```
import pandas as pd
import numpy as np
df = pd.DataFrame([[165,137,472,192],[101,92,250,144],
[29,127,201,91]],columns=['R&D','Ops','Marketing','Profit'])
df
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 3,\n  \"fields\": [\n
{\n      \"column\": \"R&D\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 68,\n        \"min\": 29,\n
\"max\": 165,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          165,\n          101,\n          29\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ops\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 23,\n        \"min\": 92,\n
\"max\": 137,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          137,\n          92,\n          127\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Marketing\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
144,\n        \"min\": 201,\n        \"max\": 472,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          472,\n
250,\n          201\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Profit\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 50,\n        \"min\": 91,\n        \"max\": 192,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          192,\n
144,\n          91\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

```
df['f0(x)'] = df['Profit'].mean()
df
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 3,\n  \"fields\": [\n
{\n      \"column\": \"R&D\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 68,\n        \"min\": 29,\n
\"max\": 165,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          165,\n          101,\n          29\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ops\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 23,\n        \"min\": 92,\n
\"max\": 137,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          137,\n          92,\n          127\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Marketing\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
144,\n        \"min\": 201,\n        \"max\": 472,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          472,\n
250,\n          201\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

\"Profit\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 50,\n   \"min\": 91,\n   \"max\": 192,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   192,\n
144,\n   91\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"f0(x)\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.0,\n   \"min\": 142.33333333333334,\n   \"max\":
142.33333333333334,\n   \"num_unique_values\": 1,\n
\"samples\": [\n   142.33333333333334\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
df['r1'] = df['Profit'] - df['f0(x)']
df
```

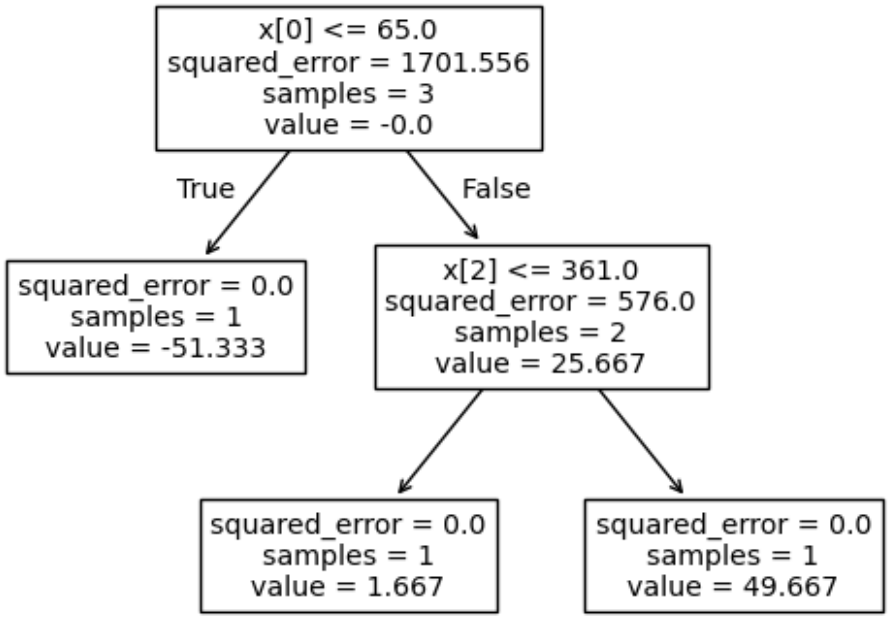
{"summary":"{\n \"name\": \"df\",\n \"rows\": 3,\n \"fields\": [\n
{\n   \"column\": \"R&D\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 68,\n   \"min\": 29,\n
\"max\": 165,\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   165,\n   101,\n   29\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Ops\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 23,\n   \"min\": 92,\n
\"max\": 137,\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   137,\n   92,\n   127\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Marketing\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
144,\n   \"min\": 201,\n   \"max\": 472,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   472,\n
250,\n   201\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Profit\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 50,\n   \"min\": 91,\n   \"max\": 192,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   192,\n
144,\n   91\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"f0(x)\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.0,\n   \"min\": 142.33333333333334,\n   \"max\":
142.33333333333334,\n   \"num_unique_values\": 1,\n
\"samples\": [\n   142.33333333333334\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"r1\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 50.52062285179522,\n
\"min\": -51.33333333333334,\n   \"max\": 49.66666666666666,\n
\"num_unique_values\": 3,\n   \"samples\": [\n
49.66666666666666\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}

```
from sklearn.tree import DecisionTreeRegressor

reg = DecisionTreeRegressor(max_depth=3)

reg.fit(df.iloc[:,0:3].values, df.iloc[:,-1])
```

```
DecisionTreeRegressor(max_depth=3)
```

```
from sklearn.tree import plot_tree
plot_tree(reg)

# 165 137   472
```

```
[Text(0.4, 0.8333333333333334, 'x[0] <= 65.0\nsquared_error =
1701.556\nsamples = 3\nvalue = -0.0'),
 Text(0.2, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue = -51.333'),
 Text(0.30000000000000004, 0.6666666666666667, 'True  '),
 Text(0.6, 0.5, 'x[2] <= 361.0\nsquared_error = 576.0\nsamples = 2\
nvalue = 25.667'),
 Text(0.5, 0.6666666666666667, '  False'),
 Text(0.4, 0.16666666666666666, 'squared_error = 0.0\nsamples = 1\
nvalue = 1.667'),
 Text(0.8, 0.16666666666666666, 'squared_error = 0.0\nsamples = 1\
nvalue = 49.667')]
```



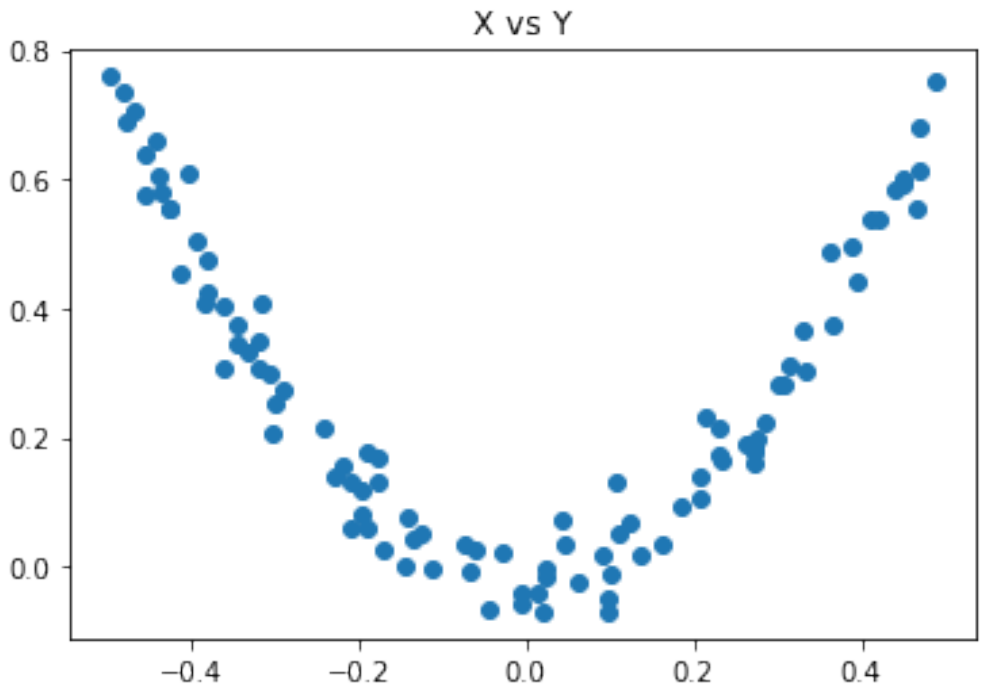

```
import numpy as np
import matplotlib.pyplot as plt
```

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

import pandas as pd

df = pd.DataFrame()

df['X'] = X.reshape(100)
df['y'] = y

df
```

```
           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

[100 rows x 2 columns]
```

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

```
Text(0.5, 1.0, 'X vs Y')
```




```
df['pred1'] = df['y'].mean()

df
```

```
           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
..       ...       ...       ...
95 -0.006204 -0.040675  0.265458
96  0.022733 -0.002305  0.265458
97 -0.072459  0.032809  0.265458
98 -0.474581  0.689516  0.265458
99 -0.392109  0.502607  0.265458

[100 rows x 3 columns]
```

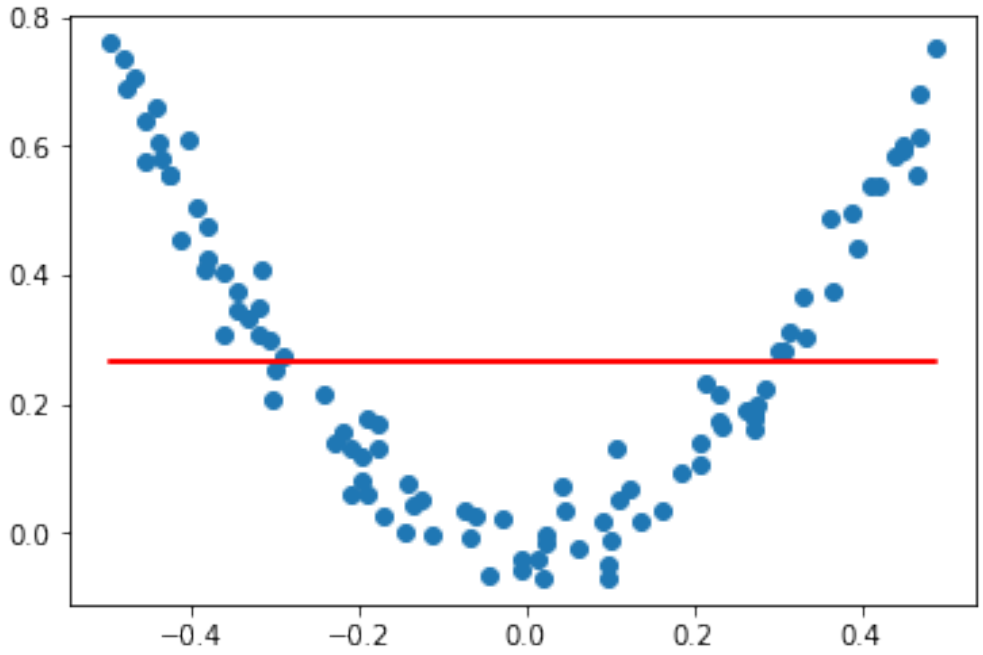
```
df['res1'] = df['y'] - df['pred1']

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 x 4 columns]
```

```
plt.scatter(df['X'],df['y'])
plt.plot(df['X'],df['pred1'],color='red')
```

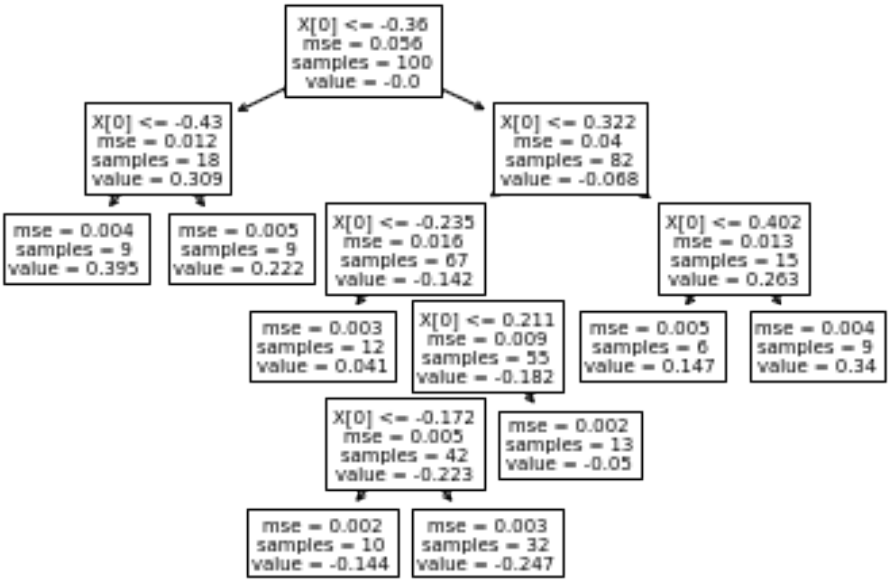
```
[<matplotlib.lines.Line2D at 0x7f40df5f2d90>]
```



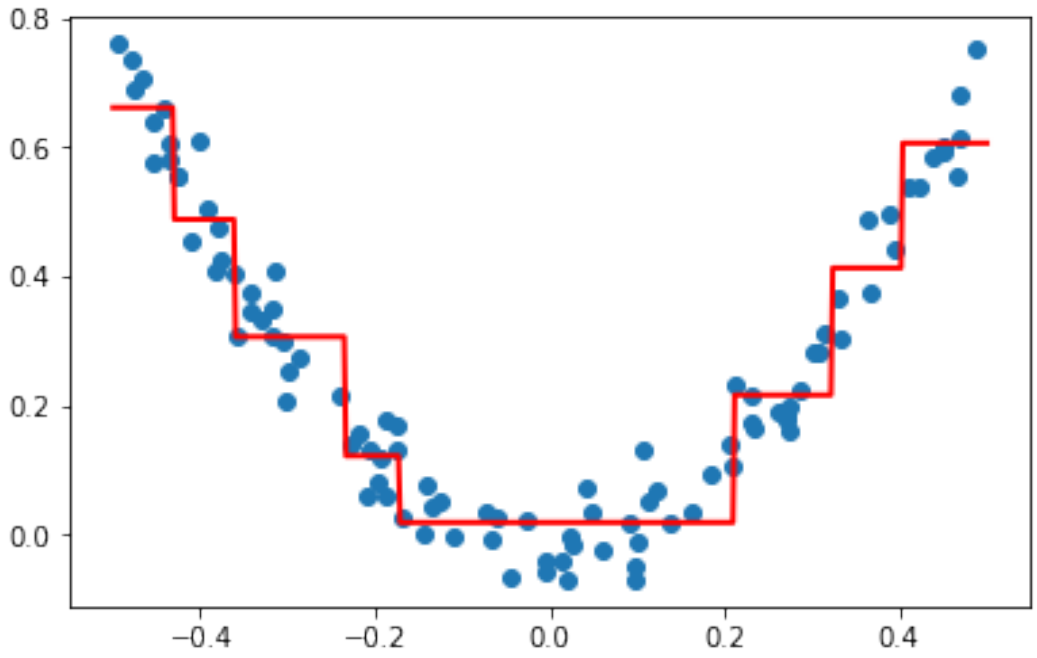

```
from sklearn.tree import DecisionTreeRegressor

tree1 = DecisionTreeRegressor(max_leaf_nodes=8)

tree1.fit(df['X'].values.reshape(100,1),df['res1'].values)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=8,
                      min_impurity_decrease=0.0,
min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0,
presort='deprecated',
                      random_state=None, splitter='best')
```

```
from sklearn.tree import plot_tree
plot_tree(tree1)
plt.show()
```




```
# generating X_test
X_test = np.linspace(-0.5, 0.5, 500)

y_pred = 0.265458 + tree1.predict(X_test.reshape(500, 1))

plt.figure(figsize=(14,4))
plt.subplot(121)
plt.plot(X_test, y_pred, linewidth=2,color='red')
plt.scatter(df['X'],df['y'])

<matplotlib.collections.PathCollection at 0x7f40df5ee810>
```




```
df['pred2'] = 0.265458 + tree1.predict(df['X'].values.reshape(100,1))

df
```

```
           X         y     pred1      res1     pred2
0  -0.125460  0.051573  0.265458 -0.213885  0.018319
1   0.450714  0.594480  0.265458  0.329021  0.605884
2   0.231994  0.166052  0.265458 -0.099407  0.215784
3   0.098658 -0.070178  0.265458 -0.335636  0.018319
4  -0.343981  0.343986  0.265458  0.078528  0.305964
..       ...       ...       ...       ...       ...
95 -0.006204 -0.040675  0.265458 -0.306133  0.018319
96  0.022733 -0.002305  0.265458 -0.267763  0.018319
97 -0.072459  0.032809  0.265458 -0.232650  0.018319
98 -0.474581  0.689516  0.265458  0.424057  0.660912
99 -0.392109  0.502607  0.265458  0.237148  0.487796

[100 rows x 5 columns]
```

```
df['res2'] = df['y'] - df['pred2']

df
```

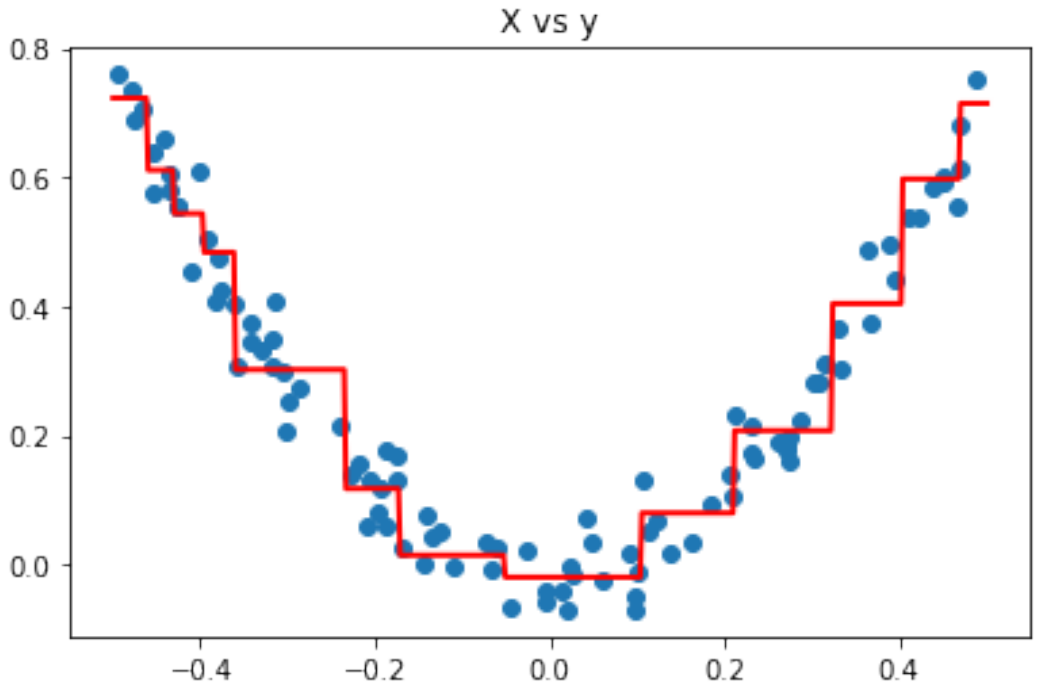
```
           X         y     pred1      res1     pred2      res2
0  -0.125460  0.051573  0.265458 -0.213885  0.018319  0.033254
1   0.450714  0.594480  0.265458  0.329021  0.605884 -0.011404
2   0.231994  0.166052  0.265458 -0.099407  0.215784 -0.049732
3   0.098658 -0.070178  0.265458 -0.335636  0.018319 -0.088497
4  -0.343981  0.343986  0.265458  0.078528  0.305964  0.038022
..       ...       ...       ...       ...       ...       ...
```

```
95 -0.006204 -0.040675  0.265458 -0.306133  0.018319 -0.058994
96  0.022733 -0.002305  0.265458 -0.267763  0.018319 -0.020624
97 -0.072459  0.032809  0.265458 -0.232650  0.018319  0.014489
98 -0.474581  0.689516  0.265458  0.424057  0.660912  0.028604
99 -0.392109  0.502607  0.265458  0.237148  0.487796  0.014810

[100 rows x 6 columns]
```

```
tree2 = DecisionTreeRegressor(max_leaf_nodes=8)

tree2.fit(df['X'].values.reshape(100,1),df['res2'].values)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=8,
                      min_impurity_decrease=0.0,
min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0,
presort='deprecated',
                      random_state=None, splitter='best')
```

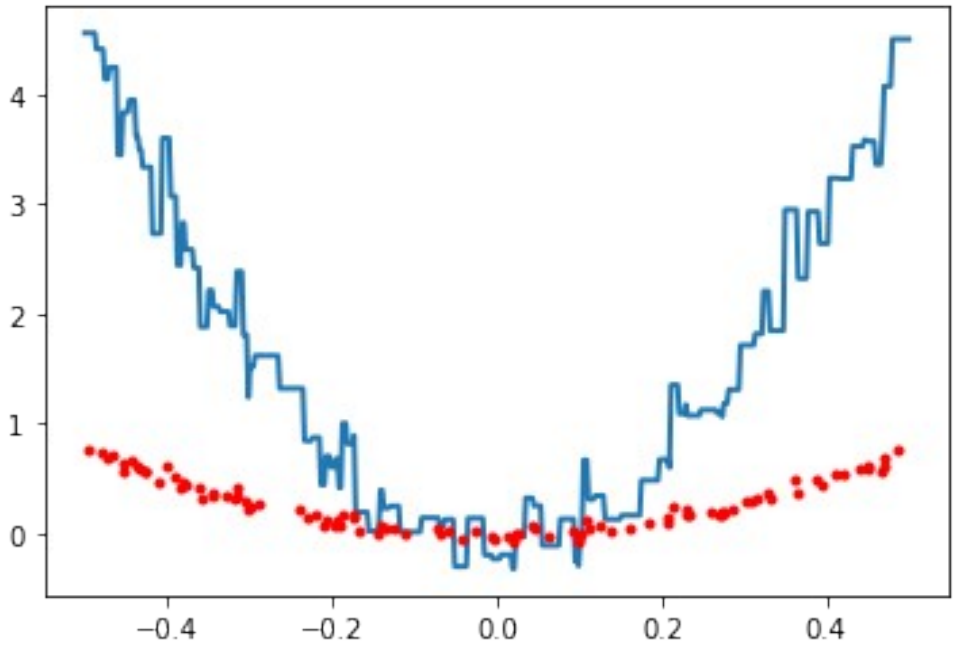
```
y_pred = 0.265458 + sum(regressor.predict(X_test.reshape(-1, 1)) for
regressor in [tree1,tree2])

plt.figure(figsize=(14,4))
plt.subplot(121)
plt.plot(X_test, y_pred, linewidth=2,color='red')
plt.scatter(df['X'],df['y'])
plt.title('X vs y')

Text(0.5, 1.0, 'X vs y')
```



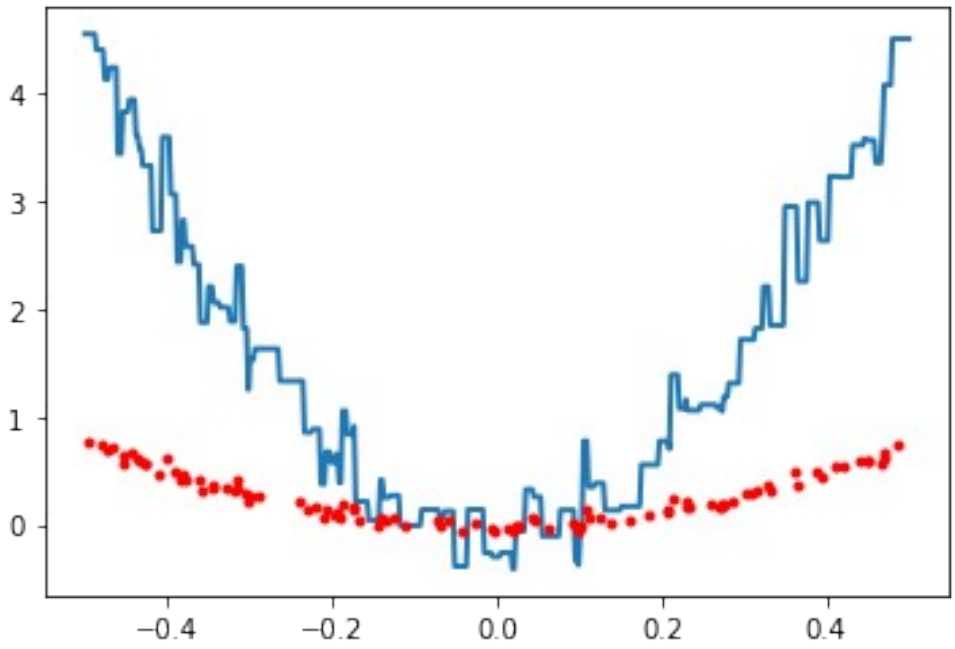

```
def gradient_boost(X,y,number,lr,count=1,regs=[],foo=None):

  if number == 0:
    return
  else:
    # do gradient boosting

    if count > 1:
      y = y - regs[-1].predict(X)
    else:
      foo = y

    tree_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(x1.reshape(-1, 1)) for
regressor in regs)

    print(number)
    plt.figure()
    plt.plot(x1, y_pred, linewidth=2)
    plt.plot(X[:, 0], foo,"r.")
    plt.show()

    gradient_boost(X,y,number-1,lr,count+1,regs,foo=foo)
```

```
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)
```

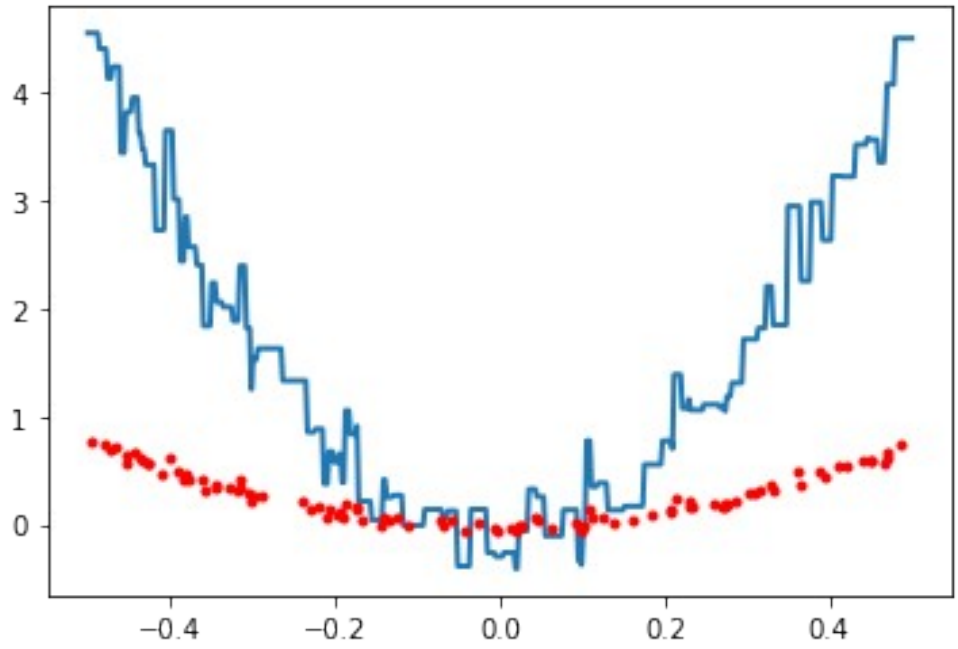
```
5
```



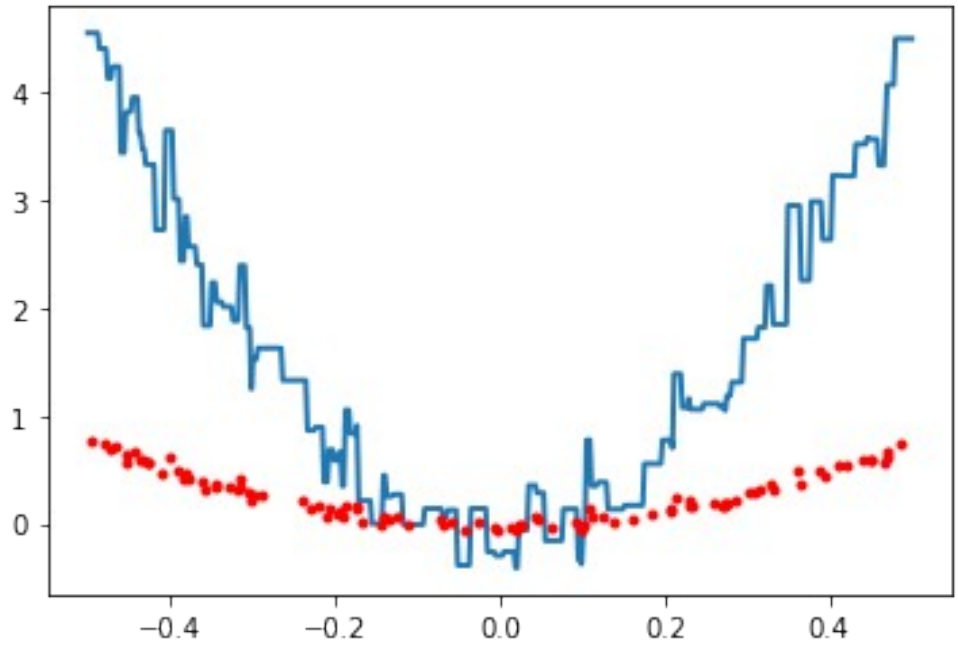

```
4
```



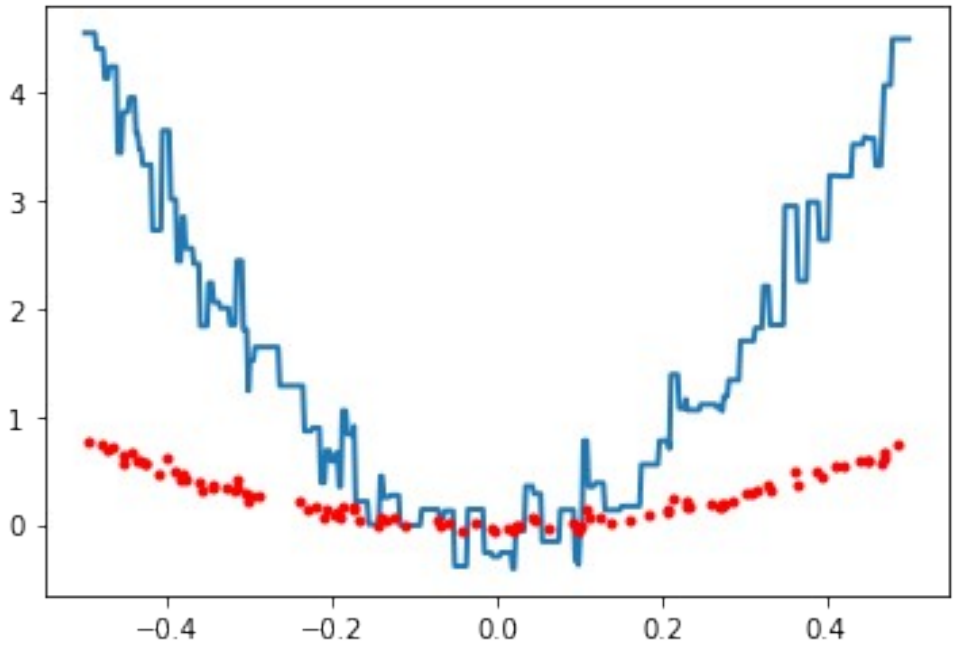
3



2



1


4
3
2
1
0
−0.4
−0.2
0.0
0.2
0.4