Ex : No: 08 a
date :- 25/02/25

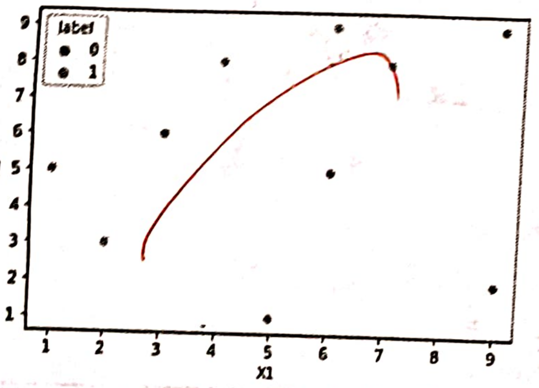
A Python Program to Implement Ada Boosting

Aim:-
To implement a python program for Ada Boosting.

Algorithm :-
Step 1:- Import Necessary Libraries
Import numpy as np
Step 2:- Load and prepare Data
Step 3: Initialize parameters
Step 4: Train weak classifiers
Step 5: Make predictions
Step 6: Evaluate the model

Program:-
```
import ~~pandas~~ numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
#Step2: Load and Prepare data
df = pd.DataFrame({
    'x1':[2,3,4,5,6,6,7,9,9],
    'x2':[5,3,6,8,1,9,5,8,9,2]
 'label':[1,1,0,1,0,1,01,0,0]
})
X = df[['x1','x2']].values
Y = df['label'].values
```

	X1	X2	label	weights
6	6	5	0	0.1
6	6	5	0	0.1
0	1	5	1	0.1
6	6	5	0	0.1
7	7	8	1	0.1
5	6	9	1	0.1
1	2	3	1	0.1
8	9	9	0	0.1
4	5	1	0	0.1
6	6	5	0	0.1



	X1	X2	label	weights
1	2	3	1	0.1
6	6	5	0	0.1
5	6	9	1	0.1
1	2	3	1	0.1
5	6	9	1	0.1
8	9	9	0	0.1
8	9	9	0	0.1
8	9	9	0	0.1
5	6	9	1	0.1
8	9	9	0	0.1

	X1	X2	label	weights	y_pred
6	6	5	0	0.1	0
6	6	5	0	0.1	0
0	1	5	1	0.1	1
6	6	5	0	0.1	0
7	7	8	1	0.1	0
5	6	9	1	0.1	0
1	2	3	1	0.1	1
8	9	9	0	0.1	0
4	5	1	0	0.1	0
6	6	5	0	0.1	0

	X1	X2	label	weights	y_pred	updated_weights
6	6	5	0	0.1	0	0.033622
6	6	5	0	0.1	0	0.033622
0	1	5	1	0.1	1	0.033622
6	6	5	0	0.1	0	0.033622
7	7	8	1	0.1	0	0.297427
5	6	9	1	0.1	0	0.297427
1	2	3	1	0.1	1	0.033622
8	9	9	0	0.1	0	0.033622
4	5	1	0	0.1	0	0.033622
6	6	5	0	0.1	0	0.033622

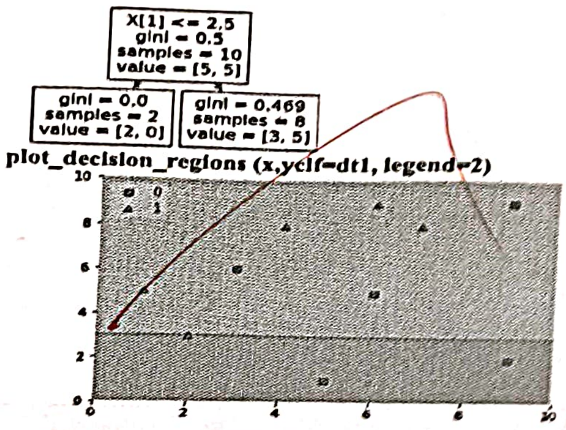
	X1	X2	label	weights	y_pred	normalized_weights	cumsum_lower	cumsum_upper
6	6	5	0	0.1	0	0.038922	0.000000	0.038922
6	6	5	0	0.1	0	0.038922	0.038922	0.077843
0	1	5	1	0.1	1	0.038922	0.077843	0.116765
6	6	5	0	0.1	0	0.038922	0.116765	0.155687
7	7	8	1	0.1	0	0.344313	0.155687	0.500000
5	6	9	1	0.1	0	0.344313	0.500000	0.844313
1	2	3	1	0.1	1	0.038922	0.844313	0.883235
8	9	9	0	0.1	0	0.038922	0.883235	0.922157
4	5	1	0	0.1	0	0.038922	0.922157	0.961078
6	6	5	0	0.1	0	0.038922	0.961078	1.000000

```
# Convert y into {-1,+1} form for AdaBoost theor
y_transformed = np.where(y==1, 1, -1)
x_transformed test_size=0.3, random_state=42)

# Step3: Intialise parameters
n_estimators = 5
n_samples = x_train.shape[0]
weights = np.ones(n_samples) / n_samples
classifiers = []
alphas = []
# step 4: Train weak classifiers
for _ in range(n_estimators):
    clf = DecisionTreeClassifier(max_depth=1)
    clf.fit(x_train, y_train sample_weight=weights)
    y_pred = clf.predict(x_train)

# Calculate weighted error
err = np.sum(weights*(y_pred!=y_train))/
np.sum(weights)
if err==0:
 err = 1e-10
# Compute alpha
alpha = 0.5* np.log((1-err)/err)
# Update weights
weights = weights * np.exp(-alpha * y_train* y_pred)
weight /= np.sum(weights) # namalise
classifiers.append(clf)
alphas.append(alpha)
# Step 5: Make prediction
def predict(x):
  final_score = np.zeros(x.shape[0])
  for clf, alpha in zip(classifiers, alphas):
```

```
[Text(0.5, 0.75, 'X[0] <= 3.5\ngini = 0.48\nsamples = 10\nvalue = [4, 6]'),
 Text(0.25, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [[illegible]'),
 Text(0.75, 0.25, 'gini = 0.375\nsamples = 8\nvalue = [0, 2]')]
```



```
            preds = clf.predict(x)
            final_score += alpha * preds
        return np.sign(final_score)
y_pred_test = predict(x_test)

# Step 6: Evaluate Model
acc = accuracy_score(y_test, y_preds_test)
# Step 7: Output Results
print("Final Accuracy on Test Set:", acc)
print("Classifier weights (alphas):", alphas).
```

Result:-

Thus the python program to implement Adaboosting has been successfully executed.