```
!pip install -q autoviz
!pip install -q -U --pre pycaret
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.0/67.0 kB 2.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.5/18.5 MB 72.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 361.8/361.8 kB 22.8 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.3/4.3 MB 97.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 95.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 93.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20.0/20.0 MB 92.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20.0/20.0 MB 82.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20.0/20.0 MB 24.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20.0/20.0 MB 82.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19.9/19.9 MB 73.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19.9/19.9 MB 96.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19.9/19.9 MB 16.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19.9/19.9 MB 84.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20.8/20.8 MB 79.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 74.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 121.9/121.9 kB 13.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.9/84.9 kB 8.2 MB/s eta 0:00:00
  Building wheel for emoji (pyproject.toml) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 484.4/484.4 kB 5.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 153.4/153.4 kB 13.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.9/81.9 kB 6.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 15.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.9/79.9 MB 12.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 106.8/106.8 kB 10.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.4/73.4 kB 6.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.1/17.1 MB 73.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.0/44.0 kB 3.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 76.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10.4/10.4 MB 100.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 140.3/140.3 kB 13.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 185.2/185.2 kB 18.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.3/2.3 MB 99.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 118.2/118.2 kB 9.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 233.6/233.6 kB 21.9 MB/s eta 0:00:00
  Building wheel for pyod (setup.py) ... done
```

```
#import library
from ast import increment_lineno
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline



#Import Libarary for Machine Learning
from scipy import stats

from statsmodels.stats.outliers_influence import variance_inflation_factor

from pycaret.regression import *
from pycaret  import regression
from sklearn.model_selection import cross_val_score


from google.colab import files
uploaded=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving Advertising[1].csv to Advertising[1] (1).csv

```
sales=pd.read_csv('Advertising[1] (1).csv')
print(sales)
```

```
     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
5           6       8.7  48.9       75.0     7.2
6           7      57.5  32.8       23.5    11.8
7           8     120.2  19.6       11.6    13.2
8           9       8.6   2.1        1.0     4.8
9          10     199.8   2.6       21.2    10.6
10         11      66.1   5.8       24.2     8.6
11         12     214.7  24.0        4.0    17.4
12         13      23.8  35.1       65.9     9.2
13         14      97.5   7.6        7.2     9.7
14         15     204.1  32.9       46.0    19.0
15         16     195.4  47.7       52.9    22.4
16         17      67.8  36.6      114.0    12.5
17         18     281.4  39.6       55.8    24.4
18         19      69.2  20.5       18.3    11.3
19         20     147.3  23.9       19.1    14.6
20         21     218.4  27.7       53.4    18.0
21         22     237.4   5.1       23.5    12.5
22         23      13.2  15.9       49.6     5.6
23         24     228.3  16.9       26.2    15.5
24         25      62.3  12.6       18.3     9.7
25         26     262.9   3.5       19.5    12.0
26         27     142.9  29.3       12.6    15.0
27         28     240.1  16.7       22.9    15.9
28         29     248.8  27.1       22.9    18.9
29         30      70.6  16.0       40.8    10.5
30         31     292.9  28.3       43.2    21.4
31         32     112.9  17.4       38.6    11.9
32         33      97.2   1.5       30.0     9.6
33         34     265.6  20.0        0.3    17.4
34         35      95.7   1.4        7.4     9.5
35         36     290.7   4.1        8.5    12.8
36         37     266.9  43.8        5.0    25.4
37         38      74.7  49.4       45.7    14.7
38         39      43.1  26.7       35.1    10.1
39         40     228.0  37.7       32.0    21.5
40         41     202.5  22.3       31.6    16.6
41         42     177.0  33.4       38.7    17.1
42         43     293.6  27.7        1.8    20.7
43         44     206.9   8.4       26.4    12.9
44         45      25.1  25.7       43.3     8.5
45         46     175.1  22.5       31.5    14.9
46         47      89.7   9.9       35.7    10.6
47         48     239.9  41.5       18.5    23.2
48         49     227.2  15.8       49.9    14.8
49         50      66.9  11.7       36.8     9.7
50         51     199.8   3.1       34.6    11.4
51         52     100.4   9.6        3.6    10.7
52         53     216.4  41.7       39.6    22.6
53         54     182.6  46.2       58.7    21.2
54         55     262.7  28.8       15.9    20.2
55         56     198.9  49.4       60.0    23.7
56         57       7.3  28.1       41.4     5.5
```

```
sales.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

```
sales.tail()
```

	Unnamed: 0	TV	Radio	Newspaper	Sales
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

```
sales.shape
```

```
(200, 5)
```

```
sales.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
```

```
sales.describe()
```

	Unnamed: 0	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	147.042500	23.264000	30.554000	14.022500
std	57.879185	85.854236	14.846809	21.778621	5.217457
min	1.000000	0.700000	0.000000	0.300000	1.600000
25%	50.750000	74.375000	9.975000	12.750000	10.375000
50%	100.500000	149.750000	22.900000	25.750000	12.900000
75%	150.250000	218.825000	36.525000	45.100000	17.400000
max	200.000000	296.400000	49.600000	114.000000	27.000000

```
sales.isnull().sum()
```

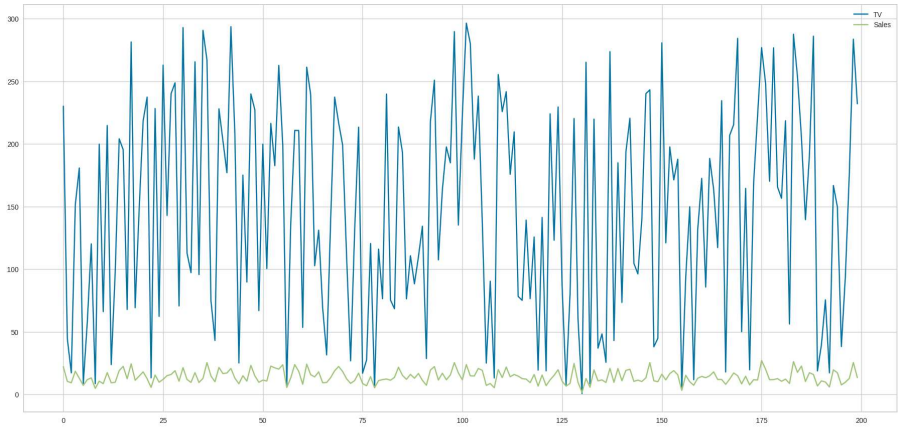
```
Unnamed: 0    0
TV            0
Radio         0
Newspaper     0
Sales         0
dtype: int64
```

```
sales.drop('Unnamed: 0', axis = 1, inplace = True)
```

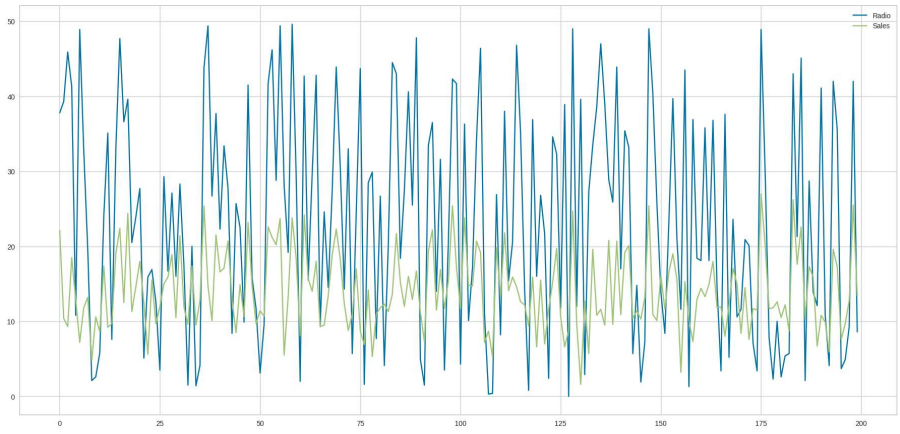
```
sales.head()
```

	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

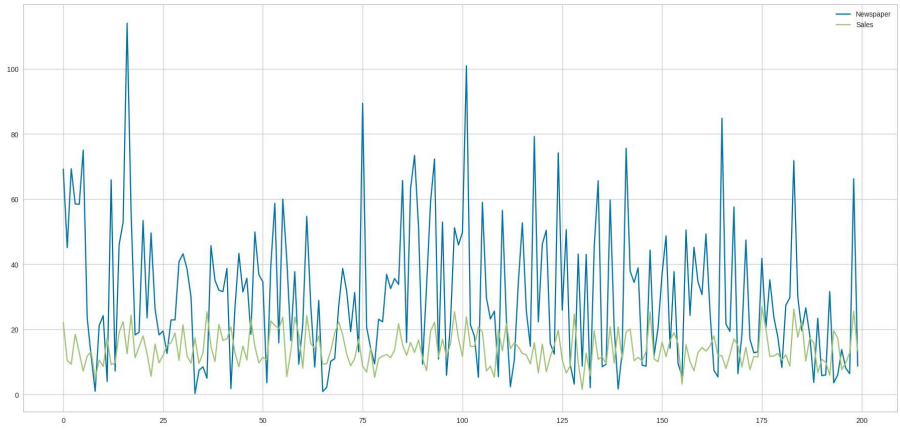
```
#TV vs Sales Listing
f = plt.figure()
f.set_figwidth(20)
f.set_figheight(10)
plt.plot(sales["TV"] ,label = "TV")
plt.plot(sales["Sales"] ,label = "Sales")
plt.legend()
plt.show()
```




```
#Radio Vs sales Listing
f = plt.figure()
f.set_figwidth(20)
f.set_figheight(10)
plt.plot(sales["Radio"] ,label = "Radio")
plt.plot(sales["Sales"] ,label = "Sales")
plt.legend()
plt.show()
```

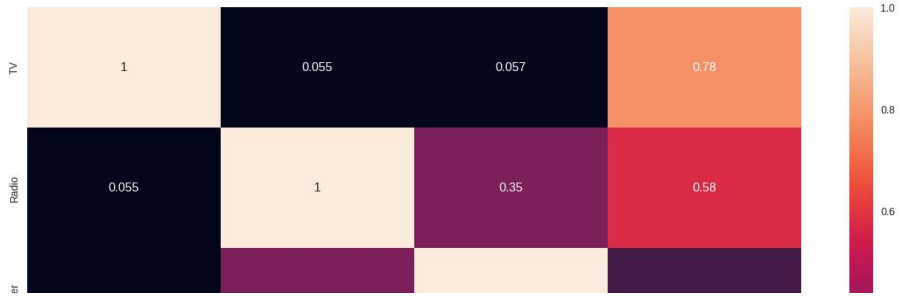


```
#Newspaper vs Sales Listing
f = plt.figure()
f.set_figwidth(20)
f.set_figheight(10)
plt.plot(sales["Newspaper"] ,label = "Newspaper")
plt.plot(sales["Sales"] ,label = "Sales")
plt.legend()
plt.show()
```




```
#correlation Graph
plt.figure(figsize=(15,8))
sns.heatmap(sales.corr(),annot=True)
```

```
<Axes: >
```




```
#Define a function for detecting a outliers
def detect_outliers(data):
    outlier_percents = {}
    for column in data.columns:
        if data[column].dtype != object:
            q1 = np.quantile(data[column], 0.25)
            q3 = np.quantile(data[column], 0.75)
            iqr = q3 - q1
            upper_bound = q3 + (1.5 * iqr)
            lower_bound = q1 - (1.5 * iqr)
            outliers = data[(data[column] > upper_bound) | (data[column] < lower_bound)][column]
            outlier_percentage = len(outliers) / len(data[column]) * 100
            outlier_percents[column] = outlier_percentage
            outlier_dataframe = pd.DataFrame(data = outlier_percents.values() ,index=outlier_percents.keys() ,columns=['Outlier_percentage'])

    return outlier_dataframe.sort_values(by = 'Outlier_percentage', ascending = False)


#Outlier_percentage
detect_outliers(sales)
```

	Outlier_percentage
Newspaper	1.0
TV	0.0
Radio	0.0
Sales	0.0

```
X = sales.drop('Sales', axis = 1)
y = sales['Sales']


#Transformed test set
sl = setup(data = sales, target = 'Sales', session_id=123)
```

	Description	Value
0	Session id	123
1	Target	Sales
2	Target type	Regression

```
#Compare different Models
compare_models()
```

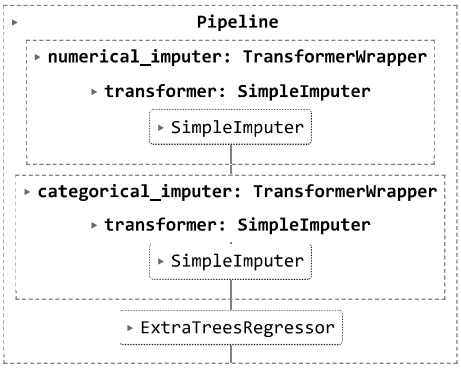
	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	0.4624	0.4099	0.6207	0.9829	0.0638	0.0524	0.1880
rf	Random Forest Regressor	0.6819	0.7522	0.8463	0.9707	0.0810	0.0698	0.2100
gbr	Gradient Boosting Regressor	0.6543	0.7543	0.8449	0.9695	0.0821	0.0685	0.2180
xgboost	Extreme Gradient Boosting	0.7092	0.8571	0.9083	0.9628	0.0842	0.0708	0.0550
ada	AdaBoost Regressor	0.9118	1.3620	1.1184	0.9487	0.1028	0.0923	0.1700
dt	Decision Tree Regressor	0.9336	1.6092	1.2321	0.9283	0.1167	0.0949	0.0280
lightgbm	Light Gradient Boosting Machine	0.9731	1.8903	1.3323	0.9196	0.1434	0.1178	0.2870
knn	K Neighbors Regressor	1.2407	2.7481	1.6193	0.8813	0.1221	0.1120	0.0340
llar	Lasso Least Angle Regression	1.3834	3.3047	1.7501	0.8674	0.1721	0.1612	0.0270
lasso	Lasso Regression	1.3834	3.3047	1.7500	0.8674	0.1721	0.1612	0.0520
en	Elastic Net	1.3842	3.3202	1.7524	0.8669	0.1734	0.1621	0.0260
lr	Linear Regression	1.3846	3.3397	1.7555	0.8663	0.1752	0.1634	0.5350
ridge	Ridge Regression	1.3846	3.3397	1.7555	0.8663	0.1752	0.1634	0.0260
lar	Least Angle Regression	1.3846	3.3397	1.7555	0.8663	0.1752	0.1634	0.0280
br	Bayesian Ridge	1.3888	3.3451	1.7580	0.8659	0.1743	0.1631	0.0270
huber	Huber Regressor	1.3405	3.4828	1.7876	0.8618	0.1780	0.1661	0.0350
omp	Orthogonal Matching Pursuit	2.6871	11.2996	3.3293	0.4990	0.2264	0.2269	0.0290
dummy	Dummy Regressor	4.3561	27.7343	5.1586	-0.0588	0.3803	0.4259	0.0300
par	Passive Aggressive Regressor	5.0575	56.0901	6.1967	-1.0559	0.3472	0.3877	0.0270

```
ExtraTreesRegressor
ExtraTreesRegressor(n_jobs=-1, random_state=123)
```

```
# create Extra Trees Regressor Model
et = create_model('et')
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE

```
et = finalize_model(et)
et
```




```
#Model Prediction
preds = predict_model(et)
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Extra Trees Regressor	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000

```
#predicted Values
preds
```

	TV	Radio	Newspaper	Sales	prediction_label
50	199.800003	3.100000	34.599998	11.400000	11.400000
127	80.199997	0.000000	9.200000	8.800000	8.800000
37	74.699997	49.400002	45.700001	14.700000	14.700000
149	44.700001	25.799999	20.600000	10.100000	10.100000
19	147.300003	23.900000	19.100000	14.600000	14.600000
104	238.199997	34.299999	5.300000	20.700001	20.700001
179	165.600006	10.000000	17.600000	12.600000	12.600000
53	182.600006	46.200001	58.700001	21.200001	21.200001
162	188.399994	18.100000	25.600000	14.900000	14.900000
158	11.700000	36.900002	45.200001	7.300000	7.300000
82	75.300003	20.299999	32.500000	11.300000	11.300000
185	205.000000	45.099998	19.600000	22.600000	22.600000
182	56.200001	5.700000	29.700001	8.700000	8.700000
189	18.700001	12.100000	23.400000	6.700000	6.700000
108	13.100000	0.400000	25.600000	5.300000	5.300000
31	112.900002	17.400000	38.599998	11.900000	11.900000
4	180.800003	10.800000	58.400002	12.900000	12.900000
178	276.700012	2.300000	23.700001	11.800000	11.800000
121	18.799999	21.700001	50.400002	7.000000	7.000000
20	218.399994	27.700001	53.400002	18.000000	18.000000
172	19.600000	20.100000	17.000000	7.600000	7.600000
88	88.300003	25.500000	73.400002	12.900000	12.900000
166	17.900000	37.599998	21.600000	8.000000	8.000000
170	50.000000	11.600000	18.400000	8.400000	8.400000
128	220.300003	49.000000	3.200000	24.700001	24.700001
72	26.799999	33.000000	19.299999	8.800000	8.800000
180	156.600006	2.600000	8.300000	10.500000	10.500000
26	142.899994	29.299999	12.600000	15.000000	15.000000
144	96.199997	14.800000	38.900002	11.400000	11.400000
52	216.399994	41.700001	39.599998	22.600000	22.600000
79	116.000000	7.700000	23.100000	11.000000	11.000000
93	250.899994	36.500000	72.300003	22.200001	22.200001
183	287.600006	43.000000	71.800003	26.200001	26.200001
119	19.400000	16.000000	22.299999	6.600000	6.600000
85	193.199997	18.400000	65.699997	15.200000	15.200000

```
sales_prediction=preds
```

```
sales_prediction.head()
```

	TV	Radio	Newspaper	Sales	prediction_label
50	199.800003	3.100000	34.599998	11.4	11.4
127	80.199997	0.000000	9.200000	8.8	8.8
37	74.699997	49.400002	45.700001	14.7	14.7
149	44.700001	25.799999	20.600000	10.1	10.1
19	147.300003	23.900000	19.100000	14.6	14.6

```
sales_prediction= sales_prediction.rename(columns={'prediction_label': 'Sales_Prediction'}).reset_index()
```

```
sales_prediction
```