Task 4: **Classification with Logistic Regression**

Objective:

uild a **binary classifier** using **Logistic Regression** and evaluate it.

Tools Used:

- `Pandas` for data handling
- `Scikit-learn` for model building and evaluation
- `Matplotlib` & `Seaborn` for visualization

📝 Steps Followed:

🔹 1. Load a Binary Classification Dataset

We use the **Breast Cancer** dataset from `sklearn.datasets`.

```python
from sklearn.datasets import load_breast_cancer
import pandas as pd

data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)
```

🔹 2. Train-Test Split & Feature Scaling

python Copy Edit

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

🔹 3. Fit Logistic Regression Model

python Copy Edit

```python
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train_scaled, y_train)
```

🔹 4. Model Evaluation

python Copy Edit

```python
from sklearn.metrics import confusion_matrix, classification_report, roc_auc_score, roc_curve
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
# Predictions
y_pred = model.predict(X_test_scaled)
y_proba = model.predict_proba(X_test_scaled)[:, 1]

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Metrics
print(classification_report(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, y_proba))
```

🔹 5. ROC Curve & Threshold Tuning

python Copy Edit

```python
# ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_proba)

plt.plot(fpr, tpr, label='ROC Curve (area = %0.2f)' % roc_auc_score(y_test, y_proba))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()

# Threshold tuning example
custom_threshold = 0.4
y_custom = (y_proba >= custom_threshold).astype(int)
print(confusion_matrix(y_test, y_custom))
```

📘 Sigmoid Function Explanation:

The **sigmoid function** used in logistic regression maps any real value to a range between 0 and 1:

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

- Used to model the **probability** of class membership
- Logistic regression uses sigmoid to convert linear combinations into probabilities.

✅ Outcome:

- Built a binary classifier using logistic regression
- Evaluated performance with metrics like **precision**, **recall**, **ROC-AUC**
- Visualized ROC curve and **tuned threshold** for better control over classification.