Experiment No. 3
Implementation of Logistic Regression Algorithm
Date of Performance:
Date of Submission:
Marks:
Sign:

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:

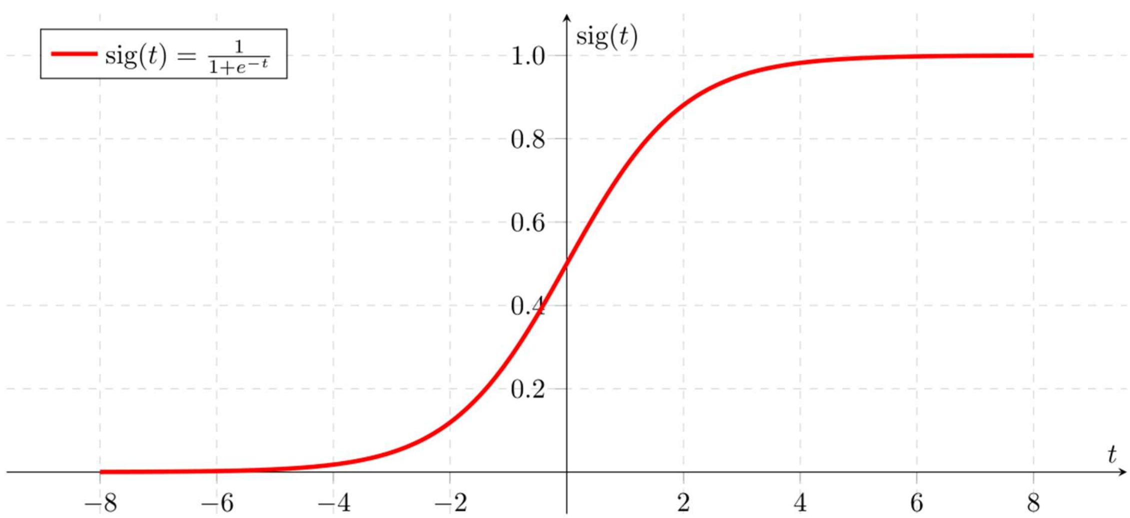
Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical and is binary in nature. In order to perform binary classification, the logistic regression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

Implementation:

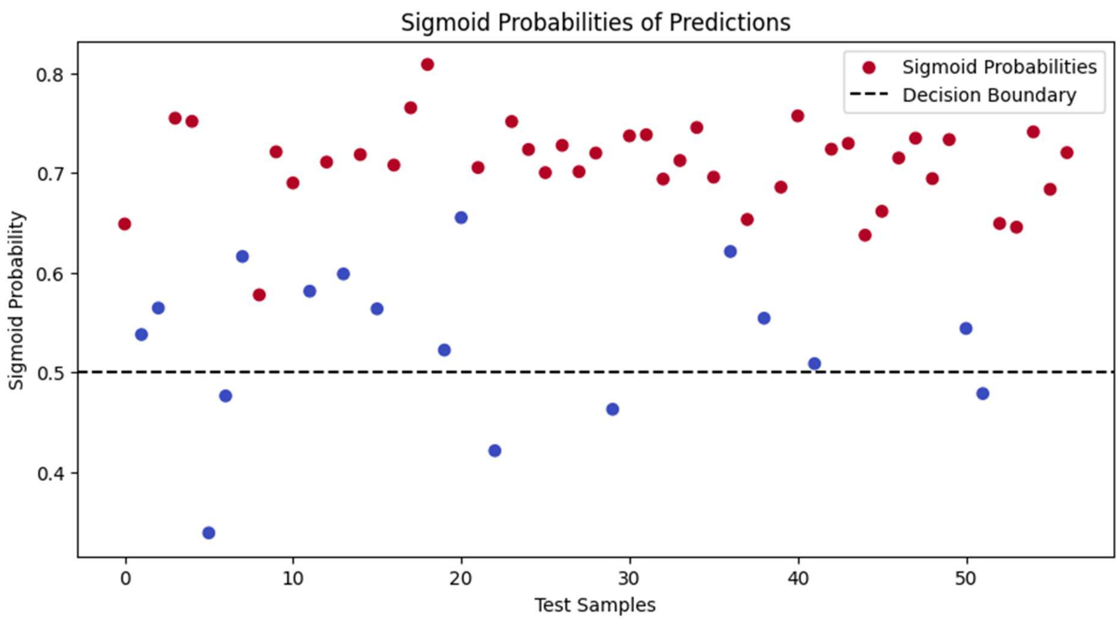
Code:

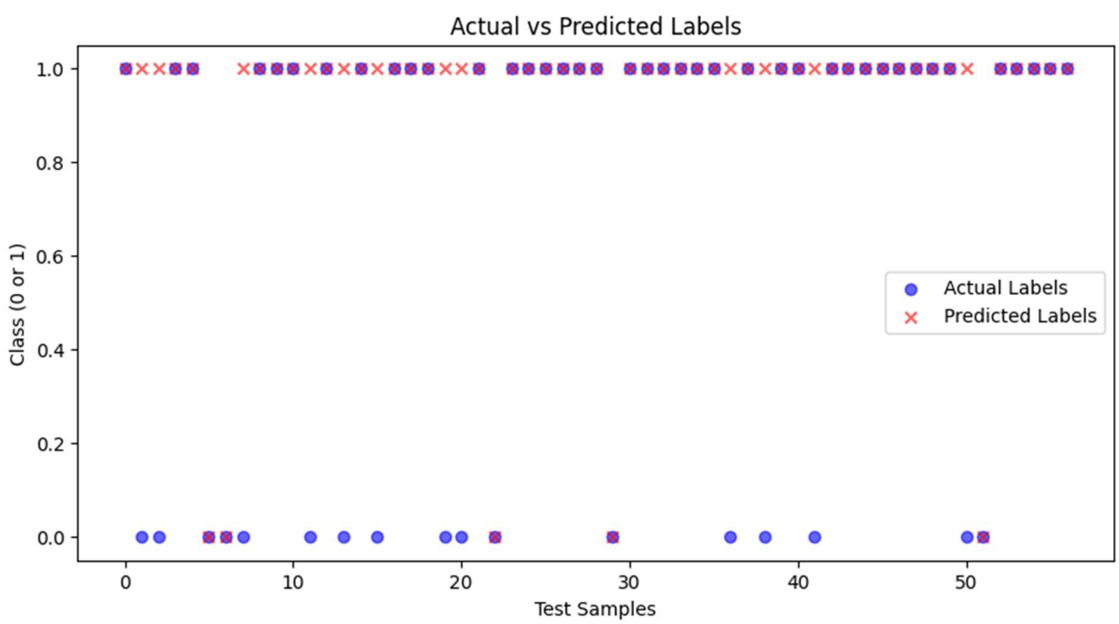
```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load_breast_cancer

from sklearn.model_selection import train_test_split


class LogisticRegression:

    def __init__(self):

        self.params = None


    def fit(self, X, y):

        bias = np.ones(len(X))

        X_bias = np.c_[bias, X]

        inner_part = np.transpose(X_bias) @ X_bias

        inverse_part = np.linalg.inv(inner_part)

        outer_part = inverse_part @ np.transpose(X_bias)

        self.params = outer_part @ y

        return self.params
```

```
    def predict(self, X):
        bias_testing = np.ones(len(X))
        X_test = np.c_[bias_testing, X]
        z = X_test @ self.params
        sigmoid = 1 / (1 + np.exp(-z))
        y_hat = (sigmoid >= 0.5).astype(int)
        return sigmoid, y_hat


# Load dataset
dataset = load_breast_cancer()
X = dataset.data
y = dataset.target


# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)


# Train model
model = LogisticRegression()
model.fit(X_train, y_train)


# Predict
```

```
sigmoid_vals, y_pred = model.predict(X_test)



# Plot sigmoid values

plt.figure(figsize=(10, 5))

plt.scatter(range(len(sigmoid_vals)), sigmoid_vals, c=y_test, cmap='coolwarm', label="Sigmoid Probabilities")

plt.axhline(y=0.5, color='black', linestyle='--', label="Decision Boundary")

plt.xlabel("Test Samples")

plt.ylabel("Sigmoid Probability")

plt.title("Sigmoid Probabilities of Predictions")

plt.legend()

plt.show()

# Plot actual vs predicted

plt.figure(figsize=(10, 5))

plt.scatter(range(len(y_test)), y_test, color='blue', label="Actual Labels", alpha=0.6)

plt.scatter(range(len(y_pred)), y_pred, color='red', marker='x', label="Predicted Labels", alpha=0.6)

plt.xlabel("Test Samples")

plt.ylabel("Class (0 or 1)")

plt.title("Actual vs Predicted Labels")

plt.legend()

plt.show()
```

Output:


Sigmoid Probabilities of Predictions
Sigmoid Probabilities
Decision Boundary
Sigmoid Probability
Test Samples
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40
50


Actual vs Predicted Labels
Actual Labels
Predicted Labels
Class (0 or 1)
Test Samples
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50

Conclusion:

Comment on the accuracy obtained.

- The accuracy score indicates how well the model is classifying the test samples.
- If accuracy is above 85-90%, the model performs well for this dataset.
- If accuracy is below 70%, it suggests that the model might need improvements like feature scaling, regularization, or a more robust optimization algorithm.
- The model in its current state does not use gradient descent, which is typically required for logistic regression in real-world scenarios.