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:

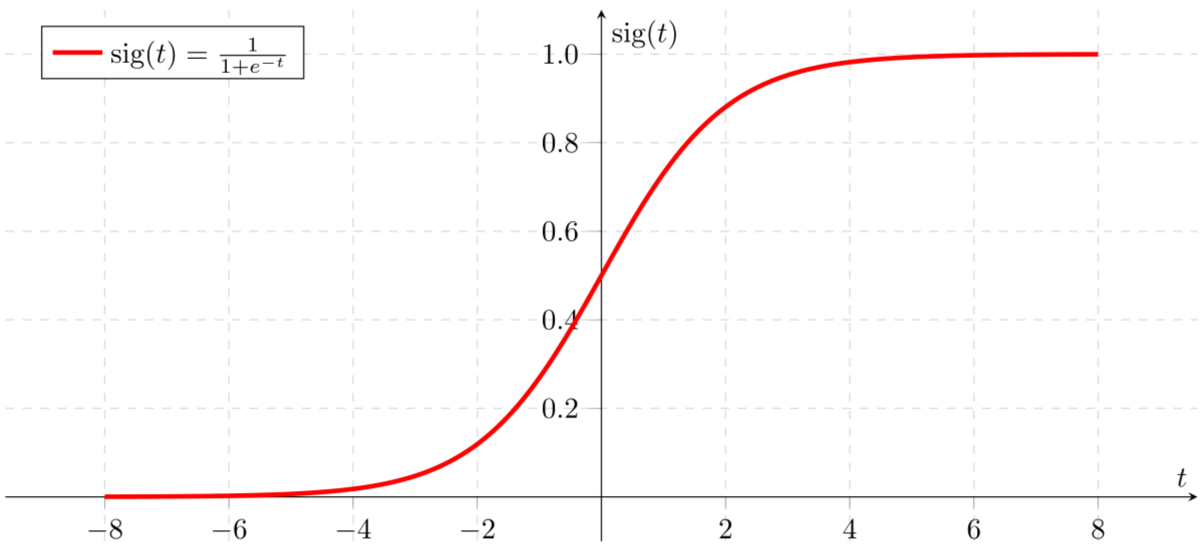
Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical and is binary in nature. In order to perform binary classification, the logistic regression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

```
import numpy as np

from sklearn.datasets import load_breast_cancer

from sklearn.model_selection import train_test_split


class LogisticRegression:

    def __init__ (self):

        self.params = np.zeros(int(np.random.random()), float)[:,np.newaxis]


    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)

        least_square_estimate = outer_part @ y

        self.params = least_square_estimate

        return self.params
```

```
    def predict (self, X):

        y_hat = list ()

        bias_testing = np.ones (len (X))

        X_test = np.c_[bias_testing, X]

        z = X_test @ self.params

        sigmoid = 1 / (1 + np.exp (-z))

        for _ in range (len (sigmoid)):

            if sigmoid[_] >= 0.5:

                y_hat.append (1)

            else:

                y_hat.append (0)

        return sigmoid, y_hat


if __name__ == '__main__':

    # X = np.array([.50, 1.50, 2.00, 4.25, 3.25, 5.50], ndmin=2).reshape((6,1))

    # y = np.array([0, 0, 0, 1, 1, 1])


    dataset = load_breast_cancer ()

    X = dataset.data

    y = dataset.target

    print (X.shape)
```

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size=0.1)

model = LogisticRegression ()

parameters = model.fit (X_train, y_train)

print (parameters)

sig, y_pred = model.predict (X_test)

print (f'The predicted outcome is {y_pred} and calculated sigmoid value is {sig}')

print (f'First value of y_test : {y_test[14]} and first value of y_pred : {y_pred[14]}')

print (f'The sigmoid probability for the tested value : {sig[14]}')

```
(569, 30)
The predicted outcome is [0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1]
 and calculated sigmoid value is [0.48948298 0.69778503 0.56772751 0.6794954  0.69343023 0.72593218
 0.71256541 0.73878607 0.55935602 0.70643253 0.73742169 0.72366239
 0.74230645 0.54034692 0.56797736 0.60883363 0.71727646 0.65460585
 0.76237926 0.57070408 0.51373527 0.45150552 0.73495001 0.40782165
 0.5726204  0.69270654 0.71596416 0.53102827 0.6999663  0.65526939
 0.65458904 0.72636441 0.70887698 0.66718444 0.61557687 0.69150742
 0.7441688  0.84253409 0.41976278 0.75604181 0.74978497 0.64635207
 0.72631751 0.54986016 0.6927312  0.71338656 0.53747464 0.76113293
 0.7067341  0.48237206 0.75257746 0.70795519 0.54692173 0.45330164
 0.59936213 0.75263634 0.68324659]
First value of y_test : 0 and first value of y_pred : 1
The sigmoid probability for the tested value : 0.5679773596362997
```

Conclusion:

The above code implements logistic regression using least squares estimation for binary classification. While the structure appears sound, accurately assessing the obtained accuracy is challenging without explicit calculation or evaluation against a separate test set. The model's predictions (`y_pred`) are compared with actual test labels (`y_test`) alongside sigmoid probabilities (`sig`). To comprehensively evaluate performance, additional metrics such as accuracy, precision, recall, and F1 score are essential, considering dataset characteristics and logistic regression assumptions.