```r
library(lattice)
library(ggplot2)
library(caret)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
library(doParallel) # parallel processing
```

```
## Loading required package: foreach

## Loading required package: iterators

## Loading required package: parallel
```

```r
registerDoParallel(10) # Registrer a parallel backend for train
getDoParWorkers()
```

```
## [1] 10
```

```r
Data <- read.csv("EEGdata.csv", sep = ",", header = TRUE)
names(Data)
```

```
##  [1] "subject.ID"         "Video.ID"           "Attention"
##  [4] "Meditation"         "Raw"                "Delta"
##  [7] "Theta"              "Alpha.1"            "Alpha.2"
## [10] "Beta.1"             "Beta.2"             "Gamma1"
## [13] "Gamma2"             "predefined.label"   "Self.defined.label"
```

```r
#colnames(Data)[colnames(Data) == 'predefined.label'] <- 'class'
colnames(Data)[colnames(Data) == 'Self.defined.label'] <- 'class'
#Data$predefined.label <-NULL

str(Data)
```

```
## 'data.frame':    12811 obs. of  15 variables:
##  $ subject.ID      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Video.ID        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Attention       : int  56 40 47 47 44 44 43 40 43 47 ...
##  $ Meditation      : int  43 35 48 57 53 66 69 61 69 69 ...
##  $ Raw             : int  278 -50 101 -5 -8 73 130 -2 17 -59 ...
##  $ Delta           : int  301963 73787 758353 2012240 1005145 1786446 635191 161098 492796 82048 ...
##  $ Theta           : int  90612 28083 383745 129350 354328 176766 122446 12119 120998 116131 ...
##  $ Alpha.1         : int  33735 1439 201999 61236 37102 59352 90107 1963 63697 47317 ...
##  $ Alpha.2         : int  23991 2240 62107 17084 88881 26157 65072 809 68242 26197 ...
##  $ Beta.1          : int  27946 2746 36293 11488 45307 15054 36230 1277 10769 41642 ...
##  $ Beta.2          : int  45097 3687 130536 62462 99603 33669 53019 3186 88403 28866 ...
##  $ Gamma1          : int  33228 5293 57243 49960 44790 33782 62938 3266 73756 32551 ...
##  $ Gamma2          : int  8293 2740 25354 33932 29749 31750 59307 2518 22676 41810 ...
##  $ predefined.label: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ class           : int  0 0 0 0 0 0 0 0 0 0 ...
```

```r
Data$class <- ifelse(Data$class == 0, "X0", "X1")
#drops <- Data$predefined.label
```

```r
Data$class <- factor(Data$class)
intrain <-createDataPartition(y = Data$class, p = 0.8, list = FALSE) #split data
#Make training and test data sets global
assign("training", Data[intrain, ] , envir = .GlobalEnv)
assign("testing",  Data[-intrain, ] , envir = .GlobalEnv)
training[["class"]] = factor(training[["class"]]) #factor the label(class) column
testing[["class"]] = factor(testing[["class"]])
dim(training)
```

```
## [1] 10250    15
```

```r
dim(testing)
```

```
## [1] 2561   15
```

```r
str(training)
```

```
## 'data.frame':    10250 obs. of  15 variables:
##  $ subject.ID      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Video.ID        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Attention       : int  56 40 44 44 43 40 43 47 48 44 ...
##  $ Meditation      : int  43 35 53 66 69 61 69 69 38 48 ...
##  $ Raw             : int  278 -50 -8 73 130 -2 17 -59 -14 72 ...
##  $ Delta           : int  301963 73787 1005145 1786446 635191 161098 492796 82048 757165 667513 ...
##  $ Theta           : int  90612 28083 354328 176766 122446 12119 120998 116131 186196 141854 ...
##  $ Alpha.1         : int  33735 1439 37102 59352 90107 1963 63697 47317 3242 75050 ...
##  $ Alpha.2         : int  23991 2240 88881 26157 65072 809 68242 26197 3841 16234 ...
##  $ Beta.1          : int  27946 2746 45307 15054 36230 1277 10769 41642 18854 45926 ...
##  $ Beta.2          : int  45097 3687 99603 33669 53019 3186 88403 28866 43021 34496 ...
##  $ Gamma1          : int  33228 5293 44790 33782 62938 3266 73756 32551 46799 74875 ...
##  $ Gamma2          : int  8293 2740 29749 31750 59307 2518 22676 41810 11928 31839 ...
##  $ predefined.label: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ class           : Factor w/ 2 levels "X0","X1": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
str(testing)
```

```
## 'data.frame':    2561 obs. of  15 variables:
##  $ subject.ID      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Video.ID        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Attention       : int  47 47 47 57 67 37 20 20 48 54 ...
##  $ Meditation      : int  48 57 34 40 53 21 56 67 48 67 ...
##  $ Raw             : int  101 -5 121 144 25 -51 -21 4 -22 186 ...
##  $ Delta           : int  758353 2012240 165360 671467 756442 632782 1099119 1969931 637681 1103969
##  $ Theta           : int  383745 129350 42119 133227 97387 162376 673426 185293 30439 83627 ...
##  $ Alpha.1         : int  201999 61236 3158 7142 59785 12692 74729 41196 21521 68911 ...
##  $ Alpha.2         : int  62107 17084 6256 14300 32241 3991 93214 21801 10035 12218 ...
##  $ Beta.1          : int  36293 11488 7270 23373 16122 5848 67019 7827 12856 44447 ...
##  $ Beta.2          : int  130536 62462 19462 65591 53157 23018 50197 35523 52824 32024 ...
##  $ Gamma1          : int  57243 49960 10984 47860 44602 22561 88444 22243 20595 86928 ...
##  $ Gamma2          : int  25354 33932 8148 16501 14331 9851 30954 23871 17686 33393 ...
##  $ predefined.label: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ class           : Factor w/ 2 levels "X0","X1": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
training <- training[,-14]
testing <- testing[,-14]
dim(training)
```

```
## [1] 10250    14
```

```r
dim(testing)
```

```
## [1] 2561   14
```

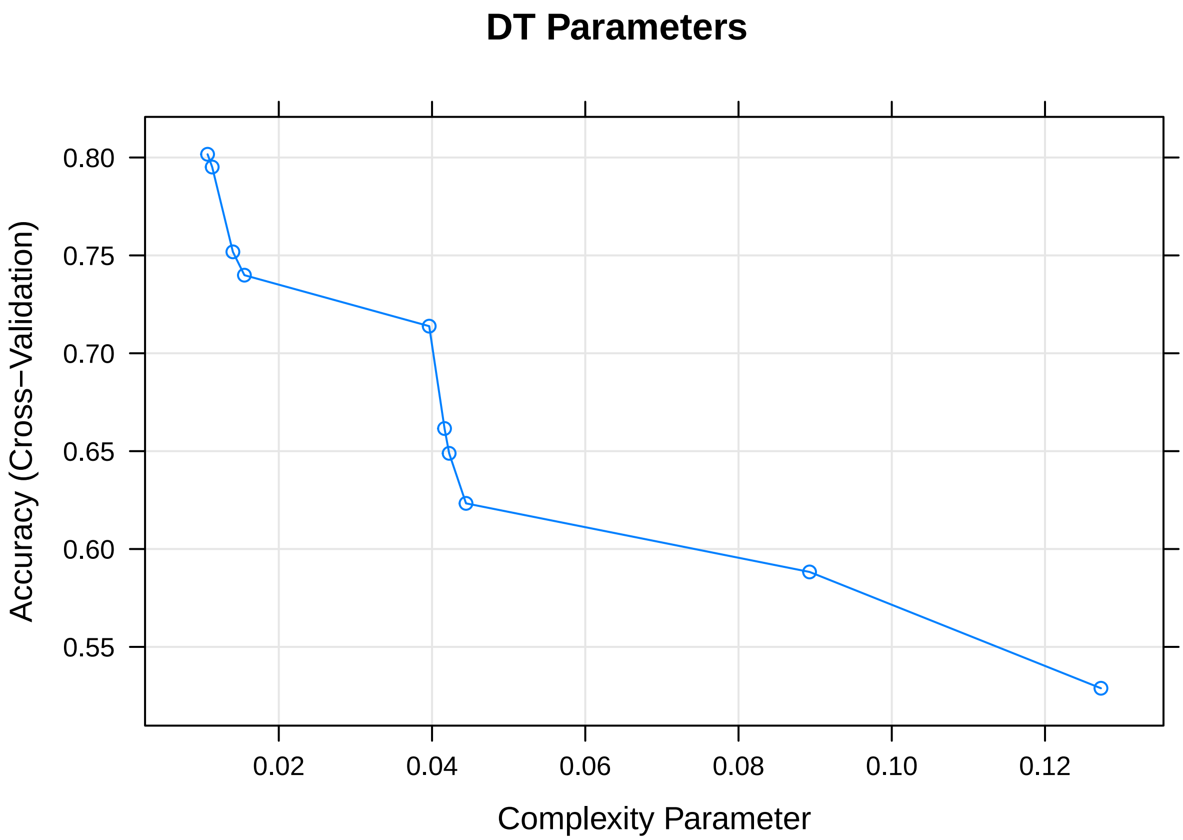
```r
#rpart
set.seed(31415)
ctrl.cross <- trainControl(method = "cv",
                           number = 10,
                           repeats = 3)
dc.Fit <-   train(
  class ~ .,
  data = training,
  method = "rpart",
  preProc = c("center", "scale"),
  tuneLength = 10,
  parms = list(split = "information"),
  trControl = ctrl.cross
)
```

```
## Loading required package: rpart
```

```r
dc.Fit
```

```
## CART
##
## 10250 samples
##    13 predictor
##     2 classes: 'X0', 'X1'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 9224, 9225, 9226, 9224, 9225, 9226, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.01070857  0.8016581  0.60162757
##   0.01130905  0.7951226  0.58837643
##   0.01401121  0.7518041  0.50005520
##   0.01551241  0.7399006  0.47516367
##   0.03963171  0.7138470  0.42330979
##   0.04163331  0.6615565  0.31518132
##   0.04223379  0.6488798  0.28794003
##   0.04443555  0.6233188  0.23604804
##   0.08927142  0.5882890  0.16142144
##   0.12730184  0.5288780  0.03473016
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.01070857.
```

```r
plot(dc.Fit, main = "DT Parameters")
```




```r
dc.Pred <- predict(dc.Fit, testing) #testing
cm <- confusionMatrix(dc.Pred, testing$class)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   X0   X1
##         X0  898  171
##         X1  350 1142
##
##                Accuracy : 0.7966
##                  95% CI : (0.7804, 0.812)
##     No Information Rate : 0.5127
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5914
##  Mcnemar's Test P-Value : 6.274e-15
##
##             Sensitivity : 0.7196
##             Specificity : 0.8698
##          Pos Pred Value : 0.8400
##          Neg Pred Value : 0.7654
##              Prevalence : 0.4873
##          Detection Rate : 0.3506
##    Detection Prevalence : 0.4174
##       Balanced Accuracy : 0.7947
##
```

```
##        'Positive' Class : X0
##
```

```r
# RF
set.seed(31415)
n <- dim(training)[2]
#gridRF <- expand.grid(mtry = seq(from=0,by=as.integer(n/10),to=n)[-1])
gridRF <-
  expand.grid(mtry = seq(from = 0, by = 1, to = n)[-1]) #may need to change this depend on your data si
ctrl.crossRF <-
  trainControl(
    method = "cv",
    number = 10,
    classProbs = TRUE,
    savePredictions = TRUE,
    allowParallel = TRUE
  )
rf.Fit <-
  train(
    class ~ .,
    data = training,
    method = "rf",
    metric = "Accuracy",
    preProc = c("center", "scale"),
    ntree = 200,
    tuneGrid = gridRF,
    trControl = ctrl.crossRF
  )
```

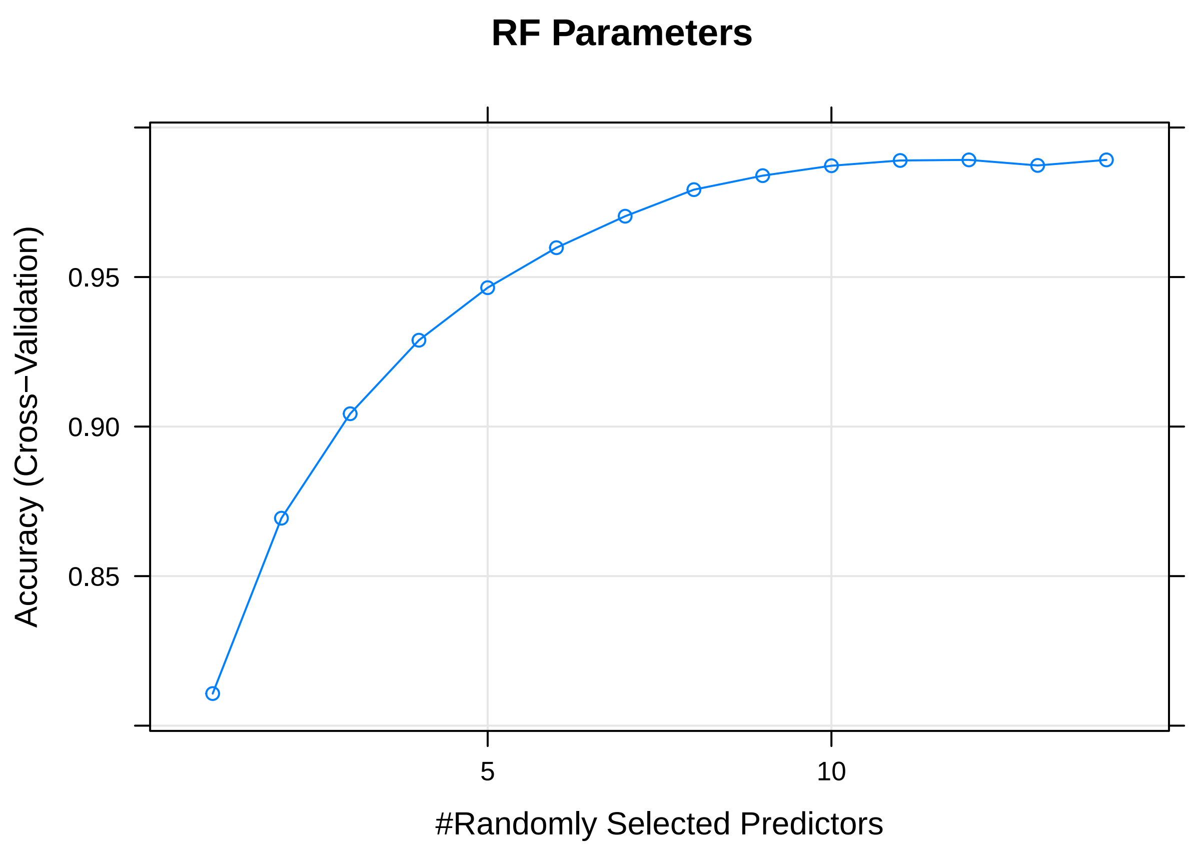
```
## Loading required package: randomForest

## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```r
rf.Fit
```

```
## Random Forest
##
## 10250 samples
##    13 predictor
##     2 classes: 'X0', 'X1'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 9224, 9225, 9226, 9224, 9225, 9226, ...
## Resampling results across tuning parameters:
##
```

```
##    mtry  Accuracy   Kappa
##     1    0.8107349  0.6209373
##     2    0.8693684  0.7380140
##     3    0.9042936  0.8080606
##     4    0.9288780  0.8574043
##     5    0.9464388  0.8926466
##     6    0.9598046  0.9194789
##     7    0.9703410  0.9406094
##     8    0.9792193  0.9583977
##     9    0.9839022  0.9677807
##    10    0.9872190  0.9744223
##    11    0.9889760  0.9779401
##    12    0.9891703  0.9783282
##    13    0.9873174  0.9746201
##    14    0.9891704  0.9783292
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 14.
```

```
plot(rf.Fit, main = "RF Parameters")
```




```
rf.Pred <- predict(rf.Fit, testing) #testing
cm <- confusionMatrix(rf.Pred, testing$class)
cm
```

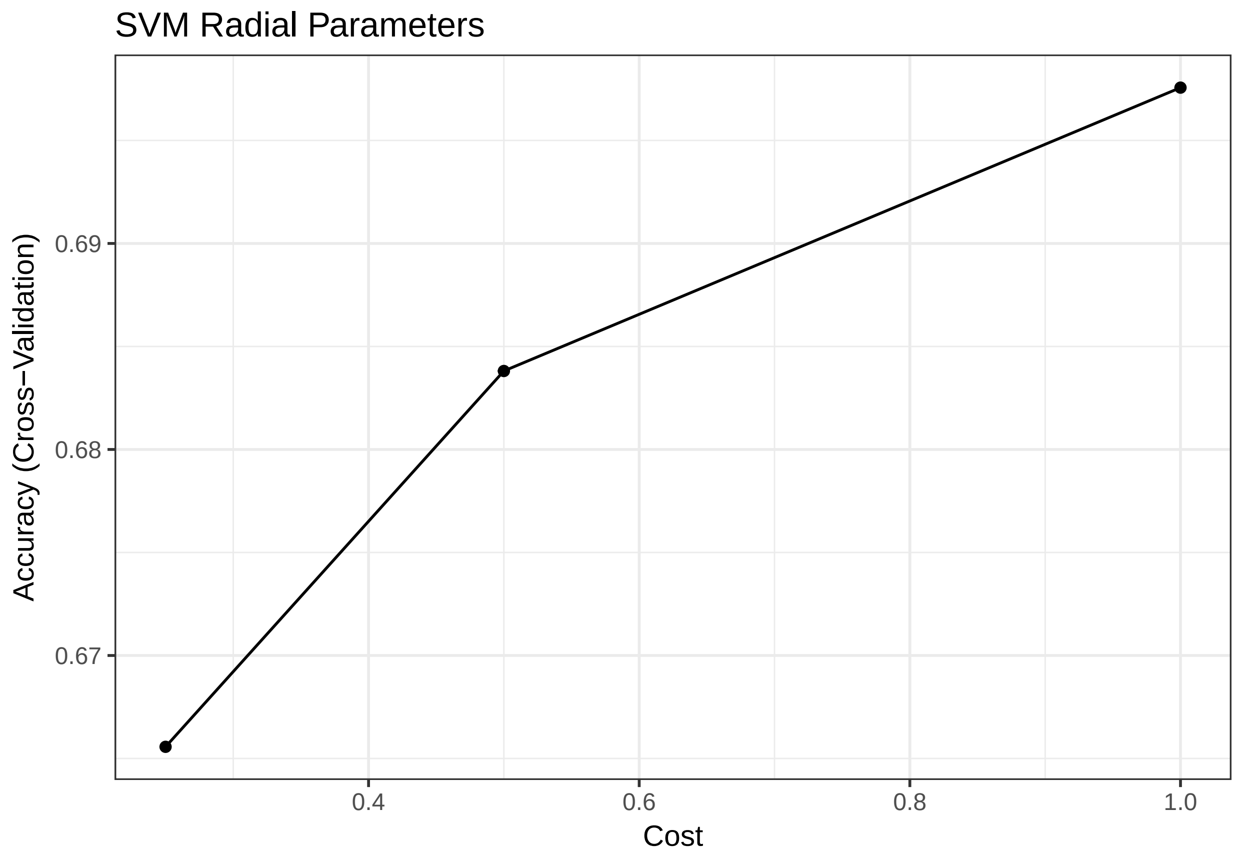
```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction   X0   X1
##         X0 1236   15
##         X1   12 1298
##
##                Accuracy : 0.9895
##                  95% CI : (0.9847, 0.993)
##     No Information Rate : 0.5127
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9789
##  Mcnemar's Test P-Value : 0.7003
##
##             Sensitivity : 0.9904
##             Specificity : 0.9886
##          Pos Pred Value : 0.9880
##          Neg Pred Value : 0.9908
##              Prevalence : 0.4873
##          Detection Rate : 0.4826
##    Detection Prevalence : 0.4885
##       Balanced Accuracy : 0.9895
##
##        'Positive' Class : X0
##
```

```r
# SVM Radial basis kernel
set.seed(31415)
control <- trainControl(method = "cv",
                        number = 10,
                        repeats = 3)
svm.radial.Fit <-
  train(class ~ .,
        data = training,
        method = "svmRadial",
        trControl = control)
svm.radial.Fit
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 10250 samples
##    13 predictor
##     2 classes: 'X0', 'X1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 9224, 9225, 9226, 9224, 9225, 9226, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.6655669  0.3301825
##   0.50  0.6838095  0.3669632
##   1.00  0.6975640  0.3945717
##
## Tuning parameter 'sigma' was held constant at a value of 0.1304632
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were sigma = 0.1304632 and C = 1.
```

```r
ggplot(svm.radial.Fit) + theme_bw() + ggtitle("SVM Radial Parameters")
```




```r
svm.radial.Pred <- predict(svm.radial.Fit, testing) #testing
print("Prediction and confusion matrix of the testing dataset")
```

```
## [1] "Prediction and confusion matrix of the testing dataset"
```

```r
cm <- confusionMatrix(svm.radial.Pred, testing$class)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  X0  X1
##         X0 851 353
##         X1 397 960
##
##                Accuracy : 0.7071
##                  95% CI : (0.6891, 0.7247)
##     No Information Rate : 0.5127
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4134
##  Mcnemar's Test P-Value : 0.1164
##
##             Sensitivity : 0.6819
##             Specificity : 0.7312
##          Pos Pred Value : 0.7068
```

```
##          Neg Pred Value : 0.7074
##              Prevalence : 0.4873
##          Detection Rate : 0.3323
##    Detection Prevalence : 0.4701
##       Balanced Accuracy : 0.7065
##
##        'Positive' Class : X0
##
```

```r
# KNN
set.seed(31415)
grid <- expand.grid(
  kmax = seq(from = 2, to = 28, by = 1),
  distance = 2,
  #kernel = c("triangular", "rectangular", "epanechnikov", "optimal"))
  kernel = "optimal"
)
ctrl.cross <- trainControl(method = "cv",
                           number = 10,
                           repeats = 3) #
knn.Fit <- train(
  class ~ .,
  #data = training,method = "knn", trControl = ctrl, preProcess = c("center","scale"), tuneLength = 20
  data = training,
  method = "kknn",
  metric = "Accuracy",
  perProc = c("center", "scale"),
  tuneGrid = grid,
  trControl = ctrl.cross
)
```

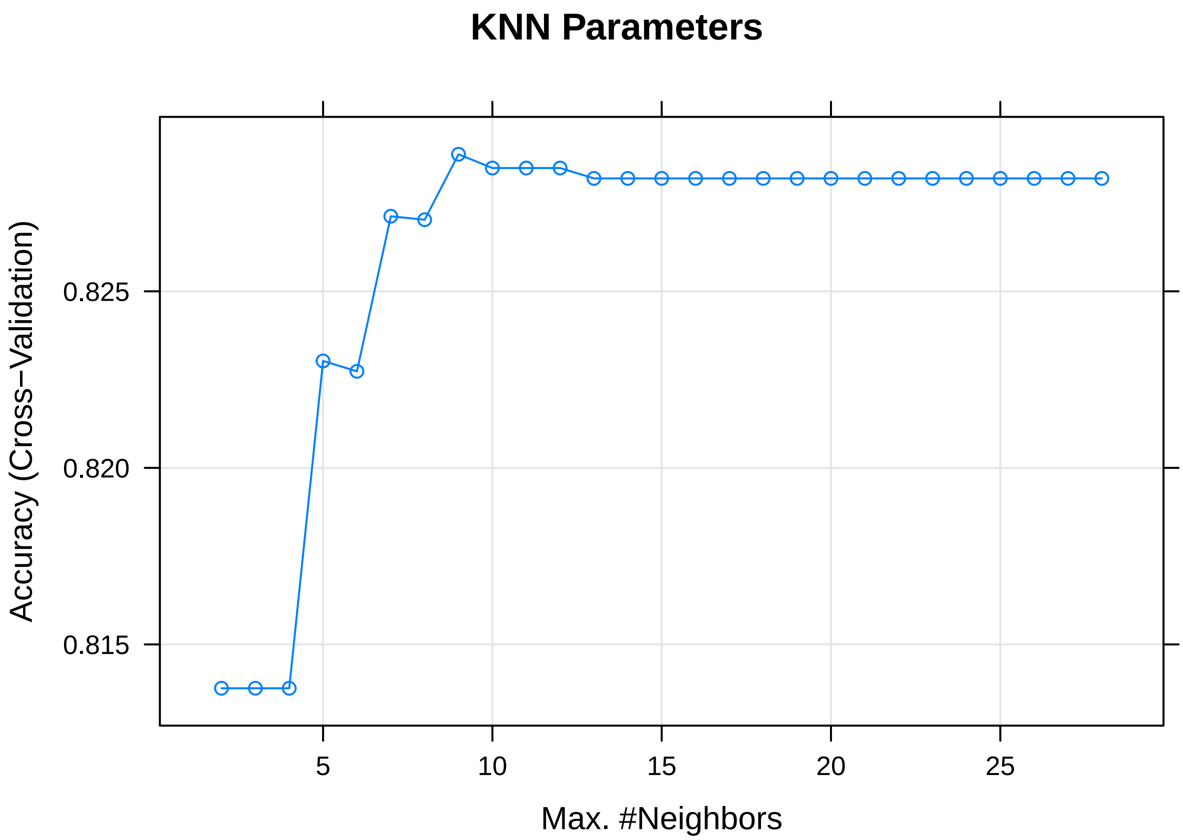
```
## Loading required package: kknn

##
## Attaching package: 'kknn'

## The following object is masked from 'package:caret':
##
##     contr.dummy
```

```r
knn.Fit
```

```
## k-Nearest Neighbors
##
## 10250 samples
##    13 predictor
##     2 classes: 'X0', 'X1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 9224, 9225, 9226, 9224, 9225, 9226, ...
## Resampling results across tuning parameters:
##
##   kmax  Accuracy   Kappa
##    2    0.8137585  0.6271751
##    3    0.8137585  0.6271751
##    4    0.8137585  0.6271751
```

```
##    5    0.8230274  0.6456603
##    6    0.8227349  0.6450548
##    7    0.8271259  0.6538310
##    8    0.8270277  0.6536290
##    9    0.8288815  0.6573331
##   10    0.8284913  0.6565517
##   11    0.8284913  0.6565517
##   12    0.8284913  0.6565517
##   13    0.8281986  0.6559744
##   14    0.8281986  0.6559744
##   15    0.8281986  0.6559744
##   16    0.8281986  0.6559744
##   17    0.8281986  0.6559744
##   18    0.8281986  0.6559744
##   19    0.8281986  0.6559744
##   20    0.8281986  0.6559744
##   21    0.8281986  0.6559744
##   22    0.8281986  0.6559744
##   23    0.8281986  0.6559744
##   24    0.8281986  0.6559744
##   25    0.8281986  0.6559744
##   26    0.8281986  0.6559744
##   27    0.8281986  0.6559744
##   28    0.8281986  0.6559744
##
## Tuning parameter 'distance' was held constant at a value of 2
##
## Tuning parameter 'kernel' was held constant at a value of optimal
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were kmax = 9, distance = 2 and
##  kernel = optimal.
```

```r
plot(knn.Fit, main = "KNN Parameters")
```




```
knn.Pred <- predict(knn.Fit, testing) #testing
print("Prediction and confusion matrix of the testing dataset")
```

```
## [1] "Prediction and confusion matrix of the testing dataset"
```

```
cm <- confusionMatrix(knn.Pred, testing$class)
cm
```

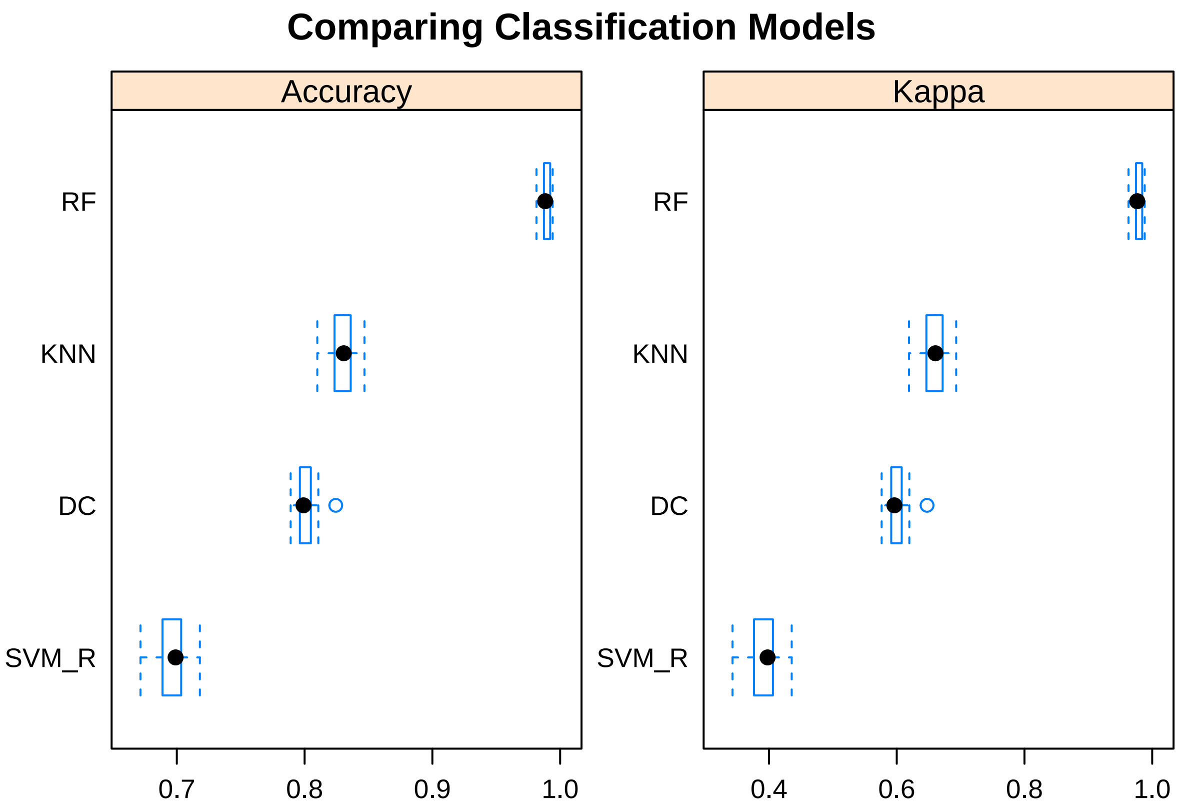
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   X0   X1
##         X0 1002  205
##         X1  246 1108
##
##                Accuracy : 0.8239
##                  95% CI : (0.8086, 0.8385)
##     No Information Rate : 0.5127
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6473
##  Mcnemar's Test P-Value : 0.05963
##
##             Sensitivity : 0.8029
##             Specificity : 0.8439
##          Pos Pred Value : 0.8302
##          Neg Pred Value : 0.8183
##              Prevalence : 0.4873
```

```
##          Detection Rate : 0.3913
##    Detection Prevalence : 0.4713
##       Balanced Accuracy : 0.8234
##
##        'Positive' Class : X0
##
```

```r
results <-
  resamples(list(
    DC = dc.Fit,
    RF = rf.Fit,
    SVM_R = svm.radial.Fit,
    KNN = knn.Fit
  ))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: DC, RF, SVM_R, KNN
## Number of resamples: 10
##
## Accuracy
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## DC    0.7890625 0.7967344 0.7991219 0.8016581 0.8043421 0.8243902    0
## RF    0.9814634 0.9875610 0.9882984 0.9891704 0.9919548 0.9941463    0
## SVM_R 0.6715400 0.6889486 0.6990244 0.6975640 0.7032432 0.7180488    0
## KNN   0.8099415 0.8234146 0.8306498 0.8288815 0.8350810 0.8468293    0
##
## Kappa
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## DC    0.5763392 0.5920572 0.5963497 0.6016276 0.6068841 0.6474850    0
## RF    0.9629065 0.9751069 0.9765870 0.9783292 0.9839029 0.9882857    0
## SVM_R 0.3428202 0.3769608 0.3977933 0.3945717 0.4057590 0.4355200    0
## KNN   0.6192435 0.6464843 0.6606373 0.6573331 0.6698194 0.6930816    0
```

```r
# box and whisker plots to compare models
scales <- list(x = list(relation = "free"),
               y = list(relation = "free"))
bwplot(results, scales = scales, main = "Comparing Classification Models")
```


Comparing Classification Models
Accuracy
Kappa
RF
KNN
DC
SVM_R
0.7
0.8
0.9
1.0
0.4
0.6
0.8
1.0