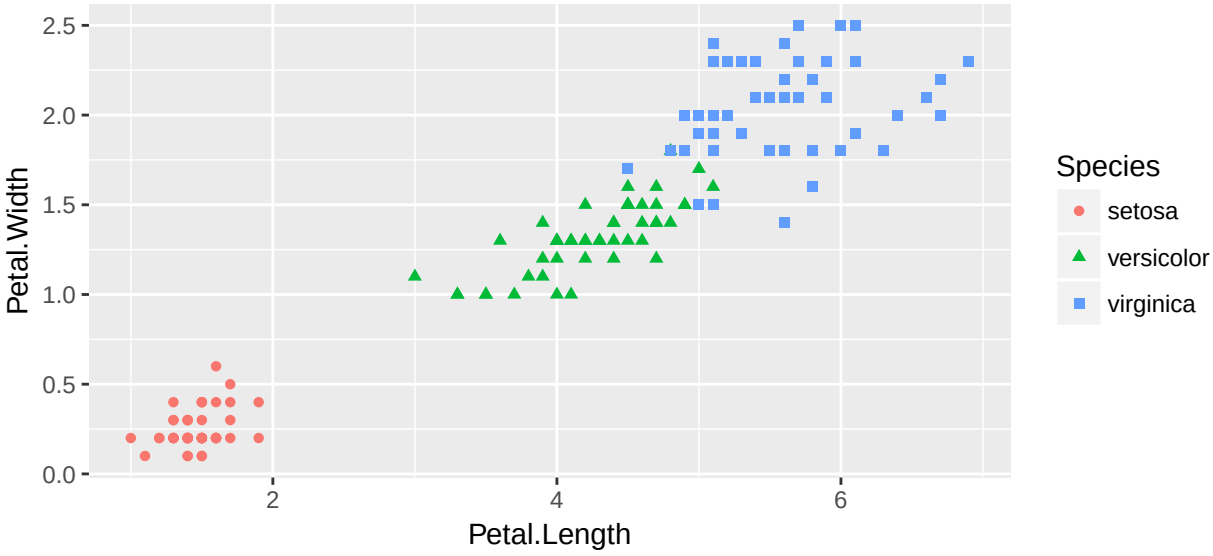
Classification

Data Set – Iris

```
# load data
data(iris)
# this is what it looks like...
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
# ...and this is what it looks like plotted
library(ggplot2)
ggplot(iris, aes(Petal.Length, Petal.Width)) +
    geom_point(aes(color = Species, shape = Species))
```



Logistic Regression

```
library(mlr)
```

```
## Loading required package: ParamHelpers
```

```
# Create task and learner
# Logistic regression can only handle two classes, so subset data accordingly
iris2 = iris[51:150,]
task = makeClassifTask(data = iris2, target = "Species")
```

```
## Warning in makeClassifTask(data = iris2, target = "Species"): Target column
## 'Species' contains empty factor levels
```

```
learner = makeLearner("classif.logreg")

# split the data into train and test set
n = nrow(iris2)
train.set = sample(n, size = 2/3*n)
test.set = setdiff(1:n, train.set)

# train a model
model = train(learner, task, subset = train.set)
model
```

```
## Model for learner.id=classif.logreg; learner.class=classif.logreg
## Trained on: task.id = iris2; obs = 66; features = 4
## Hyperparameters: model=FALSE
```

```
# now predict on the test set
predictions = predict(model, task = task, subset = test.set)
predictions
```

```
## Prediction: 34 observations
## predict.type: response
## threshold:
## time: 0.00
##    id      truth   response
## 51  1 versicolor versicolor
## 55  5 versicolor versicolor
## 57  7 versicolor versicolor
## 63 13 versicolor versicolor
## 64 14 versicolor versicolor
## 67 17 versicolor versicolor
## ... (34 rows, 3 cols)
```

```
# How did we do?
performance(predictions, measures = acc)
```

```
## acc
##   1
```

```
calculateConfusionMatrix(predictions)
```

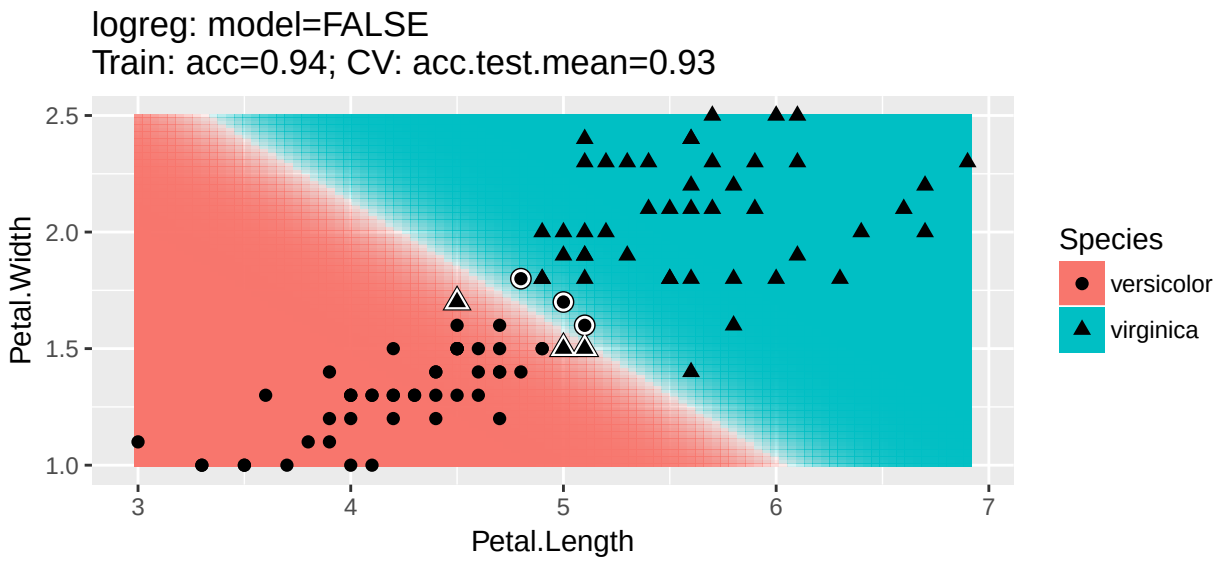
```
##             predicted
## true         versicolor virginica -err.-
##   versicolor         22         0      0
##   virginica           0        12      0
##   -err.-              0         0      0
```

```
# What does the learned model look like?
getLearnerModel(model)
```

```
##
## Call:  stats::glm(formula = f, family = "binomial", data = getTaskData(.task,
##     .subset), weights = .weights, model = FALSE)
##
## Coefficients:
##  (Intercept)  Sepal.Length   Sepal.Width  Petal.Length   Petal.Width
```

```
##      -43.450        -1.013        -6.329         8.578        15.302
##
## Degrees of Freedom: 65 Total (i.e. Null);  61 Residual
## Null Deviance:       89.97
## Residual Deviance: 10.28     AIC: 20.28
```

```
# let's plot the predictions
plotLearnerPrediction(learner, task, measures = acc,
    features = c("Petal.Length", "Petal.Width"))
```

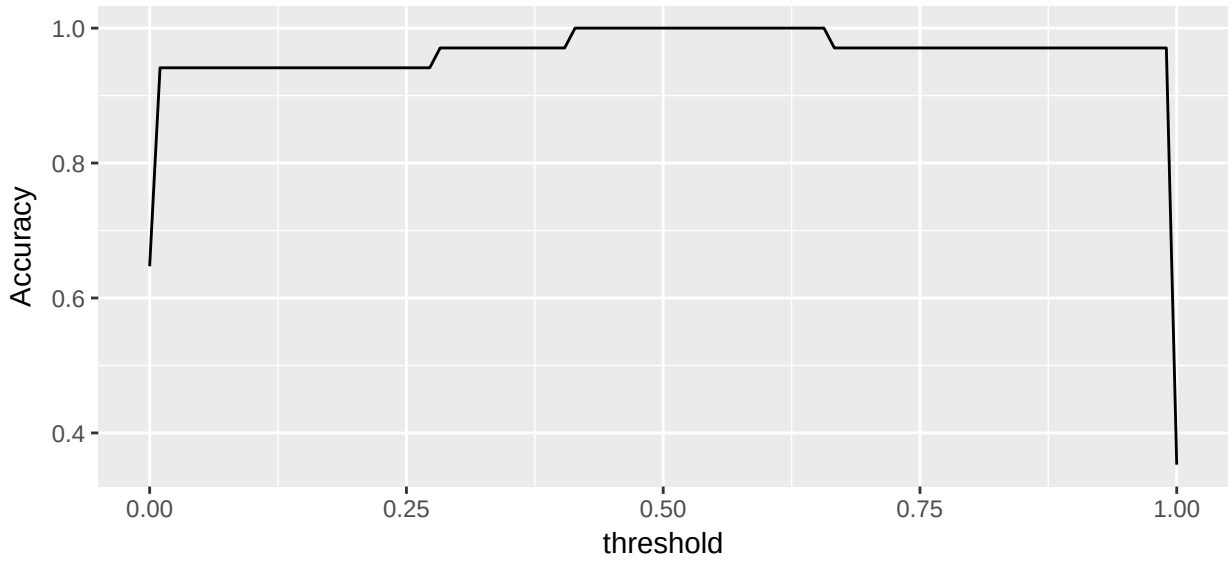


Predicting Probabilities

```
learner = makeLearner("classif.logreg", predict.type = "prob")
model = train(learner, task, subset = train.set)
predictions = predict(model, task = task, subset = test.set)
predictions
```

```
## Prediction: 34 observations
## predict.type: prob
## threshold: versicolor=0.50,virginica=0.50
## time: 0.00
##    id      truth prob.versicolor prob.virginica   response
## 51  1 versicolor       0.9998832   1.168341e-04 versicolor
## 55  5 versicolor       0.9952464   4.753576e-03 versicolor
## 57  7 versicolor       0.9973169   2.683110e-03 versicolor
## 63 13 versicolor       0.9999990   9.777177e-07 versicolor
## 64 14 versicolor       0.9980623   1.937663e-03 versicolor
## 67 17 versicolor       0.9985807   1.419326e-03 versicolor
## ... (34 rows, 5 cols)
```

```
# plot how performance changes if we move the threshold for the classes
d = generateThreshVsPerfData(predictions, measures = acc)
plotThreshVsPerf(d)
```



Using Resampling

```
# mlr can do the partitioning into train and test set automatically
rdesc = makeResampleDesc(method = "Holdout", split = 2/3)
result = resample(learner, task, rdesc, measures = acc, models = TRUE)
```

```
## [Resample] holdout iter 1: acc.test.mean=0.912
## [Resample] Aggr. Result: acc.test.mean=0.912
```

```
# get predictions
getRRPredictions(result)
```

```
## Resampled Prediction for:
## Resample description: holdout with 0.67 split rate.
## Predict: test
## Stratification: FALSE
## predict.type: prob
## threshold: versicolor=0.50,virginica=0.50
## time (mean): 0.00
##   id      truth prob.versicolor prob.virginica   response iter  set
## 1 45 versicolor    9.999951e-01   4.932926e-06 versicolor    1 test
## 2 24 versicolor    9.998081e-01   1.919057e-04 versicolor    1 test
## 3 36 versicolor    9.998266e-01   1.734407e-04 versicolor    1 test
## 4 60  virginica    9.327872e-09   1.000000e+00  virginica    1 test
## 5 71  virginica    1.400234e-06   9.999986e-01  virginica    1 test
## 6 98  virginica    4.519801e-03   9.954802e-01  virginica    1 test
## ... (34 rows, 7 cols)
```

```
# get model
getLearnerModel(result$models[[1]])
```

```
##
## Call:  stats::glm(formula = f, family = "binomial", data = getTaskData(.task,
##     .subset), weights = .weights, model = FALSE)
##
## Coefficients:
```

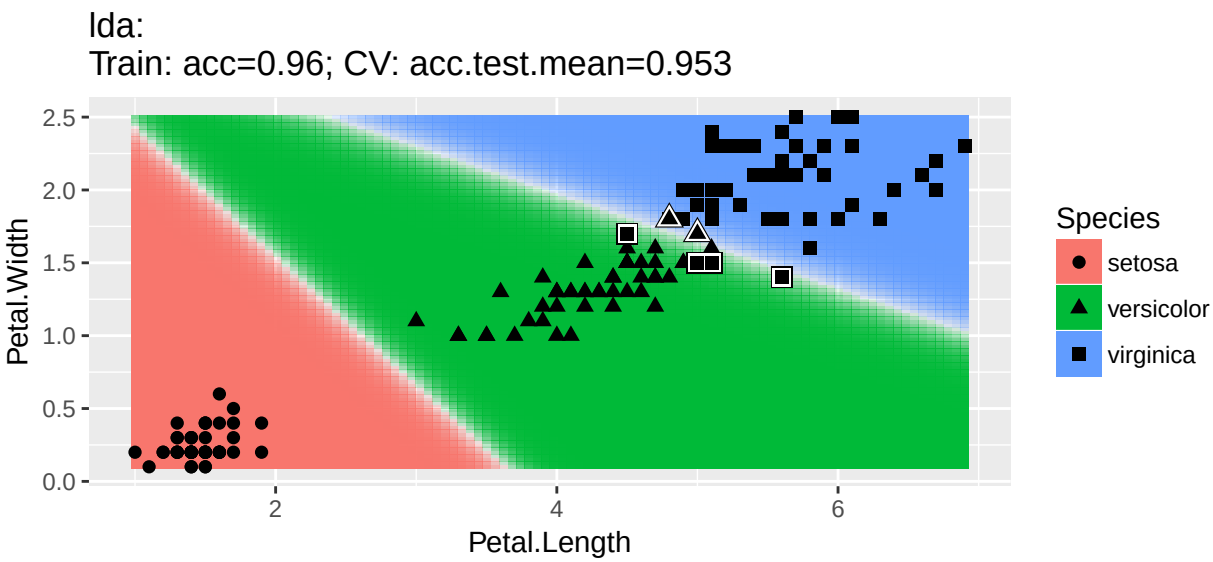
```
##  (Intercept)  Sepal.Length   Sepal.Width  Petal.Length   Petal.Width
##       -58.096        -1.513        -4.236        12.030        11.738
##
## Degrees of Freedom: 65 Total (i.e. Null);  61 Residual
## Null Deviance:       90.95
## Residual Deviance: 8.938     AIC: 18.94
```

Linear Discriminant Analysis

```
task = makeClassifTask(data = iris, target = "Species")
learner = makeLearner("classif.lda")
result = resample(learner, task, rdesc, measures = acc, models = TRUE)
```

```
## [Resample] holdout iter 1: acc.test.mean=   1
## [Resample] Aggr. Result: acc.test.mean=   1
```

```
plotLearnerPrediction(learner, task, measures = acc,
    features = c("Petal.Length", "Petal.Width"))
```




```
getLearnerModel(result$models[[1]])
```

```
## Call:
## lda(f, data = getTaskData(.task, .subset))
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##       0.34       0.34       0.32
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa         4.967647    3.432353     1.458824   0.2411765
## versicolor     5.923529    2.741176     4.282353   1.3235294
## virginica      6.503125    2.981250     5.487500   2.0250000
##
## Coefficients of linear discriminants:
```

```
##                      LD1        LD2
## Sepal.Length  0.7935573  0.2921386
## Sepal.Width   2.1863700 -2.3967032
## Petal.Length -2.6289611  0.6748038
## Petal.Width  -2.2800219 -2.6156049
##
## Proportion of trace:
##    LD1    LD2
## 0.9905 0.0095
```

Support Vector Machines

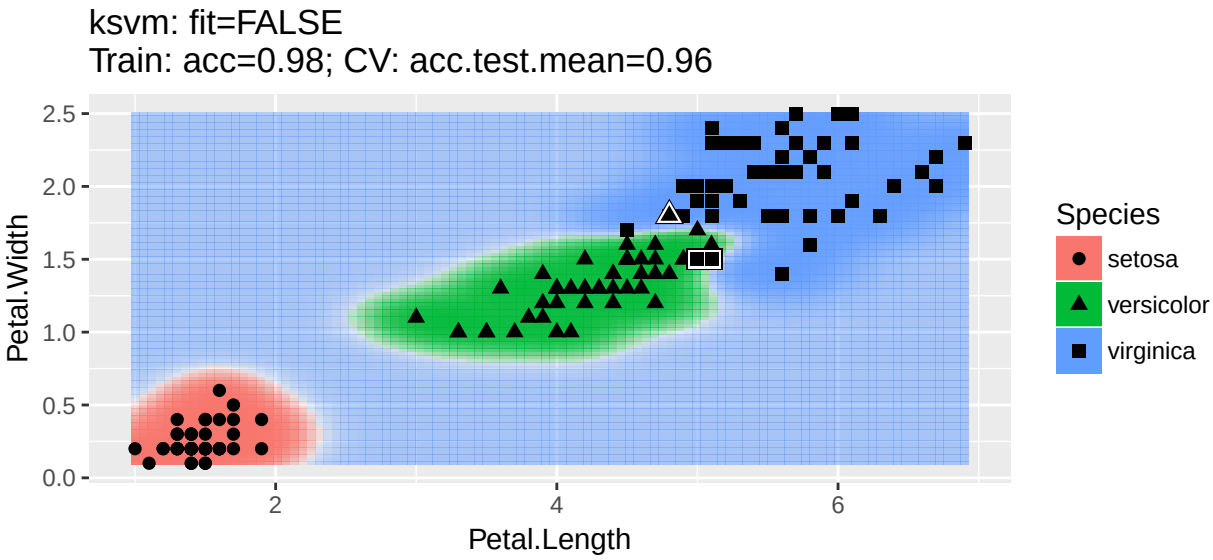
```
learner = makeLearner("classif.ksvm")
result = resample(learner, task, rdesc, measures = acc, models = TRUE)
```

```
## [Resample] holdout iter 1: acc.test.mean=0.96
## [Resample] Aggr. Result: acc.test.mean=0.96
```

```
getLearnerModel(result$models[[1]])
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  1.03364263269777
##
## Number of Support Vectors : 52
##
## Objective Function Value : -4.9243 -5.2502 -17.1656
```

```
plotLearnerPrediction(learner, task, measures = acc,
    features = c("Petal.Length", "Petal.Width"))
```



Different Kernel

```
learner = makeLearner("classif.ksvm", par.vals = list(kernel = "vanilladot"))
result = resample(learner, task, rdesc, measures = acc, models = TRUE)
```

```
## [Resample] holdout iter 1:
```

```
##  Setting default kernel parameters
```

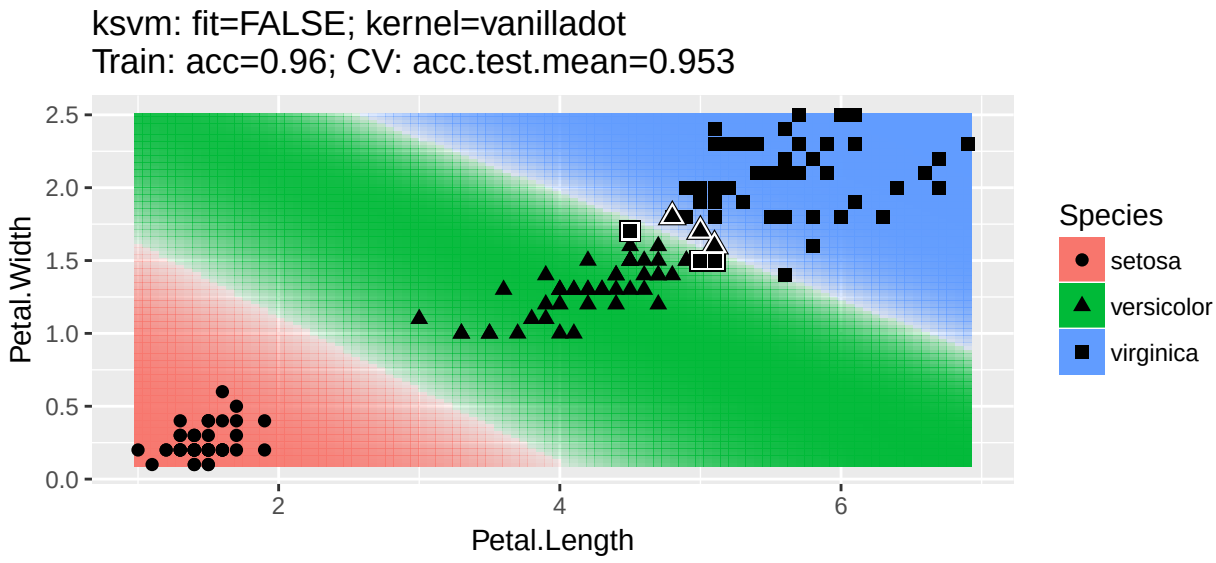
```
## acc.test.mean=0.96
## [Resample] Aggr. Result: acc.test.mean=0.96
```

```
getLearnerModel(result$models[[1]])
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 21
##
## Objective Function Value : -1.0064 -0.3371 -12.347
```

```
plotLearnerPrediction(learner, task, measures = acc,
    features = c("Petal.Length", "Petal.Width"))
```

```
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
```

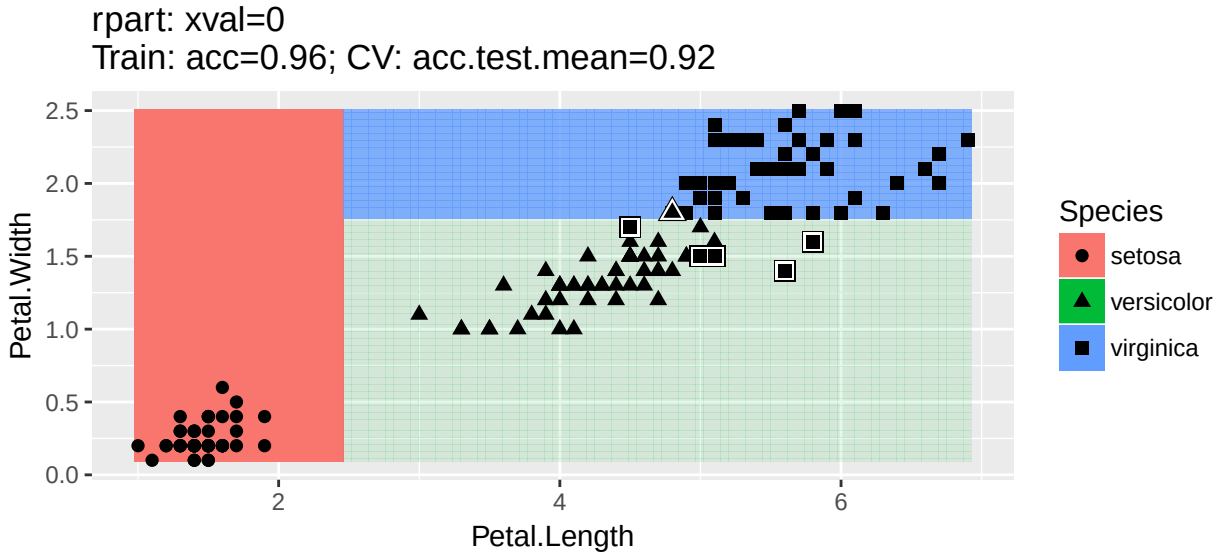


Classification Trees

```
learner = makeLearner("classif.rpart")
result = resample(learner, task, rdesc, measures = acc, models = TRUE)
```

```
## [Resample] holdout iter 1: acc.test.mean=0.94
## [Resample] Aggr. Result: acc.test.mean=0.94
```

```
plotLearnerPrediction(learner, task, measures = acc,
    features = c("Petal.Length", "Petal.Width"))
```




```
getLearnerModel(result$models[[1]])
```

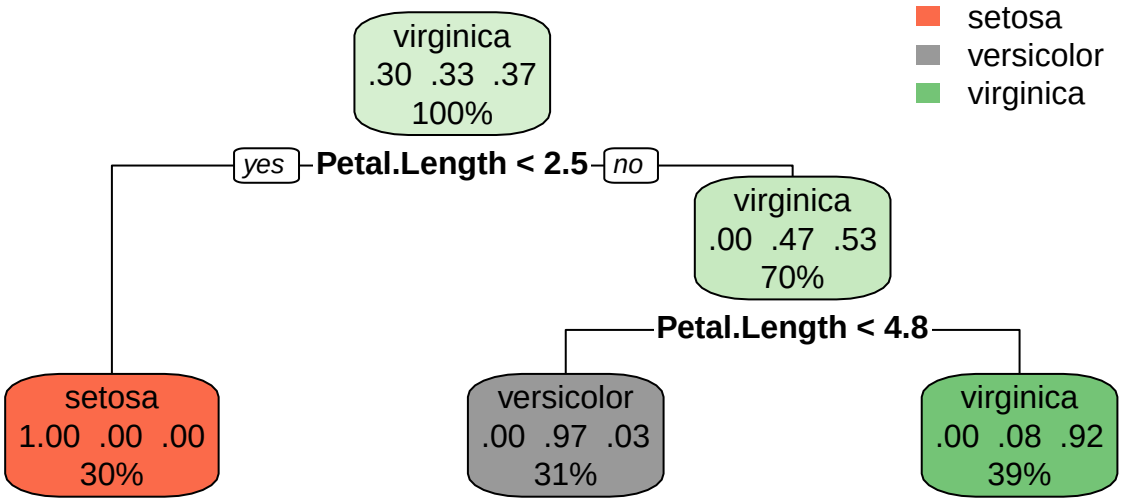
```
## n= 100
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
```

```
##
## 1) root 100 63 virginica (0.30000000 0.33000000 0.37000000)
##   2) Petal.Length< 2.45 30  0 setosa (1.00000000 0.00000000 0.00000000) *
##   3) Petal.Length>=2.45 70 33 virginica (0.00000000 0.47142857 0.52857143)
##     6) Petal.Length< 4.75 31  1 versicolor (0.00000000 0.96774194 0.03225806) *
##     7) Petal.Length>=4.75 39  3 virginica (0.00000000 0.07692308 0.92307692) *
```

```
# this is not part of mlr
library(rpart.plot)
```

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

```
rpart.plot(getLearnerModel(result$models[[1]]))
```



Random Forests

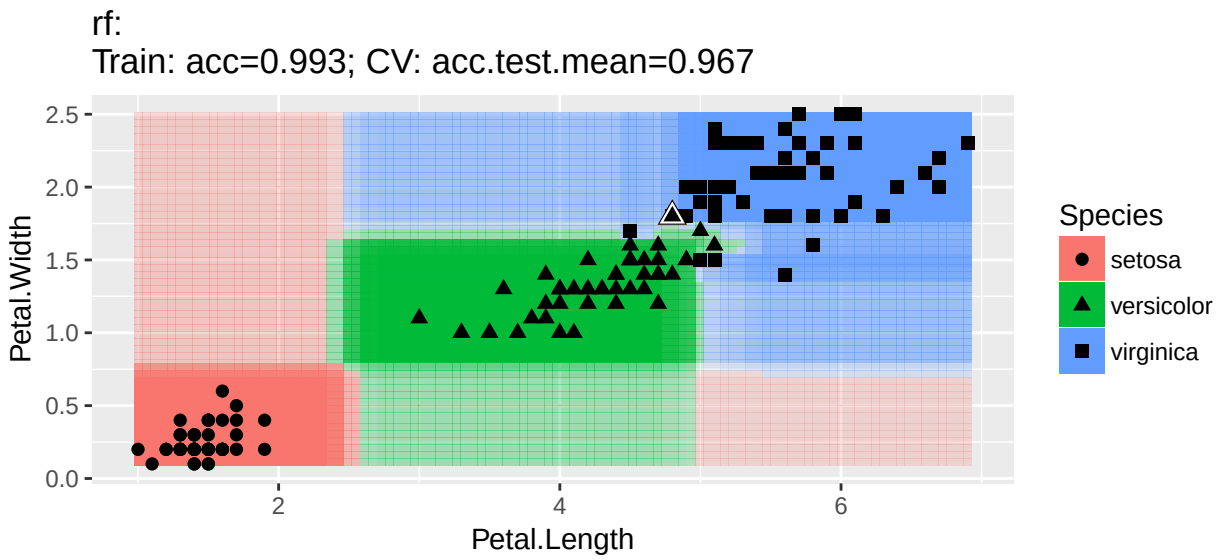
```
learner = makeLearner("classif.randomForest")
result = resample(learner, task, rdesc, measures = acc, models = TRUE)
```

```
## [Resample] holdout iter 1: acc.test.mean=0.92
## [Resample] Aggr. Result: acc.test.mean=0.92
```

```
getLearnerModel(result$models[[1]])
```

```
##
## Call:
##  randomForest(formula = f, data = data, classwt = classwt, cutoff = cutoff)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         30          0         0  0.00000000
## versicolor      0         32         1  0.03030303
## virginica       0          2        35  0.05405405
```

```
plotLearnerPrediction(learner, task, measures = acc,
    features = c("Petal.Length", "Petal.Width"))
```



More

```
listLearners(task)$class
```

```
## Warning in listLearners.character(td$type, union(props, properties), quiet, : The following learners
## classif.hdrda
## Check ?learners to see which packages you need or install mlr with all suggestions.
```

```
##  [1] "classif.bdk"                   "classif.boosting"
##  [3] "classif.C50"                   "classif.cforest"
##  [5] "classif.ctree"                 "classif.cvglmnet"
##  [7] "classif.dbnDNN"                "classif.earth"
##  [9] "classif.evtree"                "classif.extraTrees"
## [11] "classif.featureless"           "classif.fnn"
## [13] "classif.gausspr"               "classif.gbm"
## [15] "classif.geoDA"                 "classif.glmnet"
## [17] "classif.h2o.deeplearning"      "classif.h2o.gbm"
## [19] "classif.h2o.randomForest"      "classif.IBk"
## [21] "classif.J48"                   "classif.JRip"
## [23] "classif.kknn"                  "classif.knn"
## [25] "classif.ksvm"                  "classif.lda"
## [27] "classif.LiblineaRL1L2SVC"      "classif.LiblineaRL1LogReg"
## [29] "classif.LiblineaRL2L1SVC"      "classif.LiblineaRL2LogReg"
## [31] "classif.LiblineaRL2SVC"        "classif.LiblineaRMultiClassSVC"
## [33] "classif.linDA"                 "classif.lssvm"
## [35] "classif.lvq1"                  "classif.mda"
## [37] "classif.mlp"                   "classif.multinom"
## [39] "classif.naiveBayes"            "classif.nnet"
## [41] "classif.nnTrain"               "classif.OneR"
## [43] "classif.PART"                  "classif.qda"
## [45] "classif.quaDA"                 "classif.randomForest"
```

```
## [47] "classif.randomForestSRC"        "classif.ranger"
## [49] "classif.rda"                    "classif.rFerns"
## [51] "classif.rknn"                   "classif.rpart"
## [53] "classif.RRF"                    "classif.rrlda"
## [55] "classif.saeDNN"                 "classif.sda"
## [57] "classif.sparseLDA"              "classif.svm"
## [59] "classif.xgboost"                "classif.xyf"
```

```
listMeasures(task)
```

```
##  [1] "kappa"            "multiclass.brier" "multiclass.aunp"
##  [4] "multiclass.aunu"  "qsr"              "ber"
##  [7] "logloss"          "wkappa"           "timeboth"
## [10] "timepredict"      "acc"              "lsr"
## [13] "featperc"         "multiclass.au1p"  "multiclass.au1u"
## [16] "ssr"              "timetrain"        "mmce"
```