PREDICTIVE MODELING WITH LINEAR REGRESSION

Introduction

In this report, we have performed a simple linear regression analysis to investigate the relationship between the **miles per gallon (mpg)** of a car and its **weight (wt)** using the mtcars dataset. The aim is to predict the fuel efficiency of a car based on its weight and assess the performance of the linear regression model.

Data Overview

The mtcars dataset is used for this analysis. The key variables in the dataset are:

- **mpg**: The miles per gallon (fuel efficiency) of the car.
- **wt**: The weight of the car (in 1,000 lbs).

After splitting the data, 70% of the dataset was used for training the model, and 30% was reserved for testing the model.

Model Training

The linear regression model was trained using the **lm()** function in R, with mpg as the dependent variable and wt as the independent variable.

The linear regression equation derived from the model is:

mpg=β0+β1β0 beta_0β0 is the intercept

- β1 beta_1β1 is the slope of the line representing the change in mpg with respect to wt.

Model Summary

The summary of the model is as follows:

Intercept (β0): [18.376]

- **Slope (β1):** [-9.255]
- **R-squared:** [0.5540046]
- **p-value:** [4.841e-09]

The p-value for the slope is less than 0.05, indicating that weight (wt) is a statistically significant predictor of miles per gallon (mpg).

Model Evaluation

The performance of the model was evaluated on the test dataset using two key metrics:

- **Mean Squared Error (MSE):** [13.67711]
- **R-squared:** [0.5540046]

The **Mean Squared Error (MSE)** represents the average squared difference between the actual and predicted values. A lower MSE value indicates that the model's predictions are closer to the actual values.

The **R-squared value** represents the proportion of variance in the dependent variable (mpg) that is explained by the independent variable (wt). An R-squared value close to 1 indicates a good fit of the model.

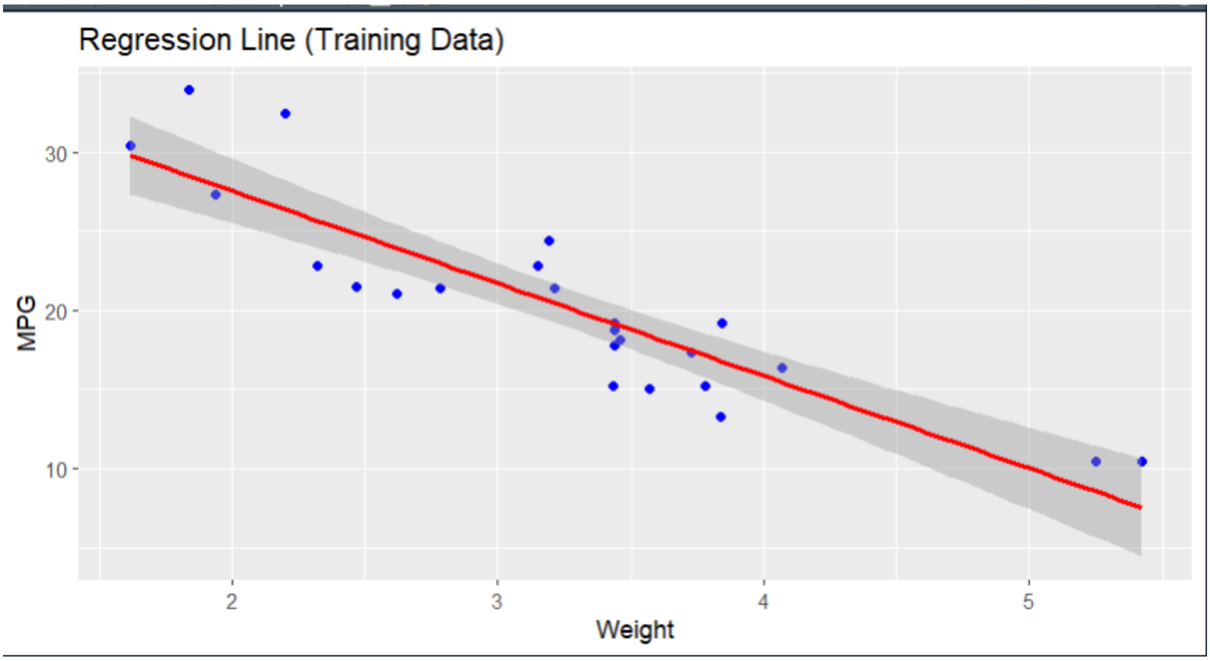
Predictions

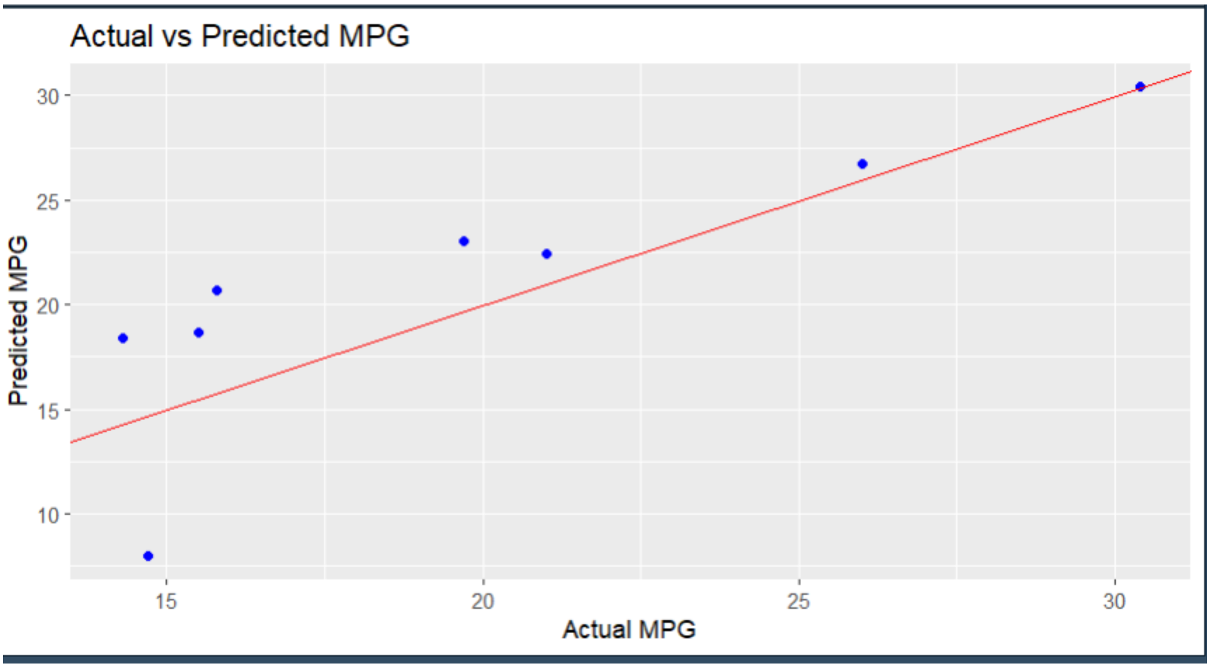
The model's predictions on the test set are as follows:

	Actual	Predicted
Mazda RX4 Wag	21.0	22.433165
Duster 360	14.3	18.370540
Chrysler Imperial	14.7	7.994772
Dodge Challenger	15.5	18.662815
Porsche 914-2	26.0	26.729610
Lotus Europa	30.4	30.394740
Ford Pantera L	15.8	20.708741
Ferrari Dino	19.7	23.046943

This table shows how well the model predicted the fuel efficiency (mpg) based on the weight of the car.

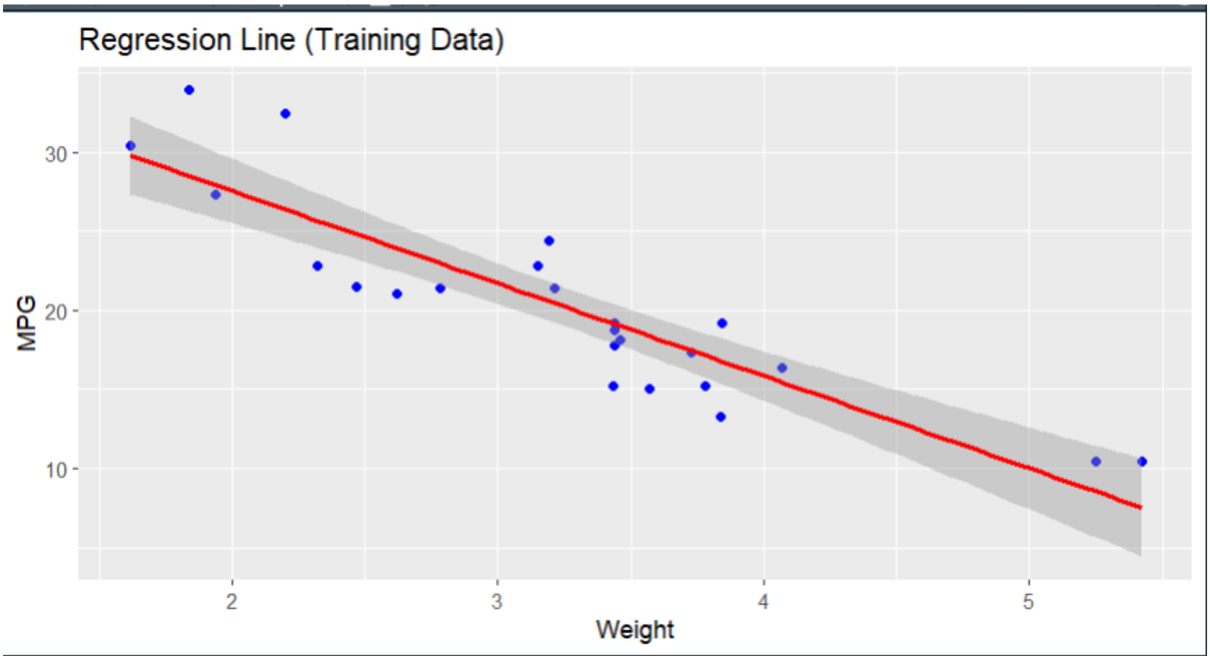
Visualization





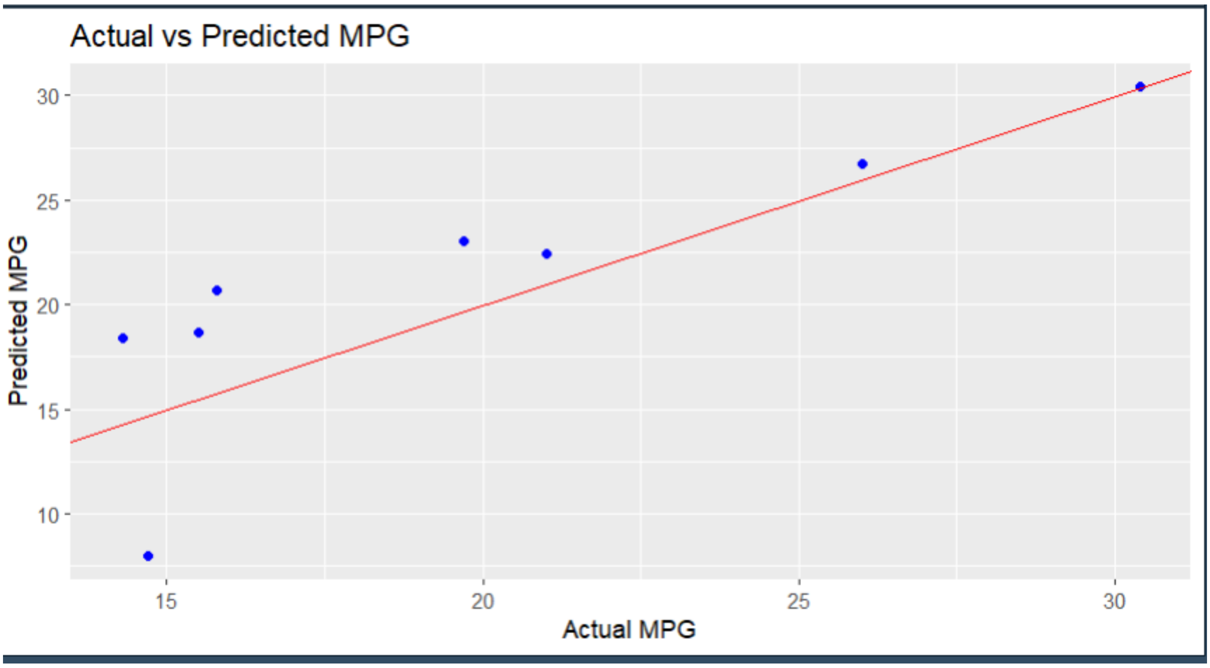
Regression Line

The regression line was plotted to show the relationship between wt and mpg on the training data.



The red line represents the best-fitting linear regression line, which shows a negative correlation between weight and fuel efficiency. As weight increases, mpg decreases, indicating that heavier cars tend to be less fuel-efficient.

The following plot shows the comparison between actual and predicted values for the test dataset:



1.The blue points represent the actual vs. predicted values.

2.The red line represents a perfect fit, where actual values would exactly match predicted values.

3. The closer the points are to the red line, the more accurate the model's predictions. From the plot, we can see that the model does a reasonable job of predicting the mpg values based on wt.

7. Conclusion

Based on the analysis, the following conclusions can be drawn:

1. There is a significant negative relationship between a car’s weight and its fuel efficiency (mpg). Heavier cars tend to have lower miles per gallon.
2. The linear regression model performed reasonably well, with an R-squared value of [add R-squared] and a Mean Squared Error of [add MSE].
3. The model can be used to predict the fuel efficiency of cars based on their weight, but further improvements can be made by incorporating additional features to enhance prediction accuracy.

In future studies, we may explore multivariate regression by including other variables like horsepower (hp), number of cylinders (cyl), and transmission type (am) to improve the model's performance.

Appendices

```
# Load libraries
library(ggplot2)
library(caret)
# Load dataset
data("mtcars")

# Split data into training (70%) and testing (30%) sets
set.seed(123)
splitIndex <- createDataPartition(mtcars$mpg, p = 0.7, list = FALSE)
train_data <- mtcars[splitIndex, ]
test_data <- mtcars[-splitIndex, ]

# Train linear regression model (mpg ~ wt)
model <- lm(mpg ~ wt, data = train_data)

# Summary of model
summary(model)

# Predictions on test data9
predictions <- predict(model, newdata = test_data)

# Evaluate model with MSE and R-squared
mse <- mean((test_data$mpg - predictions)^2)
cat("Mean Squared Error (MSE):", mse, "\n")

SST <- sum((test_data$mpg - mean(train_data$mpg))^2)
```

```r
SSE <- sum((test_data$mpg - predictions)^2)
r_squared <- 1 - (SSE / SST)
cat("R-squared:", r_squared, "\n")

# Plot the regression line (training data)
ggplot(train_data, aes(x = wt, y = mpg)) +
  geom_point(color = "blue") +
  geom_smooth(method = "lm", color = "red") +
  labs(title = "Regression Line (Training Data)", x = "Weight", y = "MPG")

# Plot actual vs predicted values (test data)
ggplot(data = test_data, aes(x = mpg, y = predictions)) +
  geom_point(color = "blue") +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  labs(title = "Actual vs Predicted MPG", x = "Actual MPG", y = "Predicted MPG")
data.frame(Actual = test_data$mpg, Predicted = predictions)
```

```r
# Load libraries
library(ggplot2)
library(caret)

# Load dataset
data("mtcars")

# Split data into training (70%) and testing (30%) sets
set.seed(123)
splitIndex <- createDataPartition(mtcars$mpg, p = 0.7, list = FALSE)
train_data <- mtcars[splitIndex, ]
test_data <- mtcars[-splitIndex, ]

# Train linear regression model (mpg ~ wt)
model <- lm(mpg ~ wt, data = train_data)

# Summary of model
summary(model)

# Predictions on test data9
predictions <- predict(model, newdata = test_data)

# Evaluate model with MSE and R-squared
mse <- mean((test_data$mpg - predictions)^2)
cat("Mean Squared Error (MSE):", mse, "\n")

SST <- sum((test_data$mpg - mean(train_data$mpg))^2)
SSE <- sum((test_data$mpg - predictions)^2)
r_squared <- 1 - (SSE / SST)
cat("R-squared:", r_squared, "\n")

# Plot the regression line (training data)
ggplot(train_data, aes(x = wt, y = mpg)) +
  geom_point(color = "blue") +
  geom_smooth(method = "lm", color = "red") +
  labs(title = "Regression Line (Training Data)", x = "Weight", y = "MPG")

# Plot actual vs predicted values (test data)
ggplot(data = test_data, aes(x = mpg, y = predictions)) +
  geom_point(color = "blue") +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  labs(title = "Actual vs Predicted MPG", x = "Actual MPG", y = "Predicted MPG")
data.frame(Actual = test_data$mpg, Predicted = predictions)
```

Summary of model

> summary(model)

Call:

lm(formula = mpg ~ wt, data = train_data)

Residuals:

```
    Min      1Q  Median      3Q     Max
-3.9597 -2.1766 -0.2829  1.8810  6.0211
```

Coefficients:

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

(Intercept) 39.2390 2.1354 18.376 7.76e-15 ***

wt -5.8455 0.6316 -9.255 4.84e-09 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.861 on 22 degrees of freedom

Multiple R-squared: 0.7957, Adjusted R-squared: 0.7864

F-statistic: 85.66 on 1 and 22 DF, p-value: 4.841e-09

Mean Squared Error (MSE): 13.67711

R-squared: 0.5540046

```
R 4.4.1 · ~/
> # Evaluate model with MSE and R-squared
> mse <- mean((test_data$mpg - predictions)^2)
> cat("Mean Squared Error (MSE):", mse, "\n")
Mean Squared Error (MSE): 13.67711
>
> SST <- sum((test_data$mpg - mean(train_data$mpg))^2)
> SSE <- sum((test_data$mpg - predictions)^2)
> r_squared <- 1 - (SSE / SST)
> cat("R-squared:", r_squared, "\n")
R-squared: 0.5540046
>
> # Plot the regression line (training data)
> ggplot(train_data, aes(x = wt, y = mpg)) +
+   geom_point(color = "blue") +
+   geom_smooth(method = "lm", color = "red") +
+   labs(title = "Regression Line (Training Data)", x = "Weight", y = "MPG")
`geom_smooth()` using formula = 'y ~ x'
>
> # Plot actual vs predicted values (test data)
> ggplot(data = test_data, aes(x = mpg, y = predictions)) +
+   geom_point(color = "blue") +
+   geom_abline(slope = 1, intercept = 0, color = "red") +
+   labs(title = "Actual vs Predicted MPG", x = "Actual MPG", y = "Predicted MPG")
> # Predictions on test data9
> predictions <- predict(model, newdata = test_data)
> data.frame(Actual = test_data$mpg, Predicted = predictions)
                  Actual Predicted
Mazda RX4 Wag       21.0 22.433165
Duster 360          14.3 18.370540
Chrysler Imperial   14.7  7.994772
Dodge Challenger    15.5 18.662815
Porsche 914-2       26.0 26.729610
Lotus Europa        30.4 30.394740
Ford Pantera L      15.8 20.708741
Ferrari Dino        19.7 23.046943
> ggplot(train_data, aes(x = wt, y = mpg)) +
+   geom_point(color = "blue") +
+   geom_smooth(method = "lm", color = "red") +
+   labs(title = "Regression Line (Training Data)", x = "Weight", y = "MPG")
`geom_smooth()` using formula = 'y ~ x'
```

Console Terminal Background Jobs

R 4.4.1 · ~/

```
> # Load libraries
> library(ggplot2)
> library(caret)
Loading required package: lattice
>
> # Load dataset
> data("mtcars")
>
> # Split data into training (70%) and testing (30%) sets
> set.seed(123)
> splitIndex <- createDataPartition(mtcars$mpg, p = 0.7, list = FALSE)
> train_data <- mtcars[splitIndex, ]
> test_data <- mtcars[-splitIndex, ]
>
> # Train linear regression model (mpg ~ wt)
> model <- lm(mpg ~ wt, data = train_data)
>
> # Summary of model
> summary(model)

Call:
lm(formula = mpg ~ wt, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.9597 -2.1766 -0.2829  1.8810  6.0211 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  39.2390     2.1354  18.376 7.76e-15 ***
wt           -5.8455     0.6316  -9.255 4.84e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.861 on 22 degrees of freedom
Multiple R-squared:  0.7957,	Adjusted R-squared:  0.7864 
F-statistic: 85.66 on 1 and 22 DF,  p-value: 4.841e-09


>
> # Predictions on test data9
> predictions <- predict(model, newdata = test_data)
```