Boston House Price Prediction

Akhil

2023-24-4

Loading Packages

```
library(mlbench)
library(ggplot2)
library(beeswarm)
```

dataset description

crim per capita crime rate by town
zn | proportion of residential land zoned for lots over 25,000 sq.ft indus | proportion of non-retail business acres per town chas | Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
nox | nitric oxides concentration (parts per 10 million)
rm | average number of rooms per dwelling
age | proportion of owner-occupied units built prior to 1940
dis | weighted distances to five Boston employment centres
rad | index of accessibility to radial highways tax | full-value property-tax rate per USD 10,000 ptratio | pupil-teacher ratio by town b | where BB is the proportion of blacks by town lstat | percentage of lower status of the population
medv | median value of owner-occupied homes in USD 1000's | target variable

Data Input

```
data(BostonHousing)

head(BostonHousing)
```

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

```
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

```
str(BostonHousing)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : num  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ b      : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```
# checking for null values
colSums(is.na(BostonHousing))
```

```
##    crim      zn   indus    chas     nox      rm     age     dis     rad     tax
##       0       0       0       0       0       0       0       0       0       0
## ptratio       b   lstat    medv
##       0       0       0       0
```

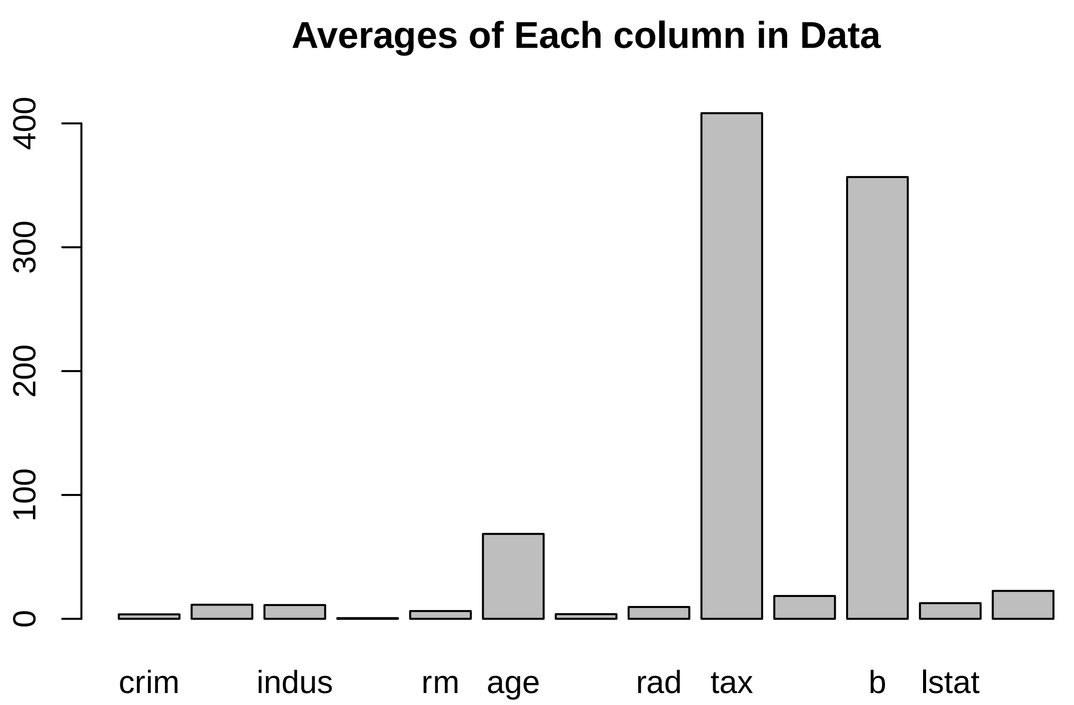
```
summary(BostonHousing)
```

```
##       crim                zn             indus       chas         nox
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   0:471   Min.   :0.3850
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1: 35   1st Qu.:0.4490
##  Median : 0.25651   Median :  0.00   Median : 9.69           Median :0.5380
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14           Mean   :0.5547
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10           3rd Qu.:0.6240
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74           Max.   :0.8710
##        rm             age              dis              rad
##  Min.   :3.561   Min.   :  2.90   Min.   : 1.130   Min.   : 1.000
##  1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100   1st Qu.: 4.000
##  Median :6.208   Median : 77.50   Median : 3.207   Median : 5.000
##  Mean   :6.285   Mean   : 68.57   Mean   : 3.795   Mean   : 9.549
##  3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188   3rd Qu.:24.000
##  Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.000
##       tax           ptratio            b              lstat
##  Min.   :187.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.73
##  1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38   1st Qu.: 6.95
##  Median :330.0   Median :19.05   Median :391.44   Median :11.36
##  Mean   :408.2   Mean   :18.46   Mean   :356.67   Mean   :12.65
```

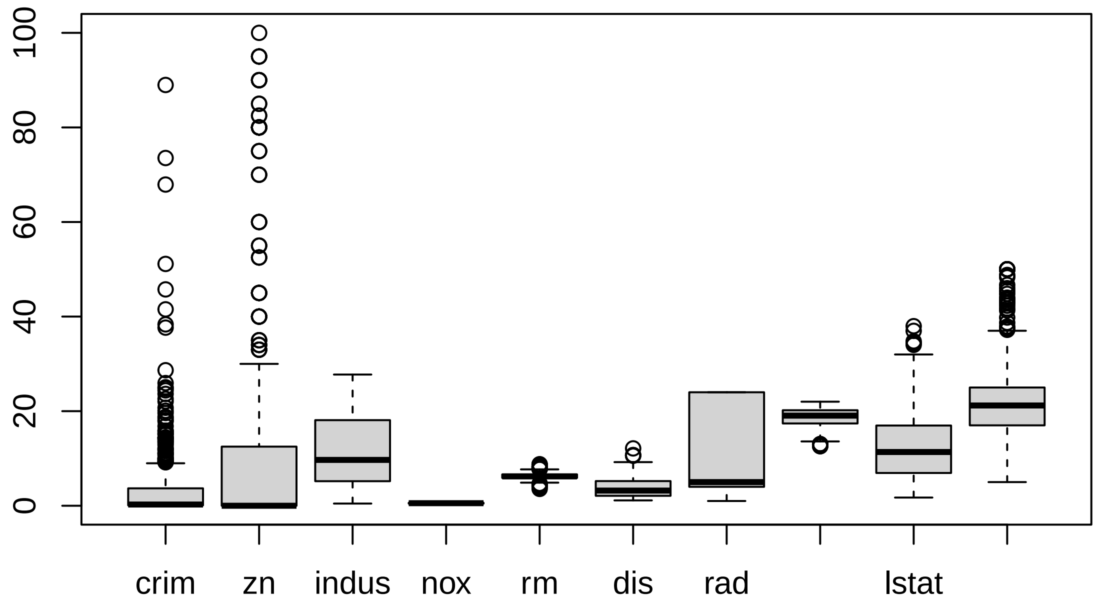
```
##   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23   3rd Qu.:16.95
##   Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.97
##        medv
##   Min.   : 5.00
##   1st Qu.:17.02
##   Median :21.20
##   Mean   :22.53
##   3rd Qu.:25.00
##   Max.   :50.00
```

Visualization

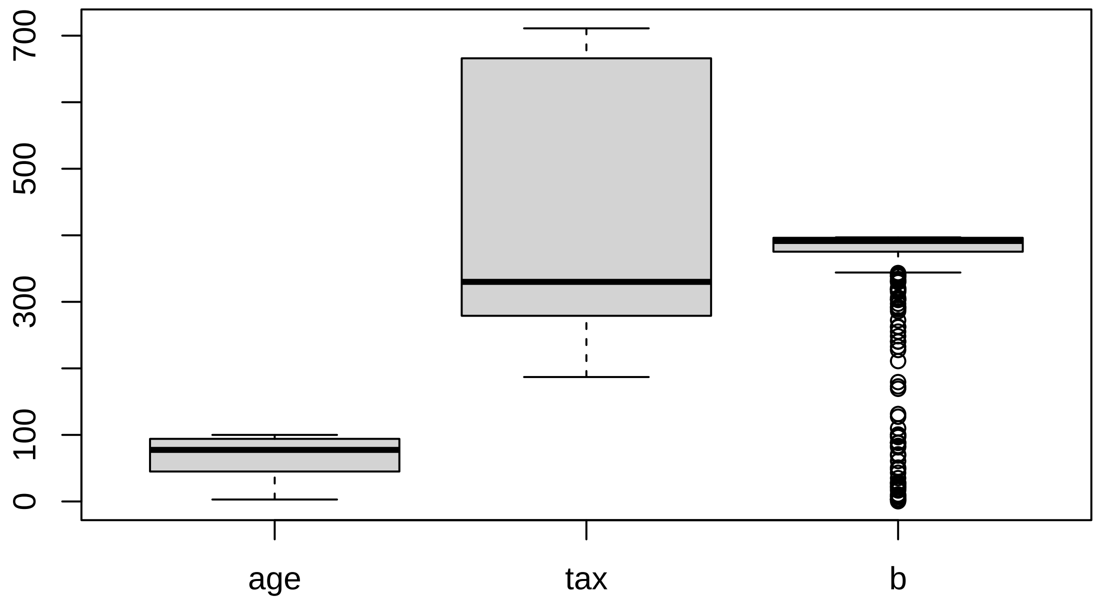
```
avg<-colMeans(BostonHousing[-4])
barplot(avg, main="Averages of Each column in Data")
```




```
boxplot(BostonHousing[-c(4,7,10,12)])
```




```
boxplot(BostonHousing[c(7,10,12)])
```

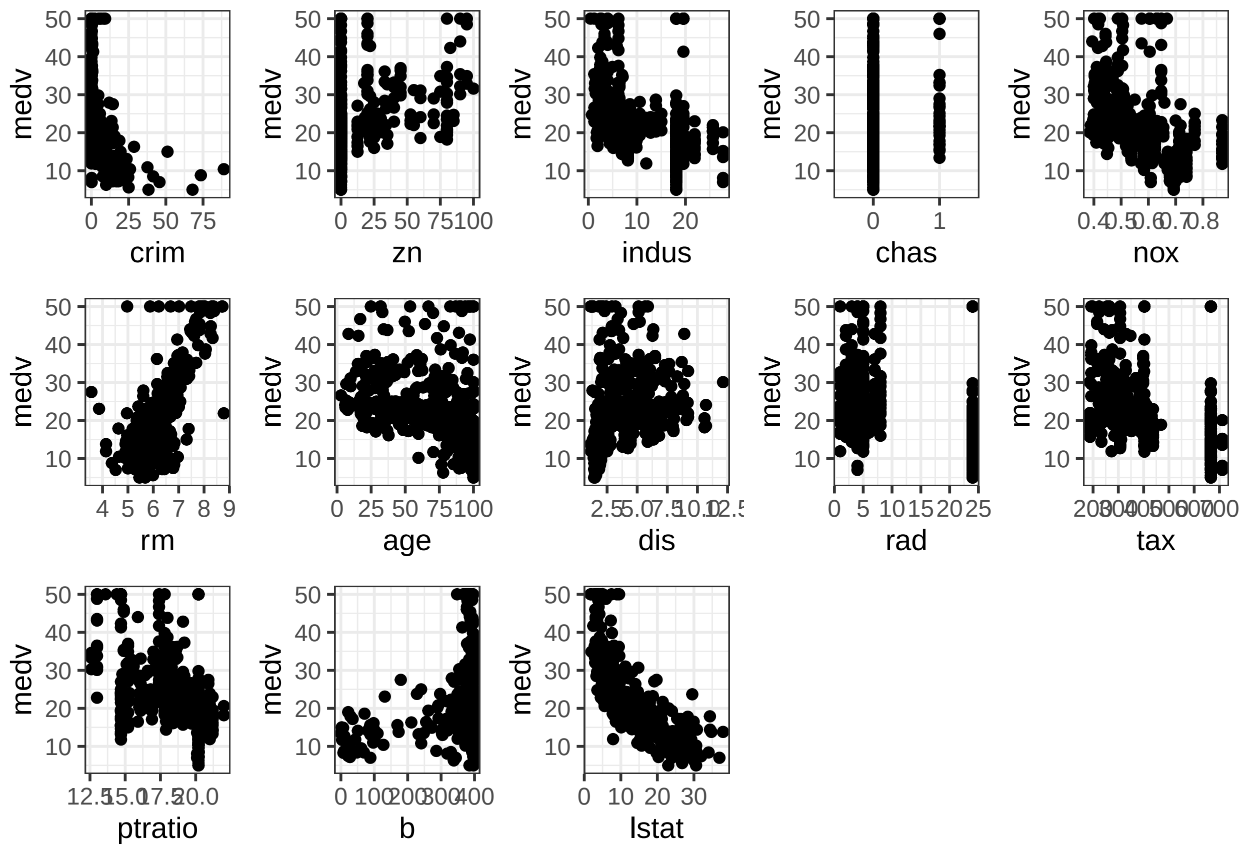



```
# choosing to plot medv vs remaining columns in dataset
price <- "medv"

# create a for loop to iterate over each column in the data frame
plots_list <- lapply(names(BostonHousing), function(var) {
  if (var != price) {
    ggplot(BostonHousing, aes_string(x = var, y = price)) +
      geom_point() +
      labs(x = var, y = price) +
      theme_bw()
  } else {
    NULL
  }
})
```

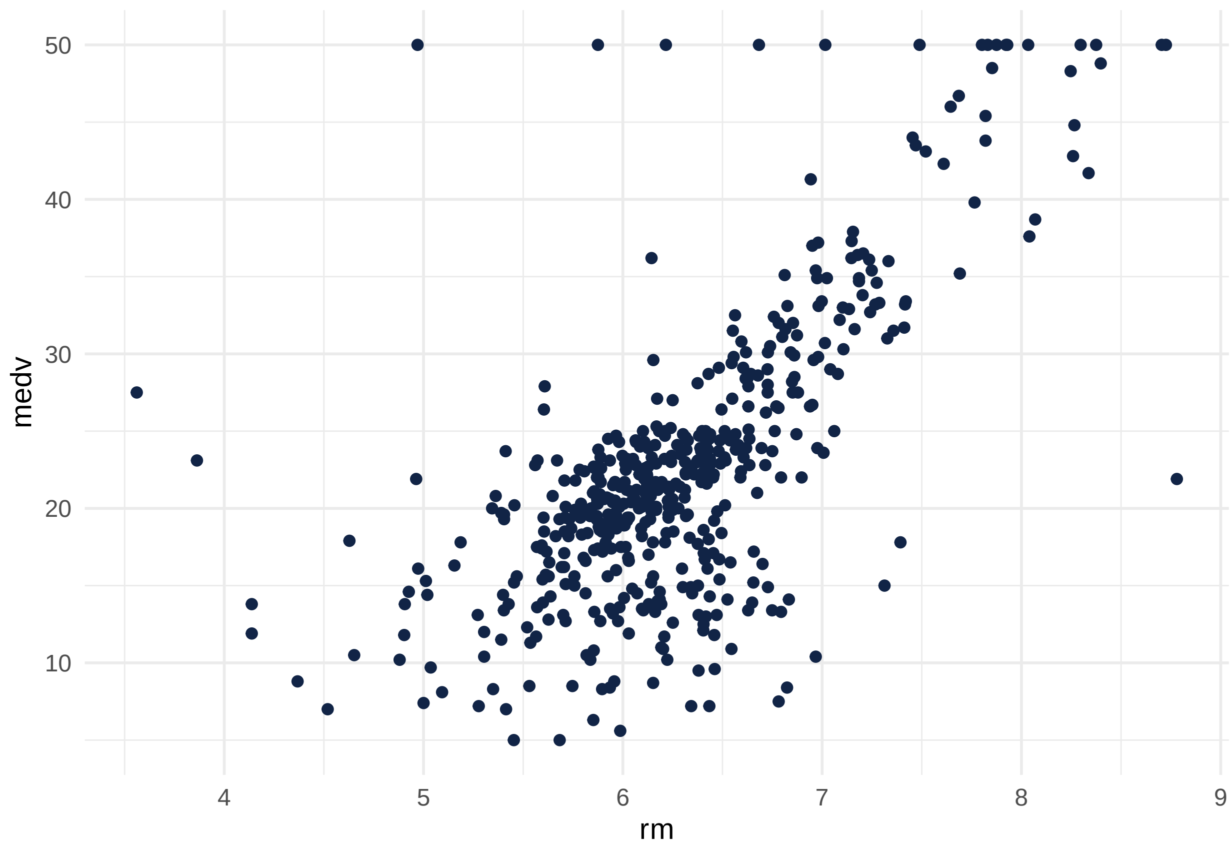
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation ideoms with `aes()`
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
library(gridExtra)
grid.arrange(grobs = plots_list, ncol = 5)
```

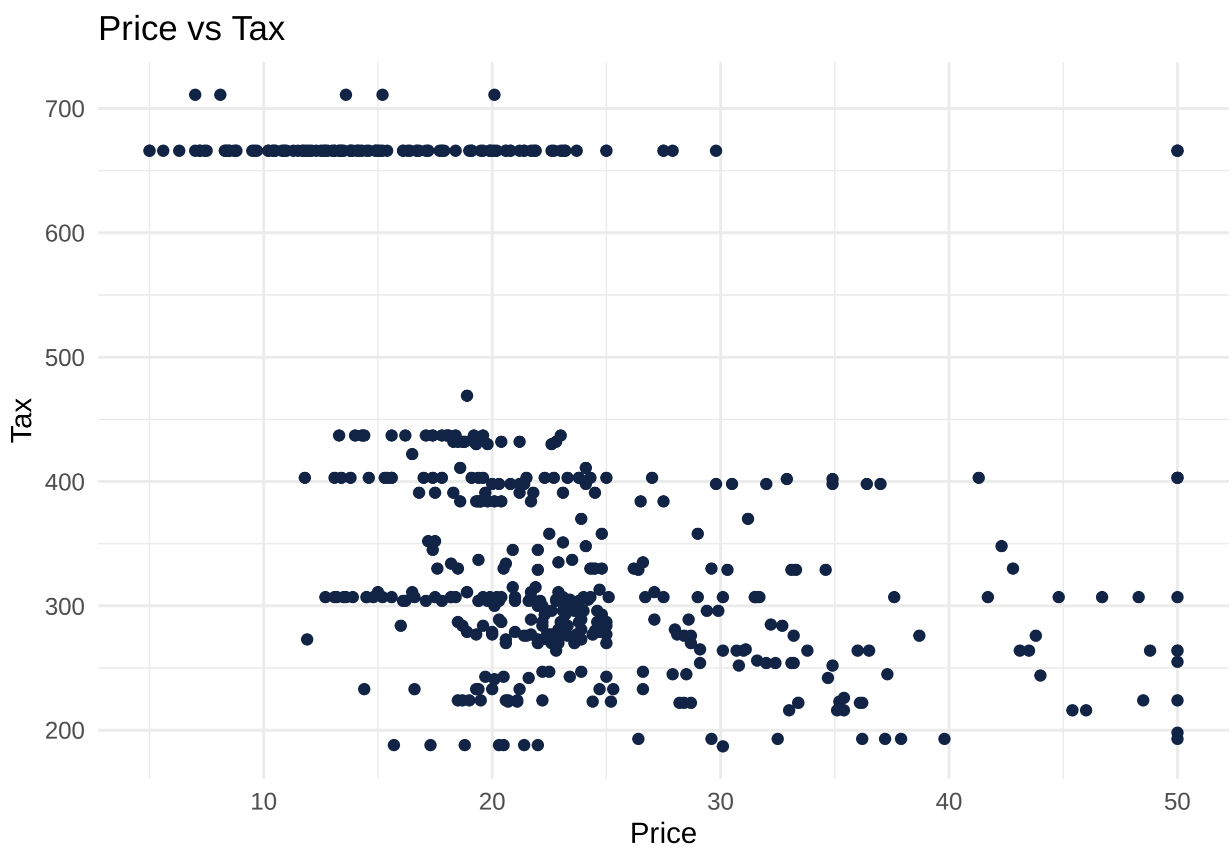


Plot between Average Room vs Price

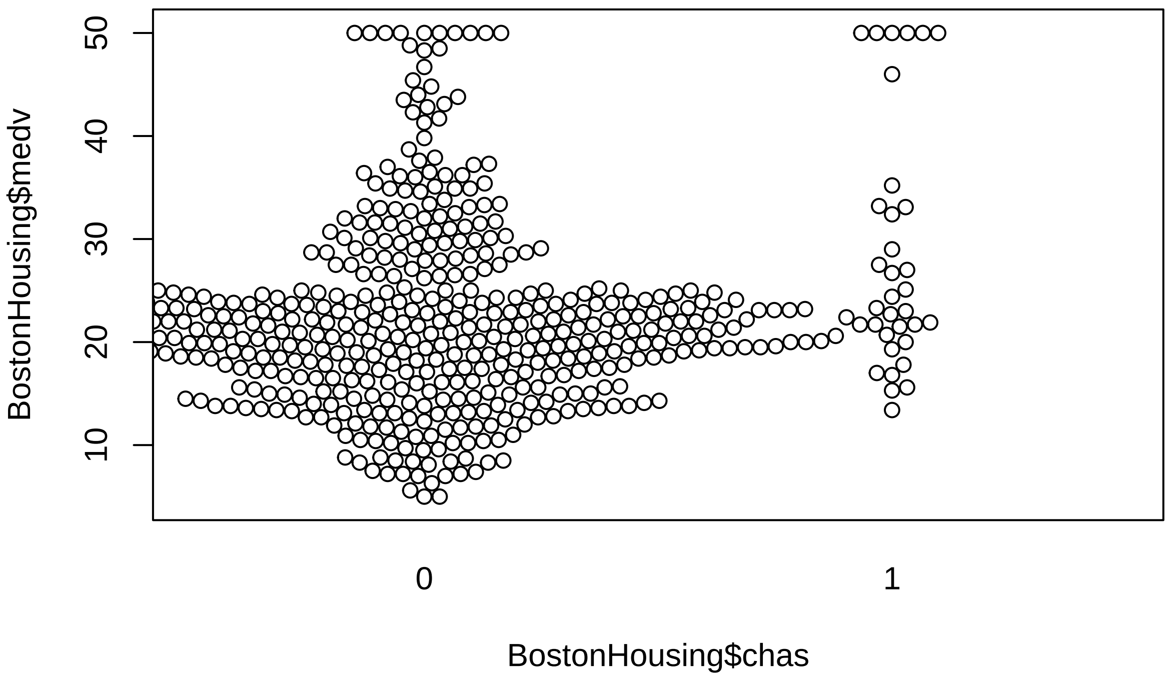
```
ggplot(BostonHousing) +
  aes(x = rm, y = medv) +
  geom_point(shape = "circle", size = 1.5, colour = "#112446") +
  theme_minimal()
```




```
ggplot(BostonHousing) +
  aes(x = medv, y = tax) +
  geom_point(shape = "circle", size = 1.5, colour = "#112446") +
  labs(x = "Price", y = "Tax", title = "Price vs Tax") +
  theme_minimal()
```



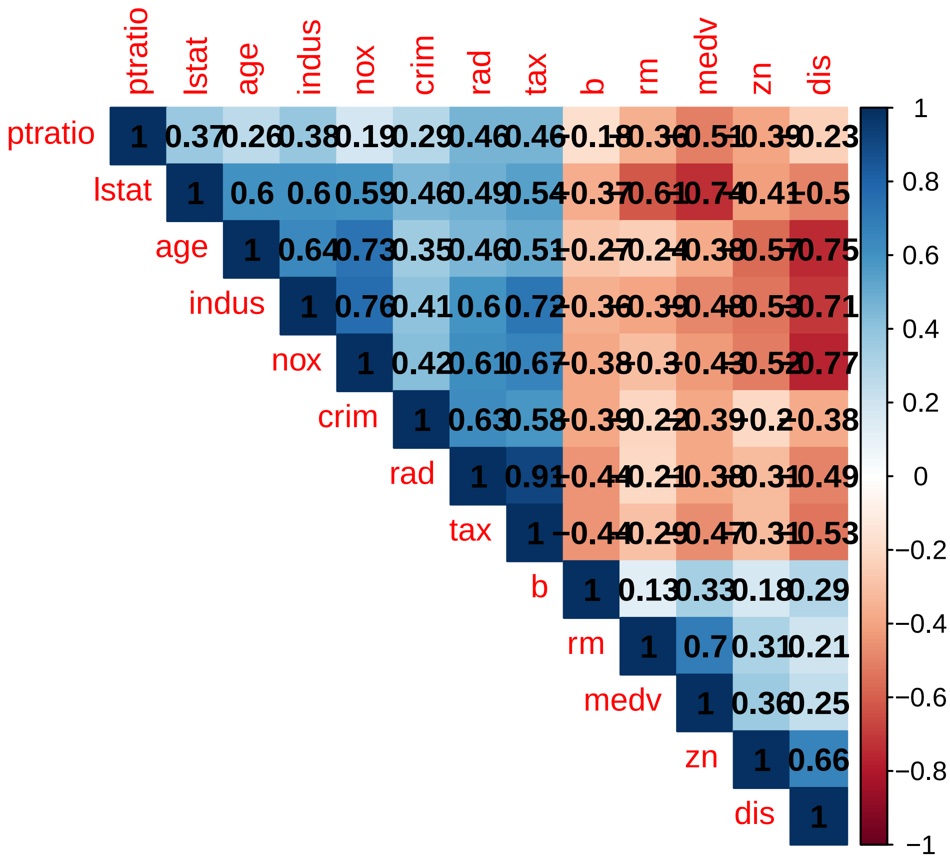

```
beeswarm(BostonHousing$medv~BostonHousing$chas)
```




```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
# correlation plot using the corrplot function
corrplot(cor(BostonHousing[-4]), method = "color", type = "upper", order = "hclust", addCoef.col = "blac
```



Model building

```
model_bh <- lm(medv~.,data=BostonHousing)
anova(model_bh)
```

```
## Analysis of Variance Table
##
## Response: medv
##            Df  Sum Sq Mean Sq  F value    Pr(>F)
## crim        1  6440.8  6440.8 286.0300 < 2.2e-16 ***
## zn          1  3554.3  3554.3 157.8452 < 2.2e-16 ***
## indus       1  2551.2  2551.2 113.2984 < 2.2e-16 ***
## chas        1  1529.8  1529.8  67.9393 1.543e-15 ***
## nox         1    76.2    76.2   3.3861 0.0663505 .
## rm          1 10938.1 10938.1 485.7530 < 2.2e-16 ***
## age         1    90.3    90.3   4.0087 0.0458137 *
## dis         1  1779.5  1779.5  79.0262 < 2.2e-16 ***
## rad         1    34.1    34.1   1.5159 0.2188325
## tax         1   329.6   329.6  14.6352 0.0001472 ***
## ptratio     1  1309.3  1309.3  58.1454 1.266e-13 ***
## b           1   593.3   593.3  26.3496 4.109e-07 ***
## lstat       1  2410.8  2410.8 107.0634 < 2.2e-16 ***
## Residuals 492 11078.8    22.5
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Selecting variables based on significance

```r
library(ISLR)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:gridExtra':
##
##     combine

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

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

```r
boston_filtered <- BostonHousing %>% select("medv","crim","zn","indus","chas","rm",
                                            "age","dis","tax","ptratio","b",
                                            "lstat")
```

Partitioning data into train and test

```r
set.seed(123)
library(caret)
```

```
## Loading required package: lattice
```

```r
#Partitioning Data into 80% Training and 20% Validation
Index_Train<-createDataPartition(boston_filtered$medv, p=0.8, list=FALSE)
boston_Train <-boston_filtered[Index_Train,]
boston_Validation  <-boston_filtered[-Index_Train,]
```

Normalizing data

```r
norm_model<-preProcess(boston_Train, method = c("center", "scale"))
#Applying Normalization model to all three data
boston_norm_Train <-predict(norm_model,boston_Train)
boston_norm_Validation  <-predict(norm_model,boston_Validation)
```

Linear Regression Model

```
linear <- lm(medv~.,data=boston_norm_Train)
summary(linear)
```

```
##
## Call:
## lm(formula = medv ~ ., data = boston_norm_Train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.78665 -0.29548 -0.07201  0.17752  2.97981
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.021381   0.027964  -0.765  0.44497
## crim        -0.043119   0.035289  -1.222  0.22248
## zn           0.090186   0.040694   2.216  0.02725 *
## indus       -0.106026   0.050236  -2.111  0.03544 *
## chas1        0.290071   0.105482   2.750  0.00623 **
## rm           0.308543   0.036711   8.405 7.86e-16 ***
## age         -0.024971   0.046554  -0.536  0.59199
## dis         -0.270850   0.052204  -5.188 3.40e-07 ***
## tax         -0.007578   0.047761  -0.159  0.87401
## ptratio     -0.162151   0.034229  -4.737 3.03e-06 ***
## b            0.100759   0.031249   3.224  0.00137 **
## lstat       -0.441561   0.047523  -9.292  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5419 on 395 degrees of freedom
## Multiple R-squared:  0.7143, Adjusted R-squared:  0.7063
## F-statistic: 89.78 on 11 and 395 DF,  p-value: < 2.2e-16
```

Decision Tree

```
library(rpart.plot)
```

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

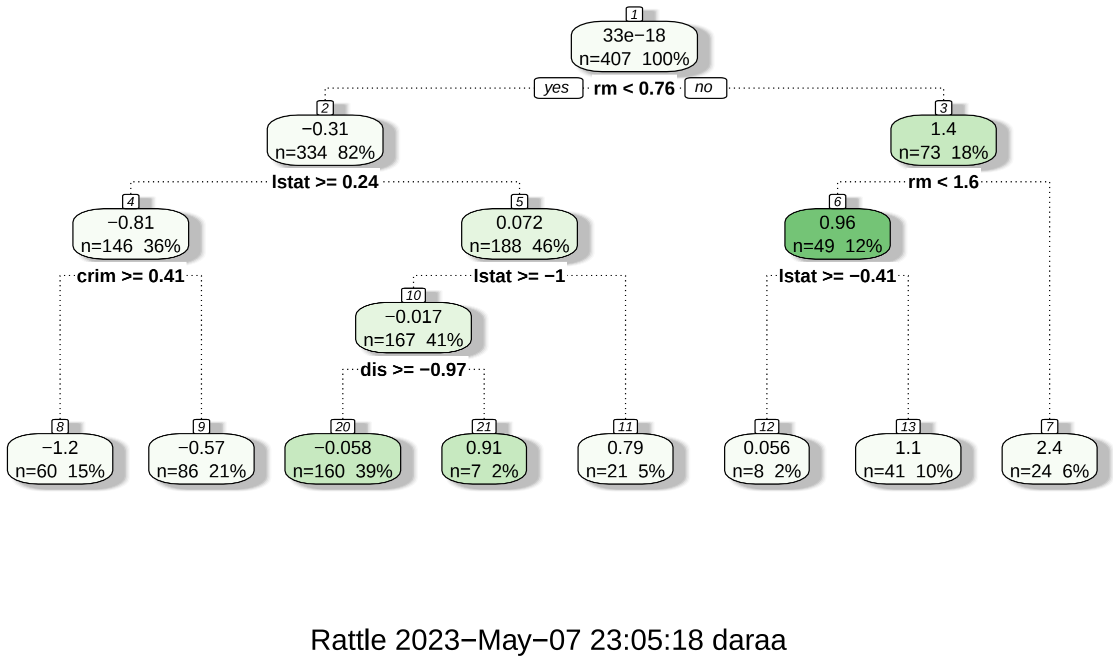
```
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(rpart)
DT=rpart(medv~.,data=boston_norm_Train, method='anova')
fancyRpartPlot(DT)
```




```r
DT_train <- caret::train(medv~.,data=boston_norm_Train,
                     method = "rpart" )
```

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

```r
DT_train
```

```
## CART
##
## 407 samples
##  11 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 407, 407, 407, 407, 407, 407, ...
## Resampling results across tuning parameters:
##
##   cp          RMSE       Rsquared   MAE
##   0.08534102  0.6604279  0.5682854  0.4723816
```

```
##   0.15865427  0.7284011  0.4721545  0.5323949
##   0.45374690  0.8093380  0.3802377  0.6012841
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.08534102.
```

Random Forest

```
set.seed(123)
Random_forest<-train(medv~., data=boston_norm_Train,method='rf')
print(Random_forest)
```

```
## Random Forest
##
## 407 samples
##  11 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 407, 407, 407, 407, 407, 407, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##    2    0.4177788  0.8369361  0.2781526
##    6    0.3754063  0.8600483  0.2590257
##   11    0.3921685  0.8445687  0.2707028
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 6.
```

SVM

```
set.seed(123)
svm<-train(medv~., data=boston_norm_Train,method='svmLinear')
print(svm)
```

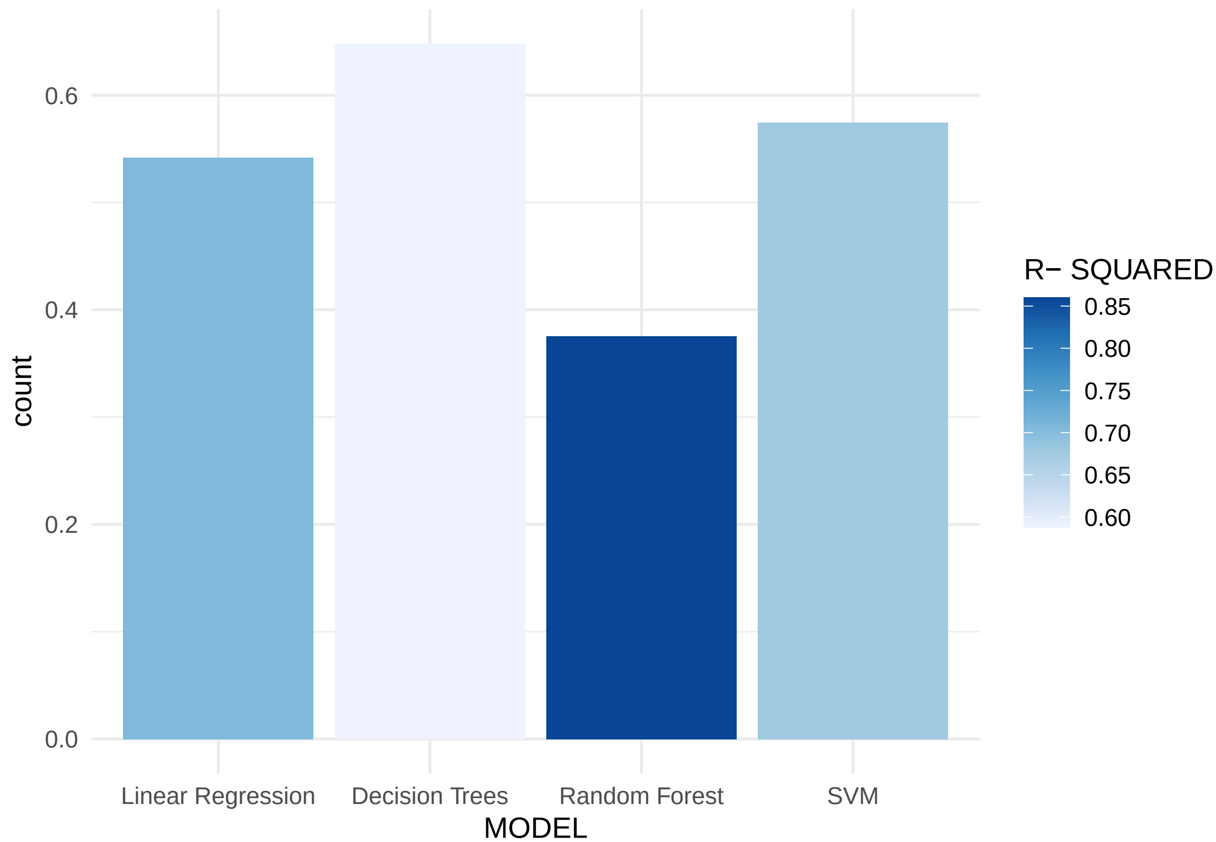
```
## Support Vector Machines with Linear Kernel
##
## 407 samples
##  11 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 407, 407, 407, 407, 407, 407, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.5744627  0.6776097  0.3528924
##
## Tuning parameter 'C' was held constant at a value of 1
```

Selecting Best Model

```
library(readr)
results <- read_csv("results.csv", col_types = cols(MODEL = col_factor(levels = c("Linear Regression",
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
results <- na.omit(results)
ggplot(results) + aes(x = MODEL, fill = `R- SQUARED`, weight = RMSE) + geom_bar() +
  scale_fill_distiller(palette = "Blues", direction = 1) + theme_minimal()
```



Based on the R- squared values from the above models, it can be seen that Random forest is performing good with more than 86%

Predicting on test data

```
set.seed(123)
Random_forest_2<-train(medv~., data=boston_Train,method='rf',
                       preProcess = c("center", "scale"))
predicted <- c(predict(Random_forest_2,boston_Validation[-1]))
```

```r
output<-as.data.frame(predicted)
output$actual <- boston_Validation$medv

output
```

```
##     predicted actual
## 3   35.106687   34.7
## 6   25.146390   28.7
## 9   18.358760   16.5
## 11  21.102163   15.0
## 14  20.209577   20.4
## 15  19.938947   18.2
## 31  14.604677   12.7
## 32  20.652347   14.5
## 36  21.052043   18.9
## 41  35.110247   34.9
## 45  22.052093   21.2
## 51  20.047930   19.7
## 54  20.770800   23.4
## 74  23.976357   23.4
## 76  23.067717   21.4
## 78  21.720560   20.8
## 79  21.093743   21.2
## 82  24.860457   23.9
## 86  26.217757   26.6
## 92  22.767870   22.0
## 105 20.020050   20.1
## 108 19.483993   20.4
## 109 20.304587   19.8
## 111 20.827193   21.7
## 120 20.121900   19.3
## 127 16.186260   15.7
## 130 16.147150   14.3
## 131 20.730813   19.2
## 138 18.281033   17.1
## 142 13.493637   14.4
## 146 15.500093   13.8
## 151 20.015437   21.5
## 152 19.761130   19.6
## 155 18.222850   17.0
## 163 46.833493   50.0
## 167 47.814030   50.0
## 168 20.219743   23.8
## 170 22.561063   22.3
## 172 20.905493   19.1
## 178 24.074203   24.6
## 182 24.570117   36.2
## 184 27.017447   32.5
## 188 27.200040   32.0
## 198 33.071363   30.3
## 203 44.927280   42.3
## 205 47.201160   50.0
## 215 18.371497   23.7
```

```
## 218 23.833997   28.7
## 221 28.319730   26.7
## 224 25.110910   30.1
## 244 24.638027   23.7
## 246 19.322300   18.5
## 247 21.748340   24.3
## 250 25.842227   26.2
## 252 27.598193   24.8
## 255 22.529070   21.9
## 257 41.468927   44.0
## 262 41.117620   43.1
## 271 21.095560   21.1
## 293 28.219563   27.9
## 294 22.710097   23.9
## 300 33.396887   29.0
## 305 33.659600   36.1
## 307 33.720593   33.4
## 312 23.620690   22.1
## 316 20.005043   16.2
## 320 21.677367   21.0
## 323 21.595203   20.4
## 326 24.908967   24.6
## 330 23.183227   22.6
## 348 24.221697   23.1
## 352 25.583637   24.1
## 355 20.084923   18.2
## 357 15.484570   17.8
## 370 40.163110   50.0
## 378 13.501350   13.3
## 393 11.563060    9.7
## 394 14.905843   13.8
## 401  9.587693    5.6
## 403 11.907750   12.1
## 405  9.486213    8.5
## 406 10.370757    5.0
## 410 14.278347   27.5
## 411 24.342193   15.0
## 417 11.015587    7.5
## 422 15.316153   14.2
## 445 11.282477   10.8
## 449 14.532750   14.1
## 453 17.220740   16.1
## 455 13.849620   14.9
## 472 20.849917   19.6
## 481 21.442207   23.0
## 484 20.732680   21.8
## 486 21.714020   21.2
## 487 19.572517   19.1
## 490 13.506037    7.0
## 493 19.939503   20.1
## 495 20.397557   24.5
## 496 19.780513   23.1
```