```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Part 2

```
df = pd.read_csv('data2.csv')
```

```
df
```

	6	148	72	35	0	33.6	0.627	50	1
0	1	85	66	29	0	26.6	0.351	31	0
1	8	183	64	0	0	23.3	0.672	32	1
2	1	89	66	23	94	28.1	0.167	21	0
3	0	137	40	35	168	43.1	2.288	33	1
4	5	116	74	0	0	25.6	0.201	30	0
...	...	...	...	...	...	...	...	...	...
762	10	101	76	48	180	32.9	0.171	63	0
763	2	122	70	27	0	36.8	0.340	27	0
764	5	121	72	23	112	26.2	0.245	30	0
765	1	126	60	0	0	30.1	0.349	47	1
766	1	93	70	31	0	30.4	0.315	23	0

767 rows × 9 columns

Q1 Split the data for training and testing in the ratio of 80:20.

```
X = df.iloc[:,:-1].values
y = df.iloc[:,-1].values
```

```
X
```

```
array([[1.00e+00, 8.50e+01, 6.60e+01, ..., 2.66e+01, 3.51e-01, 3.10e+01],
       [8.00e+00, 1.83e+02, 6.40e+01, ..., 2.33e+01, 6.72e-01, 3.20e+01],
       [1.00e+00, 8.90e+01, 6.60e+01, ..., 2.81e+01, 1.67e-01, 2.10e+01],
       ...,
       [5.00e+00, 1.21e+02, 7.20e+01, ..., 2.62e+01, 2.45e-01, 3.00e+01],
       [1.00e+00, 1.26e+02, 6.00e+01, ..., 3.01e+01, 3.49e-01, 4.70e+01],
       [1.00e+00, 9.30e+01, 7.00e+01, ..., 3.04e+01, 3.15e-01, 2.30e+01]])
```

```
y
```

```
array([0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0])
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state= 101,test_siz
```

```
X_train
```

```
array([[2.00e+00, 9.30e+01, 6.40e+01, ..., 3.80e+01, 6.74e-01, 2.30e+01],
       [0.00e+00, 9.40e+01, 0.00e+00, ..., 0.00e+00, 2.56e-01, 2.50e+01],
       [0.00e+00, 1.00e+02, 7.00e+01, ..., 3.08e+01, 5.97e-01, 2.10e+01],
       ...,
       [6.00e+00, 1.08e+02, 4.40e+01, ..., 2.40e+01, 8.13e-01, 3.50e+01],
       [9.00e+00, 1.52e+02, 7.80e+01, ..., 3.42e+01, 8.93e-01, 3.30e+01],
       [3.00e+00, 1.25e+02, 5.80e+01, ..., 3.16e+01, 1.51e-01, 2.40e+01]])
```

```
X_test
```

```
array([[  1.   , 126.   ,  60.   , ...,  30.1  ,   0.349,  47.   ],
       [  3.   , 187.   ,  70.   , ...,  36.4  ,   0.408,  36.   ],
       [  9.   , 171.   , 110.   , ...,  45.4  ,   0.721,  54.   ],
       ...,
       [  2.   ,  87.   ,   0.   , ...,  28.9  ,   0.773,  25.   ],
       [ 10.   ,  68.   , 106.   , ...,  35.5  ,   0.285,  47.   ],
       [  8.   , 112.   ,  72.   , ...,  23.6  ,   0.84 ,  58.   ]])
```

Q2 Rescale the distribution of values so that the mean of observed values is 0 and the standard deviation is 1.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

```
X_test
```

```
array([[-0.82662315,  0.0838599 , -0.36895431, ..., -0.28670804,
        -0.35209348,  1.11491365],
       [-0.2779164 ,  1.92017663,  0.09525291, ...,  0.63650517,
        -0.16348828,  0.19585588],
       [ 1.36820383,  1.43851978,  1.95208177, ...,  1.95538118,
         0.83707828,  1.6997686 ],
       ...,
       [-0.55226977, -1.09017867, -3.15419759, ..., -0.46255817,
         1.00330659, -0.7232019 ],
       [ 1.6425572 , -1.66214617,  1.76639888, ...,  0.50461757,
        -0.55668217,  1.11491365],
       [ 1.09385046, -0.33758984,  0.18809435, ..., -1.2392296 ,
         1.21748537,  2.03397143]])
```

```
y_test
```

```
array([1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0])
```

Q3 Develop a KNN classifier model and predict for the test data

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors= 5)
knn.fit(X_train,y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
y_pred = knn.predict(X_test)
y_pred
```

```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0])
```

```
y_test
```

```
array([1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0])
```

Q4 Draw up the confusion matrix.

```
from sklearn.metrics import accuracy_score,confusion_matrix,mean_squared_error
acc = accuracy_score(y_test,y_pred)
acc
```

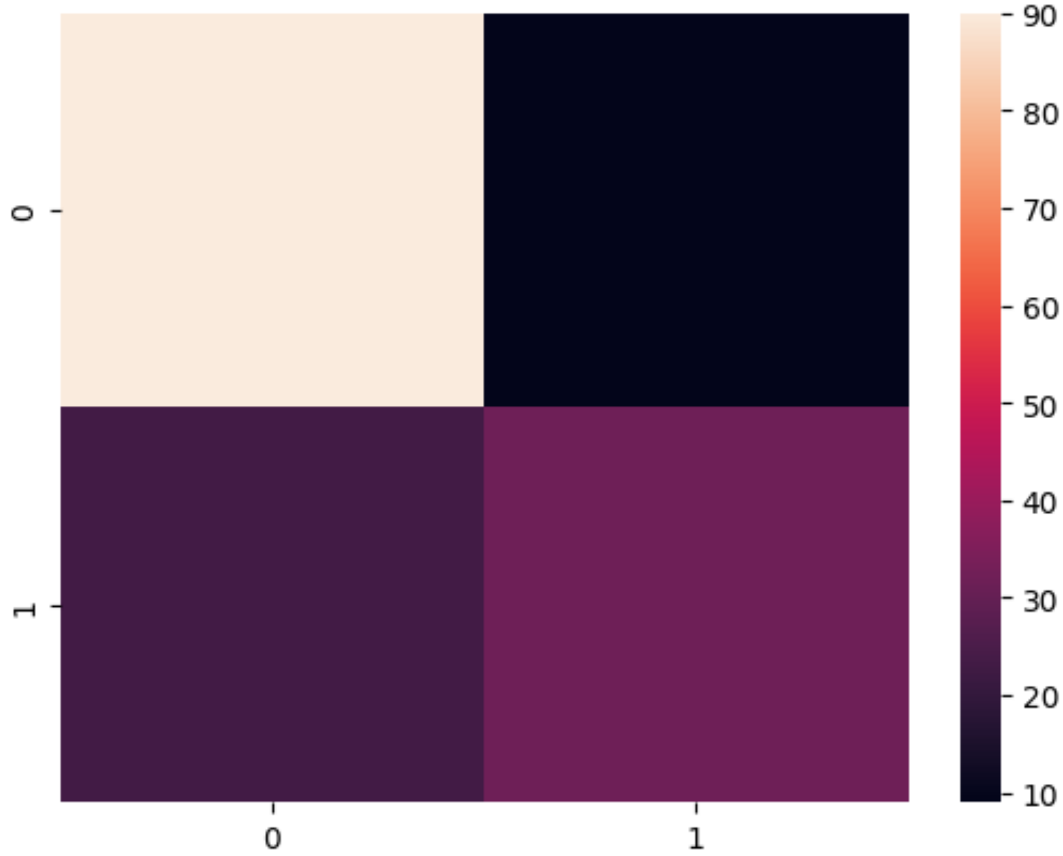
```
0.7922077922077922
```

```
cm = confusion_matrix(y_test,y_pred)
cm
```

```
array([[90,  9],
       [23, 32]])
```

```
sns.heatmap(cm)
```

```
<Axes: >
```



Q5 Identify an optimum k value based on minimum mean errors (consider a range of 20). Draw a corresponding graph between Mean error and k-value.

```
mse = mean_squared_error(y_test,y_pred)
mse
```

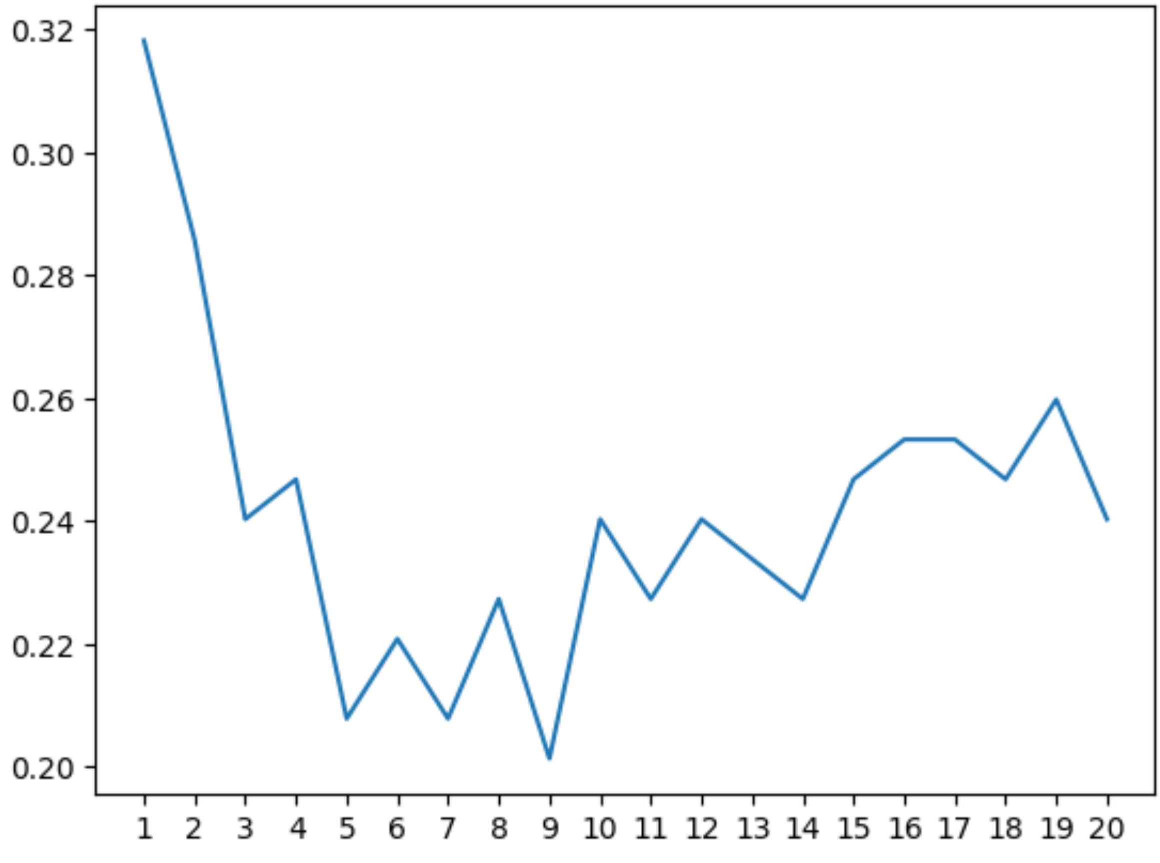
```
0.2077922077922078
```

```
mean_error = []
acc_score = []
k_values = [j for j in range(1,21)]
for i in k_values:
    model = KNeighborsClassifier(n_neighbors = i)
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test,y_pred)
    acc = accuracy_score(y_test,y_pred)
    acc_score.append(acc)
    mean_error.append(mse)
mean_error = np.array(mean_error)
```

```
mean_error
```

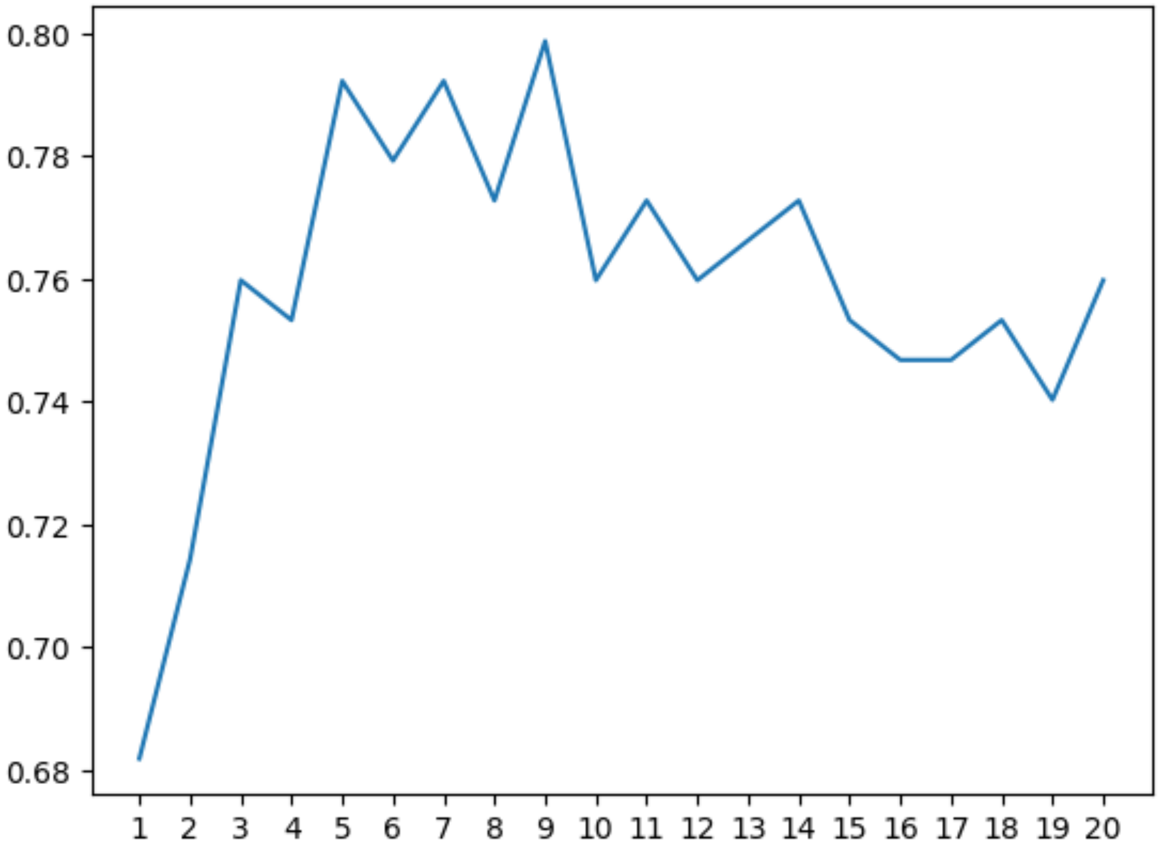
```
array([0.31818182, 0.28571429, 0.24025974, 0.24675325, 0.20779221,
       0.22077922, 0.20779221, 0.22727273, 0.2012987 , 0.24025974,
       0.22727273, 0.24025974, 0.23376623, 0.22727273, 0.24675325,
       0.25324675, 0.25324675, 0.24675325, 0.25974026, 0.24025974])
```

```
plt.plot(k_values,mean_error)
plt.xticks(range(1,21))
plt.show()
```



From the graph we see that the error is minimum when k is 9

```
plt.plot(k_values,acc_score)
plt.xticks(range(1,21))
plt.show()
```



Using accuracy_score we see that the k value is somewhere close to 10 like 9

Part 1

Q1. Draw (a) a scatter plot of money spent on TV advertisements versus Sales (b) Pair plots and Heatmap.

```
cf = pd.read_csv('data1.csv')
cf
```

	Unnamed: 0.1	Unnamed: 0	TV	Radio	Newspaper	Sales
0	0	1	230.1	37.8	69.2	22.1
1	1	2	44.5	39.3	45.1	10.4
2	2	3	17.2	45.9	69.3	9.3
3	3	4	151.5	41.3	58.5	18.5
4	4	5	180.8	10.8	58.4	12.9
...	...	...	...	...	...	...
195	195	196	38.2	3.7	13.8	7.6
196	196	197	94.2	4.9	8.1	9.7
197	197	198	177.0	9.3	6.4	12.8
198	198	199	283.6	42.0	66.2	25.5
199	199	200	232.1	8.6	8.7	13.4

200 rows × 6 columns

```
cf = cf.drop(columns= ['Unnamed: 0','Unnamed: 0.1'])
```

```
cf
```

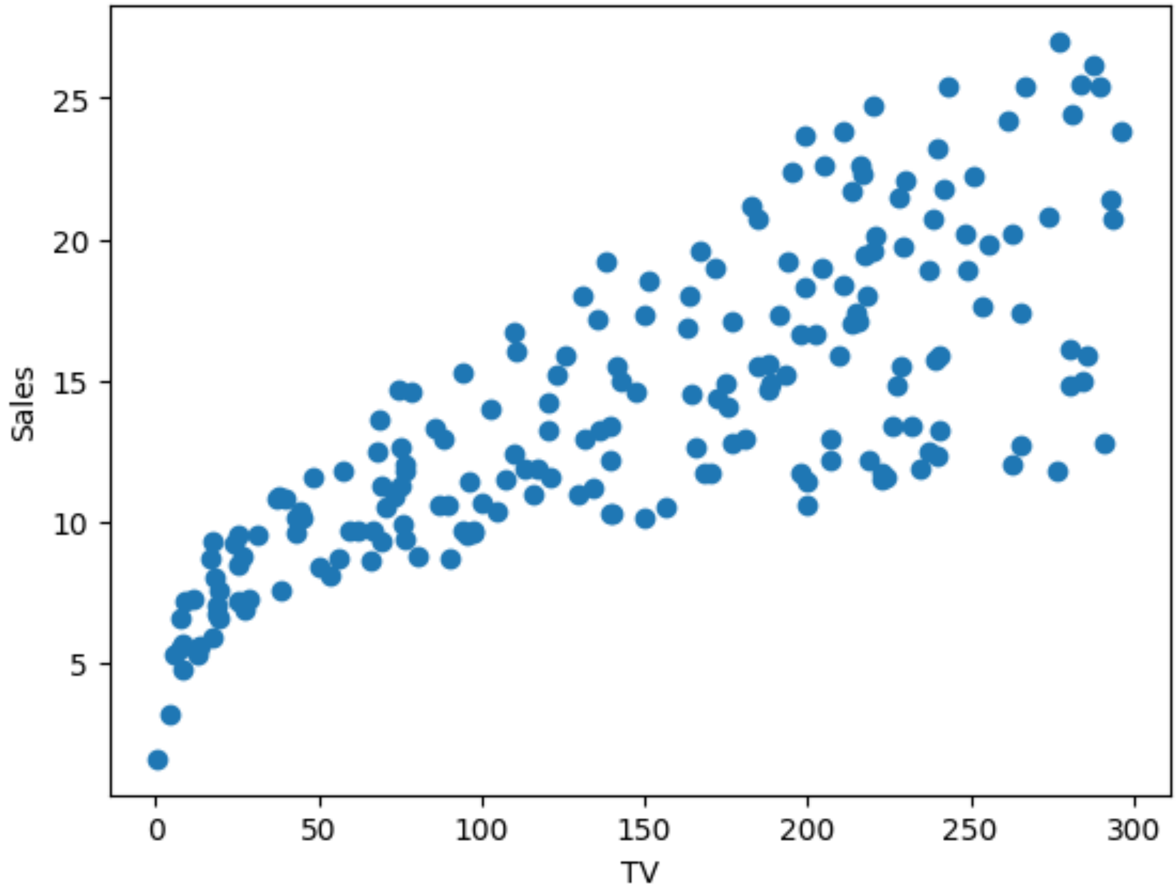
	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

200 rows × 4 columns

```
cf['TV']
```

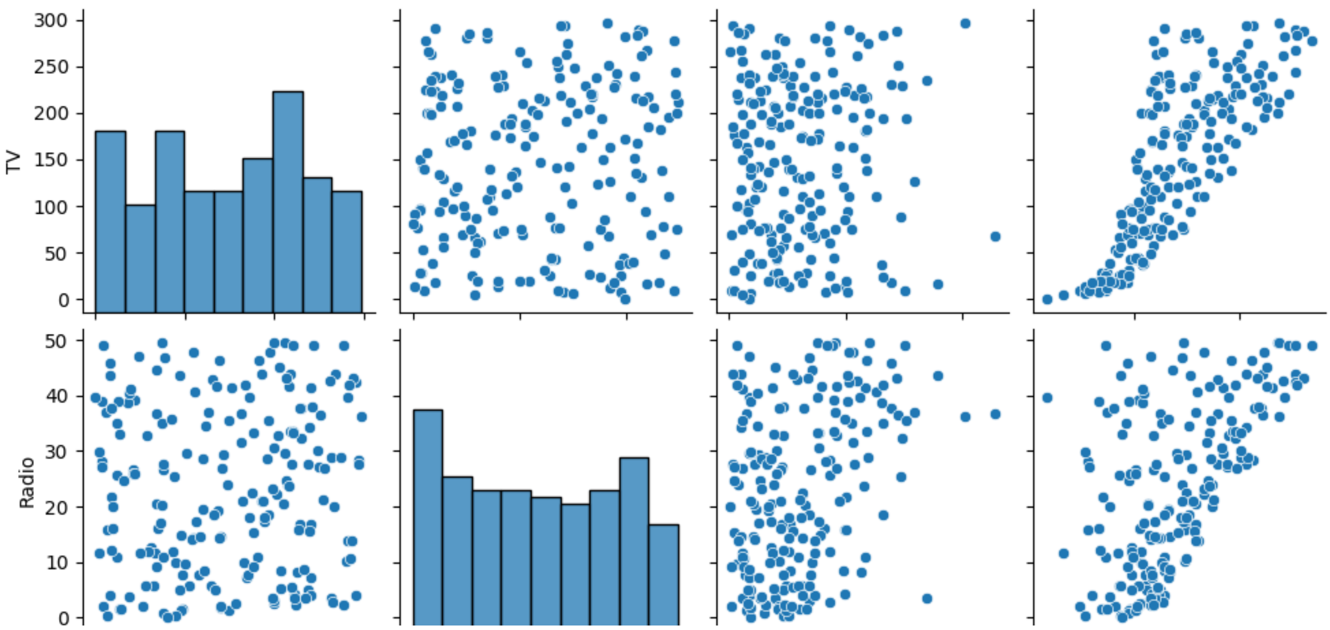
```
0      230.1
1       44.5
2       17.2
3      151.5
4      180.8
       ...
195     38.2
196     94.2
197    177.0
198    283.6
199    232.1
Name: TV, Length: 200, dtype: float64
```

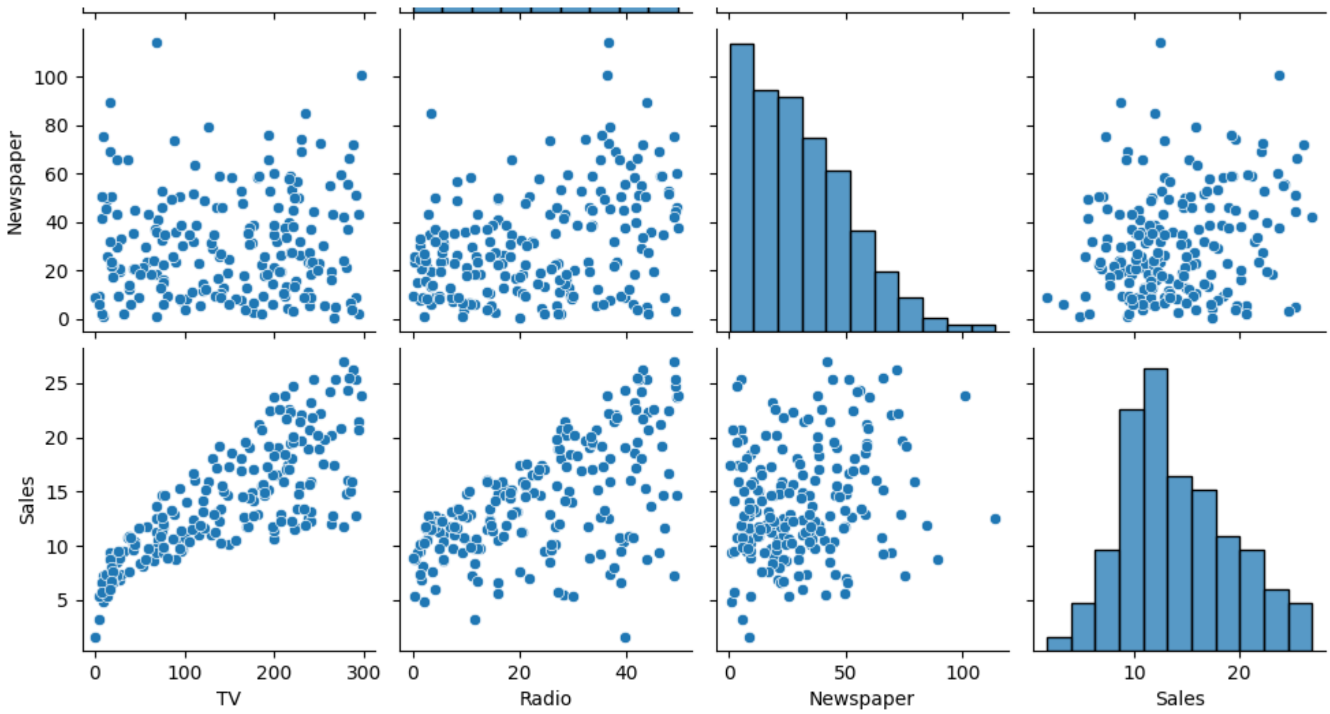
```
plt.scatter(cf['TV'],cf['Sales'])
plt.xlabel("TV")
plt.ylabel('Sales')
plt.show()
```




```
sns.pairplot(cf)
```

<seaborn.axisgrid.PairGrid at 0x1467edf30>

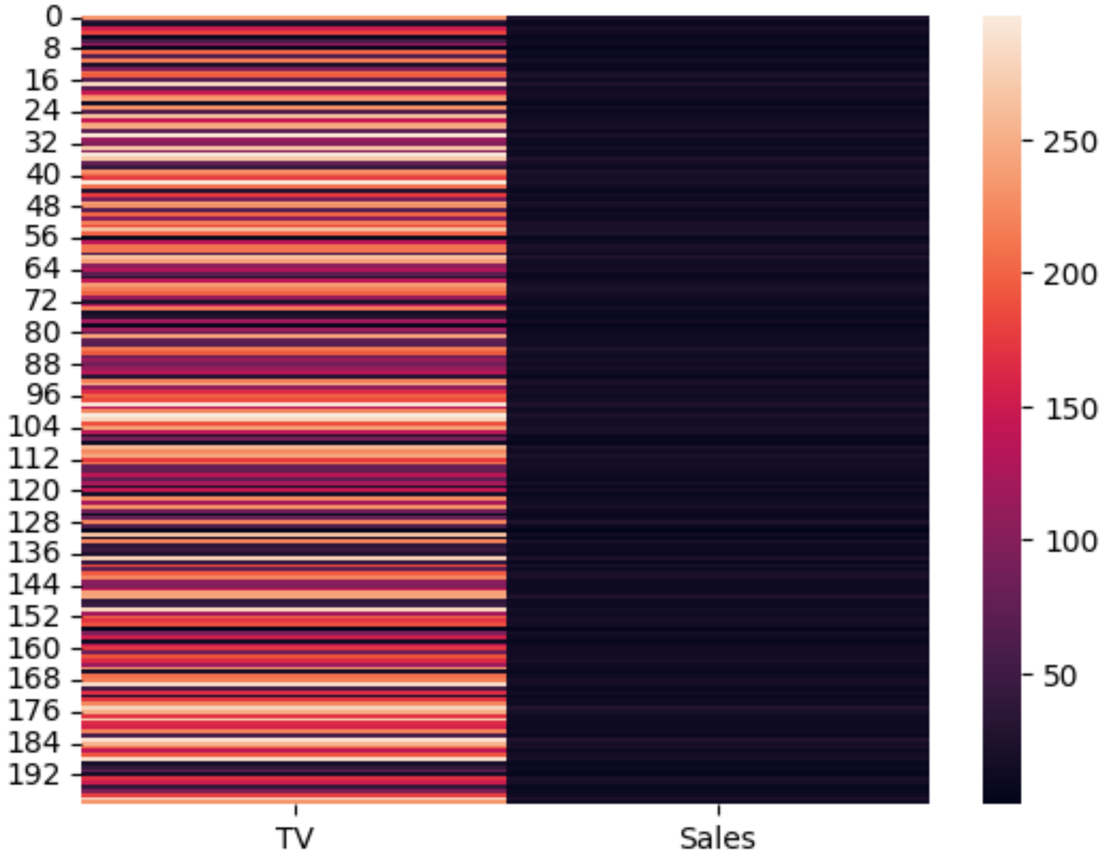





```
new_df = cf.drop(columns= ['Radio','Newspaper'])
```

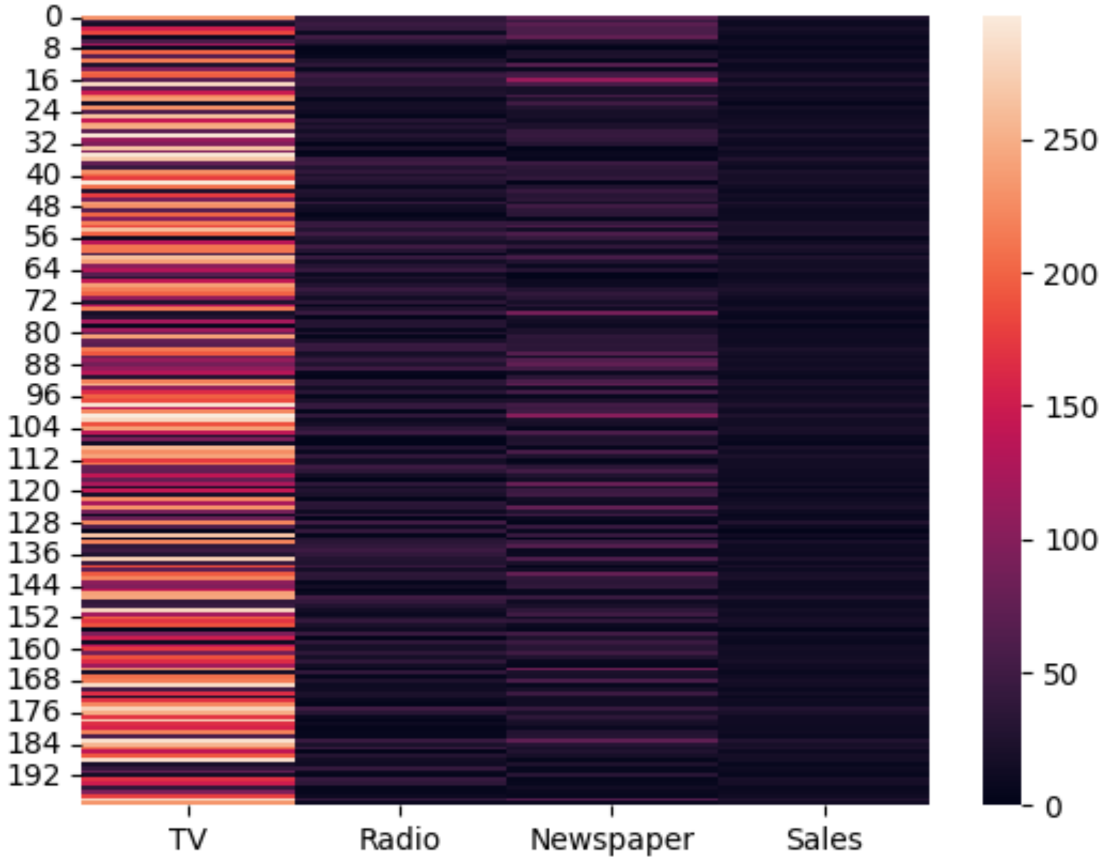
```
sns.heatmap(new_df)
```

<Axes: >



```
sns.heatmap(cf)
```

```
<Axes: >
```



Develop a Linear Regression model based on money spent on TV advertisements versus Sales.

```
new_df
```

	TV	Sales
0	230.1	22.1
1	44.5	10.4
2	17.2	9.3
3	151.5	18.5
4	180.8	12.9
...	...	...
195	38.2	7.6
196	94.2	9.7
197	177.0	12.8
198	283.6	25.5
199	232.1	13.4

200 rows × 2 columns

```
X = new_df.iloc[:,:-1].values
y = new_df.iloc[:,-1].values
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state= 42,test_size
```

```
X_test
```

```
array([[163.3],
       [195.4],
       [292.9],
       [ 11.7],
       [220.3],
       [ 75.1],
       [216.8],
       [ 50. ],
       [222.4],
       [175.1],
       [ 31.5],
       [ 56.2],
       [234.5],
       [  5.4],
       [139.5],
       [170.2],
       [  7.3],
       [197.6],
       [ 75.3],
       [237.4],
       [229.5],
       [ 67.8],
       [ 38. ],
       [250.9],
       [ 69. ],
       [ 53.5],
       [213.5],
       [139.3],
       [ 87.2],
       [  8.4],
       [199.8],
       [ 69.2],
       [198.9],
       [ 16.9],
       [280.7],
       [238.2],
       [ 48.3],
       [273.7],
       [117.2],
       [ 27.5]])
```

```
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

```
X_train
```

```
       [ 0.07819682],
       [ 1.50533154],
       [ 0.65332855],
```

```
       [-0.98769813],
       [-0.87481071],
       [-0.64547099],
       [-0.35434026],
       [ 0.9004926 ],
       [-1.4428127 ],
       [-0.9698738 ],
       [ 1.16904584],
       [ 0.81374752],
       [-0.47792228],
       [-1.68047044],
       [-0.62408179],
       [ 0.721061  ],
       [ 0.17207163],
       [ 1.36867834],
       [ 1.56118111],
       [-1.4642019 ],
       [ 1.50770812],
       [-1.34418474],
       [ 0.67471775],
       [ 1.63485501],
       [-0.56229078],
       [ 1.33897112],
       [-0.70845029],
       [ 0.59153754],
       [-0.66329532],
       [ 0.72224929],
       [ 0.91712864],
       [-0.73340435],
       [ 1.03833409],
       [-0.16421407],
       [ 0.26713472],
       [-1.57827761],
       [-1.07444321],
       [-0.89501162],
       [-0.00260681],
       [ 0.19940227],
       [-1.2538748 ],
       [ 0.78879346],
       [-1.25149822],
       [-1.77434525],
       [-0.34483395],
       [ 0.45013118],
       [-0.17609696],
       [-0.12856541],
       [-0.46722769],
       [ 0.7531448 ],
       [-1.55926499],
       [ 0.97535478],
       [ 0.81255923],
       [ 1.61584239],
       [-0.47792228],
       [-1.4855911 ],
```

```
           [ 0.64263395],
           [ 0.80424121],
           [ 0.1851428 ]
```

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Q3 With the regression line so developed, predict the sales that can be anticipated based on the money spent on TV advertisements.

```
y_pred = lr.predict(X_test)
y_pred
```

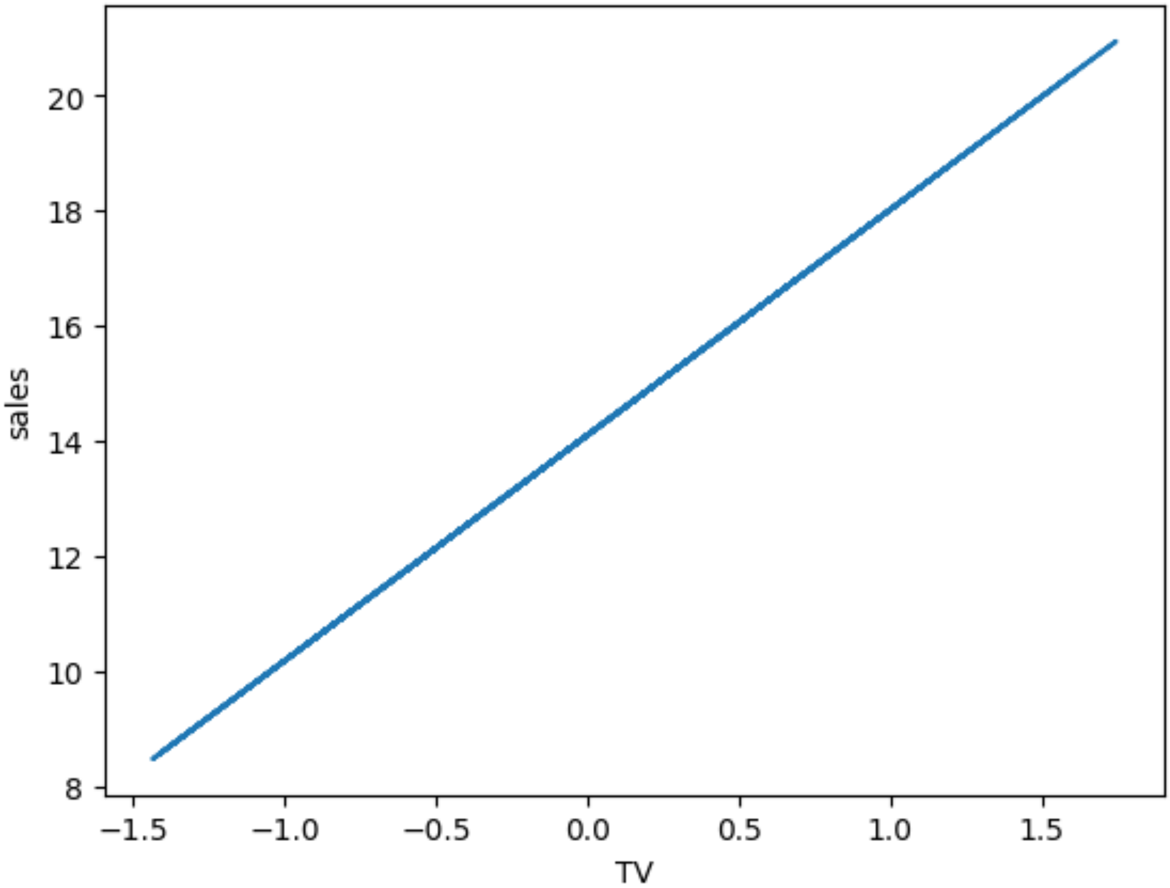
```
array([15.32048976, 16.71149708, 20.93651933,  8.75112184, 17.79050277,
       11.49846963, 17.6388353 , 10.41079724, 17.88150324, 15.83182579,
        9.60912635, 10.67946532, 18.40583934,  8.4781204 , 14.289151  ,
       15.61949133,  8.56045416, 16.80683092, 11.50713635, 18.53150667,
       18.18917153, 11.18213464,  9.8907945 , 19.11650975, 11.23413491,
       10.56246471, 17.49583455, 14.28048428, 12.02280573,  8.60812108,
       16.90216475, 11.24280162, 16.86316455,  8.97645636, 20.40784988,
       18.56617352, 10.33713019, 20.10451495, 13.32281257,  9.43579211])
```

```
y_test
```

```
array([16.9, 22.4, 21.4,  7.3, 24.7, 12.6, 22.3,  8.4, 11.5, 14.9,  9.5,
        8.7, 11.9,  5.3, 10.3, 11.7,  5.5, 16.6, 11.3, 18.9, 19.7, 12.5,
       10.9, 22.2,  9.3,  8.1, 21.7, 13.4, 10.6,  5.7, 10.6, 11.3, 23.7,
        8.7, 16.1, 20.7, 11.6, 20.8, 11.9,  6.9])
```

Q4 Draw the Regression Line superimposing on the data

```
plt.plot(X_test,y_pred)
plt.xlabel('TV')
plt.ylabel('sales')
plt.show()
```



Q5 Employ statsmodels.api and run an OLS regressor on the data. Plot the line of regression and residuals employing libraries of statsmodel. Comment on the heteroscedasticity.

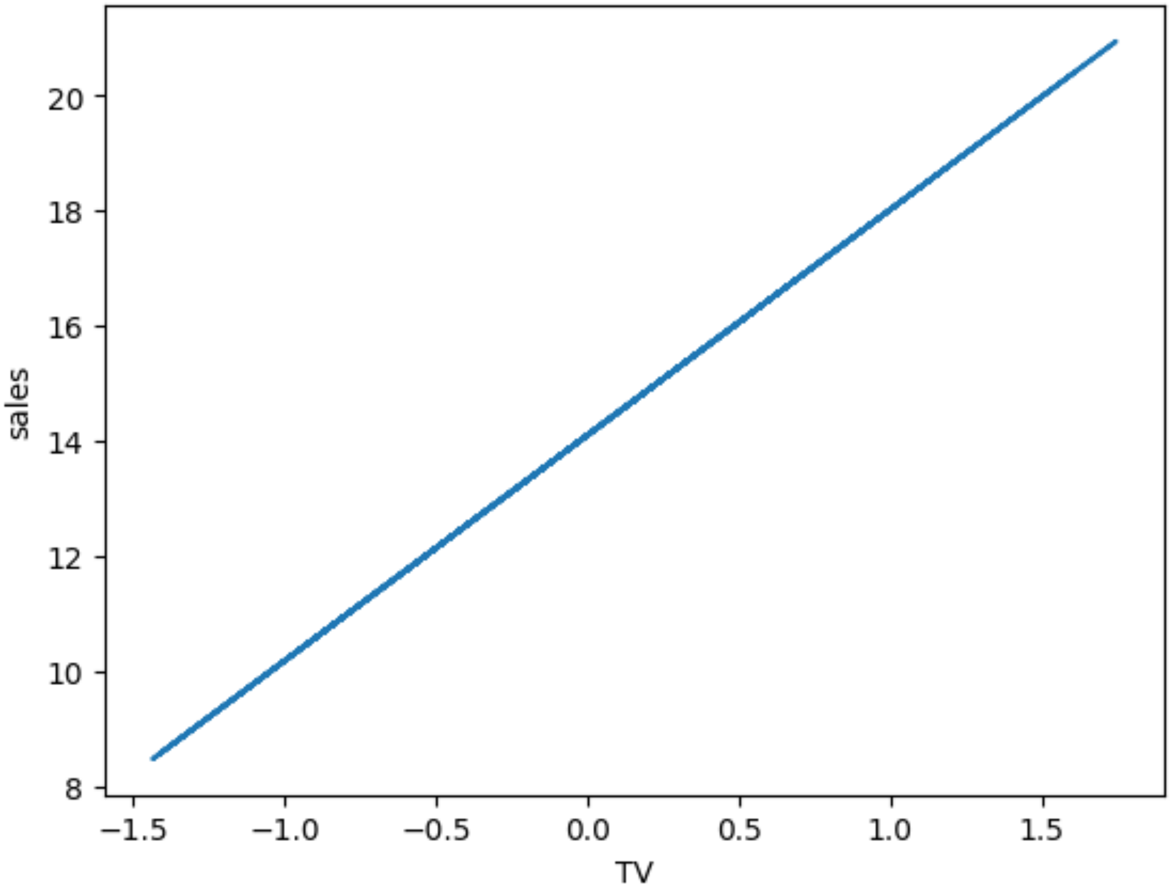
```
from statsmodels.api import OLS
```

```
residuals = y_test - y_pred
```

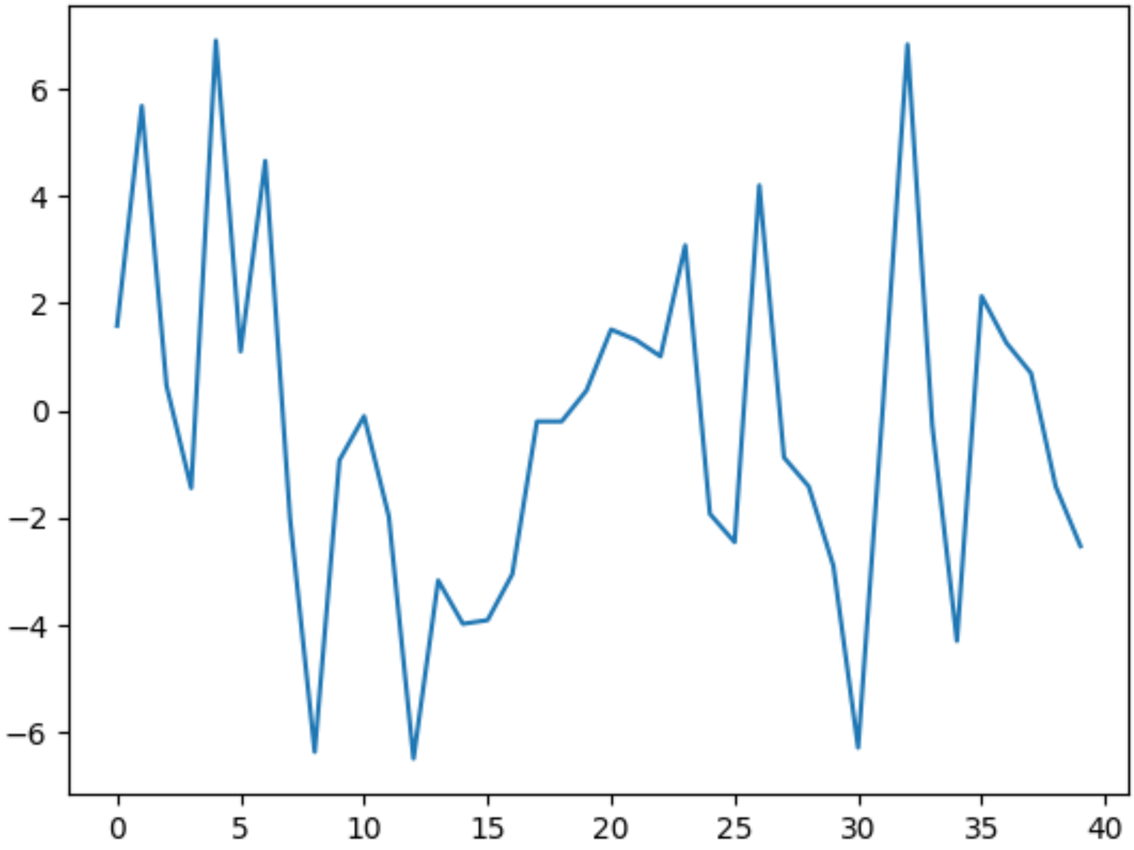
```
residuals
```

```
array([ 1.57951024,  5.68850292,  0.46348067, -1.45112184,  6.90949723,
        1.10153037,  4.6611647 , -2.01079724, -6.38150324, -0.93182579,
       -0.10912635, -1.97946532, -6.50583934, -3.1781204 , -3.989151  ,
       -3.91949133, -3.06045416, -0.20683092, -0.20713635,  0.36849333,
        1.51082847,  1.31786536,  1.0092055 ,  3.08349025, -1.93413491,
       -2.46246471,  4.20416545, -0.88048428, -1.42280573, -2.90812108,
       -6.30216475,  0.05719838,  6.83683545, -0.27645636, -4.30784988,
        2.13382648,  1.26286981,  0.69548505, -1.42281257, -2.53579211])
```

```
plt.plot(X_test,y_pred)
plt.xlabel('TV')
plt.ylabel('sales')
plt.show()
```



```
plt.plot(residuals)
plt.show()
```




```
mse =  mean_squared_error(y_test,y_pred)
mse
```

```
10.60440712647343
```

Yes over here the heteroscedasticity is pretty average as there is some deviation from the original values but still it is not too high