In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
```

In [4]:
```
df = pd.read_csv('C:\\Users\\HP\\Desktop\\Rl.csv')
```

In [5]:
```
df
```

Out[5]:

	Field	Date	Total Revenue/Income(Cr)	Cost Of Revenue (Cr)	Gross Profit (Cr)	Total Operating Expense(Cr)	Operating Income/Profit(Cr)	EBITDA(Cr)	Reconciled Depreciation (Cr)	EBIT (Cr)	...	Unnamed: 33	Unnam
0	1.0	1.3.23	216376.0	148559.00	64386.00	177936.00	26984.00	39361.00	11456.00	-958.00	...	NaN	N
1	2.0	1.12.22	220592.0	149165.00	71427.00	185345.00	25060.00	36446.00	10187.00	26259.00	...	NaN	N
2	3.0	1.9.22	232863.0	162379.00	70484.00	201639.00	21494.00	32807.00	9730.00	14131.00	...	NaN	N
3	4.0	1.6.22	223113.0	150678.00	72435.00	190253.00	29051.00	31233.00	8946.00	29745.00	...	NaN	N
4	5.0	1.3.22	211887.0	149521.00	62366.00	184010.00	23365.00	25967.00	8001.00	-3830.00	...	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	
120	NaN	NaN	NaN	2787.00	5266.00	4867.00	7793.00	4390.00	4688.00	3755.00	...	1784.00	192
121	NaN	NaN	NaN	21296.00	17740.00	15587.00	19443.00	18021.00	20539.00	15479.00	...	0.00	
122	NaN	NaN	NaN	19299.00	15792.00	13656.00	17955.00	16203.00	18549.00	13680.00	...	6720.00	622
123	NaN	NaN	NaN	19299.00	15792.00	13656.00	17955.00	16203.00	18549.00	13680.00	...	0.00	
124	NaN	NaN	NaN	28.52	20.78	17.68	26.54	23.95	27.42	21.58	...	11.29	

125 rows × 43 columns

In [6]:
```
del df['Field']
```

In [7]:
```
df = df[:40]
```

In [8]:
```
df = df.drop(df.columns[16:43], axis=1)
```

In [9]:
```
df
```

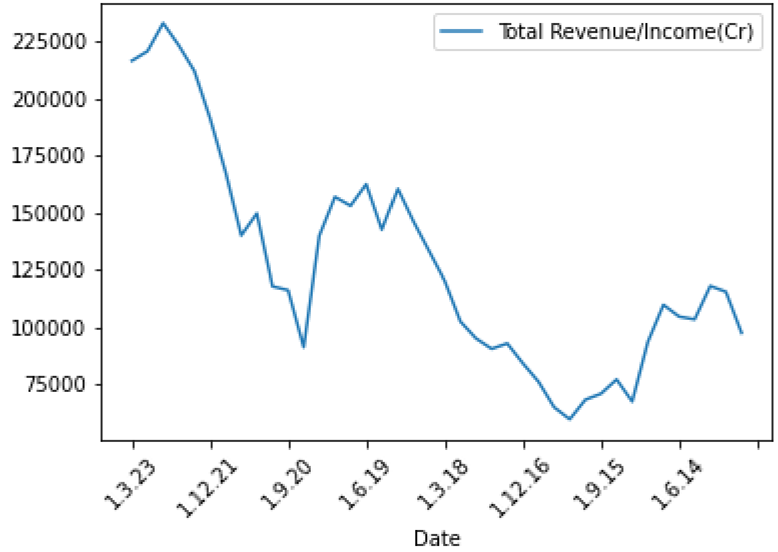
Out[9]:

	Date	Total Revenue/Income(Cr)	Cost Of Revenue (Cr)	Gross Profit (Cr)	Total Operating Expense(Cr)	Operating Income/Profit(Cr)	EBITDA(Cr)	Reconciled Depreciation (Cr)	EBIT (Cr)	Interest Expense (Cr)	Income/Profit Before Tax(Cr)	Income Tax Expense (Cr)
0	1.3.23	216376.0	148559.0	64386.0	177936.0	26984.0	39361.0	11456.0	-958.00	5819.0	24083.0	2787.(
1	1.12.22	220592.0	149165.0	71427.0	185345.0	25060.0	36446.0	10187.0	26259.00	5201.0	23006.0	5266.(
2	1.9.22	232863.0	162379.0	70484.0	201639.0	21494.0	32807.0	9730.0	14131.00	4554.0	20454.0	4867.(
3	1.6.22	223113.0	150678.0	72435.0	190253.0	29051.0	31233.0	8946.0	29745.00	0.0	27236.0	7793.(
4	1.3.22	211887.0	149521.0	62366.0	184010.0	23365.0	25967.0	8001.0	-3830.00	3556.0	22411.0	4390.(
5	1.12.21	191271.0	132413.0	58858.0	163004.0	24859.0	29039.0	7683.0	27049.00	3812.0	25227.0	4688.(
6	1.9.21	167611.0	120659.0	46952.0	28162.0	18790.0	21254.0	7230.0	7141.00	3819.0	19234.0	3755.(
7	1.6.21	139949.0	97188.0	42761.0	123464.0	16485.0	20667.0	6883.0	19134.00	3397.0	17270.0	3464.(
8	1.3.21	149575.0	108510.0	41065.0	133197.0	16378.0	20426.0	6973.0	-6146.00	4044.0	16382.0	1387.(
9	1.12.20	117860.0	78914.0	38946.0	102959.0	14901.0	19308.0	6665.0	19308.00	4326.0	14982.0	88.(
10	1.9.20	116195.0	76410.0	39785.0	98917.0	12319.0	16673.0	6626.0	3739.00	6084.0	10589.0	-13.(
11	1.6.20	91238.0	50449.0	40789.0	77686.0	15533.0	20243.0	6308.0	20243.00	6735.0	13508.0	260.(
12	1.3.20	139865.0	92622.0	47243.0	120344.0	11765.0	13195.0	6332.0	-9008.00	3972.0	9223.0	2677.(
13	1.12.19	156802.0	109334.0	47468.0	136098.0	16664.0	20165.0	5545.0	20366.00	5404.0	14962.0	3121.(
14	1.9.19	152925.0	103857.0	49068.0	131689.0	16837.0	20415.0	5315.0	20505.00	5450.0	15055.0	3703.(
15	1.6.19	162353.0	114329.0	48024.0	140672.0	16604.0	19438.0	5011.0	19475.00	5109.0	14366.0	4225.(
16	1.3.19	142493.0	95623.0	46870.0	122880.0	15786.0	17706.0	5295.0	17771.00	3913.0	13858.0	3431.(
17	1.12.18	160299.0	115261.0	45038.0	144219.0	16080.0	10201.0	0.0	16080.00	-4119.0	14445.0	4069.(
18	1.9.18	146018.0	103174.0	42844.0	130139.0	15879.0	9233.0	0.0	15879.00	-3932.0	13198.0	3649.(
19	1.6.18	133069.0	94314.0	38755.0	117581.0	15488.0	10150.0	0.0	15488.00	-3550.0	13726.0	4241.(
20	1.3.18	120143.0	87666.0	32477.0	106360.0	13783.0	11462.0	0.0	13783.00	-1760.0	13246.0	3787.(

	Date	Total Revenue/Income(Cr)	Cost Of Revenue (Cr)	Gross Profit (Cr)	Total Operating Expense(Cr)	Operating Income/Profit(Cr)	EBITDA(Cr)	Reconciled Depreciation (Cr)	EBIT (Cr)	Interest Expense (Cr)	Income/Profit Before Tax(Cr)	Income Tax Expense (Cr)
21	1.12.17	102500.0	68410.0	34090.0	89442.0	13058.0	11103.0	0.0	13058.00	-2095.0	13220.0	3775.0
22	1.9.17	95085.0	64937.0	30148.0	83807.0	11278.0	9077.0	0.0	11278.00	-2272.0	11337.0	3240.0
23	1.6.17	90537.0	65196.0	25341.0	81020.0	9517.0	10533.0	0.0	9517.00	-1119.0	11623.0	2544.0
24	1.3.17	92889.0	67697.0	25192.0	83547.0	9342.0	9150.0	0.0	9342.00	-1119.0	10279.0	2193.0
25	1.12.16	84189.0	60486.0	23703.0	75430.0	8759.0	9016.0	0.0	8759.00	-1209.0	0.0	2719.0
26	1.9.16	76161.0	56680.0	19481.0	11079.0	8402.0	10807.0	0.0	10807.00	893.0	9902.0	2708.0
27	1.6.16	64990.0	45783.0	19207.0	10709.0	8498.0	10900.0	3229.0	7671.00	1206.0	9670.0	2581.0
28	1.3.16	59696.0	40278.0	19418.0	11302.0	8116.0	10439.0	3229.0	7210.00	842.0	9597.0	2377.0
29	1.12.15	68261.0	49451.0	18810.0	10575.0	8235.0	10574.0	0.0	7345.00	921.0	9740.0	2363.0
30	1.9.15	70901.0	52622.0	18279.0	10746.0	7533.0	9476.0	0.0	6589.25	972.0	8409.0	1784.0
31	1.6.15	77130.0	58963.0	18167.0	11031.0	7136.0	9053.0	0.0	6166.25	902.0	8066.0	1929.0
32	1.3.15	67470.0	71056.0	-3586.0	-10752.0	7166.0	8308.0	0.0	5421.25	-153.0	8694.0	2080.0
33	1.12.14	93528.0	76434.0	17094.0	11359.0	5735.0	8140.0	0.0	5253.25	1137.0	6938.0	1747.0
34	1.9.14	109797.0	91768.0	18029.0	11235.0	6794.0	8851.0	0.0	6050.75	997.0	7806.0	1882.0
35	1.6.14	104640.0	87919.0	16721.0	10514.0	6207.0	8227.0	0.0	5426.75	505.0	7676.0	1765.0
36	1.3.14	103428.0	106455.0	-3027.0	-9682.0	6655.0	7289.0	0.0	4488.75	-351.0	7725.0	1759.0
37	1.12.13	118038.0	102619.0	15419.0	9544.0	5875.0	7957.0	0.0	5156.75	961.0	6990.0	1494.0
38	1.9.13	115491.0	99950.0	15541.0	9472.0	6069.0	8439.0	0.0	5090.75	959.0	7456.0	1607.0
39	1.6.13	97503.0	83564.0	13939.0	8801.0	5128.0	7520.0	0.0	4181.75	938.0	6592.0	1355.0

```
In [10]: df.plot( x = 'Date', y = 'Total Revenue/Income(Cr)')
plt.xticks(rotation = 45)
```

Out[10]:
```
(array([-5.,  0.,  5., 10., 15., 20., 25., 30., 35., 40., 45.]),
 [Text(-5.0, 0, '1.6.14'),
  Text(0.0, 0, '1.3.23'),
  Text(5.0, 0, '1.12.21'),
  Text(10.0, 0, '1.9.20'),
  Text(15.0, 0, '1.6.19'),
  Text(20.0, 0, '1.3.18'),
  Text(25.0, 0, '1.12.16'),
  Text(30.0, 0, '1.9.15'),
  Text(35.0, 0, '1.6.14'),
  Text(40.0, 0, ''),
  Text(45.0, 0, '')])
```



In [11]:
```python
df['Date'] = pd.to_datetime(df['Date'])
```

In [12]:
```python
df
```

Out[12]:

	Date	Total Revenue/Income(Cr)	Cost Of Revenue (Cr)	Gross Profit (Cr)	Total Operating Expense(Cr)	Operating Income/Profit(Cr)	EBITDA(Cr)	Reconciled Depreciation (Cr)	EBIT (Cr)	Interest Expense (Cr)	Income/Profit Before Tax(Cr)	Income Tax Expense (Cr)
0	2023-01-03	216376.0	148559.0	64386.0	177936.0	26984.0	39361.0	11456.0	-958.00	5819.0	24083.0	2787.0
1	2022-01-12	220592.0	149165.0	71427.0	185345.0	25060.0	36446.0	10187.0	26259.00	5201.0	23006.0	5266.0
2	2022-01-09	232863.0	162379.0	70484.0	201639.0	21494.0	32807.0	9730.0	14131.00	4554.0	20454.0	4867.0
3	2022-01-06	223113.0	150678.0	72435.0	190253.0	29051.0	31233.0	8946.0	29745.00	0.0	27236.0	7793.0
4	2022-01-03	211887.0	149521.0	62366.0	184010.0	23365.0	25967.0	8001.0	-3830.00	3556.0	22411.0	4390.0
5	2021-01-12	191271.0	132413.0	58858.0	163004.0	24859.0	29039.0	7683.0	27049.00	3812.0	25227.0	4688.0
6	2021-01-09	167611.0	120659.0	46952.0	28162.0	18790.0	21254.0	7230.0	7141.00	3819.0	19234.0	3755.0
7	2021-01-06	139949.0	97188.0	42761.0	123464.0	16485.0	20667.0	6883.0	19134.00	3397.0	17270.0	3464.0
8	2021-01-03	149575.0	108510.0	41065.0	133197.0	16378.0	20426.0	6973.0	-6146.00	4044.0	16382.0	1387.0
9	2020-01-12	117860.0	78914.0	38946.0	102959.0	14901.0	19308.0	6665.0	19308.00	4326.0	14982.0	88.0
10	2020-01-09	116195.0	76410.0	39785.0	98917.0	12319.0	16673.0	6626.0	3739.00	6084.0	10589.0	-13.0
11	2020-01-06	91238.0	50449.0	40789.0	77686.0	15533.0	20243.0	6308.0	20243.00	6735.0	13508.0	260.0
12	2020-01-03	139865.0	92622.0	47243.0	120344.0	11765.0	13195.0	6332.0	-9008.00	3972.0	9223.0	2677.0

	Date	Total Revenue/Income(Cr)	Cost Of Revenue (Cr)	Gross Profit (Cr)	Total Operating Expense(Cr)	Operating Income/Profit(Cr)	EBITDA(Cr)	Reconciled Depreciation (Cr)	EBIT (Cr)	Interest Expense (Cr)	Income/Profit Before Tax(Cr)	Income Tax Expense (Cr)
13	2019-01-12	156802.0	109334.0	47468.0	136098.0	16664.0	20165.0	5545.0	20366.00	5404.0	14962.0	3121.0
14	2019-01-09	152925.0	103857.0	49068.0	131689.0	16837.0	20415.0	5315.0	20505.00	5450.0	15055.0	3703.0
15	2019-01-06	162353.0	114329.0	48024.0	140672.0	16604.0	19438.0	5011.0	19475.00	5109.0	14366.0	4225.0
16	2019-01-03	142493.0	95623.0	46870.0	122880.0	15786.0	17706.0	5295.0	17771.00	3913.0	13858.0	3431.0
17	2018-01-12	160299.0	115261.0	45038.0	144219.0	16080.0	10201.0	0.0	16080.00	-4119.0	14445.0	4069.0
18	2018-01-09	146018.0	103174.0	42844.0	130139.0	15879.0	9233.0	0.0	15879.00	-3932.0	13198.0	3649.0
19	2018-01-06	133069.0	94314.0	38755.0	117581.0	15488.0	10150.0	0.0	15488.00	-3550.0	13726.0	4241.0
20	2018-01-03	120143.0	87666.0	32477.0	106360.0	13783.0	11462.0	0.0	13783.00	-1760.0	13246.0	3787.0
21	2017-01-12	102500.0	68410.0	34090.0	89442.0	13058.0	11103.0	0.0	13058.00	-2095.0	13220.0	3775.0
22	2017-01-09	95085.0	64937.0	30148.0	83807.0	11278.0	9077.0	0.0	11278.00	-2272.0	11337.0	3240.0
23	2017-01-06	90537.0	65196.0	25341.0	81020.0	9517.0	10533.0	0.0	9517.00	-1119.0	11623.0	2544.0
24	2017-01-03	92889.0	67697.0	25192.0	83547.0	9342.0	9150.0	0.0	9342.00	-1119.0	10279.0	2193.0
25	2016-01-12	84189.0	60486.0	23703.0	75430.0	8759.0	9016.0	0.0	8759.00	-1209.0	0.0	2719.0

	Date	Total Revenue/Income(Cr)	Cost Of Revenue (Cr)	Gross Profit (Cr)	Total Operating Expense(Cr)	Operating Income/Profit(Cr)	EBITDA(Cr)	Reconciled Depreciation (Cr)	EBIT (Cr)	Interest Expense (Cr)	Income/Profit Before Tax(Cr)	Income Tax Expense (Cr)
26	2016-01-09	76161.0	56680.0	19481.0	11079.0	8402.0	10807.0	0.0	10807.00	893.0	9902.0	2708.0
27	2016-01-06	64990.0	45783.0	19207.0	10709.0	8498.0	10900.0	3229.0	7671.00	1206.0	9670.0	2581.0
28	2016-01-03	59696.0	40278.0	19418.0	11302.0	8116.0	10439.0	3229.0	7210.00	842.0	9597.0	2377.0
29	2015-01-12	68261.0	49451.0	18810.0	10575.0	8235.0	10574.0	0.0	7345.00	921.0	9740.0	2363.0
30	2015-01-09	70901.0	52622.0	18279.0	10746.0	7533.0	9476.0	0.0	6589.25	972.0	8409.0	1784.0
31	2015-01-06	77130.0	58963.0	18167.0	11031.0	7136.0	9053.0	0.0	6166.25	902.0	8066.0	1929.0
32	2015-01-03	67470.0	71056.0	-3586.0	-10752.0	7166.0	8308.0	0.0	5421.25	-153.0	8694.0	2080.0
33	2014-01-12	93528.0	76434.0	17094.0	11359.0	5735.0	8140.0	0.0	5253.25	1137.0	6938.0	1747.0
34	2014-01-09	109797.0	91768.0	18029.0	11235.0	6794.0	8851.0	0.0	6050.75	997.0	7806.0	1882.0
35	2014-01-06	104640.0	87919.0	16721.0	10514.0	6207.0	8227.0	0.0	5426.75	505.0	7676.0	1765.0
36	2014-01-03	103428.0	106455.0	-3027.0	-9682.0	6655.0	7289.0	0.0	4488.75	-351.0	7725.0	1759.0
37	2013-01-12	118038.0	102619.0	15419.0	9544.0	5875.0	7957.0	0.0	5156.75	961.0	6990.0	1494.0
38	2013-01-09	115491.0	99950.0	15541.0	9472.0	6069.0	8439.0	0.0	5090.75	959.0	7456.0	1607.0

	Date	Total Revenue/Income(Cr)	Cost Of Revenue (Cr)	Gross Profit (Cr)	Total Operating Expense(Cr)	Operating Income/Profit(Cr)	EBITDA(Cr)	Reconciled Depreciation (Cr)	EBIT (Cr)	Interest Expense (Cr)	Income/Profit Before Tax(Cr)	Income Tax Expense (Cr)
39	2013-01-06	97503.0	83564.0	13939.0	8801.0	5138.0	7530.0	0.0	4181.75	938.0	6592.0	1355.0

In [13]:
```
#Seperating into feature and target variable
X = df.drop(['EPS (Earning Per Share)', 'Date'], axis=1)
y = df['EPS (Earning Per Share)']
```

In [14]:
```
X
```

Out[14]:

	Total Revenue/Income(Cr)	Cost Of Revenue (Cr)	Gross Profit (Cr)	Total Operating Expense(Cr)	Operating Income/Profit(Cr)	EBITDA(Cr)	Reconciled Depreciation (Cr)	EBIT (Cr)	Interest Expense (Cr)	Income/Profit Before Tax(Cr)	Income Tax Expense (Cr)	Net Cor Opera
0	216376.0	148559.0	64386.0	177936.0	26984.0	39361.0	11456.0	-958.00	5819.0	24083.0	2787.0	
1	220592.0	149165.0	71427.0	185345.0	25060.0	36446.0	10187.0	26259.00	5201.0	23006.0	5266.0	
2	232863.0	162379.0	70484.0	201639.0	21494.0	32807.0	9730.0	14131.00	4554.0	20454.0	4867.0	
3	223113.0	150678.0	72435.0	190253.0	29051.0	31233.0	8946.0	29745.00	0.0	27236.0	7793.0	
4	211887.0	149521.0	62366.0	184010.0	23365.0	25967.0	8001.0	-3830.00	3556.0	22411.0	4390.0	
5	191271.0	132413.0	58858.0	163004.0	24859.0	29039.0	7683.0	27049.00	3812.0	25227.0	4688.0	
6	167611.0	120659.0	46952.0	28162.0	18790.0	21254.0	7230.0	7141.00	3819.0	19234.0	3755.0	
7	139949.0	97188.0	42761.0	123464.0	16485.0	20667.0	6883.0	19134.00	3397.0	17270.0	3464.0	
8	149575.0	108510.0	41065.0	133197.0	16378.0	20426.0	6973.0	-6146.00	4044.0	16382.0	1387.0	
9	117860.0	78914.0	38946.0	102959.0	14901.0	19308.0	6665.0	19308.00	4326.0	14982.0	88.0	
10	116195.0	76410.0	39785.0	98917.0	12319.0	16673.0	6626.0	3739.00	6084.0	10589.0	-13.0	
11	91238.0	50449.0	40789.0	77686.0	15533.0	20243.0	6308.0	20243.00	6735.0	13508.0	260.0	
12	139865.0	92622.0	47243.0	120344.0	11765.0	13195.0	6332.0	-9008.00	3972.0	9223.0	2677.0	
13	156802.0	109334.0	47468.0	136098.0	16664.0	20165.0	5545.0	20366.00	5404.0	14962.0	3121.0	
14	152925.0	103857.0	49068.0	131689.0	16837.0	20415.0	5315.0	20505.00	5450.0	15055.0	3703.0	
15	162353.0	114329.0	48024.0	140672.0	16604.0	19438.0	5011.0	19475.00	5109.0	14366.0	4225.0	
16	142493.0	95623.0	46870.0	122880.0	15786.0	17706.0	5295.0	17771.00	3913.0	13858.0	3431.0	
17	160299.0	115261.0	45038.0	144219.0	16080.0	10201.0	0.0	16080.00	-4119.0	14445.0	4069.0	
18	146018.0	103174.0	42844.0	130139.0	15879.0	9233.0	0.0	15879.00	-3932.0	13198.0	3649.0	
19	133069.0	94314.0	38755.0	117581.0	15488.0	10150.0	0.0	15488.00	-3550.0	13726.0	4241.0	
20	120143.0	87666.0	32477.0	106360.0	13783.0	11462.0	0.0	13783.00	-1760.0	13246.0	3787.0	

	Total Revenue/Income(Cr)	Cost Of Revenue (Cr)	Gross Profit (Cr)	Total Operating Expense(Cr)	Operating Income/Profit(Cr)	EBITDA(Cr)	Reconciled Depreciation (Cr)	EBIT (Cr)	Interest Expense (Cr)	Income/Profit Before Tax(Cr)	Income Tax Expense (Cr)	Net Cor Opera
21	102500.0	68410.0	34090.0	89442.0	13058.0	11103.0	0.0	13058.00	-2095.0	13220.0	3775.0	
22	95085.0	64937.0	30148.0	83807.0	11278.0	9077.0	0.0	11278.00	-2272.0	11337.0	3240.0	
23	90537.0	65196.0	25341.0	81020.0	9517.0	10533.0	0.0	9517.00	-1119.0	11623.0	2544.0	
24	92889.0	67697.0	25192.0	83547.0	9342.0	9150.0	0.0	9342.00	-1119.0	10279.0	2193.0	
25	84189.0	60486.0	23703.0	75430.0	8759.0	9016.0	0.0	8759.00	-1209.0	0.0	2719.0	
26	76161.0	56680.0	19481.0	11079.0	8402.0	10807.0	0.0	10807.00	893.0	9902.0	2708.0	
27	64990.0	45783.0	19207.0	10709.0	8498.0	10900.0	3229.0	7671.00	1206.0	9670.0	2581.0	
28	59696.0	40278.0	19418.0	11302.0	8116.0	10439.0	3229.0	7210.00	842.0	9597.0	2377.0	
29	68261.0	49451.0	18810.0	10575.0	8235.0	10574.0	0.0	7345.00	921.0	9740.0	2363.0	
30	70901.0	52622.0	18279.0	10746.0	7533.0	9476.0	0.0	6589.25	972.0	8409.0	1784.0	
31	77130.0	58963.0	18167.0	11031.0	7136.0	9053.0	0.0	6166.25	902.0	8066.0	1929.0	
32	67470.0	71056.0	-3586.0	-10752.0	7166.0	8308.0	0.0	5421.25	-153.0	8694.0	2080.0	
33	93528.0	76434.0	17094.0	11359.0	5735.0	8140.0	0.0	5253.25	1137.0	6938.0	1747.0	
34	109797.0	91768.0	18029.0	11235.0	6794.0	8851.0	0.0	6050.75	997.0	7806.0	1882.0	
35	104640.0	87919.0	16721.0	10514.0	6207.0	8227.0	0.0	5426.75	505.0	7676.0	1765.0	
36	103428.0	106455.0	-3027.0	-9682.0	6655.0	7289.0	0.0	4488.75	-351.0	7725.0	1759.0	
37	118038.0	102619.0	15419.0	9544.0	5875.0	7957.0	0.0	5156.75	961.0	6990.0	1494.0	
38	115491.0	99950.0	15541.0	9472.0	6069.0	8439.0	0.0	5090.75	959.0	7456.0	1607.0	
39	97503.0	83564.0	13939.0	8801.0	5128.0	7530.0	0.0	4181.75	938.0	6593.0	1355.0	

In [15]:

```
y
```

Out[15]:
```
0     28.52
1     20.78
2     17.68
3     26.54
4     23.95
5     27.42
6     21.58
7     19.36
8     20.86
9     20.67
10    15.09
11    20.87
12     9.39
13    17.21
14    16.66
15    14.94
16    15.33
17    15.16
18    13.98
19     0.00
20    13.72
21     0.00
22    13.53
23     0.00
24    13.47
25     0.00
26    12.09
27     0.00
28    12.02
29     0.00
30    11.29
31     0.00
32     9.90
33     0.00
34    10.05
35     0.00
36     9.12
37     0.00
38     0.00
39     0.00
Name: EPS (Earning Per Share), dtype: float64
```

In [53]:
```
from sklearn.model_selection import train_test_split
```

```
# Assuming X and y are your features and labels, respectively

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [54]:
```
print("Training set size:", len(X_train))
print("Test set size:", len(X_test))
```

```
Training set size: 28
Test set size: 12
```

In [55]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df['DateEncoded'] = label_encoder.fit_transform(df['Date'])
```

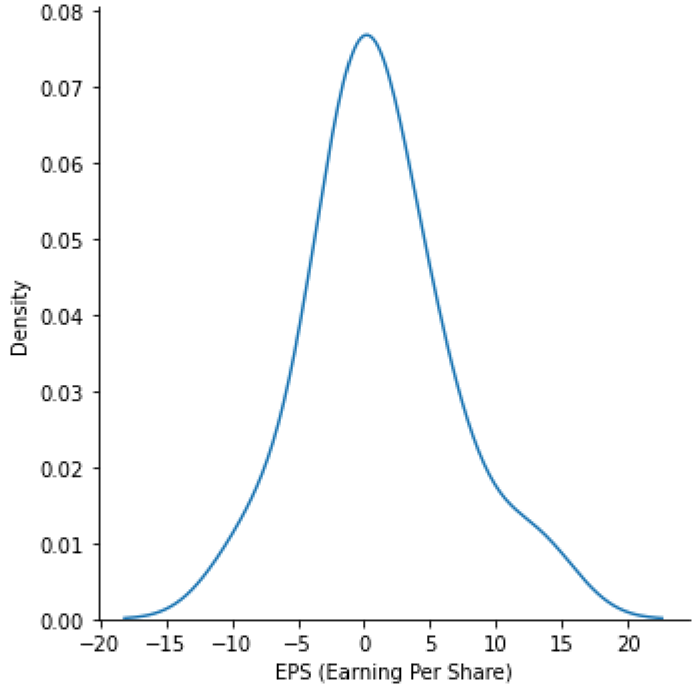
In [56]:
```
# Create a random forest regressor
model = RandomForestRegressor(n_estimators= 100 , random_state=42)
```

In [57]:
```
model = RandomForestRegressor(n_estimators= 100 , random_state=42)

# Fit the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Print the predicted values
print("Predicted values:", y_pred)
```

```
Predicted values: [12.8602 15.4445 16.0847  3.6695 22.0441  8.8111  2.1985  6.2898  1.2322
 18.5063  5.5302 19.5081]
```

In [58]:
```
#TO see if the prediction is crct we will compare it with the truth value. truth value = y_test
import seaborn as sns
sns.displot(y_pred-y_test, kind ='kde')
```

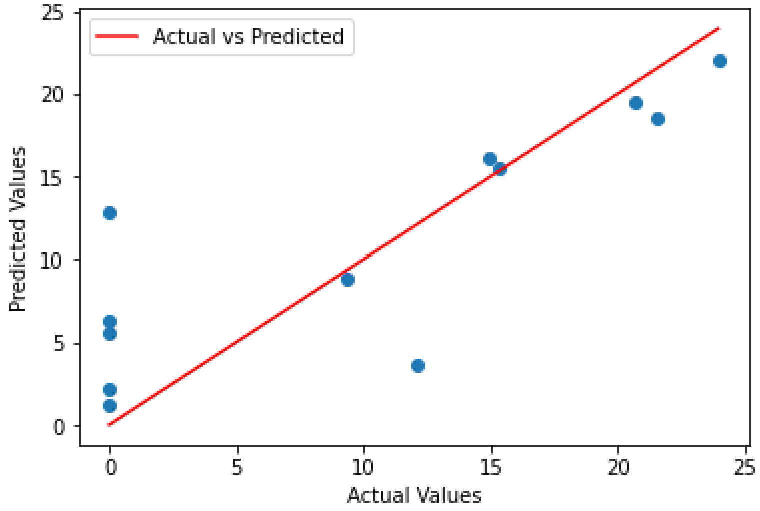
Out[58]:
```
<seaborn.axisgrid.FacetGrid at 0x1bea1e78bb0>
```



In [59]:
```
from sklearn.metrics import mean_squared_error, r2_score

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)

# Calculate R-squared score
r2 = r2_score(y_test, y_pred)
print("R-squared Score:", r2)
```

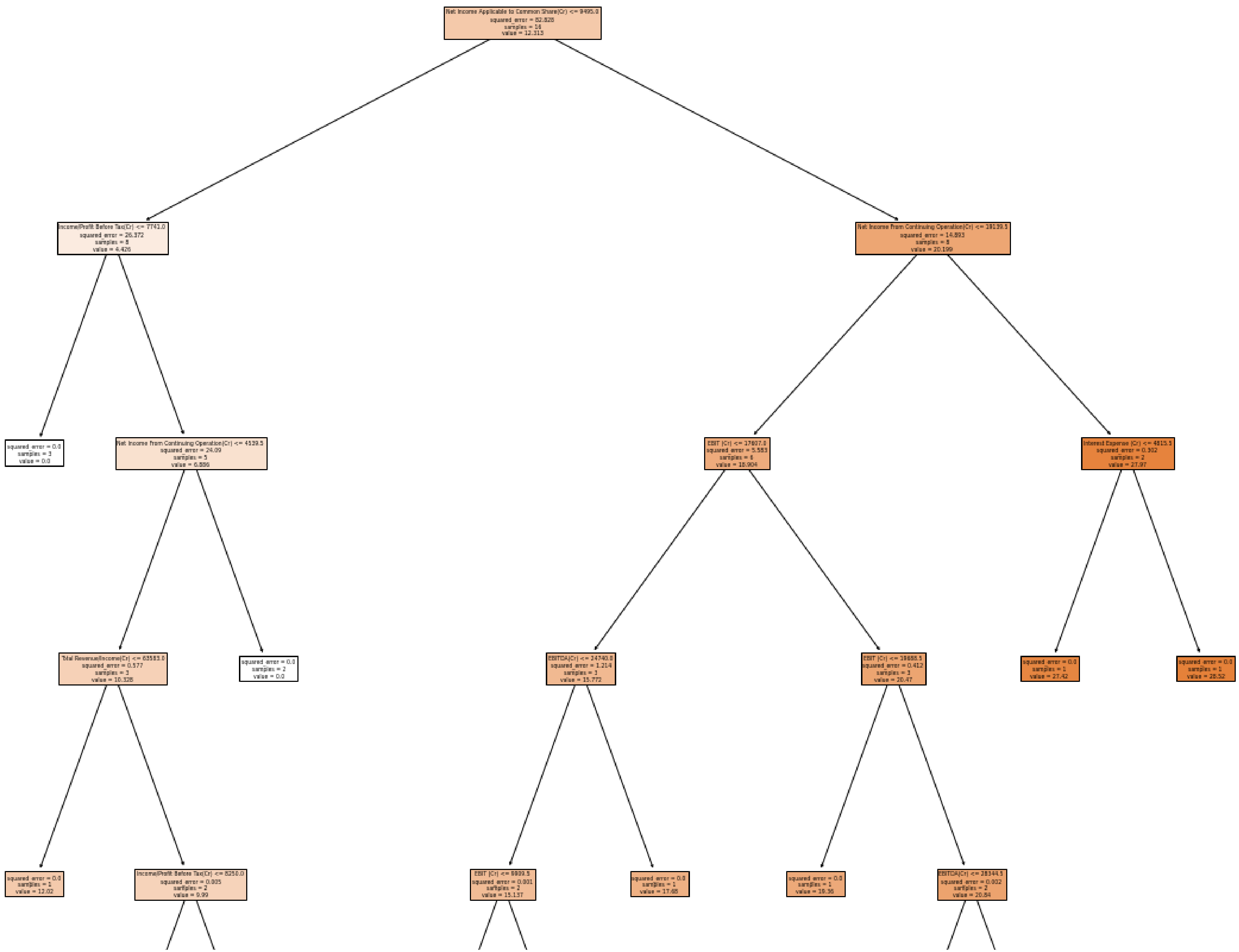
```
Mean Squared Error (MSE): 27.40622092666672
R-squared Score: 0.6710432056271378
```

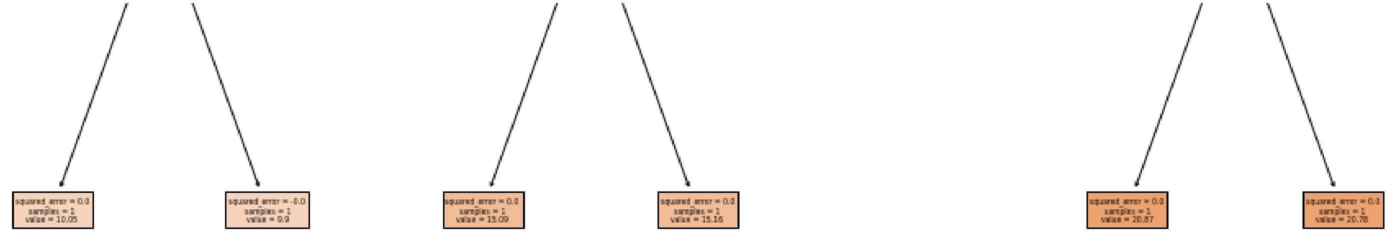
In [60]:
```
# Plotting actual vs predicted values
plt.scatter(y_test, y_pred)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'r', label='Actual vs Predicted')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.legend()
plt.show()
```



In [61]:

```
from sklearn.tree import plot_tree
# Create a Random Forest Regressor
model = RandomForestRegressor(n_estimators= 50  , random_state=42)
model.fit(X_train, y_train)

# Extract individual decision trees from the random forest
estimators = model.estimators_

# Plot the first decision tree in the ensemble
plt.figure(figsize=(20, 20))
plot_tree(estimators[0], feature_names=X.columns, filled=True)
plt.show()
```

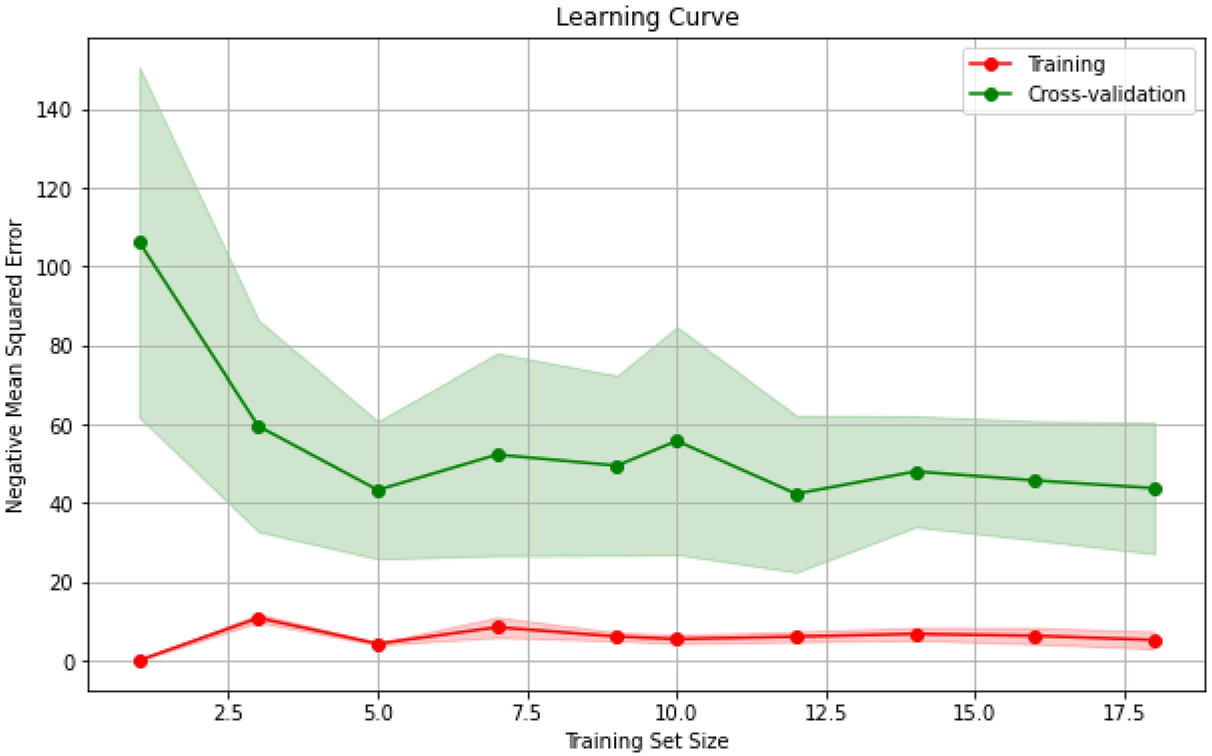

Net Income Applicable to Common Share(Cr) <= 9495.0
squared_error = 82.828
samples = 16
value = 12.313
Income/Profit Before Tax(Cr) <= 7741.0
squared_error = 26.372
samples = 8
value = 4.426
Net Income From Continuing Operation(Cr) <= 18139.5
squared_error = 14.893
samples = 8
value = 20.199
squared_error = 0.0
samples = 3
value = 0.0
Net Income From Continuing Operation(Cr) <= 4539.5
squared_error = 24.09
samples = 5
value = 6.886
EBIT (Cr) <= 17807.0
squared_error = 5.583
samples = 6
value = 18.904
Interest Expense (Cr) <= 4815.5
squared_error = 0.302
samples = 2
value = 27.97
Total Revenue/Income(Cr) <= 63583.0
squared_error = 0.577
samples = 3
value = 10.328
squared_error = 0.0
samples = 2
value = 0.0
EBITDA(Cr) <= 24740.0
squared_error = 1.214
samples = 3
value = 15.772
EBIT (Cr) <= 19688.5
squared_error = 0.412
samples = 3
value = 20.47
squared_error = 0.0
samples = 1
value = 27.42
squared_error = 0.0
samples = 1
value = 28.52
squared_error = 0.0
samples = 1
value = 12.02
Income/Profit Before Tax(Cr) <= 8250.0
squared_error = 0.005
samples = 2
value = 9.99
EBIT (Cr) <= 6909.5
squared_error = 0.001
samples = 2
value = 15.137
squared_error = 0.0
samples = 1
value = 17.68
squared_error = 0.0
samples = 1
value = 19.36
EBITDA(Cr) <= 28344.5
squared_error = 0.002
samples = 2
value = 20.84



In [66]:

```
from sklearn.model_selection import learning_curve

def plot_learning_curve(estimator, X, y, train_sizes, cv):
    train_sizes, train_scores, test_scores = learning_curve(estimator, X, y, train_sizes=train_sizes, cv=cv, scoring='neg_mean_sc

    # Calculate the mean and standard deviation of training scores
    train_mean = -np.mean(train_scores, axis=1)
    train_std = np.std(train_scores, axis=1)

    # Calculate the mean and standard deviation of test scores
    test_mean = -np.mean(test_scores, axis=1)
    test_std = np.std(test_scores, axis=1)

    # Plot the learning curve
    plt.figure(figsize=(10, 6))
    plt.plot(train_sizes, train_mean, 'o-', color='r', label='Training')
    plt.plot(train_sizes, test_mean, 'o-', color='g', label='Cross-validation')
    plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color='r')
    plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color='g')
    plt.xlabel('Training Set Size')
    plt.ylabel('Negative Mean Squared Error')
    plt.title('Learning Curve')
    plt.legend(loc='best')
    plt.grid(True)
    plt.show()

# Define your estimator/model
model = RandomForestRegressor(n_estimators= 28 , random_state=42)

# Specify the training set sizes
train_sizes = np.linspace(0.1, 1.0, 10)
```

```
# Plot the learning curve
plot_learning_curve(model, X_train, y_train, train_sizes, cv=3)
```



In [63]:
```
print("Predicted values:", y_pred)
```

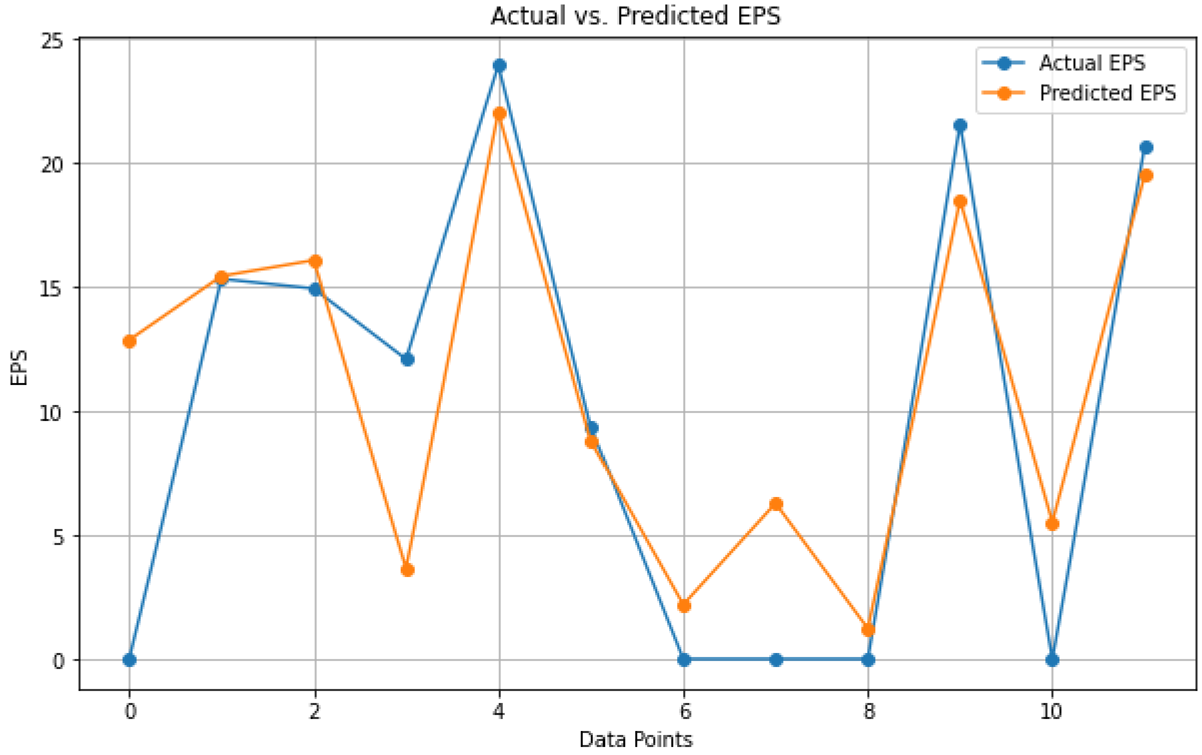
```
Predicted values: [12.8602 15.4445 16.0847  3.6695 22.0441  8.8111  2.1985  6.2898  1.2322
 18.5063  5.5302 19.5081]
```

In [65]:
```
# Assuming y_test and y_pred are numpy arrays
y_test = y_test
y_pred = y_pred

# Print the actual and predicted values for the "gross_profit" column
print("Actual EPS:")
print(y_test.values)

print("Predicted EPS:")
print(y_pred)
```

```
Actual EPS:
[ 0.    15.33 14.94 12.09 23.95  9.39  0.    0.    0.    21.58  0.    20.67]
Predicted EPS:
[12.8602 15.4445 16.0847  3.6695 22.0441  8.8111  2.1985  6.2898  1.2322
 18.5063  5.5302 19.5081]
```

In [68]:

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


# Plotting the graph
plt.figure(figsize=(10, 6))

# Data points for x-axis (e.g., assuming 100 data points)
data_points = range(len(y_test))

# Plot actual EPS
plt.plot(data_points, y_test, label='Actual EPS', marker='o')

# Plot predicted EPS
plt.plot(data_points, y_pred, label='Predicted EPS', marker='o')

# Add labels and legend
plt.xlabel('Data Points')
plt.ylabel('EPS')
plt.title('Actual vs. Predicted EPS')
plt.legend()
plt.grid(True)

# Show the plot
plt.show()
```



Actual vs. Predicted EPS
Actual EPS
Predicted EPS
EPS
Data Points
0
5
10
15
20
25
0
2
4
6
8
10

In [ ]: