Ex.No.8b

Date: 25/10/2025

A Python program to Implement Gradient Boosting.

Aim:-

To implement a python program using the Gradient Boosting model.

Algorithm:-

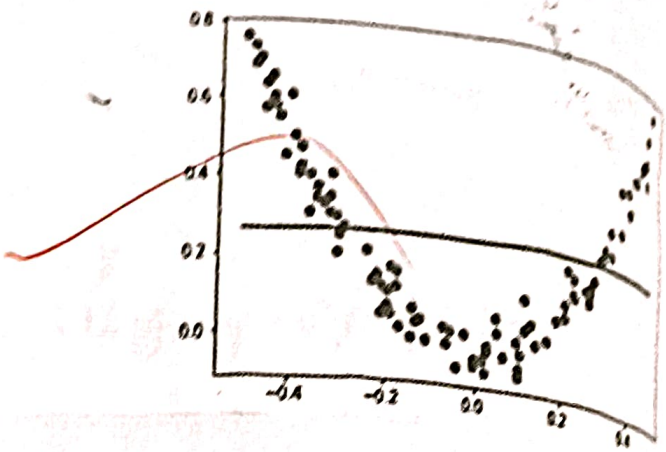
Step 1:- Import necessary libraries
Step 2:- Prepare the data
Step 3:- Initialize parameters
Step 4:- Initialize base model
Step 5:- Iterate over boosting model.
Step 6:- Make predictions on the data.
Step 7:- Evaluate the model.

Program:-

```
Import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
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()
df['X']=X.reshape(100)
df['y']=y
df
```

	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
...	...	...	...	...
95	-0.006204	-0.040675	0.265458	-0.306133
96	0.022733	-0.002305	0.265458	-0.267763
97	-0.072459	0.032809	0.265458	-0.232650
98	-0.474581	0.689516	0.265458	0.424057
99	-0.392109	0.502607	0.265458	0.237148

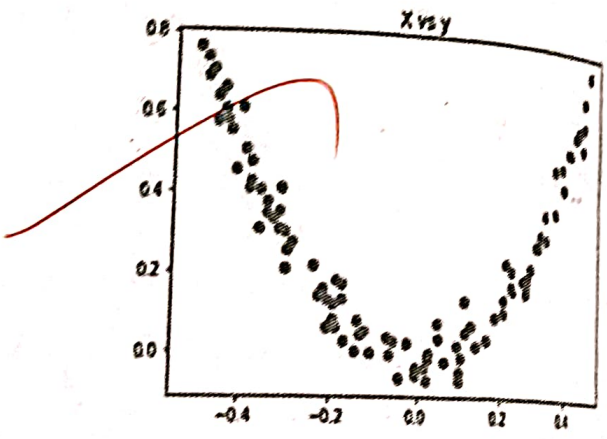
100 rows × 4 columns

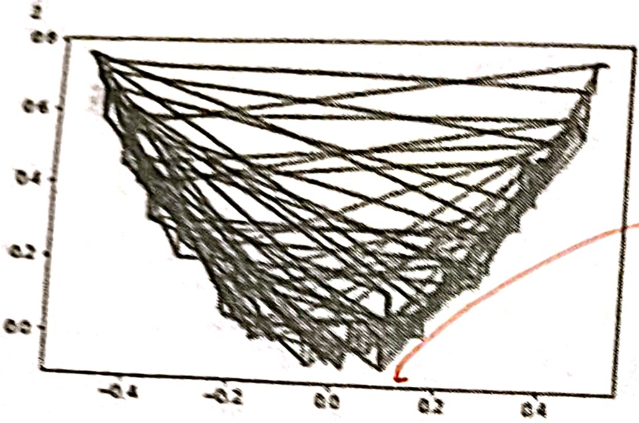


[6]:

	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
...	...	...
95	-0.006204	-0.040675
96	0.022733	-0.002305
97	-0.072459	0.032809
98	-0.474581	0.689516
99	-0.392109	0.502607

[9]: Text(0.5, 1.0, 'X vs y')





	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (
0	1	Male	19	15	
1	2	Male	21	15	
2	3	Female	20	16	
3	4	Female	23	16	
4	5	Female	31	17	
..	...	...	...	...	
195	196	Female	35	120	
196	197	Female	45	126	
197	198	Male	32	126	
198	199	Male	32	137	
199	200	Male	30	137	

[200 rows x 5 columns]

```
plt.scatter(df['x'], df['y'])
plt.title('x' vs 'y')
Text(0.5, 1.0, 'x as y')
df['pred1'] = df['y'].mean()
df
df['res1'] - df['y'] = df['pred1']
df
plt.scatter(df['x'], df['y'])
plt.plot(df['x'], df['pred1'], color='red')
from sklearn.tree import DecisionTreeRegressor
Tree1 = DecisionTreeRegressor(max_leaf_nodes=8)
Tree1.fit(df['x'].values.reshape(100,1),
          df['res1'].values)
DecisionTreeRegressor(max_leaf_nodes=8)
from sklearn.tree import plot_tree
plot_tree(tree1)
plt.show()
X_test = np.linspace(0.5, 0.5, 500)
y_pred = 0.205458 + Tree1.predict(y_test.reshape(500,1))
plt.figure(figsize=(14,4))
plt.subplot(121)
plt.plot(y_test, y_pred, linewidth=2, color='red')
plt.scatter(df['x'], df['y'])
df['pred2'] = 0.265458 + tree1.predict(df['x']
                                       .values.reshape(100,1))
df
df['res2'] = df['y'] - df['pred2']
df
```

```
Tree 2 = Decision Regression (max. leaf.nodes = 8)
Tree 2. fit (df ['x'].values. reshape (100, 1). df ['resr']
                                                values)

Decision TreeRegression (max_leaf_nodes = 8)
y_pred = 0.26545 8 sum ( regression. predict
                (x_test reshape (-1, 1))
            for regressor in [tree1, tree 2]

pu. figure (figure_(4,4))
pu. subplot (121)
pu. plot (x. test, y_pred, uniidth = 2, color = 'red')
pu. scatter (df ['x'], df ['y'])
pel. tetle ('x us y')
def gradient boost (x, y, number, lr, count = 1,
                    regs = [], boo = none).

    If number == 0:
        return

    else
        if count > 1
            y = y_regs [-1] predict (x)

    else :-
        boo = y

bree reg = DecisionTreeRegressor (max, depth = 5,
                    random state = 42)

tree_reg. fit (x y)
regs append (tree_reg)
X1 = np linspace (-0.5, 0.5, 500)
y_pred = sum (lr. regressor. predict (x, reshape)
    (-1,1)) for regressor in regs]
```

```
print (number)
plt.figure ()
plt.plot (x1, y_prod, linewidth = 2)
plt.plot (x[:,0], too, 'r')
plt.show ()
gradient_boost (x,y, number-1, lr, count+1, regs, too
                                                = too)
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, 5, lr = 1)
```

Result:

Thus, the python program to implement gradient boosting for the standard uniform distribution has been successfully implemented.