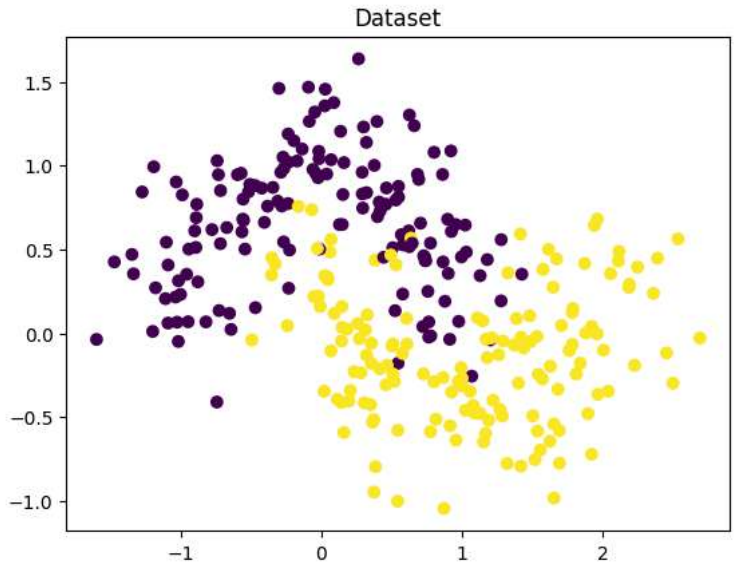
Name : Royan Dsouza

Date : 4 February

Roll NO : 21

Experiment 4 : Classification Using Logistic Regression, Decision Tree, and k-Nearest Neighbors

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix

#Create 2D dataset
X, y = make_moons(n_samples=300, noise=0.25, random_state=42)

plt.scatter(X[:,0],X[:,1], c=y)
plt.title("Dataset")
plt.show()
```




```
#Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=0
)
```

```
#Scaling (needed for LR and KNN)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
#1.Logistic Regression
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()

lr.fit(X_train_scaled, y_train)

y_pred_lr = lr.predict(X_test_scaled)

print("Logistic Regression")
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_lr))
```

```
Logistic Regression
Accuracy: 0.8777777777777778
Confusion Matrix:
 [[34  5]
 [ 6 45]]
```

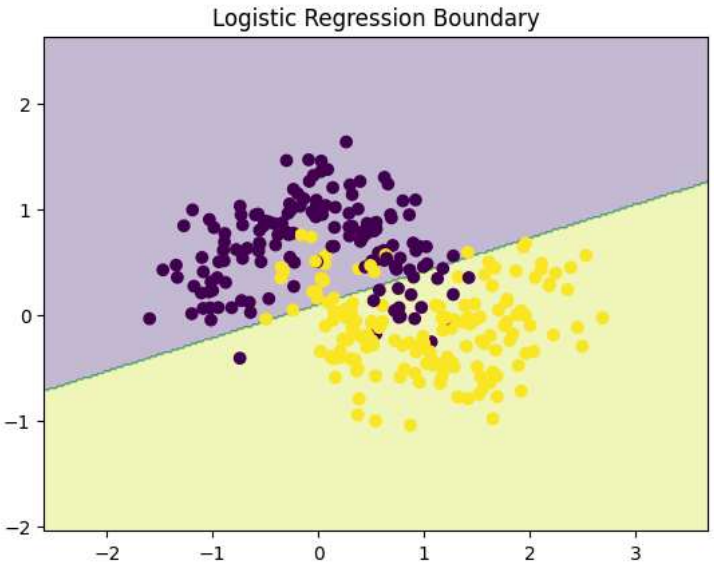
```
#2.Decison Tree
from sklearn.tree import DecisionTreeClassifier

dt =  DecisionTreeClassifier(max_depth=4)

dt.fit(X_train, y_train) #no scaling required

y_pred_dt = dt.predict(X_test)

print("Decision Tree")
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
```

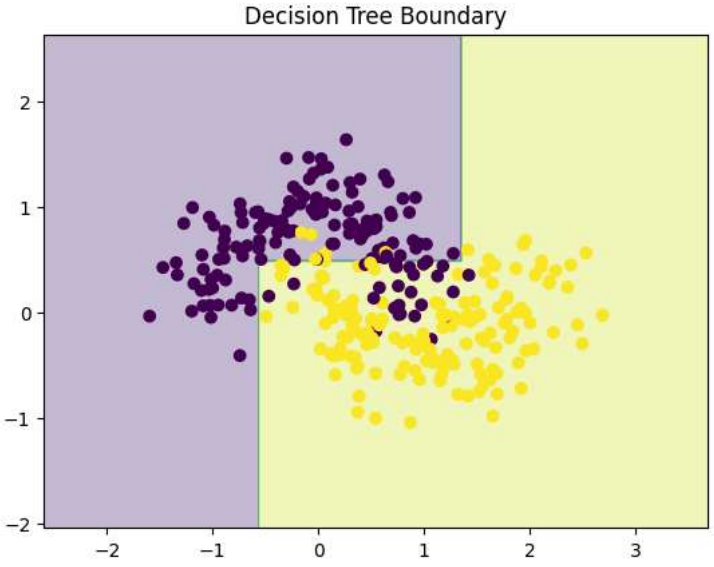
```
Decision Tree
Accuracy: 0.8555555555555555
Confusion Matrix:
 [[30  9]
 [ 4 47]]
```

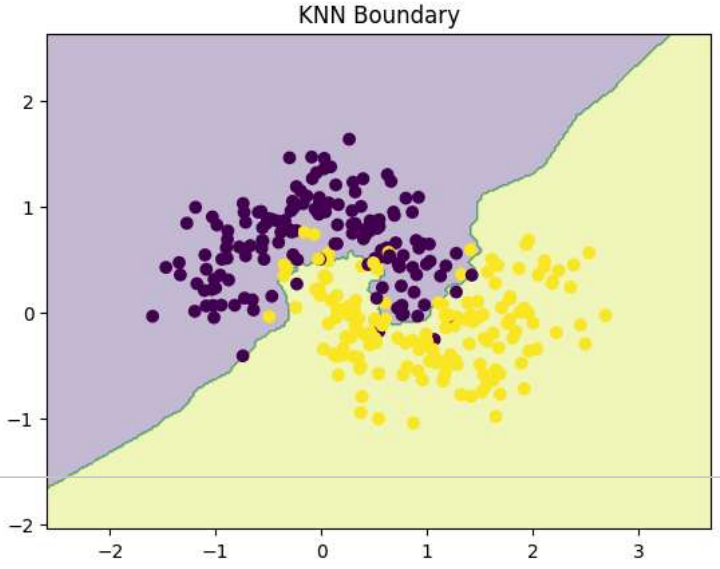
```
#3. K-Nearest Neighbor
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train_scaled, y_train)

y_pred_knn = knn.predict(X_test_scaled)

print("KNN")
print("Accuracy:", accuracy_score(y_test, y_pred_knn))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_knn))
```

```
KNN
Accuracy: 0.9111111111111111
Confusion Matrix:
 [[38  1]
 [ 7 44]]
```

```
# plotting decision boundaries function
def plot_decision_boundaries(model, scaled, title):
    h = 0.02
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1

    xx, yy = np.meshgrid(
        np.arange(x_min, x_max, h),
        np.arange(y_min, y_max, h)
    )

    grid = np.c_[xx.ravel(), yy.ravel()]

    if scaled:
        grid = scaler.transform(grid)

    Z = model.predict(grid)
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.3)
    plt.scatter(X[:, 0], X[:, 1], c=y)
    plt.title(title)
    plt.show()

plot_decision_boundaries(lr, scaled=True, title="Logistic Regression Boundary")
plot_decision_boundaries(dt, scaled=False, title="Decision Tree Boundary")
plot_decision_boundaries(knn, scaled=True, title="KNN Boundary")
```


Logistic Regression Boundary
2
1
0
−1
−2
−2
−1
0
1
2
3


Decision Tree Boundary
2
1
0
−1
−2
−2
−1
0
1
2
3


KNN Boundary
2
1
0
−1
−2
−2
−1
0
1
2
3