Name: Annie Mahajan
Registration No.: 23BRS1121

Lab - 06
KNN, Logistic Regression

Sample 1

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 1.0
Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
```

Sample 2

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
iris = load_iris()
X = iris.data
y = iris.target
unique, counts = np.unique(y, return_counts=True)
print("Original Distribution:")
for u, c in zip(unique, counts):
    print(f"Class {u}: {c}")
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.3,
    random_state=42,
    stratify=y
)
print("\nTrain Distribution:")
unique, counts = np.unique(y_train, return_counts=True)
for u, c in zip(unique, counts):
    print(f"Class {u}: {c}")
print("\nTest Distribution:")
unique, counts = np.unique(y_test, return_counts=True)
for u, c in zip(unique, counts):
    print(f"Class {u}: {c}")
print("\nProportion Check (Train):", np.bincount(y_train) / len(y_train))
print("Proportion Check (Test):", np.bincount(y_test) / len(y_test))
```

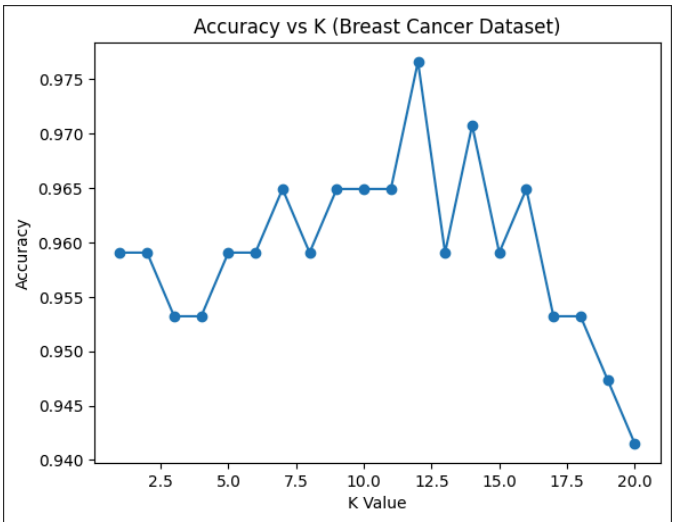
```
Original Distribution:
Class 0: 50
Class 1: 50
Class 2: 50

Train Distribution:
Class 0: 35
Class 1: 35
Class 2: 35

Test Distribution:
Class 0: 15
Class 1: 15
Class 2: 15

Proportion Check (Train): [0.33333333 0.33333333 0.33333333]
Proportion Check (Test): [0.33333333 0.33333333 0.33333333]
```

Sample 3

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
data = load_breast_cancer()
X = data.data
y = data.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
```

```
)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
k_values = range(1, 21)
accuracy = []
for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    accuracy.append(acc)
for k, acc in zip(k_values, accuracy):
    print(f"K = {k}, Accuracy = {acc:.4f}")
plt.plot(k_values, accuracy, marker='o')
plt.xlabel("K Value")
plt.ylabel("Accuracy")
plt.title("Accuracy vs K (Breast Cancer Dataset)")
plt.show()
```

```
K = 1, Accuracy = 0.9591
K = 2, Accuracy = 0.9591
K = 3, Accuracy = 0.9532
K = 4, Accuracy = 0.9532
K = 5, Accuracy = 0.9591
K = 6, Accuracy = 0.9591
K = 7, Accuracy = 0.9649
K = 8, Accuracy = 0.9591
K = 9, Accuracy = 0.9649
K = 10, Accuracy = 0.9649
K = 11, Accuracy = 0.9649
K = 12, Accuracy = 0.9766
K = 13, Accuracy = 0.9591
K = 14, Accuracy = 0.9708
K = 15, Accuracy = 0.9591
K = 16, Accuracy = 0.9649
K = 17, Accuracy = 0.9532
K = 18, Accuracy = 0.9532
K = 19, Accuracy = 0.9474
K = 20, Accuracy = 0.9415
```



Sample 4

```
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
iris = load_iris()
X = iris.data
y = iris.target
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(
   X_scaled, y, test_size=0.3, random_state=42
)
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
print("Accuracy after scaling:",
     accuracy_score(y_test, knn.predict(X_test)))
```

```
•••  Accuracy after scaling: 1.0
```

Sample 5

```
from sklearn.neighbors import KNeighborsRegressor
import numpy as np
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([2, 4, 6, 8, 10])
knn_reg = KNeighborsRegressor(n_neighbors=2)
knn_reg.fit(X, y)
print("Predicted value for 3.5:",
     knn_reg.predict([[3.5]]))
```

```
•••  Predicted value for 3.5: [7.]
```

Sample 6

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
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, confusion_matrix, classification_report
data = load_breast_cancer()
X = data.data
y = data.target
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.3, random_state=42
)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
model = LogisticRegression()
```

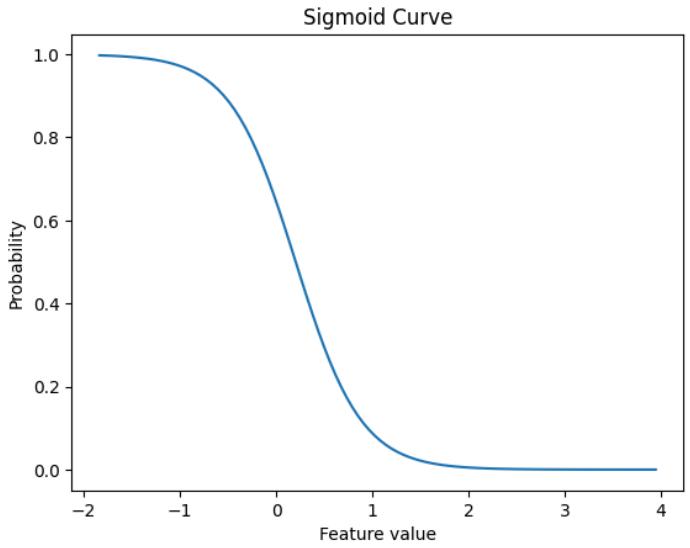
```
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)
y_prob = model.predict_proba(X_test_scaled)[:, 1]
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9824561403508771

Confusion Matrix:
 [[ 62   1]
 [  2 106]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.98      0.98        63
           1       0.99      0.98      0.99       108

    accuracy                           0.98       171
   macro avg       0.98      0.98      0.98       171
weighted avg       0.98      0.98      0.98       171
```

Sample 7

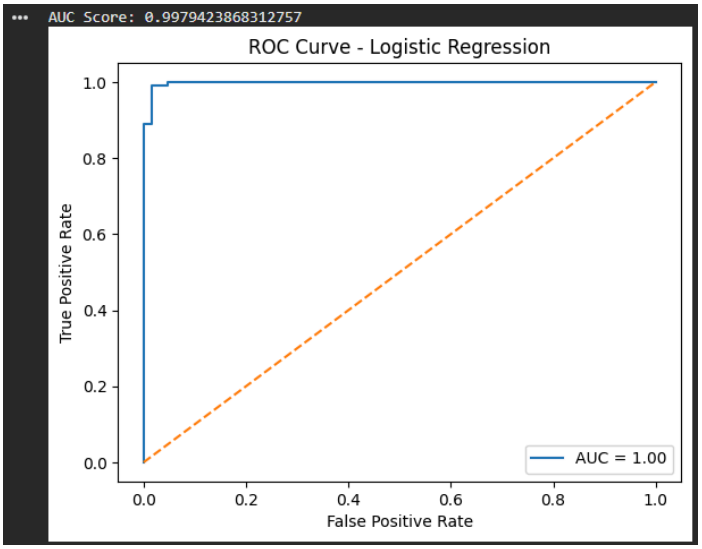
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
X = data.data[:, 0].reshape(-1, 1)
y = data.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
model = LogisticRegression()
model.fit(X_train, y_train)
x_values = np.linspace(X_train.min(), X_train.max(), 100).reshape(-1, 1)
y_values = model.predict_proba(x_values)[:, 1]
plt.plot(x_values, y_values)
plt.xlabel("Feature value")
plt.ylabel("Probability")
plt.title("Sigmoid Curve")
```

plt.show()



Sample 8

```
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc_score = roc_auc_score(y_test, y_prob)
print("AUC Score:", auc_score)
plt.plot(fpr, tpr, label=f"AUC = {auc_score:.2f}")
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - Logistic Regression")
plt.legend()
plt.show()
```



Exercise 1

Part A

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
data = pd.read_csv("diabetes.csv")
X = data.drop("Outcome", axis=1)
y = data["Outcome"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.8571428571428571
Confusion Matrix:
 [[ 98  18]
 [ 15 100]]
```

Part B

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
data = pd.read_csv("breast_cancer.csv")
X = data.drop("diagnosis", axis=1)
y = data["diagnosis"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
knn = KNeighborsClassifier(n_neighbors=5)
```

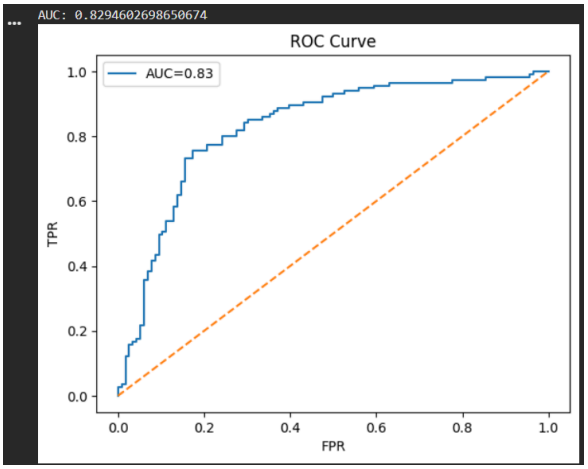
```
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
•••  Accuracy: 0.9590643274853801
```

Exercise 2

Part A

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
data = pd.read_csv("diabetes.csv")
X = data.drop("Outcome", axis=1)
y = data["Outcome"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
y_prob = model.predict_proba(X_test)[:,1]
fpr, tpr, _ = roc_curve(y_test, y_prob)
auc = roc_auc_score(y_test, y_prob)
print("AUC:", auc)
plt.plot(fpr, tpr, label=f"AUC={auc:.2f}")
plt.plot([0,1],[0,1],'--')
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC Curve")
plt.legend()
plt.show()
```



Part B

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
iris = load_iris()
X = iris.data
y = iris.target
y = (y == 0).astype(int)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
model = LogisticRegression()
model.fit(X_train, y_train)
y_prob = model.predict_proba(X_test)[:,1]
fpr, tpr, _ = roc_curve(y_test, y_prob)
auc = roc_auc_score(y_test, y_prob)
print("AUC:", auc)
plt.plot(fpr, tpr, label=f"AUC={auc:.2f}")
plt.plot([0,1],[0,1],'--')
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC Curve - Iris")
plt.legend()
plt.show()
```

```
AUC: 1.0
```