Ex.No : 7 **Roll No: 210701159**

Implement Linear and Logistic Regression

AIM:

To implement linear and logistic regression techniques in machine learning.

PROCEDURES:

Linear Regression

1. Define vectors for heights and weights.
2. Combine the heights and weights into a data frame.
3. Fit a linear regression model using height to predict weight.
4. Print the summary of the linear regression model to view model statistics.
5. Open a new graphical device for plotting.
6. Create a scatter plot of height vs. weight data points.
7. Label the plot with a title, x-axis label (Height), and y-axis label (Weight).
8. Set plot points with specific color (blue) and style (solid circle).
9. Add the fitted linear regression line to the plot.
10. Customize the regression line with red color and a thicker width.

Logistic Regression

1. Load the `mtcars` dataset.
2. Convert the `am` column from numeric to a factor with labels "Automatic" and "Manual."
3. Fit a logistic regression model to predict `am` (transmission) based on `mpg` (miles per gallon).
4. Print the summary of the logistic regression model.
5. Predict the probabilities of manual transmission using the logistic model.
6. Print the predicted probabilities for manual transmission.
7. Create a scatter plot of `mpg` vs. transmission type (manual/automatic).
8. Label the plot with a title, x-axis label (MPG), and y-axis label (Probability of Manual Transmission).

9. Set plot points with blue color and solid circles.
10. Add the logistic regression curve to the plot, colored red with a thicker line.

CODE:

LinearRegression.py

```
# Sample data
heights <- c(150, 160, 165, 170, 175, 180, 185)
weights <- c(55, 60, 62, 68, 70, 75, 80)

# Create a data frame
data <- data.frame(heights, weights)

# Fit a linear regression model
linear_model <- lm(weights ~ heights, data = data)

# Print the summary of the model
print(summary(linear_model))

# Plotting the data and regression line
dev.new()
plot(data$heights, data$weights,
    main = "Linear Regression: Weight vs. Height",
    xlab = "Height (cm)",
    ylab = "Weight (kg)",
    pch = 19, col = "blue")

# Add regression line
abline(linear_model, col = "red", lwd = 2)
```

LogisticRegression.py

```
# Load the dataset
```

```
data(mtcars)
# Convert 'am' to a factor (categorical variable)
mtcars$am <- factor(mtcars$am, levels = c(0, 1),
                    labels = c("Automatic", "Manual"))
# Fit a logistic regression model
logistic_model <- glm(am ~ mpg, data = mtcars, family = binomial)
# Print the summary of the model
print(summary(logistic_model))
# Predict probabilities for the logistic model
predicted_probs <- predict(logistic_model, type = "response")
# Display the predicted probabilities
print(predicted_probs)
# Plotting the data and logistic regression curve
plot(mtcars$mpg, as.numeric(mtcars$am) - 1,
     main = "Logistic Regression: Transmission vs. MPG",
     xlab = "Miles Per Gallon (mpg)",
     ylab = "Probability of Manual Transmission",
     pch = 19, col = "blue")
# Add the logistic regression curve
curve(predict(logistic_model, data.frame(mpg = x), type = "response"),
      add = TRUE, col = "red", lwd = 2)
```

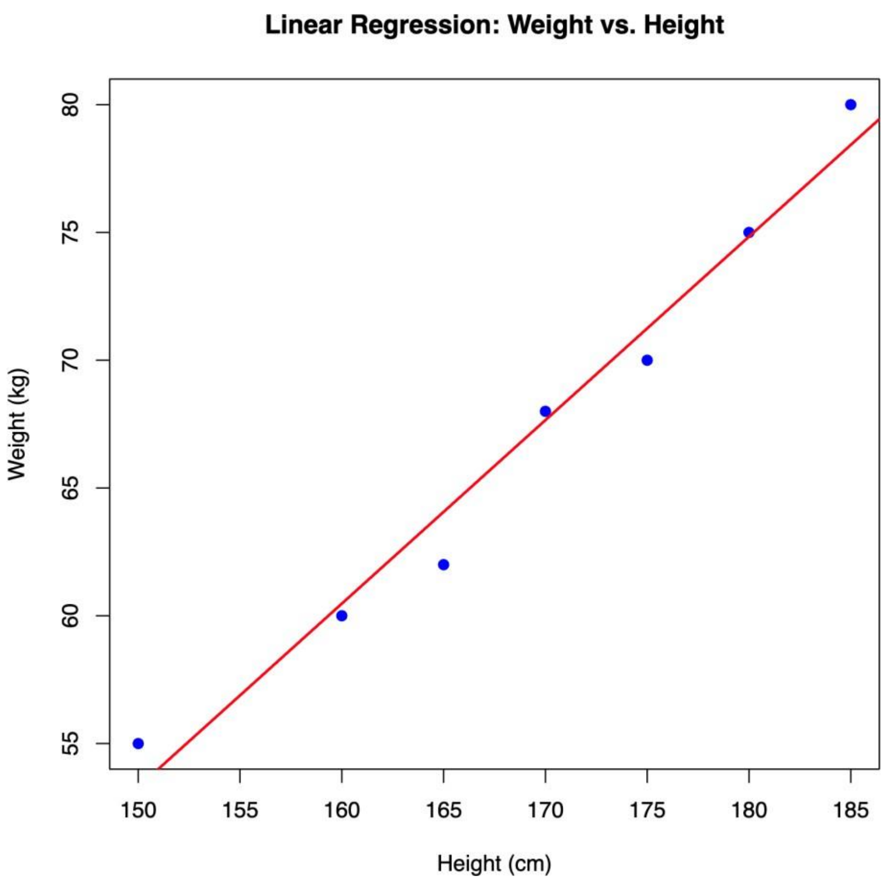
OUTPUT:

```
Call:
lm(formula = weights ~ heights, data = data)

Residuals:
      1       2       3       4       5       6       7 
 1.7049 -0.4754 -2.0656  0.3443 -1.2459  0.1639  1.5738 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -54.40984    8.74376  -6.223  0.00157 ** 
heights       0.71803    0.05154  13.932 3.42e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.521 on 5 degrees of freedom
Multiple R-squared:  0.9749,	Adjusted R-squared:  0.9699 
F-statistic: 194.1 on 1 and 5 DF,  p-value: 3.424e-05
```



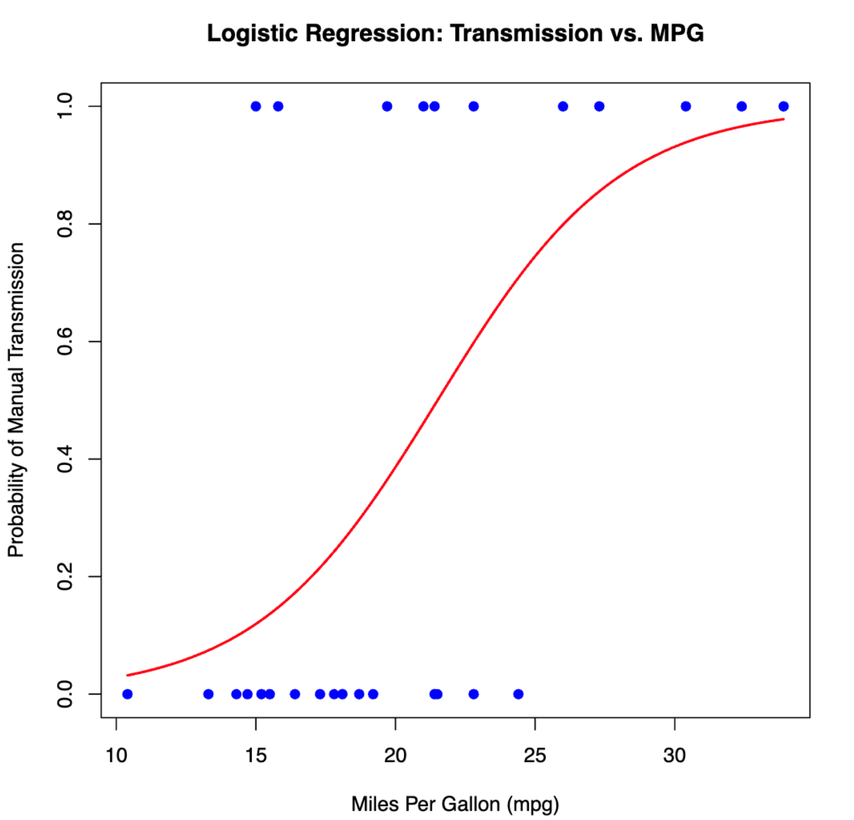
```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -6.6035     2.3514  -2.808  0.00498 **
mpg           0.3070     0.1148   2.673  0.00751 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.230  on 31  degrees of freedom
Residual deviance: 29.675  on 30  degrees of freedom
AIC: 33.675

Number of Fisher Scoring iterations: 5

          Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4 Drive
         0.46109512          0.46109512          0.59789839          0.49171990
  Hornet Sportabout             Valiant          Duster 360           Merc 240D
         0.29690087          0.25993307          0.09858705          0.70846924
           Merc 230            Merc 280           Merc 280C          Merc 450SE
         0.59789839          0.32991148          0.24260966          0.17246396
         Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Continental
         0.21552479          0.12601104          0.03197098          0.03197098
  Chrysler Imperial            Fiat 128         Honda Civic      Toyota Corolla
         0.11005178          0.96591395          0.93878132          0.97821971
      Toyota Corona    Dodge Challenger         AMC Javelin          Camaro Z28
         0.49939484          0.13650937          0.12601104          0.07446438
   Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus Europa
         0.32991148          0.85549212          0.79886349          0.93878132
     Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo 142E
         0.14773451          0.36468861          0.11940215          0.49171990
>
```



RESULT:

Thus, to implement linear and logistic regression using machine learning is completed successfully.