```
# ✅ A PYTHON PROGRAM TO IMPLEMENT ADA BOOSTING
# -----------------------------------------------------

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, plot_tree
from mlxtend.plotting import plot_decision_regions

# Step-1: Create dataset
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]

display(df)

# Plot scatter
sns.scatterplot(x=df['X1'], y=df['X2'], hue=df['label'])
plt.title("Data points")
plt.show()

# Initialize equal weights
df['weights'] = 1/df.shape[0]
display(df)

# Function to compute model weight (alpha)
def calculate_model_weight(error):
    return 0.5*np.log((1-error)/(error))
```

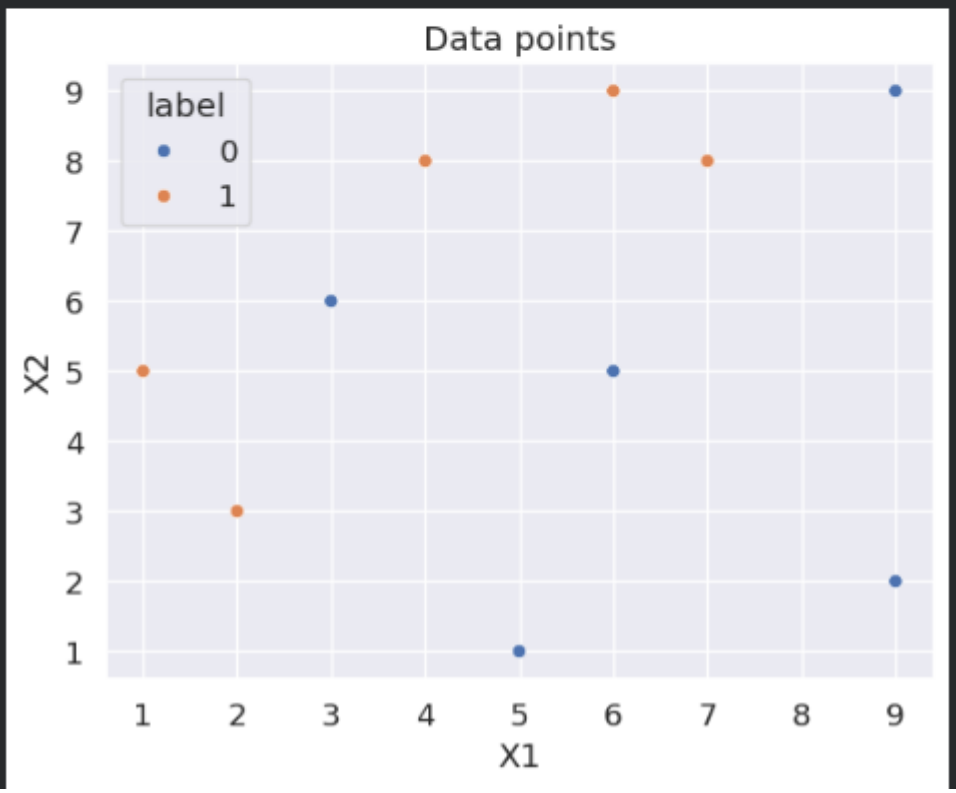
```
# Function to update row weights
def update_row_weights(row, alpha):
  if row['label'] == row['y_pred']:
    return row['weights']*np.exp(-alpha)
  else:
    return row['weights']*np.exp(alpha)

# ✅ Model-1
dt1 = DecisionTreeClassifier(max_depth=1)
x = df[['X1','X2']].values
y = df['label'].values
dt1.fit(x,y)
plot_tree(dt1)
plt.title("Decision Tree 1")
plt.show()

plot_decision_regions(x, y, clf=dt1, legend=2)
plt.title("Model-1 Decision Boundary")
plt.show()

df['y_pred'] = dt1.predict(x)
display(df)

# Error (given in question = 0.3)
alpha1 = calculate_model_weight(0.3)
df["updated_weights"] = df.apply(lambda r: update_row_weights(r,alpha1), 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']
```

```
display(df[['X1','X2','label','weights','y_pred','updated_weights','cumsum_lower','cumsum_upper']])


# Function to create new training sample
def create_new_dataset(df):
    indices = []
    for _ in range(df.shape[0]):
        a = np.random.random()
        for idx,row in df.iterrows():
            if (a > row['cumsum_lower']) and (a < row['cumsum_upper']):
                indices.append(idx)
                break
    return indices

# ✅ Second dataset
index_values = create_new_dataset(df)
second_df = df.iloc[index_values, [0,1,2,3]]
display(second_df)


# ✅ Model-2
dt2 = DecisionTreeClassifier(max_depth=1)
x2 = second_df[['X1','X2']].values
y2 = second_df['label'].values
dt2.fit(x2,y2)

plot_tree(dt2)
plt.title("Decision Tree 2")
plt.show()

plot_decision_regions(x2, y2, clf=dt2, legend=2)
plt.title("Model-2 Decision Boundary")
plt.show()
```

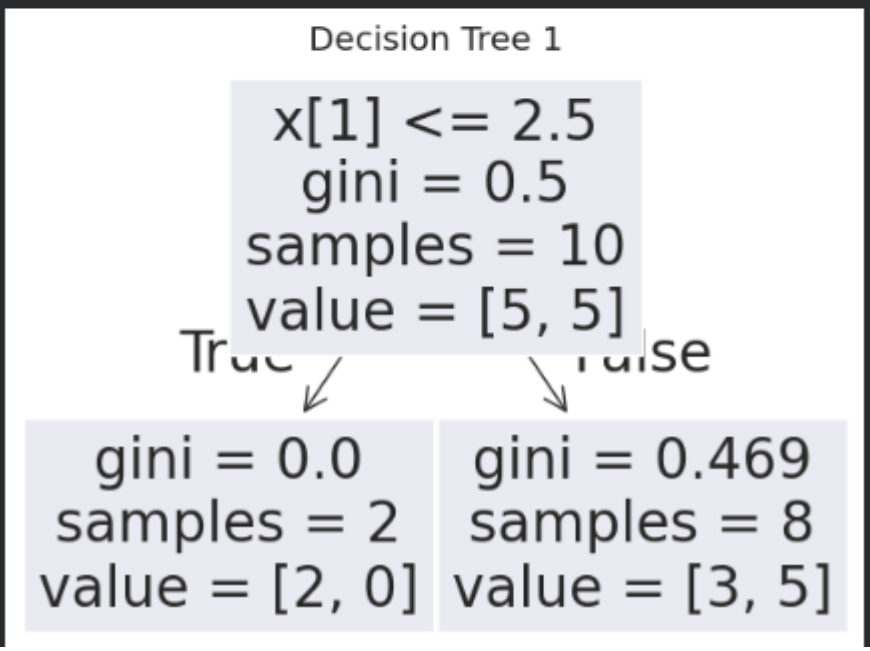
```
second_df['y_pred'] = dt2.predict(x2)

alpha2 = calculate_model_weight(0.1)  # given

display(second_df)

print("alpha2 =", alpha2)


second_df['updated_weights'] = second_df.apply(lambda r:
update_row_weights(r,alpha2),axis=1)

second_df['normalized_weights'] =
second_df['updated_weights']/second_df['updated_weights'].sum()


second_df['cumsum_upper'] = np.cumsum(second_df['normalized_weights'])

second_df['cumsum_lower'] = second_df['cumsum_upper'] - second_df['normalized_weights']


display(second_df)


# ✅ Third dataset

index_values = create_new_dataset(second_df)

third_df = second_df.iloc[index_values, [0,1,2,3]]

display(third_df)


# ✅ Model-3

dt3 = DecisionTreeClassifier(max_depth=1)

x3 = third_df[['X1','X2']].values

y3 = third_df['label'].values

dt3.fit(x3,y3)


plot_decision_regions(x3, y3, clf=dt3, legend=2)

plt.title("Model-3 Decision Boundary")

plt.show()
```

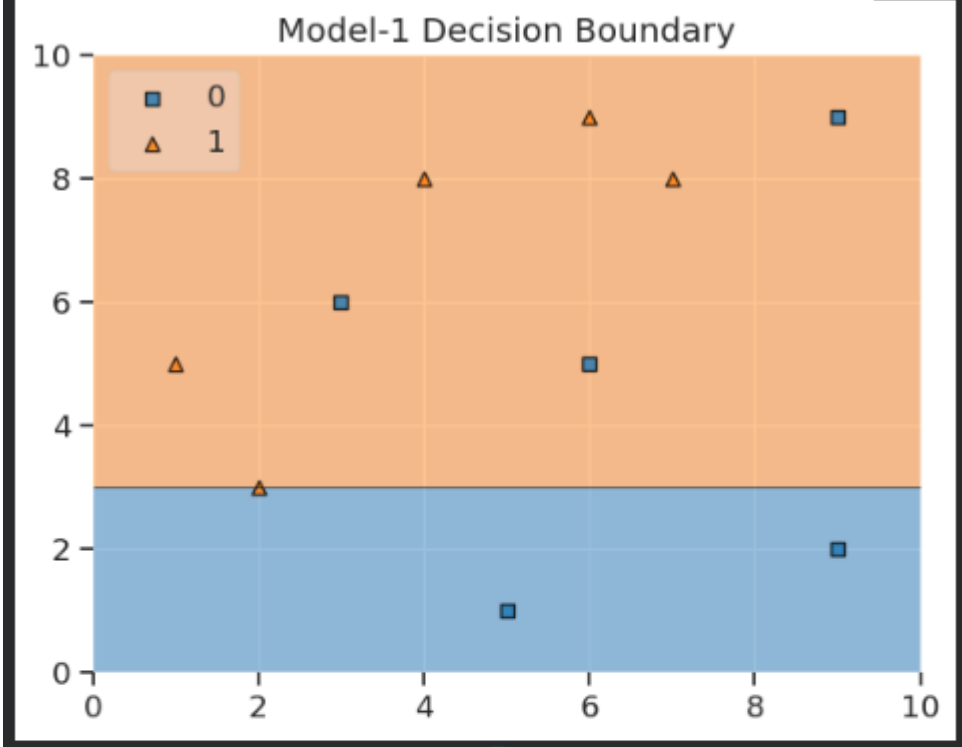
```
third_df['y_pred'] = dt3.predict(x3)
display(third_df)

alpha3 = calculate_model_weight(0.7)   # given
print("\nalpha values:", alpha1, alpha2, alpha3)

# ✅ Final prediction for a query
query1 = np.array([1,5]).reshape(1,2)
print("\nQuery = [1,5]")
print("DT1:", dt1.predict(query1))
print("DT2:", dt2.predict(query1))
print("DT3:", dt3.predict(query1))
print("Final Score:", alpha1*(1) + alpha2*(1) + alpha3*(1))
print("Final Prediction:", np.sign(alpha1*(1) + alpha2*(1) + alpha3*(1)))

query2 = np.array([9,9]).reshape(1,2)
print("\nQuery = [9,9]")
print("DT1:", dt1.predict(query2))
print("DT2:", dt2.predict(query2))
print("DT3:", dt3.predict(query2))
print("Final Score:", alpha1*(1) + alpha2*(-1) + alpha3*(-1))
print("Final Prediction:", np.sign(alpha1*(1) + alpha2*(-1) + alpha3*(-1)))
```

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



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



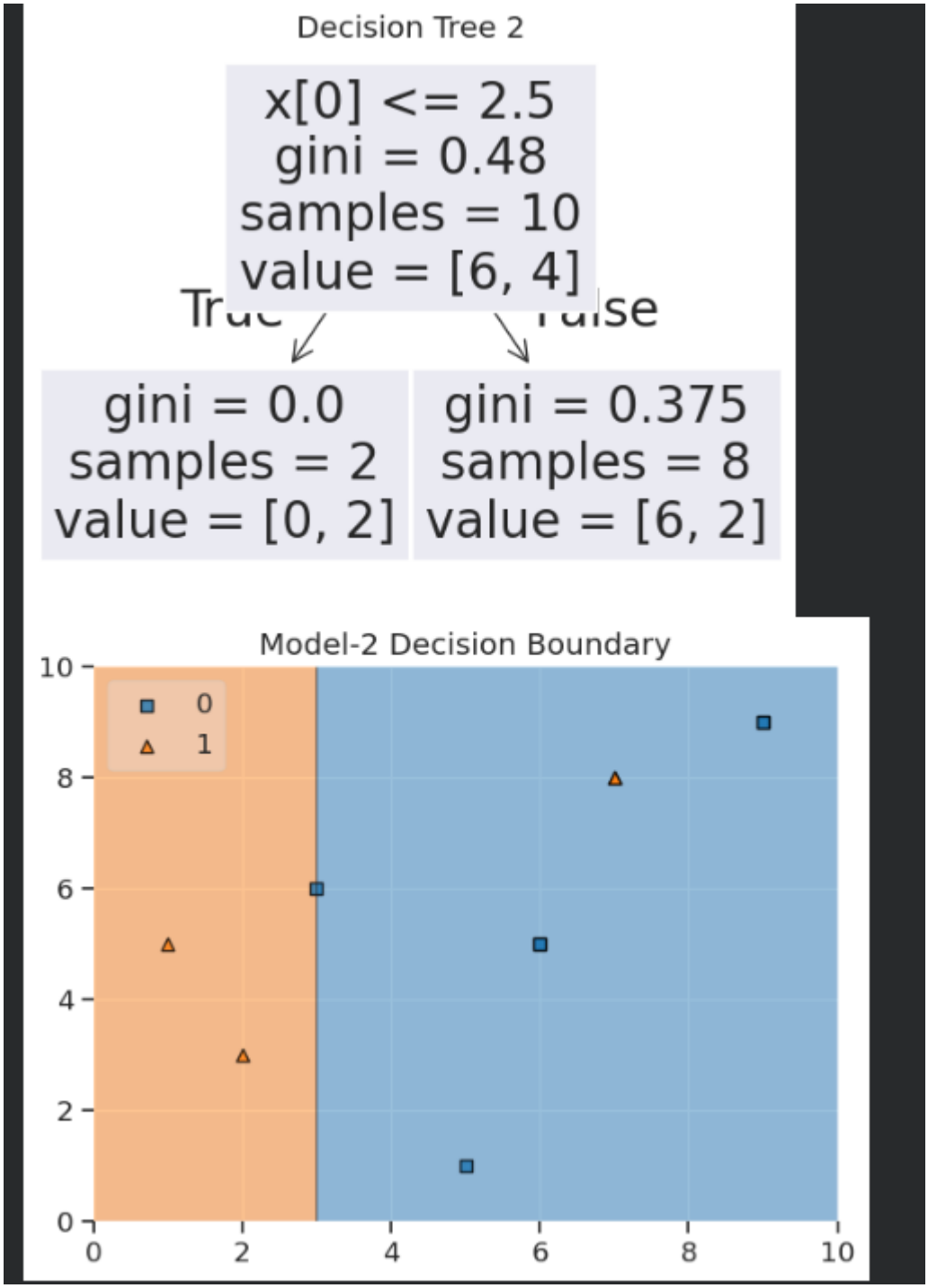


	X1	X2	label	weights	y_pred
0	1	5	1	0.1	1
1	2	3	1	0.1	1
2	3	6	0	0.1	1
3	4	8	1	0.1	1
4	5	1	0	0.1	0
5	6	9	1	0.1	1
6	6	5	0	0.1	1
7	7	8	1	0.1	1
8	9	9	0	0.1	1
9	9	2	0	0.1	0

...

	X1	X2	label	weights	y_pred	updated_weights	cumsum_lower	cumsum_upper
0	1	5	1	0.1	1	0.065465	0.000000	0.071429
1	2	3	1	0.1	1	0.065465	0.071429	0.142857
2	3	6	0	0.1	1	0.152753	0.142857	0.309524
3	4	8	1	0.1	1	0.065465	0.309524	0.380952
4	5	1	0	0.1	0	0.065465	0.380952	0.452381
5	6	9	1	0.1	1	0.065465	0.452381	0.523810
6	6	5	0	0.1	1	0.152753	0.523810	0.690476
7	7	8	1	0.1	1	0.065465	0.690476	0.761905
8	9	9	0	0.1	1	0.152753	0.761905	0.928571
9	9	2	0	0.1	0	0.065465	0.928571	1.000000

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

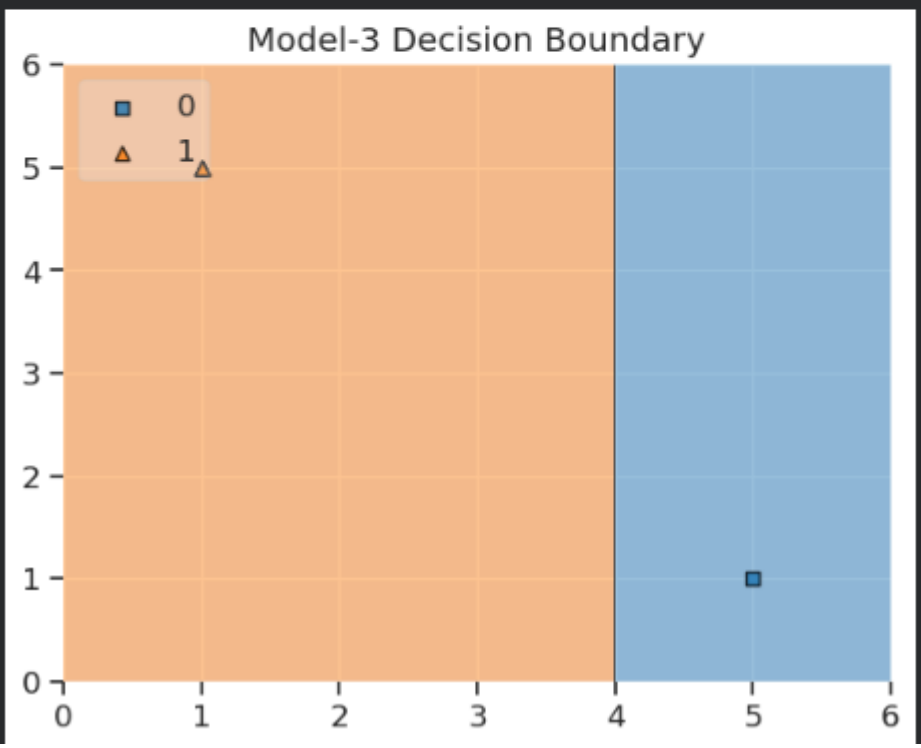

Decision Tree 2
x[0] <= 2.5
gini = 0.48
samples = 10
value = [6, 4]
True
False
gini = 0.0
samples = 2
value = [0, 2]
gini = 0.375
samples = 8
value = [6, 2]
Model-2 Decision Boundary
0
1
0
2
4
6
8
10

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

alpha2 = 1.0986122886681098

	X1	X2	label	weights	y_pred	updated_weights	normalized_weights	cumsum_upper	cumsum_lower
8	9	9	0	0.1	0	0.033333	0.038462	0.038462	0.000000
4	5	1	0	0.1	0	0.033333	0.038462	0.076923	0.038462
7	7	8	1	0.1	0	0.300000	0.346154	0.423077	0.076923
6	6	5	0	0.1	0	0.033333	0.038462	0.461538	0.423077
7	7	8	1	0.1	0	0.300000	0.346154	0.807692	0.461538
8	9	9	0	0.1	0	0.033333	0.038462	0.846154	0.807692
1	2	3	1	0.1	1	0.033333	0.038462	0.884615	0.846154
0	1	5	1	0.1	1	0.033333	0.038462	0.923077	0.884615
6	6	5	0	0.1	0	0.033333	0.038462	0.961538	0.923077
2	3	6	0	0.1	0	0.033333	0.038462	1.000000	0.961538

	X1	X2	label	weights
0	1	5	1	0.1
0	1	5	1	0.1
0	1	5	1	0.1
0	1	5	1	0.1
0	1	5	1	0.1
4	5	1	0	0.1
0	1	5	1	0.1
0	1	5	1	0.1
0	1	5	1	0.1
0	1	5	1	0.1



	X1	X2	label	weights	y_pred
0	1	5	1	0.1	1
0	1	5	1	0.1	1
0	1	5	1	0.1	1
0	1	5	1	0.1	1
0	1	5	1	0.1	1
4	5	1	0	0.1	0
0	1	5	1	0.1	1
0	1	5	1	0.1	1
0	1	5	1	0.1	1
0	1	5	1	0.1	1

```
alpha values: 0.42364893019360184 1.0986122886681098 -0.4236489301936017

Query = [1,5]
DT1: [1]
DT2: [1]
DT3: [1]
Final Score: 1.09861228866811
Final Prediction: 1.0

Query = [9,9]
DT1: [1]
DT2: [0]
DT3: [0]
Final Score: -0.2513144282809062
Final Prediction: -1.0
```