Problem Statement: The Advertising dataset captures sales revenue generated with respect to advertisement spends across multiple channels like radio, tv, and newspaper.

Objective: Build a linear regression model to:

- Interpret the coefficients of the model
- Make predictions
- Find and analyze model residuals
- Evaluate model efficiency using RMSE and R Square values

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

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

```
df.head()
```

	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

```
Df=pd.read_csv('Advertising.csv', index_col=0)
```

```
Df.head()
```

	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

```
Df.shape
```

```
(200, 4)
```

```
Df.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

```
type(Df)
```

```
pandas.core.frame.DataFrame
```

```
Df.isnull().sum()
```

```
TV           0
radio        0
newspaper    0
sales        0
dtype: int64
```

```
Df.dtypes
```

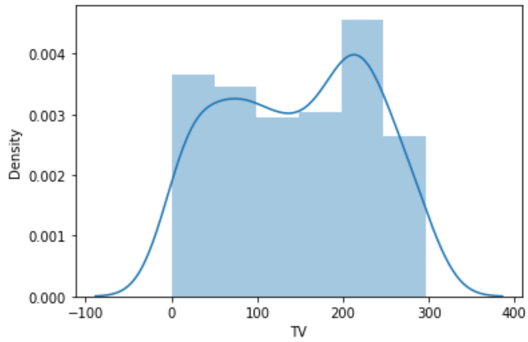
```
TV           float64
radio        float64
newspaper    float64
sales        float64
dtype: object
```

```
### check for correlation
Df.corr()
```

	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

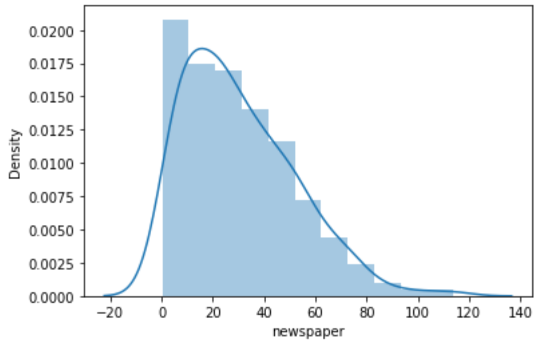
```
### Check for distribution

sns.distplot(Df['TV'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fcb4f0e8e50>
```



```
sns.distplot(Df['newspaper'])
```

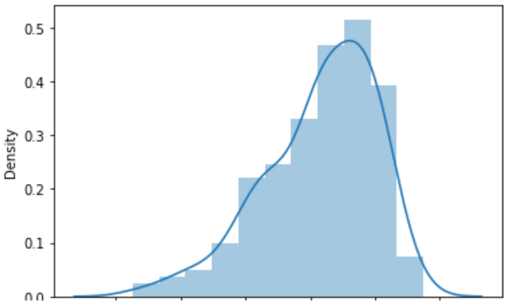
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fcb4cf36590>
```



```
### newspaper is skewed and having outliers.So need to apply transformation techniques.

Df['newspaper'] = np.log1p(Df['newspaper'])
```

```
sns.distplot(Df['newspaper'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fcb4ca437d0>
```



```
### Scaling --

from sklearn.preprocessing import MinMaxScaler


df_num=Df[['TV','radio','newspaper']]


mn=MinMaxScaler()


Df_sc=mn.fit_transform(df_num)


Df_sc
```

```
array([[0.77578627, 0.76209677, 0.88988816],
       [0.1481231 , 0.79233871, 0.79607243],
       [0.0557998 , 0.92540323, 0.89020572],
       [0.50997633, 0.83266129, 0.85299591],
       [0.60906324, 0.21774194, 0.85262066],
       [0.02705445, 0.9858871 , 0.90759788],
       [0.19208657, 0.66129032, 0.65505062],
       [0.4041258 , 0.39516129, 0.50670344],
       [0.02671627, 0.04233871, 0.09610182],
       [0.67331755, 0.05241935, 0.63305858],
       [0.2211701 , 0.11693548, 0.66133516],
       [0.72370646, 0.48387097, 0.30051383],
       [0.07811972, 0.70766129, 0.87914669],
       [0.32735881, 0.15322581, 0.41087384],
       [0.68785932, 0.66330645, 0.80038573],
       [0.65843761, 0.96169355, 0.83094475],
       [0.22691917, 0.73790323, 1.        ],
       [0.94927291, 0.7983871 , 0.84263578],
       [0.2316537 , 0.41330645, 0.60182933],
       [0.49577274, 0.48185484, 0.61088993],
       [0.73621914, 0.55846774, 0.83300465],
       [0.80047345, 0.10282258, 0.65505062],
       [0.04227257, 0.32056452, 0.81685039],
       [0.76969902, 0.34072581, 0.67837293],
       [0.20831924, 0.25403226, 0.60182933],
       [0.8867095 , 0.07056452, 0.61528586],
       [0.4808928 , 0.59072581, 0.52374121],
       [0.80960433, 0.33669355, 0.64951927],
       [0.83902604, 0.54637097, 0.64951927],
       [0.23638823, 0.32258065, 0.77422856],
       [0.98816368, 0.57056452, 0.78668313],
       [0.37943862, 0.35080645, 0.7621669 ],
       [0.32634427, 0.03024194, 0.707546  ],
       [0.89584038, 0.40322581, 0.        ],
       [0.32127156, 0.02822581, 0.41624968],
       [0.98072371, 0.08266129, 0.44370272],
       [0.90023673, 0.88306452, 0.3411873 ],
       [0.25025364, 0.99596774, 0.79895721],
       [0.14338857, 0.53830645, 0.74152332],
       [0.76868448, 0.76008065, 0.72149344],
       [0.68244843, 0.44959677, 0.71877284],
       [0.59621238, 0.6733871 , 0.76272954],
       [0.99053094, 0.55846774, 0.1711642 ],
       [0.69732837, 0.16935484, 0.68000727],
       [0.08251606, 0.51814516, 0.78718728],
       [0.58978695, 0.45362903, 0.71808747],
       [0.30098072, 0.19959677, 0.74520065],
       [0.80892797, 0.83669355, 0.60412921],
       [0.76597903, 0.31854839, 0.81816913],
       [0.22387555, 0.2358871 , 0.75178892],
       [0.67331755, 0.0625    , 0.73841188],
       [0.33716605, 0.19354839, 0.28191253],
       [0.72945553, 0.84072581, 0.76773044],
       [0.61515049, 0.93145161, 0.85374452],
       [0.88603314, 0.58064516, 0.57220536],
       [0.67027393, 0.99596774, 0.85855021],
       [0.02231992, 0.56653226, 0.777408  ],
            [0.4582347 , 0.38709677, 0.58125938],
```

```
df_sc_df=pd.DataFrame(Df_sc, columns=df_num.columns, index=df_num.index)
```

```
df_sc_df
```

	TV	radio	newspaper
1	0.775786	0.762097	0.889888
2	0.148123	0.792339	0.796072
3	0.055800	0.925403	0.890206
4	0.509976	0.832661	0.852996
5	0.609063	0.217742	0.852621
...	...	...	...
196	0.126818	0.074597	0.542605
197	0.316199	0.098790	0.434106
198	0.596212	0.187500	0.387973
199	0.956713	0.846774	0.880145
200	0.782550	0.173387	0.448351

200 rows × 3 columns

Now preprocessing done, we have to test and train the data for prediction.

```
## test train split
```

```
from sklearn.model_selection import train_test_split
```

```
X=df_sc_df
y=Df["sales"]
```

```
 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
X_train
```

	TV	radio	newspaper
135	0.122421	0.778226	0.878144
67	0.104160	0.495968	0.200953
27	0.480893	0.590726	0.523741
114	0.706459	0.415323	0.490171
169	0.726074	0.475806	0.849596
...	...	...	...
68	0.468718	0.292339	0.480428
193	0.055800	0.082661	0.718773
118	0.256003	0.016129	0.557191
48	0.808928	0.836694	0.604129
173	0.063916	0.405242	0.586273

160 rows × 3 columns

```
X_test
```

	TV	radio	newspaper
19	0.231654	0.413306	0.601829
171	0.166723	0.233871	0.602982
108	0.303348	0.006048	0.652302
99	0.977342	0.852823	0.823795
178	0.573216	0.157258	0.742140
183	0.187690	0.114919	0.705377
6	0.027054	0.985887	0.907598
147	0.809604	0.147177	0.448351
13	0.078120	0.707661	0.879147
153	0.665878	0.469758	0.548554
62	0.881299	0.860887	0.838273
126	0.292526	0.237903	0.675899
181	0.527224	0.052419	0.438956
155	0.632736	0.425403	0.466030
81	0.256003	0.538306	0.643847
8	0.404126	0.395161	0.506703
34	0.895840	0.403226	0.000000
131	0.000000	0.798387	0.448351
38	0.250254	0.995968	0.798957
75	0.719310	0.495968	0.531796
184	0.970240	0.866935	0.898001
146	0.472100	0.038306	0.455146
46	0.589787	0.453629	0.718087
160	0.443017	0.370968	0.738412
61	0.178559	0.040323	0.635059
124	0.413933	0.697581	0.520436
180	0.557660	0.201613	0.593588
186	0.690903	0.909274	0.616371
123	0.755157	0.048387	0.568210
45	0.082516	0.518145	0.787187
17	0.226919	0.737903	1.000000
56	0.670274	0.995968	0.858550
151	0.946906	0.280242	0.752966
112	0.815015	0.766129	0.652302

```
y_train
```

```
135    10.8
67      9.5
27     15.0
114    15.9
169    17.1
       ...
68     13.4
193     5.9
118     9.4
48     23.2
173     7.6
Name: sales, Length: 160, dtype: float64
```

```
y_test
```

```
19     11.3
171     8.4
108     8.7
99     25.4
178    11.7
183     8.7
6       7.2
147    13.2
```

```
13      9.2
153    16.6
62     24.2
126    10.6
181    10.5
155    15.6
81     11.8
8      13.2
34     17.4
131     1.6
38     14.7
75     17.0
184    26.2
146    10.3
46     14.9
160    12.9
61      8.1
124    15.2
180    12.6
186    22.6
123    11.6
45      8.5
17     12.5
56     23.7
151    16.1
112    21.8
23      5.6
190     6.7
130     9.7
5      12.9
84     13.6
107     7.2
Name: sales, dtype: float64
```

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

```
lr = LinearRegression()
```

```
# train the model

lr.fit(X_train, y_train)
```

```
LinearRegression()
```

```
#### Prediction

pred = lr.predict(X_test)
```

```
### evaluate the model

from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```

```
r2_score(y_test, pred)
```

```
0.8596393983901572
```

```
mean_squared_error(y_test,pred)
```

```
4.417070057077106
```

```
mean_absolute_error(y_test,pred)
```

```
1.371914969265814
```

```
### OverFitting

mod = lr.predict(X_train)
```

```
r2_score(y_train, mod)
```

```
0.906658005856106
```

```
### By comparing the test and train score, I see that model is not overfitted
```

● ×