Iris_ML_mlr.R

Ram

Sat Oct 22 21:28:22 2016

```
#   H0   --- Model could not predict the correct class of Species #
#   H1   --- Model can predict the Species class correctly        #
#   Target  -- Species  (classification problem)                  #

setwd('G:/DATASCIENCE/DS-PRACTICE-PROJECTS/3_Iris/')

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.2.5

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

#library(data.table)
library(ggplot2)

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

library(gridExtra)
library(corrplot)

## Warning: package 'corrplot' was built under R version 3.2.5

#library(GGally)
library(caret)

## Warning: package 'caret' was built under R version 3.2.5

## Loading required package: lattice

library(mlr)

## Warning: package 'mlr' was built under R version 3.2.5

## Loading required package: BBmisc

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

```
##
## Attaching package: 'BBmisc'

## The following objects are masked from 'package:dplyr':
##
##     coalesce, collapse

## Loading required package: ParamHelpers

## Warning: package 'ParamHelpers' was built under R version 3.2.5

## Loading required package: stringi

##
## Attaching package: 'mlr'

## The following object is masked from 'package:caret':
##
##     train
```

```
data(iris)
set.seed(3)
train <- sample_frac(iris, 0.8, replace=T)
rid <- as.numeric(rownames(train))
test <- iris[-rid,]

trainTask <- makeClassifTask(data = train, target = 'Species')
testTask <- makeClassifTask(data = test, target = 'Species')

head(train)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 26            5.0         3.0          1.6         0.2     setosa
## 122           5.6         2.8          4.9         2.0  virginica
## 58            4.9         2.4          3.3         1.0 versicolor
## 50            5.0         3.3          1.4         0.2     setosa
## 91            5.5         2.6          4.4         1.2 versicolor
## 91.1          5.5         2.6          4.4         1.2 versicolor
```

```
str(train)
```

```
## 'data.frame':    120 obs. of  5 variables:
##  $ Sepal.Length: num  5 5.6 4.9 5 5.5 5.5 5.7 5.1 6.7 5.6 ...
##  $ Sepal.Width : num  3 2.8 2.4 3.3 2.6 2.6 3.8 3.8 3.1 2.7 ...
##  $ Petal.Length: num  1.6 4.9 3.3 1.4 4.4 4.4 1.7 1.9 4.7 4.2 ...
##  $ Petal.Width : num  0.2 2 1 0.2 1.2 1.2 0.3 0.4 1.5 1.3 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 3 2 1 2 2
1 1 2 2 ...
```

```
summary(train)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
```

```
##  1st Qu.:5.000   1st Qu.:2.800   1st Qu.:1.500   1st Qu.:0.200
##  Median :5.700   Median :3.100   Median :4.150   Median :1.200
##  Mean   :5.784   Mean   :3.129   Mean   :3.572   Mean   :1.123
##  3rd Qu.:6.400   3rd Qu.:3.400   3rd Qu.:5.100   3rd Qu.:1.825
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :49
##  versicolor:33
##  virginica :38
##
##
##
```

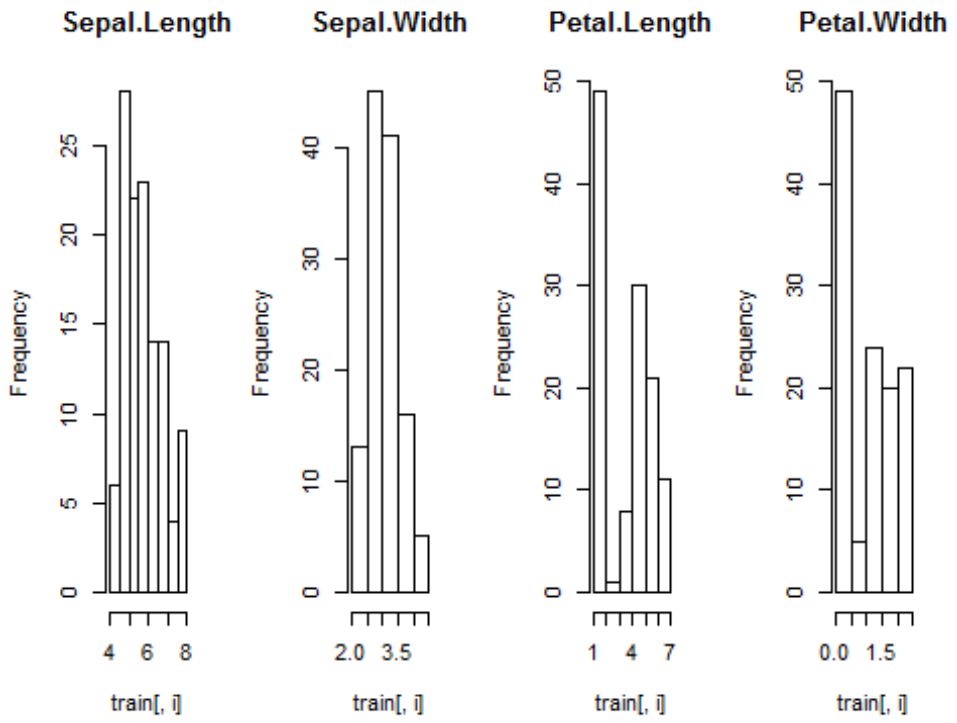
```
summarizeColumns(train)
```

```
##           name    type na     mean      disp median     mad  min  max
## 1 Sepal.Length numeric  0 5.784167 0.9214526   5.70 1.03782  4.3  7.9
## 2  Sepal.Width numeric  0 3.129167 0.4710788   3.10 0.44478  2.0  4.4
## 3 Petal.Length numeric  0 3.571667 1.9043155   4.15 2.81694  1.0  6.9
## 4  Petal.Width numeric  0 1.122500 0.8143113   1.20 1.33434  0.1  2.5
## 5      Species  factor  0       NA 0.5916667     NA      NA 33.0 49.0
##   nlevs
## 1     0
## 2     0
## 3     0
## 4     0
## 5     3
```

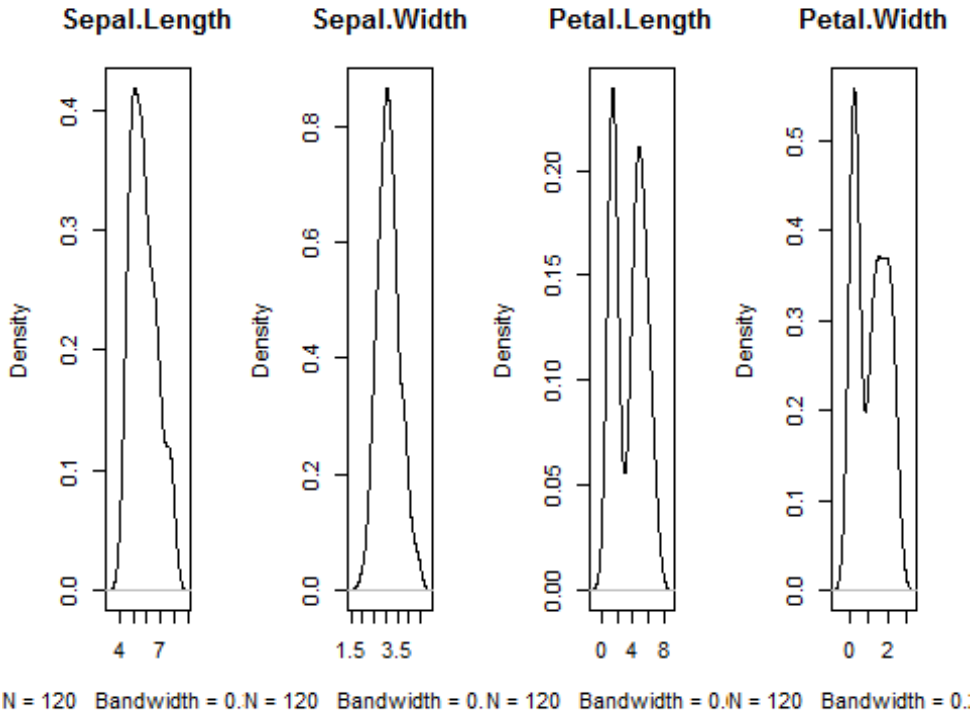
```
prop.table(table(train$Species)) * 100
```

```
##
##     setosa versicolor  virginica
##   40.83333   27.50000   31.66667
```

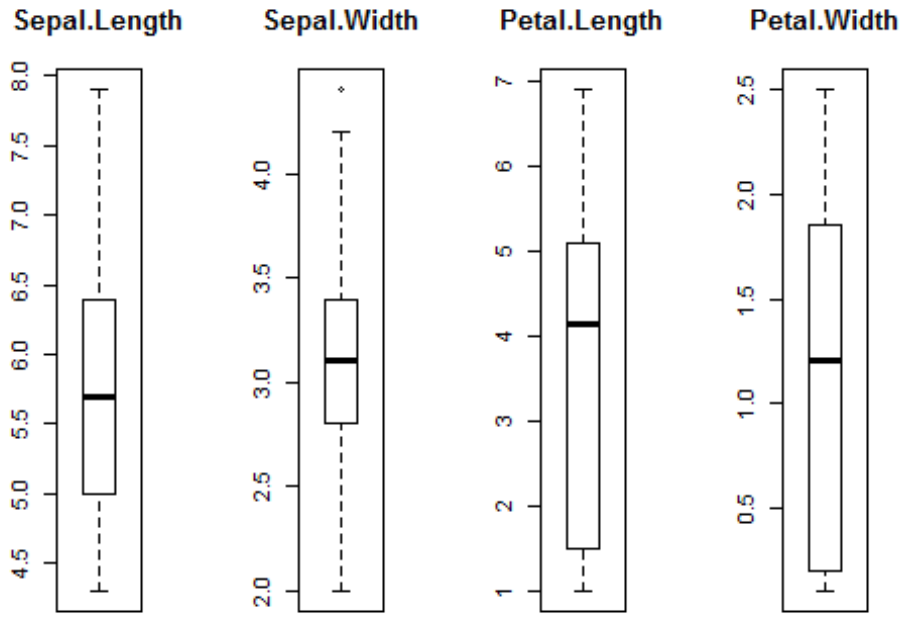
```
# Univariate Visualizations
par(mfrow=c(1,4))
for (i in (1:4)){
  hist(train[,i], main=names(train)[i])
}
```



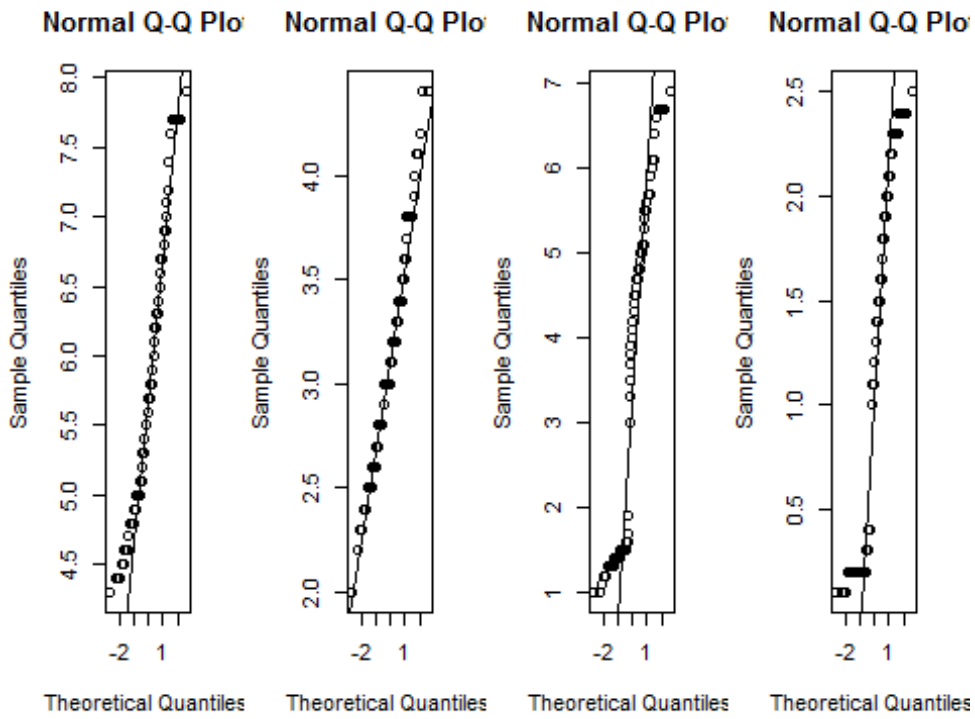

```
for (i in (1:4)){
  plot(density(train[,i]), main=names(train)[i])
}
```



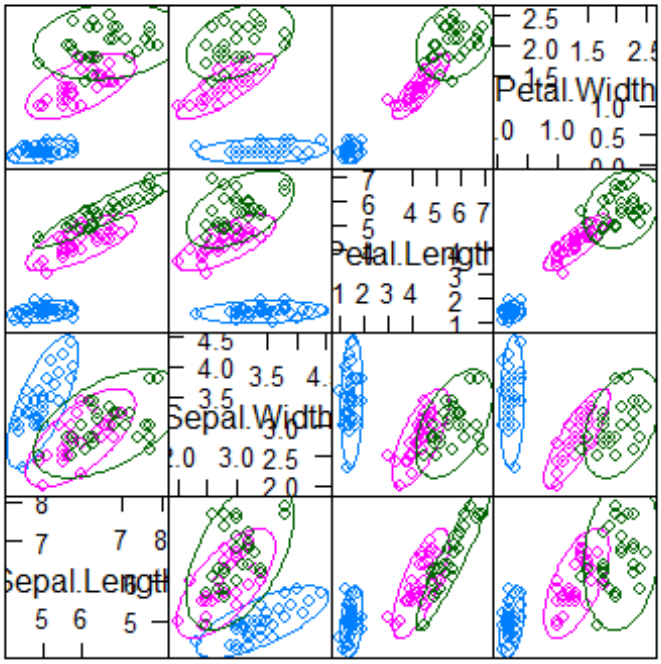
```
for (i in (1:4)){
  boxplot(train[,i], main=names(train)[i])
}
```




```
for (i in (1:4)) {
  qqnorm(train[,i])
  qqline(train[,i], main=names(train)[i])
}
```



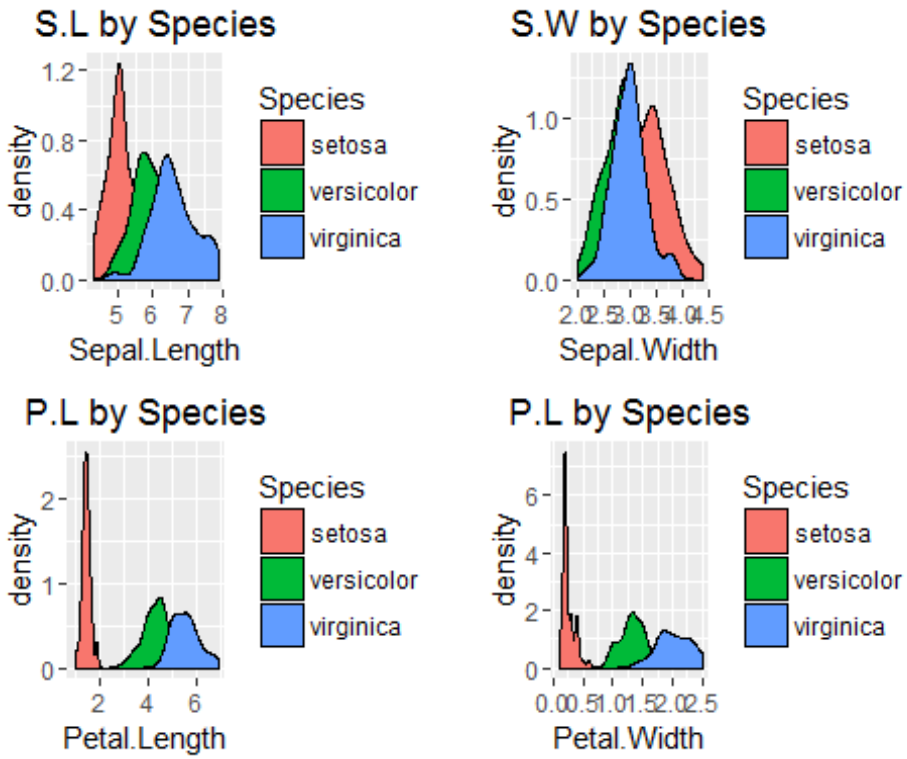

```
# Bi/Multi Variate Visualizations
#plot(train[sapply(train,is.numeric)], main = 'Iris_train Data', pch = 21, bg
= c('red','yellow', 'blue'))
featurePlot(train[,1:4], train[,5], plot='ellipse')
```



Scatter Plot Matrix

```
cor <- cor(train[,1:4], method='pearson')
cor

##              Sepal.Length  Sepal.Width Petal.Length Petal.Width
## Sepal.Length  1.000000000 -0.005315702    0.8751160   0.8107456
## Sepal.Width  -0.005315702  1.000000000   -0.3323643  -0.3035939
## Petal.Length  0.875116046 -0.332364299    1.0000000   0.9640312
## Petal.Width   0.810745650 -0.303593894    0.9640312   1.0000000

corrplot(cor, method='circle', type='lower')

SL_b <- ggplot(iris,aes(Species, Sepal.Length,fill=Species)) + geom_boxplot()
+ labs(title='S.L Vs Species')
SW_b <- ggplot(iris,aes(Species, Sepal.Width,fill=Species)) + geom_boxplot()
+ labs(title='S.W Vs Species')
PL_b <- ggplot(iris,aes(Species, Petal.Length,fill=Species)) + geom_boxplot()
+ labs(title='P.L Vs Species')
PW_b <- ggplot(iris,aes(Species, Petal.Width,fill=Species)) + geom_boxplot()
+ labs(title='P.L Vs Species')

grid.arrange(SL_b,SW_b,PL_b,PW_b, nrow=2)

SL_d <- ggplot(iris,aes(Sepal.Length, ..density.., fill=Species)) +
geom_density() + labs(title='S.L by Species')
SW_d <- ggplot(iris,aes(Sepal.Width, ..density.., fill=Species)) +
geom_density() + labs(title='S.W by Species')
PL_d <- ggplot(iris,aes(Petal.Length, ..density.., fill=Species)) +
```

```
geom_density() + labs(title='P.L by Species')
PW_d <- ggplot(iris,aes(Petal.Width, ..density.., fill=Species)) +
geom_density() + labs(title='P.L by Species')

grid.arrange(SL_d,SW_d,PL_d,PW_d, nrow=2)
```



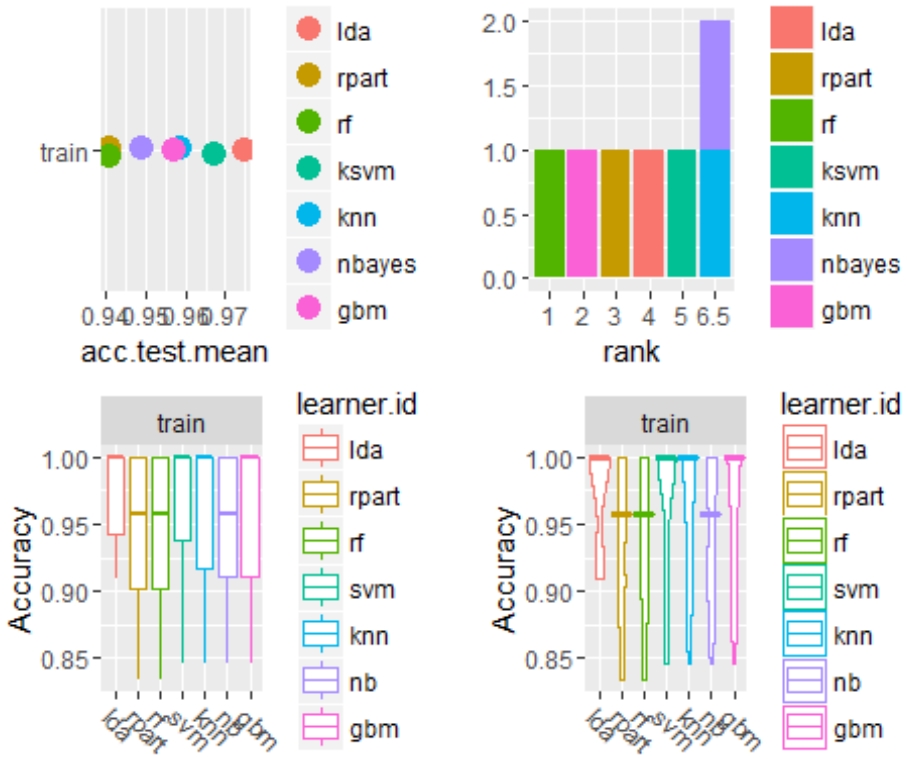

```
#      Here the data can be used as such

#listLearners()

lrns <- list(
  makeLearner('classif.lda', id='lda'),
  makeLearner('classif.rpart',id='rpart'),
  makeLearner('classif.randomForest', id='rf'),
  makeLearner('classif.ksvm',id='svm'),
  makeLearner('classif.knn', id='knn'),
  makeLearner('classif.naiveBayes', id='nb'),
#  makeLearner('classif.nnnet', id='nb'),
  makeLearner('classif.gbm', id='gbm')
)

set.seed(3)
rdesc <- makeResampleDesc(method='CV', iter=10, stratify=TRUE)
bmr <- benchmark(lrns,trainTask,rdesc,measures = acc)

## Task: train, Learner: lda
```

```
## [Resample] cross-validation iter: 1
## [Resample] cross-validation iter: 2
## [Resample] cross-validation iter: 3
## [Resample] cross-validation iter: 4
## [Resample] cross-validation iter: 5
## [Resample] cross-validation iter: 6
## [Resample] cross-validation iter: 7
## [Resample] cross-validation iter: 8
## [Resample] cross-validation iter: 9
## [Resample] cross-validation iter: 10
## [Resample] Result: acc.test.mean=0.975
## Task: train, Learner: rpart
## [Resample] cross-validation iter: 1
## [Resample] cross-validation iter: 2
## [Resample] cross-validation iter: 3
## [Resample] cross-validation iter: 4
## [Resample] cross-validation iter: 5
## [Resample] cross-validation iter: 6
## [Resample] cross-validation iter: 7
## [Resample] cross-validation iter: 8
## [Resample] cross-validation iter: 9
## [Resample] cross-validation iter: 10
## [Resample] Result: acc.test.mean=0.941
## Task: train, Learner: rf
## [Resample] cross-validation iter: 1
## [Resample] cross-validation iter: 2
## [Resample] cross-validation iter: 3
## [Resample] cross-validation iter: 4
```

```
## [Resample] cross-validation iter: 5
## [Resample] cross-validation iter: 6
## [Resample] cross-validation iter: 7
## [Resample] cross-validation iter: 8
## [Resample] cross-validation iter: 9
## [Resample] cross-validation iter: 10
## [Resample] Result: acc.test.mean=0.941
## Task: train, Learner: svm
## [Resample] cross-validation iter: 1
## [Resample] cross-validation iter: 2
## [Resample] cross-validation iter: 3
## [Resample] cross-validation iter: 4
## [Resample] cross-validation iter: 5
## [Resample] cross-validation iter: 6
## [Resample] cross-validation iter: 7
## [Resample] cross-validation iter: 8
## [Resample] cross-validation iter: 9
## [Resample] cross-validation iter: 10
## [Resample] Result: acc.test.mean=0.967
## Task: train, Learner: knn
## [Resample] cross-validation iter: 1
## [Resample] cross-validation iter: 2
## [Resample] cross-validation iter: 3
## [Resample] cross-validation iter: 4
## [Resample] cross-validation iter: 5
## [Resample] cross-validation iter: 6
## [Resample] cross-validation iter: 7
## [Resample] cross-validation iter: 8
```

```
## [Resample] cross-validation iter: 9
## [Resample] cross-validation iter: 10
## [Resample] Result: acc.test.mean=0.959
## Task: train, Learner: nb
## [Resample] cross-validation iter: 1
## [Resample] cross-validation iter: 2
## [Resample] cross-validation iter: 3
## [Resample] cross-validation iter: 4
## [Resample] cross-validation iter: 5
## [Resample] cross-validation iter: 6
## [Resample] cross-validation iter: 7
## [Resample] cross-validation iter: 8
## [Resample] cross-validation iter: 9
## [Resample] cross-validation iter: 10
## [Resample] Result: acc.test.mean=0.949
## Task: train, Learner: gbm
## [Resample] cross-validation iter: 1
## Distribution not specified, assuming multinomial ...
## [Resample] cross-validation iter: 2
## Distribution not specified, assuming multinomial ...
## [Resample] cross-validation iter: 3
## Distribution not specified, assuming multinomial ...
## [Resample] cross-validation iter: 4
## Distribution not specified, assuming multinomial ...
## [Resample] cross-validation iter: 5
## Distribution not specified, assuming multinomial ...
## [Resample] cross-validation iter: 6
## Distribution not specified, assuming multinomial ...
```

```
## [Resample] cross-validation iter: 7

## Distribution not specified, assuming multinomial ...

## [Resample] cross-validation iter: 8

## Distribution not specified, assuming multinomial ...

## [Resample] cross-validation iter: 9

## Distribution not specified, assuming multinomial ...

## [Resample] cross-validation iter: 10

## Distribution not specified, assuming multinomial ...

## [Resample] Result: acc.test.mean=0.957

bmr

##   task.id learner.id acc.test.mean
## 1   train        lda     0.9748834
## 2   train      rpart     0.9405245
## 3   train         rf     0.9405245
## 4   train        svm     0.9671911
## 5   train        knn     0.9588578
## 6   train         nb     0.9488578
## 7   train        gbm     0.9571911

sum <- plotBMRSummary(bmr, measure=acc)
bar <- plotBMRRanksAsBarChart(bmr,measure=acc)
box <- plotBMRBoxplots(bmr, measure=acc) + aes(color=learner.id)
viol <- plotBMRBoxplots(bmr, measure=acc, style='violin') +
aes(color=learner.id)

grid.arrange(sum,bar,box,viol, nrow=2)
```



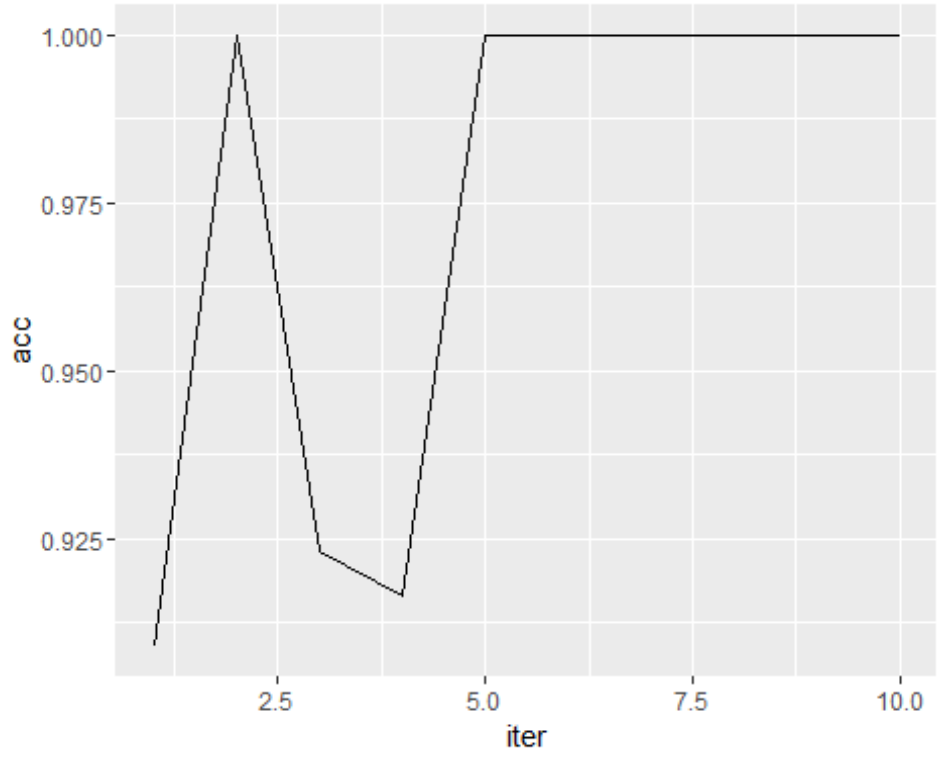

```
set.seed(3)
lda <- resample(learner='classif.lda', task=trainTask, resampling=rdesc,
                measures=list(acc,mmce))

## [Resample] cross-validation iter: 1

## [Resample] cross-validation iter: 2

## [Resample] cross-validation iter: 3

## [Resample] cross-validation iter: 4

## [Resample] cross-validation iter: 5

## [Resample] cross-validation iter: 6

## [Resample] cross-validation iter: 7

## [Resample] cross-validation iter: 8

## [Resample] cross-validation iter: 9

## [Resample] cross-validation iter: 10

## [Resample] Result: acc.test.mean=0.975,mmce.test.mean=0.0251

lda
```

```
## Resample Result
## Task: train
## Learner: classif.lda
## acc.aggr: 0.97
## acc.mean: 0.97
## acc.sd: 0.04
## mmce.aggr: 0.03
## mmce.mean: 0.03
## mmce.sd: 0.04
## Runtime: 0.142102
```

```
names(lda)
```

```
##  [1] "learner.id"     "task.id"        "measures.train" "measures.test"
##  [5] "aggr"           "pred"           "models"         "err.msgs"
##  [9] "extract"        "runtime"
```

```
ggplot(lda$measures.test, aes(iter,acc)) + geom_line()
```




```
set.seed(3)
lda_model <- train(learner='classif.lda', task=trainTask)
lda_model
```

```
## Model for learner.id=classif.lda; learner.class=classif.lda
## Trained on: task.id = train; obs = 120; features = 4
## Hyperparameters:
```

```
names(lda_model)
```

```
## [1] "learner"       "learner.model" "task.desc"     "subset"
## [5] "features"      "factor.levels" "time"

lda_model$learner.model

## Call:
## lda(f, data = getTaskData(.task, .subset))
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.4083333  0.2750000  0.3166667
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa         4.985714    3.414286     1.448980   0.2408163
## versicolor     5.948485    2.803030     4.296970   1.3272727
## virginica      6.671053    3.044737     5.678947   2.0815789
##
## Coefficients of linear discriminants:
##                     LD1        LD2
## Sepal.Length  1.053971  0.3187461
## Sepal.Width   1.015332  1.6336139
## Petal.Length -2.324102 -1.4012114
## Petal.Width  -2.954680  3.4185499
##
## Proportion of trace:
##   LD1   LD2
## 0.993 0.007

lda_pred <- predict(lda_model, newdata=test)
names(lda_pred)

## [1] "predict.type" "data"         "threshold"    "task.desc"
## [5] "time"         "error"

getConfMatrix(lda_pred)

##             predicted
## true         setosa versicolor virginica -SUM-
##   setosa         17          0         0     0
##   versicolor      0         21         0     0
##   virginica       0          1        21     1
##   -SUM-           0          1         0     1

performance(lda_pred, measures=list(acc,mmce), task=lda_model)

##        acc       mmce
## 0.98333333 0.01666667
```