Regression

Data Set – Boston Housing

```
library(mlr)
```

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

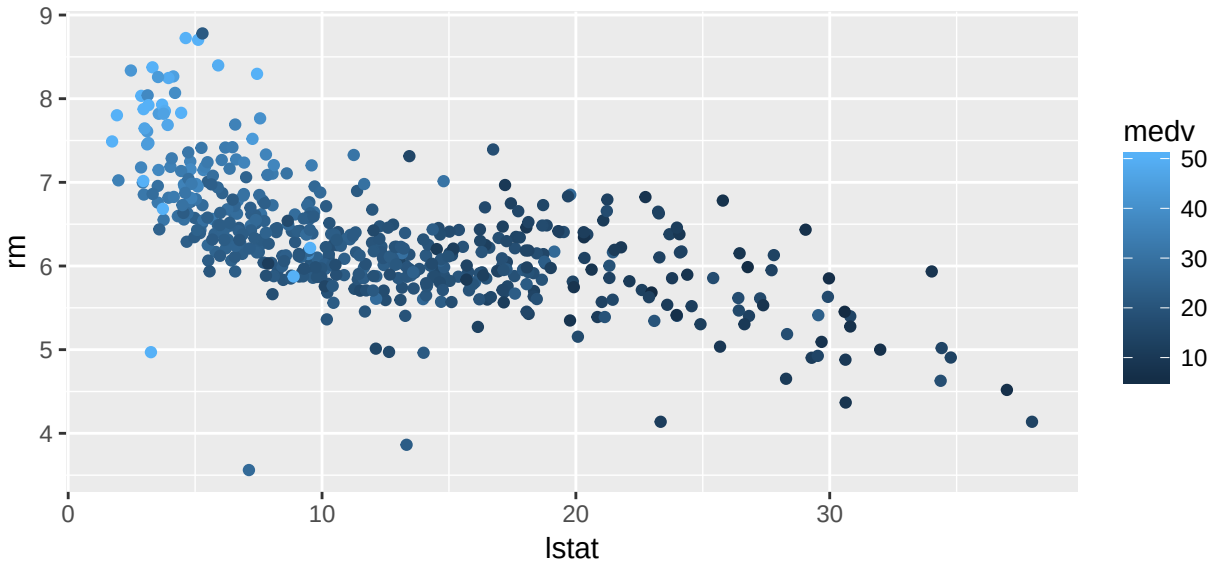
```
# mlr comes with example tasks
bh.task
```

```
## Supervised task: BostonHousing-example
## Type: regr
## Target: medv
## Observations: 506
## Features:
## numerics  factors  ordered
##       12        1        0
## Missings: FALSE
## Has weights: FALSE
## Has blocking: FALSE
```

```
# this is what it looks like...
head(getTaskData(bh.task))
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio      b
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12
##   lstat medv
## 1  4.98 24.0
## 2  9.14 21.6
## 3  4.03 34.7
## 4  2.94 33.4
## 5  5.33 36.2
## 6  5.21 28.7
```

```
# ...and this is what it looks like plotted
library(ggplot2)
ggplot(getTaskData(bh.task), aes(lstat, rm)) +
    geom_point(aes(color = medv))
```



Linear Model

```
learner = makeLearner("regr.lm")
rdesc = makeResampleDesc(method = "Holdout", split = 2/3)
result = resample(learner, bh.task, rdesc, models = TRUE)
```

```
## [Resample] holdout iter 1: mse.test.mean=  24
## [Resample] Aggr. Result: mse.test.mean=  24
```

```
getRRPredictions(result)
```

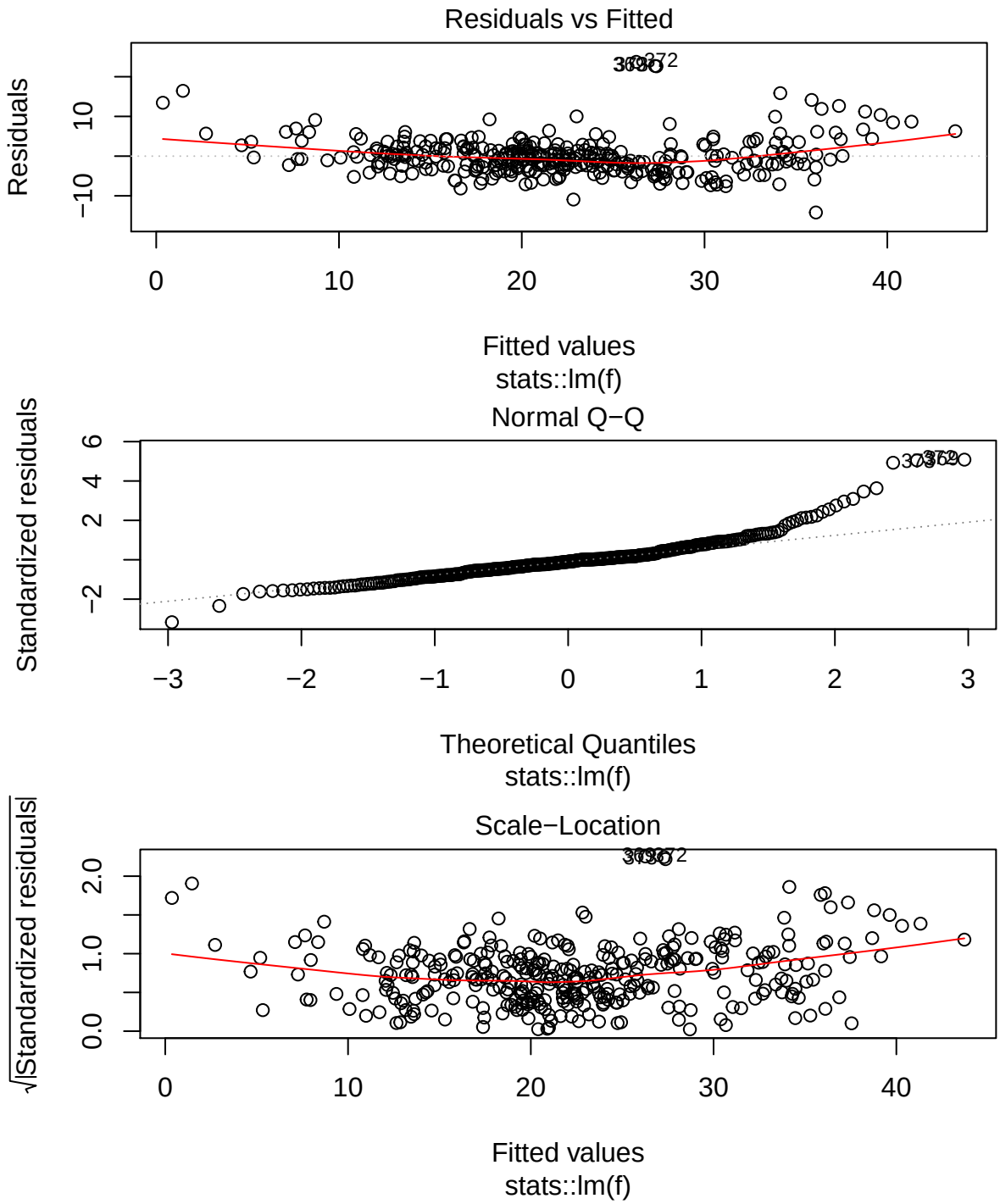
```
## Resampled Prediction for:
## Resample description: holdout with 0.67 split rate.
## Predict: test
## Stratification: FALSE
## predict.type: response
## threshold:
## time (mean): 0.01
##    id truth response iter  set
## 1 195  29.1 31.46434    1 test
## 2 205  50.0 42.39308    1 test
## 3 334  22.2 21.99126    1 test
## 4 498  18.3 19.14167    1 test
## 5 354  30.1 25.74180    1 test
## 6 279  29.1 30.12118    1 test
## ... (169 rows, 5 cols)
```

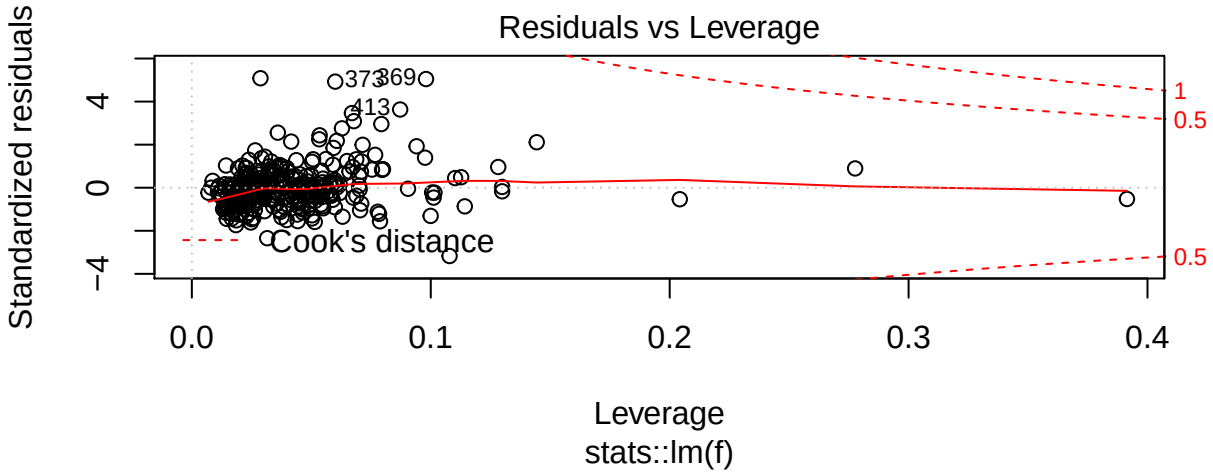
```
getLearnerModel(result$models[[1]])
```

```
##
## Call:
## stats::lm(formula = f, data = d)
##
## Coefficients:
## (Intercept)         crim           zn        indus        chas1
```

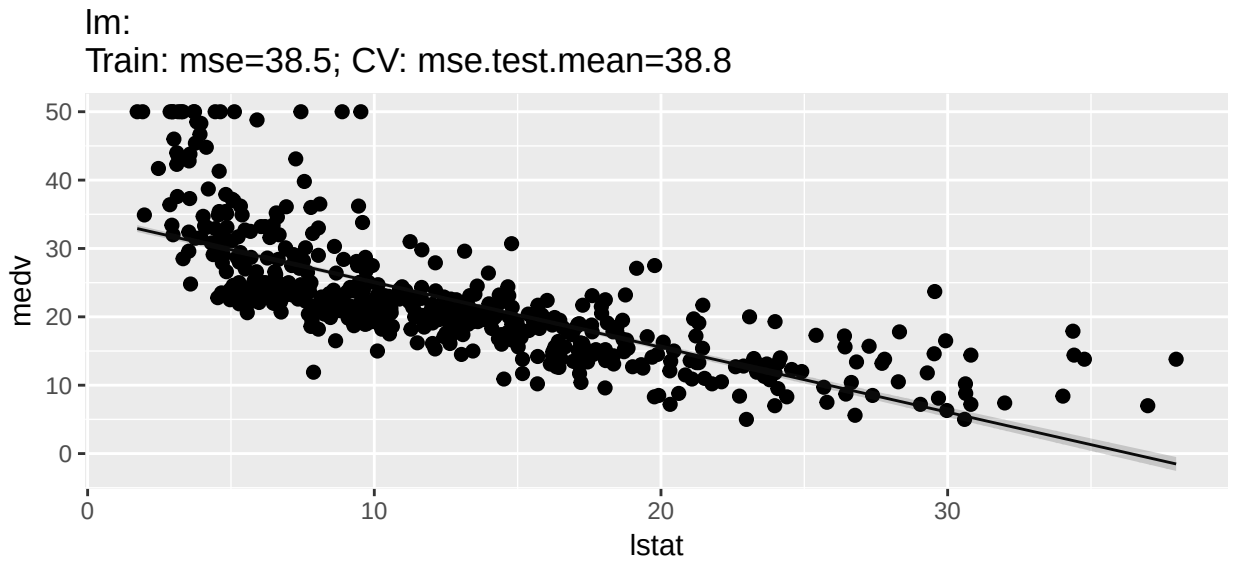
```
##   45.491865    -0.124106     0.049108     0.014981     2.632698
##         nox           rm          age          dis          rad
##  -20.197719     2.769012     0.025593    -1.471776     0.350984
##         tax      ptratio            b        lstat
##   -0.012377    -1.039238     0.009215    -0.653209
```

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

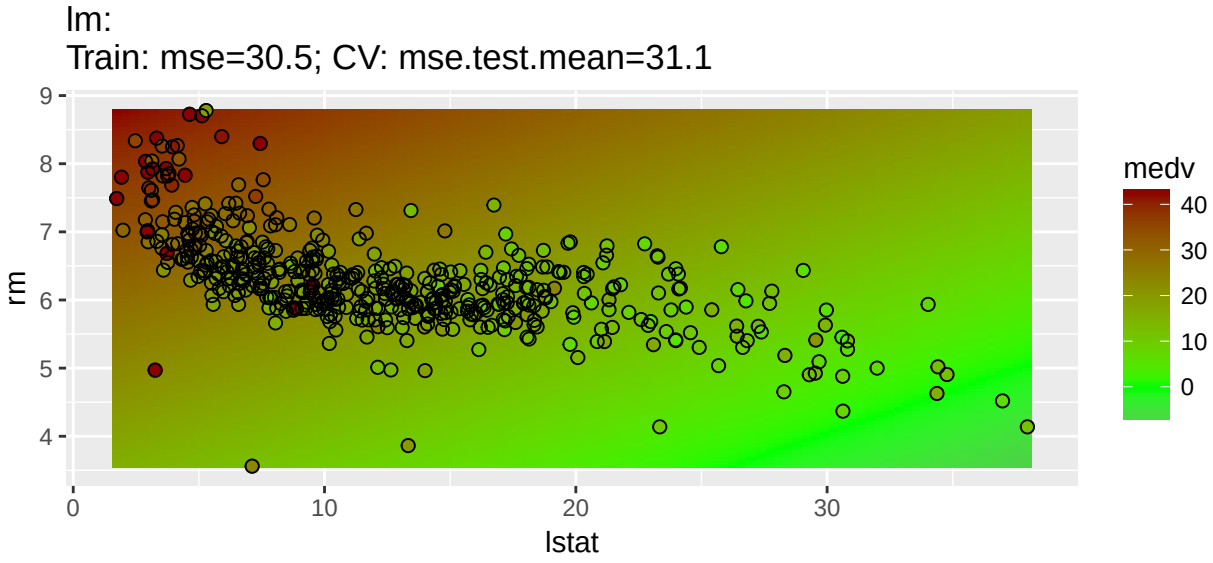





```
plotLearnerPrediction(learner, bh.task, features = c("lstat"))
```




```
plotLearnerPrediction(learner, bh.task, features = c("lstat", "rm"))
```



Regression Splines

```
learner = makeLearner("regr.earth")
result = resample(learner, bh.task, rdesc, models = TRUE)
```

```
## [Resample] holdout iter 1: mse.test.mean=12.9
## [Resample] Aggr. Result: mse.test.mean=12.9
```

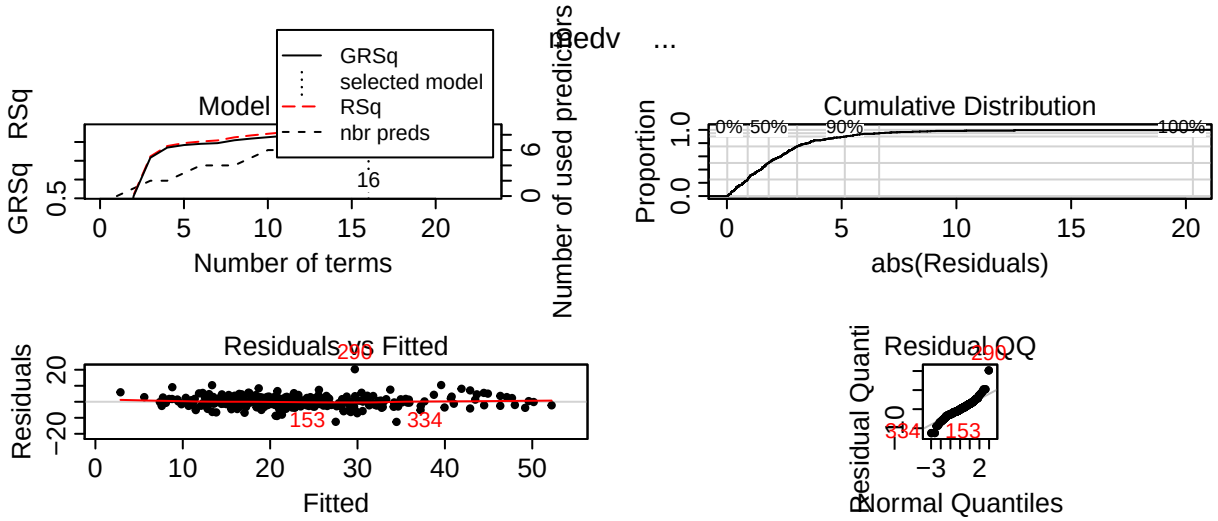
```
getRRPredictions(result)
```

```
## Resampled Prediction for:
## Resample description: holdout with 0.67 split rate.
## Predict: test
## Stratification: FALSE
## predict.type: response
## threshold:
## time (mean): 0.02
##    id truth response iter  set
## 1 294  23.9 23.94781    1 test
## 2 218  28.7 24.68479    1 test
## 3 360  22.6 19.58441    1 test
## 4  84  22.9 22.47124    1 test
## 5 422  14.2 14.55113    1 test
## 6 219  21.5 19.80240    1 test
## ... (169 rows, 5 cols)
```

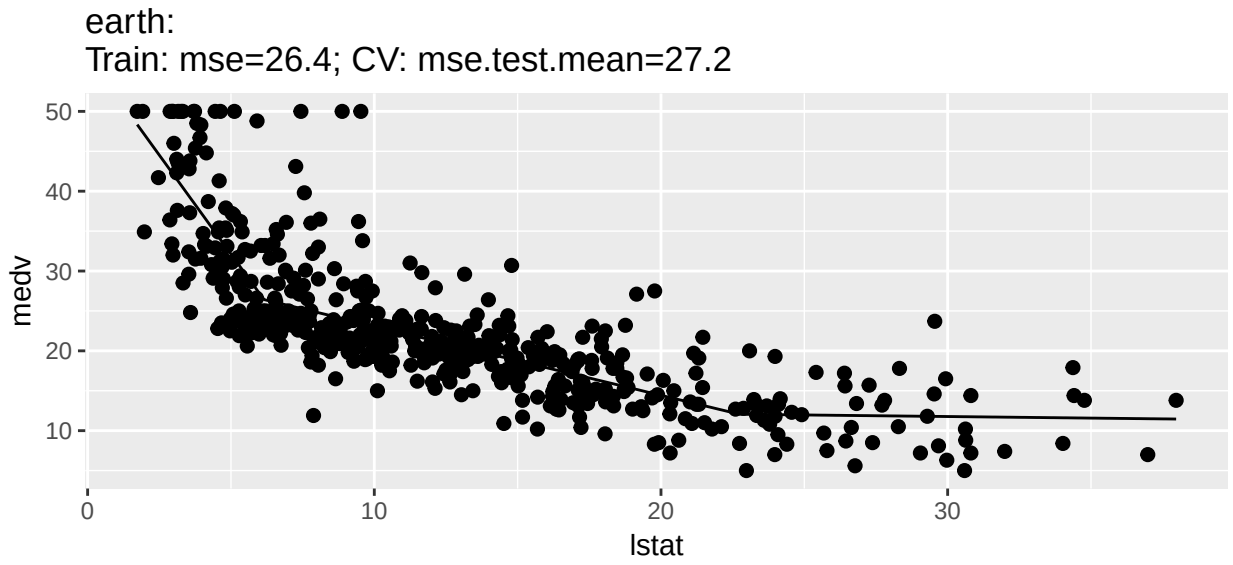
```
getLearnerModel(result$models[[1]])
```

```
## Selected 16 of 23 terms, and 8 of 13 predictors
## Termination condition: Reached nk 27
## Importance: rm, lstat, ptratio, crim, dis, tax, nox, rad, b-unused, ...
## Number of terms at each degree of interaction: 1 15 (additive model)
## GCV 13.33962    RSS 3706.435    GRSq 0.8484319    RSq 0.8742894
```

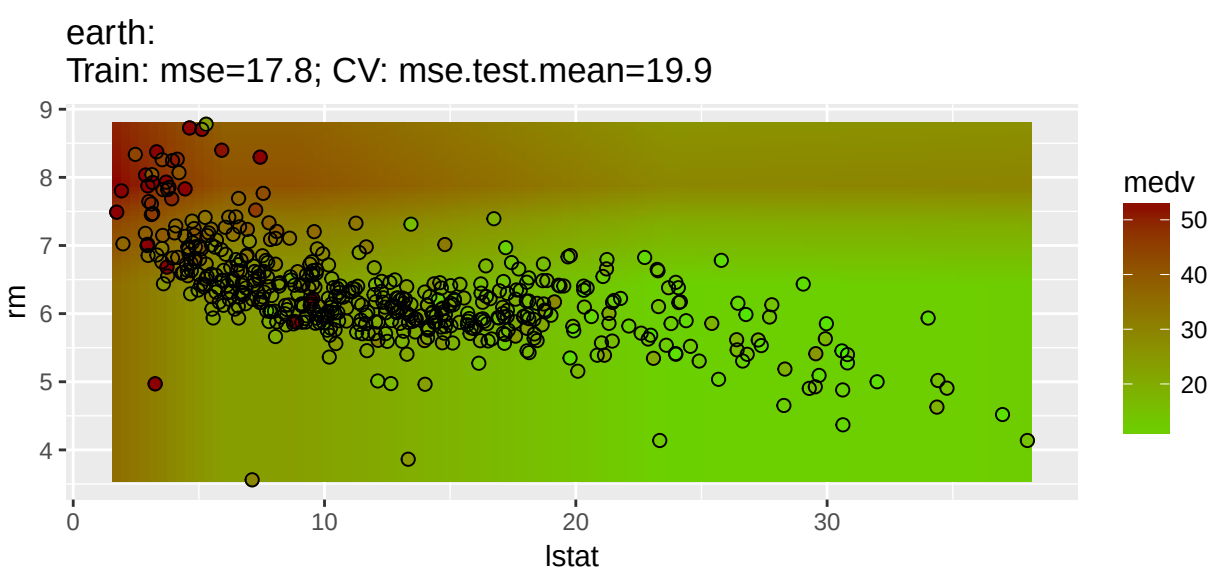
```
plot(getLearnerModel(result$models[[1]]))
```




```
plotLearnerPrediction(learner, bh.task, features = c("lstat"))
```




```
plotLearnerPrediction(learner, bh.task, features = c("lstat", "rm"))
```



Boosting

```
learner = makeLearner("regr.blackboost")
result = resample(learner, bh.task, rdesc, models = TRUE)
```

```
## [Resample] holdout iter 1: mse.test.mean=9.98
## [Resample] Aggr. Result: mse.test.mean=9.98
```

```
getRRPredictions(result)
```

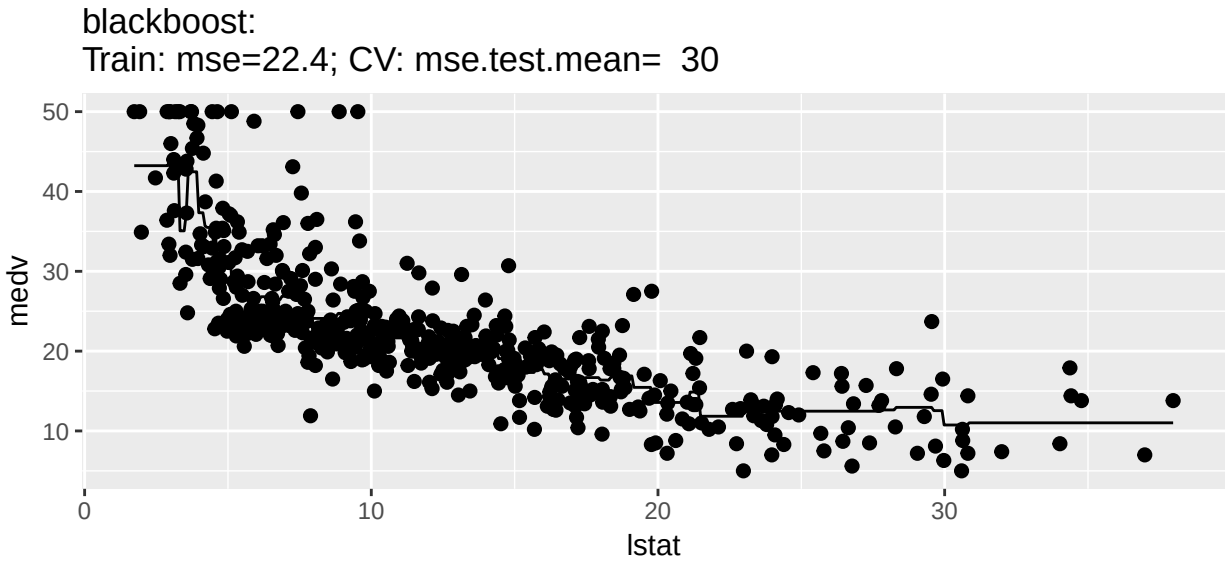
```
## Resampled Prediction for:
## Resample description: holdout with 0.67 split rate.
## Predict: test
## Stratification: FALSE
## predict.type: response
```

```
## threshold:
## time (mean): 0.02
##    id truth response iter  set
## 1 434  14.3 15.12590    1 test
## 2 367  21.9 18.30126    1 test
## 3   8  27.1 15.84473    1 test
## 4 350  26.6 26.11271    1 test
## 5  22  19.6 18.25295    1 test
## 6  21  13.6 14.00064    1 test
## ... (169 rows, 5 cols)
```

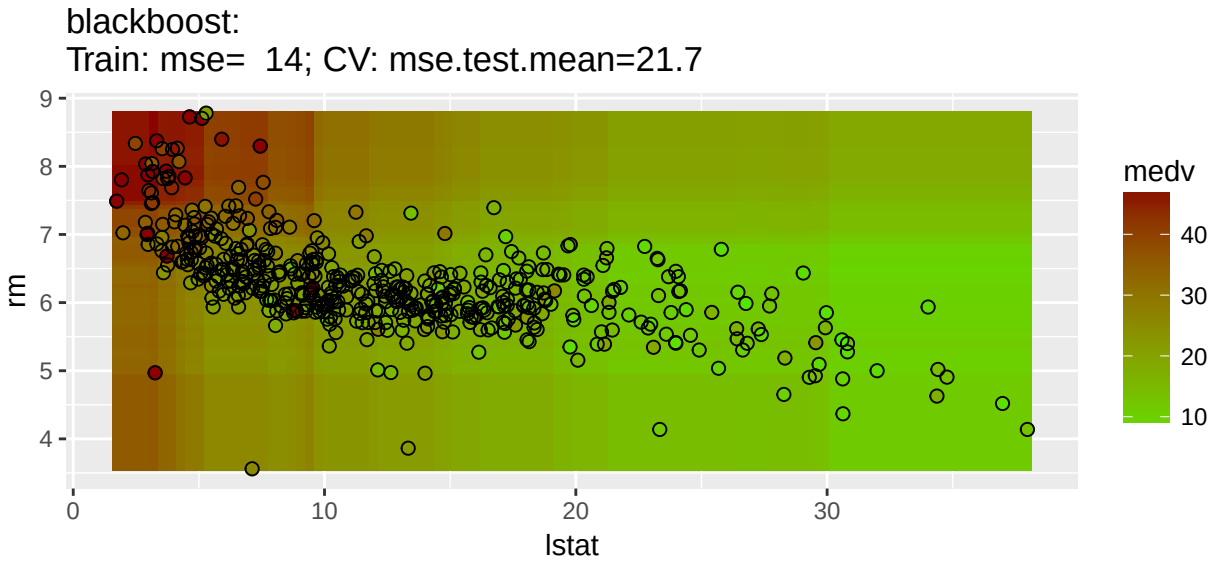
```
getLearnerModel(result$models[[1]])
```

```
##
##   Model-based Boosting
##
## Call:
## mboost::blackboost(formula = f, data = getTaskData(.task, .subset),     tree_controls = tc, control =
##
##
##   Squared Error (Regression)
##
## Loss function: (y - f)^2
##
##
## Number of boosting iterations: mstop = 100
## Step size:  0.1
## Offset:  23.01009
## Number of baselearners:  1
```

```
plotLearnerPrediction(learner, bh.task, features = c("lstat"))
```




```
plotLearnerPrediction(learner, bh.task, features = c("lstat", "rm"))
```



Support Vector Machine

```
learner = makeLearner("regr.ksvm")
result = resample(learner, bh.task, rdesc, models = TRUE)
```

```
## [Resample] holdout iter 1: mse.test.mean=  15
## [Resample] Aggr. Result: mse.test.mean=  15
```

```
getRRPredictions(result)
```

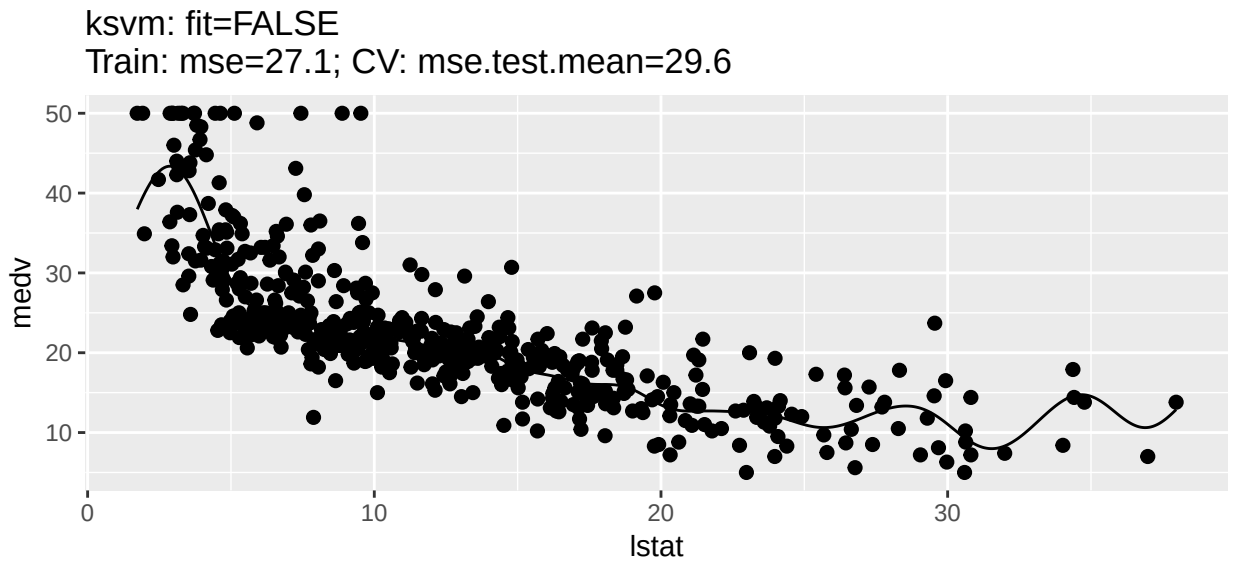
```
## Resampled Prediction for:
## Resample description: holdout with 0.67 split rate.
## Predict: test
## Stratification: FALSE
## predict.type: response
## threshold:
## time (mean): 0.01
##    id truth response iter  set
## 1 240  23.3 26.48909    1 test
## 2 161  27.0 25.71215    1 test
## 3 334  22.2 22.91698    1 test
## 4 374  13.8 10.70924    1 test
## 5 470  20.1 20.67146    1 test
## 6 124  17.3 17.22868    1 test
## ... (169 rows, 5 cols)
```

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

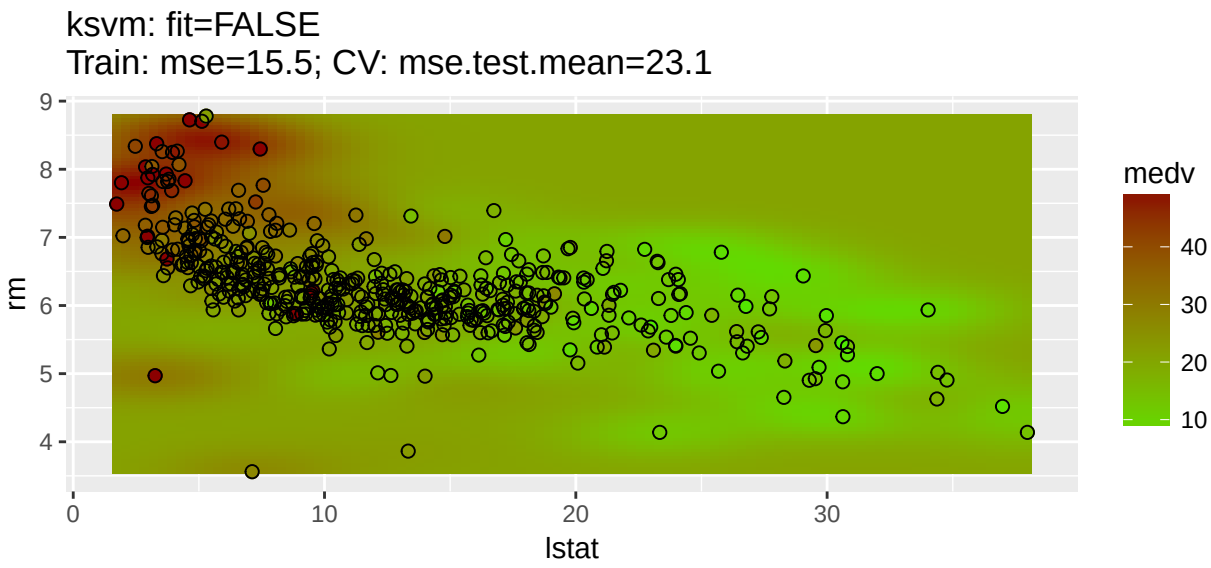
```
## Support Vector Machine object of class "ksvm"
##
## SV type: eps-svr  (regression)
##  parameter : epsilon = 0.1  cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.112559678594154
```

```
##
## Number of Support Vectors : 226
##
## Objective Function Value : -57.374
```

```
plotLearnerPrediction(learner, bh.task, features = c("lstat"))
```




```
plotLearnerPrediction(learner, bh.task, features = c("lstat", "rm"))
```



Regression Forests

```
learner = makeLearner("regr.randomForest")
result = resample(learner, bh.task, rdesc, models = TRUE)
```

```
## [Resample] holdout iter 1: mse.test.mean=10.4
## [Resample] Aggr. Result: mse.test.mean=10.4
```

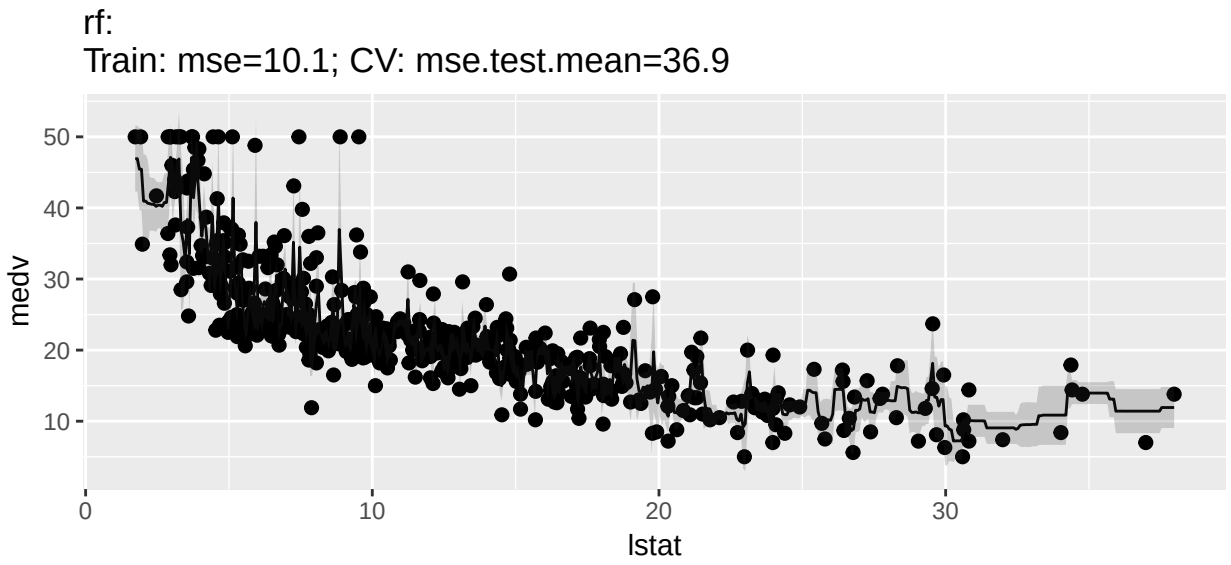
```
getRRPredictions(result)
```

```
## Resampled Prediction for:
## Resample description: holdout with 0.67 split rate.
## Predict: test
## Stratification: FALSE
## predict.type: response
## threshold:
## time (mean): 0.01
##    id truth response iter  set
## 1 130  14.3 16.16106    1 test
## 2 151  21.5 19.31184    1 test
## 3 158  41.3 33.57724    1 test
## 4 380  10.2 10.30709    1 test
## 5 350  26.6 26.34453    1 test
## 6 299  22.5 27.42540    1 test
## ... (169 rows, 5 cols)
```

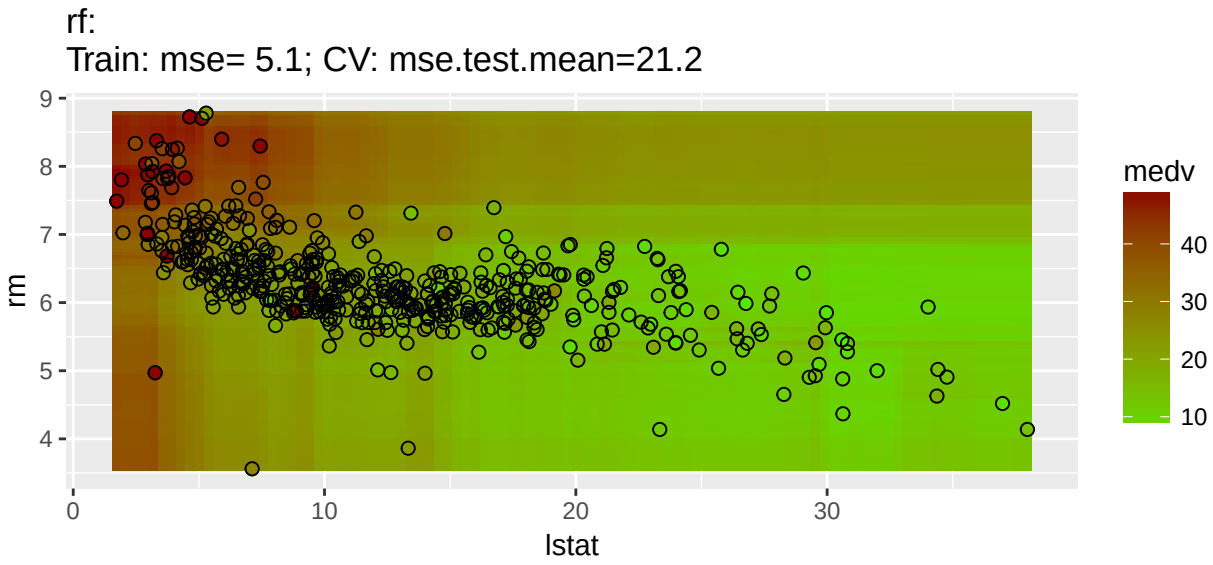
```
getLearnerModel(result$models[[1]])
```

```
##
## Call:
##  randomForest(x = data[["data"]], y = data[["target"]], keep.inbag = if (is.null(keep.inbag)) TRUE el
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 12.64845
##                     % Var explained: 84.62
```

```
plotLearnerPrediction(learner, bh.task, features = c("lstat"))
```




```
plotLearnerPrediction(learner, bh.task, features = c("lstat", "rm"))
```




```
imp = getFeatureImportance(result$models[[1]])
imp
```

```
## FeatureImportance:
## Task: BostonHousing-example
##
## Learner: regr.randomForest
## Measure: NA
## Contrast: NA
## Aggregation: function (x)  x
## Replace: NA
## Number of Monte-Carlo iterations: NA
## Local: FALSE
##       crim       zn   indus     chas      nox       rm     age      dis
## 1 1384.693 125.2321 1637.49 66.81436 1775.168 7710.023 859.009 1632.915
##        rad      tax  ptratio        b    lstat
## 1 198.9478 1046.645 1556.895 514.6052 8569.842
```

```
sort(imp$res)
```

```
##       chas       zn      rad        b     age      tax     crim  ptratio
## 1 66.81436 125.2321 198.9478 514.6052 859.009 1046.645 1384.693 1556.895
##        dis    indus      nox       rm    lstat
## 1 1632.915 1637.49 1775.168 7710.023 8569.842
```

More

```
getLearnerProperties(learner)
```

```
## [1] "numerics" "factors"  "ordered"  "se"       "oobpreds" "featimp"
```

```
getTaskDesc(bh.task)
```

```
## $id
## [1] "BostonHousing-example"
##
```

```
## $type
## [1] "regr"
##
## $target
## [1] "medv"
##
## $size
## [1] 506
##
## $n.feat
## numerics  factors  ordered
##       12        1        0
##
## $has.missings
## [1] FALSE
##
## $has.weights
## [1] FALSE
##
## $has.blocking
## [1] FALSE
##
## attr(,"class")
## [1] "RegrTaskDesc"       "SupervisedTaskDesc" "TaskDesc"
```

```
getTaskType(bh.task)
```

```
## [1] "regr"
```