# Tv Ads Linear Regression

March 28, 2023

## 1 Linear Regression

### 1.1 Simple Linear Regression

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

#### 1.1.1 Read TV Marketing Dataset

[2]:
```
df=pd.read_csv("tvmarketing.csv")
```

[3]:
```
df
```

[3]:
```
        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]
```

[4]:
```
df=df.rename(columns={"TV":"Marketing Budget"})
```

### 1.1.2 Summarizing the Data

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Marketing Budget  200 non-null    float64
 1   Sales             200 non-null    float64
dtypes: float64(2)
memory usage: 3.2 KB
```

### 1.1.3 Descriptive Summary of Data

[6]:
```
df.describe()
```

[6]:

| | Marketing Budget | 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 |

### 1.1.4 Check for Duplicate values

[7]:
```
df.duplicated().sum()
```

[7]: 0

### 1.1.5 Check for any null values

[8]:
```
df.isna().sum()
```

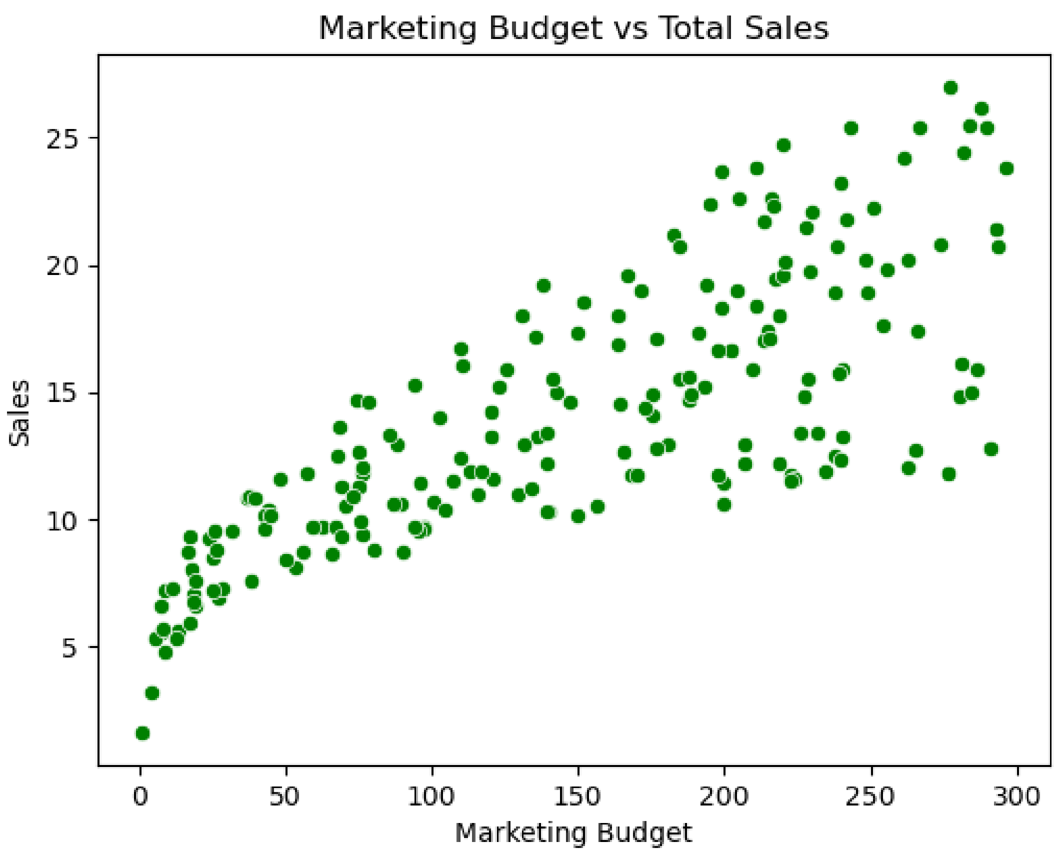
[8]:
```
Marketing Budget    0
Sales               0
dtype: int64
```

## 1.2 Visualize and Understand the Data

### 1.2.1 Scatterplot

[9]:
```
sns.scatterplot(x=df["Marketing Budget"],y=df["Sales"],c="g")
plt.title("Marketing Budget vs Total Sales")
```

[9]:
```
Text(0.5, 1.0, 'Marketing Budget vs Total Sales')
```



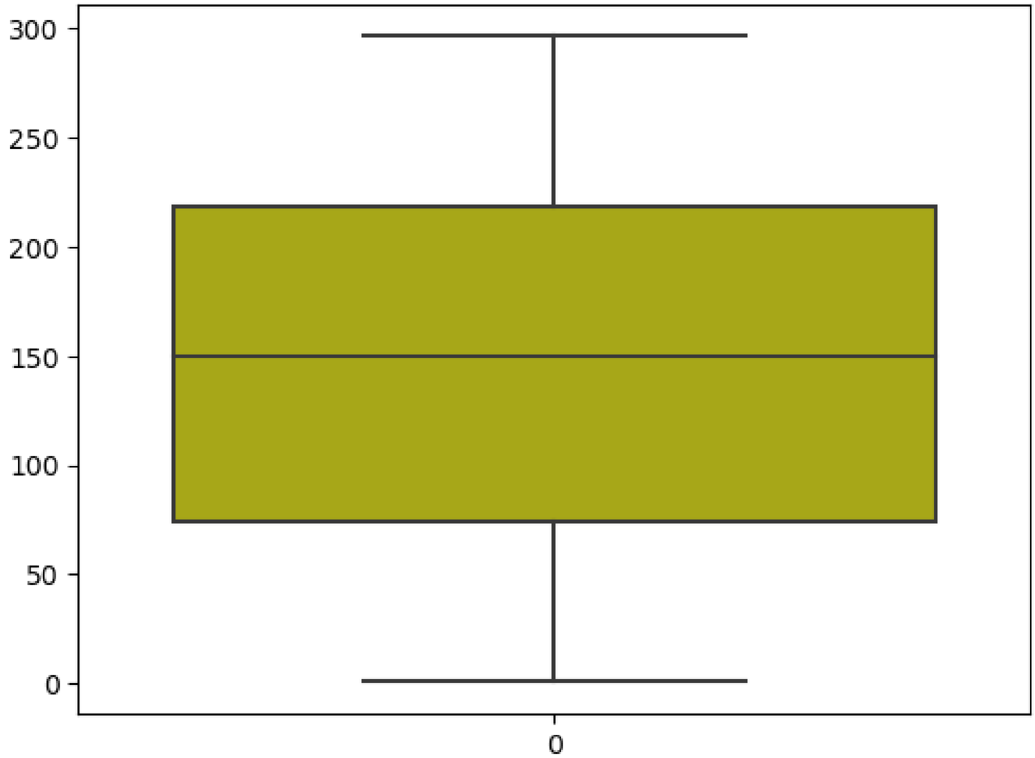

**Observation:**

- Given Data is Linear

### 1.2.2 Checking for Outliers

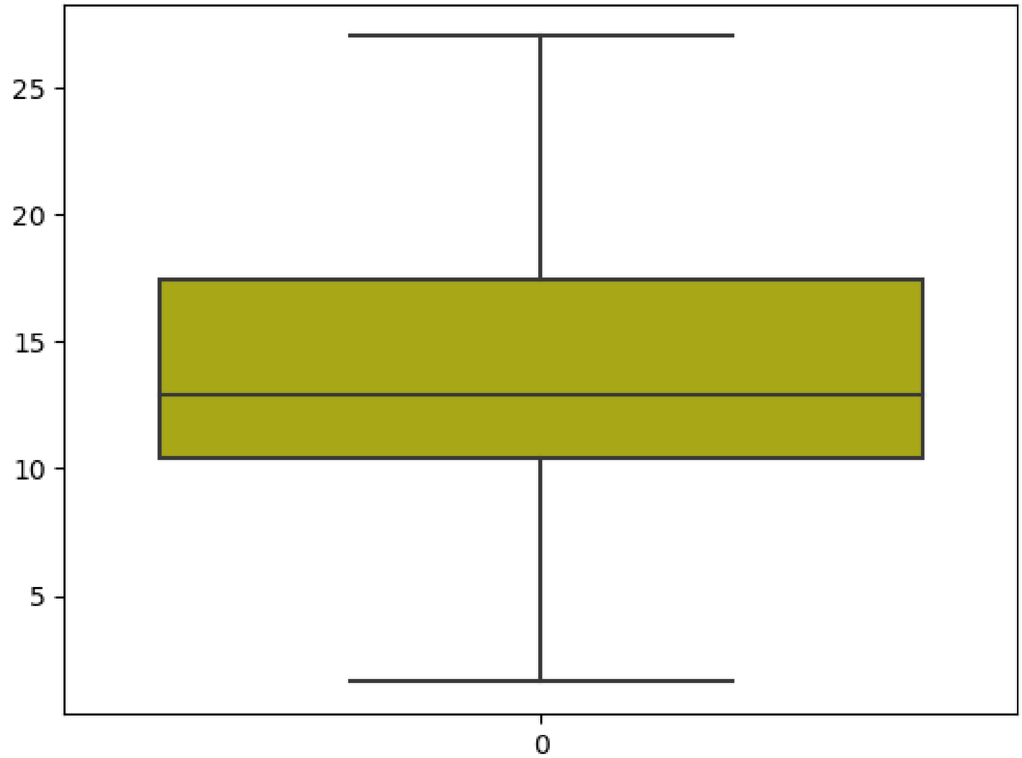
[10]:
```
sns.boxplot(df["Marketing Budget"],color="y")
```

[10]:
```
<AxesSubplot: >
```




[11]:
```
sns.boxplot(df["Sales"],color="y")
```

[11]: <AxesSubplot: >



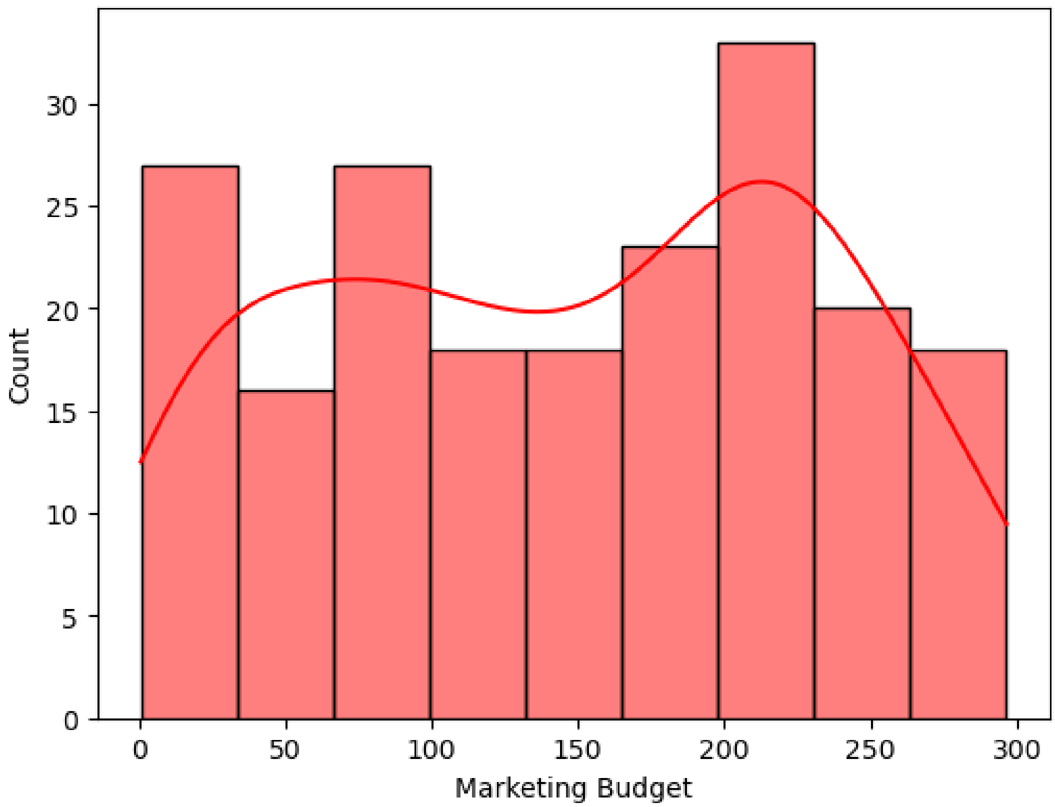

**Observations:**

- Given Data has no outliers

### 1.2.3 Check the Data Distribution Type

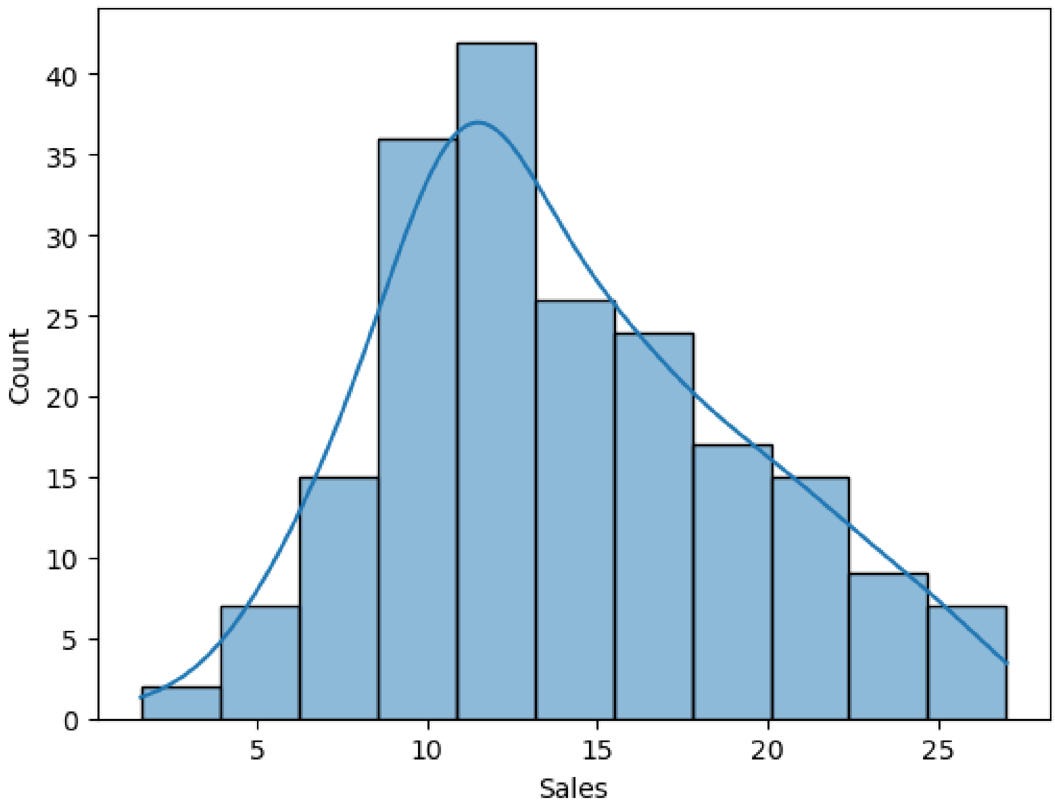
[12]:
```
sns.histplot(df["Marketing Budget"],color="r",kde=True)
```

[12]:
```
<AxesSubplot: xlabel='Marketing Budget', ylabel='Count'>
```




```
[13]: sns.histplot(df["Sales"],kde=True)
```

```
[13]: <AxesSubplot: xlabel='Sales', ylabel='Count'>
```




**Observations:**

- Marketing Budget count is nearly equal for all the Budgets, except 3 points
- Sales Data is almost Normally Distributed

### 1.2.4 Let's Standardize the Data

[14]:
```
df
```

[14]:
```
     Marketing Budget  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]
```

[15]:
```python
from sklearn.preprocessing import MinMaxScaler
```

[16]:
```python
scaler=MinMaxScaler()
```

[17]:
```python
df["MB_Scaled"]=scaler.fit_transform(df[["Marketing Budget"]])
```

[18]:
```python
df
```

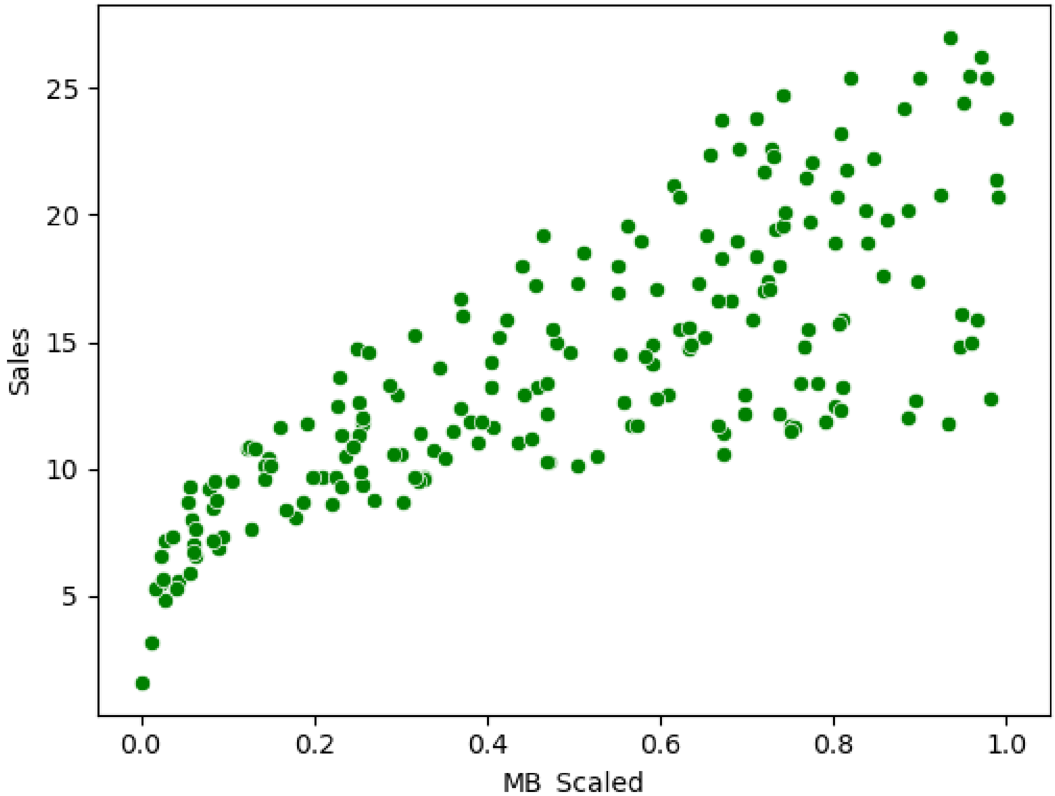
[18]:
```
     Marketing Budget  Sales  MB_Scaled
0               230.1   22.1   0.775786
1                44.5   10.4   0.148123
2                17.2    9.3   0.055800
3               151.5   18.5   0.509976
4               180.8   12.9   0.609063
..                ...    ...        ...
195              38.2    7.6   0.126818
196              94.2    9.7   0.316199
197             177.0   12.8   0.596212
198             283.6   25.5   0.956713
199             232.1   13.4   0.782550

[200 rows x 3 columns]
```

### 1.2.5 Let's see scatter plot using new scaled Data.

[19]:
```python
sns.scatterplot(x=df["MB_Scaled"],y=df["Sales"],c="g")
```

[19]:
```
<AxesSubplot: xlabel='MB_Scaled', ylabel='Sales'>
```




### 1.3 Build the Model

### 1.4 First split the Data

[20]:
```
x=df[["MB_Scaled"]]
```

[21]:
```
y=df["Sales"]
```

[22]:
```
x
```

[22]:
```
     MB_Scaled
0     0.775786
1     0.148123
2     0.055800
3     0.509976
4     0.609063
..         …
195   0.126818
196   0.316199
197   0.596212
198   0.956713
```

```
199   0.782550

[200 rows x 1 columns]
```

[23]:
```
y
```

[23]:
```
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
```

[24]:
```
from sklearn.model_selection import train_test_split
```

[25]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

[26]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

[26]:
```
((160, 1), (40, 1), (160,), (40,))
```

### 1.4.1 Import the model

[27]:
```
from sklearn.linear_model import LinearRegression
```

[28]:
```
model=LinearRegression()
```

### 1.4.2 Fit the Data into the model.

[29]:
```
model.fit(x_train,y_train)
```

[29]:
```
LinearRegression()
```

### 1.4.3 Now Predict the x_test values

[30]:
```
y_pred=model.predict(x_test)
```

[31]:
```
y_pred
```

[31]: array([16.92516649,  7.67317615, 13.33512689, 13.22532515, 17.44075728,
        17.66513475, 16.01333462, 15.43568197, 13.04868756, 16.55279536,
        19.59860023, 13.69317605, 15.17311259,  9.10059881, 17.56488099,
        20.48178816, 18.2427874 ,  7.67795014, 11.53055912, 14.19921887,
        13.74568993, 10.24158213, 12.78611818, 13.6311142 , 18.38123308,
        20.44837024, 13.5022165 ,  7.3962848 , 17.39779138, 17.74151857,
        18.9254678 , 13.79342982, 18.03273189, 12.33258924, 13.4592506 ,
        20.87802923, 19.24055107,  9.10537279, 15.28291433, 16.71511098])

#### 1.4.4 Checking the cost function metrics

[32]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

[33]:
```
mse=mean_squared_error(y_test,y_pred)
mae=mean_absolute_error(y_test,y_pred)
rmse=np.sqrt(mse)
```

[34]:
```
mse,mae,rmse
```

[34]: (12.393445953261908, 2.850318401000572, 3.520432637228258)

#### 1.4.5 Checking the accuracy of model

[35]:
```
from sklearn.metrics import r2_score
```

[36]:
```
r2=r2_score(y_test,y_pred)
r2
```

[36]: 0.5077331355308953

[37]:
```
# Adjusted r2
ar2=1-((1-r2)*(len(y)-1))/((len(y-1)-len(df.columns)-1))
ar2
```

[37]: 0.5001984386257559

### 1.5 Ckeck the slope and intercept's

[38]:
```
slope=model.coef_
slope
```

[38]: array([14.11668495])

[39]:
```
intercept=model.intercept_
intercept
```

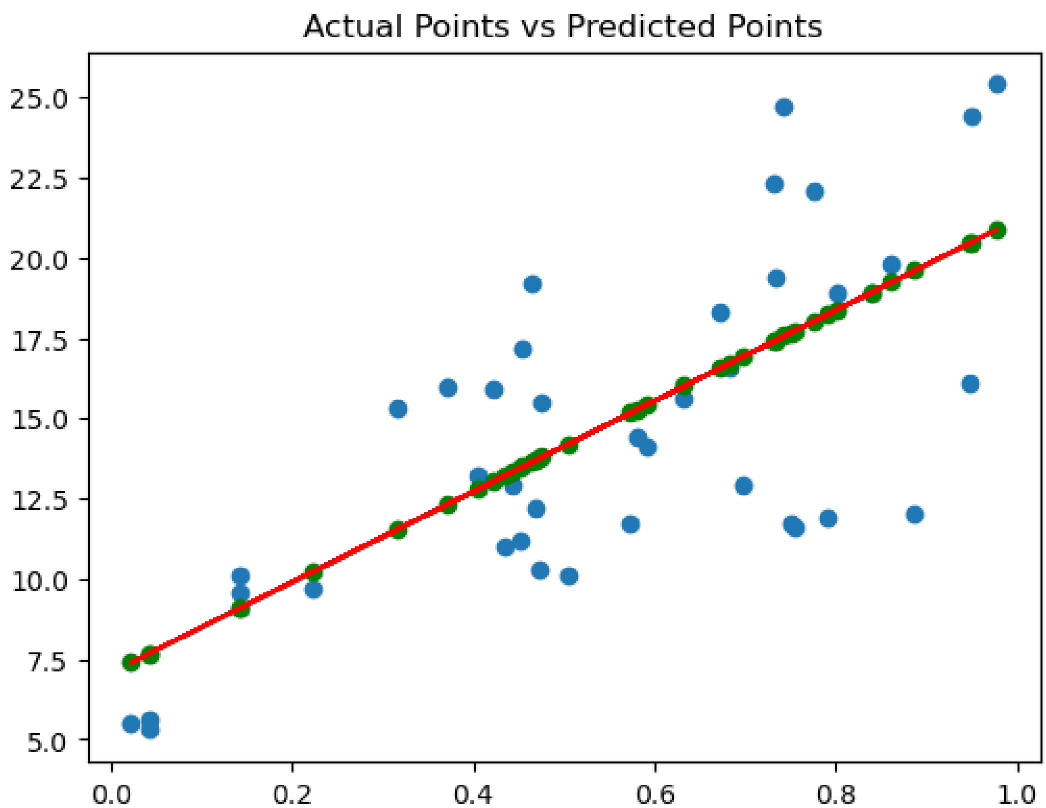
[39]: 7.081201533893286

## 1.6 Results Visualization

### 1.6.1 Actual points vs best fit line points

[40]:
```
plt.scatter(x_test,y_test)
plt.plot(x_test,y_pred,c="r")
plt.scatter(x_test,y_pred,c="g")
plt.title("Actual Points vs Predicted Points")
```

[40]: Text(0.5, 1.0, 'Actual Points vs Predicted Points')




### 1.6.2 SS Residual vs SS Total

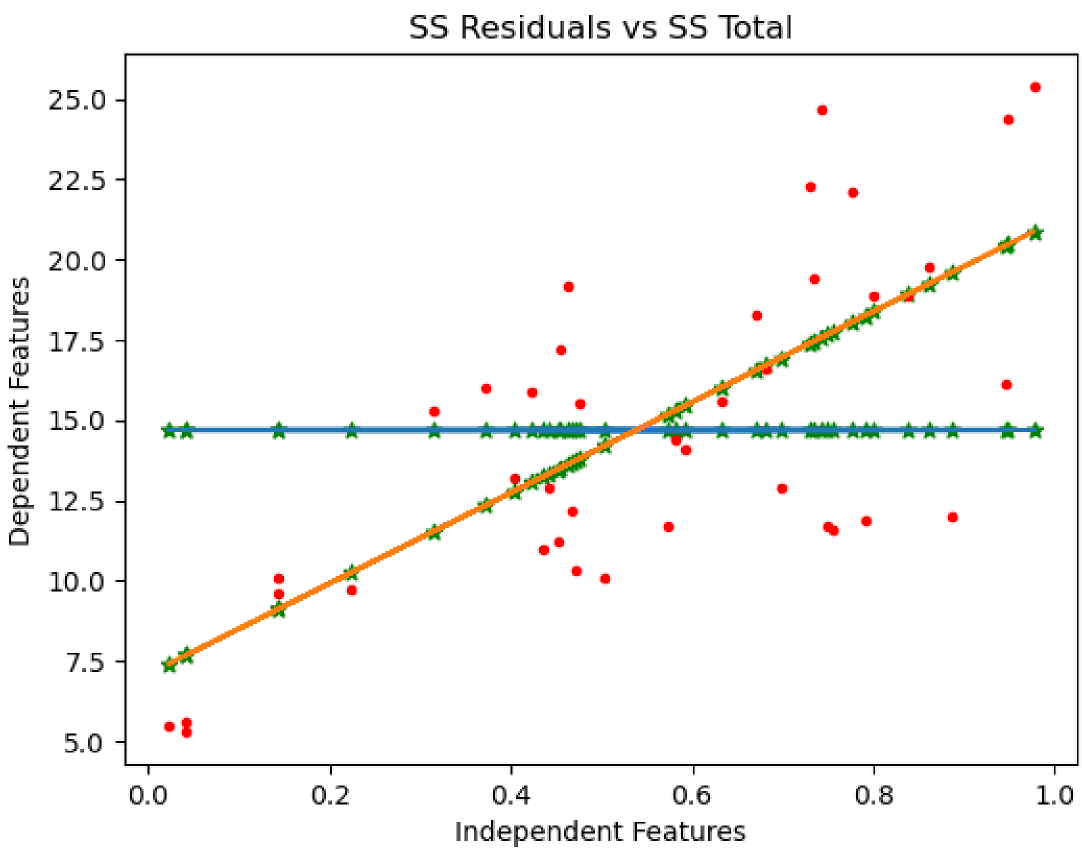
[67]:
```
y_test=pd.DataFrame(y_test,columns=["Sales"])
```

[69]:
```
y_test["Mean"]=y_test["Sales"].mean()
```

[100]:
```
plt.plot(x_test["MB_Scaled"],y_test["Mean"])
plt.plot(x_test,y_pred)
```

```
plt.scatter(x_test,y_pred,c="g",marker="*")
plt.scatter(x_test["MB_Scaled"],y_test["Mean"],c="g",marker="*")
plt.scatter(x_test["MB_Scaled"],y_test["Sales"],c="r",marker=".")
plt.xlabel("Independent Features")
plt.ylabel("Dependent Features")
plt.title("SS Residuals vs SS Total")
```

[100]: Text(0.5, 1.0, 'SS Residuals vs SS Total')




[ ]: