Oasis Infobyte Data Science Internship

Task 5

SALES PREDICTION USING PYTHON

JONNADA HANIRUDH REDDY

In [1]:
```
#importing basic libraries

import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#reading the dataset

data=pd.read_csv('Advertising.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	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 [4]:
```
# processing the data

data.shape
```

Out[4]:
```
(200, 5)
```

In [5]:
```
data.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 [6]:
```
#we don't need the 1st column so let's drop that

data=data.iloc[:,1:]
```

In [7]:
```
data.tail()
```

Out[7]:

	TV	Radio	Newspaper	Sales
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

In [8]:
```
#check for null values

data.isna().sum()
```

Out[8]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

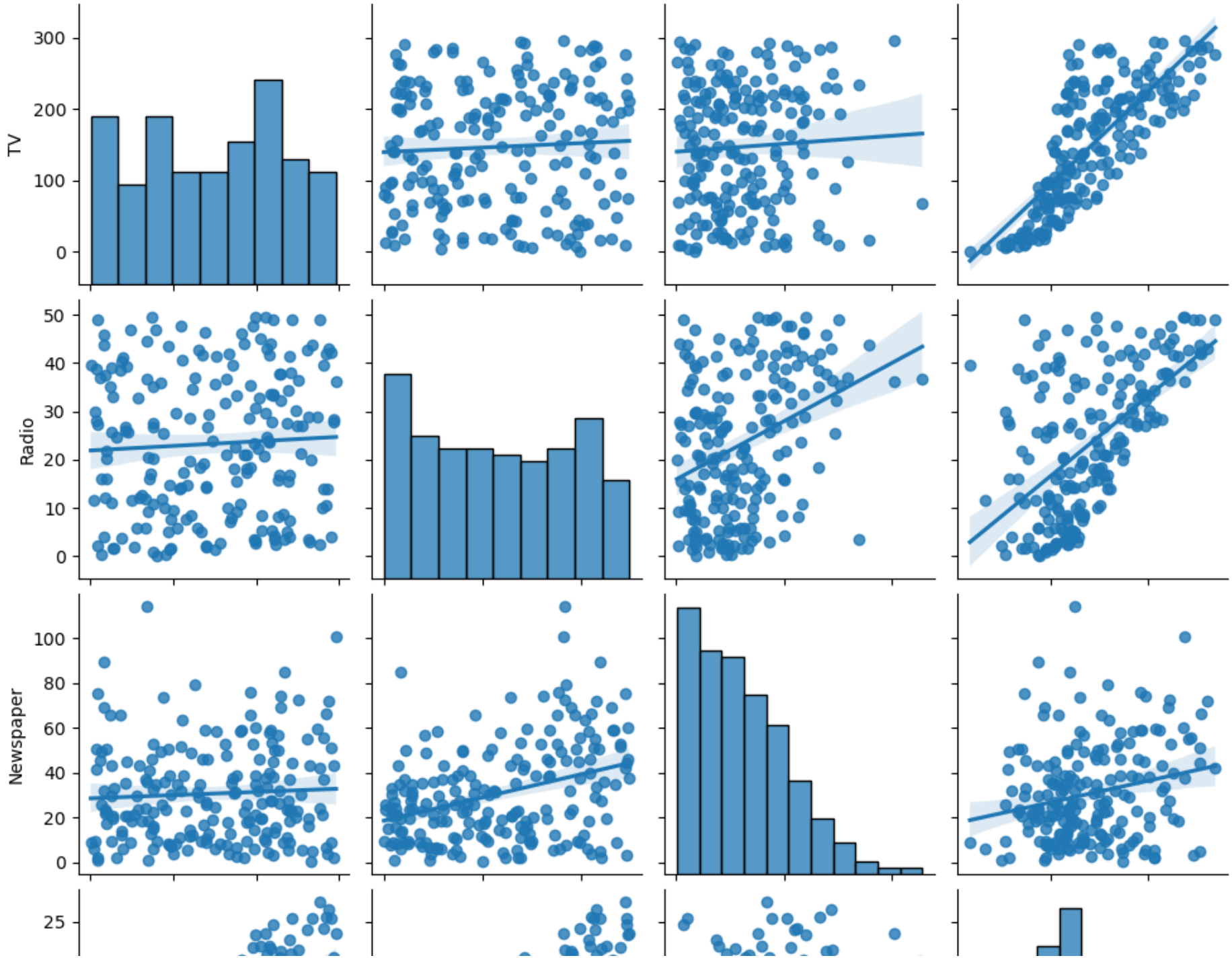
In [9]:
```
data.describe()
```

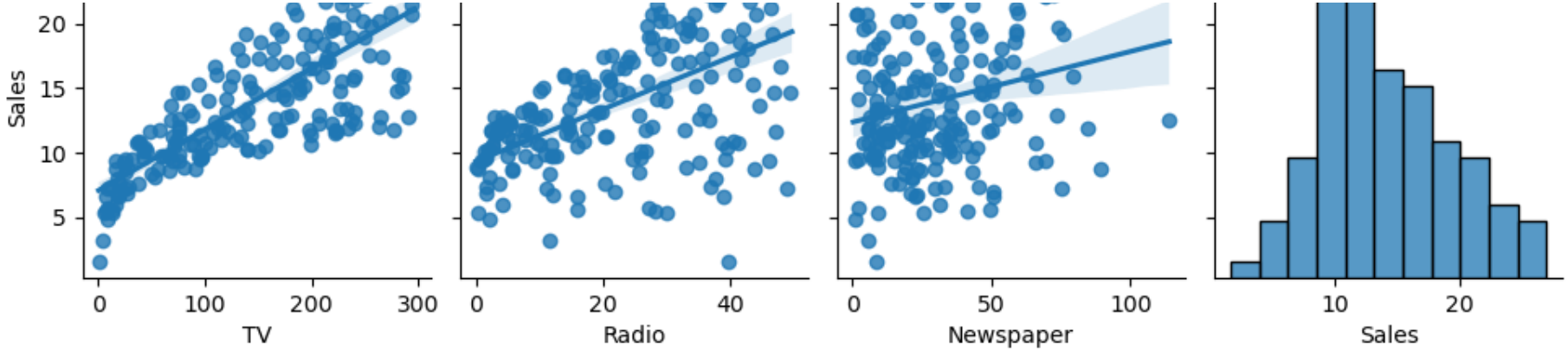
Out[9]:

	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

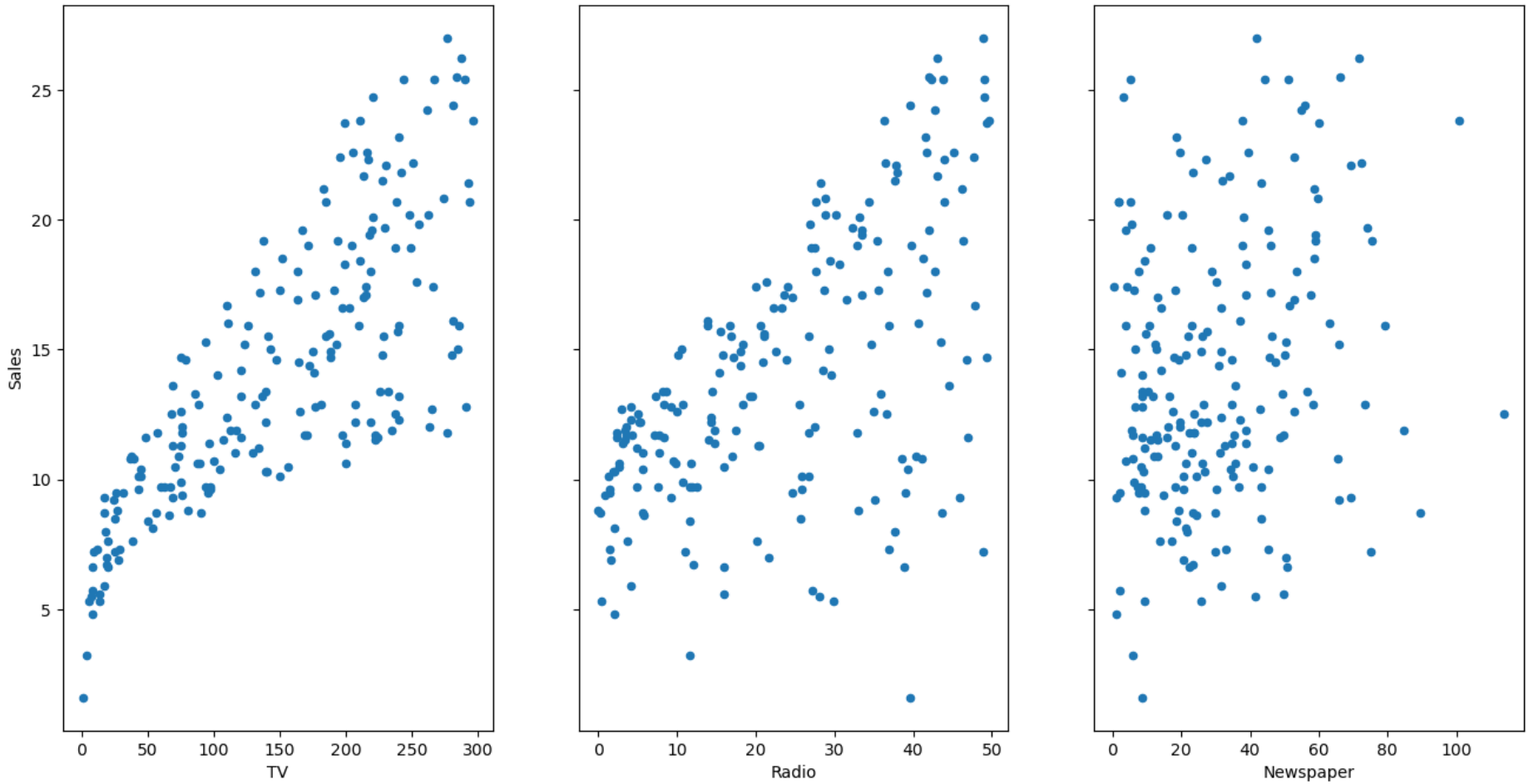
In [10]:
```
#Data Visulaization

import matplotlib.pyplot as plt
import seaborn as sns
```

In [11]:
```
sns.pairplot(data,kind="reg");
```


TV
Radio
Newspaper
300
200
100
0
50
40
30
20
10
0
100
80
60
40
20
0
25



In [12]:
```
fig,axs= plt.subplots(1,3,sharey=True)
data.plot(kind="scatter",x='TV',y='Sales',ax=axs[0],figsize=(16,8))
data.plot(kind="scatter",x='Radio',y='Sales',ax=axs[1],figsize=(16,8))
data.plot(kind="scatter",x='Newspaper',y='Sales',ax=axs[2],figsize=(16,8));
```



In [13]:
```
#rmoving the outlier from newspaper

data=data[data['Newspaper']<=90]
data.shape
```

Out[13]:
```
(198, 4)
```

In [14]:
```
data.corr()
```

Out[14]:

	TV	Radio	Newspaper	Sales
TV	1.000000	0.051978	0.049771	0.779121
Radio	0.051978	1.000000	0.346364	0.576748
Newspaper	0.049771	0.346364	1.000000	0.219555
Sales	0.779121	0.576748	0.219555	1.000000

In [15]:
```
# Separating input and output data

x=data.drop(columns=['Sales'])
y=data['Sales']
```

In [16]:
```
x.head()
```

Out[16]:

	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

In [17]:
```
y.head()
```

Out[17]:
```
0    22.1
1    10.4
2     9.3
3    18.5
4    12.9
Name: Sales, dtype: float64
```

In [18]:
```
from sklearn.model_selection import train_test_split
```

In [19]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [20]:
```
x_train.tail()
```

Out[20]:

	TV	Radio	Newspaper
77	120.5	28.5	14.2
128	220.3	49.0	3.2
82	75.3	20.3	32.5
49	66.9	11.7	36.8
97	184.9	21.0	22.0

In [21]:
```
y_test.tail()
```

Out[21]:
```
108     5.3
138     9.6
152    16.6
125    10.6
189     6.7
Name: Sales, dtype: float64
```

In [22]:
```
from sklearn.preprocessing import OneHotEncoder, StandardScaler,OrdinalEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score
```

In [23]:
```
column_trans=make_column_transformer((OneHotEncoder(sparse=False),[]),remainder='passthrough')
scaler=StandardScaler()
oe=OrdinalEncoder()
```

In [24]:
```
#Random Forest Regression Model

from sklearn.ensemble import RandomForestRegressor
r=RandomForestRegressor(n_estimators=10,random_state=0)
pipe=make_pipeline(column_trans,scaler,r)
pipe.fit(x_train,y_train)
y_pred_r=pipe.predict(x_test)
r2_score(y_test,y_pred_r)
```

Out[24]:
```
0.968614845119813
```

In [25]:
```
#Let's Check predict function working Good or Not

pipe.predict([[283.6,42.0,66.2]]) #Original ans 25.5
```

Out[25]:
```
array([25.74])
```

In [26]:
```
pipe.predict([[112.9,17.4,38.6]]) #Original ans 11.9
```

Out[26]:
```
array([12.26])
```

In [27]:
```
import pickle
```

In [28]:
```
pickle.dump(pipe,open('sales.pkl','wb'))
```

In [ ]:
```

```