Shri Vile Parle Kelavani Mandal's
**DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING**
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

**Continuous Assessment for Laboratory / Assignment sessions**

Academic Year 2022-23

Name: Ayush Jain SAP ID: 60004200182

Course: Machine Learning Laboratory Course Code: **DJ19CEEL6021**

Year: **T.Y. B.Tech.** Sem: **VII** Batch: B 3

**Department: Computer Engineering**

| Performance Indicators (Any no. of Indicators) (Maximum 5 marks per indicator) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | Σ | Avg | A1 | A2 | Σ | Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Course Outcome | 2, 4 | 2, 4 | 2, 4 | 2, 4 | 2, 4 | 3 | 2, 4 | 2, 4 | 5 | | | | | | | | |
| 1. Knowledge (Factual/Conceptual/Procedural/ Metacognitive) | 4 | 5 | 5 | 5 | | | | | | | | | | | | | |
| 2. Describe (Factual/Conceptual/Procedural/ Metacognitive) | 4 | 4 | 4 | 5 | | | | | | | | | | | | | |
| 3. Demonstration (Factual/Conceptual/Procedural/ Metacognitive) | 5 | 5 | 5 | 5 | | | | | | | | | | | | | |
| 4. Strategy (Analyse & / or Evaluate) (Factual/Conceptual/ Procedural/Metacognitive) | 4 | 4 | 4 | 4 | | | | | | | | | | | | | |
| 5. Interpret/ Develop (Factual/Conceptual/ Procedural/Metacognitive) | – | – | – | – | | | | | | | | | | | | | |
| 6. Attitude towards learning (receiving, attending, responding, valuing, organizing, characterization by value) | 4 | 4 | 4 | 4 | | | | | | | | | | | | | |
| 7. Non-verbal communication skills/ Behaviour or Behavioural skills (motor skills, hand-eye coordination, gross body movements, finely coordinated body movements speech behaviours) | – | – | – | – | | | | | | | | | | | | | |
| Total | 21 | 22 | 22 | 23 | | | | | | | | | | | | | |
| Signature of the faculty member | | | | | | | | | | | | | | | | | |

Outstanding (5), Excellent (4), Good (3), Fair (2), Needs Improvement (1)

| Laboratory marks Σ Avg. = | Assignment marks Σ Avg. = | Total Term-work (25) = |
|---|---|---|
| Laboratory Scaled to (15) = | Assignment Scaled to (10) = | Sign of the Student: |

Signature of the Faculty member:
Name of the Faculty member:

Signature of Head of the Department
Date:

# Machine Learning

Experiment 2

Ayush Jain 60004200132 B3

**Aim :** To implement univariate logistical regression in python with and without using any inbuilt function.

**Theory:**
Logistic regression is a type of regression analysis used to predict the probability of a binary outcome (i.e., 0 or 1). It is commonly used in machine learning and statistics to model relationships between a set of predictor variables (also called independent variables or features) and a binary outcome variable (also called a dependent variable or response).

The logistic regression model is based on the logistic function (also called the sigmoid function), which maps any real-valued input to an output between 0 and 1. This output represents the probability of the positive class (i.e., 1) given the input values. The formula for the logistic function is:

It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

**Linear Regression Equation:**

$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ..... + \beta_n X_n$

Where, y is a dependent variable and x1, x2 ... and Xn are explanatory variables.

**Sigmoid Function:**

$$p = \frac{1}{1 + e^{-y}}$$

**Apply Sigmoid function on linear regression:**

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ..... + \beta_n X_n)}}$$

**Properties of Logistic Regression:**

- The dependent variable in logistic regression follows Bernoulli Distribution.
- Estimation is done through maximum likelihood.

- No R Square, Model fitness is calculated through Concordance, KS-Statistics.

To apply logistic regression to a dataset, we typically first split the data into training and testing sets. Then, we fit a logistic regression model to the training data, using the predictor variables to predict the binary outcome variable. The logistic regression model estimates the probability of the positive class given the input values, based on the values of the predictor variables and the learned model parameters.

To evaluate the performance of the logistic regression model, we typically use metrics such as accuracy, precision, recall, and F1 score. These metrics provide an assessment of how well the model is able to predict the binary outcome, and can be used to compare different models or parameter settings.

**Code without inbuilt function:**

```
class LogisticRegression(object):

    def __init__(self, eta=0.01, n_iter=1000):
        self.eta = eta
        self.n_iter = n_iter

    def fit(self, X, y):
        self.w_ = np.zeros(1 + X.shape[1])
        self.cost_ = []
        for i in range(self.n_iter):
            y_val = self.activation(X)
            errors = (y - y_val)
            neg_grad = X.T.dot(errors)
            self.w_[1:] += self.eta * neg_grad
            self.w_[0] += self.eta * errors.sum()
            self.cost_.append(self._logit_cost(y, self.activation(X)))
        return self

    def _logit_cost(self, y, y_val):
        logit = -y.dot(np.log(y_val)) - ((1 - y).dot(np.log(1 - y_val)))
        return logit

    def _sigmoid(self, z):
        return 1.0 / (1.0 + np.exp(-z))

    def net_input(self, X):
        """Calculate net input"""
```

```
        return np.dot(X, self.w_[1:]) + self.w_[0]

    def activation(self, X):
        """ Activate the logistic neuron"""
        z = self.net_input(X)
        return self._sigmoid(z)

      def predict(self, X):

        # equivalent to np.where(self.activation(X) >= 0.5, 1, 0)
        return np.where(self.net_input(X) >= 0.0, 1, 0)

import pandas as pd

df = pd.read_csv('https://archive.ics.uci.edu/ml/'
        'machine-learning-databases/iris/iris.data', header=None)
df.tail()
```

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 145 | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| 146 | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| 147 | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| 148 | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| 149 | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

```
import numpy as np

# select setosa and versicolor
y = df.iloc[0:100, 4].values
y = np.where(y == 'Iris-setosa', 1, 0)

# extract sepal length and petal length
X = df.iloc[0:100, [0, 2]].values

# standardize features
X_std = np.copy(X)
X_std[:,0] = (X[:,0] - X[:,0].mean()) / X[:,0].std()
```

```
X_std[:,1] = (X[:,1] - X[:,1].mean()) / X[:,1].std()

from matplotlib.colors import ListedColormap

def plot_decision_regions(X, y, classifier, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # plot class samples
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                    alpha=0.8, c=cmap(idx),
                    marker=markers[idx], label=cl)

score = logisticRegr.score(x_test, y_test)
print("Accuracy score: " + str(round(score*100.00,2)) + "%")

cm = metrics.confusion_matrix(y_test, predictions)
print("Confusion matrix: \n", cm)
```

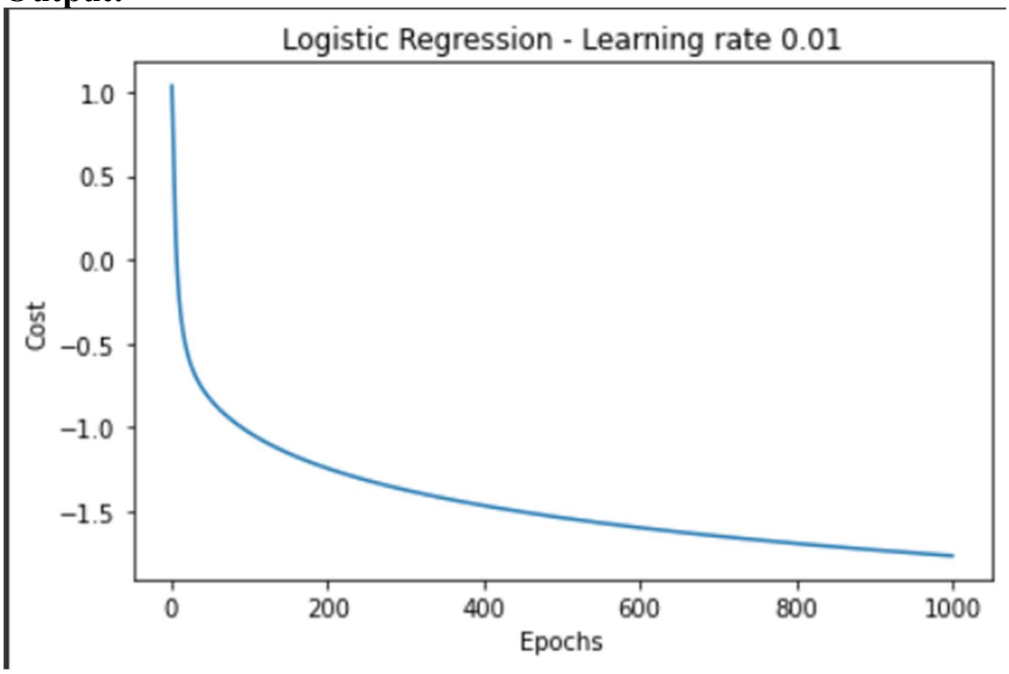
```
Accuracy score: 95.93%
Confusion matrix:
 [[45  0  0  0  0  0  0  0  0  0]
 [ 0 48  0  0  0  0  1  0  2  1]
 [ 0  2 50  1  0  0  0  0  0  0]
 [ 0  0  0 53  0  0  0  0  1  0]
 [ 0  0  0  0 48  0  0  0  0  0]
 [ 0  0  0  0  0 53  1  0  0  3]
 [ 0  1  0  0  0  0 59  0  0  0]
 [ 0  0  0  0  1  0  0 52  0  0]
 [ 0  3  1  0  0  0  0  0 55  2]
 [ 0  0  0  1  0  1  0  0  0 55]]
```

```
import matplotlib.pyplot as plt
```

```
lr = LogisticRegression(n_iter=1000, eta=0.2).fit(X_std, y)
plt.plot(range(1, len(lr.cost_) + 1), np.log10(lr.cost_))
plt.xlabel('Epochs')
plt.ylabel('Cost')
plt.title('Logistic Regression - Learning rate 0.01')

plt.tight_layout()
plt.show()
```

**Output:**




**Code with inbuilt function:**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing  import StandardScaler
from sklearn.pipeline import make_pipeline

data = pd.read_csv("data.csv")
data.drop(['Unnamed: 32',"id"], axis=1, inplace=True)
data.diagnosis = [1 if each == "M" else 0 for each in data.diagnosis]
y = data.diagnosis.values
x = data.drop(['diagnosis'], axis=1)

from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15,
random_state=42)

from sklearn import linear_model
logreg =
make_pipeline(StandardScaler(),linear_model.LogisticRegression(random_state
= 42,max_iter= 150))
print("test accuracy: {} ".format(logreg.fit(x_train, y_train).score(x_test,
y_test)))
print("train accuracy: {} ".format(logreg.fit(x_train, y_train).score(x_train,
y_train)))
```

**Output:**

```
test accuracy: 0.9767441860465116
train accuracy: 0.9875776397515528
```

**Conclusion:** We successfully implemented univariate logistical regression in python with and without using any inbuilt function.