Compare_ML_Algos

Nilufar Isakova

11/5/2017

```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'
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 ...
##  $ class             : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ Self.defined.label: int  0 0 0 0 0 0 0 0 0 0 ...
```

```r
Data$class <-ifelse(Data$class == 0,"X0","X1")
#drops <- Data$predefined.label
Data$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] 10249    15
```

```r
dim(testing)
```

```
## [1] 2562   15
```

```r
anyNA(Data)
```

```
## [1] FALSE
```

```r
#rpart
set.seed(31415)
#grid <- expand.grid(cp = 2^seq(from = -30 , to= 0, by = 2) ) tuneGrid= grid,
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)
```

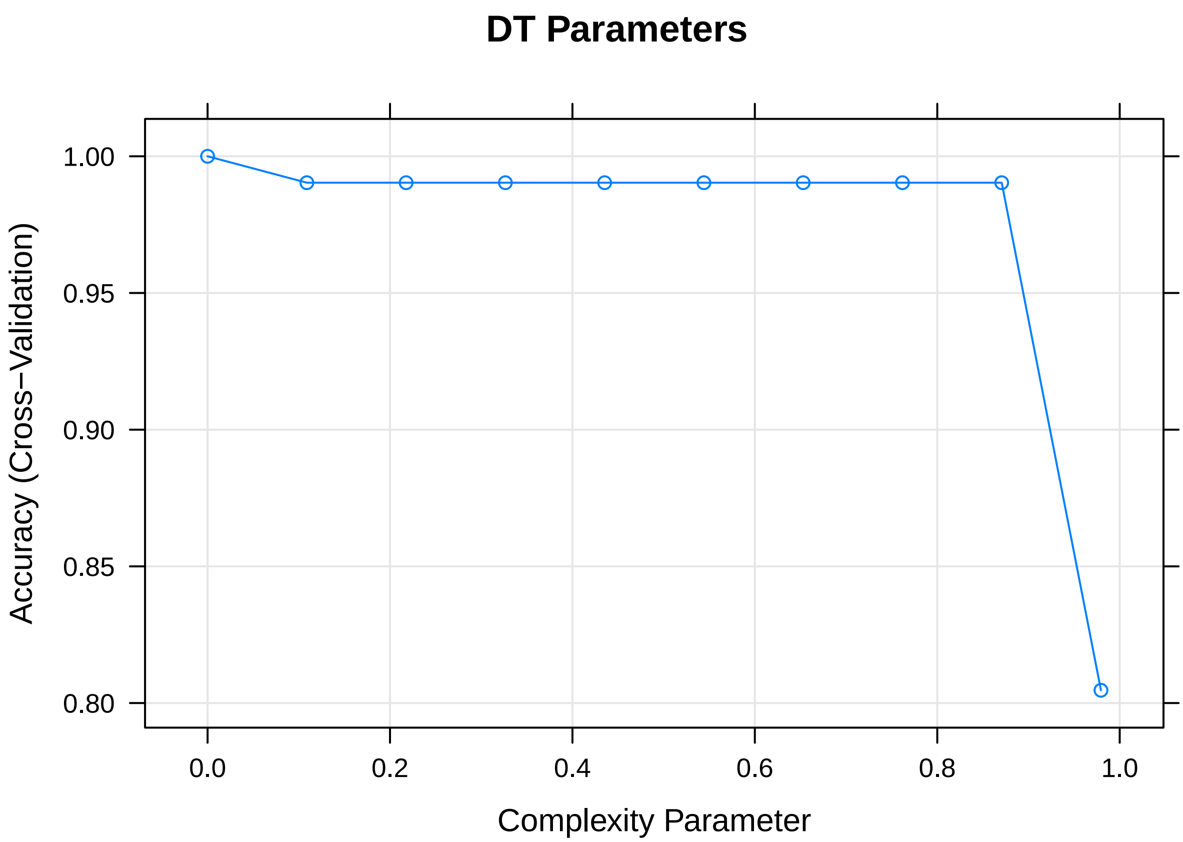
```
## Loading required package: rpart
```

```r
dc.Fit
```

```
## CART
##
## 10249 samples
##    14 predictor
##     2 classes: 'X0', 'X1'
##
## Pre-processing: centered (14), scaled (14)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 9224, 9224, 9225, 9224, 9224, 9224, ...
## Resampling results across tuning parameters:
##
##   cp         Accuracy   Kappa
##   0.0000000  1.0000000  1.0000000
##   0.1088294  0.9903399  0.9806363
##   0.2176589  0.9903399  0.9806363
##   0.3264883  0.9903399  0.9806363
##   0.4353177  0.9903399  0.9806363
##   0.5441471  0.9903399  0.9806363
##   0.6529766  0.9903399  0.9806363
##   0.7618060  0.9903399  0.9806363
##   0.8706354  0.9903399  0.9806363
##   0.9794648  0.8046816  0.5857235
```

```
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.
```

```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 1357    0
##         X1    0 1205
##
##                Accuracy : 1
##                  95% CI : (0.9986, 1)
##     No Information Rate : 0.5297
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
```

```
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5297
##          Detection Rate : 0.5297
##    Detection Prevalence : 0.5297
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : X0
##
```

```
# 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,allowPa
rf.Fit <- train(class ~ .,data = training,method = "rf",metric = "Accuracy",preProc = c("center", "scal
                ntree = 200, tuneGrid = gridRF,trControl = ctrl.crossRF)
```

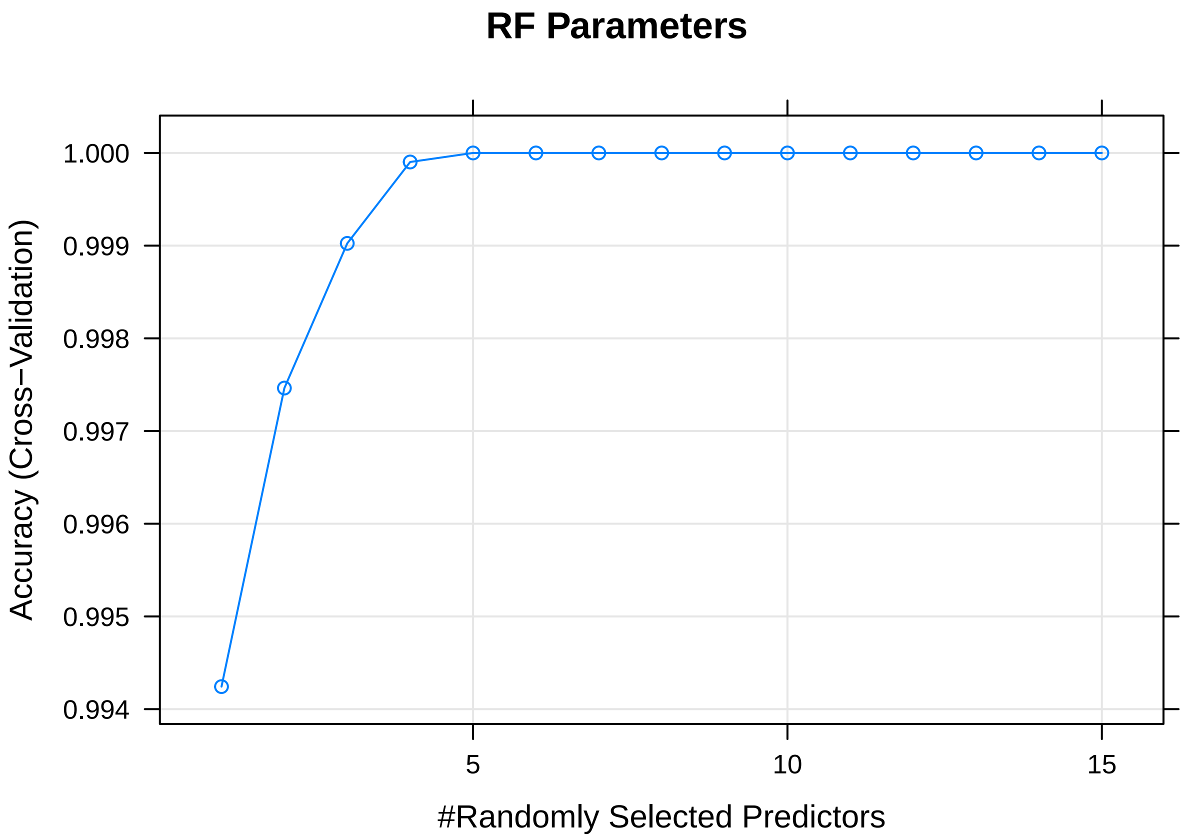
```
## Loading required package: randomForest

## 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
```

```
rf.Fit
```

```
## Random Forest
##
## 10249 samples
##    14 predictor
##     2 classes: 'X0', 'X1'
##
## Pre-processing: centered (14), scaled (14)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 9224, 9224, 9225, 9224, 9224, 9224, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.9942430  0.9884549
##    2    0.9974634  0.9949110
##    3    0.9990244  0.9980424
##    4    0.9999023  0.9998040
##    5    1.0000000  1.0000000
##    6    1.0000000  1.0000000
##    7    1.0000000  1.0000000
##    8    1.0000000  1.0000000
##    9    1.0000000  1.0000000
##   10    1.0000000  1.0000000
##   11    1.0000000  1.0000000
##   12    1.0000000  1.0000000
##   13    1.0000000  1.0000000
```

```
##    14     1.0000000  1.0000000
##    15     1.0000000  1.0000000
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 5.
```

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




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

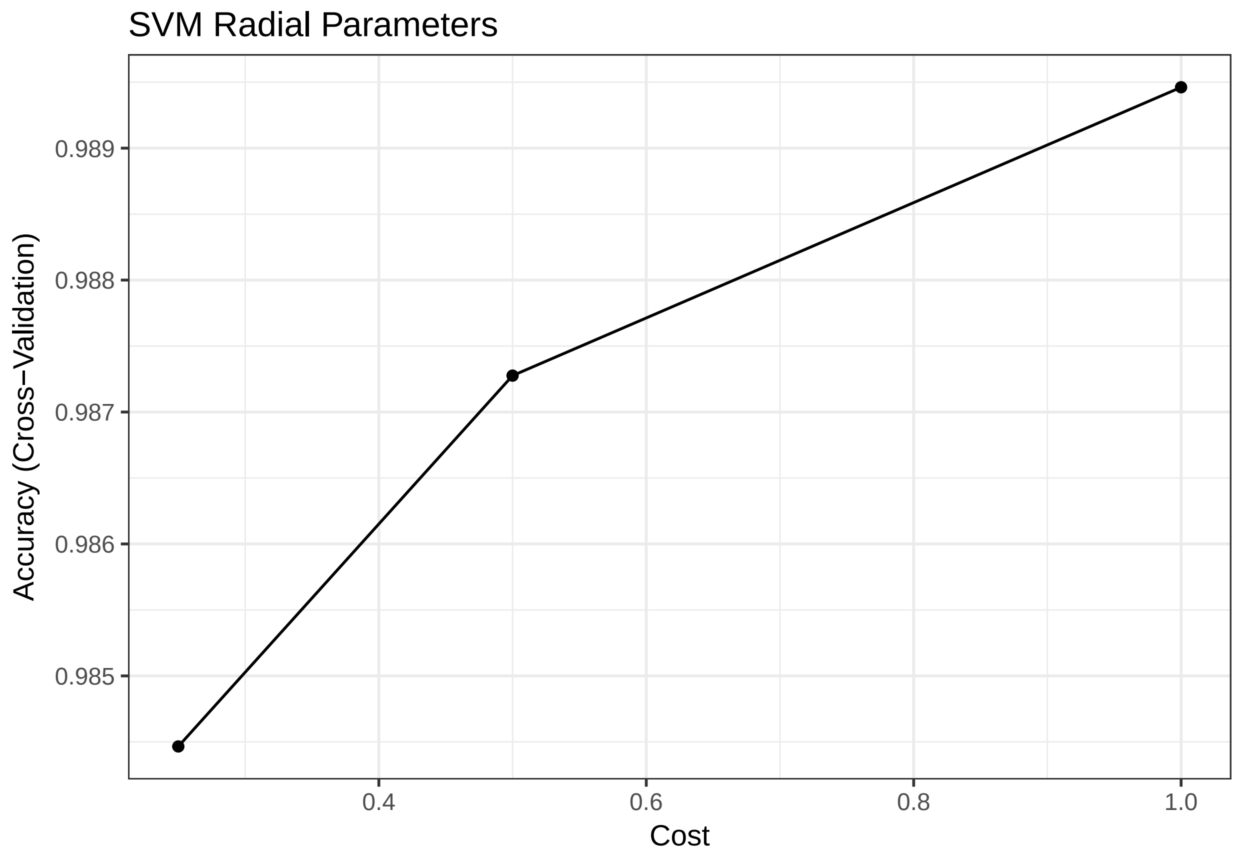
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   X0   X1
##         X0 1357    0
##         X1    0 1205
##
##                Accuracy : 1
##                  95% CI : (0.9986, 1)
##     No Information Rate : 0.5297
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
```

```
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5297
##          Detection Rate : 0.5297
##    Detection Prevalence : 0.5297
##       Balanced Accuracy : 1.0000
##
##        '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=Data, method="svmRadial", trControl=control)
svm.radial.Fit
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 12811 samples
##    14 predictor
##     2 classes: 'X0', 'X1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 11529, 11530, 11530, 11529, 11530, 11531, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.9844653  0.9687997
##   0.50  0.9872759  0.9744549
##   1.00  0.9894615  0.9788479
##
## Tuning parameter 'sigma' was held constant at a value of 0.09721334
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were sigma = 0.09721334 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 1344    9
##         X1   13 1196
##
##                Accuracy : 0.9914
##                  95% CI : (0.987, 0.9946)
##     No Information Rate : 0.5297
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9828
##  Mcnemar's Test P-Value : 0.5224
##
##             Sensitivity : 0.9904
##             Specificity : 0.9925
##          Pos Pred Value : 0.9933
##          Neg Pred Value : 0.9892
##              Prevalence : 0.5297
```

```
##          Detection Rate : 0.5246
##    Detection Prevalence : 0.5281
##       Balanced Accuracy : 0.9915
##
##        'Positive' Class : X0
##
```

```
# SVM polynomial kernel
set.seed(31415)
grid <- expand.grid(scale = 1, degree = c(1,2), C = 2^seq(from=-4,by = 1, to =6))
print("Poly Kernel SVM")
```

```
## [1] "Poly Kernel SVM"
```

```
ctrl.cross <- trainControl(method = "cv", number = 10,classProbs = TRUE,savePredictions=TRUE)
svm.poly.Fit <- train(class ~ ., data= training,
                      perProc = c("center", "scale"),
                      method = 'svmPoly', #rpart for classif. dec tree
                      metric ='Accuracy',
                      tuneGrid= grid,
                      trControl = ctrl.cross)
```

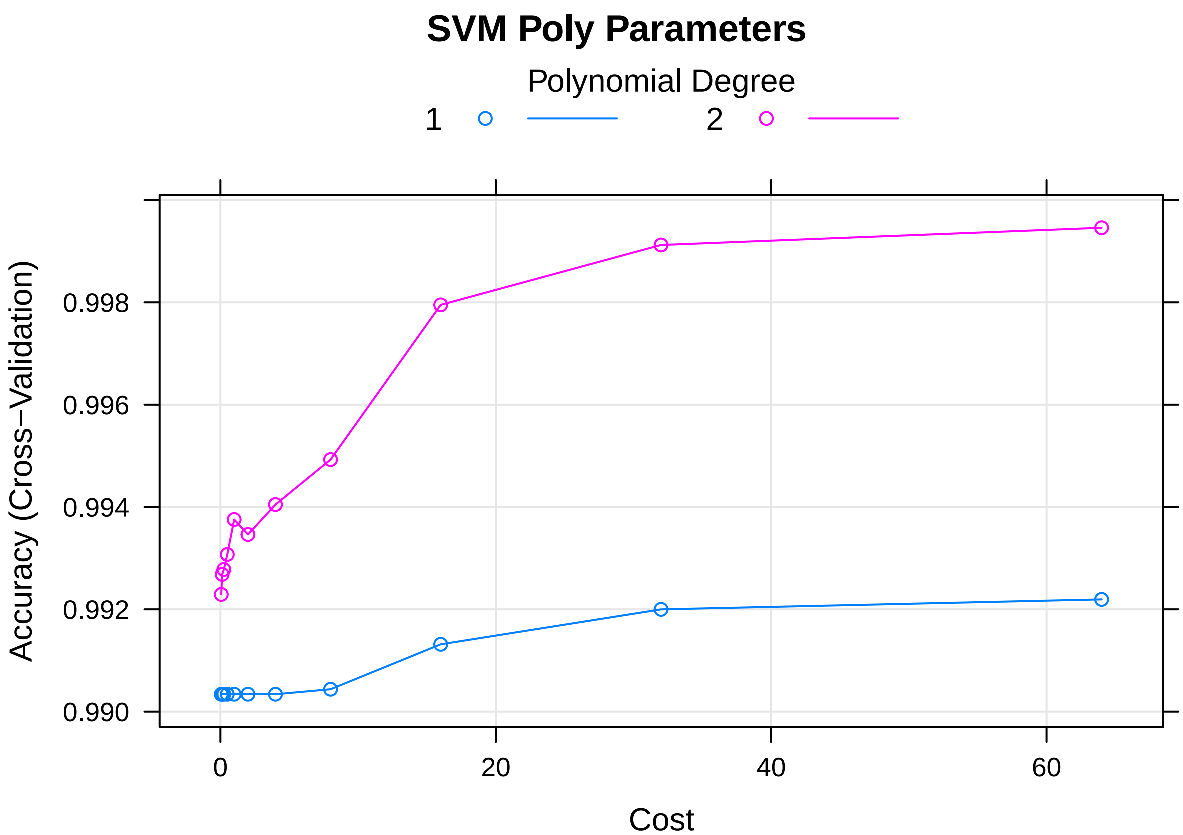
```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
svm.poly.Fit
```

```
## Support Vector Machines with Polynomial Kernel
##
## 10249 samples
##    14 predictor
##     2 classes: 'X0', 'X1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 9224, 9224, 9225, 9224, 9224, 9224, ...
## Resampling results across tuning parameters:
##
##   degree  C         Accuracy   Kappa
##   1        0.0625  0.9903399  0.9806363
##   1        0.1250  0.9903399  0.9806363
##   1        0.2500  0.9903399  0.9806363
##   1        0.5000  0.9903399  0.9806363
##   1        1.0000  0.9903399  0.9806363
##   1        2.0000  0.9903399  0.9806363
##   1        4.0000  0.9903399  0.9806363
##   1        8.0000  0.9904375  0.9808278
##   1       16.0000  0.9913158  0.9825815
##   1       32.0000  0.9919988  0.9839506
##   1       64.0000  0.9921939  0.9843435
##   2        0.0625  0.9922913  0.9845377
##   2        0.1250  0.9926819  0.9853185
##   2        0.2500  0.9927792  0.9855117
##   2        0.5000  0.9930725  0.9860995
##   2        1.0000  0.9937553  0.9874672
##   2        2.0000  0.9934627  0.9868786
```

```
##   2        4.0000  0.9940480  0.9880520
##   2        8.0000  0.9949264  0.9898151
##   2       16.0000  0.9979512  0.9958873
##   2       32.0000  0.9991220  0.9982375
##   2       64.0000  0.9994580  0.9989121
##
## Tuning parameter 'scale' was held constant at a value of 1
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were degree = 2, scale = 1 and C = 64.
```

```
plot(svm.poly.Fit,main="SVM Poly Parameters")
```




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

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

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   X0   X1
##         X0 1356    4
##         X1    1 1201
##
##                Accuracy : 0.998
##                  95% CI : (0.9955, 0.9994)
```

```
##     No Information Rate : 0.5297
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9961
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9993
##             Specificity : 0.9967
##          Pos Pred Value : 0.9971
##          Neg Pred Value : 0.9992
##              Prevalence : 0.5297
##          Detection Rate : 0.5293
##    Detection Prevalence : 0.5308
##       Balanced Accuracy : 0.9980
##
##        'Positive' Class : X0
##
```

```
# 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"), tuneLe
  data = training,
  method = "kknn",
  metric = "Accuracy",
  perProc = c("center", "scale"),
  tuneGrid = grid,
  trControl = ctrl.cross)
```

```
## Loading required package: kknn
```

```
##
## Attaching package: 'kknn'
```

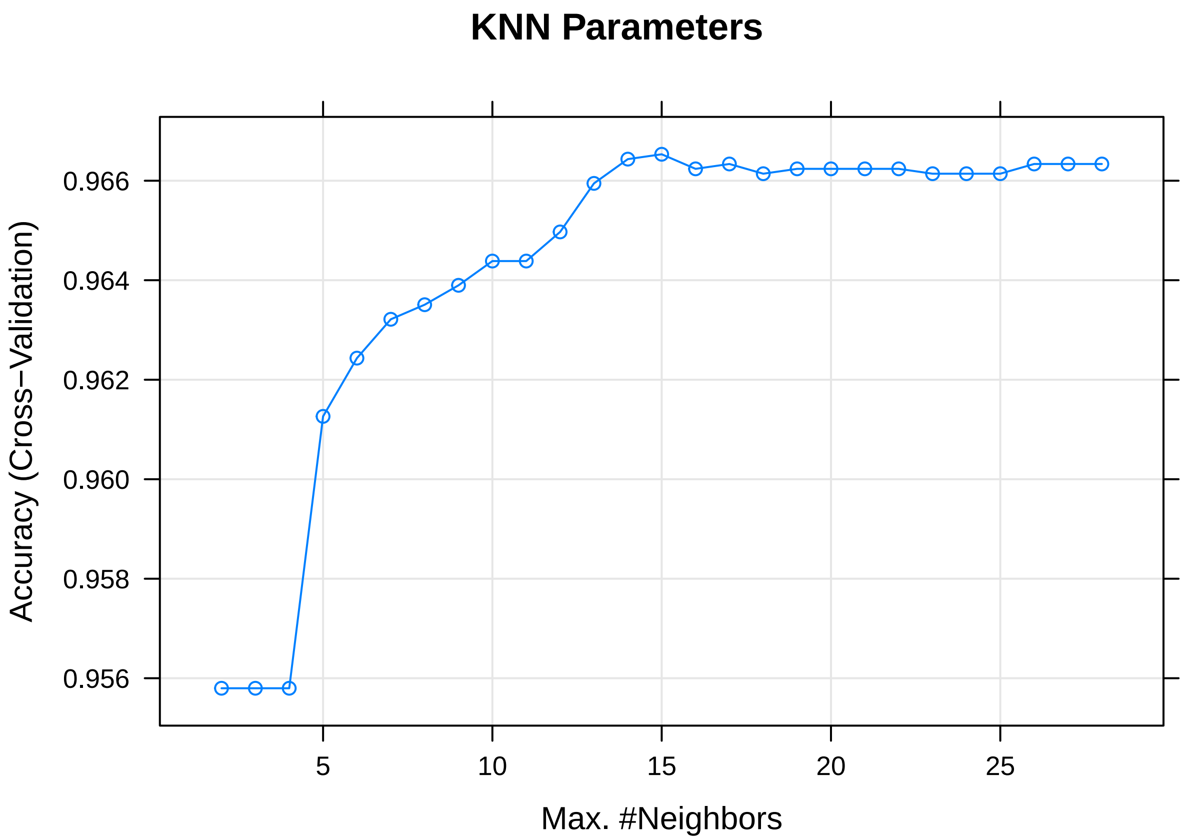
```
## The following object is masked from 'package:caret':
##
##     contr.dummy
```

```
knn.Fit
```

```
## k-Nearest Neighbors
##
## 10249 samples
##    14 predictor
##     2 classes: 'X0', 'X1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 9224, 9224, 9225, 9224, 9224, 9224, ...
## Resampling results across tuning parameters:
##
##   kmax  Accuracy   Kappa
##    2    0.9557986  0.9112412
```

```
##     3    0.9557986  0.9112412
##     4    0.9557986  0.9112412
##     5    0.9612630  0.9221788
##     6    0.9624340  0.9245281
##     7    0.9632146  0.9260909
##     8    0.9635075  0.9266787
##     9    0.9638977  0.9274552
##    10    0.9643853  0.9284392
##    11    0.9643853  0.9284372
##    12    0.9649707  0.9296140
##    13    0.9659461  0.9315733
##    14    0.9664339  0.9325484
##    15    0.9665315  0.9327439
##    16    0.9662387  0.9321536
##    17    0.9663361  0.9323483
##    18    0.9661410  0.9319579
##    19    0.9662386  0.9321468
##    20    0.9662386  0.9321468
##    21    0.9662386  0.9321468
##    22    0.9662386  0.9321468
##    23    0.9661410  0.9319487
##    24    0.9661410  0.9319487
##    25    0.9661410  0.9319487
##    26    0.9663361  0.9323407
##    27    0.9663361  0.9323407
##    28    0.9663361  0.9323407
##
## 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 = 15, distance = 2
##  and kernel = optimal.
```

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




```r
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"
```

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

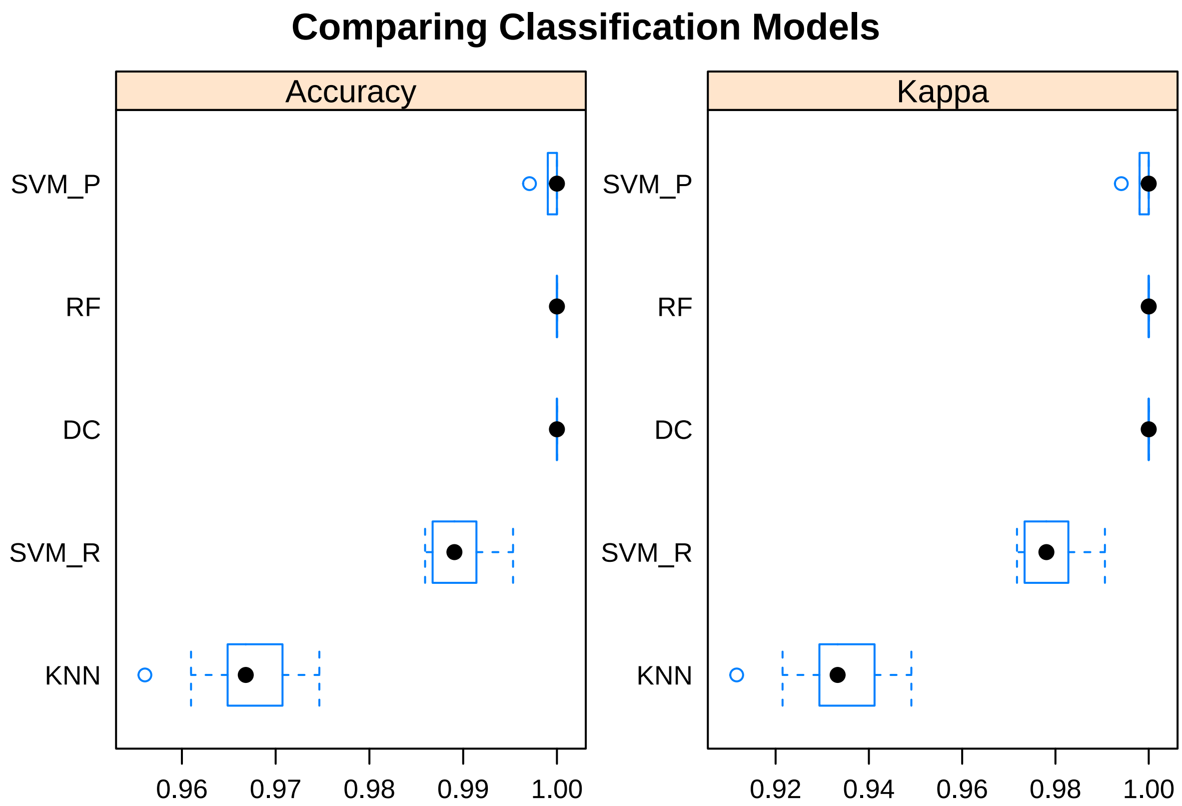
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   X0   X1
##         X0 1323   63
##         X1   34 1142
##
##                Accuracy : 0.9621
##                  95% CI : (0.954, 0.9692)
##     No Information Rate : 0.5297
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9239
##  Mcnemar's Test P-Value : 0.00447
##
##             Sensitivity : 0.9749
##             Specificity : 0.9477
##          Pos Pred Value : 0.9545
##          Neg Pred Value : 0.9711
##              Prevalence : 0.5297
```

```
##          Detection Rate : 0.5164
##    Detection Prevalence : 0.5410
##       Balanced Accuracy : 0.9613
##
##        'Positive' Class : X0
##
```

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

```
##
## Call:
## summary.resamples(object = results)
##
## Models: DC, RF, SVM_P, SVM_R, KNN
## Number of resamples: 10
##
## Accuracy
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## DC    1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000    0
## RF    1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000    0
## SVM_P 0.9970732 0.9990244 1.0000000 0.9994580 1.0000000 1.0000000    1
## SVM_R 0.9859375 0.9869320 0.9890665 0.9894615 0.9912196 0.9953198    0
## KNN   0.9560547 0.9651220 0.9668131 0.9665315 0.9697561 0.9746589    0
##
## Kappa
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## DC    1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000    0
## RF    1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000    0
## SVM_P 0.9941249 0.9980421 1.0000000 0.9989121 1.0000000 1.0000000    1
## SVM_R 0.9717545 0.9737683 0.9780620 0.9788479 0.9823856 0.9906066    0
## KNN   0.9116544 0.9298956 0.9333013 0.9327439 0.9392741 0.9491089    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
SVM_P
RF
DC
SVM_R
KNN
0.96
0.97
0.98
0.99
1.00
0.92
0.94
0.96
0.98
1.00