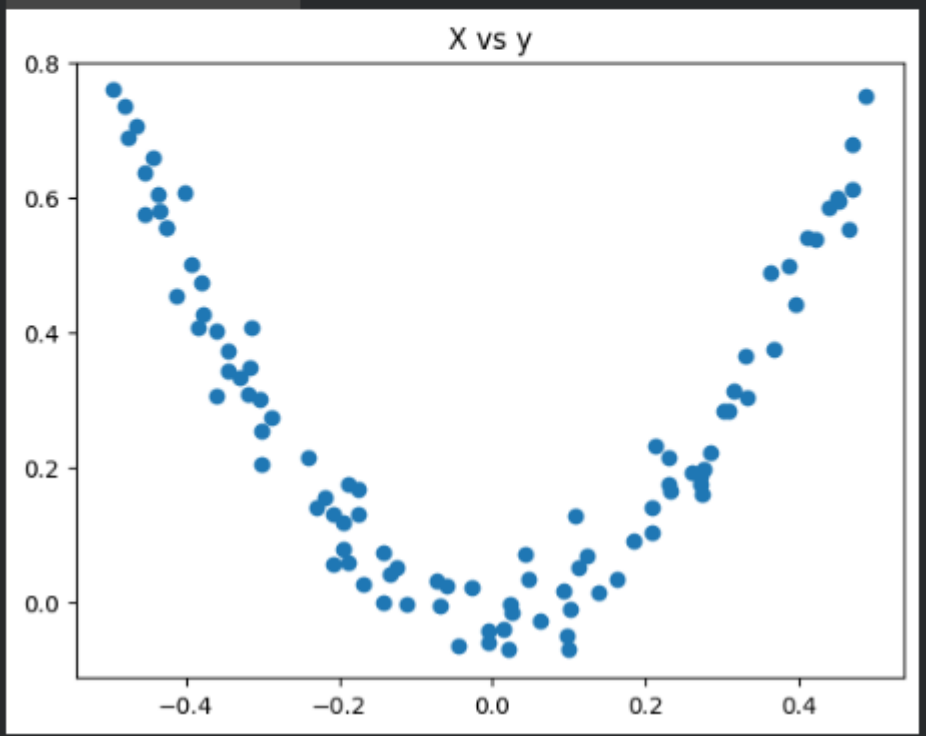
```
# ✅ A PYTHON PROGRAM USING THE GRADIENT BOOSTING MODEL
# ----------------------------------------------------

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import plot_tree

# Step-1: Generate Dataset
np.random.seed(42)
X = np.random.rand(100, 1) - 0.5
y = 3 * X[:, 0] ** 2 + 0.05 * np.random.randn(100)

df = pd.DataFrame({'X': X.reshape(100), 'y': y})
display(df.head())

# Step-2: Scatter Plot
plt.scatter(df['X'], df['y'])
plt.title('X vs y')
plt.show()

# Step-3: Initial Prediction = Mean
df['pred1'] = df['y'].mean()
display(df.head())

# Step-4: Residuals
df['res1'] = df['y'] - df['pred1']
display(df.head())

plt.scatter(df['X'], df['y'])
```

```
plt.plot(df['X'], df['pred1'], color='red')
plt.title("Initial Predictor vs Data")
plt.show()

# Step-5: Train First Tree on Residuals
tree1 = DecisionTreeRegressor(max_leaf_nodes=8)
tree1.fit(df['X'].values.reshape(-1,1), df['res1'])

plt.figure(figsize=(10,4))
plot_tree(tree1, filled=True)
plt.title("Tree 1")
plt.show()

# Step-6: Predict & Plot Model-1
X_test = np.linspace(-0.5, 0.5, 500)
y_pred = df['pred1'].iloc[0] + tree1.predict(X_test.reshape(-1,1))

plt.figure(figsize=(10,4))
plt.plot(X_test, y_pred, linewidth=2, color='red')
plt.scatter(df['X'], df['y'])
plt.title("Model-1 Boosted Fit")
plt.show()

# Step-7: Add predictions to DF
df['pred2'] = df['pred1'] + tree1.predict(df['X'].values.reshape(-1,1))
df['res2'] = df['y'] - df['pred2']
display(df.head())

# Step-8: Train Second Tree
tree2 = DecisionTreeRegressor(max_leaf_nodes=8)
tree2.fit(df['X'].values.reshape(-1,1), df['res2'])
```

```
y_pred2 = df['pred1'].iloc[0] + tree1.predict(X_test.reshape(-1,1)) +
tree2.predict(X_test.reshape(-1,1))


plt.figure(figsize=(10,4))

plt.plot(X_test, y_pred2, linewidth=2, color='red')

plt.scatter(df['X'], df['y'])

plt.title('Model-2 Boosted Fit')

plt.show()


# ✅ Final Boosting Function

def gradient_boost(X, y, number, lr, count=1, regs=[], base_y=None):

  if number == 0:

    return


  # Compute residual

  if count > 1:

    y = y - regs[-1].predict(X)

  else:

    base_y = y


  # Train weak regressor

  tree_reg = DecisionTreeRegressor(max_depth=5, random_state=42)

  tree_reg.fit(X, y)

  regs.append(tree_reg)


  # Plot progressive boosting

  x_line = np.linspace(-0.5, 0.5, 500)

  y_pred = sum(lr * reg.predict(x_line.reshape(-1, 1)) for reg in regs)


  print(f"Iteration {count}")
```

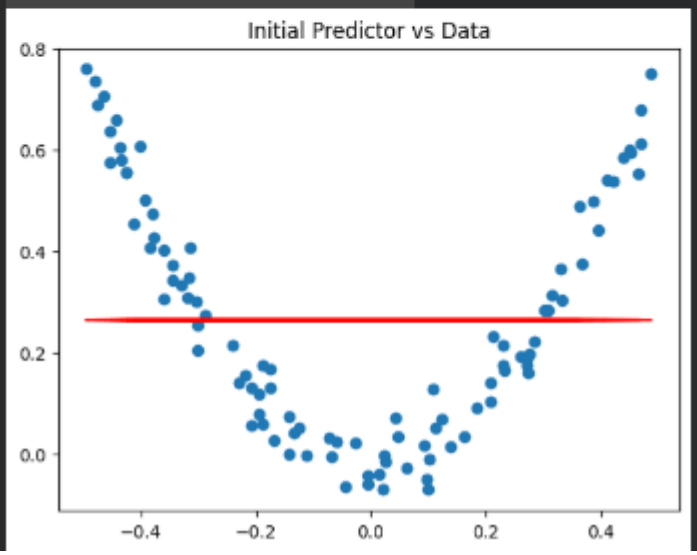
```
    plt.figure()
    plt.plot(x_line, y_pred, linewidth=2)
    plt.scatter(X[:,0], base_y, color="red")
    plt.title(f"Boosted Model - Iteration {count}")
    plt.show()

    gradient_boost(X, y, number-1, lr, count+1, regs, base_y)

# ✨ Run Gradient Boosting with 5 iterations
np.random.seed(42)
X = np.random.rand(100, 1) - 0.5
y = 3*X[:, 0]**2 + 0.05 * np.random.randn(100)

gradient_boost(X, y, number=5, lr=1)
```

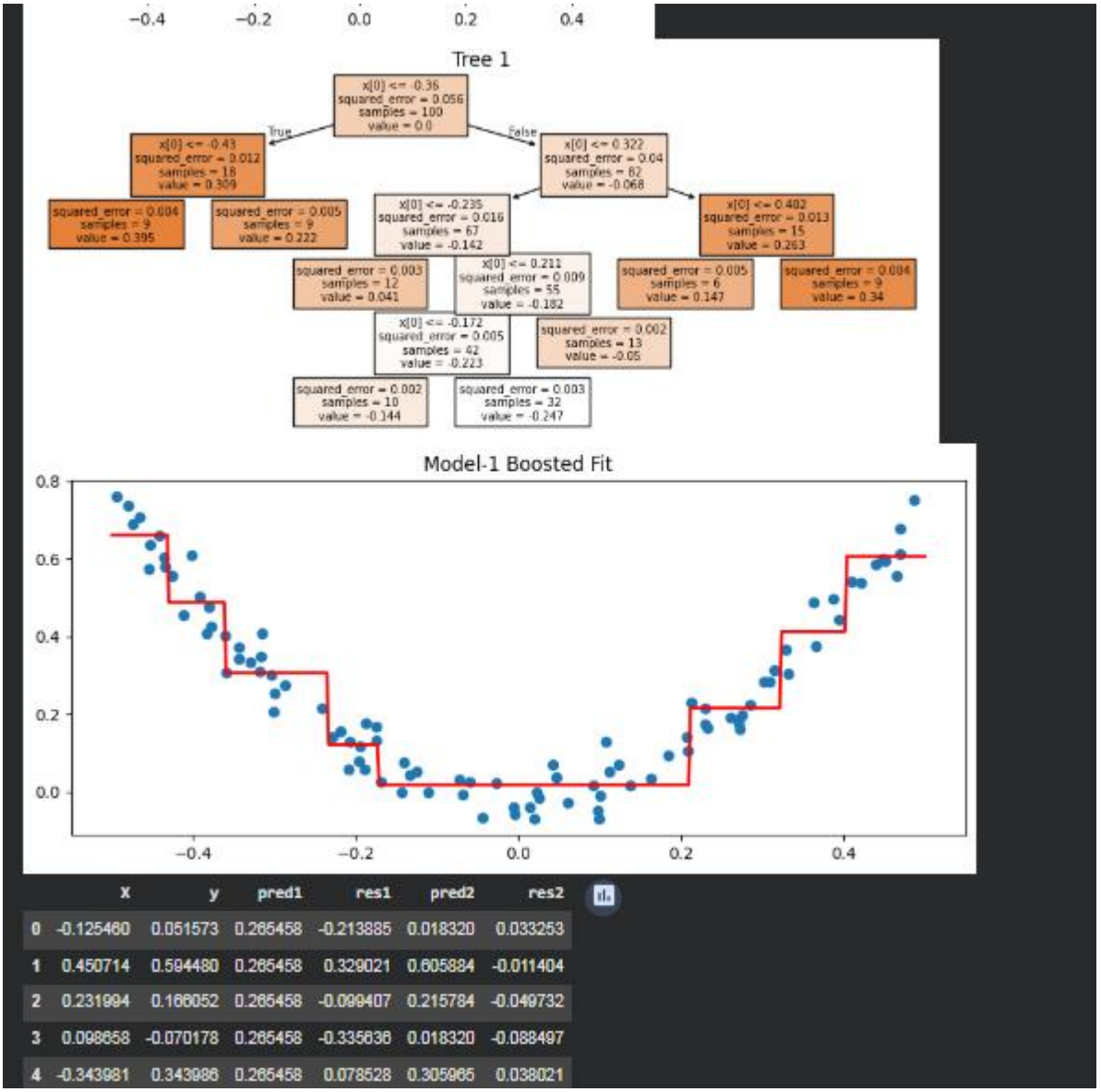
	X	y
0	-0.125460	0.051573
1	0.450714	0.594480
2	0.231994	0.166052
3	0.098658	-0.070178
4	-0.343981	0.343986



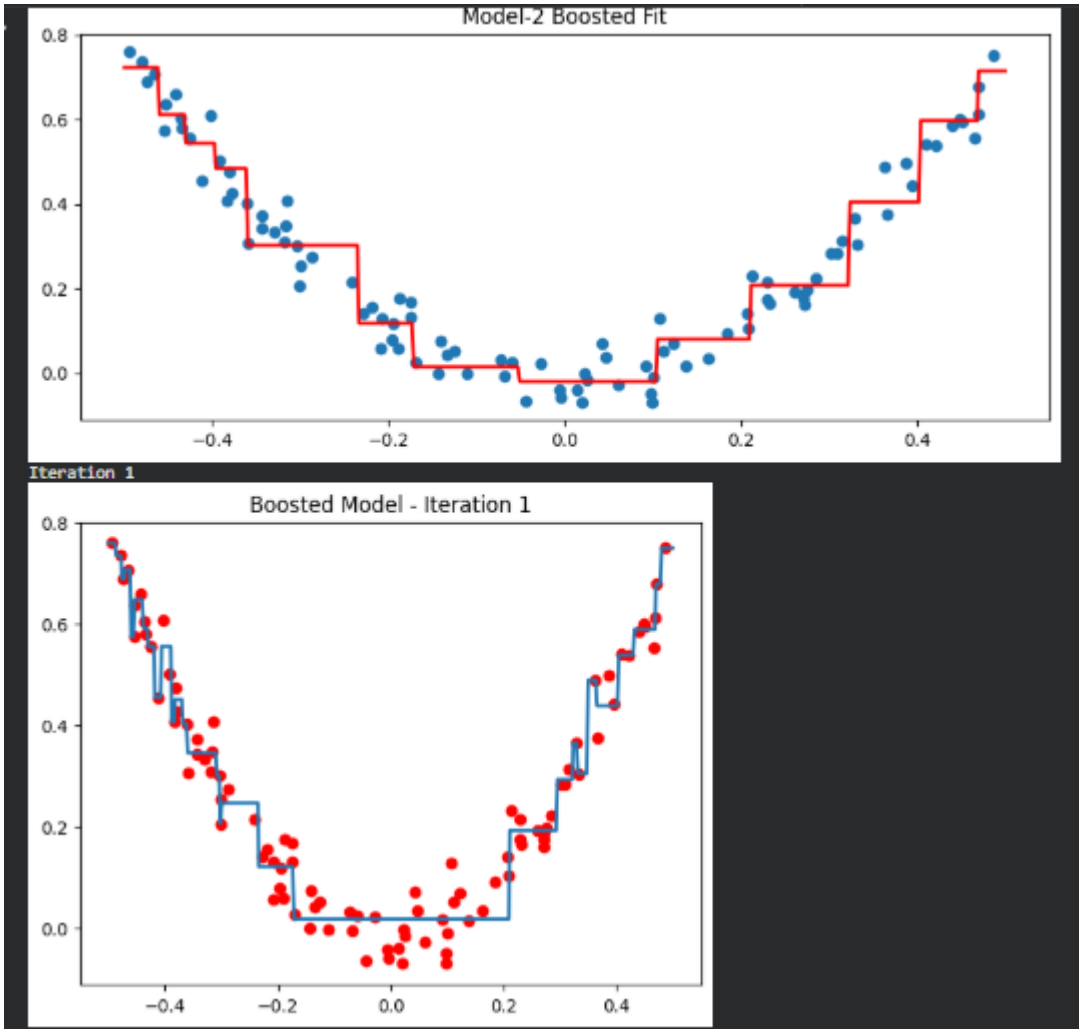
	X	y	pred1
0	-0.125460	0.051573	0.265458
1	0.450714	0.594480	0.265458
2	0.231994	0.166052	0.265458
3	0.098658	-0.070178	0.265458
4	-0.343981	0.343986	0.265458

	X	y	pred1	res1
0	-0.125460	0.051573	0.265458	-0.213885
1	0.450714	0.594480	0.265458	0.329021
2	0.231994	0.166052	0.265458	-0.099407
3	0.098658	-0.070178	0.265458	-0.335636
4	-0.343981	0.343986	0.265458	0.078528

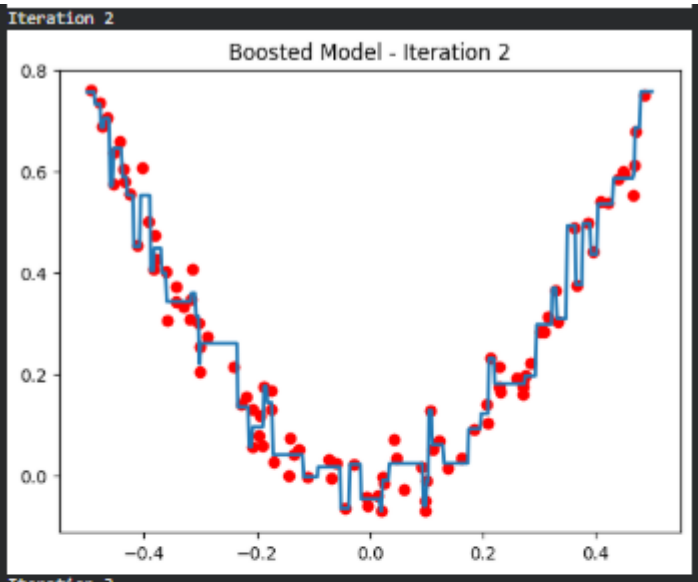




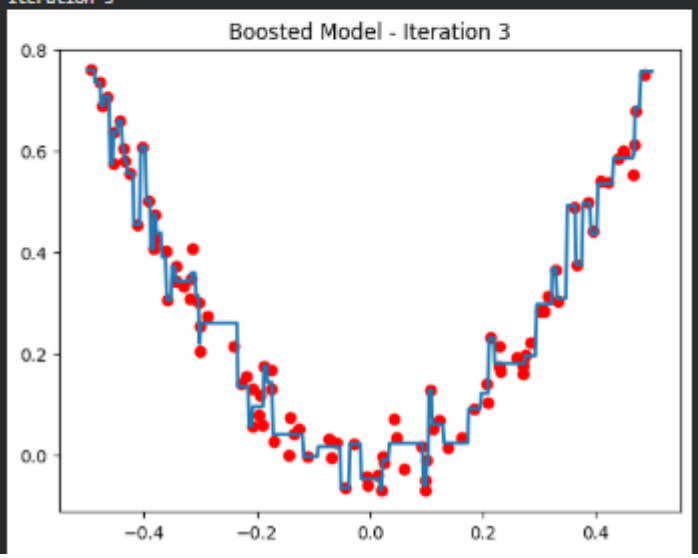
	X	y	pred1	res1	pred2	res2
0	-0.125460	0.051573	0.265458	-0.213885	0.018320	0.033253
1	0.450714	0.594480	0.265458	0.329021	0.605884	-0.011404
2	0.231994	0.166052	0.265458	-0.099407	0.215784	-0.049732
3	0.098658	-0.070178	0.265458	-0.335636	0.018320	-0.088497
4	-0.343981	0.343986	0.265458	0.078528	0.305965	0.038021


Model-2 Boosted Fit
0.8
0.6
0.4
0.2
0.0
−0.4
−0.2
0.0
0.2
0.4
Iteration 1
Boosted Model - Iteration 1
0.8
0.6
0.4
0.2
0.0
−0.4
−0.2
0.0
0.2
0.4

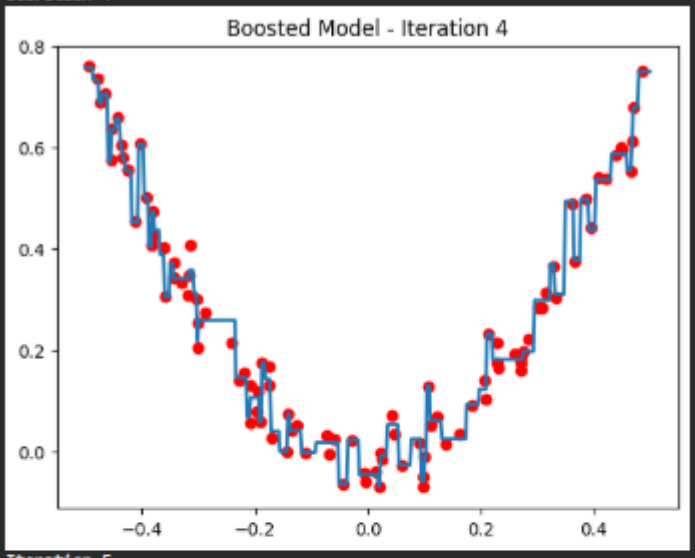
Iteration 2



Iteration 3



Iteration 4



Iteration 5