EXPERIMENT NO. 2

AIM: Study and Implement Logistic Regression.

Theory:

Logistic Regression is a statistical method used for binary classification problems, where the output is categorical and typically binary (e.g., "yes" or "no", "0" or "1"). Unlike linear regression, where the output is continuous, logistic regression predicts the probability of an instance belonging to a specific class. It is widely used in machine learning for tasks such as predicting whether a customer will buy a product, whether a patient has a disease, or whether a passenger will survive a disaster, as in the Titanic dataset.

The logistic regression model uses the following steps:

1. Logistic Function (Sigmoid Function):

The core of logistic regression is the **sigmoid function**, which maps any real-valued number to a value between 0 and 1. This makes it useful for predicting probabilities. The sigmoid function is given by:

$$\sigma(z) = 1/ (1+e^{-z})$$

Where:

- z is the linear combination of input features and weights (i.e., $z = w_1x_1 + w_2x_2 + ... + w_nx_n + b$ where w represents the weights, x the features, and b the intercept).
- The sigmoid function ensures that the output is between 0 and 1, representing the probability that the instance belongs to a particular class.

2. Decision Boundary:

Logistic regression uses a decision boundary at 0.5 to classify an instance:

- If the predicted probability is greater than or equal to 0.5, the instance is classified as class 1.
- If the predicted probability is less than 0.5, it is classified as class 0.

3. Cost Function:

Logistic regression uses a different cost function than linear regression because it deals with probabilities. The **log-loss** or **binary cross-entropy loss** is used to measure the error in the predictions. The goal of training the model is to minimize this cost function, thereby improving the accuracy of predictions.

The cost function is given by:

$$\text{Cost}(h_0(x),y) = - [y\log(h_0(x)) + (1-y)\log(1-(h_0(x))]$$

Where:

- $h_0(x)$ is the predicted probability of class 1,
- y is the true class label (either 0 or 1).

4. Optimization:

To find the optimal weights (w) and intercept (b), logistic regression uses **gradient descent**. The gradient descent algorithm iteratively adjusts the weights to minimize the cost function until convergence is reached.

5. Applications:

Logistic regression is widely used in many fields:

- Predicting whether a customer will buy a product (e.g., in marketing).
- Classifying whether an email is spam or not (e.g., in spam filters).
- Predicting whether a patient has a disease based on medical data.
- Predicting whether a passenger survived on the Titanic, as shown in the code below.

Code:

```
# In[1]:
#Importing the libraries
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# In[2]:
#import csv files
data = pd.read_csv('train.csv')
data.head()

# In[3]:
plt.scatter(data['Age'], data['Survived'], marker='+', color='red')
plt.xlabel('Age')
plt.ylabel('Survived (0 = No, 1 = Yes)')
plt.show()

#In[4]:
X = data[['Age', 'Pclass']]  # Features: Age and Pclass (Passenger Class)
y = data['Survived']
```

```
#In[5]:
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=42)

#In[6]:
model = LogisticRegression()
model.fit(X_train, y_train)

#In[7]:
y_predicted = model.predict(X_test)
y_probabilities = model.predict_proba(X_test)
print(y_probabilities)

#In[8]:
accuracy = model.score(X_test, y_test)
print(f"Accuracy: {accuracy}")

#In[9]:
print(f"Model Coefficients: {model.coef_}")
print(f"Model Intercept: {model.intercept_}")

#In[10]:
import math
def sigmoid(x):
return 1 / (1 + math.exp(-x))

#In[11]:
def prediction_function(age, pclass):
z = model.coef_[0][0] * age + model.coef_[0][1] * pclass + model.intercept_[0]
y = sigmoid(z)
return y

#In[12]:
print(prediction_function(35, 2))

#In[13]:
predictions_df = pd.DataFrame({'Age': X_test['Age'], 'Pclass': X_test['Pclass'],
' Predicted_Survived': y_predicted})
predictions_df.to_csv('titanic_logistic_regression_predictions.csv', index=False)
```

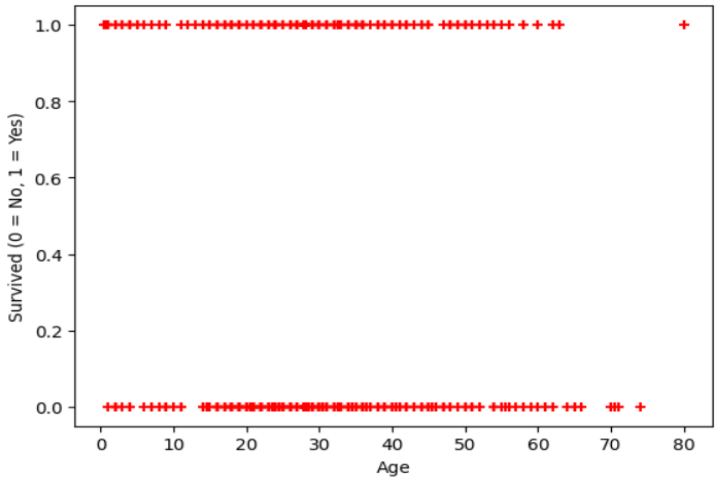
Output Snapshots:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
C:\Users\anshi\AppData\Local\Temp\ipykernel_14904\913469570.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  data['Age'].fillna(data['Age'].median(), inplace=True)
```



LogisticRegression

LogisticRegression()

```
[[0.76599239 0.23400761]
 [0.57149249 0.42850751]
 [0.71899014 0.28100986]
 [0.38179552 0.61820448]
 [0.6801992  0.3198008 ]
 [0.2981144  0.7018856 ]
 [0.76599239 0.23400761]
 [0.6934456  0.3065544 ]
 [0.6934456  0.3065544 ]
 [0.25504977 0.74495023]
 [0.37342597 0.62657403]
 [0.84271111 0.15728889]
 [0.76599239 0.23400761]
 [0.77685099 0.22314901]
 [0.60871608 0.39128392]
 [0.23789698 0.76210302]
 [0.41009523 0.58990477]
 [0.76599239 0.23400761]
```

```
Accuracy: 0.7541899441340782
```

```
Model Coefficients: [[-0.03079509 -0.99026077]]
Model Intercept: [2.64722617]

0.39864277196351866
```

Learning Outcome: