**Tanmay Gupta**

**Kofi Twumasi**

**Math 678 Statistical Methods for data Science**

**Boston Housing Data Set**

**Code:**

Packages used:

library(MASS)

library(ggplot2)

library(dplyr)

library(tidyverse)

library(corrplot)

library(leaps)

library(rpart)

library(mgcv)

library(glmnet)

library(boot)

library(rpart.plot)

library(caret)

library(e1071)

library(DMwR)

**# Viewing first few rows of all the variables using head function**

head(Boston)

crim zn indus chas nox rm age dis rad

1 0.00632 18 2.31 0 0.538 6.575 65.2 4.0900 1

2 0.02731 0 7.07 0 0.469 6.421 78.9 4.9671 2

3 0.02729 0 7.07 0 0.469 7.185 61.1 4.9671 2

4 0.03237 0 2.18 0 0.458 6.998 45.8 6.0622 3

5 0.06905 0 2.18 0 0.458 7.147 54.2 6.0622 3

6 0.02985 0 2.18 0 0.458 6.430 58.7 6.0622 3

tax ptratio black lstat medv

1 296 15.3 396.90 4.98 24.0

2 242 17.8 396.90 9.14 21.6

3 242 17.8 392.83 4.03 34.7

4 222 18.7 394.63 2.94 33.4

5 222 18.7 396.90 5.33 36.2

6 222 18.7 394.12 5.21 28.7

**#Taking a glimpse at the data to see its structure**

glimpse(Boston)

Observations: 506

Variables: 14

$ crim <dbl> 0.00632, 0.02731, 0.02729, 0.03...

$ zn <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...

$ indus <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2...

$ chas <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...

$ nox <dbl> 0.538, 0.469, 0.469, 0.458, 0.4...

$ rm <dbl> 6.575, 6.421, 7.185, 6.998, 7.1...

$ age <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 5...

$ dis <dbl> 4.0900, 4.9671, 4.9671, 6.0622,...

$ rad <int> 1, 2, 2, 3, 3, 3, 5, 5, 5, 5, 5...

$ tax <dbl> 296, 242, 242, 222, 222, 222, 3...

$ ptratio <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 1...

$ black <dbl> 396.90, 396.90, 392.83, 394.63,...

$ lstat <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5...

$ medv <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 2...

**#scatterplot showing linear relationship between all the variables and our target variable medv**

Boston %>%

gather(key, val, -medv) %>%

ggplot(aes(x = val, y = medv)) +

geom\_point() +

stat\_smooth(method = "lm", se = TRUE, col = "blue") +

facet\_wrap(~key, scales = "free") +

theme\_gray() +

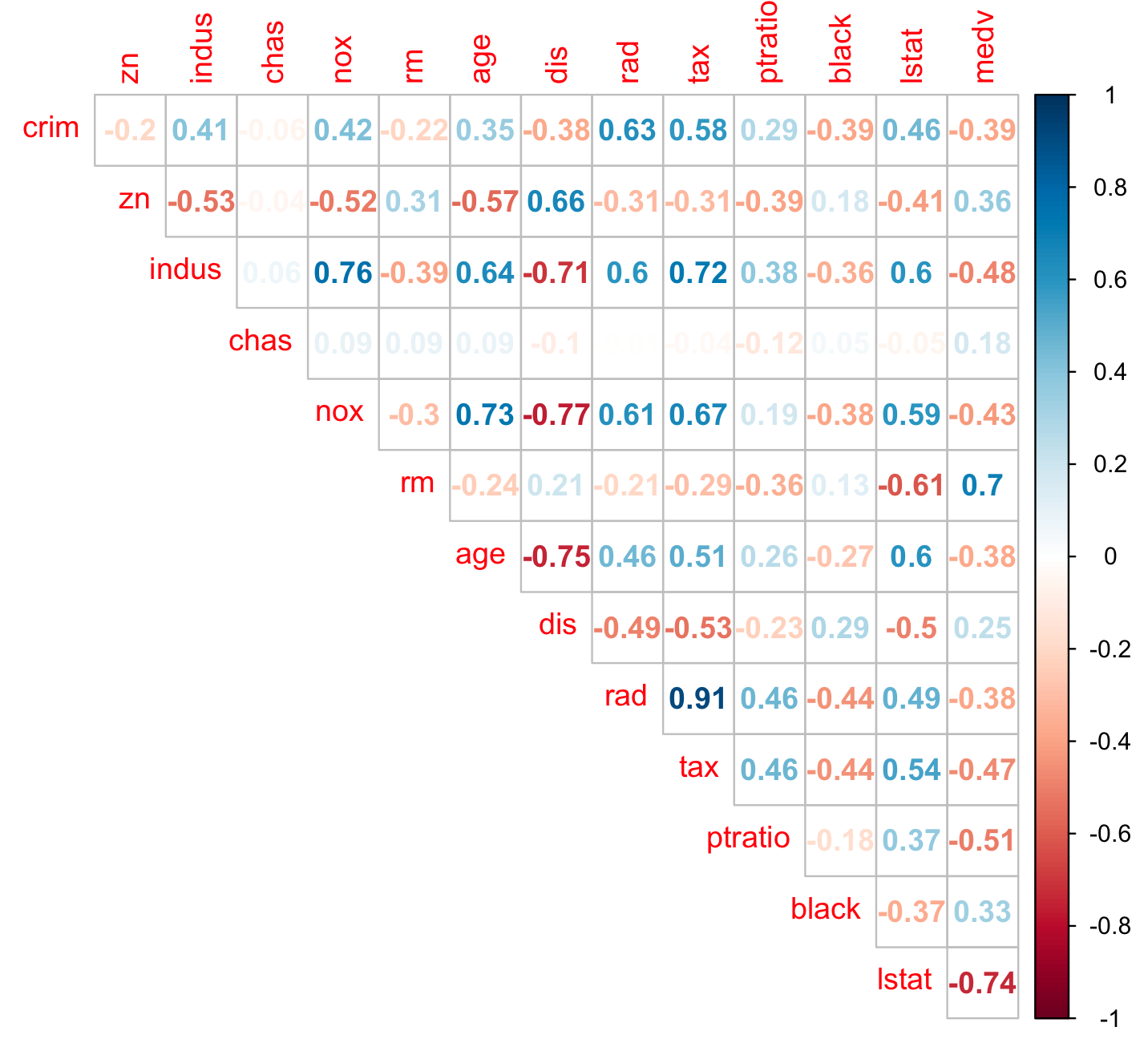
ggtitle("Scatter plot of dependent variables vs medv")

A screenshot of a cell phone

Description automatically generated

**#correlation plot showing correlation between all the variables**

> corrplot(cor(Boston), method = "number", type ="upper", diag = FALSE)



**#Splitting the dataset and taking 80% as training set and 20% as testing i.e 404 rows as training and 102 as testing**

data("Boston")

set.seed(100)

train\_index<- sample(nrow(Boston), nrow(Boston)\* 0.8)

Boston\_train<- Boston[train\_index, ]

Boston\_test<- Boston[-train\_index, ]

**#linear model on training set**

model\_reg <- lm(medv ~ ., data = Boston\_train)

model\_reg\_sum <- summary(model\_reg)

Model\_reg\_sum

Call:

lm(formula = medv ~ ., data = Boston\_train)

Residuals:

Min 1Q Median 3Q Max

-10.4773 -2.7210 -0.5807 1.7300 24.7911

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 40.136203 5.717257 7.020 9.89e-12

crim -0.103364 0.034426 -3.003 0.002850

zn 0.048977 0.015446 3.171 0.001640

indus 0.018130 0.072197 0.251 0.801856

chas 3.103563 0.923009 3.362 0.000849

nox -18.599930 4.283486 -4.342 1.80e-05

rm 3.432782 0.471435 7.282 1.84e-12

age 0.011374 0.014906 0.763 0.445897

dis -1.471879 0.222852 -6.605 1.31e-10

rad 0.314074 0.073161 4.293 2.23e-05

tax -0.011651 0.004267 -2.731 0.006610

ptratio -0.986587 0.145275 -6.791 4.16e-11

black 0.007513 0.003174 2.367 0.018433

lstat -0.588967 0.055680 -10.578 < 2e-16

(Intercept) \*\*\*

crim \*\*

zn \*\*

indus

chas \*\*\*

nox \*\*\*

rm \*\*\*

age

dis \*\*\*

rad \*\*\*

tax \*\*

ptratio \*\*\*

black \*

lstat \*\*\*

---

Signif. codes:

0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.731 on 390 degrees of freedom

Multiple R-squared: 0.7441, Adjusted R-squared: 0.7355

F-statistic: 87.21 on 13 and 390 DF, p-value: < 2.2e-16

**#linear model on test**

model\_reg\_test<- predict(model\_reg, newdata = Boston\_test)

head(model\_reg\_test)

1 15 17 19 28

30.16385 19.30443 20.24563 15.99887 14.47698

37

22.22397

**#using best subset**

model\_1 <- regsubsets(medv ~., data= Boston\_train, nbest=1, nvmax= 13)

summary(model\_1)

Subset selection object

Call: regsubsets.formula(medv ~ ., data = Boston\_train, nbest = 1,

nvmax = 13)

13 Variables (and intercept)

Forced in Forced out

crim FALSE FALSE

zn FALSE FALSE

indus FALSE FALSE

chas FALSE FALSE

nox FALSE FALSE

rm FALSE FALSE

age FALSE FALSE

dis FALSE FALSE

rad FALSE FALSE

tax FALSE FALSE

ptratio FALSE FALSE

black FALSE FALSE

lstat FALSE FALSE

1 subsets of each size up to 13

Selection Algorithm: exhaustive

crim zn indus chas nox rm age dis rad

1 ( 1 ) " " " " " " " " " " " " " " " " " "

2 ( 1 ) " " " " " " " " " " "\*" " " " " " "

3 ( 1 ) " " " " " " " " " " "\*" " " " " " "

4 ( 1 ) " " " " " " " " " " "\*" " " "\*" " "

5 ( 1 ) " " " " " " " " "\*" "\*" " " "\*" " "

6 ( 1 ) " " " " " " "\*" "\*" "\*" " " "\*" " "

7 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*" " "

8 ( 1 ) "\*" " " " " "\*" "\*" "\*" " " "\*" "\*"

9 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*" "\*"

10 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*" "\*"

11 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*" "\*"

12 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*"

13 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

tax ptratio black lstat

1 ( 1 ) " " " " " " "\*"

2 ( 1 ) " " " " " " "\*"

3 ( 1 ) " " "\*" " " "\*"

4 ( 1 ) " " "\*" " " "\*"

5 ( 1 ) " " "\*" " " "\*"

6 ( 1 ) " " "\*" " " "\*"

7 ( 1 ) " " "\*" " " "\*"

8 ( 1 ) " " "\*" " " "\*"

9 ( 1 ) " " "\*" " " "\*"

10 ( 1 ) "\*" "\*" " " "\*"

11 ( 1 ) "\*" "\*" "\*" "\*"

12 ( 1 ) "\*" "\*" "\*" "\*"

13 ( 1 ) "\*" "\*" "\*" "\*"

**##Variable selection using stepwise selection**

Null linear model

nullmodel <- lm(medv ~ 1, data = Boston\_train)

Full linear model

fullmodel <- lm(medv ~ ., data = Boston\_train)

**#forward selection**

model\_2<-step(nullmodel, scope = list(lower = nullmodel, upper = fullmodel), direction = "forward")

Start: AIC=1794.1

medv ~ 1

Df Sum of Sq RSS AIC

+ lstat 1 18882.8 15226 1470.2

+ rm 1 16276.0 17832 1534.1

+ ptratio 1 9523.9 24584 1663.8

+ tax 1 7627.9 26480 1693.8

+ indus 1 7324.6 26784 1698.4

+ nox 1 5913.2 28195 1719.2

+ rad 1 5021.7 29087 1731.8

+ crim 1 5018.5 29090 1731.8

+ age 1 4779.6 29329 1735.1

+ zn 1 4293.4 29815 1741.7

+ black 1 3449.2 30659 1753.0

+ dis 1 2093.9 32014 1770.5

+ chas 1 1056.1 33052 1783.4

<none> 34108 1794.1

Step: AIC=1470.24

medv ~ lstat

Df Sum of Sq RSS AIC

+ rm 1 2941.58 12284 1385.5

+ ptratio 1 2209.92 13016 1408.9

+ chas 1 796.68 14429 1450.5

+ dis 1 737.33 14488 1452.2

+ age 1 295.97 14930 1464.3

+ tax 1 158.27 15067 1468.0

+ crim 1 81.23 15144 1470.1

+ zn 1 79.62 15146 1470.1

<none> 15226 1470.2

+ black 1 63.28 15162 1470.6

+ indus 1 37.42 15188 1471.2

+ nox 1 18.58 15207 1471.8

+ rad 1 3.99 15222 1472.1

Step: AIC=1385.51

medv ~ lstat + rm

Df Sum of Sq RSS AIC

+ ptratio 1 1337.28 10947 1341.0

+ chas 1 641.96 11642 1365.8

+ dis 1 419.55 11864 1373.5

+ black 1 201.23 12083 1380.8

+ tax 1 178.43 12106 1381.6

+ crim 1 170.05 12114 1381.9

+ age 1 75.82 12208 1385.0

<none> 12284 1385.5

+ rad 1 43.67 12240 1386.1

+ zn 1 19.76 12264 1386.9

+ indus 1 8.34 12276 1387.2

+ nox 1 0.25 12284 1387.5

Step: AIC=1340.95

medv ~ lstat + rm + ptratio

Df Sum of Sq RSS AIC

+ dis 1 546.03 10401 1322.3

+ chas 1 414.26 10532 1327.4

+ black 1 169.40 10777 1336.7

+ age 1 131.58 10815 1338.1

+ crim 1 59.78 10887 1340.7

<none> 10947 1341.0

+ rad 1 47.26 10899 1341.2

+ zn 1 22.12 10924 1342.1

+ indus 1 10.28 10936 1342.6

+ tax 1 2.79 10944 1342.8

+ nox 1 0.45 10946 1342.9

Step: AIC=1322.27

medv ~ lstat + rm + ptratio + dis

Df Sum of Sq RSS AIC

+ nox 1 499.37 9901.2 1304.4

+ chas 1 295.49 10105.1 1312.6

+ black 1 237.57 10163.0 1314.9

+ indus 1 196.54 10204.1 1316.6

+ zn 1 156.10 10244.5 1318.2

+ crim 1 142.85 10257.8 1318.7

+ tax 1 119.46 10281.2 1319.6

<none> 10400.6 1322.3

+ age 1 25.87 10374.7 1323.3

+ rad 1 0.42 10400.2 1324.3

Step: AIC=1304.4

medv ~ lstat + rm + ptratio + dis + nox

Df Sum of Sq RSS AIC

+ chas 1 350.10 9551.1 1291.8

+ zn 1 153.36 9747.9 1300.1

+ black 1 150.70 9750.6 1300.2

+ rad 1 89.12 9812.1 1302.7

+ crim 1 86.99 9814.3 1302.8

<none> 9901.2 1304.4

+ indus 1 19.25 9882.0 1305.6

+ age 1 1.44 9899.8 1306.3

+ tax 1 0.80 9900.4 1306.4

Step: AIC=1291.85

medv ~ lstat + rm + ptratio + dis + nox + chas

Df Sum of Sq RSS AIC

+ zn 1 160.673 9390.5 1287.0

+ black 1 125.439 9425.7 1288.5

+ rad 1 108.724 9442.4 1289.2

+ crim 1 64.013 9487.1 1291.1

<none> 9551.1 1291.8

+ indus 1 28.105 9523.0 1292.7

+ tax 1 1.193 9550.0 1293.8

+ age 1 0.436 9550.7 1293.8

Step: AIC=1287

medv ~ lstat + rm + ptratio + dis + nox + chas + zn

Df Sum of Sq RSS AIC

+ black 1 142.489 9248.0 1282.8

+ crim 1 104.162 9286.3 1284.5

+ rad 1 66.011 9324.5 1286.2

<none> 9390.5 1287.0

+ indus 1 28.314 9362.2 1287.8

+ age 1 6.578 9383.9 1288.7

+ tax 1 5.502 9385.0 1288.8

Step: AIC=1282.82

medv ~ lstat + rm + ptratio + dis + nox + chas + zn + black

Df Sum of Sq RSS AIC

+ rad 1 115.883 9132.1 1279.7

+ crim 1 64.248 9183.7 1282.0

<none> 9248.0 1282.8

+ indus 1 19.159 9228.8 1284.0

+ age 1 2.476 9245.5 1284.7

+ tax 1 0.018 9248.0 1284.8

Step: AIC=1279.73

medv ~ lstat + rm + ptratio + dis + nox + chas + zn + black +

rad

Df Sum of Sq RSS AIC

+ crim 1 188.587 8943.5 1273.3

+ tax 1 186.216 8945.9 1273.4

<none> 9132.1 1279.7

+ indus 1 33.416 9098.7 1280.2

+ age 1 7.356 9124.7 1281.4

Step: AIC=1273.3

medv ~ lstat + rm + ptratio + dis + nox + chas + zn + black +

rad + crim

Df Sum of Sq RSS AIC

+ tax 1 199.421 8744.1 1266.2

<none> 8943.5 1273.3

+ indus 1 39.008 8904.5 1273.5

+ age 1 7.751 8935.8 1275.0

Step: AIC=1266.19

medv ~ lstat + rm + ptratio + dis + nox + chas + zn + black +

rad + crim + tax

Df Sum of Sq RSS AIC

<none> 8744.1 1266.2

+ age 1 12.7502 8731.3 1267.6

+ indus 1 1.1286 8743.0 1268.1

**#backward selection**

model.step.b <- step(fullmodel, direction = "backward")

Start: AIC=1201.86

medv ~ crim + zn + indus + chas + nox + rm + age + dis + rad +

tax + ptratio + black + lstat

Df Sum of Sq RSS AIC

- age 1 3.44 7386.9 1200.0

- indus 1 7.05 7390.5 1200.2

<none> 7383.5 1201.9

- tax 1 112.28 7495.8 1206.0

- nox 1 199.71 7583.2 1210.6

- black 1 199.71 7583.2 1210.6

- zn 1 231.31 7614.8 1212.3

- chas 1 275.73 7659.2 1214.7

- crim 1 315.16 7698.6 1216.7

- rad 1 337.70 7721.2 1217.9

- dis 1 887.29 8270.8 1245.7

- ptratio 1 965.06 8348.5 1249.5

- rm 1 1065.56 8449.0 1254.3

- lstat 1 2623.59 10007.1 1322.7

Step: AIC=1200.04

medv ~ crim + zn + indus + chas + nox + rm + dis + rad + tax +

ptratio + black + lstat

Df Sum of Sq RSS AIC

- indus 1 7.02 7393.9 1198.4

<none> 7386.9 1200.0

- tax 1 114.91 7501.8 1204.3

- black 1 197.09 7584.0 1208.7

- nox 1 221.79 7608.7 1210.0

- zn 1 238.13 7625.0 1210.9

- chas 1 272.31 7659.2 1212.7

- crim 1 314.86 7701.8 1214.9

- rad 1 346.57 7733.5 1216.6

- dis 1 970.23 8357.1 1247.9

- ptratio 1 978.06 8365.0 1248.3

- rm 1 1077.57 8464.5 1253.1

- lstat 1 3042.49 10429.4 1337.4

Step: AIC=1198.43

medv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio +

black + lstat

Df Sum of Sq RSS AIC

<none> 7393.9 1198.4

- tax 1 114.18 7508.1 1202.6

- black 1 195.82 7589.8 1207.0

- nox 1 217.41 7611.3 1208.1

- zn 1 231.58 7625.5 1208.9

- chas 1 279.57 7673.5 1211.4

- crim 1 317.76 7711.7 1213.4

- rad 1 348.05 7742.0 1215.0

- ptratio 1 974.03 8368.0 1246.4

- dis 1 1048.81 8442.7 1250.0

- rm 1 1071.22 8465.2 1251.1

- lstat 1 3038.19 10432.1 1335.5

**#stepwise selection**

model\_3<- step(nullmodel, scope = list(lower = nullmodel, upper = fullmodel), direction = "both")

Call:

lm(formula = medv ~ lstat + rm + ptratio + dis + chas + black +

zn + nox + crim + rad + tax, data = Boston\_train)

Residuals:

Min 1Q Median 3Q Max

-10.7147 -2.6841 -0.3391 1.7645 24.1730

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 36.313437 5.556195 6.536 1.97e-10 \*\*\*

lstat -0.650201 0.051231 -12.691 < 2e-16 \*\*\*

rm 3.458593 0.458938 7.536 3.39e-13 \*\*\*

ptratio -0.960444 0.133653 -7.186 3.39e-12 \*\*\*

dis -1.444988 0.193780 -7.457 5.74e-13 \*\*\*

chas 3.624134 0.941359 3.850 0.000138 \*\*\*

black 0.008730 0.002709 3.222 0.001379 \*\*

zn 0.048326 0.013792 3.504 0.000511 \*\*\*

nox -12.432692 3.662008 -3.395 0.000756 \*\*\*

crim -0.127936 0.031170 -4.104 4.93e-05 \*\*\*

rad 0.268332 0.062466 4.296 2.20e-05 \*\*\*

tax -0.008222 0.003342 -2.460 0.014309 \*

---

Signif. codes:

0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.343 on 392 degrees of freedom

Multiple R-squared: 0.7694, Adjusted R-squared: 0.7629

F-statistic: 118.9 on 11 and 392 DF, p-value: < 2.2e-16

**#linear model without indus and age**

best\_reg <- lm(medv ~ . -indus -age, data = Boston\_train)

best\_reg\_sum <- summary(best\_reg)

Best\_reg\_sum

Call:

lm(formula = medv ~ . - indus - age, data = Boston\_train)

Residuals:

Min 1Q Median 3Q Max

-10.7147 -2.6841 -0.3391 1.7645 24.1730

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 36.313437 5.556195 6.536 1.97e-10 \*\*\*

crim -0.127936 0.031170 -4.104 4.93e-05 \*\*\*

zn 0.048326 0.013792 3.504 0.000511 \*\*\*

chas 3.624134 0.941359 3.850 0.000138 \*\*\*

nox -12.432692 3.662008 -3.395 0.000756 \*\*\*

rm 3.458593 0.458938 7.536 3.39e-13 \*\*\*

dis -1.444988 0.193780 -7.457 5.74e-13 \*\*\*

rad 0.268332 0.062466 4.296 2.20e-05 \*\*\*

tax -0.008222 0.003342 -2.460 0.014309 \*

ptratio -0.960444 0.133653 -7.186 3.39e-12 \*\*\*

black 0.008730 0.002709 3.222 0.001379 \*\*

lstat -0.650201 0.051231 -12.691 < 2e-16 \*\*\*

---

Signif. codes:

0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.343 on 392 degrees of freedom

Multiple R-squared: 0.7694, Adjusted R-squared: 0.7629

F-statistic: 118.9 on 11 and 392 DF, p-value: < 2.2e-16

**#linear model without indus and age on test**

best\_reg\_test <- predict(best\_reg, newdata = Boston\_test)

head(best\_reg\_test)<-

1 15 17 19 28

30.21196 19.12151 20.63056 16.38052 14.33074

37

22.31900

**#checking how the model performed**

**#for MSE and other statistics**

regr.eval(Boston\_test$medv, best\_reg\_test)

mae mse rmse mape

3.5843402 23.4398715 4.8414741 0.1697427

**> summary(best\_reg)$r.sq**

[1] 0.7436371

**#actul and predicted correlation**

actuals\_pred<- data.frame(cbind(actuals= Boston\_test$medv, predicted= best\_reg\_test))

correlation\_accuracy<- cor(actuals\_pred)

Correlation\_accuracy

Actuals predicteds

actuals 1.0000000 0.8014211

predicteds 0.8014211 1.0000000

head(actuals\_pred)

actuals predicteds

5 36.2 27.587846

8 27.1 18.331634

10 18.9 18.064179

18 17.5 16.306267

24 14.5 12.583526

29 18.4 18.996181

30 21.0 20.363455

35 13.5 12.391412

36 18.9 23.543609

49 14.4 6.031424

50 19.4 15.922743

51 19.7 20.283239

54 23.4 23.684976

59 23.3 21.671556

60 19.6 20.719591

62 16.0 17.451382

63 22.2 23.743188

65 33.0 22.589801

67 19.4 25.333112

75 24.1 25.570757

78 20.8 23.003678

86 26.6 27.426661

89 23.6 30.626177

94 25.0 28.873639

96 28.4 28.408533

101 27.5 24.513815

105 20.1 21.183578

109 19.8 22.290925

123 20.5 18.968241

124 17.3 13.872235

134 18.4 15.819658

142 14.4 1.836079

153 15.3 22.998216

157 13.1 15.085199

159 24.3 29.878577

160 23.3 27.550928

161 27.0 34.562580

164 50.0 43.205106

166 25.0 25.846604

172 19.1 24.507270

176 29.4 30.990002

180 37.2 32.602061

181 39.8 33.792977

186 29.6 23.731611

187 50.0 35.356959

188 32.0 33.592158

212 19.3 15.994333

213 22.4 22.557913

214 28.1 24.764028

215 23.7 8.494987

226 50.0 39.208864

227 37.6 37.416245

232 31.7 33.098733

234 48.3 36.864297

236 24.0 24.865309

238 31.5 32.639594

239 23.7 28.279134

240 23.3 28.091615

243 22.2 23.411110

244 23.7 27.466826

248 20.5 19.390662

256 20.9 21.663346

257 44.0 37.265336

259 36.0 36.741511

263 48.8 41.127617

264 31.0 34.272637

266 22.8 28.814073

274 35.2 35.637644

275 32.4 37.037781

277 33.2 36.061379

279 29.1 29.985753

284 50.0 45.174617

287 20.1 19.185345

304 33.1 32.731285

313 19.4 22.993456

317 17.8 16.579682

330 22.6 24.763327

350 26.6 22.370600

355 18.2 14.634080

364 16.8 22.268148

365 21.9 39.399151

368 23.1 11.959220

372 50.0 25.659516

375 13.8 -1.349784

376 15.0 25.142170

382 10.9 17.703051

383 11.3 12.981402

385 8.8 2.210588

386 7.2 6.645153

390 11.5 14.183525

410 27.5 18.892348

414 16.3 11.106036

415 7.0 -6.632606

422 14.2 18.700032

423 20.8 18.793342

427 10.2 16.363666

433 16.1 21.866345

437 9.6 14.787339

446 11.8 11.561530

448 12.6 18.552451

464 20.2 23.492107

484 21.8 21.563085

**#Linear SVM**

set.seed(2)

model\_svm<- svm(medv~ .,data=Boston\_train, kernel= 'linear')

summary(model\_svm)

Call:

svm(formula = medv ~ ., data = Boston\_train,

kernel = "linear")

Parameters:

SVM-Type: eps-regression

SVM-Kernel: linear

cost: 1

gamma: 0.07692308

epsilon: 0.1

Number of Support Vectors: 309

**#tuning for finding the best parameters**

set.seed(2)

tune\_svm\_lin<- tune(svm, medv~ ., data= Boston\_train, kernel= "linear", ranges=list(cost=c(0.01,.1,1,5,10)))

summary(tune\_svm\_lin)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:

cost

10

- best performance: 25.2932

- Detailed performance results:

cost error dispersion

1 0.01 26.92046 12.01956

2 0.10 25.66859 11.21275

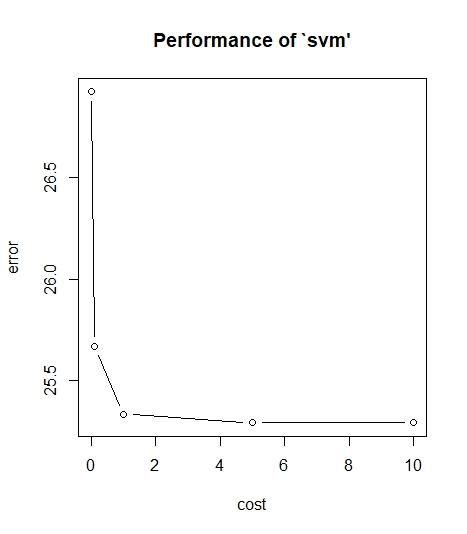
3 1.00 25.33545 11.04718

4 5.00 25.29396 11.00178

5 10.00 25.29320 11.00401

**#plot for tuning**

plot(tune\_svm\_lin)



The error seems to be good.

**Now we make a new Linear SVM model with tuned parameters**

**#Making Linear SVM model with tuned parameters**

set.seed(100)

model\_svm\_1<- svm(medv~ ., data= Boston\_train, kernel= "linear", cost= tune\_svm\_lin$best.parameters$cost)

summary(model\_svm\_1)

Call:

svm(formula = medv ~ ., data = Boston\_train,

kernel = "linear", cost = tune\_svm\_lin$best.parameters$cost)

Parameters:

SVM-Type: eps-regression

SVM-Kernel: linear

cost: 10

gamma: 0.07692308

epsilon: 0.1

Number of Support Vectors: 310

**#predicting the values with on our test set**

test\_svm\_lin<- predict(model\_svm\_1, Boston\_test)

head(test\_svm\_lin)

1 15 17 19 28

28.62775 19.23619 20.43664 16.03825 15.40242

37

21.17409

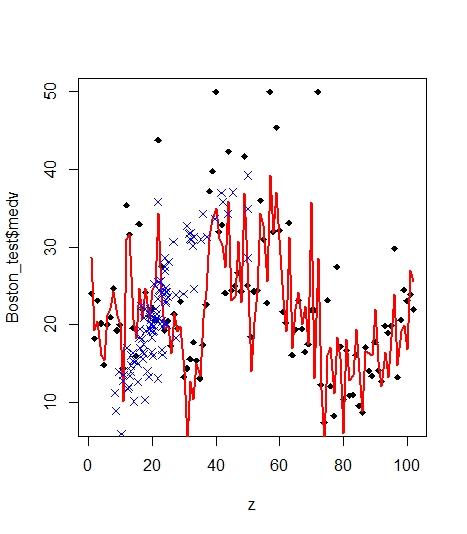
**#Plot showing the values of our model**

z<- 1:length(Boston\_test$medv)

plot(z, Boston\_test$medv, pch=18, col= 'black')

lines(z, test\_svm\_lin, lwd='2', col='red')

points(Boston\_test$medv,test\_svm\_lin,col="blue", pch=4)



**#checking how the model performed**

**#for MSE and other statistics**

library(DMwR)

regr.eval(Boston\_test$medv, test\_svm\_lin)

mae mse rmse mape

3.3271864 23.0390352 4.7998995 0.1493791

**#actual and predicted**

actuals\_pred<- data.frame(cbind(actuals= Boston\_test$medv, predicteds=test\_svm\_lin ))

correlation\_accuracy <- cor(actuals\_pred)

Correlation\_accuracy

actuals predicteds

actuals 1.0000000 0.8631993

predicteds 0.8631993 1.0000000

**# Final Prediction of our test set using Linear SVM with tuned parameters**

actuals\_pred

actuals predicteds

1 24.0 28.627746

15 18.2 19.236189

17 23.1 20.436644

19 20.2 16.038254

28 14.8 15.402418

37 20.0 21.174091

38 21.0 22.063930

44 24.7 24.295116

46 19.3 21.248932

47 20.0 20.058137

49 14.4 10.137850

56 35.4 31.009724

58 31.6 31.686794

60 19.6 20.696434

62 16.0 18.301995

65 33.0 24.661772

68 22.0 20.758293

71 24.2 24.725057

79 21.2 20.804848

88 22.2 24.492070

95 20.6 25.272753

99 43.8 34.303128

101 27.5 23.903506

120 19.3 19.516859

123 20.5 19.863764

124 17.3 16.306838

126 21.4 21.459926

132 19.6 19.218937

133 23.0 19.810039

139 13.3 14.008510

142 14.4 4.934552

144 15.6 12.660498

149 17.8 10.388180

150 15.4 14.996849

157 13.1 13.397564

171 17.4 20.882303

175 22.6 24.356076

180 37.2 31.310237

181 39.8 33.622903

187 50.0 34.942355

188 32.0 31.092874

201 32.9 30.168329

202 24.1 27.460668

203 42.3 35.846681

207 24.4 23.144358

216 25.0 23.615347

221 26.7 30.682717

231 24.3 22.865942

233 41.7 36.947382

237 25.1 28.080698

246 18.5 14.011586

247 24.3 20.276222

251 24.4 24.210947

259 36.0 34.308475

264 31.0 32.752392

266 22.8 25.595248

268 50.0 39.238162

276 32.0 31.861267

281 45.4 37.062663

285 32.2 30.969303

295 21.7 23.338696

298 20.3 19.146272

304 33.1 31.326365

311 16.1 17.031401

313 19.4 21.995445

322 23.1 24.168136

337 19.5 20.059287

343 16.5 22.441162

346 17.5 17.498950

365 21.9 35.825974

367 21.9 13.107568

370 50.0 28.588007

384 12.3 11.741178

388 7.4 4.918494

392 23.2 16.123537

403 12.1 17.061543

404 8.3 11.204422

410 27.5 18.358312

412 17.2 15.656773

418 10.4 6.002973

421 16.7 18.174595

428 10.9 12.794658

429 11.0 13.513316

433 16.1 19.437666

437 9.6 13.991617

438 8.7 8.915251

442 17.1 16.536488

449 14.1 16.366958

451 13.4 16.063460

454 17.8 22.024767

456 14.1 15.341034

457 12.7 12.058295

466 19.9 16.459681

467 19.0 13.295974

471 19.9 18.376489

474 29.8 23.936480

476 13.3 14.775134

488 20.6 19.189972

495 24.5 19.951015

496 23.1 16.794535

504 23.9 26.981222

505 22.0 25.578945

**# Radial SVM**

model\_reg = svm(medv~., data=Boston\_train)

print(model\_reg)

Call:

svm(formula = medv ~ ., data = Boston\_train)

Parameters:

SVM-Type: eps-regression

SVM-Kernel: radial

cost: 1

gamma: 0.07692308

epsilon: 0.1

Number of Support Vectors: 276

**#tuning for finding the best parameters**

set.seed(2)

tune\_in<- tune(svm,medv~ ., data= Boston\_train, kernel= 'radial', ranges= list(cost=c(0.01,.1,1,5,10,100, gamma=c(.01,.1,1,5,10), epsilon=c(.01,.1,1,5,10,100))))

summary(tune\_in)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:

cost gamma

10 0.1

- best performance: 11.3672

- Detailed performance results:

cost gamma error dispersion

1 1e-02 0.01 71.46240 24.094505

2 1e-01 0.01 38.73136 16.647765

3 1e+00 0.01 19.02250 10.956381

4 5e+00 0.01 13.82254 9.476899

5 1e+01 0.01 12.80531 8.959213

6 1e+02 0.01 11.42924 7.334064

7 1e-02 0.10 70.34035 24.198407

8 1e-01 0.10 36.31663 14.696630

9 1e+00 0.10 16.13286 8.011591

10 5e+00 0.10 11.81101 6.025321

11 1e+01 0.10 11.36720 6.207712

12 1e+02 0.10 12.04449 6.333936

13 1e-02 1.00 83.98876 26.488310

14 1e-01 1.00 70.31037 24.259602

15 1e+00 1.00 41.54392 15.912765

16 5e+00 1.00 33.93026 13.326012

17 1e+01 1.00 33.94647 13.337634

18 1e+02 1.00 33.94647 13.337634

19 1e-02 5.00 86.10120 26.731839

20 1e-01 5.00 83.88164 26.418717

21 1e+00 5.00 72.79584 23.892023

22 5e+00 5.00 68.74580 21.682659

23 1e+01 5.00 68.74580 21.682659

24 1e+02 5.00 68.74580 21.682659

25 1e-02 10.00 86.29387 26.754508

26 1e-01 10.00 85.60887 26.695272

27 1e+00 10.00 81.10297 25.486084

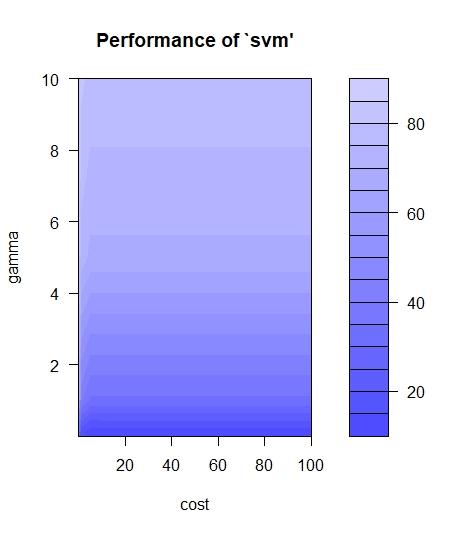
28 5e+00 10.00 78.93765 23.395881

29 1e+01 10.00 78.93765 23.395881

30 1e+02 10.00 78.93765 23.395881

**#plot for tuning**

plot(tune\_in)



The darker the region is ther better our model is(because the RMSE is closer to zero in dark region) i.e we should try another grid with narrower values

**#tuning again with narrower values**

set.seed(2)

tune\_in\_2<- tune(svm,medv~ ., data= Boston\_train, kernel= 'radial', ranges= list(cost=c(0.01,.1,1,5), gamma=c(.01,.02,.1)))

summary(tune\_in\_2)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:

cost gamma

5 0.1

- best performance: 11.93499

- Detailed performance results:

cost gamma error dispersion

1 0.01 0.001 83.85416 23.032389

2 0.10 0.001 67.44644 22.134869

3 1.00 0.001 35.55381 15.265218

4 5.00 0.001 25.75720 11.850417

5 0.01 0.010 71.60184 22.573769

6 0.10 0.010 38.73287 17.106824

7 1.00 0.010 19.08542 10.997803

8 5.00 0.010 13.75738 9.426302

9 0.01 0.020 67.62265 22.656869

10 0.10 0.020 32.55100 15.531679

11 1.00 0.020 16.43944 10.473683

12 5.00 0.020 12.81424 8.907366

13 0.01 0.100 70.48614 23.469406

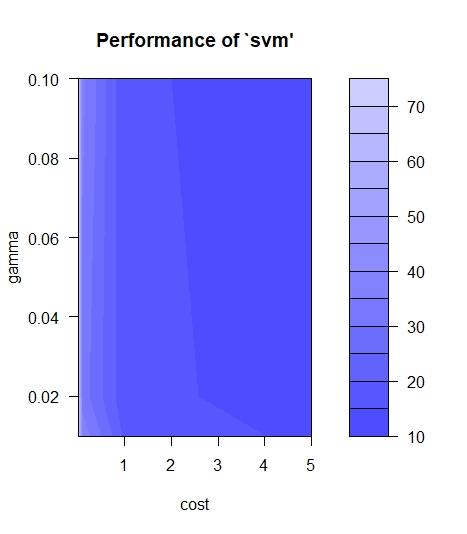
14 0.10 0.100 36.45328 17.473986

15 1.00 0.100 16.02744 10.775593

16 5.00 0.100 11.93499 7.372711

**#plot for tuning with narrower values**

plot(tune\_in\_2)



**This one has pretty good values**

**So now we will make the model with these tuned parameters**

**#Making the Radial SVM with the tuned parameters**

model\_reg\_1 = svm(medv~., data=Boston\_train, cost= tune\_in\_2$best.parameters$cost, gamma= tune\_in\_2$best.parameters$gamma)

summary(model\_reg\_1)

Call:

svm(formula = medv ~ ., data = Boston\_train,

cost = tune\_in\_2$best.parameters$cost,

gamma = tune\_in\_2$best.parameters$gamma)

Parameters:

SVM-Type: eps-regression

SVM-Kernel: radial

cost: 5

gamma: 0.1

epsilon: 0.1

Number of Support Vectors: 269

**#Now we predict for our test set with tuned parameters**

pred\_2<- predict(model\_reg\_1, Boston\_test)

head(pred\_2)

1 15 17 19 28

26.79445 17.82960 21.41105 19.49985 14.43252

37

20.83635

**#Plot showing the values of our model**

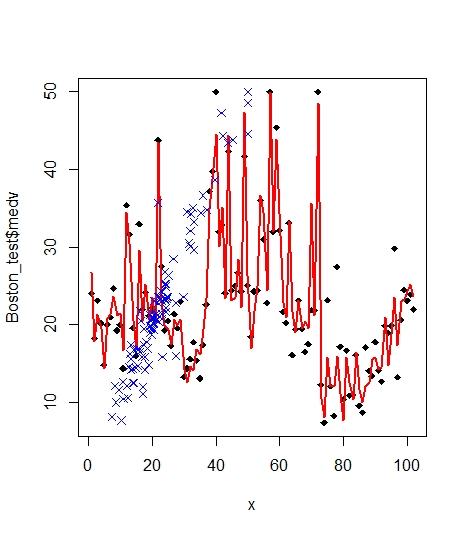
#plotting the points with our test prediction

x<- 1:length(Boston\_test$medv)

plot(x, Boston\_test$medv, pch=18, col= 'black')

lines(x, pred\_2, lwd='2', col='red')

points(Boston\_test$medv,pred\_2,col="blue", pch=4)



We can clearly see that radial SVM plot is better than Linear SVM plot as it covers almost all the points

**#checking how the model performed**

**#MSE and other statistics for our updated model with tuned parameters**

library(DMwR)

regr.eval(Boston\_test$medv, pred\_2)

mae mse rmse mape

1.95081195 8.39005380 2.89655896 0.09770888

**#for actual and predicted correlation**

actuals\_pred<- data.frame(cbind(actuals= Boston\_test$medv, predicteds=pred\_1 ))

correlation\_accuracy <- cor(actuals\_pred)

correlation\_accuracy

actuals predicteds

actuals 1.0000000 0.9256528

predicteds 0.9256528 1.0000000

**# Final Prediction of our test set using radial SVM with tuned parameters**

actuals\_pred

actuals predicteds

1 24.0 27.941877

15 18.2 17.807753

17 23.1 21.079734

19 20.2 19.087113

28 14.8 15.011225

37 20.0 20.936853

38 21.0 21.829590

44 24.7 24.376837

46 19.3 21.367929

47 20.0 21.415293

49 14.4 17.774216

56 35.4 34.440330

58 31.6 31.166294

60 19.6 19.989856

62 16.0 17.257955

65 33.0 28.625950

68 22.0 20.190943

71 24.2 25.132512

79 21.2 20.345291

88 22.2 23.156538

95 20.6 20.850370

99 43.8 40.620972

101 27.5 22.733889

120 19.3 19.901036

123 20.5 19.307021

124 17.3 17.133516

126 21.4 20.591555

132 19.6 19.596216

133 23.0 20.334480

139 13.3 14.404239

142 14.4 12.325253

144 15.6 13.913548

149 17.8 13.525558

150 15.4 15.614095

157 13.1 17.379175

171 17.4 20.744746

175 22.6 23.376392

180 37.2 32.192683

181 39.8 36.458230

187 50.0 40.967666

188 32.0 30.593878

201 32.9 32.955795

202 24.1 23.157850

203 42.3 42.435088

207 24.4 22.725438

216 25.0 23.236031

221 26.7 28.557641

231 24.3 22.113355

233 41.7 45.746756

237 25.1 26.667908

246 18.5 16.910610

247 24.3 20.668223

251 24.4 25.095300

259 36.0 36.317219

264 31.0 35.028920

266 22.8 24.651703

268 50.0 48.947403

276 32.0 32.517621

281 45.4 44.417285

285 32.2 33.576956

295 21.7 22.467438

298 20.3 19.987236

304 33.1 33.622170

311 16.1 22.189882

313 19.4 19.604647

322 23.1 23.227696

337 19.5 20.230878

343 16.5 22.256512

346 17.5 19.188332

365 21.9 32.473065

367 21.9 20.267949

370 50.0 32.647728

384 12.3 10.579867

388 7.4 8.580917

392 23.2 15.317695

403 12.1 12.761132

404 8.3 12.095730

410 27.5 14.181014

412 17.2 11.152562

418 10.4 9.110382

421 16.7 16.603615

428 10.9 11.536478

429 11.0 11.384980

433 16.1 16.870720

437 9.6 11.177415

438 8.7 9.334865

442 17.1 12.851494

449 14.1 13.745888

451 13.4 13.751358

454 17.8 17.892255

456 14.1 13.597658

457 12.7 13.177244

466 19.9 20.790671

467 19.0 13.860758

471 19.9 18.254008

474 29.8 23.338856

476 13.3 13.528473

488 20.6 22.640185

495 24.5 21.938236

496 23.1 21.611579

504 23.9 24.882723

505 22.0 23.342336

**#KNN for regression**

**library(caret)**

**#making the training model with best value of k**

knn\_1<- train(medv ~ ., data = Boston\_train, method = "knn", preProcess = c("center", "scale"))

knn\_1

k-Nearest Neighbors

404 samples

13 predictor

Pre-processing: centered (13), scaled (13)

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 404, 404, 404, 404, 404, 404, ...

Resampling results across tuning parameters:

k RMSE Rsquared MAE

5 5.072968 0.7185519 3.215948

7 5.097911 0.7195252 3.232659

9 5.073360 0.7293292 3.239225

RMSE was used to select the optimal model using

the smallest value.

The final value used for the model was k = 5.

**#predicting the values with KNN on our test set**

knnpredict<- predict(knn\_1, newdata= Boston\_test)

head(knnpredict)

[1] 27.16 17.32 20.58 20.44 14.92 20.22

**#checking how the model performed**

**#for MSE and other statistics**

library(DMwR)

regr.eval(Boston\_test$medv, knnpredict

mae mse rmse mape

3.0110131 19.6764132 4.4358103 0.1325033

**#for actual and predicted correlation**

actuals\_pred<- data.frame(cbind(actuals= Boston\_test$medv, predicteds=knnpredict ))

correlation\_accuracy <- cor(actuals\_pred)

correlation\_accuracy

actuals predicteds

actuals 1.0000000 0.8827484

predicteds 0.8827484 1.0000000

**# Final Prediction of our test set using KNN with k=5**

**actuals\_pred**

actuals predicteds

1 24.0 27.16000

2 18.2 17.32000

3 23.1 20.58000

4 20.2 20.44000

5 14.8 14.92000

6 20.0 20.22000

7 21.0 21.44000

8 24.7 24.54000

9 19.3 22.42000

10 20.0 21.92000

11 14.4 17.54000

12 35.4 30.24000

13 31.6 30.58000

14 19.6 22.06000

15 16.0 19.12000

16 33.0 27.04000

17 22.0 21.91667

18 24.2 23.32000

19 21.2 20.70000

20 22.2 23.40000

21 20.6 22.40000

22 43.8 36.40000

23 27.5 21.34000

24 19.3 20.06000

25 20.5 18.84000

26 17.3 18.16000

27 21.4 18.84000

28 19.6 18.16000

29 23.0 18.16000

30 13.3 15.94000

31 14.4 11.82000

32 15.6 17.38000

33 17.8 17.38000

34 15.4 17.38000

35 13.1 18.44000

36 17.4 22.80000

37 22.6 26.32000

38 37.2 28.56000

39 39.8 34.20000

40 50.0 36.72000

41 32.0 30.84000

42 32.9 31.10000

43 24.1 22.44000

44 42.3 40.96000

45 24.4 23.28000

46 25.0 23.36000

47 26.7 24.58000

48 24.3 21.48000

49 41.7 43.88000

50 25.1 25.12000

51 18.5 18.68000

52 24.3 23.98000

53 24.4 25.74000

54 36.0 34.84000

55 31.0 34.84000

56 22.8 28.12000

57 50.0 43.84000

58 32.0 28.18000

59 45.4 35.64000

60 32.2 31.18000

61 21.7 24.22000

62 20.3 22.00000

63 33.1 30.80000

64 16.1 20.98000

65 19.4 20.22000

66 23.1 22.90000

67 19.5 19.58000

68 16.5 21.84000

69 17.5 20.24000

70 21.9 38.88000

71 21.9 17.90000

72 50.0 32.24000

73 12.3 10.76000

74 7.4 10.28000

75 23.2 15.50000

76 12.1 12.30000

77 8.3 9.90000

78 27.5 14.02000

79 17.2 12.10000

80 10.4 12.34000

81 16.7 13.98000

82 10.9 10.48000

83 11.0 12.70000

84 16.1 12.96000

85 9.6 11.26000

86 8.7 8.86000

87 17.1 14.40000

88 14.1 14.66000

89 13.4 12.58000

90 17.8 14.40000

91 14.1 12.80000

92 12.7 13.12000

93 19.9 21.02000

94 19.0 13.48000

95 19.9 18.64000

96 29.8 22.90000

97 13.3 16.12000

98 20.6 21.60000

99 24.5 20.10000

100 23.1 20.28000

101 23.9 20.84000

102 22.0 20.48000

**Now we perform Linear SVM for classification by making a dummy variable for medv by taking medv > than the median as ‘1’ and medv<median as ‘0’**

**SVM - Linear Kernal:**

**#we read a file with the dummy variable**

dataset <- read.csv(file.choose(), header=TRUE)

head(dataset)

X crim zn indus chas nox rm age dis rad tax ptratio black lstat medv dummy

1 1 0.00632 18 2.31 0 0.538 6.575 65.2 4.0900 1 296 15.3 396.90 4.98 24.0 1

2 2 0.02731 0 7.07 0 0.469 6.421 78.9 4.9671 2 242 17.8 396.90 9.14 21.6 1

3 3 0.02729 0 7.07 0 0.469 7.185 61.1 4.9671 2 242 17.8 392.83 4.03 34.7 1

4 4 0.03237 0 2.18 0 0.458 6.998 45.8 6.0622 3 222 18.7 394.63 2.94 33.4 1

5 5 0.06905 0 2.18 0 0.458 7.147 54.2 6.0622 3 222 18.7 396.90 5.33 36.2 1

6 6 0.02985 0 2.18 0 0.458 6.430 58.7 6.0622 3 222 18.7 394.12 5.21 28.7 1

**> dataset <- dataset[14:16]**

**> head(dataset)**

lstat medv dummy

1 4.98 24.0 1

2 9.14 21.6 1

3 4.03 34.7 1

4 2.94 33.4 1

5 5.33 36.2 1

6 5.21 28.7 1

**> dataset$dummy <- factor(dataset$dummy, levels =c(0,1))**

**> library(caTools)**

**> set.seed(506)**

**> split <- sample.split(dataset$dummy, SplitRatio = 0.75)**

**> training\_set <- subset(dataset, split ==TRUE)**

**> test\_set <- subset(dataset, split ==FALSE)**

**> split**

[1] TRUE TRUE TRUE TRUE FALSE FALSE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE

[16] FALSE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE

[31] TRUE TRUE TRUE FALSE FALSE TRUE TRUE FALSE FALSE FALSE TRUE FALSE FALSE TRUE TRUE

[46] TRUE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE TRUE TRUE

[61] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE

[76] TRUE FALSE TRUE TRUE TRUE TRUE FALSE FALSE TRUE FALSE FALSE TRUE TRUE FALSE TRUE

[91] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE TRUE FALSE TRUE FALSE FALSE

[106] TRUE TRUE FALSE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE

[121] TRUE FALSE TRUE TRUE TRUE FALSE TRUE TRUE FALSE FALSE FALSE FALSE TRUE TRUE TRUE

[136] FALSE TRUE TRUE TRUE TRUE FALSE FALSE TRUE TRUE FALSE TRUE TRUE FALSE TRUE TRUE

[151] FALSE TRUE TRUE FALSE FALSE FALSE TRUE TRUE FALSE FALSE TRUE TRUE TRUE TRUE FALSE

[166] TRUE FALSE TRUE FALSE FALSE TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE TRUE TRUE

[181] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE

[196] FALSE TRUE FALSE TRUE FALSE TRUE FALSE FALSE TRUE TRUE FALSE TRUE TRUE TRUE TRUE

[211] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE FALSE TRUE

[226] TRUE TRUE TRUE TRUE FALSE TRUE TRUE FALSE TRUE FALSE TRUE TRUE TRUE TRUE FALSE

[241] TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE

[256] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE

[271] TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE

[286] TRUE FALSE TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE FALSE TRUE FALSE

[301] TRUE TRUE FALSE TRUE TRUE TRUE TRUE FALSE FALSE FALSE TRUE TRUE FALSE TRUE TRUE

[316] TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[331] TRUE FALSE TRUE FALSE FALSE TRUE FALSE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE

[346] TRUE FALSE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE FALSE TRUE TRUE

[361] TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE FALSE TRUE TRUE

[376] TRUE TRUE TRUE TRUE FALSE FALSE TRUE TRUE TRUE TRUE FALSE FALSE TRUE TRUE TRUE

[391] FALSE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE

[406] TRUE TRUE FALSE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE

[421] TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE

[436] TRUE FALSE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE

[451] TRUE FALSE FALSE FALSE TRUE TRUE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[466] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE TRUE TRUE TRUE TRUE TRUE

[481] TRUE TRUE TRUE FALSE TRUE TRUE FALSE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE

[496] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE

**> training\_set[-3] <- scale(training\_set[-3])**

**> test\_set[-3] <- scale(test\_set[-3])**

**> library(e1071)**

**> classifier <- svm(formula = dummy ~ ., data = training\_set, type = 'C-classification', kernel='linear')**

**> classifier**

Call:

svm(formula = dummy ~ ., data = training\_set, type = "C-classification", kernel = "linear")

Parameters:

SVM-Type: C-classification

SVM-Kernel: linear

cost: 1

Number of Support Vectors: 75

> y\_pred <- predict(classifier, newdata = test\_set[-3])

> y\_pred

5 6 12 16 22 26 30 34 35 38 39 40 42 43 47 48 56 58 61 73 77 82 83

1 1 0 0 0 0 1 0 0 1 1 1 1 1 0 0 1 1 0 1 0 1 1

85 86 89 91 99 100 102 104 105 108 110 120 122 126 129 130 131 132 136 141 142 145 148

1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

151 154 155 156 159 160 165 167 169 170 176 178 192 196 198 200 202 203 206 219 224 230 233

1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

235 240 243 249 255 269 276 280 284 287 289 295 298 300 303 308 309 310 313 320 332 334 335

1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 0 0 1 0 1 1

337 342 347 351 357 358 368 373 380 381 386 387 391 393 400 408 411 420 423 429 437 439 449

0 1 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

450 452 453 454 457 458 474 475 484 487 490 497 504 506

0 0 0 0 0 0 1 0 1 0 0 0 1 0

Levels: 0 1

**> cm <- table(test\_set[, 3], y\_pred)**

**> cm**

y\_pred

0 1

0 50 1

1 0 53

**> set <- training\_set**

**> X1 <- seq(min(set[, 1]) -1, max(set[, 1]) + 1, by =0.01)**

**> X2 <- seq(min(set[, 2]) -1, max(set[, 2]) + 1, by =0.01)**

> grid\_set <- expand.grid(X1, X2)

> colnames(grid\_set) <- c('lstat', 'medv')

> y\_grid <- predict(classifier, newdata = grid\_set)

> plot(set[, 3], main ='SVM (Training set)', xlab = 'lstat', ylab = 'medv', xlim=range(X1), ylim=range(X2))

> plot(set[, 3], main ='SVM (Training set)', xlab = 'lstat', ylab = 'medv', xlim=range(X1), ylim=range(X2))

> contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)

> points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'coral1', 'aquamarine'))

> points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))

>

