**LINEAR REGRESSION**

**Aim**

The Linear regression model aims at predicting brain weight based on the head size accurately. It will provide a linear equation that represents the regression line through given data points, which allow us to estimate the brain weight for new data points.

**Problem Description**

Understanding the relationship between head size and brain weight is crucial for gaining awareness into the function and development of brain. It provides insights into brain health, aids in diagnostic processes, and contributes to our understanding of neurological development and conditions. This model can have implications in various fields including medical research, neurology, clinical practice, anthropology, forensics and public health.

**Implementation**

Start with choosing a database. Plot the database graphically with feature in the x axis and target value in the y axis. Finding the mean of values in respective axis and mark it as a point in the regression line. Determine the slope of regression line by dividing the summation of the product of difference of x to x mean and y to y mean by summation of the square of difference of x to x mean.

Slope = Σ (x-x̅) (y-y̅) / Σ (y-y̅)²

Determine the equation of regression line using y = mx +c and plot it.

Calculate the Mean Square Error which quantifies the average squared difference between the actual values and predicted values.

MSE = 1/n x Σ (y- y̅)

Then calculate R-square value which provides a relative measure of how well the model’s predictions explain the variability in the data, by calculating the ratio of explained variance to the total variance. It ranges from 0 to 1.

R- Square = Σ(yp- y̅)² / Σ (y- y̅)

When R² value approaches 1, the regression line accuracy is good. As it reduces to 0, there are too many data outlying from the line, which we cannot use for prediction. Still there are exceptions like psychology, which are difficult to predict due to their characteristic of complexity, the R² value comes under 0.5.

**Algorithm**

1. Initialize parameters (the coefficients of linear regression equation.)
2. Define the Linear Regression Equation.
3. Calculate predictions.
4. Calculate the Mean Squared Error.
5. Minimizing the error.
6. Model evaluation by R-squared method.
7. Make predictions.

**Code – Using ScikitLearn**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

#Importing numpy which is a library for numerical operations, pandas for data manipulation and analysis, function for splitting datasets into training and testing sets, class for creating linear regression model, functions for evaluating regression models, and a library for creating visualizations.

df = pd.read\_csv(r"C:\Users\student\Documents\ML\headbrain.csv")

# Load your dataset

data=pd.read\_csv(r"C:\Users\student\Documents\ML\headbrain.csv")

print(data.shape)

data.head()

# Reading the data

X = df[['Head Size(cm^3)']]

Y = df['Brain Weight(grams)']

# Assuming 'X' is the feature and 'Y' is the target variable

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)

# Split the data into training and testing sets

reg = LinearRegression()

reg.fit(X\_train, Y\_train)

Y\_pred = reg.predict(X\_test)

mse = mean\_squared\_error(Y\_test, Y\_pred)

r2 = r2\_score(Y\_test, Y\_pred)

# Create a linear regression model

# Train the model on the training set

# Make predictions on the test set

# Evaluate the model

print("Mean Squared Error:", mse)

print("R-squared:", r2)

plt.scatter(X\_test, Y\_test, color='red', marker='.')

plt.plot(X\_test, Y\_pred, color='green', linewidth=1)

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Linear Regression Example')

plt.show()

# Plotting the regression line

**Code – from scratch**

%matplotlib inline

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

plt.rcParams['figure.figsize']=(20.0, 10.0)

data=pd.read\_csv(r"C:\Users\student\Documents\ML\headbrain.csv")

print(data.shape)

data.head()

#Reading data

X = data['Head Size(cm^3)'].values

Y = data['Brain Weight(grams)'].values

#collecting X and Y

mean\_x = np.mean(X)

mean\_y = np.mean(Y)

# Mean X and Y

m= len(X)

#Total number of values

numer = 0

denom = 0

for i in range(m):

numer += (X[i] - mean\_x) \* (Y[i] - mean\_y)

denom += (X[i] - mean\_x) \*\* 2

b1 = numer / denom

b0 = mean\_y - (b1 \* mean\_x)

print(b1, b0)

# Using the formula to calculate b1 and b2

max\_x = np.max(X) + 100

min\_x = np.min(X)- 100

# plotting values and regression line

x = np.linspace(min\_x, max\_x, 1000)

y = b0 + b1 \* x

# Calculating line values x and y

plt.plot(x,y, color = 'green', label = 'Regreesion line')

plt.scatter(X,Y,c='red' , label = 'Scatter plot')

plt.xlabel('Head size in cms')

plt.ylabel('Brain Weight in grams')

plt.legend()

plt.show()

#Plotting

ss\_t = 0

ss\_r = 0

for i in range(m):

y\_pred = b0 + b1 \* X[i]

ss\_t += (Y[i] - mean\_y)\*\*2

ss\_r += (Y[i] - y\_pred)\*\*2

r2 = 1 - (ss\_r/ss\_t)

print(r2)

#Calculating R-square

**Output**



