Multivariate Linear Regression

Varun Bansal

2023-03-07

Setting the work directory.

Loading and attaching all the necessary packages.

```
#Load packages

if(!require(tinytex)){
  install.packages("tinytex")
  library("tinytex")
}
```

```
## Loading required package: tinytex
```

```
if (!require(lattice)) {
  install.packages("lattice")
  library(lattice)
}
```

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

```
if (!require(gridExtra)) {
  install.packages("gridExtra")
  library(gridExtra)
}
```

```
## Loading required package: gridExtra
```

```
if(!require(corrgram)){install.packages("corrgram")}
```

```
## Loading required package: corrgram

##
## Attaching package: 'corrgram'

## The following object is masked from 'package:lattice':
##
##     panel.fill
```

```
library("corrgram")

if (!require(corrplot)) {
  install.packages("corrplot")
  library(corrplot)
}
```

```
## Loading required package: corrplot
```

```
## corrplot 0.92 loaded
```

Reading file

```
# Reading File
df <- read.csv("Mail_Order_Delivery_Time.txt", header = TRUE, sep = ",")
head(df, 5)
```

```
##     DL  VN PG CS   ML DM HZ       CR  WT
## 1  8.1 324  5 13  313  C  N  Sup Del 216
## 2  8.4 135  2 13  830  I  N  Sup Del 160
## 3  8.6 391  3 12  304  C  N  Sup Del  25
## 4 11.3 245  6  7 1258  C  N  Sup Del  67
## 5  5.4 321  1  2  221  C  N Def Post  14
```

Preliminary and Exploratory

Renaming all variables with my initials appended

```
# Appending initials to all variables in the data frame

df_VB <- df
colnames(df_VB) <- paste(colnames(df_VB), "VB", sep = "_")
head(df_VB, 5)
```

```
##   DL_VB VN_VB PG_VB CS_VB ML_VB DM_VB HZ_VB    CR_VB WT_VB
## 1   8.1   324     5    13   313     C     N  Sup Del   216
## 2   8.4   135     2    13   830     I     N  Sup Del   160
## 3   8.6   391     3    12   304     C     N  Sup Del    25
## 4  11.3   245     6     7  1258     C     N  Sup Del    67
## 5   5.4   321     1     2   221     C     N Def Post    14
```

Looking for outliers in the data

Exploring the summary of dataset

```
summary(df_VB)
```

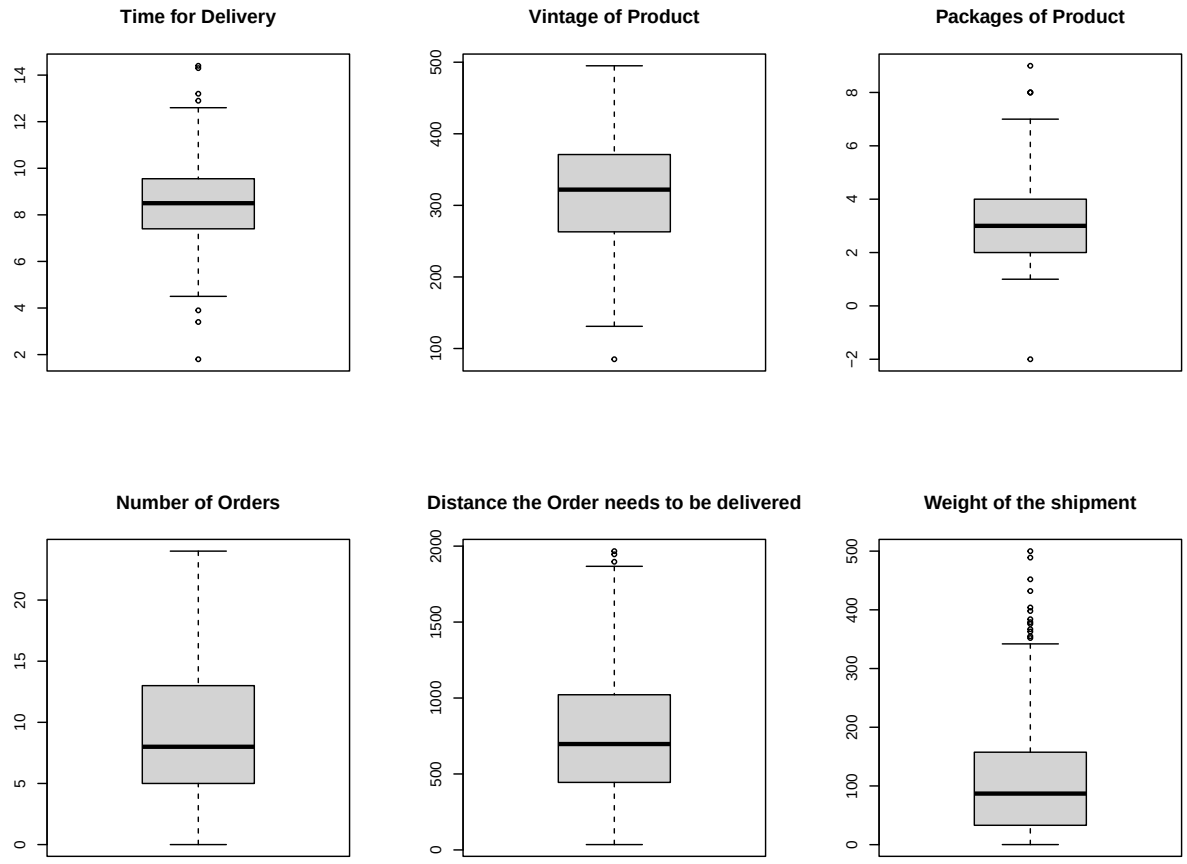
```
##      DL_VB            VN_VB           PG_VB            CS_VB
##  Min.   : 1.800   Min.   : 85.0   Min.   :-2.000   Min.   : 0.000
##  1st Qu.: 7.400   1st Qu.:263.0   1st Qu.: 2.000   1st Qu.: 5.000
##  Median : 8.500   Median :322.0   Median : 3.000   Median : 8.000
##  Mean   : 8.464   Mean   :318.6   Mean   : 2.951   Mean   : 9.228
##  3rd Qu.: 9.550   3rd Qu.:371.0   3rd Qu.: 4.000   3rd Qu.:13.000
##  Max.   :14.400   Max.   :495.0   Max.   : 9.000   Max.   :24.000
##      ML_VB          DM_VB              HZ_VB              CR_VB
##  Min.   :  35.0   Length:487         Length:487         Length:487
##  1st Qu.: 444.5   Class :character   Class :character   Class :character
##  Median : 697.0   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 754.0
##  3rd Qu.:1021.5
##  Max.   :1967.0
##      WT_VB
##  Min.   :  0.1
##  1st Qu.: 33.0
##  Median : 87.0
##  Mean   :107.1
##  3rd Qu.:157.5
##  Max.   :500.0
```

Most of the columns look fine, but PG Column i.e packages of product have been ordered has a negative value. Packages of products ordered cannot be less than 0.

Creating Boxplots

Lets also check this through plotting boxplots.

```
# Boxplots
par(mfrow=c(2,3))
boxplot(df_VB$DL_VB, main = "Time for Delivery")
boxplot(df_VB$VN_VB, main = "Vintage of Product")
boxplot(df_VB$PG_VB, main = "Packages of Product")
boxplot(df_VB$CS_VB, main = "Number of Orders")
boxplot(df_VB$ML_VB, main = "Distance the Order needs to be delivered")
boxplot(df_VB$WT_VB, main = "Weight of the shipment")
```




```
par(mfrow=c(1,1))
```

The box plots show the same, packages of product have been ordered has a negative value

```
#Checking the row
subset(df_VB, PG_VB <0)
```

```
##     DL_VB VN_VB PG_VB CS_VB ML_VB DM_VB HZ_VB   CR_VB WT_VB
## 159   8.7   276    -2     9   671     C     H Sup Del   115
```

Removing the value

```
df_VB <- df_VB[!(df_VB$PG_VB < 0), ]
```

For other columns there are some outliers, but there is no reason to remove them. For example, some of the packages can be very heavy than the normal packages.

Comparing if one Carrier has faster delivery times than the other

We will use two-sample t-test to compare the average delivery times for each Carrier.

```r
# Two-sample t-test
t.test(DL_VB ~ CR_VB, data = df_VB)
```

```
##
##  Welch Two Sample t-test
##
## data:  DL_VB by CR_VB
## t = -6.9608, df = 440.63, p-value = 0.00000000001234
## alternative hypothesis: true difference in means between group Def Post and group Sup Del is not equa
## 95 percent confidence interval:
##  -1.3525268 -0.7569259
## sample estimates:
## mean in group Def Post  mean in group Sup Del
##               7.845274               8.900000
```

Here p-value is significant less than 0.05. This indicates that there is evidence to suggest that one Carrier has faster delivery times than the other.

On average, the delivery time of Def Post is faster than that of Sup Del. Def Post has an average delivery time of 7.84, while Sup Del has an average delivery time of 8.90.

Splitting the dataframe into a training and a test file

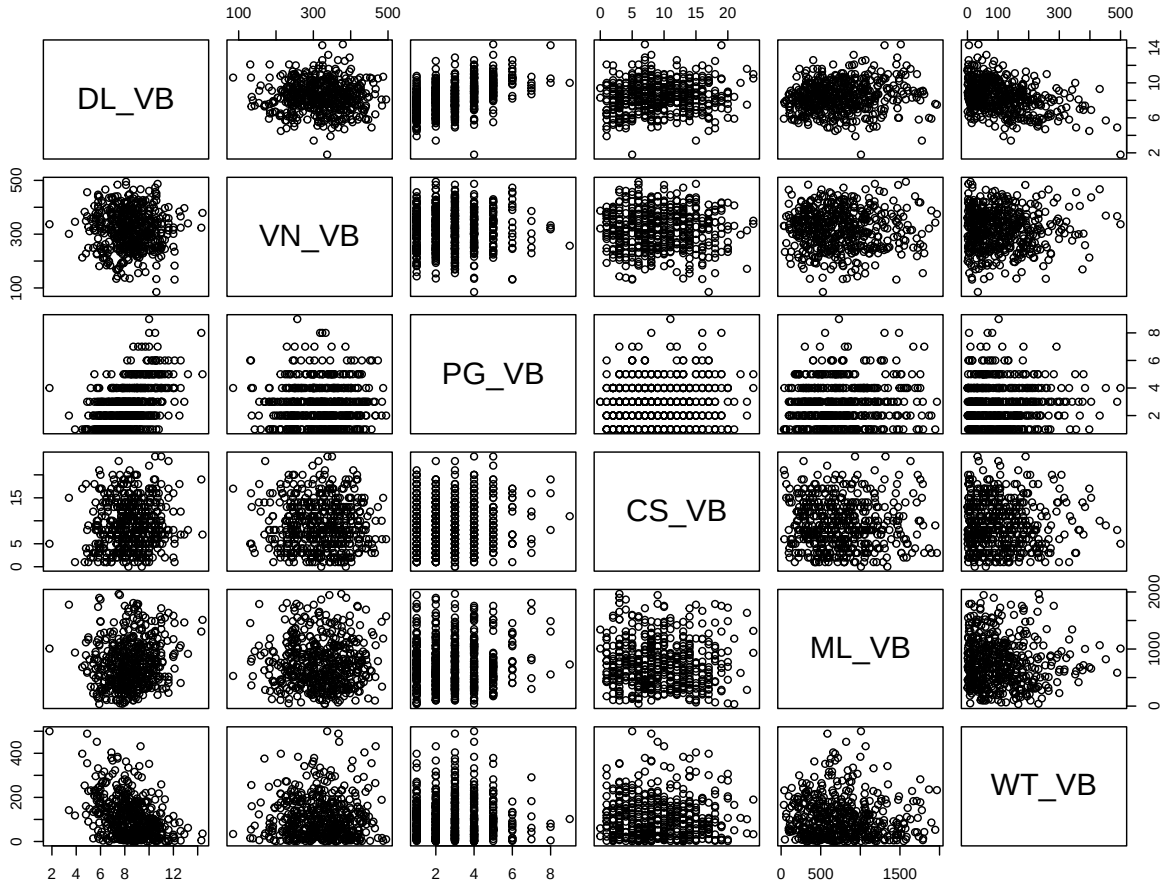
```r
# Choosing the sampling rate
sr <- 0.8

# Find the number of rows of data
n.row <- nrow(df_VB)

#Choose the rows for the traning sample

set.seed(5021)
training.rows <- sample(1:n.row, sr*n.row, replace=FALSE)

#Assign to the training sample

train_VB <- subset(df_VB[training.rows,])

# Assign the balance to the Test Sample

test_VB <- subset(df_VB[-c(training.rows),])
```

Simple Linear Regression

Correlations

Creating the Scatterplot matrix to see relationship between variables.

```
# Scatterplot matrix
pairs(df_VB[,-c(6,7,8)])
```



From the scatter plots, there looks to be a relationship between 'Time for delivery' and 'Packages of product have been ordered'. Also there is a weak relationship between 'Time for delivery' and 'Weight of the shipment'.

Checking the correlation between variables using cor() function.

```
correlations <- cor(df_VB[,-c(6,7,8)])
correlations
```

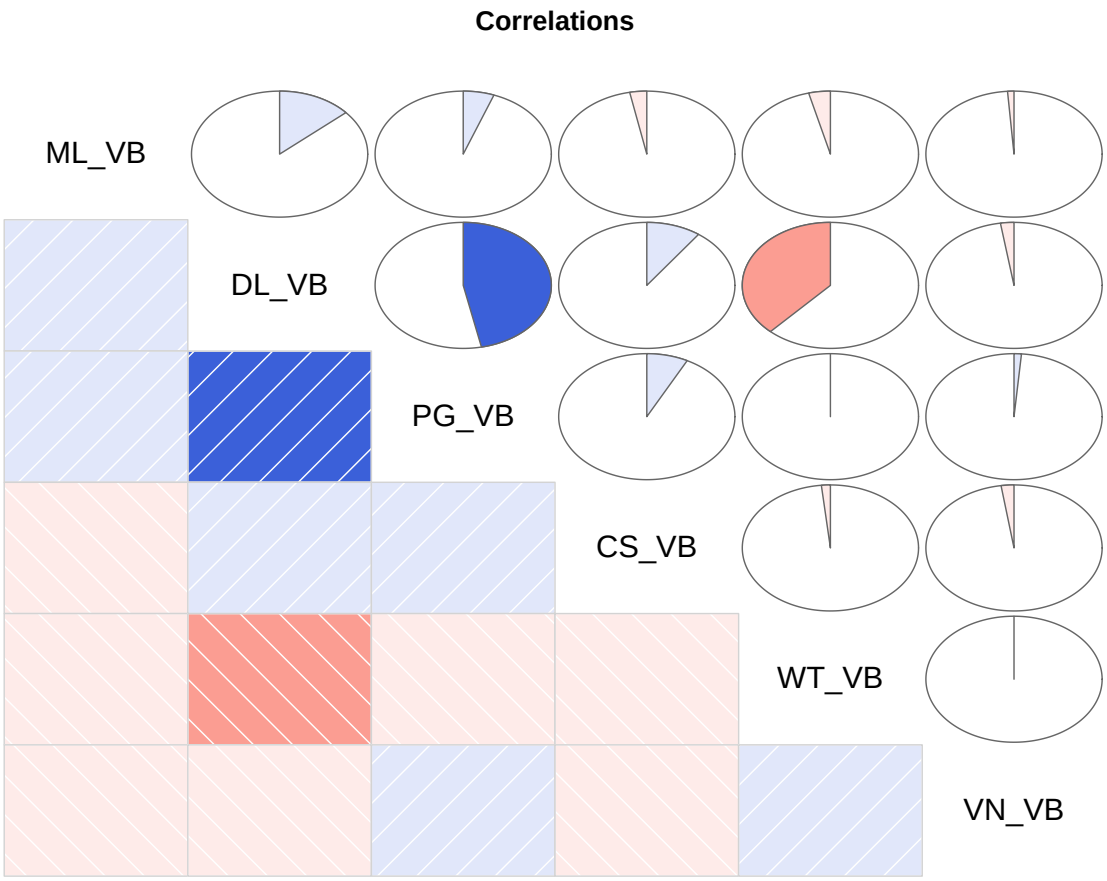
```
##             DL_VB        VN_VB        PG_VB       CS_VB       ML_VB
## DL_VB  1.00000000 -0.024683349  0.466856441  0.09987708  0.13553123
## VN_VB -0.02468335  1.000000000  0.013551011 -0.02352559 -0.01176178
## PG_VB  0.46685644  0.013551011  1.000000000  0.07570014  0.05667762
## CS_VB  0.09987708 -0.023525595  0.075700139  1.00000000 -0.03116592
## ML_VB  0.13553123 -0.011761783  0.056677623 -0.03116592  1.00000000
## WT_VB -0.38095580  0.004080291 -0.006430403 -0.01640212 -0.03973898
##              WT_VB
## DL_VB -0.380955799
## VN_VB  0.004080291
## PG_VB -0.006430403
```

```
## CS_VB -0.016402122
## ML_VB -0.039738979
## WT_VB  1.000000000
```

The correlation function confirms that there is a relationship between 'Time for delivery' and 'Packages of product have been ordered'. Also there is a weak relationship between 'Time for delivery' and 'Weight of the shipment'.

Creating visual representation of Correlation between varibales.

```
corrgram(df_VB, order=TRUE, lower.panel=panel.shade,
         upper.panel=panel.pie, text.panel=panel.txt,
         main="Correlations")
```



Creating Simple Linear Regression model using time for delivery as the dependent variable and weight of the shipment as the independent.

Creating simple linear regression model using time for delivery as the dependent variable and weight of the shipment as the independent.

```
SLR_DL_WT <- lm(DL_VB ~ WT_VB, data=train_VB)
SLR_DL_WT
```

```
##
## Call:
## lm(formula = DL_VB ~ WT_VB, data = train_VB)
##
## Coefficients:
## (Intercept)        WT_VB
##    9.272892    -0.007913
```

Creating Scatter plot

```
plot(DL_VB ~ WT_VB, data=train_VB,
     main="Time for delivery by weight of the shipment(with Regression Line)")
abline(SLR_DL_WT)
```

Creating Simple Linear Regression model using time for delivery as the dependent variable and distance the shipment needs to travel as the independent

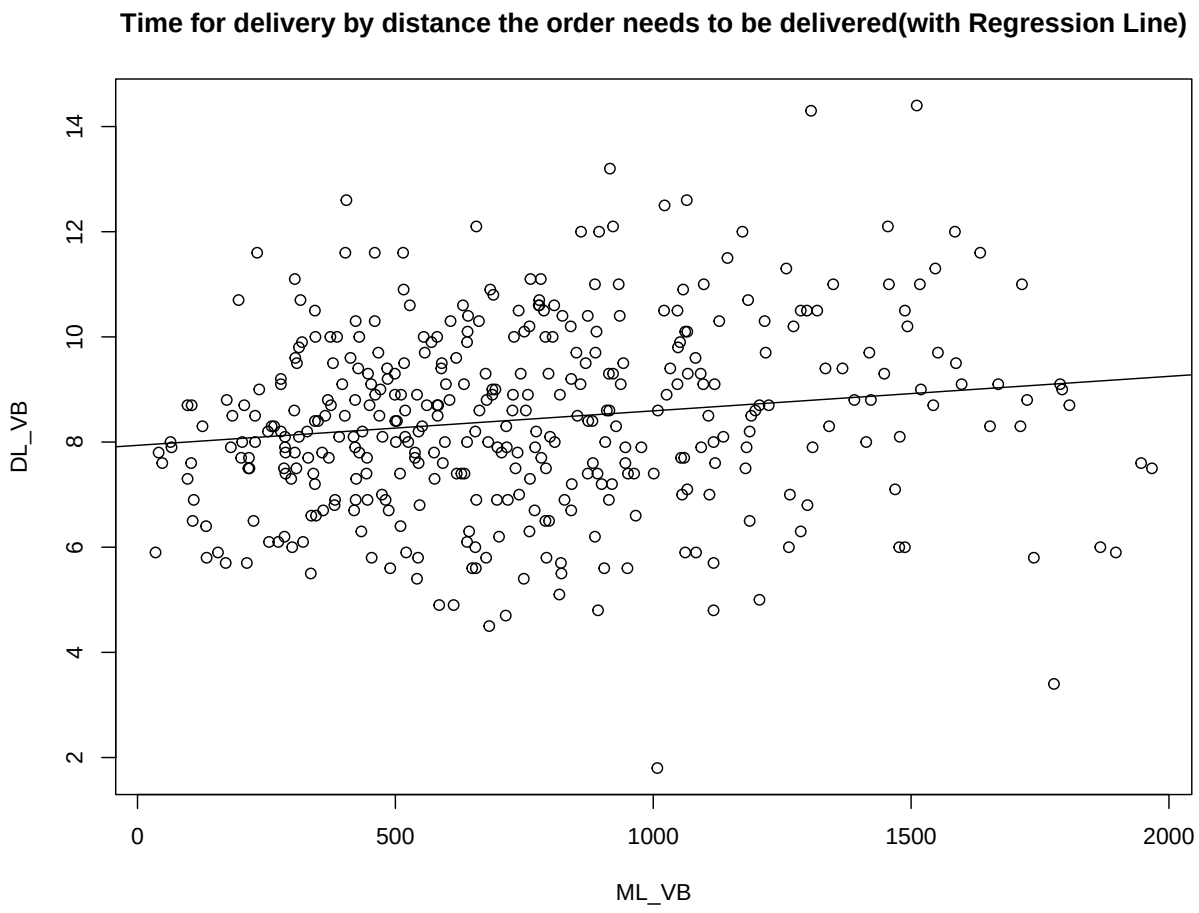
Creating simple linear regression model using time for delivery as the dependent variable and distance the shipment needs to travel as the independent.

```
SLR_DL_ML <- lm(DL_VB ~ ML_VB, data=train_VB)
SLR_DL_ML
```

```
##
## Call:
## lm(formula = DL_VB ~ ML_VB, data = train_VB)
##
## Coefficients:
## (Intercept)        ML_VB
##    7.937281     0.000656
```

Creating Scatter Plot

```
plot(DL_VB ~ ML_VB, data=train_VB,
     main="Time for delivery by distance the order needs to be delivered(with Regression Line)")
abline(SLR_DL_ML)
```



Comparing the models using F-Stat, 2, RMSE for train and test.

RMSE For Model 1

```
pred <- predict(SLR_DL_WT, newdata=train_VB)

RMSE_trn <- sqrt(mean((train_VB$DL_VB - pred)^2))
print(paste("RSME for SLR_DL_WT training dataset: ", round(RMSE_trn,3)))
```

```
## [1] "RSME for SLR_DL_WT training dataset:  1.628"
```

```
RMSE_test <- sqrt(mean((test_VB$DL_VB - pred)^2))
print(paste("RSME for SLR_DL_WT test dataset: ", round(RMSE_test,3)))
```

```
## [1] "RSME for SLR_DL_WT test dataset:  1.701"
```

RMSE For Model 2

```
pred <- predict(SLR_DL_ML, newdata=train_VB)

RMSE_trn <- sqrt(mean((train_VB$DL_VB - pred)^2))
print(paste("RSME for SLR_DL_ML train dataset: ", round(RMSE_trn,3)))
```

```
## [1] "RSME for SLR_DL_ML train dataset:  1.762"
```

```
RMSE_test <- sqrt(mean((test_VB$DL_VB - pred)^2))
print(paste("RSME for SLR_DL_ML test dataset: ", round(RMSE_test,3)))
```

```
## [1] "RSME for SLR_DL_ML test dataset:  1.541"
```

Summary of Model 1

```
summary(SLR_DL_WT)
```

```
##
## Call:
## lm(formula = DL_VB ~ WT_VB, data = train_VB)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7492 -1.2222  0.0303  1.1487  5.4120
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.2728918  0.1265965  73.248   <2e-16 ***
## WT_VB       -0.0079135  0.0009012  -8.781   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.632 on 386 degrees of freedom
## Multiple R-squared:  0.1665, Adjusted R-squared:  0.1643
## F-statistic:  77.1 on 1 and 386 DF,  p-value: < 2.2e-16
```

Summary of Model 2

```
summary(SLR_DL_ML)
```

```
##
## Call:
## lm(formula = DL_VB ~ ML_VB, data = train_VB)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7985 -1.1131 -0.0343  1.1768  5.5060
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.9372810  0.1862226  42.623  < 2e-16 ***
## ML_VB       0.0006560  0.0002162   3.034  0.00257 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.767 on 386 degrees of freedom
## Multiple R-squared:  0.0233, Adjusted R-squared:  0.02077
## F-statistic: 9.207 on 1 and 386 DF,  p-value: 0.002574
```

Comparing the models

Model 1:

Multiple R-squared: 0.1665
Adjusted R-squared: 0.1643
F-statistic: 77.1 on 1 and 386 DF
p-value: $<$ 2.2e-16

Model 2:

Multiple R-squared: 0.0233
Adjusted R-squared: 0.02077
F-statistic: 9.207 on 1 and 386 DF
p-value: 0.002574

Model 1 has an adjusted R-squared of 0.1643 and a residual standard error of 1.632, while Model 2 has an adjusted R-squared of 0.02077 and a residual standard error of 1.767. Model 1 has a higher Multiple R-squared and a lower p-value compared to Model 2. The lower residual standard error of model 1 indicates that the model's predictions are more accurate

Therefore, we can conclude that Model 1 is more superior than Model 2.

Model Development – Multivariate

Multivariate model using all variables

```
MLR_DL_full_VB <- lm(DL_VB ~ . , data=train_VB, na.action=na.omit )
MLR_DL_full_VB
```

```
##
## Call:
## lm(formula = DL_VB ~ ., data = train_VB, na.action = na.omit)
##
## Coefficients:
##  (Intercept)         VN_VB         PG_VB         CS_VB         ML_VB
##    7.1991946    -0.0004398     0.5481047     0.0192520     0.0003549
##       DM_VBI        HZ_VBN  CR_VBSup Del         WT_VB
##    0.5458472    -0.8304367     0.9965803    -0.0068021
```

```
summary(MLR_DL_full_VB)
```

```
##
## Call:
## lm(formula = DL_VB ~ ., data = train_VB, na.action = na.omit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8827 -0.7130 -0.0111  0.7547  4.0899
##
## Coefficients:
##                Estimate Std. Error t value          Pr(>|t|)
## (Intercept)   7.1991946  0.4047926  17.785           < 2e-16 ***
## VN_VB        -0.0004398  0.0008847  -0.497          0.619363
## PG_VB         0.5481047  0.0421164  13.014           < 2e-16 ***
## CS_VB         0.0192520  0.0121953   1.579          0.115252
## ML_VB         0.0003549  0.0001550   2.289          0.022621 *
## DM_VBI        0.5458472  0.1405410   3.884          0.000121 ***
## HZ_VBN       -0.8304367  0.1852021  -4.484 0.000009723855597 ***
## CR_VBSup Del  0.9965803  0.1308633   7.615 0.000000000000212 ***
## WT_VB        -0.0068021  0.0006963  -9.769           < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.251 on 379 degrees of freedom
## Multiple R-squared:  0.5192, Adjusted R-squared:  0.5091
## F-statistic: 51.16 on 8 and 379 DF,  p-value: < 2.2e-16
```

R-squared value is 0.5192, which is moderate. The residual standard error is 1.251. Significance tests of the individual coefficients indicate that only five of the independent variables have statistically significant effects on the dependent variable.

Calculating RSME For Training and Test for Model 1

```
pred <- predict(MLR_DL_full_VB, newdata=train_VB)

RMSE_trn <- sqrt(mean((train_VB$DL_VB - pred)^2))
print(paste("RSME for MLR_DL_full_VB training dataset: ", round(RMSE_trn,3)))
```

```
## [1] "RSME for MLR_DL_full_VB training dataset:  1.236"
```

```
RMSE_test <- sqrt(mean((test_VB$DL_VB - pred)^2))
print(paste("RSME for MLR_DL_full_VB test dataset: ", round(RMSE_test,3)))
```

```
## [1] "RSME for MLR_DL_full_VB test dataset:  2.04"
```

RMSE for the test dataset is higher than that for the training dataset, which suggests that the model might be overfitting slightly.

Multivariate model using backward selection

```
MLR_DL_backward_VB = step(MLR_DL_full_VB, direction="backward", details=TRUE)
```

```
## Start:  AIC=182.69
## DL_VB ~ VN_VB + PG_VB + CS_VB + ML_VB + DM_VB + HZ_VB + CR_VB +
##     WT_VB
##
##         Df Sum of Sq    RSS    AIC
## - VN_VB  1     0.387 593.55 180.95
## <none>               593.16 182.69
## - CS_VB  1     3.900 597.06 183.24
## - ML_VB  1     8.201 601.37 186.02
## - DM_VB  1    23.609 616.77 195.84
## - HZ_VB  1    31.467 624.63 200.75
## - CR_VB  1    90.766 683.93 235.94
## - WT_VB  1   149.375 742.54 267.84
## - PG_VB  1   265.069 858.23 324.02
##
## Step:  AIC=180.95
## DL_VB ~ PG_VB + CS_VB + ML_VB + DM_VB + HZ_VB + CR_VB + WT_VB
##
##         Df Sum of Sq    RSS    AIC
## <none>               593.55 180.95
## - CS_VB  1     4.014 597.57 181.56
## - ML_VB  1     8.536 602.09 184.49
## - DM_VB  1    23.625 617.18 194.09
## - HZ_VB  1    31.778 625.33 199.18
## - CR_VB  1    92.949 686.50 235.39
## - WT_VB  1   149.558 743.11 266.14
## - PG_VB  1   264.683 858.23 322.02
```

```
summary(MLR_DL_backward_VB)
```

```
##
## Call:
## lm(formula = DL_VB ~ PG_VB + CS_VB + ML_VB + DM_VB + HZ_VB +
##     CR_VB + WT_VB, data = train_VB, na.action = na.omit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8853 -0.7051 -0.0153  0.7706  4.0544
##
## Coefficients:
##                Estimate Std. Error t value          Pr(>|t|)
## (Intercept)   7.0536528  0.2792892  25.256           < 2e-16 ***
## PG_VB         0.5473420  0.0420468  13.017           < 2e-16 ***
## CS_VB         0.0195133  0.0121719   1.603          0.109733
## ML_VB         0.0003609  0.0001544   2.338          0.019924 *
## DM_VBI        0.5460376  0.1404012   3.889          0.000119 ***
## HZ_VBN       -0.8339327  0.1848852  -4.511 0.000008629261670 ***
## CR_VBSup Del  1.0032215  0.1300507   7.714 0.000000000000108 ***
## WT_VB        -0.0068059  0.0006955  -9.785           < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.25 on 380 degrees of freedom
## Multiple R-squared:  0.5189, Adjusted R-squared:   0.51
## F-statistic: 58.55 on 7 and 380 DF,  p-value: < 2.2e-16
```

R-squared value is 0.5189, which is moderate. The residual standard error is 1.25. Significance tests of the individual coefficients indicate that only five of the independent variables have statistically significant effects on the dependent variable.

Calculating RSME For Training and Test for Model 2

```
pred <- predict(MLR_DL_backward_VB, newdata=train_VB)

RMSE_trn <- sqrt(mean((train_VB$DL_VB - pred)^2))
print(paste("RSME for MLR_DL_backward_VB training dataset: ", round(RMSE_trn,3)))
```

```
## [1] "RSME for MLR_DL_backward_VB training dataset:  1.237"
```

```
RMSE_test <- sqrt(mean((test_VB$DL_VB - pred)^2))
print(paste("RSME for MLR_DL_backward_VB test dataset: ", round(RMSE_test,3)))
```

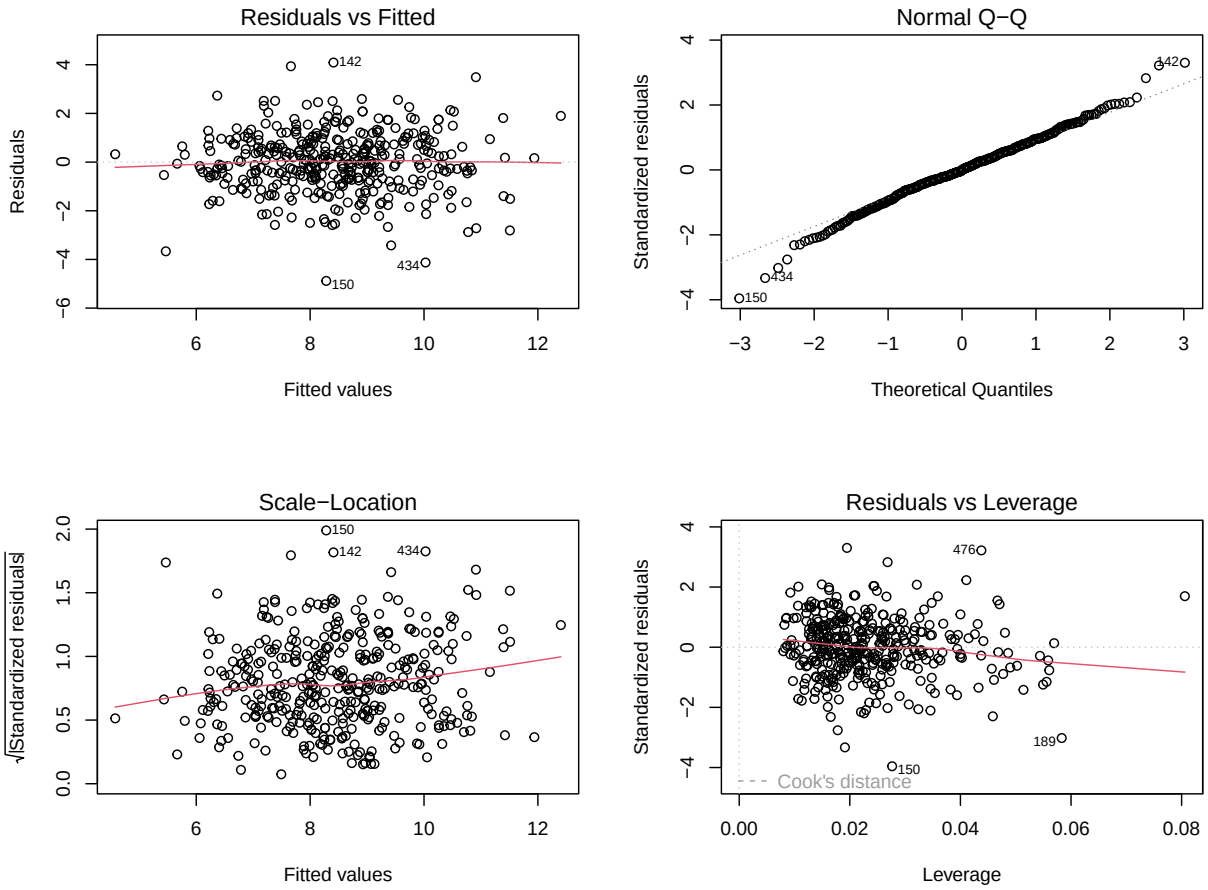
```
## [1] "RSME for MLR_DL_backward_VB test dataset:  2.039"
```

RMSE for the test dataset is higher than that for the training dataset, which suggests that the model might be overfitting slightly.

Model Evaluation – Verifying Assumptions - Multivariate

Model Evaluation - Multivariate Model using all the variables.

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




```
par(mfrow = c(1, 1))
```

Observation for Multivariate Model using all the variables.

Residual vs Fitted plot: The residuals are randomly scattered around the zero line, with no distinct pattern or trend suggesting mean of zero

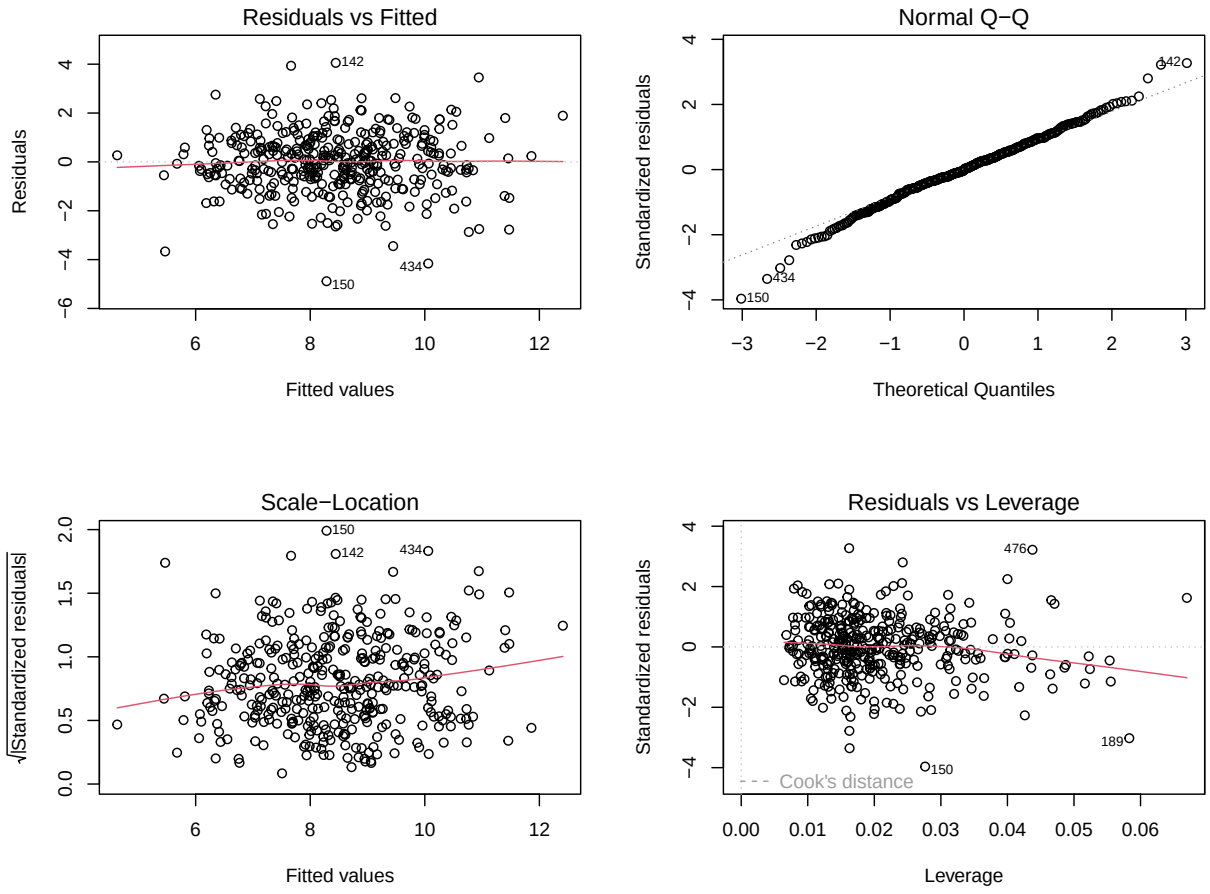
Normal Q-Q plot: Te points fall close to the diagonal line, suggesting the residuals to be normally distributed.

Scale-Location plot: The points are randomly scattered around a horizontal line, suggesting the variance of the residuals is constant, which is good for a linear model.

Residual vs Leverage plot: Observations that fall outside the dashed lines (Cook's distance) are considered influential and should be examined closely, but here we can barely see Cook's distance lines because all cases are well inside of the Cook's distance line.

Model Evaluation - Multivariate Model using backward selection.

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




```
par(mfrow = c(1, 1))
```

Observation for Multivariate Model using all the variables.

Residual vs Fitted plot: The residuals are randomly scattered around the zero line, with no distinct pattern or trend suggesting mean of zero

Normal Q-Q plot: Te points fall close to the diagonal line, suggesting the residuals to be normally distributed.

Scale-Location plot: The points are randomly scattered around a horizontal line, suggesting the variance of the residuals is constant, which is good for a linear model.

Residual vs Leverage plot: Observations that fall outside the dashed lines (Cook's distance) are considered influential and should be examined closely, but here we can barely see Cook's distance lines because all cases are well inside of the Cook's distance line.

Final Recommendation - Multivariate

Both models meet the regression assumptions and perform similarly in terms of RMSE on the training and test datasets. The corrected R-squared value of Model 2 is marginally higher, indicating that it explains more variance in the response variable.

Also model 2 gets rid of 1 variable i.e. VN(Vintage of product), which was not significantly related to the outcome variable.

As a result, I advise using Model 2 to make predictions and draw conclusions.