**Part 1**:

**Typical Linear Regression Equation** -> 

The above equation generally conforms to the form of a straight-line eqn. -> y = MX + c. In the above equation:

β0 = Y-intercept of the line.

β1 = Coefficient of the independent variable, x. This can also be thought of as the slope of the best-fit regression line.

Y – Dependent variable which needs to be predicted. Denotation E(Y|x) translates to the expected value of Y given x.

1. **Loss Function** -> This function measures the discrepancy between the observed value and the expected value of Y.

Mathematically -> 

Where, y (hat) = observed value

y = Expected value

The metric is a scalar quantity and hence is squared to measure the magnitude only.

1. **Ordinary Least Squares(OLS) Regression** -> This is used to determine the predicted values which will stay closest to the original best fit line of the linear regression. It is done in three steps ->
2. Compute the residual i.e. the loss between the expected and measured values of the dependent variable.
3. Compute the residual sum of squared loss (RSS) to get an overall picture of the residual spread.
4. Find the values of the predicted coefficients and dependent variable which give the minimum RSS.

Chart, scatter chart

Description automatically generated

In the above image, the red line depicts the line of best fit with the minimum RSS.

1. The numerical metrics/measures which depict the accuracy and proper fit of linear regression are:
2. **Residual Standard Error (RSE)**: This is an estimate of the standard deviation of the error term. This simply indicates how much the observations will deviate from the true regression line. Mathematically,

A picture containing text, clock, watch

Description automatically generated

For Simple, linear regression, p = 1. Here,

n -> number of observations

p -> number of predictors

Smaller the RSE, the better the model.

1. **R-squared** -> This measure indicates how much of the underlying variance of the data is captured by the independent variables considered in the model.

Another derived metric, Adjusted R2 also considers the number of predictors considered in the model. High deviated between the Adjusted R2 and R2 can indicate a poor-fitting model. This can also be cross-checked by measures like AIC and BIC to attain a more parsimonious fit.

Larger values of both the Adjusted R2 and R2 are better and indicate a better fitting model.

Mathematically,

Diagram

Description automatically generated with low confidence

iii.) F – Stat -> Assesses multiple coefficients simultaneously, and compares the fitted model with the original model i.e. the intercept model.

The larger the F-stat, the better the model.

Mathematically,

Text

Description automatically generated

1. **Training Set** -> The subset of a data or stand-alone data which is used to “train” the model for further pattern recognition.

Training error is measured by the Mean Squared Error( MSE) and Root Mean Squared Error (RMSE).

**Testing Set** -> The subset of a data or stand-alone data which is used to test the effectiveness of the created model to judge its out-of-sample performance.

Testing error is also measured by the Mean Squared Error( MSE) and Root Mean Squared Error (RMSE).

**Part 2**:

# Import the data

cars <- read.csv(file.choose())

View(cars)

# Set the required Seed and subset the data

set.seed(490)

random\_seed <- c("Mersenne-Twister", 490)

index <- sample(1:nrow(cars),0.5\*nrow(cars), replace = FALSE)

cars\_train <- cars[index,]

cars\_test <- cars[-index,]

############################ Simple Linear Regression #######################################

## Training data modeling

lm1 <- lm(mpg~weight, data = cars\_train)

summary(lm1)

mse1 <- mean(residuals(lm1)^2)

mse1

rmse1 <- sqrt(mse1)

rmse1

rss <- sum(residuals(lm1)^2)

rss

rse <- sqrt(rss/197)

rse

lm1\_predict <- predict(lm1, cars\_test)

summary(lm1\_predict)

lm1\_test\_ssl <- sum((cars\_test$mpg - lm1\_predict)^2)

sprintf("SSL/SSR/SSE: %f", lm1\_test\_ssl)

lm1\_test\_mse <- lm1\_test\_ssl/nrow(cars\_test)

sprintf("MSE: %f", lm1\_test\_mse)

lm1\_test\_rmse <- sqrt(lm1\_test\_mse)

sprintf("RMSE: %f", lm1\_test\_rmse)

plot(mpg~weight, cars\_test)

abline(lm1)

## Full data modeling

lm1\_fd <- lm(mpg~weight, data = cars)

summary(lm1\_fd)

mse1\_fd <- mean(residuals(lm1\_fd)^2)

mse1\_fd

rmse1\_fd <- sqrt(mse1\_fd)

rmse1\_fd

rss\_fd <- sum(residuals(lm1\_fd)^2)

rss\_fd

rse\_fd <- sqrt(rss\_fd/197)

rse\_fd

############################### Multiple Linear Regression ##########################################

## Training data modeling

lm2 <- lm(mpg~cylinders + displacement + weight + acceleration + model.year, data = cars\_train)

summary(lm2)

lm2\_mse <- mean(residuals(lm2)^2)

lm2\_mse

lm2\_rmse <- sqrt(lm2\_mse)

lm2\_rss <- sum(residuals(lm2)^2)

lm2\_rss

lm2\_rse <- sqrt(lm2\_rss/193)

lm2\_rse

lm2\_predict <- predict(lm2,cars\_test)

summary(lm2\_predict)

lm2\_test\_ssl <- sum((cars\_test$mpg - lm2\_predict)^2)

sprintf("SSL/SSR/SSE: %f", lm2\_test\_ssl)

lm2\_test\_mse <- lm2\_test\_ssl/nrow(cars\_test)

sprintf("MSE: %f", lm2\_test\_mse)

lm2\_test\_rmse <- sqrt(lm2\_test\_mse)

sprintf("RMSE: %f", lm2\_test\_rmse)

scatter.smooth(cars\_test$weight,cars\_test$mpg,main = "weight ~ mpg")

## Full data modeling

lm2\_fd <- lm(mpg~cylinders + displacement + weight + acceleration + model.year, data = cars)

summary(lm2\_fd)

lm2\_fd\_mse <- mean(residuals(lm2\_fd)^2)

lm2\_fd\_mse

lm2\_fd\_rmse <- sqrt(lm2\_fd\_mse)

lm2\_fd\_rss <- sum(residuals(lm2\_fd)^2)

lm2\_fd\_rss

lm2\_fd\_rse <- sqrt(lm2\_fd\_rss/392)

lm2\_fd\_rse

###################################### Parsimonious Model #######################################

## Training data modeling

lm3 <- lm(mpg~cylinders + model.year, data = cars\_train)

summary(lm3)

lm3\_mse <- mean(residuals(lm3)^2)

lm3\_mse

lm3\_rmse <- sqrt(lm3\_mse)

lm3\_rmse

lm3\_rss <- sum(residuals(lm3)^2)

lm3\_rss

lm3\_rse <- sqrt(lm3\_rss/196)

lm3\_rse

lm3\_predict <- predict(lm3,cars\_test)

summary(lm3\_predict)

lm3\_test\_ssl <- sum((cars\_test$mpg - lm3\_predict)^2)

sprintf("SSL/SSR/SSE: %f", lm3\_test\_ssl)

lm3\_test\_mse <- lm3\_test\_ssl/nrow(cars\_test)

sprintf("MSE: %f", lm3\_test\_mse)

lm3\_test\_rmse <- sqrt(lm3\_test\_mse)

sprintf("RMSE: %f", lm3\_test\_rmse)

## Full data modeling

lm3\_fd <- lm(mpg~cylinders + model.year, data = cars)

summary(lm3\_fd)

lm3\_fd\_mse <- mean(residuals(lm3\_fd)^2)

lm3\_fd\_mse

lm3\_fd\_rmse <- sqrt(lm3\_fd\_mse)

lm3\_fd\_rmse

lm3\_fd\_rss <- sum(residuals(lm3\_fd)^2)

lm3\_fd\_rss

lm3\_fd\_rse <- sqrt(lm3\_fd\_rss/395)

lm3\_fd\_rse

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Statistic | Simple Linear Regression | | | Multiple Regression | | | Multiple Regression  (only significant variables) | | |
| Train | Test | All | Train | Test | All | Train | Test | All |
| MSE | 18.63 | 18.97 | 18.78 | 10.47 | 13.07 | 11.65 | 16.10 | 19.28 | 17.68 |
| RMSE | 4.31 | 4.355 | 4.33 | 3.23 | 3.61 | 3.41 | 4.01 | 4.39 | 4.20 |
| RSS | 3707.65 | - | 7474.81 | 2084.48 | - | 4639.8 | 3205.25 | - | 7037.21 |
| RSE | 4.33 | - | 6.15 | 3.28 | - | 3.44 | 4.04 | - | 4.22 |
| R^2 | 0.6842 | - | 0.691 | 0.822 | - | 0.806 | 0.724 | - | 0.708 |
| Adj. R^2 | 0.6826 | - | 0.691 | 0.817 | - | 0.808 | 0.727 | - | 0.709 |
| F-Statistic | 426.8 | - | 888.9 | 178.8 | - | 331.4 | 261 | - | 483.2 |

**Parsimonious Model -> -16.61 + (-2.99)\*cylinders + 0.74 \* Model.year [Basis the model based on full dataset]**