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:

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

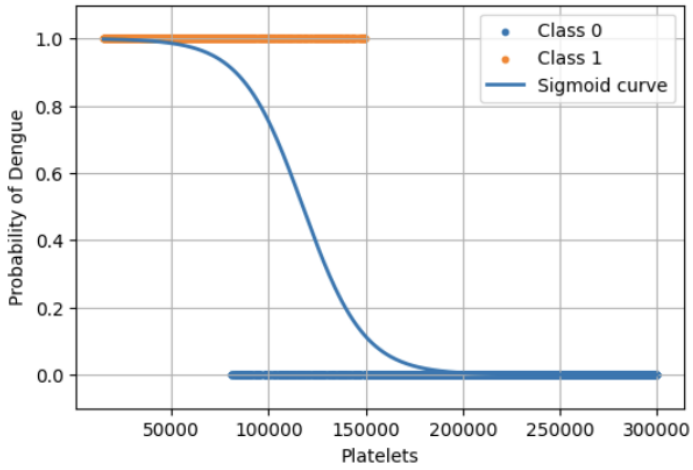
where:

- **w** represents the weight vector.
- **x** represents the input features.
- **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

index	Age	Fever Days	Platelets	Hematocrit	WBC	Headache	EyePain	Muscle Pain	Joint Pain	Rash	Nausea	Vomiting	Abdominal Pain	Bleeding	Lethargy	Dengue
0	61	6	56777	49.49572857	3205	0	1	0	0	0	1	1	0	0	0	1
1	24	7	61250	42.41093608	3738	1	0	0	0	1	0	0	1	1	0	1
2	70	7	88034	49.66143051	3052	1	0	1	0	0	1	0	0	0	0	1
3	30	6	55130	50.20159282	2713	1	0	0	1	0	0	1	1	0	0	1
4	33	3	64346	43.46980401	4888	1	1	0	1	1	1	1	0	1	0	1

Block 4: Dataset Preview (Last 5 Rows)

Input

data.tail()

Output

index	Age	Fever Days	Platelets	Hematocrit	WBC	Headache	EyePain	Muscle Pain	Joint Pain	Rash	Nausea	Vomiting	Abdominal Pain	Bleeding	Lethargy	Dengue
4995	59	2	123791	47.18861033	4239	1	0	0	0	0	0	0	0	0	0	0
4996	11	4	111575	41.84560446	5989	1	0	1	1	1	1	1	0	0	0	0
4997	19	2	128198	40.2842782	4775	1	1	1	0	0	0	0	0	0	1	0
4998	11	0	141619	47.3939162	4625	1	1	0	0	1	1	1	1	0	1	0
4999	69	2	90619	46.47148497	5982	1	0	0	0	1	0	1	0	0	0	0

Block 5: Statistical Summary of Dataset

Input

data.describe()

Output

	Age	Fever Days	Platelets	Hematocrit	WBC	Headache	Eye Pain	Muscle Pain	Joint Pain	Rash	Nausea	Vomiting	Abdominal Pain	Bleeding	Lethargy	Dengue
count	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0
mean	41.85	3.41	141522.48	41.86	6156.58	0.517	0.492	0.508	0.500	0.496	0.501	0.507	0.496	0.226	0.502	0.460
std	18.80	2.05	87116.73	4.81	2594.06	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.418	0.500	0.498
min	10.0	0.0	15060.0	32.02	2002.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	26.0	2.0	62365.5	38.43	3848.75	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50%	42.0	3.0	130972.0	42.02	6169.0	1.0	0.0	1.0	0.5	0.0	1.0	1.0	0.0	0.0	1.0	0.0
75%	58.0	5.0	221014.25	44.89	8302.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0
max	74.0	7.0	299988.0	51.99	10993.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

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

Dataset Preview:

index	Age	FeverDays	Platelets	Hematocrit	WBC	Headache	EyePain	MusclePain	JointPain	Rash	Nausea	Vomiting	AbdominalPain	Bleeding	Lethargy	Dengue
0	61	6	56777	49.4957	3205	0	1	0	0	0	1	1	0	0	0	0
1	24	7	61250	42.4109	3738	1	0	0	0	1	0	0	1	1	0	0
2	70	7	88034	49.6614	3052	1	0	1	0	0	1	0	0	0	0	0
3	30	6	55130	50.2016	2713	1	0	0	1	0	0	1	1	0	0	0
4	33	3	64346	43.4698	4888	1	1	0	1	1	1	1	0	1	0	0

Code Block 13: Encoding Headache Column

Input

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

data.head()

Output

Dataset Preview:

index	Age	FeverDays	Platelets	Hematocrit	WBC	Headache	EyePain	MusclePain	JointPain	Rash	Nausea	Vomiting	AbdominalPain	Bleeding	Lethargy	Dengue
0	61	6	56777	49.4957	3205	1	1	0	0	0	1	1	0	0	0	0
1	24	7	61250	42.4109	3738	0	0	0	0	1	0	0	1	1	0	0
2	70	7	88034	49.6614	3052	0	0	1	0	0	1	0	0	0	0	0
3	30	6	55130	50.2016	2713	0	0	0	1	0	0	1	1	0	0	0
4	33	3	64346	43.4698	4888	0	1	0	1	1	1	1	0	1	0	0

Code Block 14: Encoding Rash Column

Input

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

print("Encoding complete. Checked head:")

data.head()

Output

Encoding complete. Checked head:
Dataset Preview:

index	Age	FeverDays	Platelets	Hematocrit	WBC	Headache	EyePain	MusclePain	JointPain	Rash	Nausea	Vomiting	AbdominalPain	Bleeding	Lethargy	Dengue
0	61	6	56777	49.4957	3205	1	1	0	0	1	1	1	0	0	0	0
1	24	7	61250	42.4109	3738	0	0	0	0	0	0	0	1	1	0	0
2	70	7	88034	49.6614	3052	0	0	1	0	1	1	0	0	0	0	0
3	30	6	55130	50.2016	2713	0	0	0	1	1	0	1	1	0	0	0
4	33	3	64346	43.4698	4888	0	1	0	1	0	1	1	0	1	0	0

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

Feature Matrix (X) Preview:

	Age	FeverDays	Platelets	Hematocrit	WBC	Headache	EyePain	MusclePain	JointPain	Rash	Nausea	Vomiting	AbdominalPain	Bleeding	Lethargy
0	61	6	56777	49.495729	3205	0	1	0	0	0	1	1	0	0	0
1	24	7	61250	42.410936	3738	1	0	0	0	1	0	0	1	1	0
2	70	7	88034	49.661431	3052	1	0	1	0	0	1	0	0	0	0
3	30	6	55130	50.201593	2713	1	0	0	1	0	0	1	1	0	0
4	33	3	64346	43.469804	4888	1	1	0	1	1	1	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	59	2	123791	47.188610	4239	1	0	0	0	0	0	0	0	0	0
4996	11	4	111575	41.845604	5989	1	0	1	1	1	1	1	0	0	0
4997	19	2	128198	40.284278	4775	1	1	1	0	0	0	0	0	0	1
4998	11	0	141619	47.393916	4625	1	1	0	0	1	1	1	1	0	1
4999	69	2	90619	46.471485	5982	1	0	0	0	1	0	1	0	0	0

5000 rows × 15 columns

Code Block 17: Display Target Vector (y)

Input

y.head()

Output

Target Vector (y) Preview:

	Dengue
0	0
1	0
2	0
3	0
4	0

Name: Dengue, dtype: int64

Block 18: Logistic Regression Model Training

Input

model = LogisticRegression(C=1.0, penalty='l2', solver='liblinear')

model.fit(X_train, y_train)

Output

```
Model trained successfully
        LogisticRegression
LogisticRegression(solver='liblinear')
```

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

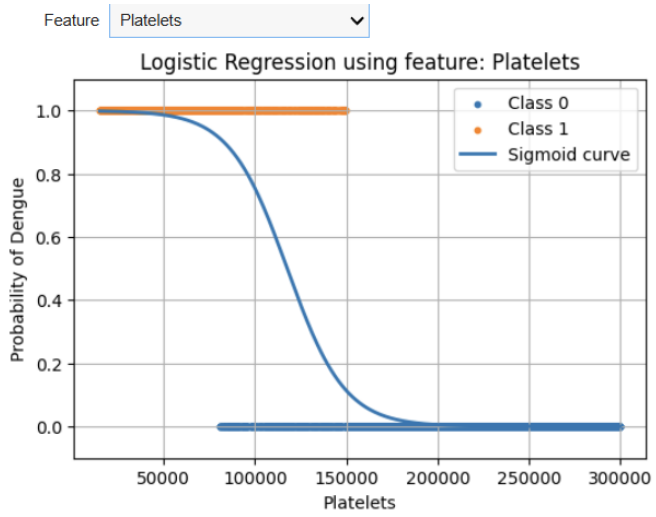
```
Feature-target data ready -> 15 numeric features selected for modeling
```

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

```
Y_train_pred = model.predict(X_train)

print(" Training Performance ")

print(f"Accuracy : {accuracy_score(y_train, Y_train_pred):.4f}")

print(f"Precision: {precision_score(y_train, Y_train_pred):.4f}")

print(f"Recall   : {recall_score(y_train, Y_train_pred):.4f}")

print(f"F1 Score : {f1_score(y_train, Y_train_pred):.4f}")
```

Output

Training Performance

Accuracy : 0.9732

Precision: 0.9712

Recall : 0.9707

F1 Score : 0.9709

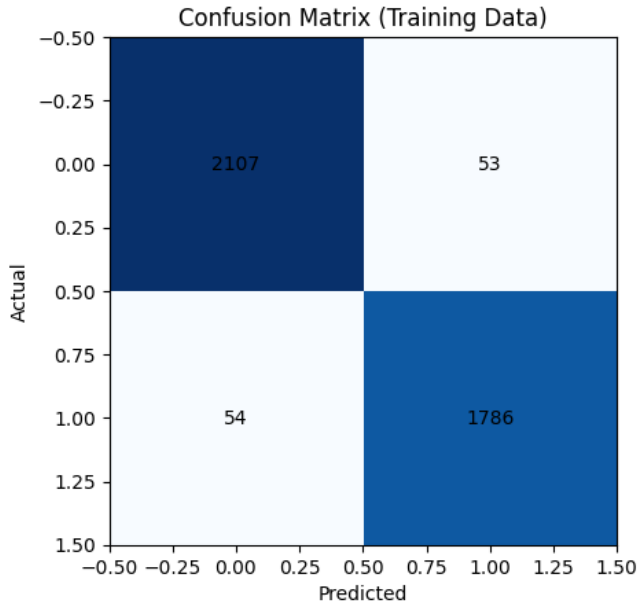
Block 22: Confusion Matrix (Training Data)

Input

```
cm_train = confusion_matrix(y_train, Y_train_pred)

sns.heatmap(cm_train, cmap="Blues")

plt.title("Confusion Matrix (Training Data)")

for i in range(len(cm_train)):
```

```
    for j in range(len(cm_train)):
        plt.text(j, i, cm_train[i][j], ha='center', va='center')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Output



Block 23: Model evaluation matrix

Input

```
Y_test_pred = model.predict(X_test)
print("Testing Performance")
print(f"Accuracy : {accuracy_score(y_test, Y_test_pred):.4f}")
print(f"Precision: {precision_score(y_test, Y_test_pred):.4f}")
print(f"Recall   : {recall_score(y_test, Y_test_pred):.4f}")
print(f"F1 Score : {f1_score(y_test, Y_test_pred):.4f}")
```

Output

```
Testing Performance
Accuracy : 0.9740
Precision: 0.9657
Recall : 0.9783
```

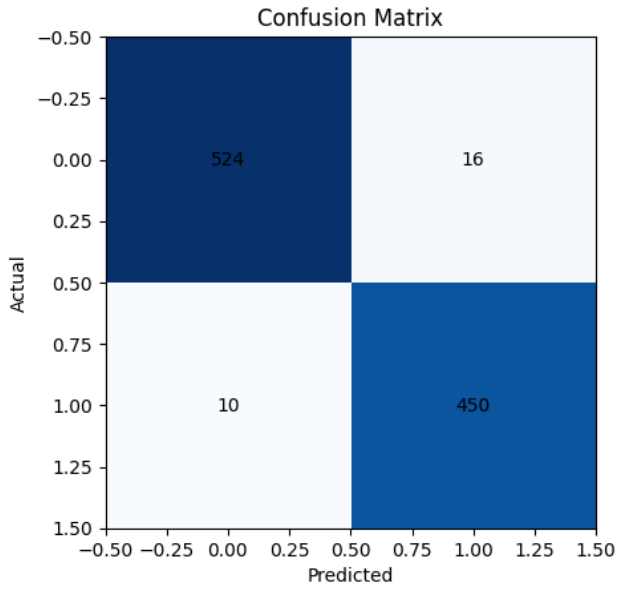
F1 Score : 0.9719

Block 24: Confusion Matrix (Testing Data)

Input

```
cm_test = confusion_matrix(y_test, Y_test_pred)
sns.heatmap(cm_test, annot=True, cmap="Blues", fmt='d')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix for Testing Data")
plt.show()
```

Output



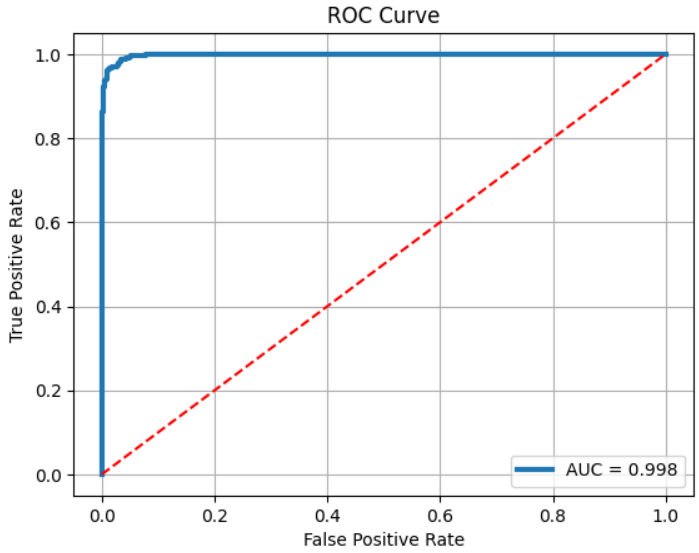
Block 25: ROC Curve

```
Y_proba = model.predict_proba(X_test)[:,1]
fpr, tpr, th = roc_curve(Y_test, Y_proba)
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, linewidth=3, 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

```
ROC metrics calculated. AUC = 0.998
```



Block 26: 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

Select a row to see prediction:

Index	Age	FeverDays	Hematocrit	WBC	Headache	EyePain	MusclePain	JointPain	Rash	Nausea	Vomiting	AbdominalPain	Bleeding	Lethargy
1434	69	3	44.363337	4203	1	1	1	0	0	0	1	1	0	0
1435	23	6	42.601771	2706	1	1	0	1	0	0	1	0	1	1
1436	11	7	48.035335	3961	0	0	1	0	1	0	0	1	0	1
1437	26	4	51.648016	4519	0	1	0	1	0	1	0	0	1	0
1438	37	3	42.108583	4666	1	1	0	0	0	1	0	0	1	1

```
------ 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
```