Name : Prajwal Kola

Roll No: 33

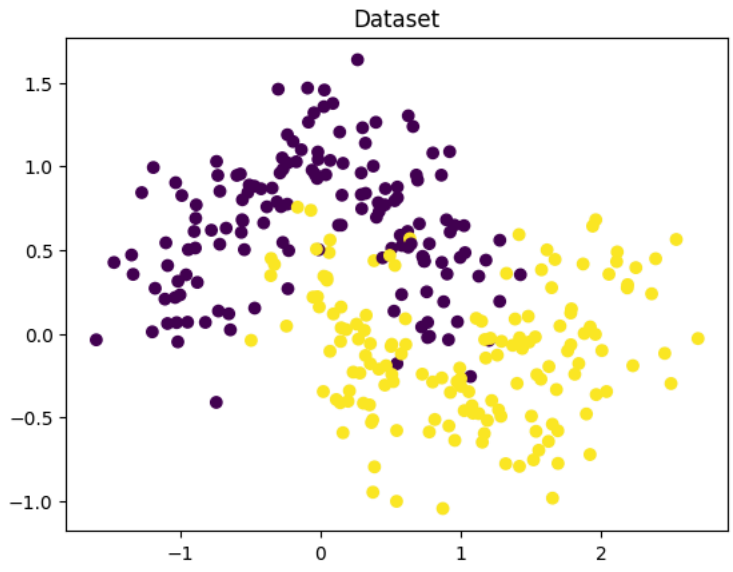
Expt.-4: Classification Using Logistic Regression, Decision Tree, and k-Nearest Neighbors

```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\hp\valenka\lib\site-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in c:\users\hp\valenka\lib\site-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in c:\users\hp\valenka\lib\site-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\hp\valenka\lib\site-packages (from scikit-learn) (1.5.3)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\hp\valenka\lib\site-packages (from scikit-learn) (3.6.0)

[notice] A new release of pip is available: 25.3 -> 26.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
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
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
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()
```




```
x_train , x_test , y_train , y_test = train_test_split(
    x,y, test_size=0.3 , random_state=0
)
```

```
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

Logistic Regression

```
lr = LogisticRegression()
lr.fit(x_train_scaled,y_train)
y_pred_lr = lr.predict(x_test_scaled)
print("Logisitic Regression")
print("Accuracy:",accuracy_score(y_test,y_pred_lr))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred_lr))
```

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

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
lr = LogisticRegression()
lr.fit(x_train,y_train)
y_pred_lr = lr.predict(x_test)
print("Logisitic Regression")
print("Accuracy:",accuracy_score(y_test,y_pred_lr))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred_lr))
```

```
Logisitic Regression
Accuracy: 0.8666666666666667
Confusion Matrix:
 [[34  5]
 [ 7 44]]
```

Decision Tree

```
dt=DecisionTreeClassifier(max_depth=4)
dt.fit(x_train_scaled,y_train)
y_pred_dt = dt.predict(x_test_scaled)
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]]
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
dt=DecisionTreeClassifier(max_depth=4)
dt.fit(x_train,y_train)
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]]
```

K Nearest Neighbors

```
k=KNeighborsClassifier(n_neighbors=5)
k.fit(x_train_scaled,y_train)
y_pred_k = k.predict(x_test_scaled)
print("K Neighbors Classifier")
print("Accuracy:",accuracy_score(y_test,y_pred_k))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred_k))
```

```
K Neighbors Classifier
Accuracy: 0.9111111111111111
Confusion Matrix:
 [[38  1]
 [ 7 44]]
```

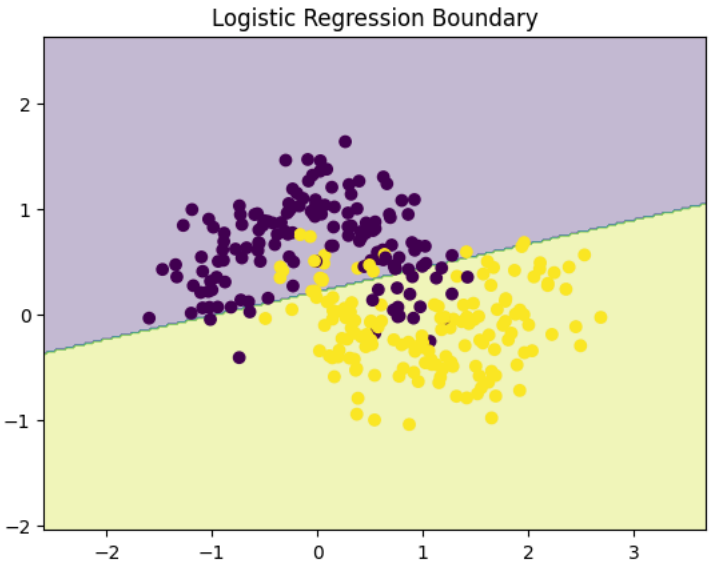
Unsupported Cell Type. Double-Click to inspect/edit the content.

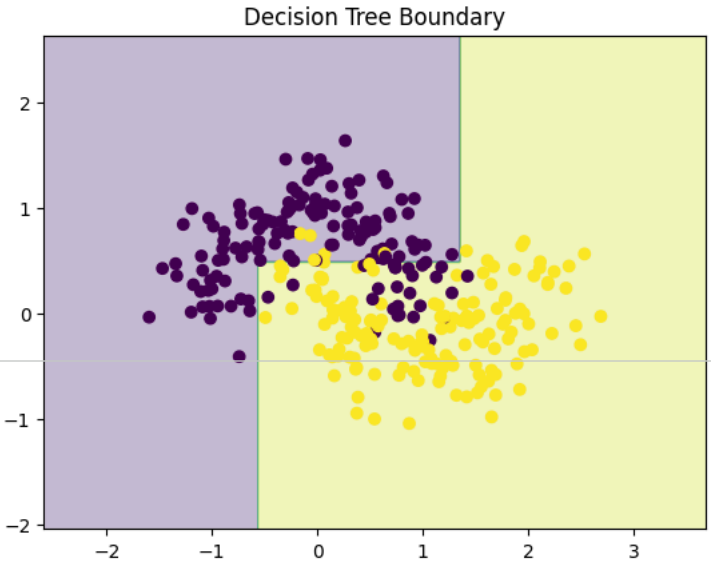
```
k=KNeighborsClassifier(n_neighbors=3)
k.fit(x_train,y_train)
y_pred_k = k.predict(x_test)
print("K Neighbors Classifier")
print("Accuracy:",accuracy_score(y_test,y_pred_k))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred_k))
```

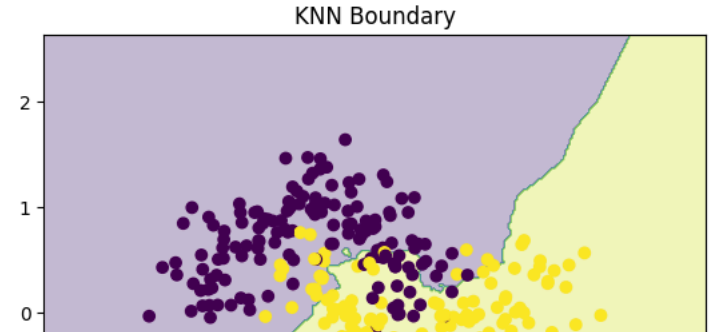
```
K Neighbors Classifier
Accuracy: 0.9333333333333333
Confusion Matrix:
 [[38  1]
 [ 5 46]]
```

Plot Decision Boundaries

```
def plot_boundary(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_boundary(lr,True,"Logistic Regression Boundary")
plot_boundary(dt,False,"Decision Tree Boundary")
plot_boundary(k,True,"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