Sales Project

September 27, 2023

[143]:
```
# Importing dependencies

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import seaborn as sns
import matplotlib.pyplot as plt
```

[144]:
```
data= pd.read_csv('advertising.csv')
```

[145]:
```
data.head()
```

[145]:

	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	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

[146]:
```
data.tail()
```

[146]:

	TV	Radio	Newspaper	Sales
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

[147]:
```
data.shape
```

[147]: (200, 4)

This means there are 200 rows and 4 columns

[148]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

As the content shows that there are no null values so we can proceed with our dataset

[149]:
```
data.describe()
# This will provide the description of our dataset
```

[149]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

[150]:
```
data.isnull().sum()
```

[150]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

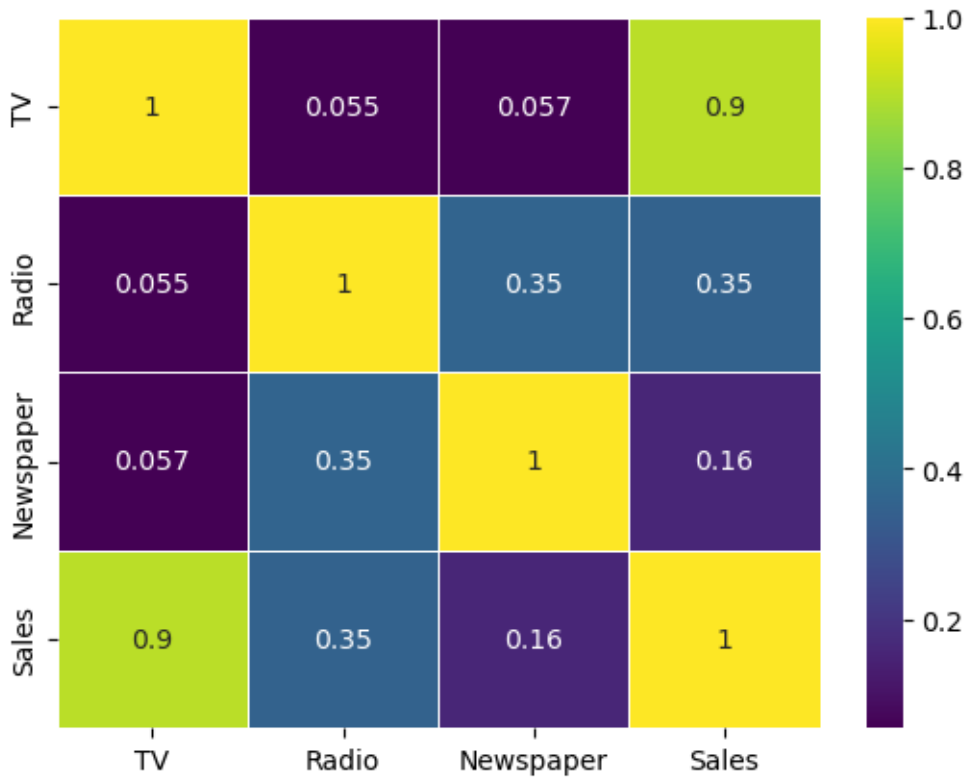
[151]:
```
data.corr()
```

[151]:

	TV	Radio	Newspaper	Sales
TV	1.000000	0.054809	0.056648	0.901208
Radio	0.054809	1.000000	0.354104	0.349631
Newspaper	0.056648	0.354104	1.000000	0.157960
Sales	0.901208	0.349631	0.157960	1.000000

The great aspect of the pandas module is corr() method. The corr() method calculated the realtionship between each column in your data set. df.corr()

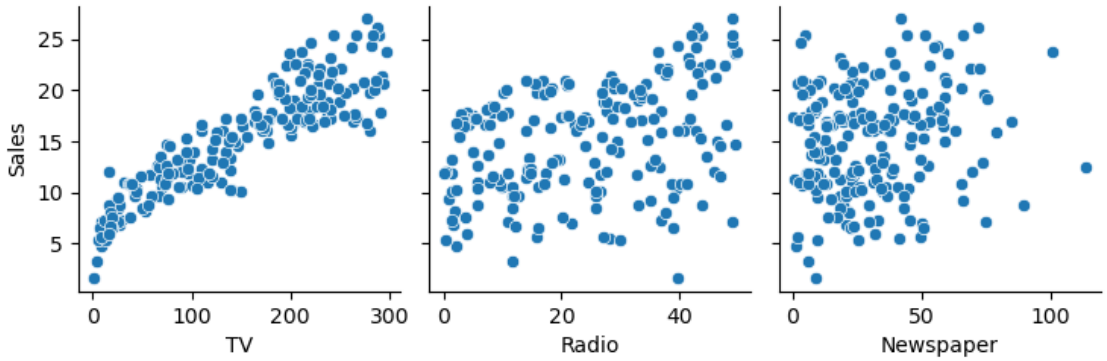
[152]:
```
sns.heatmap(data.corr(),annot=True,cmap='viridis',linewidths=0.5)
plt.show()
```



Heatmap is use as grahical image to process the data correlation within each other, annote is used to describe the values within the graph

##Hence the graph shows that the Sales are highly co-related to TV

[153]:
```
sns.
  ↪pairplot(data,x_vars=['TV','Radio','Newspaper'],y_vars='Sales',kind='scatter')
plt.show()
```

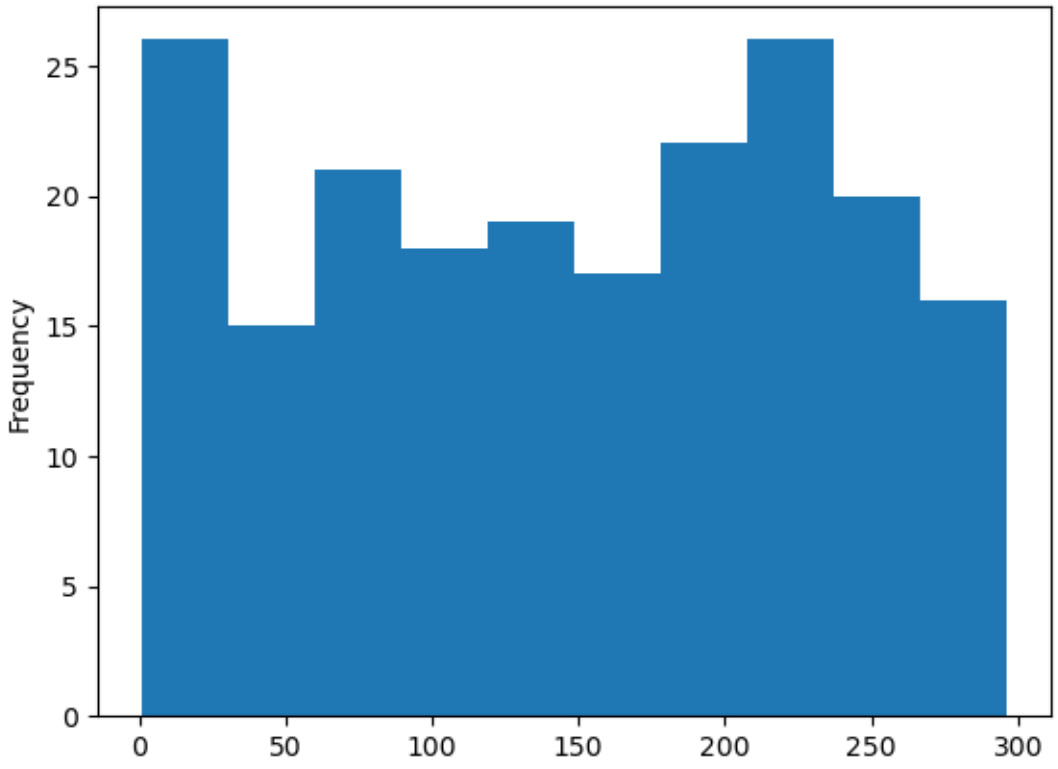


This graph shows the pair of Tv, Radio ,Newspaper which corresponds to x and sales to y

This graph describes that when advertising cost increases in TV ads then sales will increase while Radio and Newspaper are not predictable

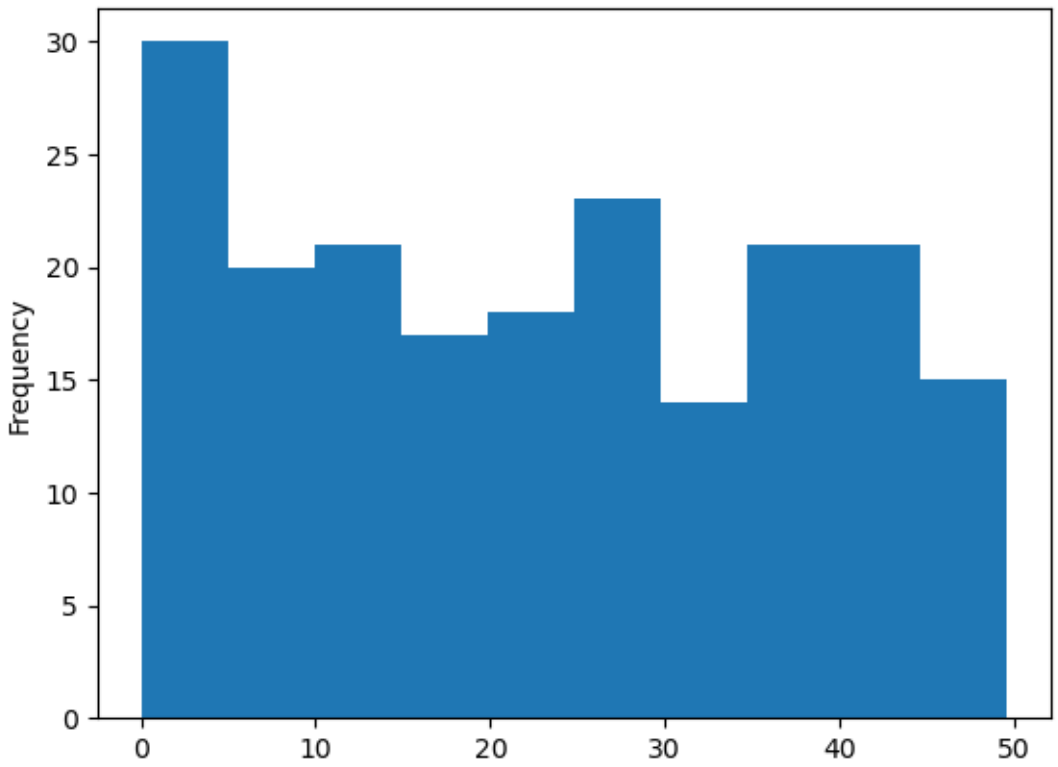
[154]:
```
data['TV'].plot.hist(xlabel='TV')
```

[154]:
```
<Axes: ylabel='Frequency'>
```



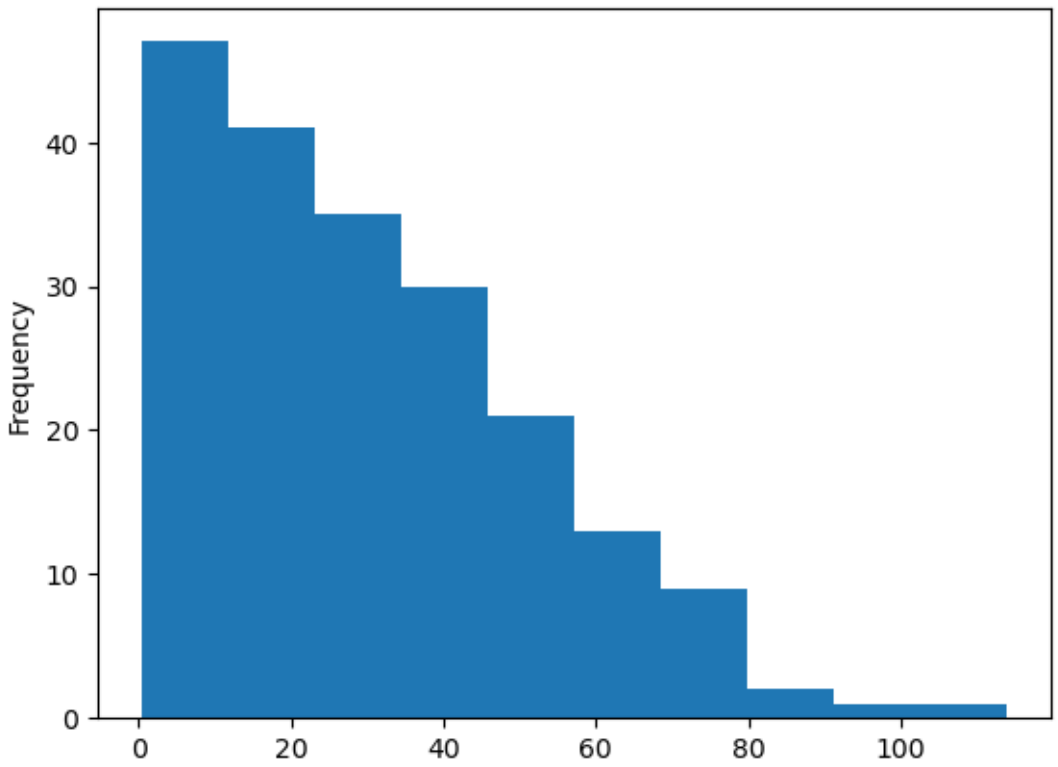
[155]:
```
data['Radio'].plot.hist()
```

[155]:
```
<Axes: ylabel='Frequency'>
```

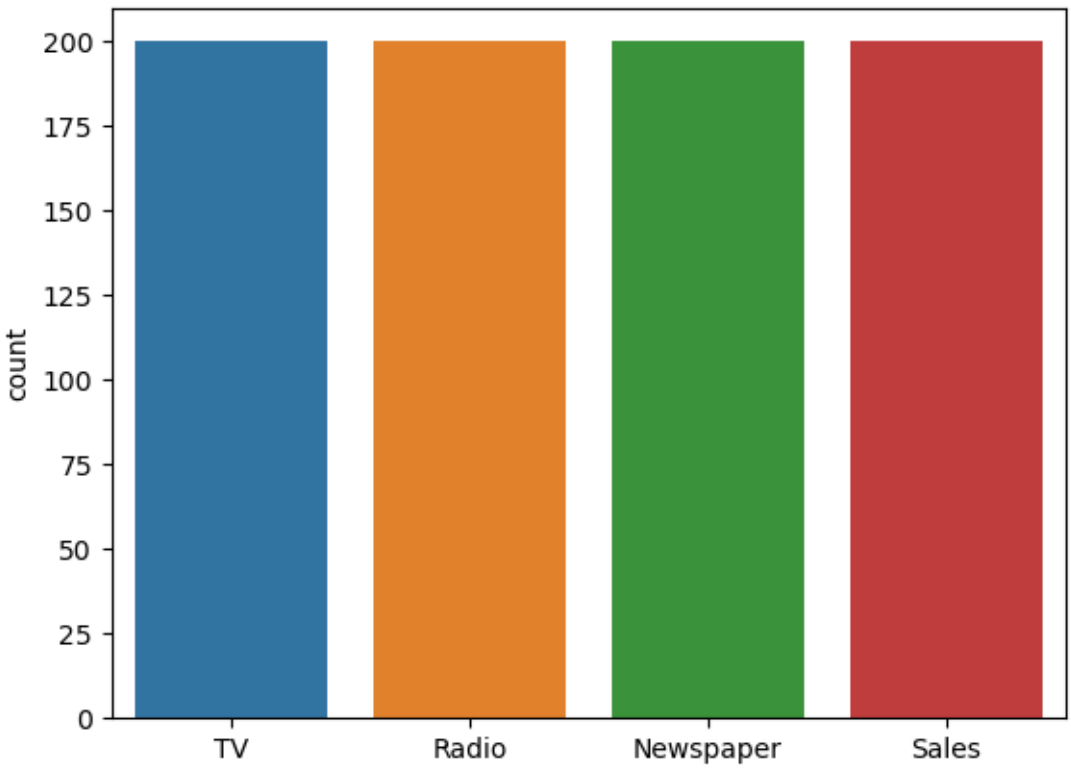


[156]:
```
data['Newspaper'].plot.hist()
```

[156]:
```
<Axes: ylabel='Frequency'>
```



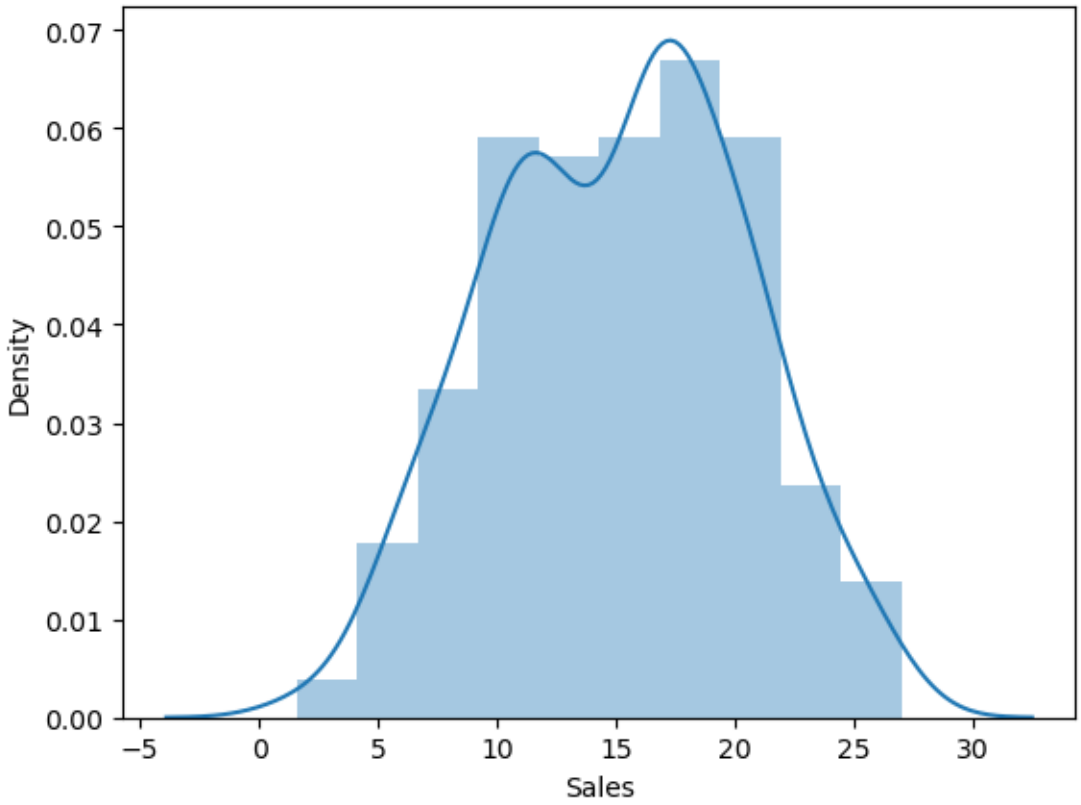
[157]:
```
sns.countplot(data)
plt.show()
```



[158]:
```
sns.distplot(data['Sales'],kde=True)
```

```
C:\Users\morea\AppData\Local\Temp\ipykernel_3336\3371230658.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['Sales'],kde=True)
```

[158]:
```
<Axes: xlabel='Sales', ylabel='Density'>
```



[159]:
```
x=data.drop(columns=['Sales'],axis=1)
y=data['Sales']
```

[160]:
```
x
```

[160]:
```
        TV  Radio  Newspaper
0    230.1   37.8       69.2
1     44.5   39.3       45.1
2     17.2   45.9       69.3
3    151.5   41.3       58.5
4    180.8   10.8       58.4
..     …     …          …
195   38.2    3.7       13.8
196   94.2    4.9        8.1
197  177.0    9.3        6.4
198  283.6   42.0       66.2
199  232.1    8.6        8.7

[200 rows x 3 columns]
```

[161]:
```
y
```

[161]:
```
0      22.1
1      10.4
2      12.0
3      16.5
4      17.9
       …
195     7.6
196    14.0
197    14.8
198    25.5
199    18.4
Name: Sales, Length: 200, dtype: float64
```

[162]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

using the x_train ytrain spliting the data for training and testing

[163]:
```
clf=LinearRegression()
```

[164]:
```
clf
```

[164]:
```
LinearRegression()
```

[165]:
```
clf.fit(x_train,y_train)
```

[165]:
```
LinearRegression()
```

[166]:
```
y_pred=clf.predict(x_test)
```

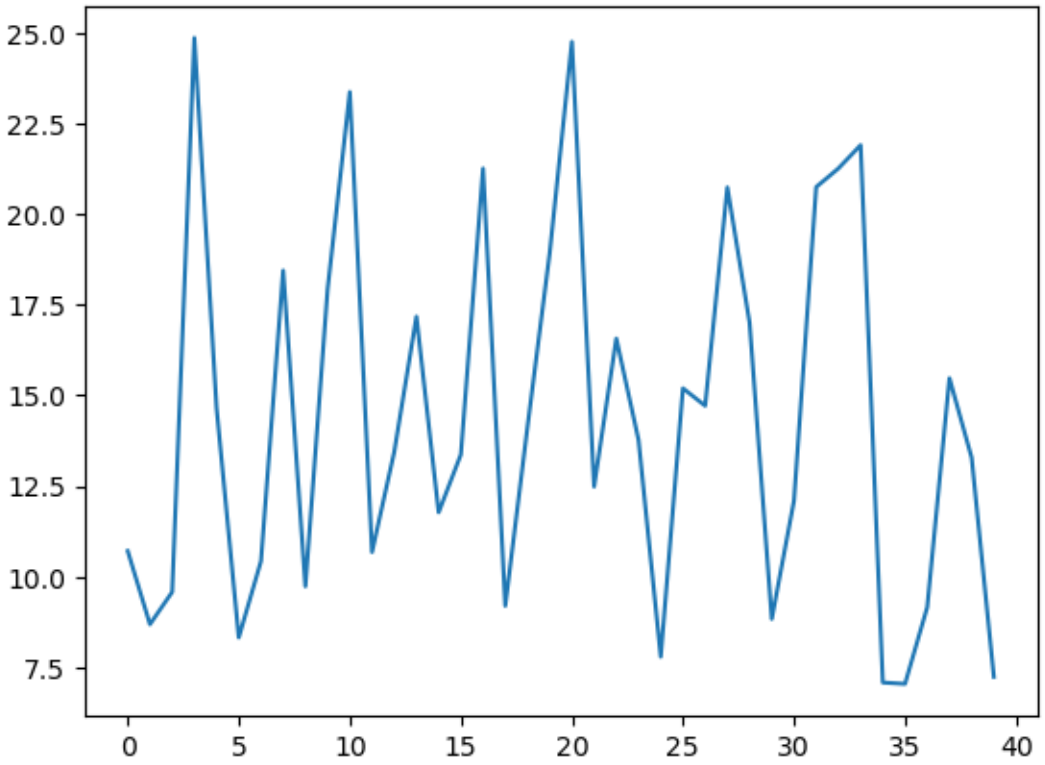
[167]:
```
from sklearn.metrics import accuracy_score
```

[168]:
```
y_pred
```

[168]:
```
array([10.70988945,  8.68629773,  9.5778695 , 24.86207988, 14.65584473,
        8.3214275 , 10.43048002, 18.4453765 ,  9.73394291, 17.9290798 ,
       23.369886  , 10.67916356, 13.44032325, 17.17416235, 11.77380187,
       13.37072678, 21.26009906,  9.19666875, 14.13201846, 18.9260716 ,
       24.75507991, 12.48481182, 16.57130583, 13.77344772,  7.79299106,
       15.19648316, 14.71607944, 20.73862119, 17.01041859,  8.83474391,
       12.09424377, 20.74886454, 21.26147987, 21.90420095,  7.08087067,
        7.04431681,  9.15949871, 15.47796148, 13.28282334,  7.23769883])
```

[169]:
```
plt.plot(y_pred)
```

[169]:
```
[<matplotlib.lines.Line2D at 0x2a1a4b5e490>]
```



[170]:
```
clf.coef_
```

[170]:
```
array([ 0.05368006,  0.11152624, -0.00351166])
```

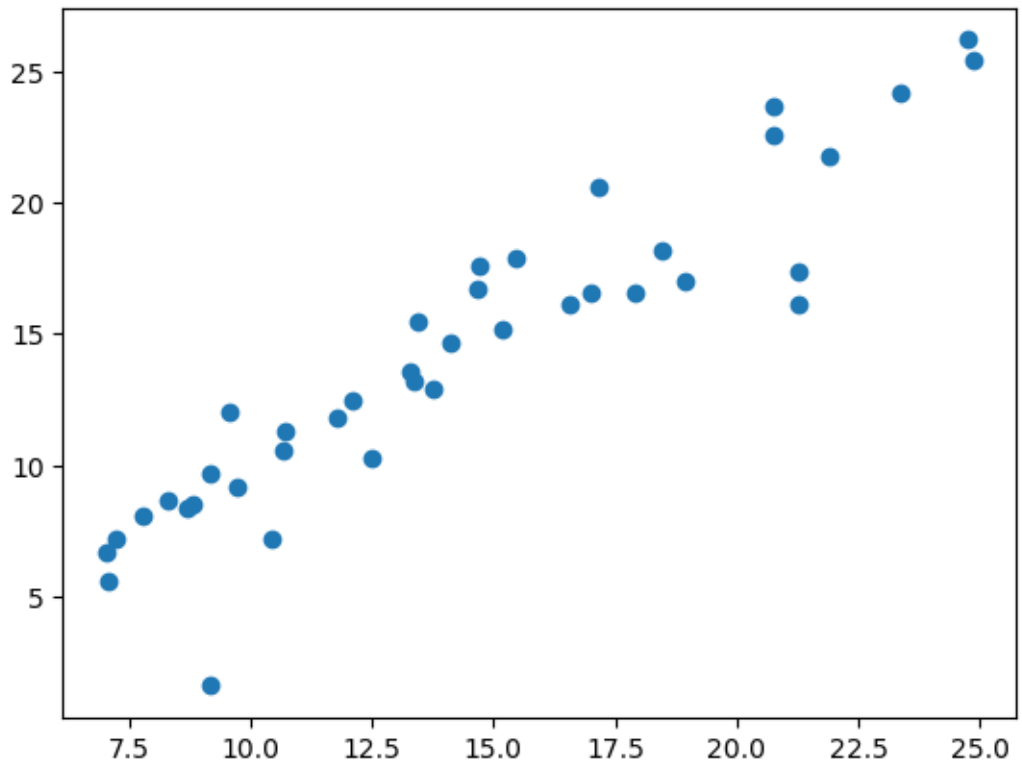
[171]:
```
clf.intercept_
```

[171]:
```
4.773205203269832
```

[172]:
```
0.05368006*69.2+4.7732
# The value of first value of newspaper is 69.2 so the value is
# not same but quite close
```

[172]:
```
8.487860152
```

[173]:
```
plt.scatter(y_pred,y_test)
plt.show()
```



1 # Performing Random Forest Regression

Random forest regression is used to solve business problems where a company needs to predict a continuous value. For example, predicting future prices, revenue, or comparing performance.

[174]:
```
rf_model = RandomForestRegressor(random_state=42)
```

[175]:
```
rf_model.fit(x_train, y_train)
```

[175]:
```
RandomForestRegressor(random_state=42)
```

[176]:
```
y_pred = rf_model.predict(x_test)
```

[177]:
```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

[178]:
```
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r2}")
```

```
Mean Absolute Error (MAE): 0.9202999999999989
Mean Squared Error (MSE): 1.8357477500000041
```

```
R-squared (R²): 0.9450014305787326
```

mean_absolute_error (MAE): It measures the average absolute difference between the predicted and actual values. Lower values indicate better performance.

mean_squared_error (MSE): It measures the average squared difference between predicted and actual values. Lower values indicate better performance.

r2_score (R-squared or coefficient of determination): It measures how well the model explains the variance in the target variable. A higher R-squared value indicates a better fit of the model to the data.

[ ]:

3.03K
Unit Sales Total

6.11K
Sales by Newspaper

4.65K
Sales by Radio

29.41K
Sales by TV

of TV	Sum of Radio	Sum of Newspaper	Sales
0.70	39.60	8.70	1.60
4.10	11.60	5.70	3.20
8.60	2.10	1.00	4.80
18.50	30.30	35.00	5.30
7.30	28.10	41.40	5.50
13.20	15.90	49.60	5.60
8.40	27.20	2.10	5.70
17.20	4.10	31.60	5.90
27.20	54.90	72.90	6.60
18.70	12.10	23.40	6.70
27.50	1.60	20.70	6.90
18.80	21.70	50.40	7.00
33.70	59.90	104.70	7.20
40.30	38.40	78.20	7.30
57.80	23.80	30.80	7.60
17.90	37.60	21.60	8.00
53.50	2.00	21.40	8.10
50.00	11.60	18.40	8.40
25.10	25.70	43.30	8.50
73.10	49.40	119.10	8.70
26.80	33.00	19.30	8.80
22.80	25.10	65.00	9.20
08.50	**4,652.80**	**6,110.80**