Sales Prediction using Python

Data Collection

Importing pandas library for accessing the data

```
import pandas as pd
```

```
a=pd.read_csv('https://docs.google.com/spreadsheets/d/11tF6SH9oeHXPVfJRICUfX6NwlxS9D_eKfrbNPCoof0M/export?format=csv&gid=0')
```

Data Organization

```
a.head(3)
```

	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

```
a.tail(3)
```

	TV	Radio	Newspaper	Sales
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

Getting the data's info/summary

```
a.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
```

```
a.describe()
```

	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

Data Cleaning

Checking for null values

```
a.isnull().sum()
```

```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

Checking for duplicates

```
a.duplicated()
```

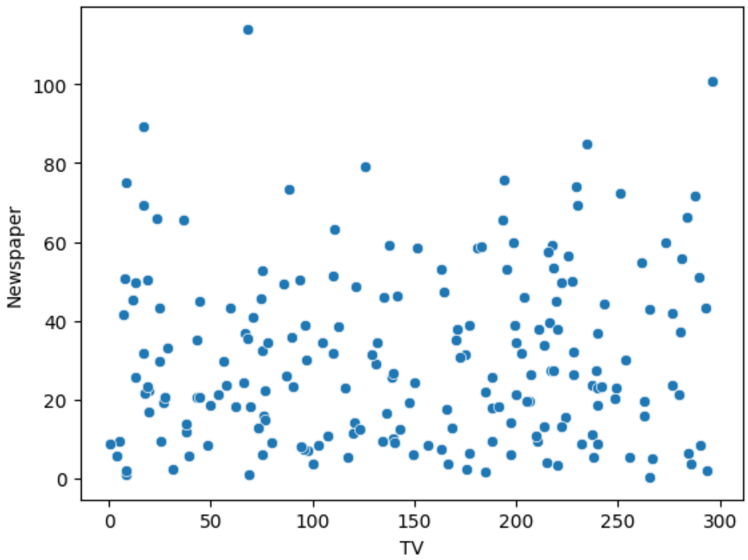
```
0      False
1      False
2      False
3      False
4      False
       ...
195    False
196    False
197    False
198    False
199    False
Length: 200, dtype: bool
```

Exploratory Data Analysis

```
#importing libraries for EDA
import seaborn as sns
import matplotlib.pyplot as plt
```

```
sns.scatterplot(data=a,x='TV',y='Newspaper')
```

```
<Axes: xlabel='TV', ylabel='Newspaper'>
```



Data Preparation - removing outliers

```
a[(a['TV']>50) & (a['Newspaper']>100)]
```

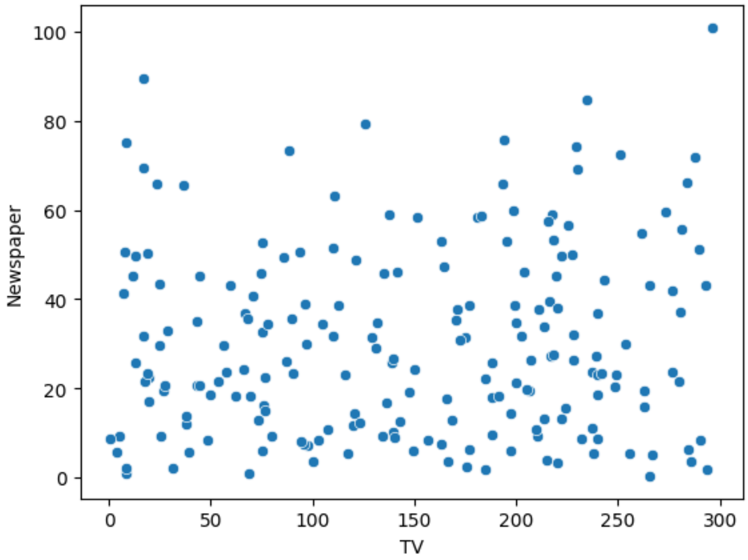
	TV	Radio	Newspaper	Sales
16	67.8	36.6	114.0	12.5
101	296.4	36.3	100.9	23.8

```
a=a.drop(index=16)
```

```
i=[]
for x in range(0,len(a)):
  i.append(x)
a.index=i
```

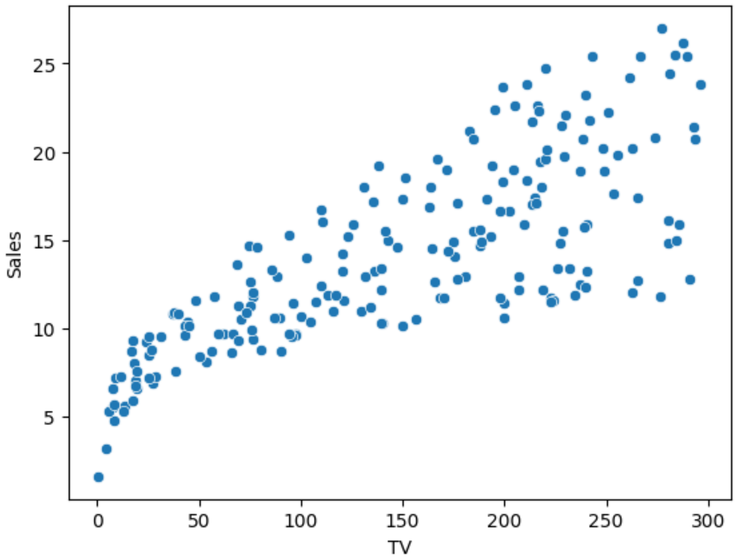
```
a1=sns.scatterplot(data=a,x='TV',y='Newspaper')
print(a1)
```

```
Axes(0.125,0.11;0.775x0.77)
```




```
# Checking the relationship between TV and Sales
sns.scatterplot(x='TV',y='Sales',data=a)
```

```
<Axes: xlabel='TV', ylabel='Sales'>
```



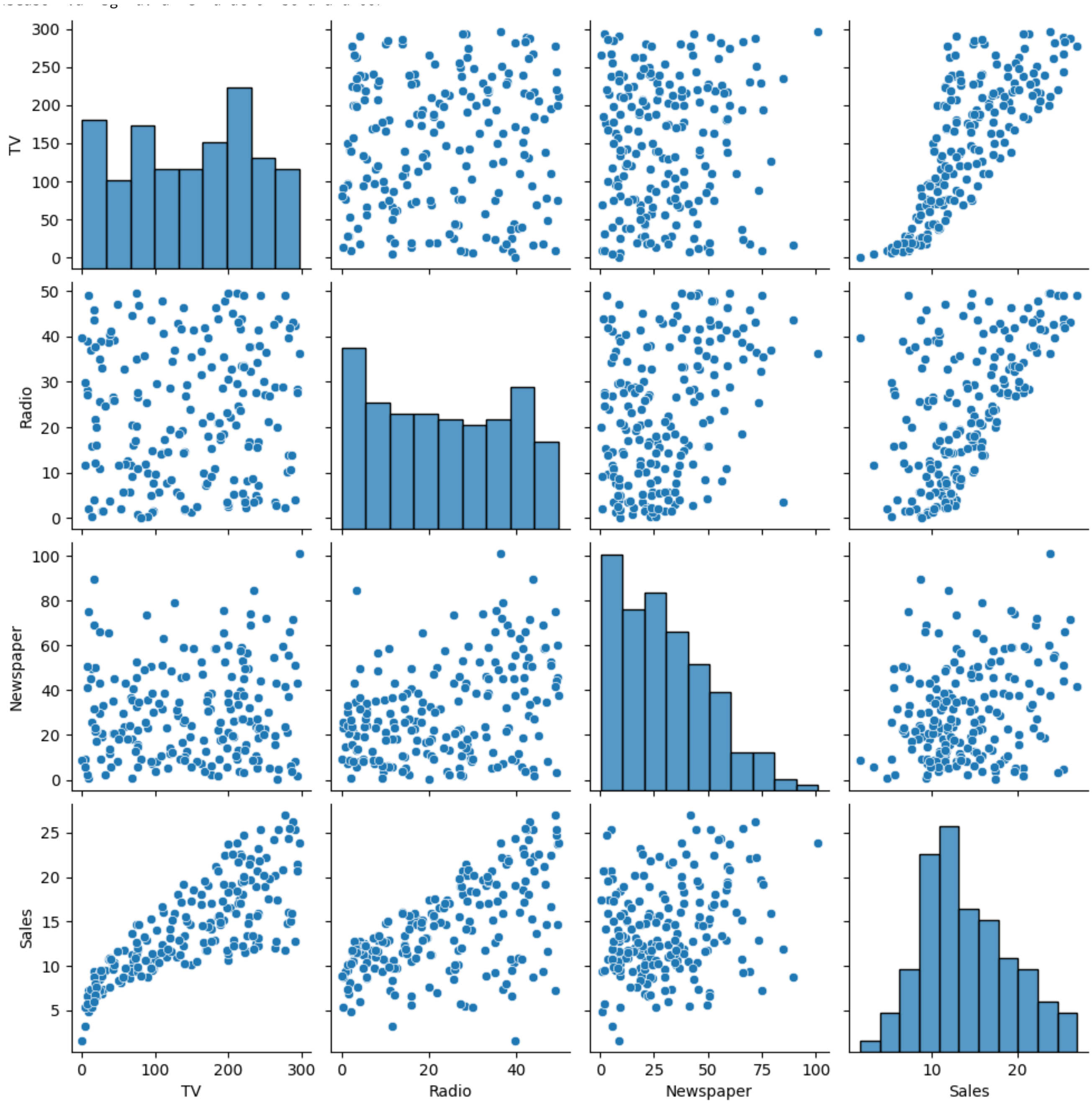
The above graph shows, as the TV values increases, Sales also increases linearly

```
sns.displot(x=a['Sales'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7b67a1a15810>
```

The distribution plot of sales, shows that, between 10 and 12, the values are higher

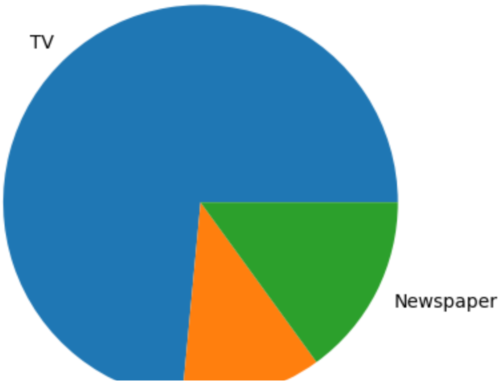
```
sns.pairplot(data=a)
```



Pie chart is plotted to show how the 3 media platforms are distributed

```
a_1=a['TV'].mean().round()
a_2=a['Radio'].mean().round()
a_3=a['Newspaper'].mean().round()
print(a_1,a_2,a_3)
T=a_1+a_2+a_3
print(T) # T for total
A=(a_1/T)*100
B=(a_2/T)*100
C=(a_3/T)*100
print(A,B,C) # individuals
plt.pie(x=[A,B,C],labels=['TV','Radio','Newspaper'])
```

```
147.0 23.0 30.0
200.0
73.5 11.5 15.0
([<matplotlib.patches.Wedge at 0x7b67a0d64100>,
  <matplotlib.patches.Wedge at 0x7b67a0d0dcf0>,
  <matplotlib.patches.Wedge at 0x7b67a0d64700>],
 [Text(-0.7403138014243597, 0.8135941712061451, 'TV'),
  Text(0.2902604572813063, -1.0610131323121534, 'Radio'),
  Text(0.9801072373902886, -0.4993894304199647, 'Newspaper')])
```



From the above piechart, we can infer that the sales value of TV is the highest

Data Preparation

```
# Due to the absence of text values, no dummy data is created
```

Dividing the data into input and output

```
x=a.drop(columns=['Sales']) # input column
y=a['Sales'] # output column
```

Standardising the data

```
# importing module for data standardisation
from sklearn.preprocessing import StandardScaler
```

```
s=StandardScaler()
```

```
# standardising the input column
x=pd.DataFrame(data=s.fit_transform(x),columns=x.columns) # x the inpt column has standardised values now.
```

Predictive Modelling

Since, the data is composed of continous variables, 2 algorithms are used for building the model - Linear Regression and K Nearest Regression

```
# importing module for model building and training
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2) # 80% data is to train the model and remaining 20% for testing
```

(1) Using Linear Regression

```
from sklearn.linear_model import LinearRegression
```

```
l1=LinearRegression() # defining a model
```

```
l1.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
# checking the accuracy of the model
l1.score(x_test,y_test)
```

```
0.8447907914936302
```

(2) K Nearest Regression

```
from sklearn.neighbors import KNeighborsRegressor
```

```
k=KNeighborsRegressor(n_neighbors=2) # defining the model
```

```
k.fit(x_train,y_train) # training the model
```

```
▾ KNeighborsRegressor
KNeighborsRegressor(n_neighbors=2)
```

```
k.score(x_test,y_test)
```

```
0.9414069683489297
```

```
k_b=KNeighborsRegressor(n_neighbors=5) # checking b changing the number of neighbors
k_b.fit(x_train,y_train) # training the model
k_b.score(x_test,y_test)
```

```
0.9507449402043913
```

Finding R2 score for choosing the better algorithm

```
# r2_score is imported
from sklearn.metrics import r2_score
```

```
y_pred=l1.predict(x_test) # for Linear Regression
print('R2 score of Linear Regression:',r2_score(y_test,y_pred))

y1_pred=k.predict(x_test) # for K Nearest Regression
print('R2 score of K Nearest Regression:',r2_score(y_test,y1_pred))
```

```
R2 score of Linear Regression: 0.8447907914936302
R2 score of K Nearest Regression: 0.9414069683489297
```

By comparing (1) and (2), K Nearest Regression with R2 score of 94 is found to be the suitable algorithm for the sales prediction.

So, K Nearest Regression algorithm is chosen to build the model for sales prediction