Experiment No. 3

Aim: Implementation of Logistic Regression Algorithm.

Objective: Able to perform various feature engineering tasks, apply logistic regression on the given dataset and maximize the accuracy.

Theory:

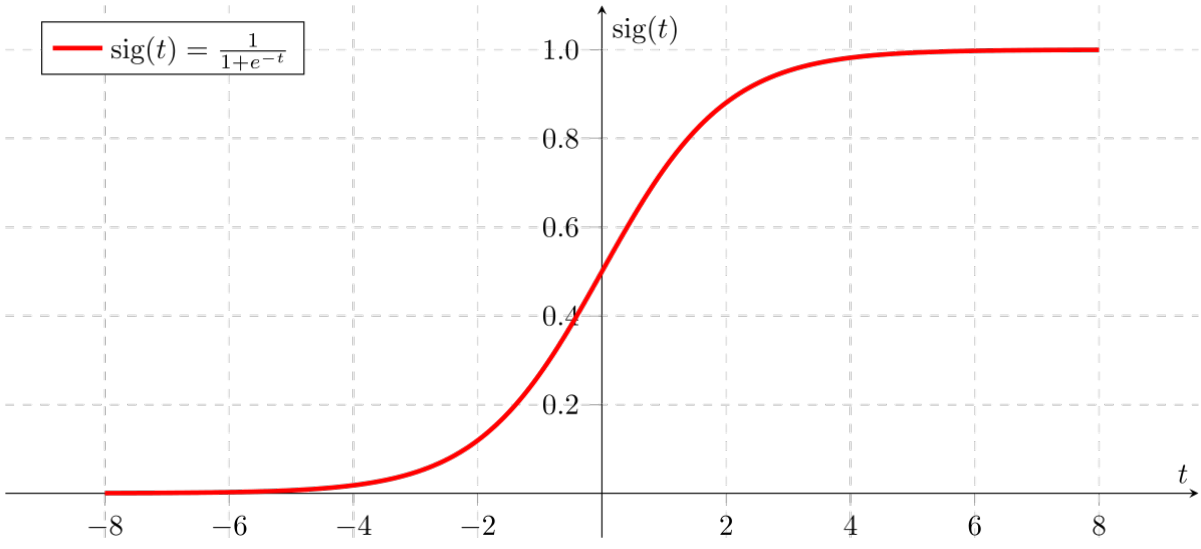
Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical and is binary in nature. In order to perform binary classification, the logistic regression techniques makes use of Sigmoid function.

For example,

To predict whether an email is spam (1) or (0)

Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.



From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

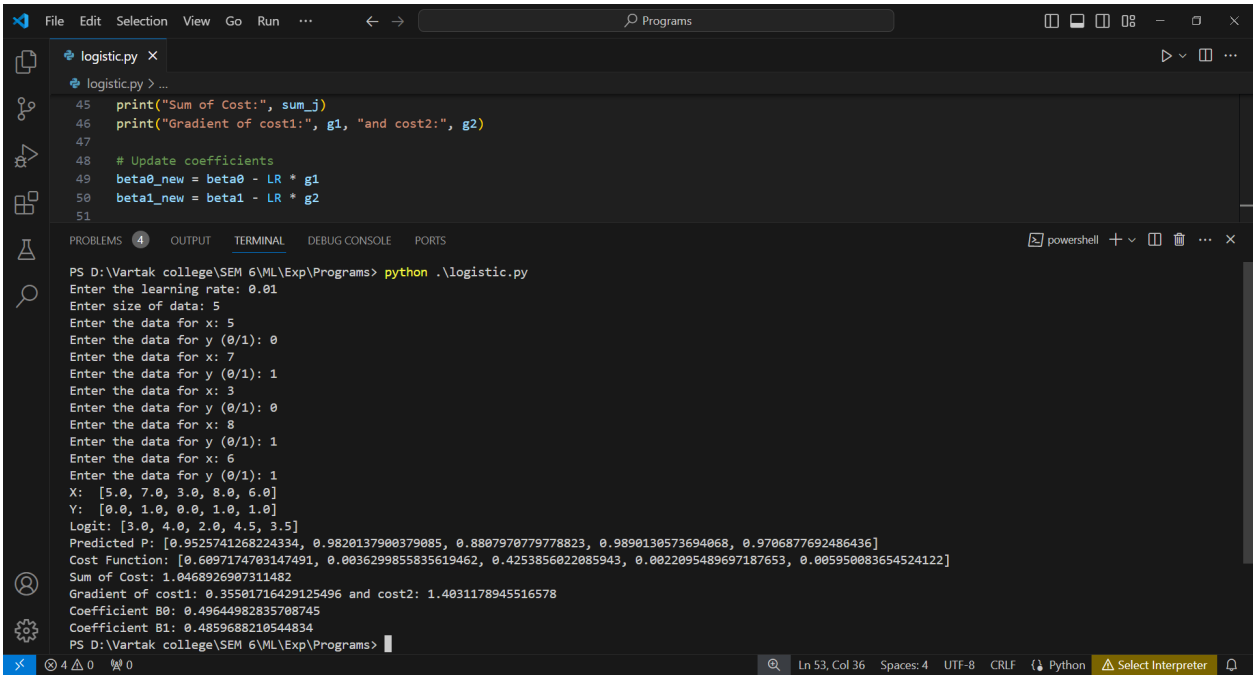
Program:

Logistic Regression:

```
import numpy as np
x = []
y = []
# Initialize coefficients
beta0 = 0.5
beta1 = 0.5
logit = []
p = []
j = []
sum_j = 0
# Learning rate
LR = float(input("Enter the learning rate: "))
# Example usage
n = int(input("Enter size of data: "))
# Collect data points
for i in range(n):
    x.append(float(input("Enter the data for x: ")))
    y.append(float(input("Enter the data for y (0/1): ")))
print("X: ",x)
print("Y: ",y)
for i in range(n):
    logit_i = (beta0 + beta1 * x[i])
    p_i = 1 / (1 + np.exp(-logit_i))
    j_i = (-1/5) * ((y[i] * np.log(p_i)) + (1 - y[i]) * np.log(1 - p_i))
    sum_j += j_i
    logit.append(logit_i)
    p.append(p_i)
    j.append(j_i)
# Compute gradients
g1 = (1/n) * sum(p_i - y_i for p_i, y_i in zip(p, y))
g2 = (1/n) * sum((p_i - y_i) * x_i for p_i, y_i, x_i in zip(p, y, x))
print("Logit:", logit)
print("Predicted P:", p)
print("Cost Function:", j)
print("Sum of Cost:", sum_j)
print("Gradient of cost1:", g1, "and cost2:", g2)
```

```
# Update coefficients
beta0_new = beta0 - LR * g1
beta1_new = beta1 - LR * g2
print("Coefficient B0:", beta0_new)
print("Coefficient B1:", beta1_new)
```

Output:



Conclusion:

Logistic regression, a fundamental classification algorithm, utilizes the sigmoid function to predict categorical outcomes. Unlike linear regression, logistic regression is bounded between 0 and 1, making it suitable for binary classification tasks. By iteratively updating coefficients based on gradients, logistic regression optimizes the likelihood function to maximize accuracy. Feature engineering plays a crucial role in enhancing model performance, allowing for better discrimination between classes. Overall, logistic regression is a powerful tool for binary classification tasks, widely used in various domains for predicting outcomes such as spam emails or tumor malignancy.