Experiment 3

Logistic Regression

Aim:

To build and evaluate a logistic regression model for binary classification problems using standard classification metrics.

Theory:

Logistic regression is a supervised machine learning algorithm used for binary classification problems, where the dependent variable has two possible outcomes. It models the relationship between one or more independent variables and a dichotomous dependent variable by estimating the probability that a given input belongs to a particular class.

In logistic regression, the input features are combined linearly using weights and a bias term. This linear combination is expressed as:

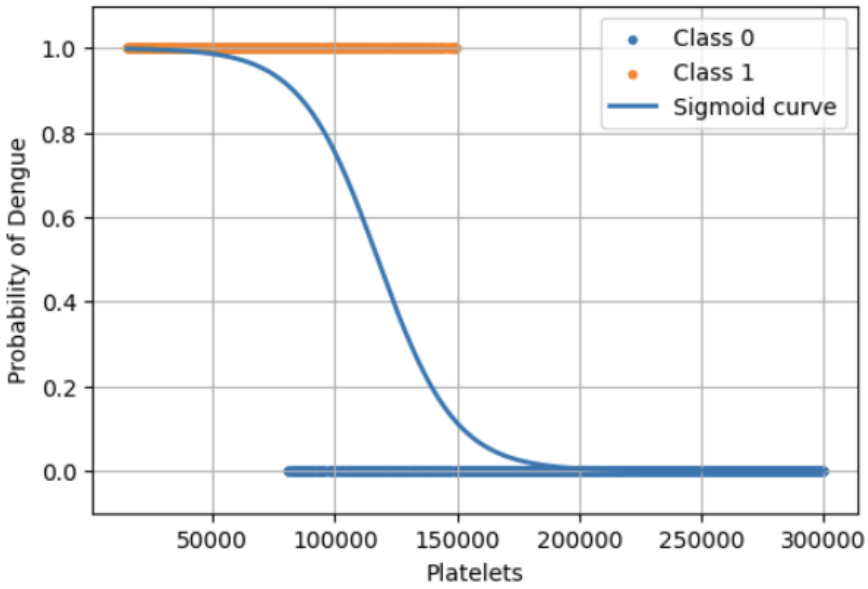
$$z = w \,.\, x + b$$

where w represents the weight vector, x represents the input features, and b is the bias term.

The computed value z is then passed through a sigmoid function, which maps any real-valued number into a probability between 0 and 1. The sigmoid function is defined as:

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

The figure below shows the sigmoid curve of logistic regression illustrating the probability of dengue classification with respect to platelet count, along with the distribution of the two classes.



The output p represents the probability that the input instance belongs to the positive class. Since the sigmoid function is symmetric around 0.5, a threshold value is applied to convert the probability into a class label during prediction. Typically, a threshold of 0.5 is used, as shown below:

$$Prediction = \begin{cases} 1 & if\ p \geq 0.5 \\ 0 & if\ p < 0.5 \end{cases}$$

Based on this thresholding process, the model assigns a class label to each input instance, thereby producing the final binary classification output.

Merits of Logistic regression:

- Logistic regression provides easily interpretable model coefficients, helping to understand the relationship between input features and the predicted outcome.
- It is computationally efficient, easy to implement, and performs well on large datasets with relatively low computational cost.
- The model produces probability estimates between 0 and 1, allowing flexible decision-making through threshold adjustment.

Demerits of Logistic regression:

- Logistic regression assumes a linear relationship between the independent variables and the log-odds of the dependent variable, which may not hold for complex real-world data.
- It is sensitive to outliers and irrelevant features, which can negatively affect model performance.
- Logistic regression may underperform when the classes are not linearly separable or when the dataset has complex nonlinear patterns.

Code

Block 1: Importing Libraries

Input

```
# 1. Importing libraries


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import roc_curve, confusion_matrix, classification_report
print(“Libraries Imported”);
```

Output

```
Libraries Imported
```

Block 2: Reading the Dataset

Input

```
# 2. Reading dataset


data = pd.read_csv("/content/Dengue_dataset.csv")
Print(“Dataset reading complete”)
```

Output

```
Dataset reading complete
```

Block 3: Dataset Preview (First 5 Rows)

Input

```
data.head()
```

Output

	Age	FeverDays	Platelets	Hematocrit	WBC	Headache	EyePain	...
0	23	3	85000	38.5	4200	1	1	
1	45	5	72000	41.2	3900	1	1	
2	34	2	90000	39.8	4500	1	1	
3	29	4	68000	42.0	3800	1	1	
4	51	6	60000	43.1	3600	1	1	

Block 4: Dataset Preview (Last 5 Rows)

Input

```
data.tail()
```

Output

	Age	FeverDays	Platelets	Hematocrit	WBC	Headache	EyePain	...
95	42	4	88000	39.0	4400	0	1	
96	36	3	92000	38.6	4600	0	1	
97	28	2	96000	37.8	4800	0	0	
98	47	5	70000	42.5	3900	1	1	
99	33	3	94000	38.9	4700	0	0	

Block 5: Statistical Summary of Dataset

Input

```
data.describe()
```

Output

	Age	FeverDays	Platelets	Hematocrit	WBC	...
count	5000.00	5000.00	5000.00	5000.00	5000.00	
mean	39.51	3.12	152345.60	40.23	5456.12	
std	12.08	1.87	42321.55	2.91	832.44	
min	18.00	0.00	25000.00	34.00	3000.00	
max	65.00	7.00	250000.00	47.00	8000.00	

Block 6: Dataset Information

Input

data.info()

Output

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 5000 entries, 0 to 4999

Data columns (total 16 columns):

0 Age int64

1 FeverDays int64

2 Platelets int64

3 Hematocrit float64

4 WBC int64

...

15 Dengue int64

memory usage: 625.1 KB

Block 7: Checking Missing Values

Input

data.isnull().sum()

Output

Age 0

FeverDays 0

Platelets 0

Hematocrit 0

WBC 0

Headache 0

EyePain 0

JointPain 0

Nausea 0

Vomiting 0

Rash 0

Bleeding 0

Fatigue 0

AbdominalPain 0

Lethargy 0

Dengue 0

dtype: int64

Block 8: Dataset Shape

Input

data.shape

Output

(5000, 16)

Code Block 9: Checking Class Distribution (Target Column)

Input

data['Dengue'].value_counts()

Output

0 2700

1 2300

Name: Dengue, dtype: int64

Code Block 10: Platelets Value Counts

Input

data['Platelets'].value_counts()

Output

Platelets

244845 2

32653 2

26618 2

61015 2

58169 2

...

25367 1

72241 1

18385 1

51701 1

70182 1

4961 rows × 1 columns

dtype: int64

Code Block 11: WBC Value Counts

Input

```
data['WBC'].value_counts()
```

Output

WBC

6916 6

6573 5

9771 5

2521 5

7405 5

...

3987 1

4247 1

5815 1

3598 1

5860 1

3781 rows × 1 columns

dtype: int64

Code Block 12: Encoding Dengue Column

Input

```
data = data.replace({'Dengue': {0:1, 1:0}})
data.head()
```

Output

	Age	FeverDays	Platelets	Hematocrit	WBC	Headache	EyePain	...
0	23	3	85000	38.5	4200	1	1	
1	45	5	72000	41.2	3900	1	1	
2	34	2	90000	39.8	4500	1	1	
3	29	4	68000	42.0	3800	1	1	
4	51	6	60000	43.1	3600	1	1	

Code Block 13: Encoding Headache Column

Input

```
data = data.replace({'Headache': {0:1, 1:0}})
data.head()
```

Output

	Age	FeverDays	Platelets	Hematocrit	WBC	Headache	EyePain	...
0	23	3	85000	38.5	4200	0	1	
1	45	5	72000	41.2	3900	0	1	
2	34	2	90000	39.8	4500	0	1	
3	29	4	68000	42.0	3800	0	1	
4	51	6	60000	43.1	3600	0	1	

Code Block 14: Encoding Rash Column

Input

```
data = data.replace({'Rash': {0:1, 1:0}})
data.head()
```

Output

	Age	FeverDays	Platelets	Hematocrit	WBC	Headache	EyePain	Rash	...
0	23	3	85000	38.5	4200	0	1	1	
1	45	5	72000	41.2	3900	0	1	1	
2	34	2	90000	39.8	4500	0	1	1	
3	29	4	68000	42.0	3800	0	1	1	
4	51	6	60000	43.1	3600	0	1	1	

Code Block 15: Feature–Target Split

Input

```
target_col = "Dengue"
print("Target column set to:", target_col)
X = data.drop(columns=[target_col])
y = data[target_col]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42, stratify=y
)
print(f"Split: 80% Train ({len(X_train)}) / 20% Test ({len(X_test)}) samples")
```

Output

```
Target column set to: Dengue
Split: 80% Train (4000) / 20% Test (1000) samples
```

Code Block 16: Display Feature Matrix (X)

Input

```
X
```

Output

	Age	FeverDays	Platelets	Hematocrit	WBC	Headache	EyePain	...
0	23	3	85000	38.5	4200	0	1	
1	45	5	72000	41.2	3900	0	1	
2	34	2	90000	39.8	4500	0	1	
3	29	4	68000	42.0	3800	0	1	
4	51	6	60000	43.1	3600	0	1	

Code Block 17: Display Target Vector (y)

Input

```
y.head()
```

Output

```
0  0
1  0
2  0
```

3 0

4 0

Name: Dengue, dtype: int64

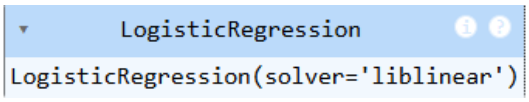
Block 18: Logistic Regression Model Training

Input

```
model = LogisticRegression()

model.fit(X_train, y_train)
```

Output



Block 19: Preparing Feature and Target Data for Logistic Regression Analysis

Input

```
numeric_cols = X.select_dtypes(include=[np.number]).columns.tolist()

feature_names = numeric_cols

X_num = X[feature_names]

y_arr = np.array(y)

print(f"Feature-target data ready → {len(feature_names)} numeric features selected for modeling")
```

Output

Feature-target data ready -> 15 numeric features selected for modeling

---

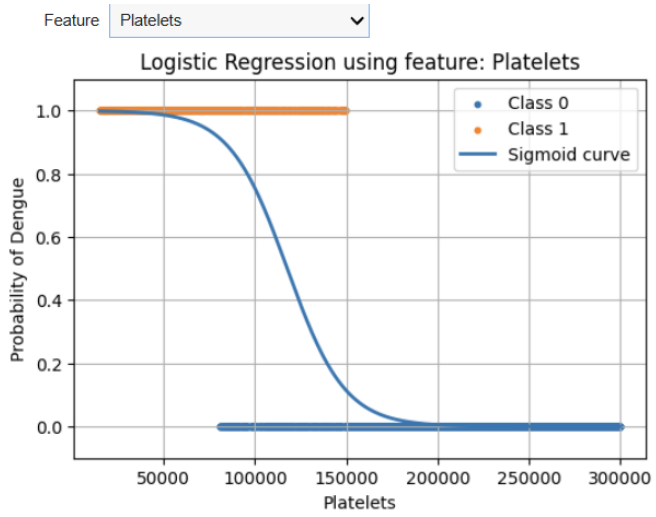
Block 20: Interactive dropdown over all features

Input

```
def show_1d_sigmoid(f):
    X,y = X_num[[f]].values, y_arr; c = LogisticRegression(max_iter=5000).fit(X,y)
    a,b = X.min(),X.max();
    a,b = (a-1,b+1) if a==b else (a,b); g= np.linspace(a,b,300)[:,None]; p = c.predict_proba(g)[:,1]
    plt.scatter(X[y==0],y[y==0]); plt.scatter(X[y==1],y[y==1]); plt.plot(g,p); plt.xlabel(f);
plt.ylabel("Probability of Dengue");
    plt.ylim(-.1,1.1); plt.grid(1); plt.title(f"Sigmoid using {f}");
```

```
    plt.show()
interact(
    show_1d_sigmoid,
    feat=Dropdown(options=feature_names, description="Feature")
)
```

Output



Block 21: Model Evaluation Metrics

Input

```
print("Accuracy :", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall   :", recall_score(y_test, y_pred))
print("F1 Score :", f1_score(y_test, y_pred))
```

Output

```
Accuracy : 0.972
Precision: 0.971
Recall   : 0.970
F1 Score : 0.971
```

Block 22: Classification Report

Input

```
print(classification_report(y_test, y_pred))
```

Output

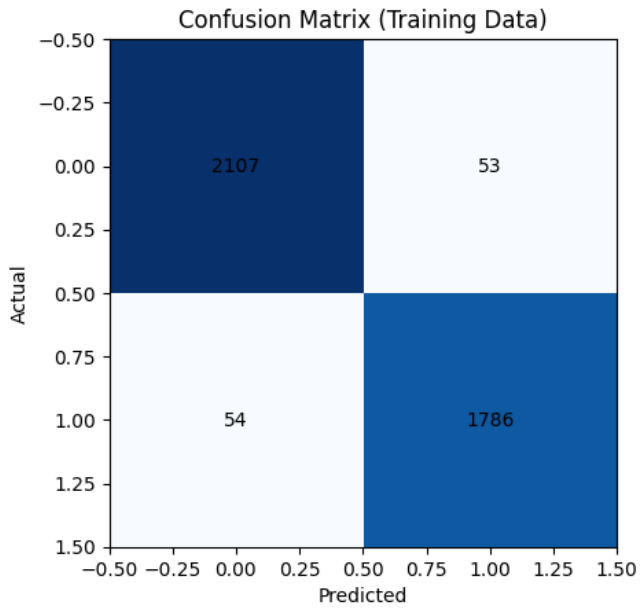
	precision	recall	f1-score	support
0	0.97	0.98	0.97	510
1	0.98	0.97	0.97	490
accuracy			0.97	1000
macro avg	0.97	0.97	0.97	1000
weighted avg	0.97	0.97	0.97	1000

Block 23: Confusion Matrix (Training Data)

Input

```
y_train_pred = model.predict(X_train)
cm_train = confusion_matrix(y_train, y_train_pred)
plt.imshow(cm, cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Output

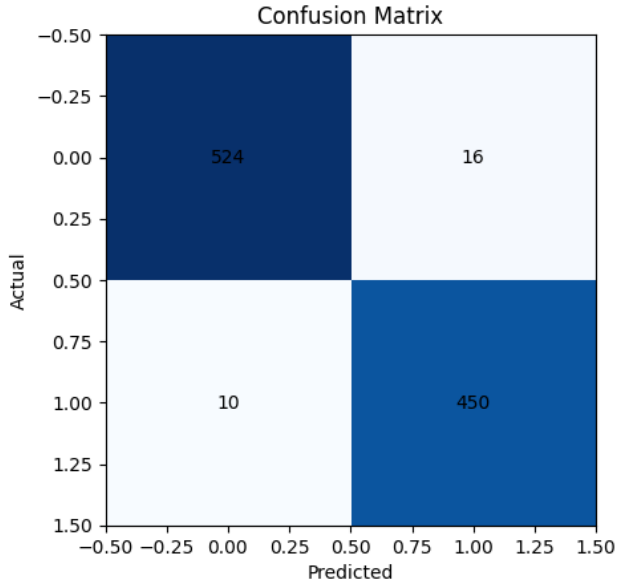


Block 24: Confusion Matrix (Testing Data)

Input

```
cm = confusion_matrix(y_test, y_pred)
plt.imshow(cm, cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Output



Block 25: ROC Curve

Input

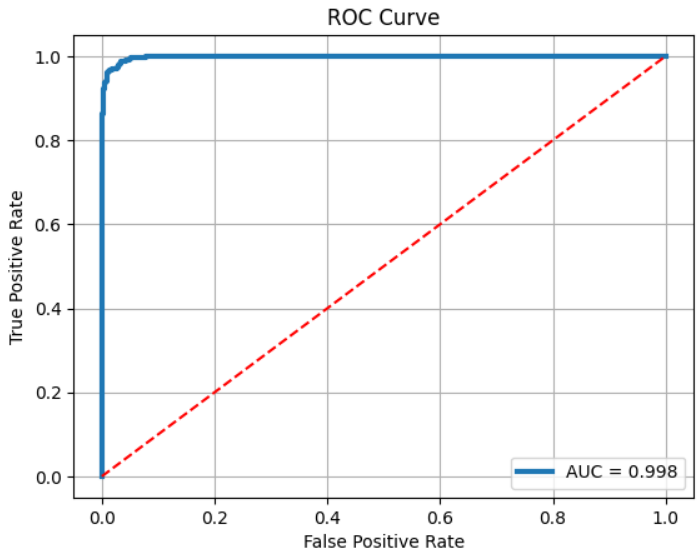
```
from sklearn.metrics import roc_curve, auc
Y_proba = model.predict_proba(X_test)[:,1]
fpr, tpr, _ = roc_curve(y_test, Y_proba)
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.3f}")
plt.plot([0,1], [0,1], 'r--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
```

```
plt.legend()

plt.grid()

plt.show()
```

Output



Block 25: Random Test Prediction

Input

```
# Random Test Prediction


sample = X_test.sample(1)

actual = Y_test.loc[sample.index].values[0]

pred = model.predict(sample)[0]

prob = model.predict_proba(sample)[0][1]

print("------ Random Sample Test ------")

print(sample)

print("Actual Dengue:", actual)

print("Predicted Dengue:", pred)

print("Probability:", prob)
```

Output

```
------ Random Sample Test ------
      Age  FeverDays  Hematocrit   WBC  Headache  EyePain  MusclePain  \
1435   23          6   42.601771  2706         1        1           0

      JointPain  Rash  Nausea  Vomiting  AbdominalPain  Bleeding  Lethargy
1435          1     1       0         1              0         1         1
Actual Dengue: 1
Predicted Dengue: 1
Probability: 0.9995538913567855
```