In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
df = pd.read_csv("D:\\Data Science\\8.Machine Learning Algorithms\\1.Linear Regressn\\Advertising.csv")
```

In [5]:
```
df
```

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
...	...	...	...	...
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

In [6]:
```
df.columns
```

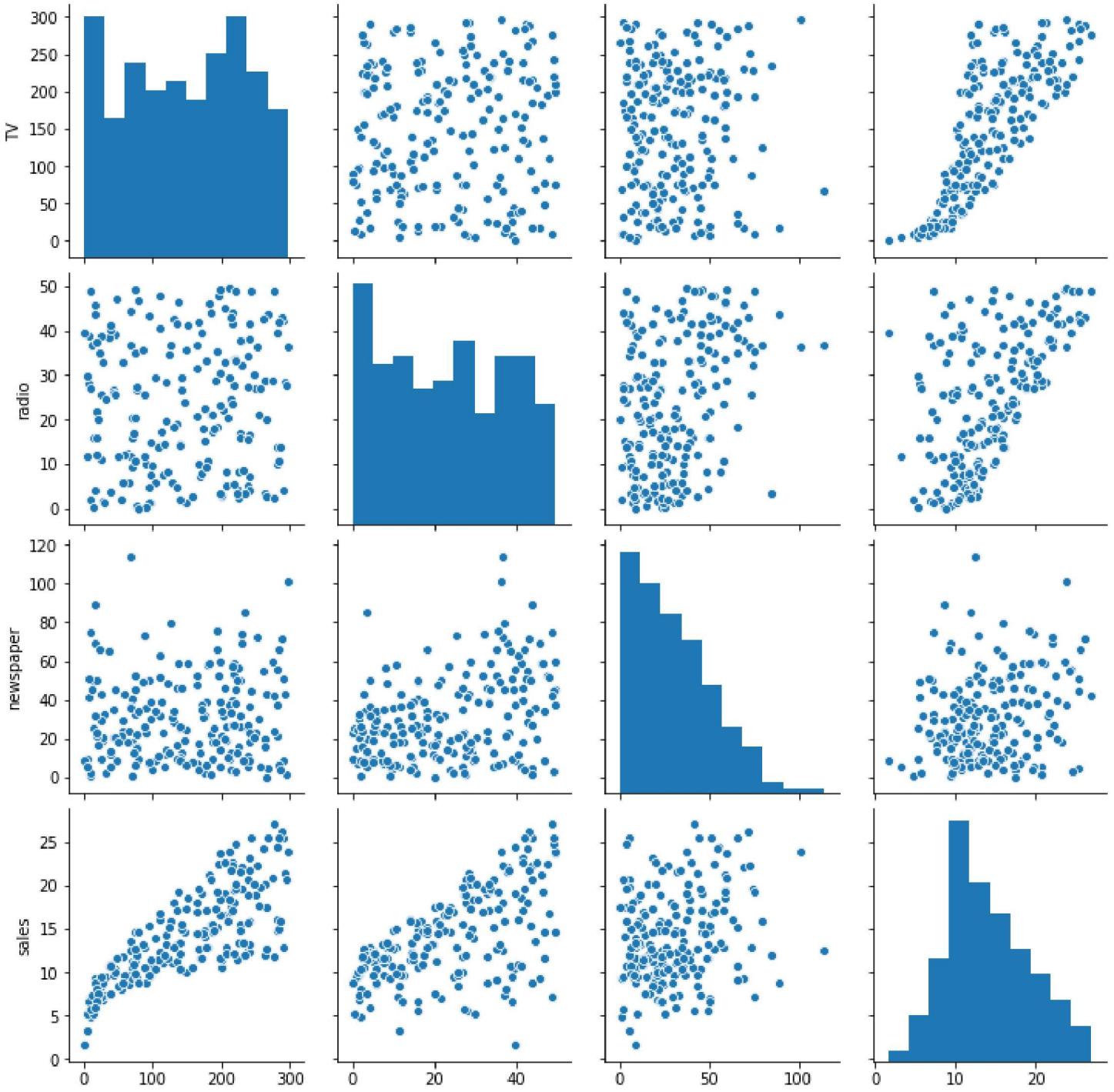
Out[6]:
```
Index(['TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

In [15]:
```
df.describe()
```

Out[15]:

	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 [17]:
```
sns.pairplot(df)
plt.show()
```


TV
radio
newspaper
sales
0
50
100
150
200
250
300
0
10
20
30
40
50
0
20
40
60
80
100
120
0
5
10
15
20
25
0
100
200
300
0
20
40
0
50
100
0
10
20

TV radio newspaper sales

In [18]:
```
df.corr()
```

Out[18]:

	TV	radio	newspaper	sales
TV	1.000000	0.054809	0.056648	0.782224
radio	0.054809	1.000000	0.354104	0.576223
newspaper	0.056648	0.354104	1.000000	0.228299
sales	0.782224	0.576223	0.228299	1.000000

In [19]:
```
X = df.drop('sales',axis=1)
y = df["sales"]
```

In [20]:
```
X
```

Out[20]:

	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 × 3 columns

In [21]:
```
y
```

Out[21]:
```
0      22.1
1      10.4
2       9.3
3      18.5
4      12.9
       ...
195     7.6
196     9.7
197    12.8
198    25.5
199    13.4
Name: sales, Length: 200, dtype: float64
```

In [28]:
```
from sklearn.preprocessing import PolynomialFeatures
polynomial_converter = PolynomialFeatures(degree=2,include_bias= False)
```

In [29]:
```
X_poly = polynomial_converter.fit_transform(X)
```

In [30]:
```
X_poly.shape
```

Out[30]:
```
(200, 9)
```

In [31]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X_poly,y,test_size=0.3,random_state=101)
```

In [32]:
```
from sklearn.linear_model import LinearRegression
model   =  LinearRegression()
model.fit(X_train,y_train)
```

Out[32]:
```
LinearRegression()
```

In [33]:
```
train_predictions = model.predict(X_train)
test_predictions = model.predict(X_test)
```

In [34]:
```
train_res = y_train-train_predictions
test_res=  y_test-test_predictions
```

In [35]:
```
model.score(X_train,y_train)
```

Out[35]: 0.9868638137712757

In [36]:
```
model.score(X_test,y_test)
```

Out[36]: 0.9843529333146783

In [37]:
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model,X_poly,y,cv=5)
scores
```

Out[37]: array([0.98795615, 0.98937857, 0.99129812, 0.95889074, 0.99374691])

In [39]:
```
scores.mean()
```

Out[39]: 0.9842540981580088

In [43]:
```
from sklearn.metrics import mean_absolute_error
MAE = mean_absolute_error(y_test,test_predictions)
MAE
```

Out[43]: 0.4896798044803816

In [44]:
```
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test,test_predictions)
MSE
```

Out[44]: 0.4417505510403749

In [46]:
```
RMSE = np.sqrt(MSE)
RMSE
```

Out[46]: 0.6646431757269271

In [58]:
```
train_rmse_errors=[]
test_rmse_errors = []

for d in range(1,10):
    polynomial_converter = PolynomialFeatures(degree=d,include_bias=False)
    X_poly = polynomial_converter.fit_transform(X)

    X_train,X_test,y_train,y_test = train_test_split(X_poly,y,test_size=0.3,random_state=101)

    model = LinearRegression()
    model.fit(X_train,y_train)

    train_pred = model.predict(X_train)
    test_pred = model.predict(X_test)

    # Calculate Errors

    train_RMSE = np.sqrt(mean_squared_error(y_train,train_pred))
    test_RMSE = np.sqrt(mean_squared_error(y_test,test_pred))

    train_rmse_errors.append(train_RMSE)
    test_rmse_errors.append(test_RMSE)
```

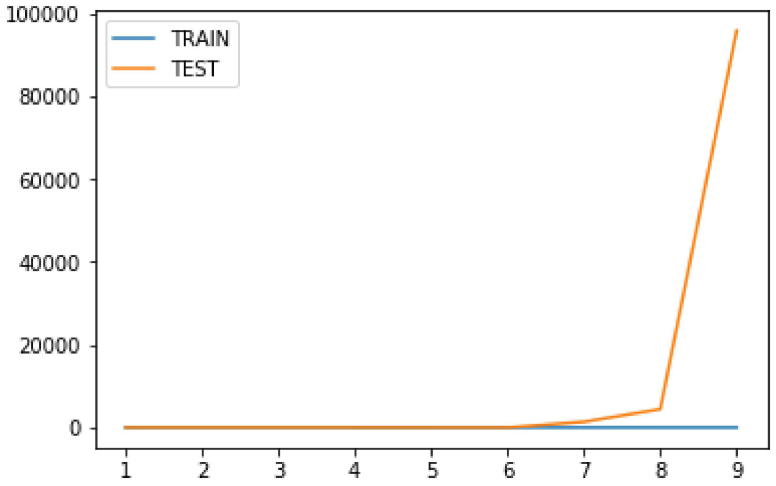
In [59]:
```
train_rmse_errors
```

Out[59]:
```
[1.7345941243293763,
 0.5879574085292228,
 0.43393443569020657,
 0.351708368839935,
 0.25093429470317896,
 0.19704459846551506,
 5.421420485986764,
 0.14180598547565168,
 0.16654227321644913]
```

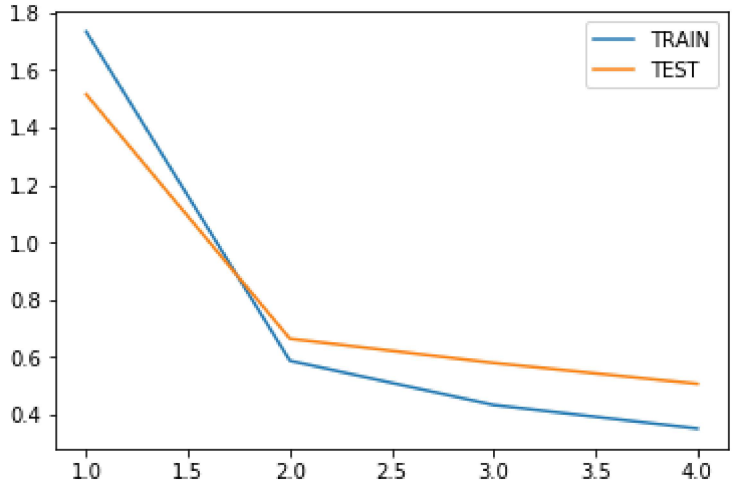
In [60]:
```
test_rmse_errors
```

Out[60]:
```
[1.5161519375993877,
 0.6646431757269271,
 0.5803286825165038,
 0.5077742649213964,
 2.5758311664662017,
 4.4926997025114845,
 1381.404421689979,
 4449.599748615518,
 95891.24543526022]
```

In [63]:
```
plt.plot(range(1,10),train_rmse_errors,label="TRAIN")
plt.plot(range(1,10),test_rmse_errors,label="TEST")
plt.legend()
plt.show()
```



In [65]:
```
plt.plot(range(1,5),train_rmse_errors[:4],label="TRAIN")
plt.plot(range(1,5),test_rmse_errors[:4],label="TEST")
plt.legend()
plt.show()
```



In [72]:
```
final_Poly_converter  =PolynomialFeatures(degree=3,include_bias=False)
final_model = LinearRegression()
final_model.fit(final_Poly_converter.fit_transform(X),y)
```

Out[72]:
```
LinearRegression()
```

In [74]:
```
from joblib import dump,load
```

In [75]:
```
dump(final_model , 'sales_poly_model.joblib')
```

Out[75]:
```
['sales_poly_model.joblib']
```

In [77]:
```
dump(final_Poly_converter,'poly_converter.joblib')
```

Out[77]:
```
['poly_converter.joblib']
```

In [78]:
```
loaded_poly = load("poly_converter.joblib")
loaded_model = load("sales_poly_model.joblib")
```

In [79]:
```
campaign_poly=  loaded_poly.transform([[149,22,12]])
```

```
C:\Users\Nazar\anaconda3\lib\site-packages\sklearn\base.py:446: UserWarning: X does not have valid feature names, but
PolynomialFeatures was fitted with feature names
  "X does not have valid feature names, but"
```

In [80]:
```
final_model.predict(campaign_poly)
```

Out[80]:
```
array([14.64501014])
```

In [ ]: