0.4232 0.3043 0.6116 0.6756 0.5375 0.4719
##      V41    V42    V43    V44    V45    V46    V47    V48    V49    V50
## 1 0.0510 0.2834 0.2825 0.4256 0.2641 0.1386 0.1051 0.1343 0.0383 0.0324
## 2 0.1674 0.0583 0.1401 0.1628 0.0621 0.0203 0.0530 0.0742 0.0409 0.0061
## 3 0.4647 0.2587 0.2129 0.2222 0.2111 0.0176 0.1348 0.0744 0.0130 0.0106
##      V51    V52    V53    V54    V55    V56    V57    V58    V59    V60
## 1 0.0232 0.0027 0.0065 0.0159 0.0072 0.0167 0.0180 0.0084 0.0090 0.0032
## 2 0.0125 0.0084 0.0089 0.0048 0.0094 0.0191 0.0140 0.0049 0.0052 0.0044
## 3 0.0033 0.0232 0.0166 0.0095 0.0180 0.0244 0.0316 0.0164 0.0095 0.0078
##   Class
## 1     R
## 2     R
## 3     R
```

Check the summary of data

```r
summary(DataFrame)
```

```
##        V1                V2                V3                V4
##  Min.   :0.00150   Min.   :0.00060   Min.   :0.00150   Min.   :0.00580
##  1st Qu.:0.01335   1st Qu.:0.01645   1st Qu.:0.01895   1st Qu.:0.02438
##  Median :0.02280   Median :0.03080   Median :0.03430   Median :0.04405
##  Mean   :0.02916   Mean   :0.03844   Mean   :0.04383   Mean   :0.05389
##  3rd Qu.:0.03555   3rd Qu.:0.04795   3rd Qu.:0.05795   3rd Qu.:0.06450
##  Max.   :0.13710   Max.   :0.23390   Max.   :0.30590   Max.   :0.42640
##        V5                V6                V7                V8
##  Min.   :0.00670   Min.   :0.01020   Min.   :0.0033   Min.   :0.00550
##  1st Qu.:0.03805   1st Qu.:0.06703   1st Qu.:0.0809   1st Qu.:0.08042
##  Median :0.06250   Median :0.09215   Median :0.1070   Median :0.11210
##  Mean   :0.07520   Mean   :0.10457   Mean   :0.1217   Mean   :0.13480
##  3rd Qu.:0.10028   3rd Qu.:0.13412   3rd Qu.:0.1540   3rd Qu.:0.16960
##  Max.   :0.40100   Max.   :0.38230   Max.   :0.3729   Max.   :0.45900
##        V9               V10               V11               V12
##  Min.   :0.00750   Min.   :0.0113   Min.   :0.0289   Min.   :0.0236
##  1st Qu.:0.09703   1st Qu.:0.1113   1st Qu.:0.1293   1st Qu.:0.1335
```

```
##  Median :0.15225   Median :0.1824   Median :0.2248   Median :0.2490
##  Mean   :0.17800   Mean   :0.2083   Mean   :0.2360   Mean   :0.2502
##  3rd Qu.:0.23342   3rd Qu.:0.2687   3rd Qu.:0.3016   3rd Qu.:0.3312
##  Max.   :0.68280   Max.   :0.7106   Max.   :0.7342   Max.   :0.7060
##       V13               V14               V15               V16
##  Min.   :0.0184   Min.   :0.0273   Min.   :0.0031   Min.   :0.0162
##  1st Qu.:0.1661   1st Qu.:0.1752   1st Qu.:0.1646   1st Qu.:0.1963
##  Median :0.2640   Median :0.2811   Median :0.2817   Median :0.3047
##  Mean   :0.2733   Mean   :0.2966   Mean   :0.3202   Mean   :0.3785
##  3rd Qu.:0.3513   3rd Qu.:0.3862   3rd Qu.:0.4529   3rd Qu.:0.5357
##  Max.   :0.7131   Max.   :0.9970   Max.   :1.0000   Max.   :0.9988
##       V17               V18               V19               V20
##  Min.   :0.0349   Min.   :0.0375   Min.   :0.0494   Min.   :0.0656
##  1st Qu.:0.2059   1st Qu.:0.2421   1st Qu.:0.2991   1st Qu.:0.3506
##  Median :0.3084   Median :0.3683   Median :0.4350   Median :0.5425
##  Mean   :0.4160   Mean   :0.4523   Mean   :0.5048   Mean   :0.5630
##  3rd Qu.:0.6594   3rd Qu.:0.6791   3rd Qu.:0.7314   3rd Qu.:0.8093
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##       V21               V22               V23               V24
##  Min.   :0.0512   Min.   :0.0219   Min.   :0.0563   Min.   :0.0239
##  1st Qu.:0.3997   1st Qu.:0.4069   1st Qu.:0.4502   1st Qu.:0.5407
##  Median :0.6177   Median :0.6649   Median :0.6997   Median :0.6985
##  Mean   :0.6091   Mean   :0.6243   Mean   :0.6470   Mean   :0.6727
##  3rd Qu.:0.8170   3rd Qu.:0.8320   3rd Qu.:0.8486   3rd Qu.:0.8722
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##       V25               V26               V27               V28
##  Min.   :0.0240   Min.   :0.0921   Min.   :0.0481   Min.   :0.0284
##  1st Qu.:0.5258   1st Qu.:0.5442   1st Qu.:0.5319   1st Qu.:0.5348
##  Median :0.7211   Median :0.7545   Median :0.7456   Median :0.7319
##  Mean   :0.6754   Mean   :0.6999   Mean   :0.7022   Mean   :0.6940
##  3rd Qu.:0.8737   3rd Qu.:0.8938   3rd Qu.:0.9171   3rd Qu.:0.9003
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##       V29               V30               V31               V32
##  Min.   :0.0144   Min.   :0.0613   Min.   :0.0482   Min.   :0.0404
##  1st Qu.:0.4637   1st Qu.:0.4114   1st Qu.:0.3456   1st Qu.:0.2814
##  Median :0.6808   Median :0.6071   Median :0.4904   Median :0.4296
##  Mean   :0.6421   Mean   :0.5809   Mean   :0.5045   Mean   :0.4390
##  3rd Qu.:0.8521   3rd Qu.:0.7352   3rd Qu.:0.6420   3rd Qu.:0.5803
##  Max.   :1.0000   Max.   :1.0000   Max.   :0.9657   Max.   :0.9306
##       V33               V34               V35               V36
##  Min.   :0.0477   Min.   :0.0212   Min.   :0.0223   Min.   :0.0080
##  1st Qu.:0.2579   1st Qu.:0.2176   1st Qu.:0.1794   1st Qu.:0.1543
##  Median :0.3912   Median :0.3510   Median :0.3127   Median :0.3211
##  Mean   :0.4172   Mean   :0.4032   Mean   :0.3926   Mean   :0.3848
##  3rd Qu.:0.5561   3rd Qu.:0.5961   3rd Qu.:0.5934   3rd Qu.:0.5565
##  Max.   :1.0000   Max.   :0.9647   Max.   :1.0000   Max.   :1.0000
##       V37               V38               V39               V40
##  Min.   :0.0351   Min.   :0.0383   Min.   :0.0371   Min.   :0.0117
##  1st Qu.:0.1601   1st Qu.:0.1743   1st Qu.:0.1740   1st Qu.:0.1865
##  Median :0.3063   Median :0.3127   Median :0.2835   Median :0.2781
##  Mean   :0.3638   Mean   :0.3397   Mean   :0.3258   Mean   :0.3112
##  3rd Qu.:0.5189   3rd Qu.:0.4405   3rd Qu.:0.4349   3rd Qu.:0.4244
##  Max.   :0.9497   Max.   :1.0000   Max.   :0.9857   Max.   :0.9297
##       V41               V42               V43               V44
```

```
##  Min.   :0.0360   Min.   :0.0056   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.1631   1st Qu.:0.1589   1st Qu.:0.1552   1st Qu.:0.1269
##  Median :0.2595   Median :0.2451   Median :0.2225   Median :0.1777
##  Mean   :0.2893   Mean   :0.2783   Mean   :0.2465   Mean   :0.2141
##  3rd Qu.:0.3875   3rd Qu.:0.3842   3rd Qu.:0.3245   3rd Qu.:0.2717
##  Max.   :0.8995   Max.   :0.8246   Max.   :0.7733   Max.   :0.7762
##       V45               V46               V47               V48
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.09448   1st Qu.:0.06855   1st Qu.:0.06425   1st Qu.:0.04512
##  Median :0.14800   Median :0.12135   Median :0.10165   Median :0.07810
##  Mean   :0.19723   Mean   :0.16063   Mean   :0.12245   Mean   :0.09142
##  3rd Qu.:0.23155   3rd Qu.:0.20037   3rd Qu.:0.15443   3rd Qu.:0.12010
##  Max.   :0.70340   Max.   :0.72920   Max.   :0.55220   Max.   :0.33390
##       V49               V50               V51               V52
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.000000   Min.   :0.000800
##  1st Qu.:0.02635   1st Qu.:0.01155   1st Qu.:0.008425   1st Qu.:0.007275
##  Median :0.04470   Median :0.01790   Median :0.013900   Median :0.011400
##  Mean   :0.05193   Mean   :0.02042   Mean   :0.016069   Mean   :0.013420
##  3rd Qu.:0.06853   3rd Qu.:0.02527   3rd Qu.:0.020825   3rd Qu.:0.016725
##  Max.   :0.19810   Max.   :0.08250   Max.   :0.100400   Max.   :0.070900
##       V53               V54               V55
##  Min.   :0.000500   Min.   :0.001000   Min.   :0.00060
##  1st Qu.:0.005075   1st Qu.:0.005375   1st Qu.:0.00415
##  Median :0.009550   Median :0.009300   Median :0.00750
##  Mean   :0.010709   Mean   :0.010941   Mean   :0.00929
##  3rd Qu.:0.014900   3rd Qu.:0.014500   3rd Qu.:0.01210
##  Max.   :0.039000   Max.   :0.035200   Max.   :0.04470
##       V56               V57               V58
##  Min.   :0.000400   Min.   :0.00030   Min.   :0.000300
##  1st Qu.:0.004400   1st Qu.:0.00370   1st Qu.:0.003600
##  Median :0.006850   Median :0.00595   Median :0.005800
##  Mean   :0.008222   Mean   :0.00782   Mean   :0.007949
##  3rd Qu.:0.010575   3rd Qu.:0.01043   3rd Qu.:0.010350
##  Max.   :0.039400   Max.   :0.03550   Max.   :0.044000
##       V59               V60            Class
##  Min.   :0.000100   Min.   :0.000600   M:111
##  1st Qu.:0.003675   1st Qu.:0.003100   R: 97
##  Median :0.006400   Median :0.005300
##  Mean   :0.007941   Mean   :0.006507
##  3rd Qu.:0.010325   3rd Qu.:0.008525
##  Max.   :0.036400   Max.   :0.043900
```

Lets check the data set again

```
str(DataFrame)
```

```
## 'data.frame':    208 obs. of  61 variables:
##  $ V1   : num  0.02 0.0453 0.0262 0.01 0.0762 0.0286 0.0317 0.0519 0.0223 0.0164 ...
##  $ V2   : num  0.0371 0.0523 0.0582 0.0171 0.0666 0.0453 0.0956 0.0548 0.0375 0.0173 ...
##  $ V3   : num  0.0428 0.0843 0.1099 0.0623 0.0481 ...
##  $ V4   : num  0.0207 0.0689 0.1083 0.0205 0.0394 ...
##  $ V5   : num  0.0954 0.1183 0.0974 0.0205 0.059 ...
##  $ V6   : num  0.0986 0.2583 0.228 0.0368 0.0649 ...
```

```
##  $ V7   : num  0.154 0.216 0.243 0.11 0.121 ...
##  $ V8   : num  0.16 0.348 0.377 0.128 0.247 ...
##  $ V9   : num  0.3109 0.3337 0.5598 0.0598 0.3564 ...
##  $ V10  : num  0.211 0.287 0.619 0.126 0.446 ...
##  $ V11  : num  0.1609 0.4918 0.6333 0.0881 0.4152 ...
##  $ V12  : num  0.158 0.655 0.706 0.199 0.395 ...
##  $ V13  : num  0.2238 0.6919 0.5544 0.0184 0.4256 ...
##  $ V14  : num  0.0645 0.7797 0.532 0.2261 0.4135 ...
##  $ V15  : num  0.066 0.746 0.648 0.173 0.453 ...
##  $ V16  : num  0.227 0.944 0.693 0.213 0.533 ...
##  $ V17  : num  0.31 1 0.6759 0.0693 0.7306 ...
##  $ V18  : num  0.3 0.887 0.755 0.228 0.619 ...
##  $ V19  : num  0.508 0.802 0.893 0.406 0.203 ...
##  $ V20  : num  0.48 0.782 0.862 0.397 0.464 ...
##  $ V21  : num  0.578 0.521 0.797 0.274 0.415 ...
##  $ V22  : num  0.507 0.405 0.674 0.369 0.429 ...
##  $ V23  : num  0.433 0.396 0.429 0.556 0.573 ...
##  $ V24  : num  0.555 0.391 0.365 0.485 0.54 ...
##  $ V25  : num  0.671 0.325 0.533 0.314 0.316 ...
##  $ V26  : num  0.641 0.32 0.241 0.533 0.229 ...
##  $ V27  : num  0.71 0.327 0.507 0.526 0.7 ...
##  $ V28  : num  0.808 0.277 0.853 0.252 1 ...
##  $ V29  : num  0.679 0.442 0.604 0.209 0.726 ...
##  $ V30  : num  0.386 0.203 0.851 0.356 0.472 ...
##  $ V31  : num  0.131 0.379 0.851 0.626 0.51 ...
##  $ V32  : num  0.26 0.295 0.504 0.734 0.546 ...
##  $ V33  : num  0.512 0.198 0.186 0.612 0.288 ...
##  $ V34  : num  0.7547 0.2341 0.2709 0.3497 0.0981 ...
##  $ V35  : num  0.854 0.131 0.423 0.395 0.195 ...
##  $ V36  : num  0.851 0.418 0.304 0.301 0.418 ...
##  $ V37  : num  0.669 0.384 0.612 0.541 0.46 ...
##  $ V38  : num  0.61 0.106 0.676 0.881 0.322 ...
##  $ V39  : num  0.494 0.184 0.537 0.986 0.283 ...
##  $ V40  : num  0.274 0.197 0.472 0.917 0.243 ...
##  $ V41  : num  0.051 0.167 0.465 0.612 0.198 ...
##  $ V42  : num  0.2834 0.0583 0.2587 0.5006 0.2444 ...
##  $ V43  : num  0.282 0.14 0.213 0.321 0.185 ...
##  $ V44  : num  0.4256 0.1628 0.2222 0.3202 0.0841 ...
##  $ V45  : num  0.2641 0.0621 0.2111 0.4295 0.0692 ...
##  $ V46  : num  0.1386 0.0203 0.0176 0.3654 0.0528 ...
##  $ V47  : num  0.1051 0.053 0.1348 0.2655 0.0357 ...
##  $ V48  : num  0.1343 0.0742 0.0744 0.1576 0.0085 ...
##  $ V49  : num  0.0383 0.0409 0.013 0.0681 0.023 0.0264 0.0507 0.0285 0.0777 0.0092 ...
##  $ V50  : num  0.0324 0.0061 0.0106 0.0294 0.0046 0.0081 0.0159 0.0178 0.0439 0.0198 ...
##  $ V51  : num  0.0232 0.0125 0.0033 0.0241 0.0156 0.0104 0.0195 0.0052 0.0061 0.0118 ...
##  $ V52  : num  0.0027 0.0084 0.0232 0.0121 0.0031 0.0045 0.0201 0.0081 0.0145 0.009 ...
##  $ V53  : num  0.0065 0.0089 0.0166 0.0036 0.0054 0.0014 0.0248 0.012 0.0128 0.0223 ...
##  $ V54  : num  0.0159 0.0048 0.0095 0.015 0.0105 0.0038 0.0131 0.0045 0.0145 0.0179 ...
##  $ V55  : num  0.0072 0.0094 0.018 0.0085 0.011 0.0013 0.007 0.0121 0.0058 0.0084 ...
##  $ V56  : num  0.0167 0.0191 0.0244 0.0073 0.0015 0.0089 0.0138 0.0097 0.0049 0.0068 ...
##  $ V57  : num  0.018 0.014 0.0316 0.005 0.0072 0.0057 0.0092 0.0085 0.0065 0.0032 ...
##  $ V58  : num  0.0084 0.0049 0.0164 0.0044 0.0048 0.0027 0.0143 0.0047 0.0093 0.0035 ...
##  $ V59  : num  0.009 0.0052 0.0095 0.004 0.0107 0.0051 0.0036 0.0048 0.0059 0.0056 ...
##  $ V60  : num  0.0032 0.0044 0.0078 0.0117 0.0094 0.0062 0.0103 0.0053 0.0022 0.004 ...
```

```
##  $ Class: Factor w/ 2 levels "M","R": 2 2 2 2 2 2 2 2 2 2 ...
```

Lets create the train and test data set.Target variable is Class

```
library(caTools)
library(caret)
ind = createDataPartition(DataFrame$Class, p = 2/3, list = FALSE)
trainDF<-DataFrame[ind,]
testDF<-DataFrame[-ind,]
```

We will be using the caret package for crossvalidation.Function named train in caret package is used for crossvalidation. Let's choose the paramters for the train function in caret number = 5(It means we are using 5 fold cross-validation) method="cv"(Means we are using cross-validation.You can also choose other like LOOCV or repeated CV,etc.) classProbs=TRUE(It will give the probabilities for each class.Not just the class labels)

```
ControlParamteres <- trainControl(method = "cv",
                                  number = 5,
                                  savePredictions = TRUE,
                                  classProbs = TRUE
)
```

We will put the above paramter in the model below in trControl argument

Following are the Tuning parameters which one can tune for xgboost model in caret:

1. nrounds (# Boosting Iterations) It is the number of iterations the model runs before it stops.With higher value of nrounds model will take more time and vice-versa.

2. max_depth (Max Tree Depth) Higher value of max_depth will create more deeper trees or we can say it will create more complex model.Higher value of max_depth may create overfitting and lower value of max_depth may create underfitting.All depends on data in hand.Default value is 6. range: [1,infinity]

3. eta (Shrinkage) It is learning rate which is step size shrinkage which actually shrinks the feature weights. With high value of eta,model will run fast and vice versa.With higher eta and lesser nrounds,model will take lesser time to run.With lower eta and higher nrounds model will take more time. range: [0,1]

4. gamma (Minimum Loss Reduction) It is minimum loss reduction required to make a further partition on a leaf node of the tree. The larger value will create more conservative model.
   One can play with this parameter also but mostly other parameters are used for model tuning. range: [0,infinity]

5. colsample_bytree (Subsample Ratio of Columns) Randomly choosing the number of columns out of all columns or variables at a time while tree building process.You can think of mtry paramter in random forest to begin understanding more about this.Higher value may create overfitting and lower value may create underfitting.One needs to play with this value. range: (0,1]

6. min_child_weight (Minimum Sum of Instance Weight) You can try to begin with thinking of min bucket size in decision tree(rpart).It is like number of observations a terminal node.If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning. In linear regression mode, this simply corresponds to minimum number of instances needed to be in each node range: [0,infinity]

Why do we need model tuning? As we have already seen there are lot of parameters in xgboost model like eta,colsample_bytree,etc.You do not know which values of each parameters would give you the best predictive model.So you need to create a grid of several combinations of parameters which you think that can deliver best results.You can start by your intuition and later on keep on modifying the paramters till you are satisfied with the results. Here,for demonstration purpose I'm only choosing two values of colsample_bytree and two values of max_depth.For rest of the parameters single value is taken.

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

To check how this grid looks like type as below.It gives four combinations of parameters.You can choose more combinations if you need or want.

```
parametersGrid
```

```
##   eta colsample_bytree max_depth nrounds gamma min_child_weight
## 1 0.1              0.5         3     100     1                2
## 2 0.1              0.7         3     100     1                2
## 3 0.1              0.5         6     100     1                2
## 4 0.1              0.7         6     100     1                2
```

Let's now do the 5-fold crossvalidation for the xboost the model with the chosen parameters grid using train function.We will put the parametersGrid in the tuneGrid argument and controlParameters in trControl argument of train function. To know more about the train function type and run ?train in the console

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

```
## Loading required package: xgboost
```

```
## Warning: package 'xgboost' was built under R version 3.2.5
```

```
## Loading required package: plyr
```

```
## Warning: package 'plyr' was built under R version 3.2.5
```

Let's check the crossvalidation results for parameters tuning for xgboost model.We can easily see that there are four rows with each having the combination and their corresponding accuracy and kappa,etc.For max_depth=3 and colsample_bytree=0.5(rest values are fixed as we choose),the value of accuracy and kappa is 0.8203 and 0.6375(approx) respectively. As the max_depth =3 and colsample_bytree=0.7 gives the best accuracy,the final model is choosen for [nrounds = 100, max_depth = 3, eta = 0.1, gamma = 1, colsample_bytree = 0.7 and min_child_weight = 2].You can choose any customized metric other than accuracy.You have to put that in trainControl function.

```
modelxgboost
```

```
## eXtreme Gradient Boosting
##
## 139 samples
##  60 predictor
##   2 classes: 'M', 'R'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 111, 111, 112, 111, 111
## Resampling results across tuning parameters:
##
##   max_depth  colsample_bytree  Accuracy   Kappa      Accuracy SD
##   3          0.5               0.8211640  0.6387160  0.10367114
##   3          0.7               0.8066138  0.6087258  0.09566611
##   6          0.5               0.8423280  0.6826879  0.10234625
##   6          0.7               0.7994709  0.5956444  0.11664837
##   Kappa SD
##   0.2096112
##   0.1942108
##   0.2064117
##   0.2360170
##
## Tuning parameter 'nrounds' was held constant at a value of 100
##
## Tuning parameter 'gamma' was held constant at a value of 1
##
## Tuning parameter 'min_child_weight' was held constant at a value of 2
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were nrounds = 100, max_depth = 6,
##  eta = 0.1, gamma = 1, colsample_bytree = 0.5 and min_child_weight = 2.
```

Let's check the predictions on the test data set

```
predictions<-predict(modelxgboost,testDF)
```

Let's check the confusion matrix

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

```
##            actual
## predictions  M  R
##           M 33  6
##           R  4 26
```