Oasis Infobyte Internship

Intern Name-Akshay Anandkar

Task 5-SALES PREDCITION USING PYTHON

Sales prediction means predicting how much of a product people will buy based on factors such as the amount you spend to advertise your product, the segment of people you advertise for, or the platform you are advertising on about your product.

Typically, a product and service-based business always need their Data Scientist to predict their future sales with every step they take to manipulate the cost of advertising their product. So let’s start the task of sales prediction with machine learning using Python.

In [1]:
```
#import all required liabraries
import pandas as pd
import numpy as np
```

In [2]:
```
#importing sales dataset
sales=pd.read_csv(r"D:\Data-Science-Internship\Advertising.csv")
sales.head()
```

Out[2]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [3]:
```
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  200 non-null    int64  
 1   TV          200 non-null    float64
 2   Radio       200 non-null    float64
 3   Newspaper   200 non-null    float64
 4   Sales       200 non-null    float64
dtypes: float64(4), int64(1)
memory usage: 7.9 KB
```

In [4]:
```
#droping Unnamed: 0 column
sales.drop(columns=['Unnamed: 0'],inplace=True)
```

In [5]:
```
sales.head()
```

Out[5]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

In [6]:
```
sales.isnull().sum()
```

Out[6]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

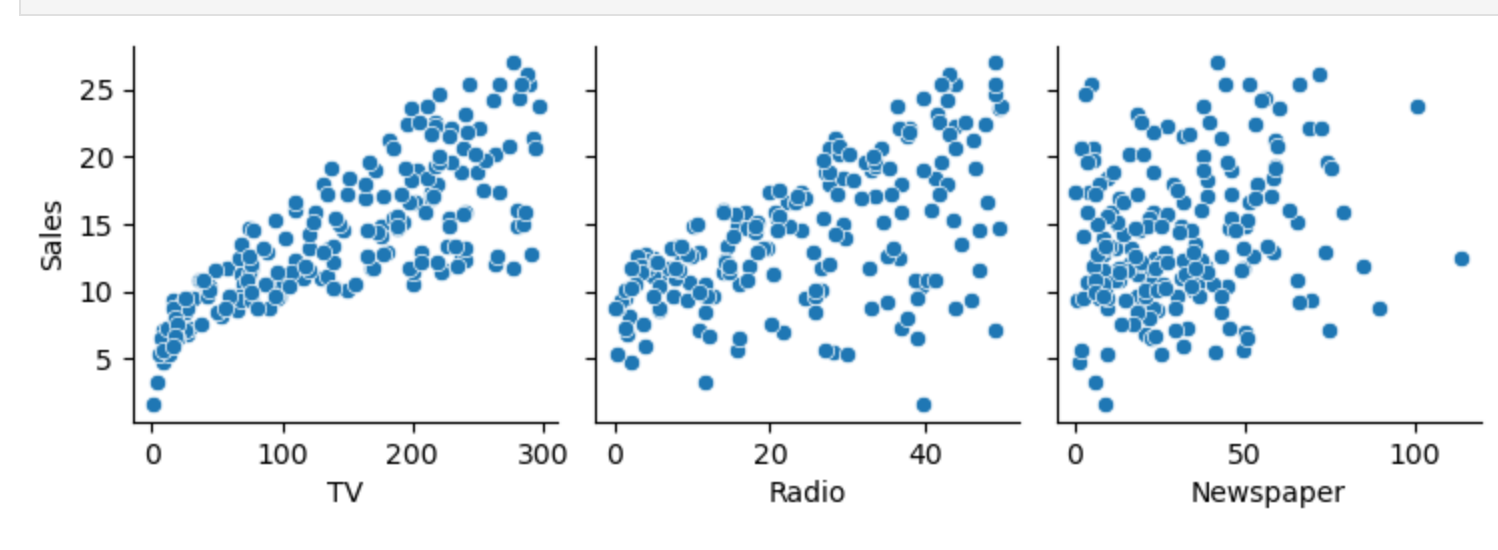
In [7]:
```
sales.describe()
```

Out[7]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	14.022500
std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max	296.400000	49.600000	114.000000	27.000000

In [8]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

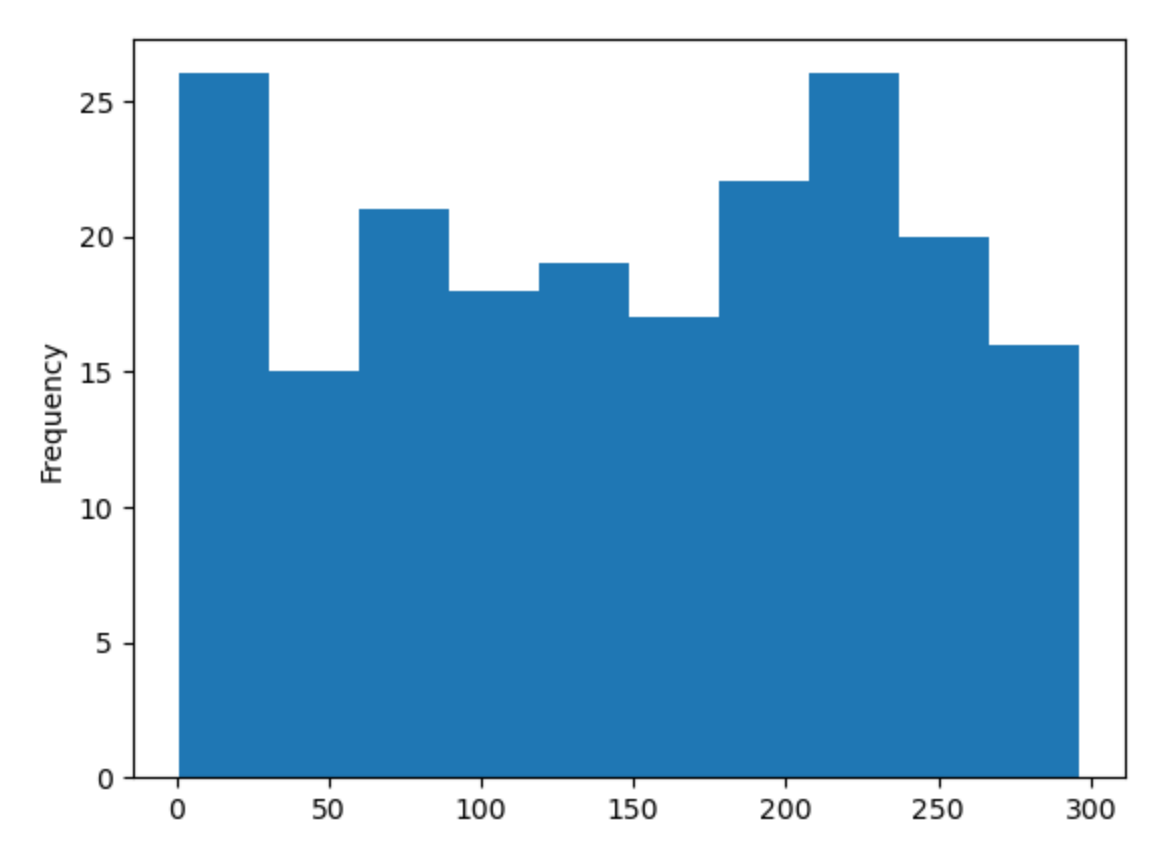
In [9]:
```
#pairplot
sns.pairplot(sales,x_vars=['TV','Radio','Newspaper'],y_vars='Sales',kind='scatter')
plt.show()
```



from pairplot we can see when we are advertising cost on TV then sales is increasing but for Radio and Newspaper its unpredictable

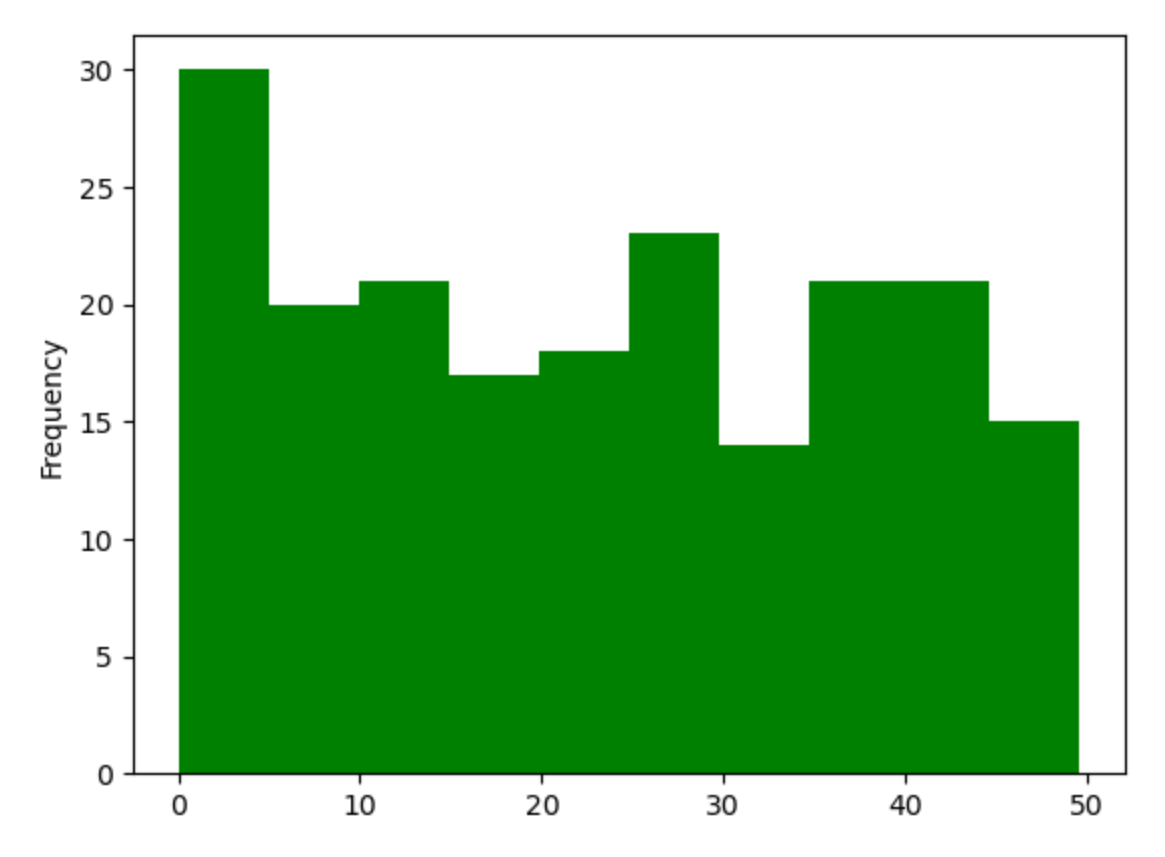
In [10]:
```
sales['TV'].plot.hist(bins=10)
```

Out[10]:
```
<Axes: ylabel='Frequency'>
```



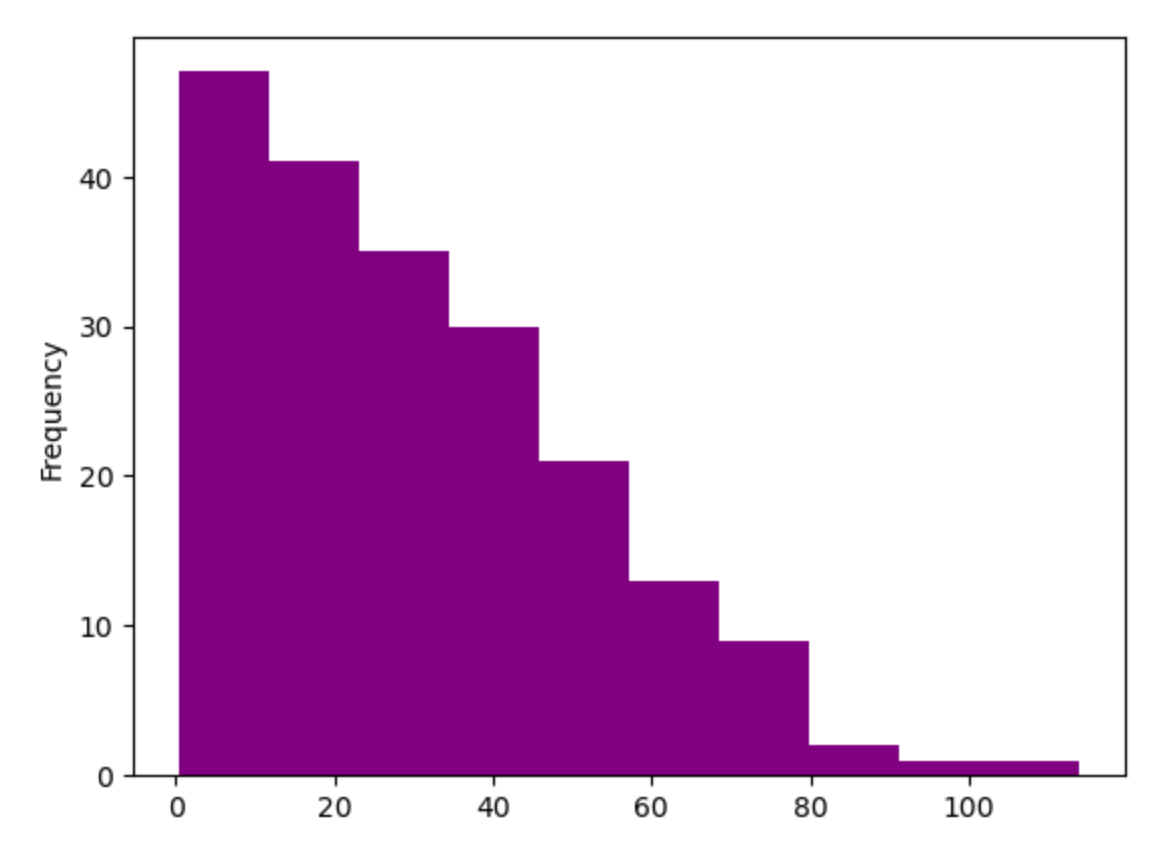
In [11]:
```
sales['Radio'].plot.hist(bins=10,color='green',xlabel='Radio')
```

Out[11]:
```
<Axes: ylabel='Frequency'>
```



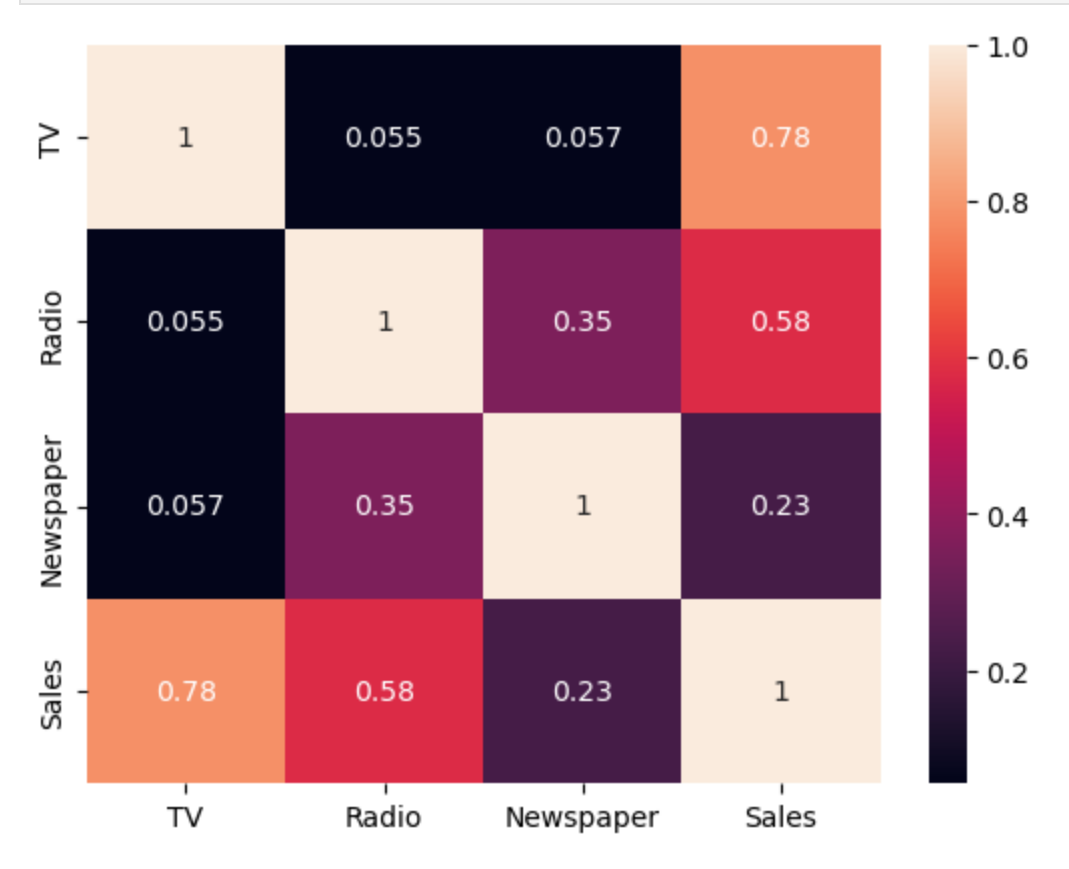
In [12]:
```
sales['Newspaper'].plot.hist(bins=10,color='purple',xlabel='Newspaper')
```

Out[12]:
```
<Axes: ylabel='Frequency'>
```



we can see low advertisement cost in newpaper

In [13]:
```
# heatmap
sns.heatmap(sales.corr(),annot=True)
plt.show()
```



here sales is highly correlated with TV

In [14]:
```
#BUILDING MODEL
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(sales[['TV']],sales[['Sales']],test_size=0.2,random_state=23)
```

In [15]:
```
print(X_train)
```

```
        TV
47   239.9
10    66.1
172   19.6
44    25.1
128  220.3
..     ...
91    28.6
31   112.9
182   56.2
40   202.5
83    68.4

[160 rows x 1 columns]
```

In [16]:
```
print(y_train)
```

```
     Sales
47    23.2
10     8.6
172    7.6
44     8.5
128   24.7
..     ...
91     7.3
31    11.9
182    8.7
40    16.6
83    13.6

[160 rows x 1 columns]
```

In [17]:
```
print(X_test)
```

```
        TV
72    26.8
30   292.9
188  286.0
77   120.5
14   204.1
119   19.4
97   184.9
19   147.3
198  283.6
174  222.4
185  205.0
75    16.9
46    89.7
81   239.8
191   75.5
199  232.1
157  149.8
24    62.3
92   217.7
140   73.4
175  276.9
52   216.4
163  163.5
117   76.4
132    8.4
101  296.4
58   210.8
154  187.8
51   100.4
82    75.3
131  265.2
18    69.2
167  206.8
107   90.4
141  193.7
102  280.2
8      8.6
171  164.5
20   218.4
105  137.9
```

In [18]:
```
print(y_test)
```

```
     Sales
72     8.8
30    21.4
188   15.9
77    14.2
14    19.0
119    6.6
97    15.5
19    14.6
198   25.5
174   11.5
185   22.6
75     8.7
46    10.6
81    12.3
191    9.9
199   13.4
157   10.1
24     9.7
92    19.4
140   10.9
175   27.0
52    22.6
163   18.0
117    9.4
132    5.7
101   23.8
58    23.8
154   15.6
51    10.7
82    11.3
131   12.7
18    11.3
167   12.2
107    8.7
141   19.2
102   14.8
8      4.8
171   14.5
20    18.0
105   19.2
```

In [19]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
```

Out[19]:
```
▾ LinearRegression
LinearRegression()
```

In [20]:
```
y_pred=lr.predict(X_test)
print(y_pred)
```

```
[[ 8.36369717]
 [20.90476329]
 [20.57957217]
 [12.77969828]
 [16.719695  ]
 [ 8.01494148]
 [15.81481537]
 [14.04275943]
 [20.46646222]
 [17.5821584 ]
 [16.76211124]
 [ 7.89711861]
 [11.32812054]
 [18.40220557]
 [10.65888665]
 [18.03931113]
 [14.1605823 ]
 [10.03678191]
 [17.36065141]
 [10.55991544]
 [20.15069693]
 [17.29938352]
 [14.80625162]
 [10.70130288]
 [ 7.49652086]
 [21.06971531]
 [17.03546029]
 [15.9514899 ]
 [11.83240242]
 [10.64946082]
 [19.59928591]
 [10.36197302]
 [16.8469437 ]
 [11.36111095]
 [16.22955187]
 [20.30622312]
 [ 7.50594669]
 [14.85338077]
 [17.39364181]
 [13.59974545]]
```

In [21]:
```
lr.coef_
```

Out[21]:
```
array([[0.04712915]])
```

In [22]:
```
lr.intercept_
```

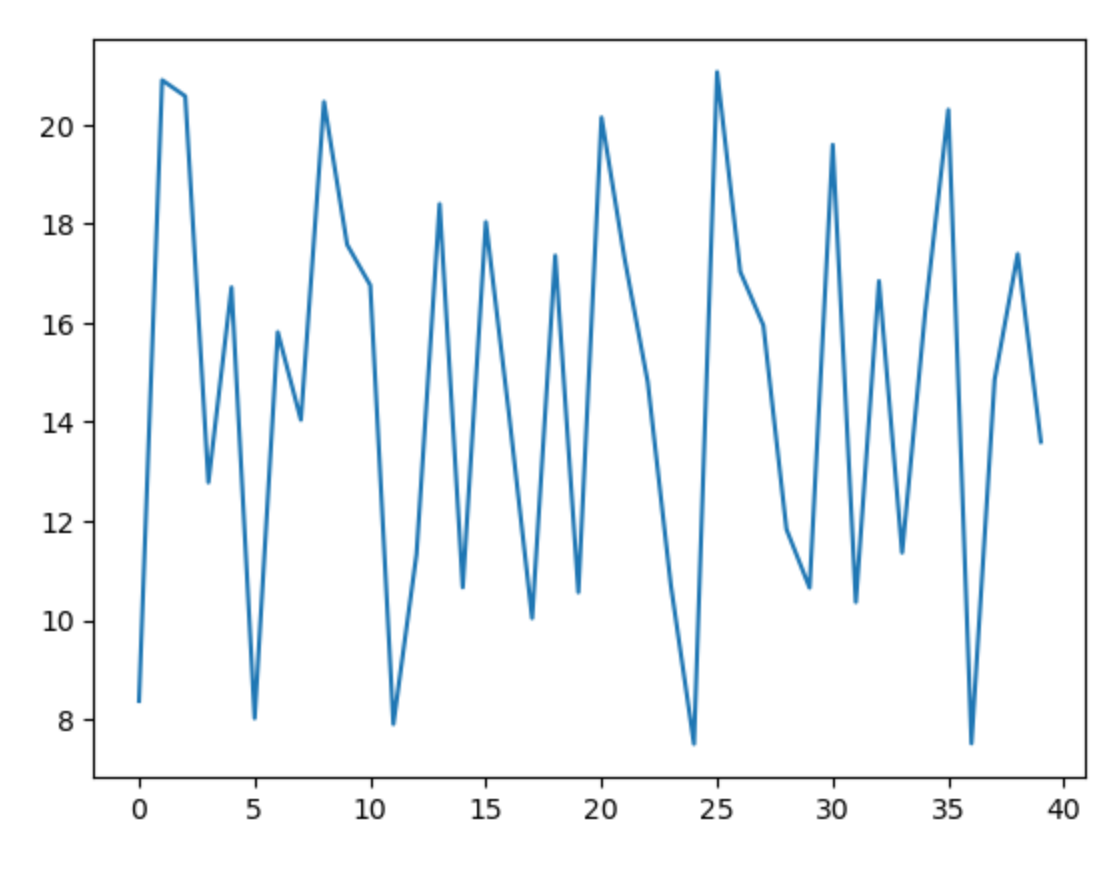
Out[22]:
```
array([7.10063602])
```

In [23]:
```
0.04712915*26.8+7.10063602
```

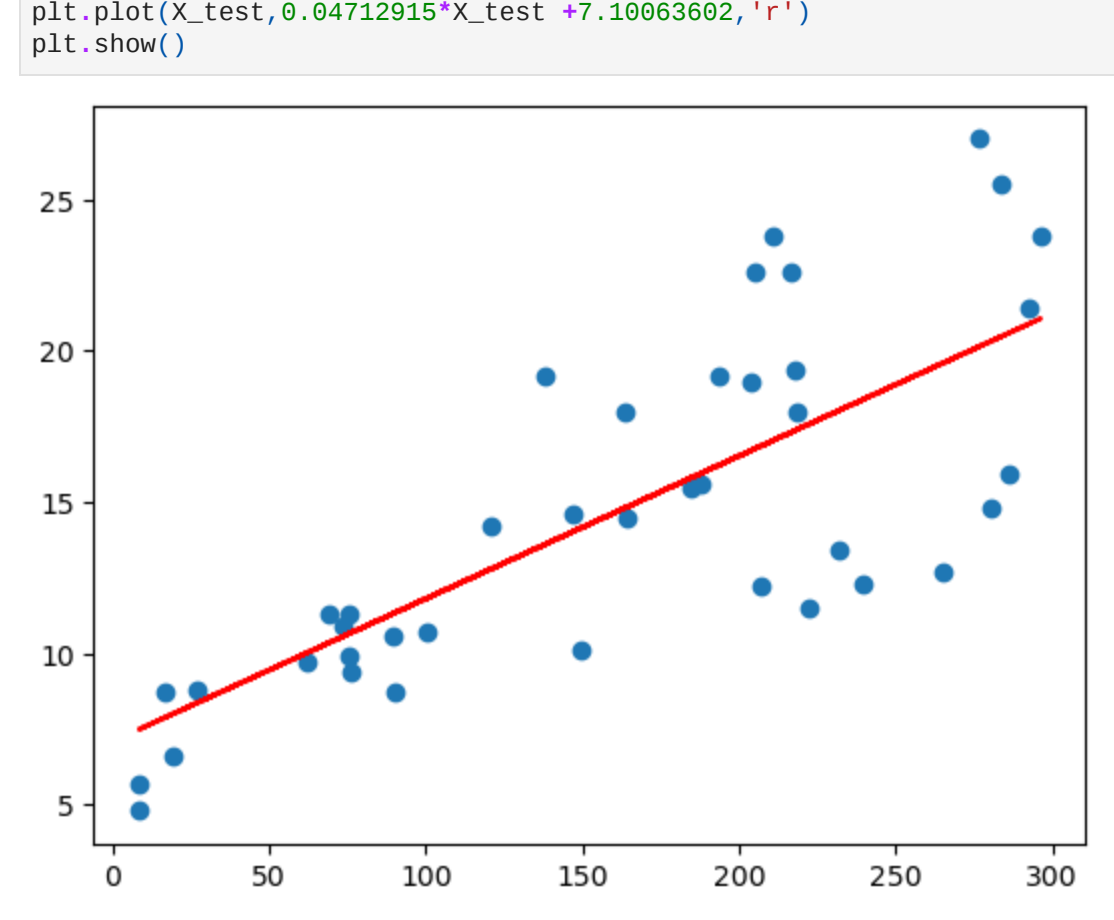
Out[23]:
```
8.36369724
```

In [24]:
```
plt.plot(y_pred)
```

Out[24]:
```
[<matplotlib.lines.Line2D at 0x2d4aea05ea0>]
```



In [25]:
```
plt.scatter(X_test,y_test)
plt.plot(X_test,0.04712915*X_test +7.10063602,'r')
plt.show()
```



Conclusion- Above model is successful for predcition

In [ ]: