# EX.NO: 8 A PYTHON PROGRAM TO

# DATE : 18.10.2024 IMPLEMENT ADA BOOSTING

**AIM:**

To write a python program to implement ADA Boosting.

# PROCEDURE:

Implementing ADA Boosting using the dataset involve the following steps:

**Step1:Import Necessary Libraries**

First, import the libraries that are essential for data manipulation,visualization,and model building.

import numpy as np importpandasaspd

from sklearn.tree import DecisionTreeClassifier

frommlxtend.plottingimportplot\_decision\_regions import seaborn as sns

from sklearn.metrics import accuracy\_score

**Step2:Load and prepare data**

**sns.scatterplot(x=df['X1'], y=df['X2'], hue=df['label'])**

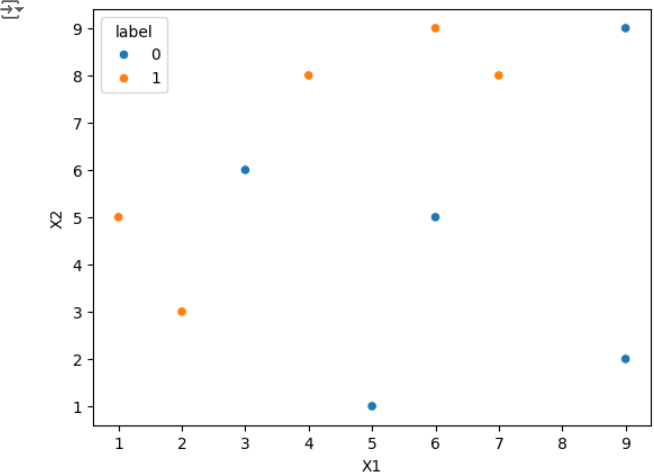
**df['weights'] = 1 / df.shape[0]**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **df = pd.DataFrame()** | | | | | | | | |
| **df['X1'] = [1, 2,** | **3,** | **4,** | **5,** | **6,** | **6,** | **7,** | **9,** | **9]** |
| **df['X2'] = [5, 3,** | **6,** | **8,** | **1,** | **9,** | **5,** | **8,** | **9,** | **2]** |
| **df['label'] = [1,** | **1,** | **0,** | **1,** | **0,** | **1,** | **0,** | **1,** | **0, 0]** |

x = df.iloc[:, 0:2].values

y = df.iloc[:, 2].values

# OUTPUT:



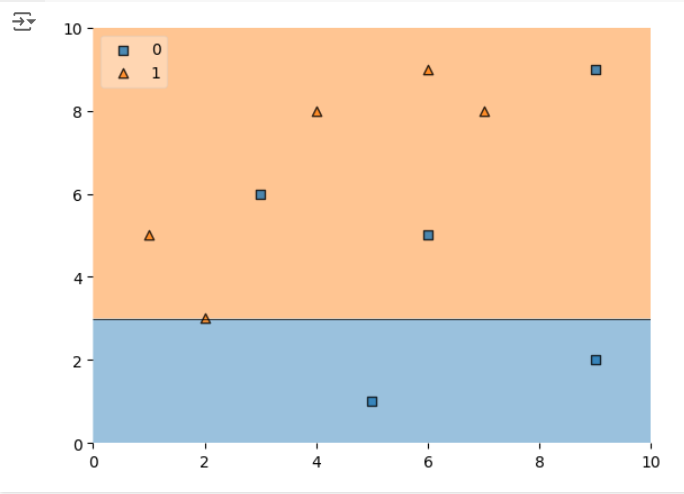
**Step3:Trainthe1stmodel**

# Step 2: Train 1st Model

dt1=DecisionTreeClassifier(max\_depth=1) dt1.fit(x, y)

plot\_decision\_regions(x,y,clf=dt1,legend=2) df['y\_pred'] = dt1.predict(x)

# OUTPUT:



**Step4:Calculate model weight**

# Step 4: Update Weights

defupdate\_row\_weights(row,alpha=0.423): if row['label'] == row['y\_pred']:

returnrow['weights']\*np.exp(-alpha) else:

return row['weights'] \* np.exp(alpha)

df['updated\_weights']= df.apply(update\_row\_weights, axis=1)

df['normalized\_weights']=df['updated\_weights']/ df['updated\_weights'].sum()

df['cumsum\_upper'] = np.cumsum(df['normalized\_weights'])

df['cumsum\_lower'] = df['cumsum\_upper'] - df['normalized\_weights']

**Step5:Create new dataset**

# Step 5: Create New Dataset

defcreate\_new\_dataset(df): indices = []

foriinrange(df.shape[0]): a = np.random.random()

for index, row in df.iterrows():

ifrow['cumsum\_upper']>aanda>row['cumsum\_lower']: indices.append(index)

return indices

index\_values = create\_new\_dataset(df)

second\_df = df.iloc[index\_values, [0, 1, 2, 3]]

**Step6:Train 2nd model**

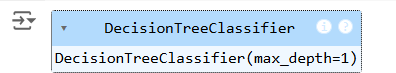
# Step 6: Train 2nd Model

dt2=DecisionTreeClassifier(max\_depth=1) x = second\_df.iloc[:, 0:2].values

y = second\_df.iloc[:, 2].values

dt2.fit(x, y)

# OUTPUT:



**Step7:Plot decision tree and calculate model weights for 2nd model**

# Plot the decision tree for the second model plot\_decision\_regions(x,y,clf=dt2,legend=2) second\_df['y\_pred'] = dt2.predict(x)

#Step7:CalculateModelWeightfor2ndModel alpha2 = calculate\_model\_weight(0.1)

print(f"Alpha2: {alpha2}")

**Step8:update weights for 2nd model**

# Step 8: Update Weights for 2nd Model

defupdate\_row\_weights(row,alpha=1.09): if row['label'] == row['y\_pred']:

returnrow['weights']\*np.exp(-alpha) else:

return row['weights'] \* np.exp(alpha)

second\_df['updated\_weights']= second\_df.apply(update\_row\_weights, axis=1)

second\_df['nomalized\_weights']=second\_df['updated\_weights']/ second\_df['updated\_weights'].sum()

second\_df['cumsum\_upper'] = np.cumsum(second\_df['nomalized\_weights'])

second\_df['cumsum\_lower']=second\_df['cumsum\_upper']- second\_df['nomalized\_weights']

**Step9:Calculate alpha for 3rd model**

#Step9:CalculateAlphafor3rdModel alpha3 = calculate\_model\_weight(0.7)

print(f"Alpha3: {alpha3}")

# Step 10: Accuracy Calculation y\_true = second\_df['label'].values y\_pred=second\_df['y\_pred'].values

#CalculateaccuracyfortheAdaBoostmodel accuracy = accuracy\_score(y\_true, y\_pred)

print(f"Accuracy of the AdaBoost model: {accuracy:.4f}")

# OUTPUT:

**ALPHA3:-0.4236489301936017**

**Accuracy of the Ada Boosting model:0.80000**

# RESULT:

Thus the python program to implement Ada boosting has been executed successfully and the results have been verified.