```
import pandas as pd
df=pd.read_csv('/content/dafeml.csv')
df1=pd.read_csv('/content/dafeml.csv')
df.head()
```

	Brand_name	Year	TV	Radio	Social Media	Influencer	Sponsering	location	F
0	B1	2001	16.0	6.566231	2.907983	0.10	4.0	Urban	
1	B3	2001	13.0	9.237765	2.409567	0.23	2.0	Rural	
2	B2	2001	41.0	15.886446	2.913410	0.40	10.0	Rural	1
3	B5	2001	83.0	30.020028	6.922304	NaN	4.0	Urban	2

```
#null value -columwise
df.isna().sum()
```

```
Brand_name       0
Year             0
TV              21
Radio           15
Social Media    17
Influencer      21
Sponsering      11
location        11
Pre_Sales       17
Aft_sales       17
dtype: int64
```

```
df.columns
```

```
Index(['Brand_name', 'Year', 'TV', 'Radio', 'Social Media', 'Influencer',
       'Sponsering', 'location', 'Pre_Sales', 'Aft_sales'],
      dtype='object')
```

```
col=df.columns.values
```

```
for all in col:
  print(f"\033[1m{all}")
  print("--------------")
  print(df[all].value_counts())
```

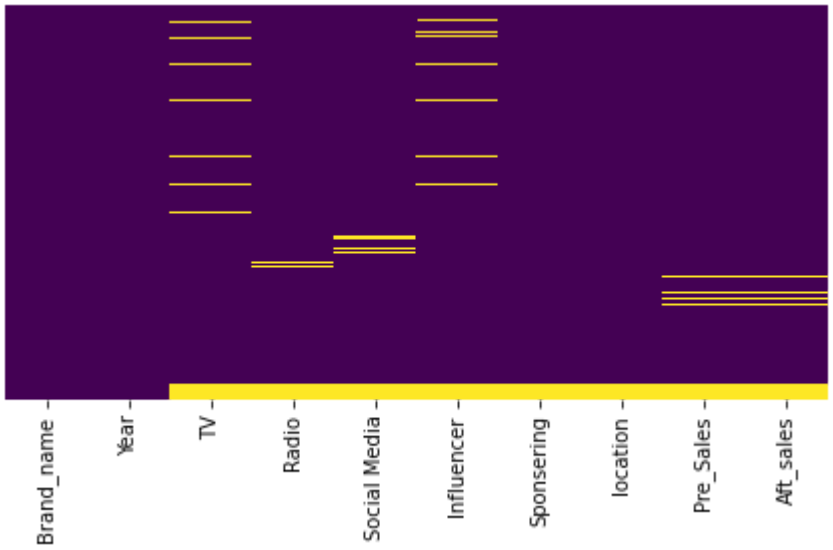
```
Brand_name
--------------
```

```
B1     31
B3     31
B2     31
B5     31
B7     31
B8     31
B9     31
B10    31
B6     31
B4     31
Name: Brand_name, dtype: int64
Year
--------------
2009    42
2006    31
2001    30
2002    30
2003    30
2004    30
2010    30
2005    29
2007    29
2008    29
Name: Year, dtype: int64
TV
--------------
49.0    7
27.0    7
78.0    7
40.0    6
63.0    6
       ..
51.0    1
61.0    1
71.0    1
28.0    1
50.0    1
Name: TV, Length: 87, dtype: int64
Radio
--------------
6.566231     1
30.498157    1
26.192673    1
27.148173    1
13.221237    1
            ..
11.213689    1
24.359044    1
3.934783     1
38.415225    1
25.583440    1
Name: Radio, Length: 295, dtype: int64
Social Media
--------------
2.907983    1
0.962942    1
```

Finding the null values in the dataset using isnull() function in Heatmap function

```
import matplotlib.pyplot as plt
import seaborn as sns
def get_heatmap(df):
    #This function gives heatmap of all NaN values
    plt.figure(figsize=(6,4))
    sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
    plt.tight_layout()
    return plt.show()

get_heatmap(df)
```



Dropping the null values using Dropna()

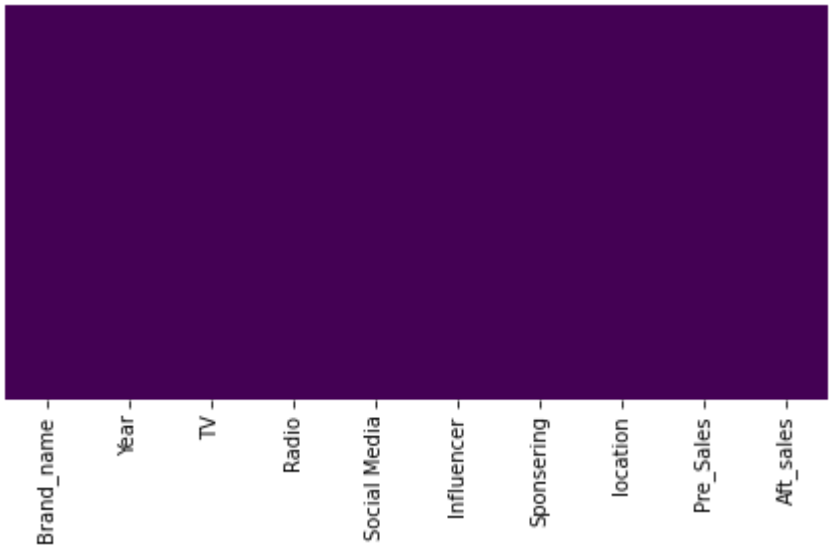
```
df = df.dropna()
df.head()
```

	Brand_name	Year	TV	Radio	Social Media	Influencer	Sponsering	location	Pre_Sa
0	B1	2001	16.0	6.566231	2.907983	0.10	4.0	Urban	54.732
1	B3	2001	13.0	9.237765	2.409567	0.23	2.0	Rural	46.677
2	B2	2001	41.0	15.886446	2.913410	0.40	10.0	Rural	150.177
4	B7	2001	15.0	8.437408	1.405998	0.03	6.0	Rural	56.594

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

```
def get_heatmap(df):
    #This function gives heatmap of all NaN values
    plt.figure(figsize=(6,4))
    sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
    plt.tight_layout()
    return plt.show()

get_heatmap(df)
```



Checking number of null values after dropping them

```
#null value -columwise
df.isna().sum()
```

```
Brand_name      0
Year            0
TV              0
Radio           0
Social Media    0
Influencer      0
Sponsering      0
location        0
Pre_Sales       0
Aft_sales       0
dtype: int64
```

```
df.head()
```

	Brand_name	Year	TV	Radio	Social Media	Influencer	Sponsering	location	Pre_Sa
0	B1	2001	16.0	6.566231	2.907983	0.10	4.0	Urban	54.732

```
df.Brand_name.value_counts()
```

```
B10    29
B6     29
B1     28
B8     28
B2     27
B3     26
B7     26
B4     26
B5     25
B9     23
Name: Brand_name, dtype: int64
```

***As both "BRAND_NAME" and "Location" are in text format we are using "LABEL ENCODER" to convert into numeric ***

```
#data processing
from sklearn import preprocessing

#labelencodingg
LE=preprocessing.LabelEncoder()

#fitting it to our dataset
df.Brand_name=LE.fit_transform(df.Brand_name)
df.head(5)
```

	Brand_name	Year	TV	Radio	Social Media	Influencer	Sponsering	location	Pre_Sa
0	0	2001	16.0	6.566231	2.907983	0.10	4.0	Urban	54.732
1	3	2001	13.0	9.237765	2.409567	0.23	2.0	Rural	46.677
2	2	2001	41.0	15.886446	2.913410	0.40	10.0	Rural	150.177
4	7	2001	15.0	8.437408	1.405998	0.03	6.0	Rural	56.594

We can se that the brand names are converted into numeric values

```
#data processing
from sklearn import preprocessing

#labelencodingg
LE=preprocessing.LabelEncoder()
```

```
#fitting it to our dataset
df.location=LE.fit_transform(df.location)
df.head(5)
```

	Brand_name	Year	TV	Radio	Social Media	Influencer	Sponsering	location	Pre_Sa
0	0	2001	16.0	6.566231	2.907983	0.10	4.0	1	54.732
1	3	2001	13.0	9.237765	2.409567	0.23	2.0	0	46.677
2	2	2001	41.0	15.886446	2.913410	0.40	10.0	0	150.177
4	7	2001	15.0	8.437408	1.405998	0.03	6.0	0	56.594

Urban and Rural are changed into numeric

```
df.location.value_counts()
```

```
1    141
0    126
Name: location, dtype: int64
```

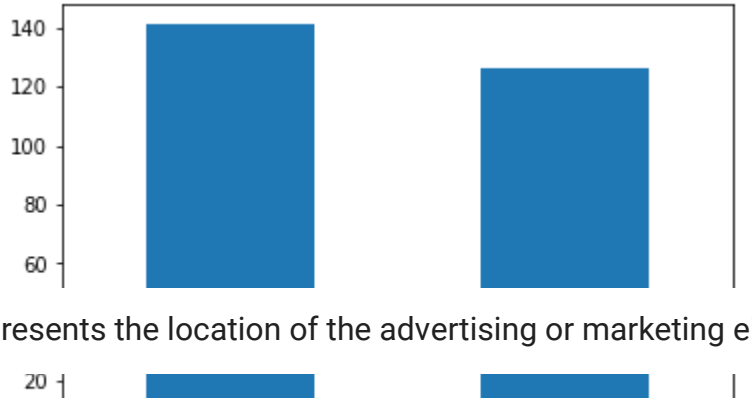
```
#finding the datatypes of Attributes
df.dtypes
```

```
Brand_name        int64
Year              int64
TV              float64
Radio           float64
Social Media    float64
Influencer      float64
Sponsering      float64
location          int64
Pre_Sales       float64
Aft_sales       float64
dtype: object
```

____Visualization____

```
df.location.value_counts().plot.bar()
```

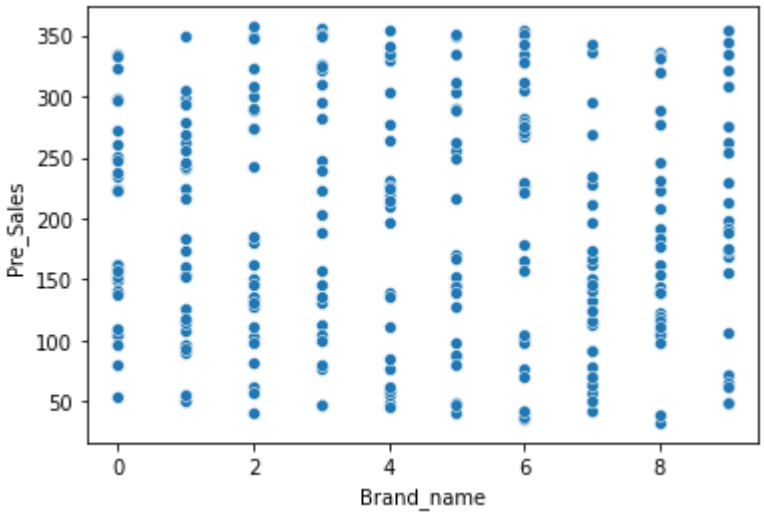
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa38dd6f210>
```



It, represents the location of the advertising or marketing either it is Rural or Urban

```
#scatterplot
sns.scatterplot(x=df.Brand_name,y = df1.Pre_Sales)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa38dcfc410>
```



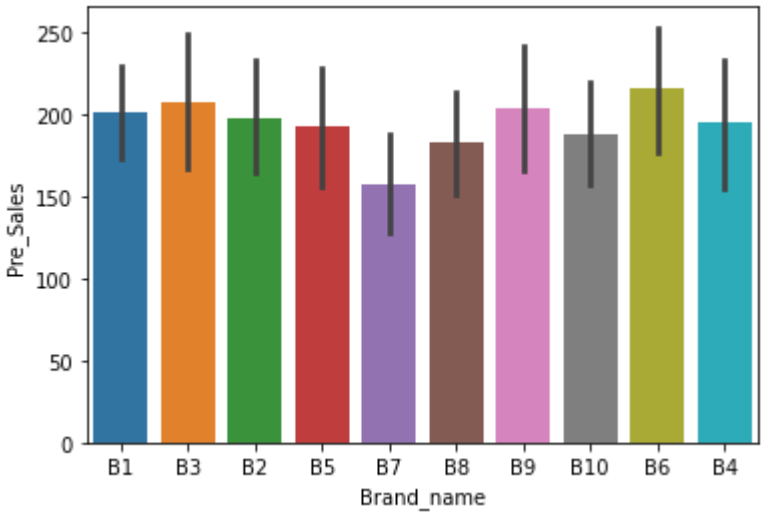
This graph helps us to know the sales of a particular brand when it was not advertised

```
#scatterplot
sns.scatterplot(x=df.Brand_name,y = df1.Aft_sales)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa38dc81590>
```

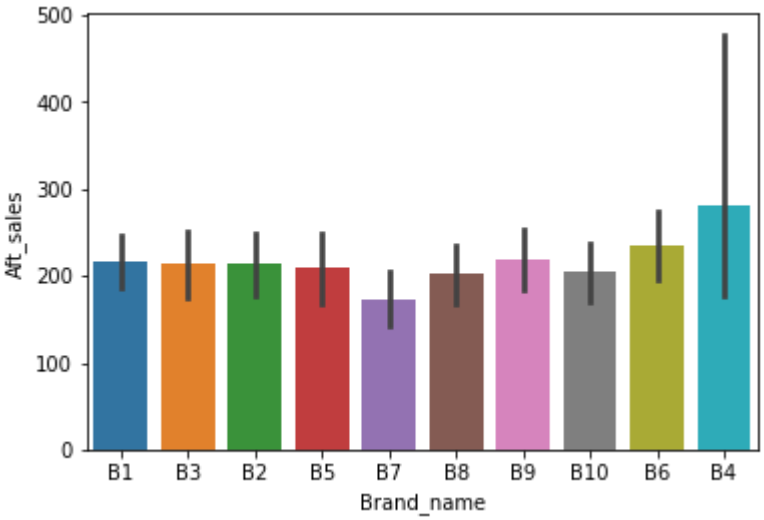
This graph helps us to know the sales of a particular brand when it is advertised

```
sns.barplot(x='Brand_name',y='Pre_Sales',data=df1);
```



It shows the sales of particular brand and it's sales before marketing

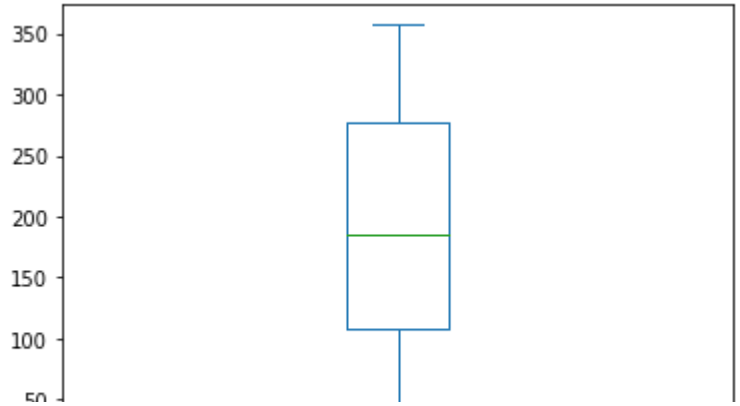
```
sns.barplot(x='Brand_name',y='Aft_sales',data=df1);
```



It shows the sales of particular brand and it's sales after marketing

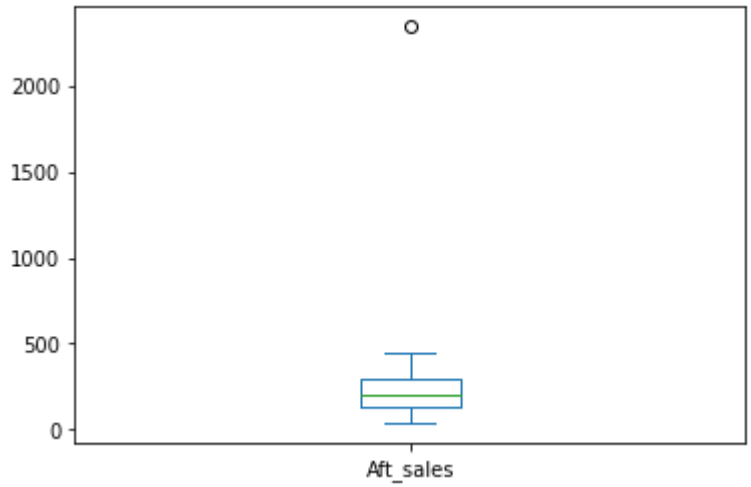
```
df1.Pre_Sales.plot.box()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa38da4f610>
```

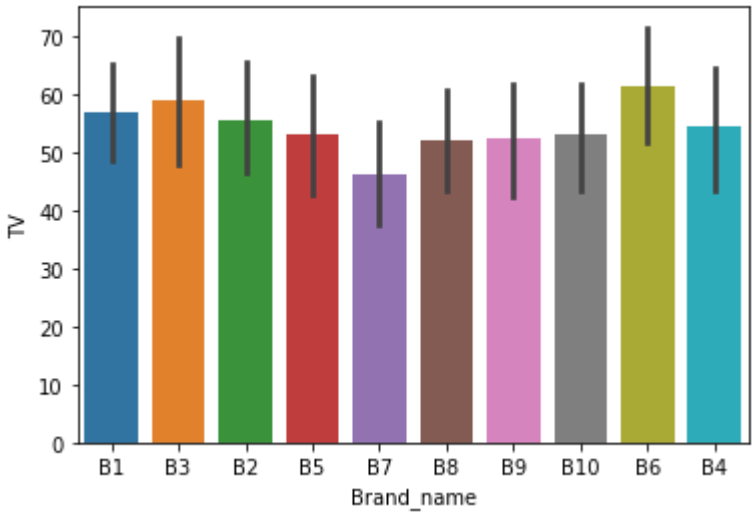



```
df1.Aft_sales.plot.box()
```

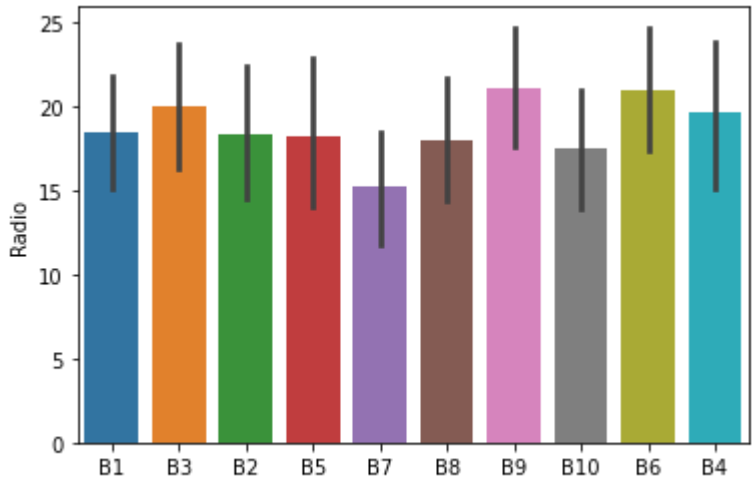
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa391bc3dd0>
```



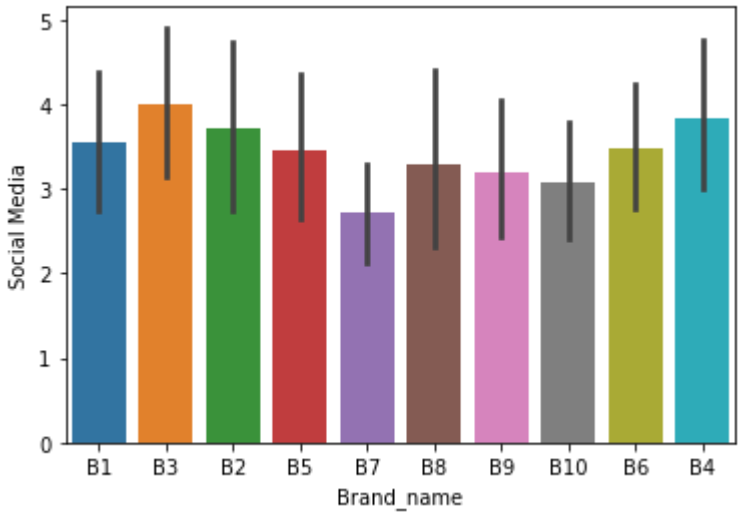

```
sns.barplot(x='Brand_name',y='TV',data=df1);
```



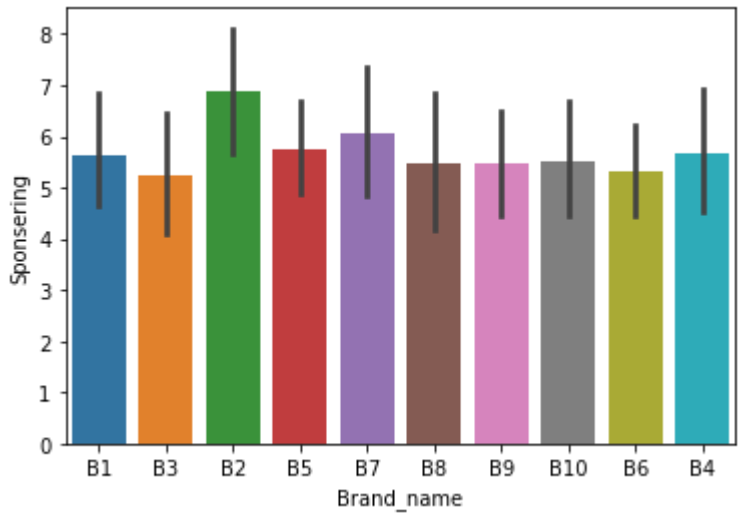

```
sns.barplot(x='Brand_name',y='Radio',data=df1);
```



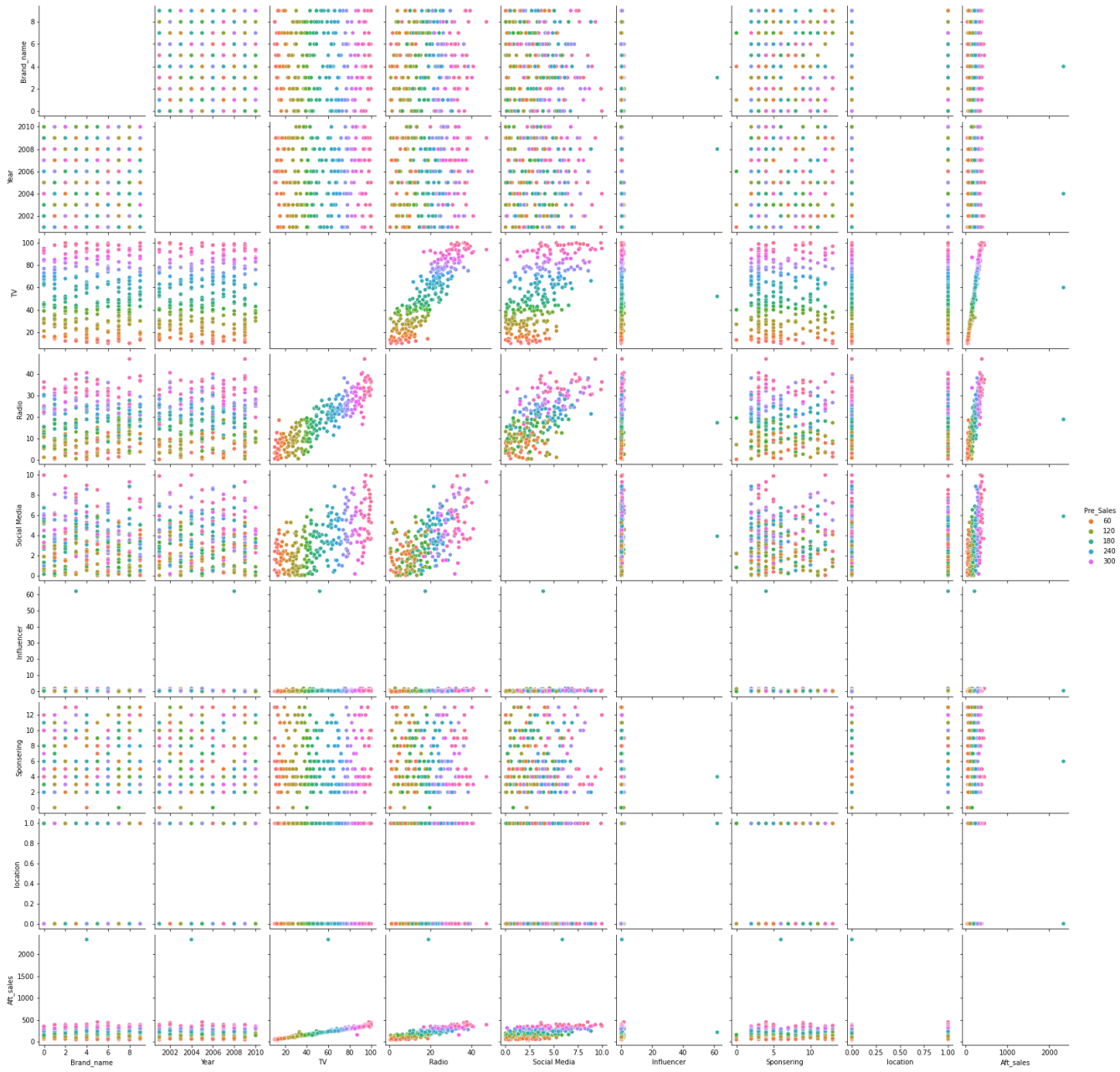

```
sns.barplot(x='Brand_name',y='Social Media',data=df1);
```




```
sns.barplot(x='Brand_name',y='Sponsering',data=df1);
```



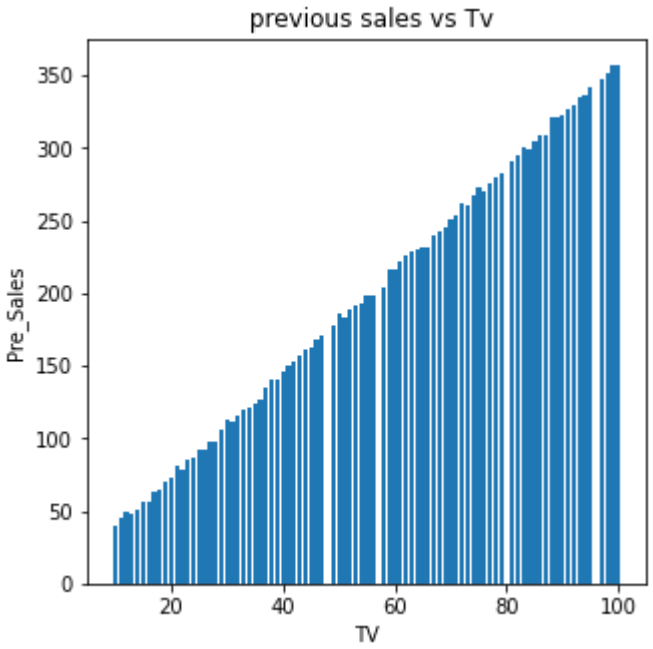

```
sns.pairplot(df,hue = 'Pre_Sales',diag_kind = "kde",kind = "scatter",palette = "husl")
plt.show()
```



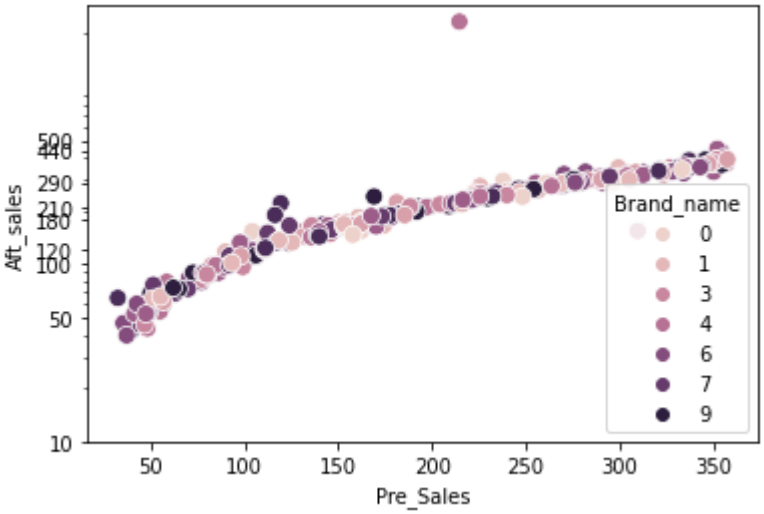
This is pairplot where the selecte dattribute is compared to all other attributes from the datasets

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(5,5))
plt.title(' previous sales vs Tv') # add title
plt.xlabel("TV")  # adds x -axis label
plt.ylabel("Pre_Sales")  # adds y -axis label
plt.bar(df1.TV,df1.Pre_Sales)   # generates bar graph
plt.show()
```

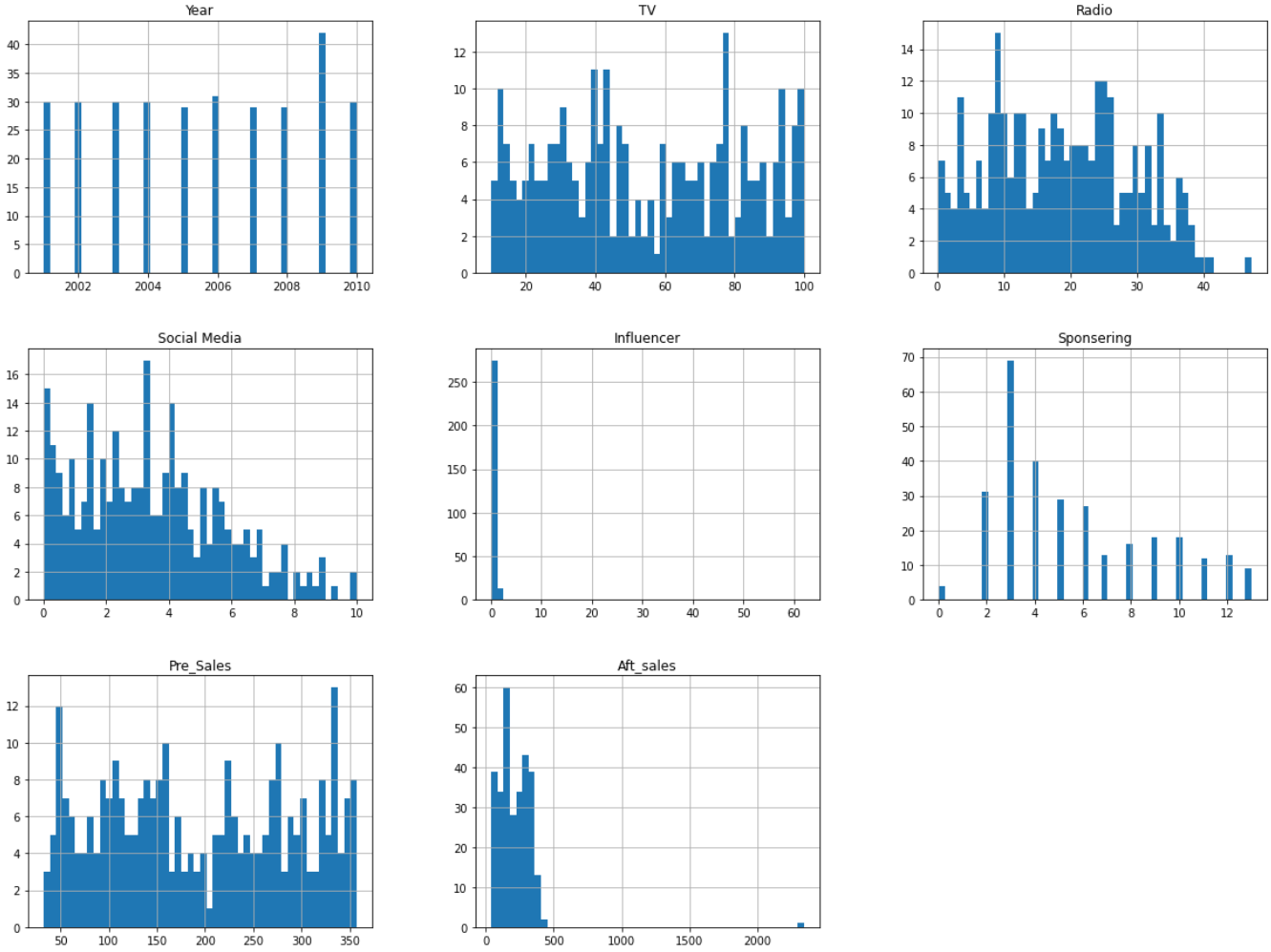



```
import matplotlib.pyplot as plt
from matplotlib.ticker import FormatStrFormatter
ax=sns.scatterplot(x='Pre_Sales',y = 'Aft_sales',hue="Brand_name",data = df,s=80)
ax.set_yscale('log')
ax.set_yticks([10,50,100,120,180,210,290,440,500])
# ax.set_yticklabels([0.03, 0.1, 0.3, 1, 3, 10, 30, 100, 300, 1000])
ax.yaxis.set_major_formatter(FormatStrFormatter('%g'))
plt.show()
```



here, the comparision between previous sales and after sales of all the brand names

```
df1.hist(bins=50,figsize=(20,15))
plt.show()
```



here each groups numbers into ranges

```
X=df.iloc[:,:-1].values
Y=df.iloc[:,1:].values
```

```
import numpy as np
train_x=np.array(df[['Brand_name','Year','TV','Radio','Social Media','Influencer','Sponsering
train_y=np.array(df[['Aft_sales']])
```

```
print(train_x)
```

```
[[0.00000000e+00 2.00100000e+03 1.60000000e+01 ... 4.00000000e+00
  1.00000000e+00 5.47327571e+01]
 [3.00000000e+00 2.00100000e+03 1.30000000e+01 ... 2.00000000e+00
  0.00000000e+00 4.66778970e+01]
 [2.00000000e+00 2.00100000e+03 4.10000000e+01 ... 1.00000000e+01
  0.00000000e+00 1.50177829e+02]
 ...
 [9.00000000e+00 2.00900000e+03 7.70000000e+01 ... 3.00000000e+00
  0.00000000e+00 2.75388673e+02]
 [1.00000000e+00 2.00900000e+03 2.70000000e+01 ... 3.00000000e+00
  0.00000000e+00 9.33517594e+01]
 [6.00000000e+00 2.00900000e+03 7.70000000e+01 ... 6.00000000e+00
  0.00000000e+00 2.76083898e+02]]
```

```
print(train_y)
```

```
[[  60.3224 ]
 [  58.769  ]
 [ 172.334  ]
 [  59.342  ]
 [ 111.324  ]
 [ 132.331  ]
 [ 323.443  ]
 [  43.441  ]
 [ 259.228  ]
 [ 148.442  ]
 [ 132.245  ]
 [ 222.231  ]
 [ 117.99   ]
 [ 432.442  ]
 [ 342.241  ]
 [ 154.456  ]
 [ 142.22   ]
 [ 332.321  ]
 [ 212.227  ]
 [ 112.165  ]
 [  79.332  ]
 [ 150.999  ]
 [ 246.332  ]
 [ 320.465  ]
 [  52.331  ]
 [ 288.33   ]
 [ 162.332  ]
 [ 170.117  ]
 [ 169.669  ]
 [ 289.889  ]
 [ 333.332  ]
 [ 241.441  ]
 [ 163.663  ]
 [ 287.887  ]
```

```
[ 253.324  ]
[ 448.567  ]
[ 112.115  ]
[  64.66453]
[ 132.576  ]
[ 132.334  ]
[ 133.56987]
[ 132.11243]
[ 352.117  ]
[ 332.3214 ]
[ 289.453  ]
[ 179.114  ]
[  42.4421 ]
[ 241.521  ]
[  59.765  ]
[  83.662  ]
[  79.753  ]
[ 296.553  ]
[ 332.632  ]
[ 224.33786]
[ 163.876  ]
[ 254.445  ]
[ 312.542  ]
[  68.664  ]
```

```
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,random_state=0)
```

```
df.isnull().any()
```

```
Brand_name      False
Year            False
TV              False
Radio           False
Social Media    False
Influencer      False
Sponsering      False
location        False
Pre_Sales       False
Aft_sales       False
dtype: bool
```

```
#training the model
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

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

```
LinearRegression()
```

```
#testing the model
y_pred = model.predict(x_test)
```

```
#Finding the accuracy - How Accurate the model is?
import numpy as np
print("Mean Square Error(MSE): %.2f" % np.mean(((y_pred - y_test)** 2)**0.5))
```

```
Mean Square Error(MSE): 2.84
```

```
print("Predicted value for training data:",model.score(x_train,y_train))
print("Training accuracy:",model.score(x_train,y_train)*100)
```

```
Predicted value for training data: 0.9264212351585284
Training accuracy: 92.64212351585283
```

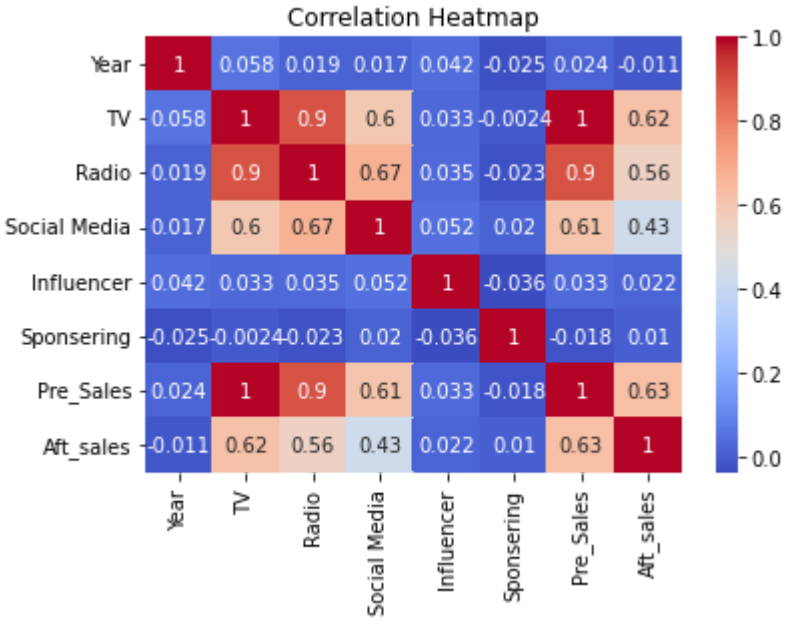
```
print("Predicted value for training data:",model.score(x_test,y_test))
print("Training accuracy:",model.score(x_test,y_test)*100)
```

```
Predicted value for training data: 0.9574148222675143
Training accuracy: 95.74148222675143
```

```
accuracy=model.score(X,Y)*100
print(accuracy)
```

```
92.86935993439064
```

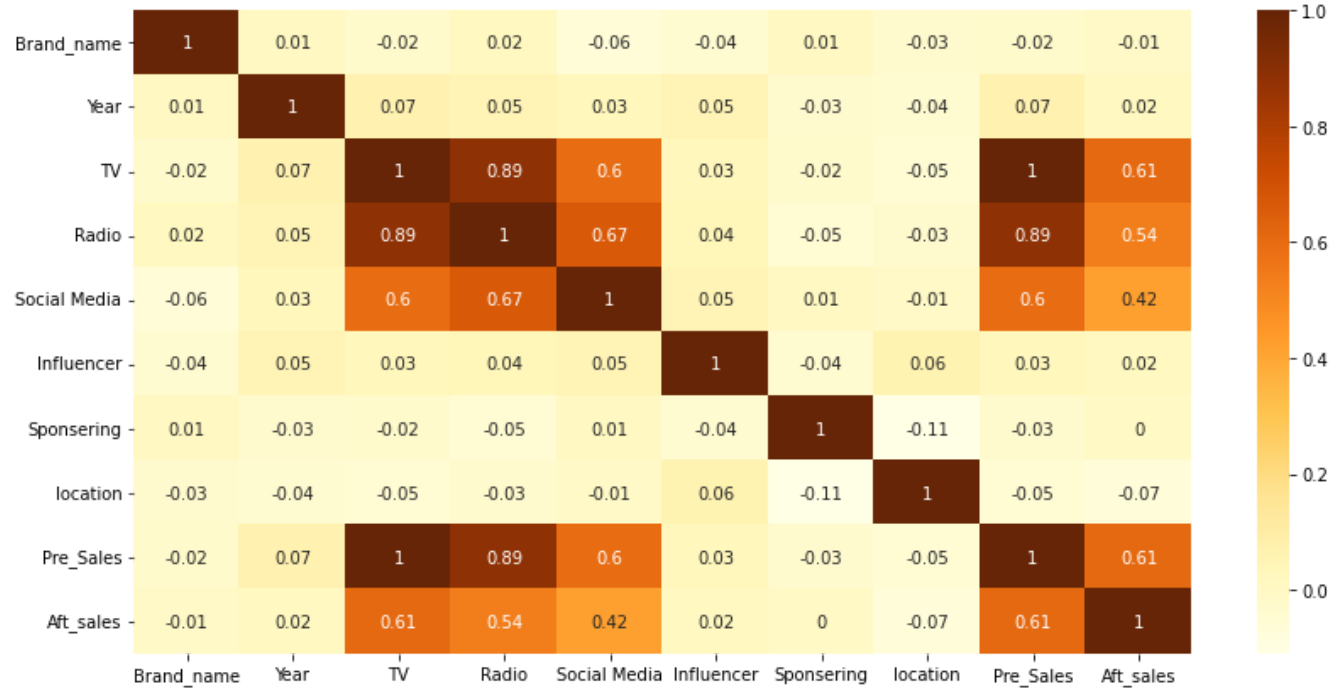
```
sns.heatmap(df1.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```




```
correlation = df.corr().round(2)
plt.figure(figsize = (14,7))
```

```
sns.heatmap(correlation, annot = True, cmap = 'YlOrBr')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa38db0be90>
```



This shows that, dark orange ->will have high corr light orange ->2nd highest

```
y_pred=model.predict(x_test)
y_pred
```

```
array([[ 2.01000000e+03,  7.60000000e+01,  2.61926733e+01,
         4.12760090e+00,  5.20000000e-01,  3.00000000e+00,
         8.04298590e-16,  2.69603006e+02,  2.93063652e+02],
       [ 2.00900000e+03,  3.60000000e+01,  9.09677989e+00,
         3.81883449e+00,  4.10000000e-01,  3.00000000e+00,
         1.00000000e+00,  1.27092674e+02,  1.49020364e+02],
       [ 2.00400000e+03,  7.10000000e+01,  2.60239494e+01,
         3.11642848e+00,  2.40000000e-01,  6.00000000e+00,
         7.88620940e-16,  2.53986110e+02,  2.88797808e+02],
       [ 2.00200000e+03,  6.20000000e+01,  2.43451888e+01,
         5.15148317e+00,  2.50000000e-01,  2.00000000e+00,
         1.13811005e-15,  2.24961019e+02,  2.78841989e+02],
       [ 2.00900000e+03,  6.00000000e+01,  1.94999565e+01,
         4.91233617e+00,  6.20000000e-01,  3.00000000e+00,
         1.00000000e+00,  2.16268157e+02,  2.40996164e+02],
       [ 2.00600000e+03,  1.20000000e+01,  1.28482798e+01,
         2.81037534e+00,  2.50000000e-01,  6.00000000e+00,
         1.00000000e+00,  4.26515220e+01,  5.24289357e+01],
       [ 2.01000000e+03,  6.30000000e+01,  2.60359863e+01,
         3.29081689e+00,  6.90000000e-01,  1.20000000e+01,
         6.40195693e-17,  2.25486232e+02,  2.39150453e+02],
       [ 2.01000000e+03,  4.30000000e+01,  9.72673605e+00,
```

```
              3.37998227e-01,  9.00000000e-02,  6.00000000e+00,
             -1.72290162e-16,  1.52254668e+02,  1.64731921e+02],
            [ 2.00300000e+03,  3.10000000e+01,  8.84392465e+00,
              1.04810858e-01,  7.00000000e-02,  2.00000000e+00,
              1.00000000e+00,  1.11127749e+02,  1.19340390e+02],
            [ 2.00600000e+03,  4.70000000e+01,  2.41691319e+01,
              2.08791026e+00,  5.10000000e-01,  1.00000000e+01,
              1.00000000e+00,  1.61717087e+02,  1.51669744e+02],
            [ 2.00800000e+03,  2.70000000e+01,  8.30139310e+00,
              2.97735659e-01,  3.70000000e-01,  6.00000000e+00,
              1.00000000e+00,  9.81619220e+01,  9.60999451e+01],
            [ 2.00700000e+03,  9.90000000e+01,  3.01405691e+01,
              6.54615802e+00,  6.20000000e-01,  1.00000000e+01,
              1.00000000e+00,  3.51359890e+02,  3.87304428e+02],
            [ 2.00700000e+03,  9.40000000e+01,  2.93446099e+01,
              3.43648101e+00,  7.20000000e-01,  2.00000000e+00,
              1.64235455e-15,  3.36372616e+02,  3.64427589e+02],
            [ 2.00600000e+03,  4.90000000e+01,  2.04236547e+01,
              4.34677736e+00,  3.20000000e-01,  7.00000000e+00,
              1.00000000e+00,  1.78374677e+02,  2.05484915e+02],
            [ 2.00400000e+03,  4.40000000e+01,  1.66093140e+01,
              5.15906663e+00,  3.50000000e-01,  4.00000000e+00,
              2.42531405e-16,  1.57490072e+02,  2.10544056e+02],
            [ 2.00300000e+03,  4.60000000e+01,  1.26950923e+01,
              3.88458384e+00,  1.60000000e+00,  5.00000000e+00,
              7.19645768e-17,  1.62191334e+02,  2.02696973e+02],
            [ 2.00500000e+03,  2.10000000e+01,  1.39737111e+00,
              1.44459113e+00,  6.00000000e-02,  1.00000000e+01,
             -5.12515256e-16,  7.75479477e+01,  1.22880473e+02],
            [ 2.00700000e+03,  6.90000000e+01,  2.32904511e+01,
              5.67168608e+00,  6.30000000e-01,  8.00000000e+00,
              5.46051018e-16,  2.45979961e+02,  2.94381088e+02],
            [ 2.00600000e+03,  4.90000000e+01,  2.48710493e+01,
              5.03628143e+00,  4.10000000e-01,  4.00000000e+00,
              1.58654543e-16,  1.74986164e+02,  2.11941655e+02],
            [ 2.00700000e+03,  6.50000000e+01,  1.16564879e+01,
```

```
print('Coefficients: ', model.coef_)
```

```
Coefficients:  [[ 6.47636906e-16  1.00000000e+00 -1.66533454e-16  1.70002901e-16
  -7.52869989e-16 -2.29417180e-16  1.73472348e-17 -2.29850861e-16
  -3.33066907e-16]
 [ 2.83062545e-16 -2.10942375e-15  1.00000000e+00 -9.10729825e-17
   5.55111512e-16  4.06467394e-16 -7.40510084e-17  2.59883261e-16
  -1.11022302e-16]
 [ 3.86463507e-16  5.41233725e-16  1.22211269e-15  1.00000000e+00
   1.72062885e-16  1.73296165e-16  3.04931861e-17  4.34121326e-16
   3.42087469e-15]
 [ 9.57308286e-17 -3.71230824e-16  3.88578059e-16 -1.00939222e-16
   1.00000000e+00 -4.35063227e-16  2.19483177e-16  9.88656856e-17
   4.61436445e-16]
 [-1.15485744e-17  4.46691295e-17 -5.22043346e-16 -3.30275087e-17
  -3.07913417e-17  1.00000000e+00 -4.59057976e-17 -3.69081054e-16
   2.12069945e-16]
 [-2.71538042e-16  3.98986399e-17 -6.98226199e-17  2.28251662e-16
```

```
 -4.68294023e-16 -1.05293649e-15  1.00000000e+00 -2.85772823e-16
 -2.50233861e-16]
[-4.83379418e-18 -2.51534904e-17 -2.54353830e-16 -1.58903381e-17
  1.98680048e-17 -4.67494424e-17 -5.71408426e-17  1.00000000e+00
  8.02309608e-17]
[ 1.89541672e-15 -5.88418203e-15  3.88578059e-16  5.89805982e-16
 -1.52308721e-15 -6.28403579e-15  1.53956708e-16  2.74693462e-15
  1.00000000e+00]
[ 4.85657006e-01 -2.42765768e+00 -4.14770338e+00 -1.71712861e+00
  7.99483083e+00 -6.38541218e+00  2.79077342e-01 -1.62109638e+01
  2.26500427e+00]]
```