**LINEAR REGRESSION:**

Definition:

Linear Regression is a supervised modeling technique for continuous data that generates a response based on the set of input features. It is used for explaining the linear relationship between a single variable Y, called the response (output or dependent variable), and one or more predictor (input, independent or explanatory variables).

It’s a simple regression problem if only a single variable X is considered, otherwise it takes the form of a multiple regression problem, that is if more than one predictor is used in the model.

**R CODE**

library(MASS) : This library is used to get the new data sets (Boston in our case)in R language.

data(): Shows you all the datasets available in the R software.

data(Boston): Gives you the data set in the console

?Boston: Description about the dataset and its variables

View(Boston) : View as table in Console

set.seed(2): .Random.seed is an integer vector, containing the random number generator (RNG) state for random number generation in R. It can be saved and restored, but should not be altered by the user.

library(caTools) : Contains several basic utility functions including: moving (rolling, running) window statistic functions, read/write for GIF and ENVI binary files, fast calculation of AUC, LogitBoost classifier, base64 encoder/decoder, round-off error free sum and cumsum, etc.

split<-sample.split(Boston$medv,SplitRatio = 0.7) :

Used to split the whole dataset into a desired splitRatio datasets.In our case it is 70%.

training\_data <- subset(Boston,split == "TRUE") :

As the split is TRUE we will get a training\_data of 70% Boston data set.

test\_data <- subset(Boston,split == "FALSE"):

As the split is FALSE we will get a test\_data of remaining 30% Boston data set.

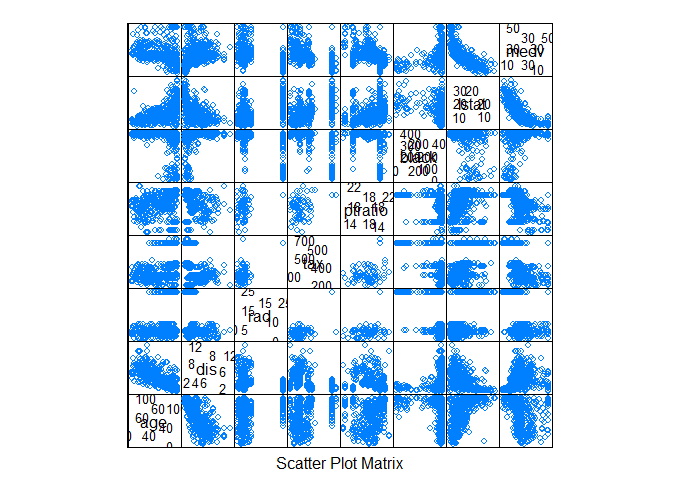
attach(Boston) : The database is attached to the R search path. This means that the database is searched by R when evaluating a variable, so objects in the database can be accessed by simply giving their names.

install.packages("lattice"): The lattice add-on package is an implementation of Trellis graphics for R. It is a powerful and elegant high-level data visualization system with an emphasis on multivariate data. It is designed to meet most typical graphics needs with minimal tuning, but can also be easily extended to handle most nonstandard requirements

splom(~Boston[c(7:14)],groups=NULL,data=Boston,axis.line.tck=0,axis.test.alpha=0):

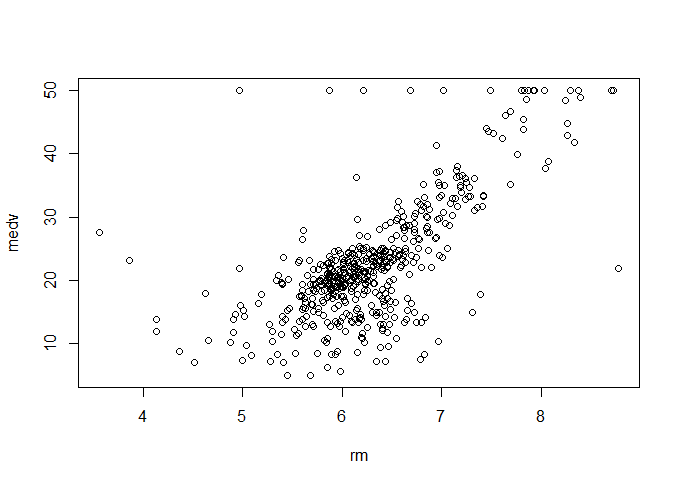
Used to draw Scatter Plot Matrices with given data set.In this command it is a Scatter Plot Matrix of Boston data set with last seven variables.

**splom** – Scatter Plot Matrices



**From this plot we can get to know how each predictor variable is related to the target table**

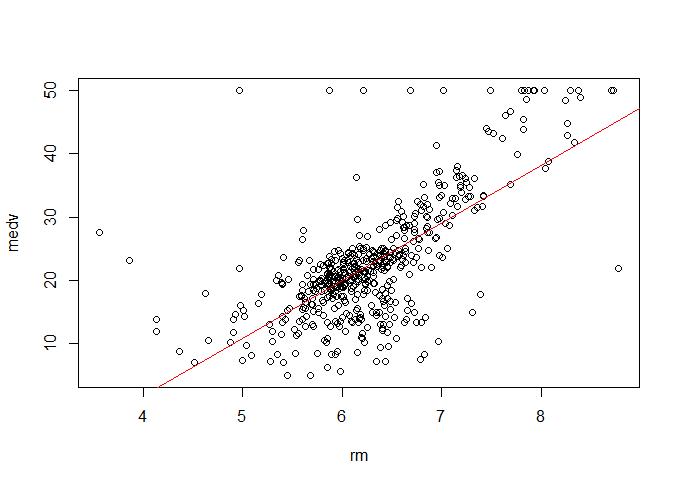
plot(rm,medv): Generic function for plotting of R objects. In this case we are using the rm and medv variables to plot a graph. We can see that instead of using Boston$medv or Boston$rm we are directly using rm and medv as we have attached our Boston dataset we can directly call the variable names.



**From this plot we can get to know the detailed relation between one predictor and one target variable**

abline(lm(medv~rm),col ='red') :

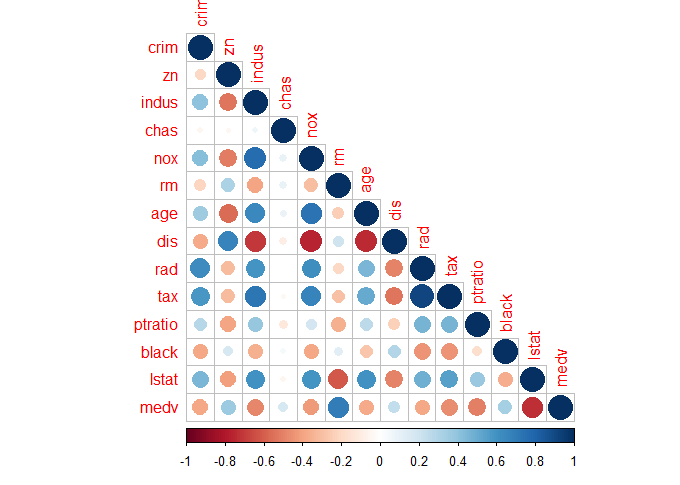
1. **lm** is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although [aov](http://127.0.0.1:27551/help/library/stats/help/aov) may provide a more convenient interface for these).
2. **abline** is used to add one or more straight lines through the current plot.

****

**From this plot we can get to know the detailed relation between one predictor and one target variable along with the absolute.**

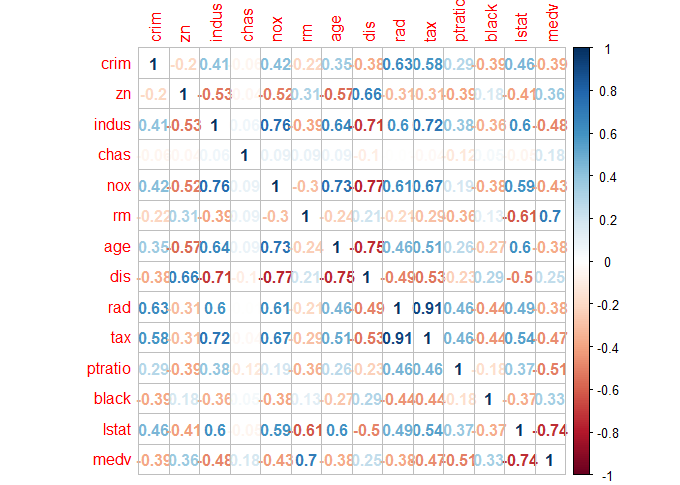
install.packages("corrplot") : The **corrplot** package is a graphical display of a correlation matrix, confidence interval. It also contains some algorithms to do matrix reordering. In addition, corrplot is good at details, including choosing color, text labels, color labels, layout, etc.

corrplot(cr,type="lower"): This function is used to plot the correlation matrix with the set **cr** or any dataset which shows us the lower bound values.



**From the above correlation plot all the circles with the colors near to the dark blue can be said to have collinearity.**

corrplot(cr,method = "number"): This function is used to plot the correlation matrix with numbers showing the extent of the correlation.



**From the correlation plot the variables whose values are nearer to one can be said to have collinearity**

install.packages("caret"): The **caret** package (short for Classification And REgression Training) contains functions to streamline the model training process for complex regression and classification problems.

Bost = subset(Boston,select = -c(medv)):

This function will return a subset of the Boston data which will be selected from the condition i.e.. all the columns except the medv column.

numericData <- Bost[sapply(Bost,is.numeric)]:

**lapply** returns a list of the same length as X, each element of which is the result of applying FUN to the corresponding element of X.

**sapply** is a user-friendly version and wrapper of lapply by default returning a vector, matrix or, if simplify = "array", an array if appropriate, by applying simplify2array(). sapply(x, f, simplify = FALSE, USE.NAMES = FALSE) is the same as lapply(x, f).

So This function will change all the variables in the Boston subset to numeric values.

descror <- cor(numericData): **cor** compute the variance of x and the covariance or correlation of x and y if these are vectors. Ifx and y are matrices then the covariances (or correlations) between the columns of x and the columns of y are computed.

install.packages("car"): This package is installed inorder to run the VIF model.

model<-lm(medv~.,data = training\_data): The lm function creates a linear model with lm with the data equals to training data and medv as the target variable….including all other variables which is denoted by the dot (.) in the syntax.

model: In order to read the model and get to know the intercepts of each variable,

Call:

lm(formula = medv ~ ., data = training\_data)

Coefficients:

(Intercept) crim zn indus chas

41.975065 -0.124709 0.060213 0.007313 2.366739

nox rm age dis rad

-21.354016 3.483738 -0.006623 -1.859822 0.337442

tax ptratio black lstat

-0.012126 -0.886151 0.008751 -0.590932

sumary(model): This will summarize all the required calculations as shows in the console.

Call:

lm(formula = medv ~ ., data = training\_data)

Residuals:

Min 1Q Median 3Q Max

-15.4036 -2.8472 -0.5166 1.8768 24.3219

Coefficients:

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

(Intercept) 41.975065 6.171855 6.801 4.45e-11 \*\*\*

crim -0.124709 0.036198 -3.445 0.000639 \*\*\*

zn 0.060213 0.016620 3.623 0.000334 \*\*\*

indus 0.007313 0.073690 0.099 0.921005

chas 2.366739 1.074791 2.202 0.028308 \*

nox -21.354016 4.722318 -4.522 8.38e-06 \*\*\*

rm 3.483738 0.487656 7.144 5.23e-12 \*\*\*

age -0.006623 0.015952 -0.415 0.678241

dis -1.859822 0.245499 -7.576 3.17e-13 \*\*\*

rad 0.337442 0.076949 4.385 1.53e-05 \*\*\*

tax -0.012126 0.004312 -2.812 0.005201 \*\*

ptratio -0.886151 0.164662 -5.382 1.35e-07 \*\*\*

black 0.008751 0.003099 2.824 0.005009 \*\*

lstat -0.590932 0.060140 -9.826 < 2e-16 \*\*\*

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.914 on 353 degrees of freedom

Multiple R-squared: 0.7408,

Adjusted R-squared: 0.7312

F-statistic: 77.6 on 13 and 353 DF,

p-value: < 2.2e-16

**As we can see from the above summarization we can omit the variables with the p- value lesser than 0.05.In short we need to keep only those variables which have atleast one star.**

vif(model): Calculates variance-inflation and generalized variance-inflation factors for linear, generalized linear, and other models.

crim zn indus chas rm rad age dis

1.654423 2.199178 3.202118 1.057805 1.815010 2.977940 2.995316 3.896029

nox ptratio black lstat

4.480751 1.982718 1.364592 2.641080

model<-lm(medv~ crim + zn + chas + rm + rad + dis + nox + ptratio + black + lstat,data = training\_data):

As we have concluded from the above summarization we have omitted those two variables and created a new model.

summary(model):

Call:

lm(formula = medv ~ crim + zn + chas + rm + rad + dis + nox +

ptratio + black + lstat, data = training\_data)

Residuals:

Min 1Q Median 3Q Max

-16.0923 -3.0470 -0.4139 1.8288 24.0622

Coefficients:

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

(Intercept) 40.350201 6.199887 6.508 2.59e-10 \*\*\*

crim -0.122429 0.036489 -3.355 0.000878 \*\*\*

zn 0.051728 0.016282 3.177 0.001618 \*\*

chas 2.603143 1.073338 2.425 0.015793 \*

rm 3.637570 0.476458 7.635 2.10e-13 \*\*\*

rad 0.166086 0.049779 3.336 0.000938 \*\*\*

dis -1.770642 0.230796 -7.672 1.64e-13 \*\*\*

nox -24.618054 4.363543 -5.642 3.44e-08 \*\*\*

ptratio -0.962238 0.163382 -5.889 8.97e-09 \*\*\*

black 0.009152 0.003120 2.934 0.003566 \*\*

lstat -0.608516 0.057900 -10.510 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.961 on 356 degrees of freedom

Multiple R-squared: 0.7336,

Adjusted R-squared: 0.7261

F-statistic: 98.04 on 10 and 356 DF,

p-value: < 2.2e-16

vif(model):

crim zn chas rm rad dis nox ptratio

1.649807 2.166891 1.050043 1.705751 2.924322 3.380372 3.775964 1.929811

black lstat

1.358641 2.402366

Anova(model): Compute analysis of variance (or deviance) tables for one or more fitted model objects.

Analysis of Variance Table

Response: medv

Df Sum Sq Mean Sq F value Pr(>F)

crim 1 4629.5 4629.5 188.117 < 2.2e-16 \*\*\*

zn 1 3874.7 3874.7 157.447 < 2.2e-16 \*\*\*

chas 1 776.2 776.2 31.540 3.941e-08 \*\*\*

rm 1 8447.3 8447.3 343.252 < 2.2e-16 \*\*\*

rad 1 324.3 324.3 13.179 0.0003246 \*\*\*

dis 1 272.5 272.5 11.073 0.0009675 \*\*\*

nox 1 1112.0 1112.0 45.187 7.134e-11 \*\*\*

ptratio 1 1435.5 1435.5 58.331 2.064e-13 \*\*\*

black 1 537.4 537.4 21.836 4.218e-06 \*\*\*

lstat 1 2718.3 2718.3 110.455 < 2.2e-16 \*\*\*

Residuals 356 8761.0 24.6

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

From the above table ewe can get

DF –Degree of Freedom

SSE – Sum Of Squares

SSM – Sum of Mean Squares

F-value and p-value.

pred<-predict(model,test\_data): **predict** is a generic function for predictions from the results of various model fitting functions. The function invokes particular *methods* which depend on the [class](http://127.0.0.1:27551/help/library/stats/help/class) of the first argument.

It will predict the outcome for all the rows in the test data.

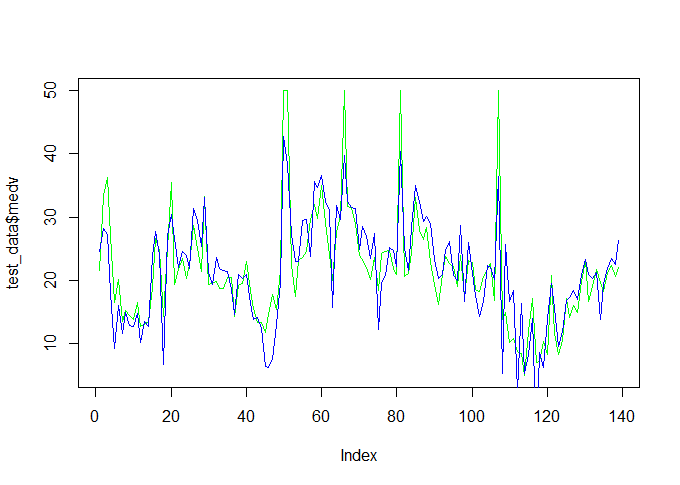
plot(test\_data$medv,type="l",lty = 1,col="green"):

This function is used to plot the graph of the medv variable in our test data with green color.

lines(pred,type='l',col="blue"):

**Lines** - A generic function taking coordinates given in various ways and joining the corresponding points with line segments.

This function is used to draw the graph of the predicted medv value from the test data using predict function on top of the above graph in green color

**From the above graph we can see that the predicted model and the actual are almost in sync.**

**So we can conclude that the predicted model is accurate.**