simplelinearregression

August 25, 2023

1 Simple Linear Regression

- Linear Regression is used for predicting the relationship between the independent and dependent variable. It aims to predict the good fit line that describes the relationship by minimizing the mean_squared_error of the predicted value and the actual value.
- In simple linear regression, we only use one independent variable or feature to predict the target variable.
- Slope-intercept formulae,
  Y=B0 + B1X where, y= target variable B0= intercept B1=slope/coefficient X=Predictive variable
- Linear regression aims to find the slope value and intercept value to predict the target feature.
- Have used the Tv-Marketing data set from kaggle.com; where, Independent variable(X)- TV and
  target variable(Y)- Sales.

[ ]:

```python
# Importing necessary libraries

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

[ ]:

```python
# Onboarding data

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

```
<IPython.core.display.HTML object>
```

```
Saving tvmarketing.csv to tvmarketing.csv
```

[ ]:

```python
# DataFrame

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

```
[ ]:        TV  Sales
     0    230.1   22.1
     1     44.5   10.4
     2     17.2    9.3
     3    151.5   18.5
     4    180.8   12.9
     ..      …     …
     195   38.2    7.6
     196   94.2    9.7
     197  177.0   12.8
     198  283.6   25.5
     199  232.1   13.4

     [200 rows x 2 columns]
```

```
[ ]: # Shallow copy

     df_copy=df.copy()
```

2 Exploratory Data Analysis

```
[ ]: df.head()
```

```
[ ]:      TV  Sales
     0  230.1   22.1
     1   44.5   10.4
     2   17.2    9.3
     3  151.5   18.5
     4  180.8   12.9
```

```
[ ]: df.shape
```

```
[ ]: (200, 2)
```

```
[ ]: df.columns
```

```
[ ]: Index(['TV', 'Sales'], dtype='object')
```

```
[ ]: # Technical Report

     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
```

```
---  ------  --------------  -----
 0   TV      200 non-null    float64
 1   Sales   200 non-null    float64
dtypes: float64(2)
memory usage: 3.2 KB
```

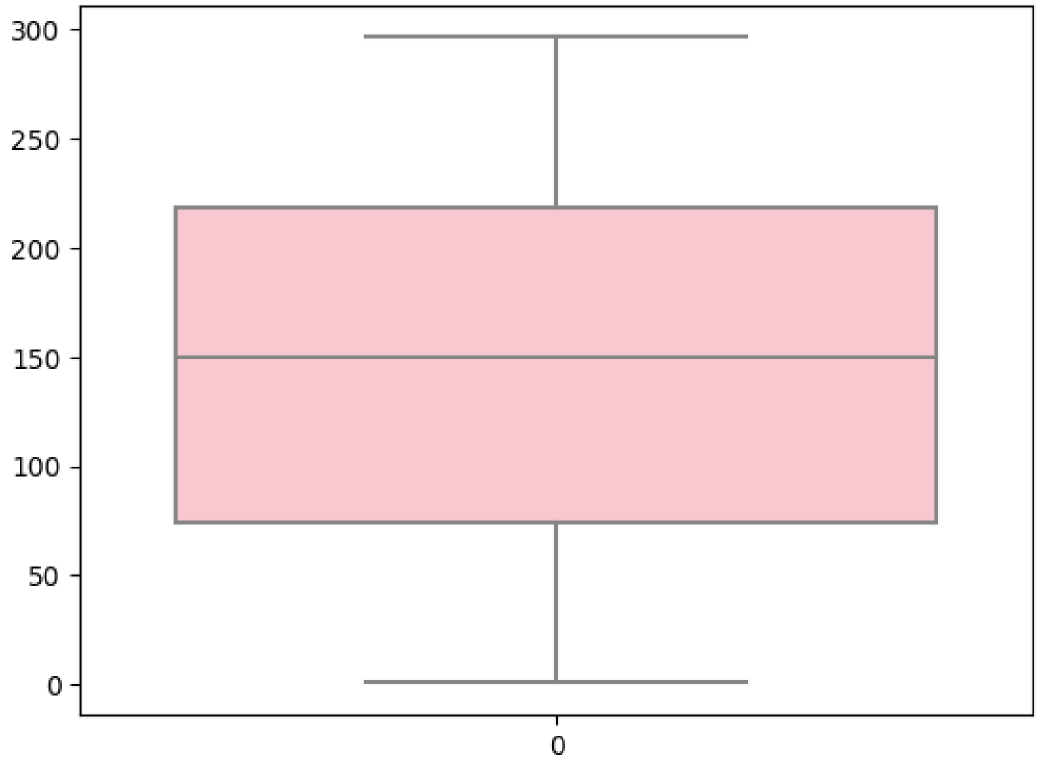
```
[ ]: # Statistical Report

     df.describe()
```

```
[ ]:                 TV       Sales
     count  200.000000  200.000000
     mean   147.042500   14.022500
     std     85.854236    5.217457
     min      0.700000    1.600000
     25%     74.375000   10.375000
     50%    149.750000   12.900000
     75%    218.825000   17.400000
     max    296.400000   27.000000
```

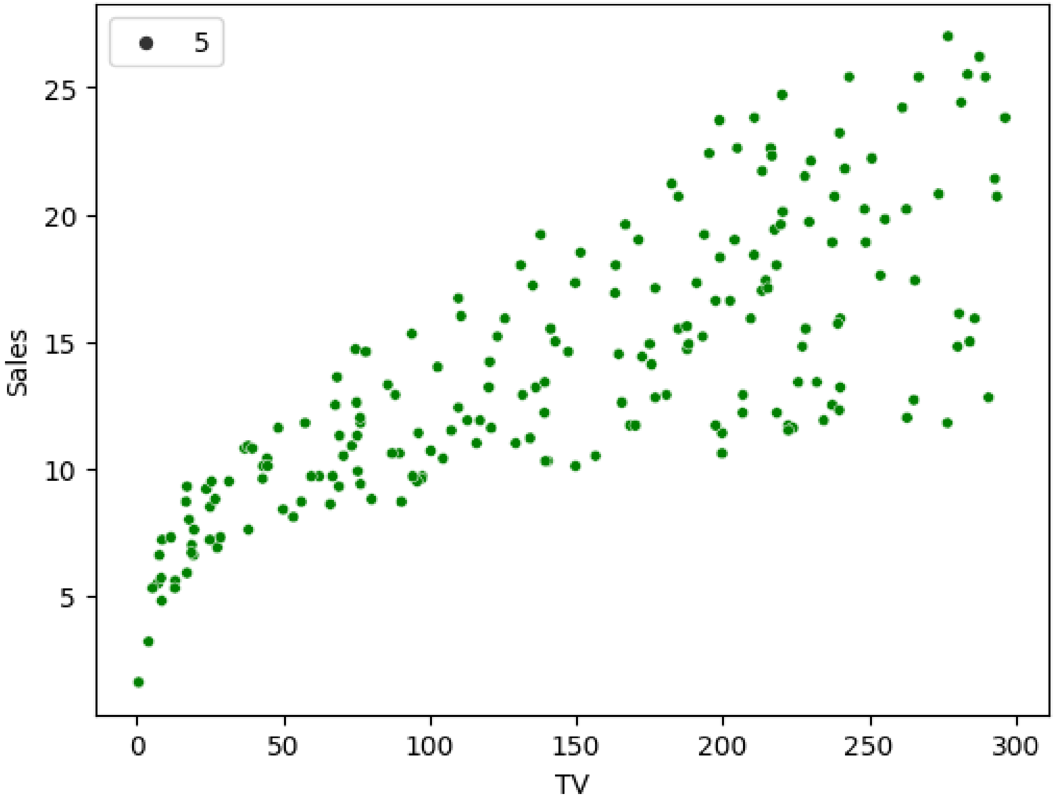
2.1 Univariate Analysis

```
[ ]: sns.boxplot(df['TV'],color='pink')
     plt.show()
```



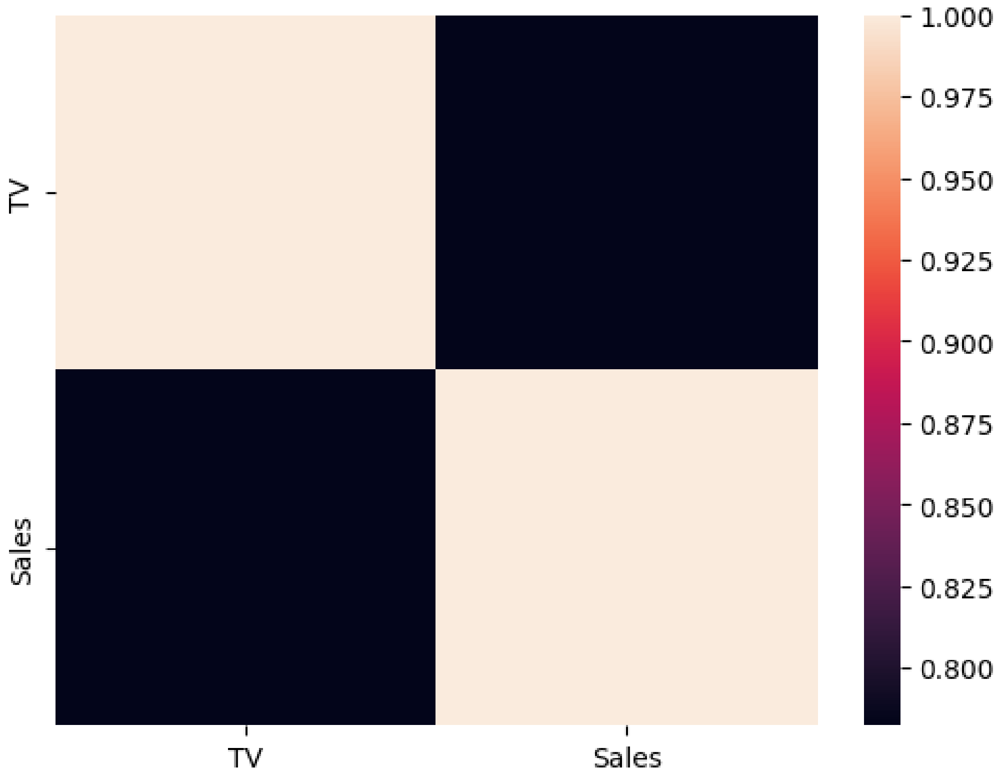
[ ]:

```
sns.scatterplot(df,x='TV',y='Sales',size=5,color='green')
plt.show()
```



[ ]:

```
sns.heatmap(df.corr())
plt.show()
```



```
[ ]: # Null values

     df.isna().sum()/len(df)*100
```

```
[ ]: TV       0.0
     Sales    0.0
     dtype: float64
```

```
[ ]: # Duplicate values

     df.duplicated().sum()
```

```
[ ]: 0
```

3 Standardization

```
[ ]: X=df.iloc[:,[0]]
     Y=df.iloc[:,[-1]]
```

```
[ ]: from sklearn.preprocessing import StandardScaler
```

```
sc=StandardScaler()
X_sc=sc.fit_transform(X)
```

4 Train Test Split

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

x_train,x_test,y_train,y_test=train_test_split(X_sc,Y,test_size=0.
 ↪3,random_state=4)
```

[ ]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
y_predict=lr.predict(x_test)
```

[ ]:
```
y_predict
```

[ ]:
```
array([[17.16400307],
       [13.33072137],
       [17.4340204 ],
       [20.1631241 ],
       [ 8.95740501],
       [18.46587233],
       [13.28732572],
       [15.01832966],
       [11.06932625],
       [16.0260729 ],
       [19.41093298],
       [18.00298548],
       [10.45214378],
       [12.74729107],
       [19.04930263],
       [20.60190226],
       [19.61826771],
       [14.7434906 ],
       [ 8.88507894],
       [17.10614222],
       [18.37425931],
       [20.32224146],
       [18.5381984 ],
       [11.42613486],
       [18.37908105],
       [17.87762029],
       [15.28352525],
       [ 9.58423094],
       [16.65289884],
       [ 8.71631811],
       [10.49553943],
       [10.14837429],
       [16.78308577],
       [15.25459483],
       [15.07136878],
       [ 7.74714877],
       [17.5352769 ],
       [10.10980039],
       [17.34722912],
       [12.10599992],
       [13.70199519],
       [ 8.59095292],
       [14.3625733 ],
       [ 7.37587494],
       [ 7.7133966 ],
       [12.10599992],
       [17.76672032],
       [13.52359089],
       [ 7.95930524],
       [17.26525957],
       [17.69921599],
       [15.86695555],
       [10.08086996],
       [13.91415166],
       [ 7.64107053],
       [11.85526954],
       [16.69629449],
       [10.21587862],
       [ 7.00942286],
       [ 9.81567437]])
```

```
[ ]: y_test
```

```
[ ]:      Sales
     11    17.4
     99    17.2
     128   24.7
     175   27.0
     1     10.4
     111   21.8
     90    11.2
     177   11.7
```

```
88     12.9
187    17.3
61     24.2
199    13.4
191     9.9
123    15.2
184    17.6
188    15.9
33     17.4
171    14.5
138     9.6
84     21.7
81     12.3
102    14.8
147    25.4
34      9.5
47     23.2
124    19.7
112    14.1
6      11.8
14     19.0
190    10.8
80     11.8
18     11.3
167    12.2
45     14.9
153    19.0
119     6.6
100    11.7
83     13.6
181    12.2
71     12.4
26     15.0
134    10.8
180    10.5
158     7.3
189     6.7
89     16.7
48     14.8
116    12.2
12      9.2
69     22.3
110    13.4
154    15.6
16     12.5
19     14.6
2       9.3
```

```
143    10.4
185    22.6
29     10.5
155     3.2
24      9.7
```

[ ]:
```
# Intercept value

lr.intercept_
```

[ ]:
```
array([13.90173569])
```

[ ]:
```
# Slope value

lr.coef_
```

[ ]:
```
array([[4.1293042]])
```

5 Model Evaluation

[ ]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score

MAE=mean_absolute_error(y_test,y_predict)
MSE=mean_squared_error(y_test,y_predict)
RMSE=np.sqrt(MSE)
print("MAE",MAE)
print("MSE",MSE)
print("RMSE",RMSE)
```

```
MAE 2.7404123217291905
MSE 11.404170638824256
RMSE 3.3770061650557075
```

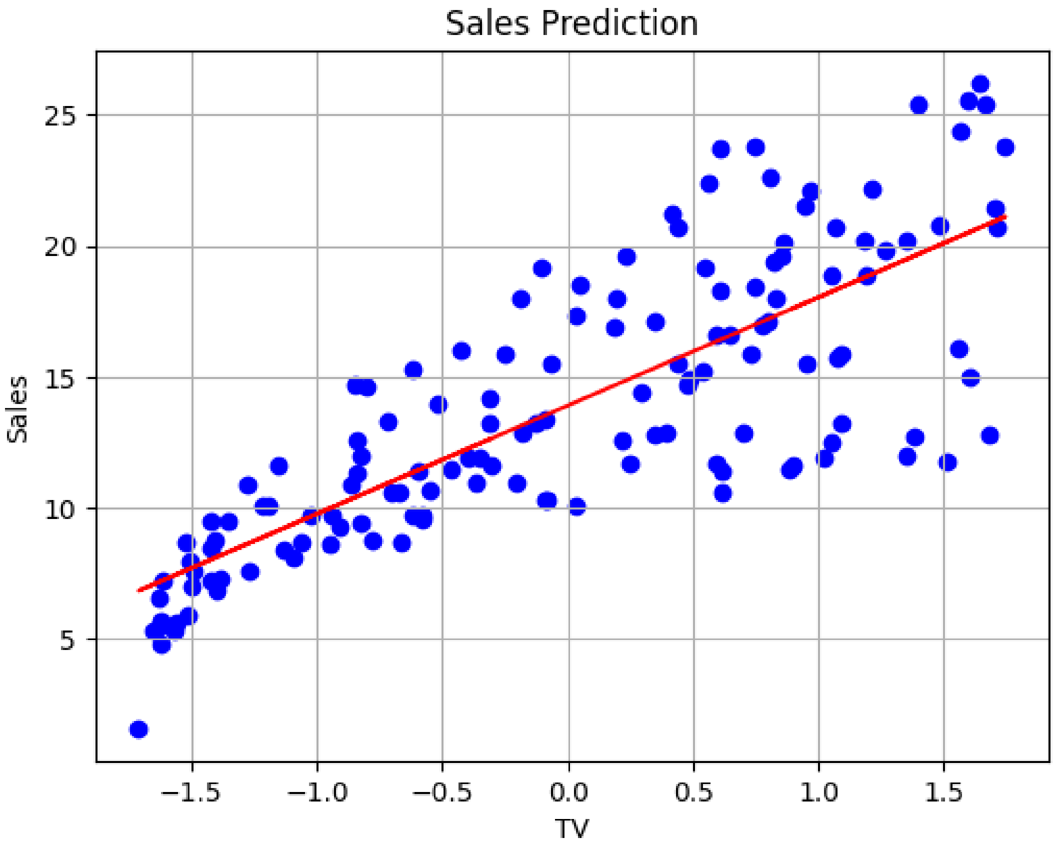
[ ]:
```
R2=r2_score(y_test,y_predict)
print("R2",R2)
```

```
R2 0.5524131166103536
```

The R2 score reveals that the model accuarcy is 55% (i.e) 55% of the data fits in the good fit line/regression line.

6 Data visualization

[ ]:
```
plt.scatter(x_train,y_train,color='blue')
plt.plot(x_train,lr.predict(x_train),color='red')
plt.title('Sales Prediction')
plt.xlabel('TV')
plt.ylabel('Sales')
plt.grid()
plt.show()
```


Sales Prediction
Sales
TV