Omkar Gurav
BE IT
8048

Date : / /
Page No.:

Assignment 1

Answer the following questions

a) Explain various Supervised Learning Algorithms.

→

There are many supervised learning algorithms. Some of them are:-

① Linear regression:-

It is used to estimate real values based on continuous variables. Here, we establish relationship between independent & dependent variables by fitting a best line.

② Logistic regression:-

It is classification algorithm. It is used to estimate discrete values based on given set of independent variables. It predicts the probability of occurence of an event by fitting data to a logit function.

③ Decision Tree:-

It is classification algorithm which is used most of the time for better performance. It works for both categorical & continous dependent variables.

④ Support vector Machine:-

It is classification method. We plot each dataitem as point in n dimensional space with the value of each feature being the value of a particular coordinate.

Date: / /
Page No.:

⑤ Naive Bayes:-

It is classification technique based on Bayes' theorem. Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

⑥ KNN:-

It can be used for both regression & classification. It is simple algorithm that stores all available cases & classifies new cases by majority vote of its k neighbours.

⑦ Random Forest:-

Random Forest is trademark term for an ensemble of decision trees. To classify a new object based on attributes, each ~~attribute~~ tree gives a classification & we say the tree 'votes' for that class. The forest chooses the classification having the most votes.

b. Explain in detail Regression, univariate regression & Multivariate regression.

→

Regression is a type of supervised learning. It is a technique used to model & analyse the relationship between variables & often times they contribute & are related to producing particular outcome together.

Univariate regression:-

The method in the simple case of a single feature is called univariate regression.

Date : / /
Page No.:

For a feature x & target variable Y, regression coefficient is

$$\hat{b} = n\frac{\sigma_{xy}}{n\sigma_{xx}} = \frac{\sigma_{xy}}{\sigma_{xx}}$$ σ_{xx} is variance of variable X.

The covariance is calculated in units of X times units of Y. The variance in units of X squared. So regression coefficient is calculated in units of Y per unit of X.

Univariate linear regression in matrix form as

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} a + \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} b + \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

$$y = a + Xb + \epsilon$$

In second form of this eq^n, y, a, X & ϵ are n-vectors, and b is a scalar.

Multivariate regression:-

Multiple regression simultaneously considers the influence of multiple explanatory variables on a response variable Y. In case of the features, all that changes is that X becomes an n-by-d matrix along with b becomes a d-vector of regression coefficients.

Using homogeneous coordinates to make things easier these eq^n as follows:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

$$y = X \cdot W + \epsilon$$

Date : / /
Page No.:

with X° an n-by-(d+1) matrix whose first column is all 1s & the left over columns are the columns of X, & w has the intercept as its first entry Furthermore the regression coefficients as the left over d entries.

$$y = Xw + \epsilon$$

Here, X having d columns & w having d rows.

$$\hat{w} = (X^T X)^{-1} X^T y$$

Linear_regression.R

```
# Univariate linear regression

   uni_data <-
read.csv('C:/Users/DELL/Downloads/Data_Set_for_Univariate_Regression.csv')

   summary(uni_data)

   # Checking if there are any NA values
   sum(is.na(uni_data))

   # Creating random sample
   index <- sample(1:nrow(uni_data),size = 0.7*nrow(uni_data) )

   # Spliting dataset into training and testing using index
   train <- uni_data[index,]

   test <- uni_data[-index,]

   # Linear regression model for univariate data
   univariate_model <- lm(Y~X, train)

   summary(univariate_model)

   # Predicting on train set
   train_predictions <- predict(univariate_model, data.frame(X = train$X))

   # Predicting on test set
   test_predictions <- predict(univariate_model, data.frame(X = test$X))
```

```
   require("Metrics")



   Train_MSE <- mse(train$Y, train_predictions)
   Train_MSE

   Test_MSE <- mse(test$Y, test_predictions)
   Test_MSE

   xla =  c("Train_Error","Test_Error")
   vec = c(Train_MSE,Test_MSE)

   par(mfrow=c(1,1))
   # Plotting Train MSE and Test MSE
   barplot(vec, names.arg = xla, xlab = 'Errors', main="Training and Testing Error", ylim = c(0,
2000))

   # Plotting regression line for univariate model
   plot(train$X,train$Y,xlab = 'X', ylab = 'Y')
   abline(univariate_model, col = 'red')



# Multivariate linear regression

   multi_data <- read.csv('C:/Users/DELL/Downloads/Regression_Data_set_Batch1.csv')

   summary(multi_data)

   # Checking if there are any NA values
```

```
sum(is.na(multi_data))

# Checking which variables are numerical
str(multi_data)

# Converting character variables into numeric variables
multi_data[,c(6:10,12,13)] <- lapply(multi_data[,c(6:10,12,13)], as.factor )
str(multi_data)

multi_data[,c(6:10,12,13)] <- lapply(multi_data[,c(6:10,12,13)], as.integer )
str(multi_data)

# Correlation matrix of independent variables and dependent variable
cor(multi_data)



# Creating random sample
index_1 <- sample(1:nrow(multi_data),size = 0.7*nrow(multi_data) )

# Spliting dataset into training and testing using index
train_1 <- multi_data[index_1,]

test_1 <- multi_data[-index_1,]

# Linear regression model for multivariate data
multivariate_model_1 <- lm(price ~ area + bathrooms + airconditioning, train_1)
summary(multivariate_model_1)

multivariate_model_2 <- lm(price ~ area + bathrooms + stories, train_1)
summary(multivariate_model_2)
```

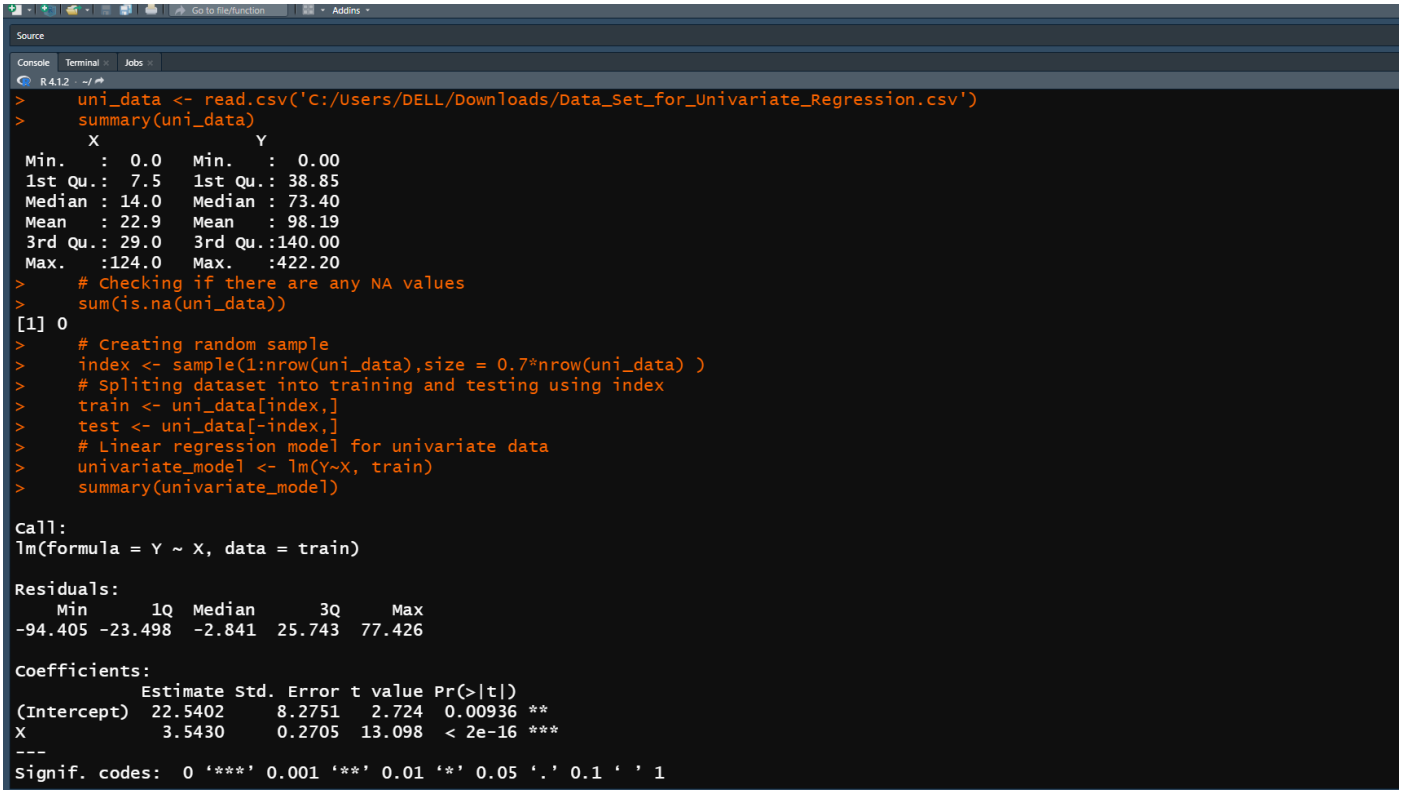
```
multivariate_model_3 <- lm(price ~ area + bathrooms + stories + airconditioning, train_1)
summary(multivariate_model_3)

multivariate_model_4 <- lm(price ~ ., train_1)
summary(multivariate_model_4)



# Predicting on train set
train_predictions_1 <- predict(multivariate_model_1, train_1)

train_predictions_2 <- predict(multivariate_model_2, train_1)

train_predictions_3 <- predict(multivariate_model_3, train_1)

train_predictions_4 <- predict(multivariate_model_4, train_1)

# Predicting on test set
test_predictions_1 <- predict(multivariate_model_1, test_1)

test_predictions_2 <- predict(multivariate_model_2, test_1)

test_predictions_3 <- predict(multivariate_model_3, test_1)

test_predictions_4 <- predict(multivariate_model_4, test_1)



require("Metrics")

Train_MSE_1 <- mse(train_1$price, train_predictions_1)
```

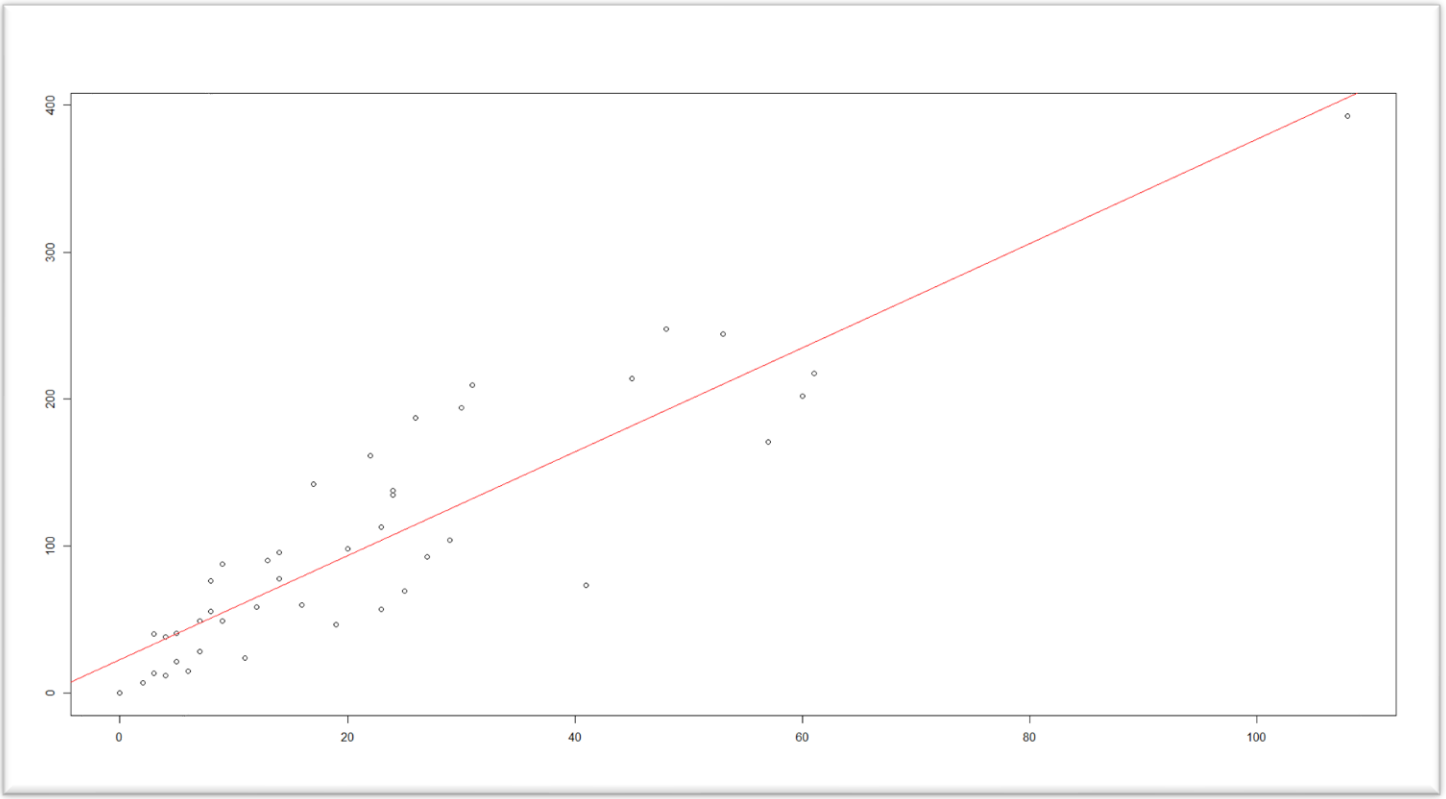
```
Train_MSE_2 <- mse(train_1$price, train_predictions_2)

Train_MSE_3 <- mse(train_1$price, train_predictions_3)

Train_MSE_4 <- mse(train_1$price, train_predictions_4)

c(Train_MSE_1, Train_MSE_2, Train_MSE_3, Train_MSE_4)



Test_MSE_1 <- mse(test_1$price, test_predictions_1)

Test_MSE_2 <- mse(test_1$price, test_predictions_2)

Test_MSE_3 <- mse(test_1$price, test_predictions_3)

Test_MSE_4 <- mse(test_1$price, test_predictions_4)

c(Test_MSE_1, Test_MSE_2, Test_MSE_3, Test_MSE_4)



xla =  c("Train_Error","Test_Error")



vec_1 = c(Train_MSE_1,Test_MSE_1)
vec_2 = c(Train_MSE_2,Test_MSE_2)
vec_3 = c(Train_MSE_3,Test_MSE_3)
vec_4 = c(Train_MSE_4,Test_MSE_4)

# Setting graphical parameters to divide the plotting area into 2 by 2
```

```
par(mfrow=c(2,2))

# Plotting Train MSE and Test MSE
barplot(vec_1, names.arg = xla, xlab = 'Errors', main="Training and Testing Errors of Model 1", ylim = c(0.0e+00,2.0e+12))

barplot(vec_2, names.arg = xla, xlab = 'Errors', main="Training and Testing Errors of Model 2", ylim = c(0.0e+00,2.0e+12))

barplot(vec_3, names.arg = xla, xlab = 'Errors', main="Training and Testing Errors of Model 3", ylim = c(0.0e+00,2.0e+12))

barplot(vec_4, names.arg = xla, xlab = 'Errors', main="Training and Testing Errors of Model 4", ylim = c(0.0e+00,2.0e+12))


library(caret)

# K-fold Cross-Validation for model 4

set.seed(125)

train_control <- trainControl(method = "cv", number = 10)

k_fold_model <- train(price ~ ., data = train_1, method = "lm", trControl = train_control)

print(k_fold_model)
```

Output :

```
R 4.1.2 · ~/
     Min       1Q   Median       3Q      Max
-94.405 -23.498  -2.841  25.743  77.426

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  22.5402     8.2751   2.724  0.00936 **
X             3.5430     0.2705  13.098  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 37.67 on 42 degrees of freedom
Multiple R-squared:  0.8033,    Adjusted R-squared:  0.7986
F-statistic: 171.5 on 1 and 42 DF,  p-value: < 2.2e-16

>      # Predicting on train set
>      train_predictions <- predict(univariate_model, data.frame(X = train$X))
>      # Predicting on test set
>      test_predictions <- predict(univariate_model, data.frame(X = test$X))
>      require("Metrics")
>      Train_MSE <- mse(train$Y, train_predictions)
>      Train_MSE
[1] 1354.532
>      Test_MSE <- mse(test$Y, test_predictions)
>      Test_MSE
[1] 1140.59
>      xla =  c("Train_Error","Test_Error")
>      vec = c(Train_MSE,Test_MSE)
>      par(mfrow=c(1,1))
>      # Plotting Train MSE and Test MSE
>      barplot(vec, names.arg = xla, xlab = 'Errors', main="Training and Testing Error", ylim = c(0, 2000))
>      # Plotting regression line for univariate model
>      plot(train$X,train$Y,xlab = 'X', ylab = 'Y')
>      abline(univariate_model, col = 'red')
>
```






```
File  Edit  Code  View  Plots  Session  Build  Debug  Profile  Tools  Help
Source
Console  Terminal  Jobs
R 4.1.2  ~/

>     multi_data <- read.csv('C:/Users/DELL/Downloads/Regression_Data_set_Batch1.csv')
>     summary(multi_data)
     price              area          bedrooms       bathrooms        stories        mainroad          guestroom          basement
 Min.   : 1750000   Min.   : 1650   Min.   :1.000   Min.   :1.000   Min.   :1.000   Length:545         Length:545         Length:545
 1st Qu.: 3430000   1st Qu.: 3600   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000   Class :character   Class :character   Class :characte
 Median : 4340000   Median : 4600   Median :3.000   Median :1.000   Median :2.000   Mode  :character   Mode  :character   Mode  :characte
 Mean   : 4766729   Mean   : 5151   Mean   :2.965   Mean   :1.286   Mean   :1.806
 3rd Qu.: 5740000   3rd Qu.: 6360   3rd Qu.:3.000   3rd Qu.:2.000   3rd Qu.:2.000
 Max.   :13300000   Max.   :16200   Max.   :6.000   Max.   :4.000   Max.   :4.000
 hotwaterheating    airconditioning       parking         prefarea         furnishingstatus
 Length:545         Length:545         Min.   :0.0000   Length:545         Length:545
 Class :character   Class :character   1st Qu.:0.0000   Class :character   Class :character
 Mode  :character   Mode  :character   Median :0.0000   Mode  :character   Mode  :character
                                       Mean   :0.6936
                                       3rd Qu.:1.0000
                                       Max.   :3.0000
>     # Checking if there are any NA values
>     sum(is.na(multi_data))
[1] 0
>     # Checking which variables are numerical
>     str(multi_data)
'data.frame':   545 obs. of  13 variables:
 $ price           : int  13300000 12250000 12250000 12215000 11410000 10850000 10150000 10150000 9870000 9800000 ...
 $ area            : int  7420 8960 9960 7500 7420 7500 8580 16200 8100 5750 ...
 $ bedrooms        : int  4 4 3 4 4 3 4 5 4 3 ...
 $ bathrooms       : int  2 4 2 2 1 3 3 3 1 2 ...
 $ stories         : int  3 4 2 2 2 1 4 2 2 4 ...
 $ mainroad        : chr  "yes" "yes" "yes" "yes" ...
 $ guestroom       : chr  "no" "no" "no" "no" ...
 $ basement        : chr  "no" "no" "yes" "yes" ...
 $ hotwaterheating : chr  "no" "no" "no" "no" ...
 $ airconditioning : chr  "yes" "yes" "no" "yes" ...
 $ parking         : int  2 3 2 3 2 2 2 0 2 1 ...
 $ prefarea        : chr  "yes" "no" "yes" "yes" ...
```

```
>     # Converting character variables into numeric variables
>     multi_data[,c(6:10,12,13)] <- lapply(multi_data[,c(6:10,12,13)], as.factor )
>     str(multi_data)
'data.frame':   545 obs. of  13 variables:
 $ price           : int  13300000 12250000 12250000 12215000 11410000 10850000 10150000 10150000 9870000 9800000 ...
 $ area            : int  7420 8960 9960 7500 7420 7500 8580 16200 8100 5750 ...
 $ bedrooms        : int  4 4 3 4 4 3 4 5 4 3 ...
 $ bathrooms       : int  2 4 2 2 1 3 3 3 1 2 ...
 $ stories         : int  3 4 2 2 2 1 4 2 2 4 ...
 $ mainroad        : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ guestroom       : Factor w/ 2 levels "no","yes": 1 1 1 1 2 1 1 1 2 2 ...
 $ basement        : Factor w/ 2 levels "no","yes": 1 1 2 2 2 2 1 1 2 1 ...
 $ hotwaterheating : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ airconditioning : Factor w/ 2 levels "no","yes": 2 2 1 2 2 2 2 1 2 2 ...
 $ parking         : int  2 3 2 3 2 2 2 0 2 1 ...
 $ prefarea        : Factor w/ 2 levels "no","yes": 2 1 2 2 1 2 2 1 2 2 ...
 $ furnishingstatus: Factor w/ 3 levels "furnished","semi-furnished",..: 1 1 2 1 1 2 2 3 1 3 ...
>     multi_data[,c(6:10,12,13)] <- lapply(multi_data[,c(6:10,12,13)], as.integer )
>     str(multi_data)
'data.frame':   545 obs. of  13 variables:
 $ price           : int  13300000 12250000 12250000 12215000 11410000 10850000 10150000 10150000 9870000 9800000 ...
 $ area            : int  7420 8960 9960 7500 7420 7500 8580 16200 8100 5750 ...
 $ bedrooms        : int  4 4 3 4 4 3 4 5 4 3 ...
 $ bathrooms       : int  2 4 2 2 1 3 3 3 1 2 ...
 $ stories         : int  3 4 2 2 2 1 4 2 2 4 ...
 $ mainroad        : int  2 2 2 2 2 2 2 2 2 2 ...
 $ guestroom       : int  1 1 1 1 2 1 1 1 2 2 ...
 $ basement        : int  1 1 2 2 2 2 1 1 2 1 ...
 $ hotwaterheating : int  1 1 1 1 1 1 1 1 1 1 ...
 $ airconditioning : int  2 2 1 2 2 2 2 1 2 2 ...
 $ parking         : int  2 3 2 3 2 2 2 0 2 1 ...
 $ prefarea        : int  2 1 2 2 1 2 2 1 2 2 ...
 $ furnishingstatus: int  1 1 2 1 1 2 2 3 1 3 ...
>     # Correlation matrix of independent variables and dependent variable
```

```
>     # Correlation matrix of independent variables and dependent variable
>     cor(multi_data)
                     price       area    bedrooms   bathrooms     stories    mainroad   guestroom     basement hotwaterheating
price            1.00000000  0.535997346  0.36649403  0.51754534  0.42071237  0.29689849  0.25551729  0.187056598     0.093072844
area             0.53599735  1.000000000  0.15185849  0.19381953  0.08399605  0.28887411  0.14029659  0.047416989    -0.009229236
bedrooms         0.36649403  0.151858486  1.00000000  0.37393024  0.40856424 -0.01203324  0.08054870  0.097312424     0.046048887
bathrooms        0.51754534  0.193819531  0.37393024  1.00000000  0.32616471  0.04239762  0.12646884  0.102105706     0.067159096
stories          0.42071237  0.083996051  0.40856424  0.32616471  1.00000000  0.12170613  0.04353767 -0.172393617     0.018846511
mainroad         0.29689849  0.288874114 -0.01203324  0.04239762  0.12170613  1.00000000  0.09233692  0.044002081    -0.011781490
guestroom        0.25551729  0.140296590  0.08054870  0.12646884  0.04353767  0.09233692  1.00000000  0.372065708    -0.010307884
basement         0.18705660  0.047416989  0.09731242  0.10210571 -0.17239362  0.04400208  0.37206571  1.000000000     0.004384836
hotwaterheating  0.09307284 -0.009229236  0.04604889  0.06715910  0.01884651 -0.01178149 -0.01030788  0.004384836     1.000000000
airconditioning  0.45295408  0.222393104  0.16060326  0.18691503  0.29360200  0.10542300  0.13817877  0.047341189    -0.130022833
parking          0.38439365  0.352980481  0.13926990  0.17749582  0.04554709  0.20443255  0.03746575  0.051497175     0.067863888
prefarea         0.32977705  0.234778798  0.07902306  0.06347174  0.04442487  0.19987578  0.16089694  0.228082853    -0.059411382
furnishingstatus -0.30472146 -0.171445361 -0.12324400 -0.14355950 -0.10467233 -0.15672586 -0.11832757 -0.112830732    -0.031628204
                 airconditioning     parking    prefarea furnishingstatus
price                 0.45295408  0.38439365  0.32977705       -0.3047215
area                  0.22239310  0.35298048  0.23477880       -0.1714454
bedrooms              0.16060326  0.13926990  0.07902306       -0.1232440
bathrooms             0.18691503  0.17749582  0.06347174       -0.1435595
stories               0.29360200  0.04554709  0.04442487       -0.1046723
mainroad              0.10542300  0.20443255  0.19987578       -0.1567259
guestroom             0.13817877  0.03746575  0.16089694       -0.1183276
basement              0.04734119  0.05149718  0.22808285       -0.1128307
hotwaterheating      -0.13002283  0.06786389 -0.05941138       -0.0316282
airconditioning       1.00000000  0.15917268  0.11738210       -0.1504773
parking               0.15917268  1.00000000  0.09162706       -0.1775386
prefarea              0.11738210  0.09162706  1.00000000       -0.1076860
furnishingstatus     -0.15047729 -0.17753861 -0.10768597        1.0000000
>     # Creating random sample
>     index_1 <- sample(1:nrow(multi_data),size = 0.7*nrow(multi_data) )
>     # Spliting dataset into training and testing using index
>     train_1 <- multi_data[index_1,]
```

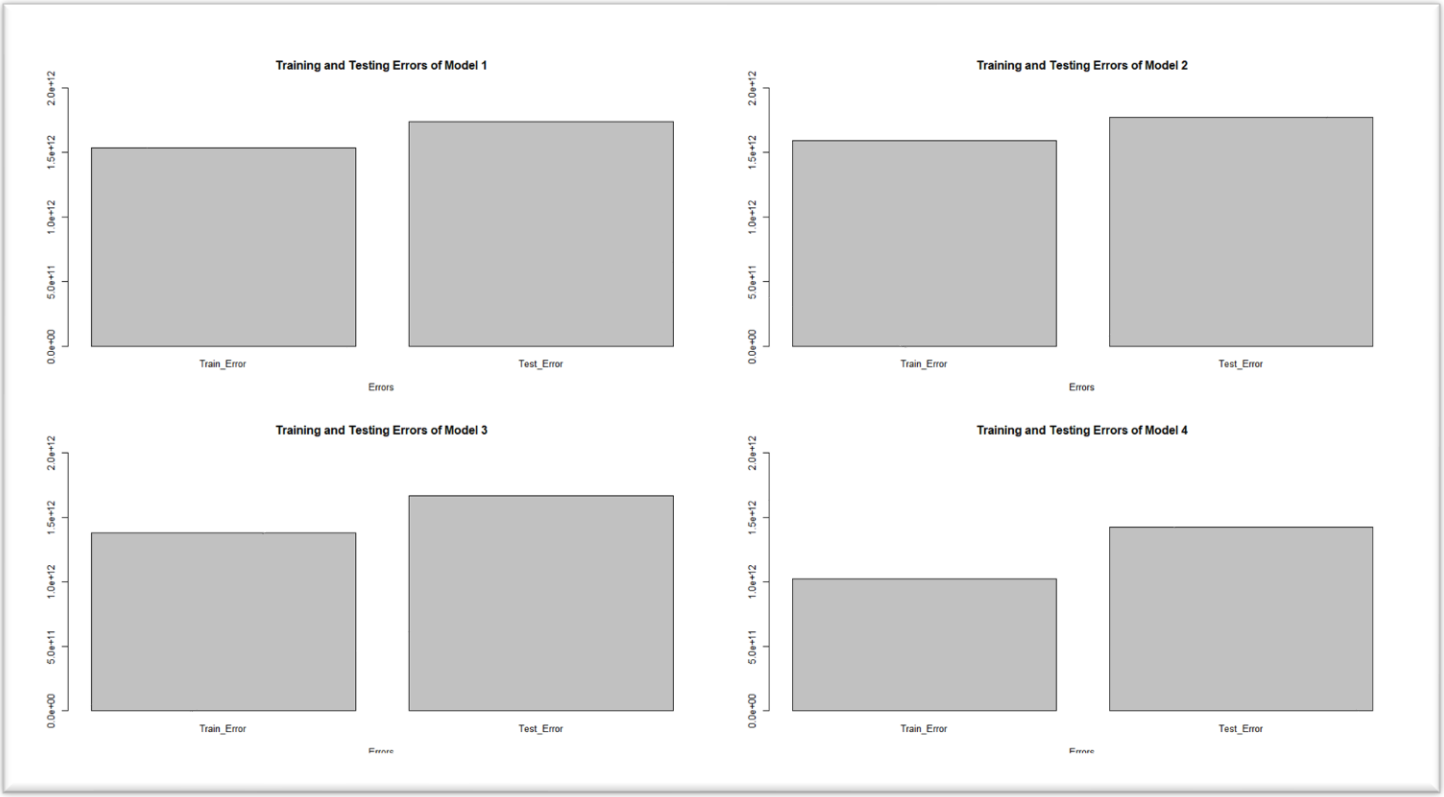
```
>     test_1 <- multi_data[-index_1,]
>     # Linear regression model for multivariate data
>     multivariate_model_1 <- lm(price ~ area + bathrooms + airconditioning, train_1)
>     summary(multivariate_model_1)

Call:
lm(formula = price ~ area + bathrooms + airconditioning, data = train_1)

Residuals:
     Min       1Q   Median       3Q      Max
-3436356  -790866  -141484   590780  5897490

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     -418335.9   257108.4  -1.627    0.105
area               346.9       32.9  10.546   <2e-16 ***
bathrooms       1316012.2   131038.2  10.043   <2e-16 ***
airconditioning 1307254.3   143901.2   9.084   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1245000 on 377 degrees of freedom
Multiple R-squared:  0.5331,    Adjusted R-squared:  0.5294
F-statistic: 143.5 on 3 and 377 DF,  p-value: < 2.2e-16

>     multivariate_model_2 <- lm(price ~ area + bathrooms + stories, train_1)
>     summary(multivariate_model_2)

Call:
lm(formula = price ~ area + bathrooms + stories, data = train_1)

Residuals:
     Min       1Q   Median       3Q      Max
-3696417  -785287   -66463   512689  6014158
```

```
>     summary(multivariate_model_2)

Call:
lm(formula = price ~ area + bathrooms + stories, data = train_1)

Residuals:
     Min       1Q   Median       3Q      Max
-3696417  -785287   -66463   512689  6014158

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 4.246e+04  2.432e+05   0.175    0.861
area        4.170e+02  3.258e+01  12.802  < 2e-16 ***
bathrooms   1.119e+06  1.394e+05   8.028 1.27e-14 ***
stories     6.370e+05  7.860e+04   8.104 7.48e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1269000 on 377 degrees of freedom
Multiple R-squared:  0.5154,    Adjusted R-squared:  0.5115
F-statistic: 133.6 on 3 and 377 DF,  p-value: < 2.2e-16

>     multivariate_model_3 <- lm(price ~ area + bathrooms + stories + airconditioning, train_1)
>     summary(multivariate_model_3)

Call:
lm(formula = price ~ area + bathrooms + stories + airconditioning,
    data = train_1)

Residuals:
     Min       1Q   Median       3Q      Max
-3165151  -727591  -101104   563659  5623754

Coefficients:
```

```
> summary(multivariate_model_3)

Call:
lm(formula = price ~ area + bathrooms + stories + airconditioning,
    data = train_1)

Residuals:
     Min       1Q   Median       3Q      Max
-3165151  -727591  -101104   563659  5623754

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     -745206.17  249528.62  -2.986  0.00301 **
area                359.38      31.32  11.474  < 2e-16 ***
bathrooms       1071982.95  130151.83   8.236 2.97e-15 ***
stories          488829.52   75871.19   6.443 3.59e-10 ***
airconditioning 1072214.16  141524.37   7.576 2.79e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1183000 on 376 degrees of freedom
Multiple R-squared:  0.5795,    Adjusted R-squared:  0.5751
F-statistic: 129.6 on 4 and 376 DF,  p-value: < 2.2e-16

> multivariate_model_4 <- lm(price ~ ., train_1)
> summary(multivariate_model_4)

Call:
lm(formula = price ~ ., data = train_1)

Residuals:
     Min       1Q   Median       3Q      Max
-2621497  -652546   -39612   529878  5163065
```

```
> summary(multivariate_model_4)

Call:
lm(formula = price ~ ., data = train_1)

Residuals:
     Min       1Q   Median       3Q      Max
-2621497  -652546   -39612   529878  5163065

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)      -3.067e+06  5.438e+05  -5.640 3.39e-08 ***
area              2.346e+02  3.041e+01   7.714 1.15e-13 ***
bedrooms          1.709e+05  8.405e+04   2.034 0.042696 *
bathrooms         7.882e+05  1.207e+05   6.532 2.16e-10 ***
stories           4.842e+05  7.525e+04   6.435 3.86e-10 ***
mainroad          4.390e+05  1.563e+05   2.808 0.005250 **
guestroom         5.496e+05  1.545e+05   3.557 0.000425 ***
basement          3.385e+05  1.288e+05   2.628 0.008947 **
hotwaterheating   5.140e+05  2.687e+05   1.913 0.056501 .
airconditioning   9.128e+05  1.272e+05   7.179 3.91e-12 ***
parking           2.537e+05  6.844e+04   3.707 0.000242 ***
prefarea          6.690e+05  1.366e+05   4.898 1.45e-06 ***
furnishingstatus -2.005e+05  7.233e+04  -2.772 0.005863 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1028000 on 368 degrees of freedom
Multiple R-squared:  0.6892,    Adjusted R-squared:  0.6791
F-statistic: 68.01 on 12 and 368 DF,  p-value: < 2.2e-16

> # Predicting on train set
> train_predictions_1 <- predict(multivariate_model_1, train_1)
> train_predictions_2 <- predict(multivariate_model_2, train_1)
```

```
parking           2.537e+05  6.844e+04   3.707 0.000242 ***
prefarea          6.690e+05  1.366e+05   4.898 1.45e-06 ***
furnishingstatus -2.005e+05  7.233e+04  -2.772 0.005863 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1028000 on 368 degrees of freedom
Multiple R-squared:  0.6892,    Adjusted R-squared:  0.6791
F-statistic: 68.01 on 12 and 368 DF,  p-value: < 2.2e-16

>     # Predicting on train set
>     train_predictions_1 <- predict(multivariate_model_1, train_1)
>     train_predictions_2 <- predict(multivariate_model_2, train_1)
>     train_predictions_3 <- predict(multivariate_model_3, train_1)
>     train_predictions_4 <- predict(multivariate_model_4, train_1)
>     # Predicting on test set
>     test_predictions_1 <- predict(multivariate_model_1, test_1)
>     test_predictions_2 <- predict(multivariate_model_2, test_1)
>     test_predictions_3 <- predict(multivariate_model_3, test_1)
>     test_predictions_4 <- predict(multivariate_model_4, test_1)
>     require("Metrics")
>     Train_MSE_1 <- mse(train_1$price, train_predictions_1)
>     Train_MSE_2 <- mse(train_1$price, train_predictions_2)
>     Train_MSE_3 <- mse(train_1$price, train_predictions_3)
>     Train_MSE_4 <- mse(train_1$price, train_predictions_4)
>     c(Train_MSE_1, Train_MSE_2, Train_MSE_3, Train_MSE_4)
[1] 1.534651e+12 1.593050e+12 1.382069e+12 1.021552e+12
>     Test_MSE_1 <- mse(test_1$price, test_predictions_1)
>     Test_MSE_2 <- mse(test_1$price, test_predictions_2)
>     Test_MSE_3 <- mse(test_1$price, test_predictions_3)
>     Test_MSE_4 <- mse(test_1$price, test_predictions_4)
>     c(Test_MSE_1, Test_MSE_2, Test_MSE_3, Test_MSE_4)
[1] 1.739848e+12 1.773202e+12 1.670386e+12 1.426217e+12
>
```



```
>     xla =  c("Train_Error","Test_Error")
>     vec_1 = c(Train_MSE_1,Test_MSE_1)
>     vec_2 = c(Train_MSE_2,Test_MSE_2)
>     vec_3 = c(Train_MSE_3,Test_MSE_3)
>     vec_4 = c(Train_MSE_4,Test_MSE_4)
>     # Setting graphical parameters to divide the plotting area into 2 by 2
>     par(mfrow=c(2,2))
>     # Plotting Train MSE and Test MSE
>     barplot(vec_1, names.arg = xla, xlab = 'Errors', main="Training and Testing Errors of Model 1", ylim = c(0.0e+00,2.0e+12))
>     barplot(vec_2, names.arg = xla, xlab = 'Errors', main="Training and Testing Errors of Model 2", ylim = c(0.0e+00,2.0e+12))
>     barplot(vec_3, names.arg = xla, xlab = 'Errors', main="Training and Testing Errors of Model 3", ylim = c(0.0e+00,2.0e+12))
>     barplot(vec_4, names.arg = xla, xlab = 'Errors', main="Training and Testing Errors of Model 4", ylim = c(0.0e+00,2.0e+12))
>
```

File Edit Code View Plots Session Build Debug Profile Tools Help

Source

Console Terminal Jobs

R 4.1.2 ~/

```
>     library(caret)
Loading required package: ggplot2
Loading required package: lattice

Attaching package: 'caret'

The following objects are masked from 'package:Metrics':

    precision, recall

Warning messages:
1: In doTryCatch(return(expr), name, parentenv, handler) :
  display list redraw incomplete
2: In doTryCatch(return(expr), name, parentenv, handler) :
  invalid graphics state
3: In doTryCatch(return(expr), name, parentenv, handler) :
  invalid graphics state
>     set.seed(125)
>     train_control <- trainControl(method = "cv", number = 10)
>     k_fold_model <- train(price ~ ., data = train_1, method = "lm", trControl = train_control)
>     print(k_fold_model)
Linear Regression

381 samples
 12 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 343, 342, 343, 341, 343, 344, ...
Resampling results:

  RMSE     Rsquared   MAE
  1034765  0.6962721  770351.3
```