Future Sales Prediction with Machine Learning

November 7, 2023

1 Future Sales Prediction with Machine Learning

[1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

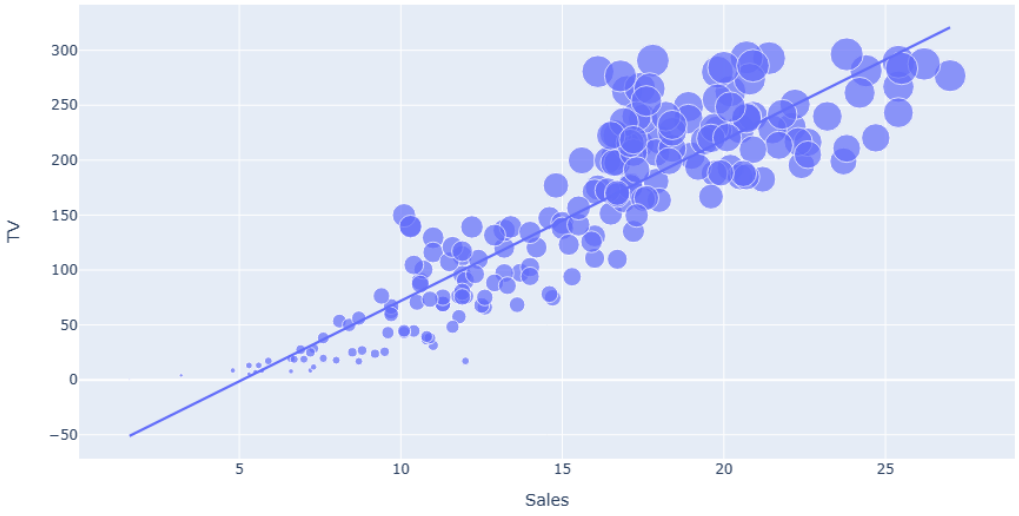
[2]:
```
data = pd.read_csv("D:/Masters_Degree/udemy/advertising.csv")
print(data.head())
```

```
      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   12.0
3  151.5   41.3       58.5   16.5
4  180.8   10.8       58.4   17.9
```

[3]:
```
print(data.isnull().sum())
```

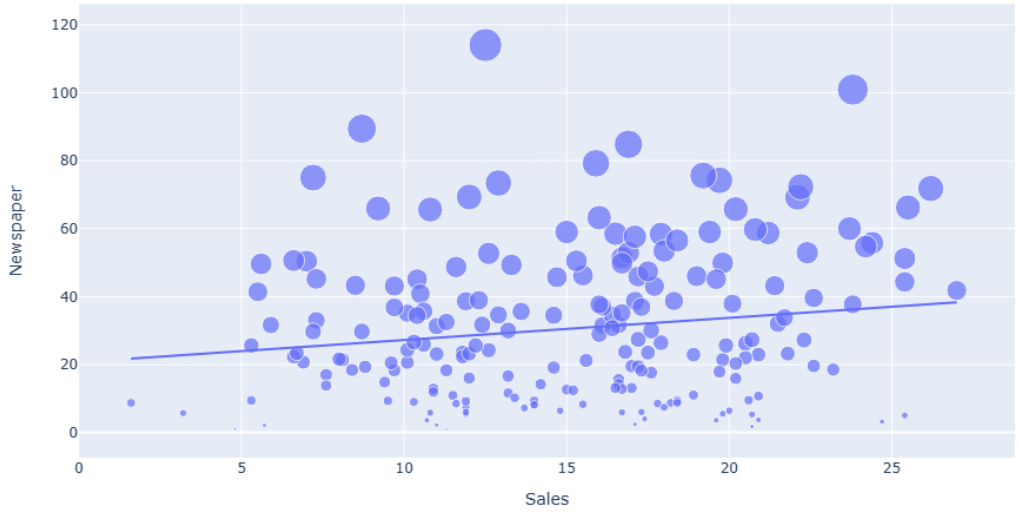
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

[4]:
```
import plotly.express as px
import plotly.graph_objects as go
figure = px.scatter(data_frame = data, x="Sales",
                    y="TV", size="TV", trendline="ols")
figure.show()
```

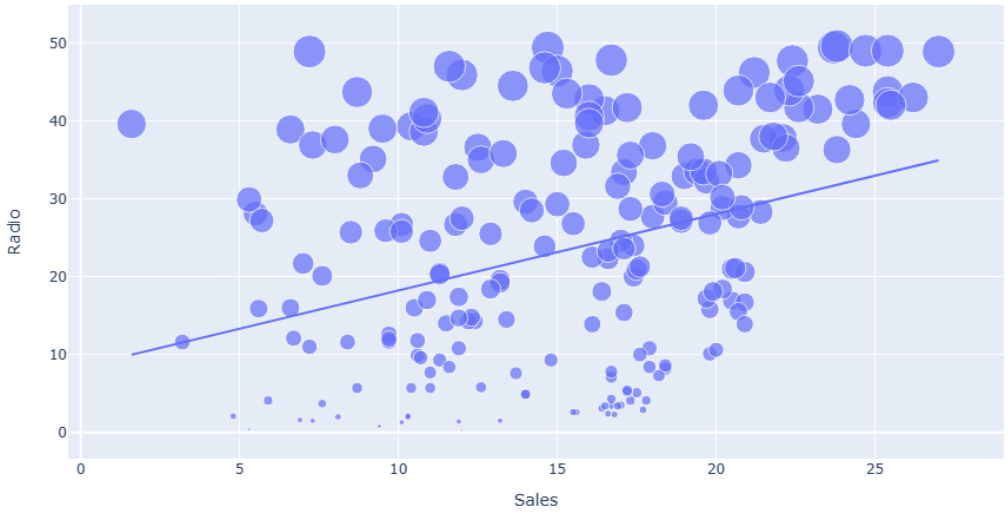


As from the plot we can see that when the investment of advertising on TV increases the sale of product also increased. It is linear relation or directly proportional.

[5]:
```
figure = px.scatter(data_frame = data, x="Sales",
                    y="Newspaper", size="Newspaper", trendline="ols")
figure.show()
```



[6]:
```
figure = px.scatter(data_frame = data, x="Sales",
                    y="Radio", size="Radio", trendline="ols")
figure.show()
```



[7]:
```
correlation = data.corr()
print(correlation["Sales"].sort_values(ascending=False))
```

```
Sales        1.000000
TV           0.901208
Radio        0.349631
Newspaper    0.157960
Name: Sales, dtype: float64
```

1.1 Future Sales Prediction Model

[8]:
```
x = np.array(data.drop(["Sales"], 1))
y = np.array(data["Sales"])
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.2,
                                                random_state=42)
```

```
C:\Users\umair\AppData\Local\Temp\ipykernel_14796\2488982787.py:1:
FutureWarning:

In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only.
```

[9]:
```
model = LinearRegression()
model.fit(xtrain, ytrain)
print(model.score(xtest, ytest))
```

```
0.9059011844150826
```

[11]:
```
#features = [[TV, Radio, Newspaper]]
#features = np.array([[230.1, 37.8, 69.2]])
predicted = model.predict(xtest)
```

[12]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

[13]:
```
mae = mean_absolute_error(ytest, predicted)
mse = mean_squared_error(ytest, predicted)
rmse = np.sqrt(mse)
r2 = r2_score(ytest, predicted)
```

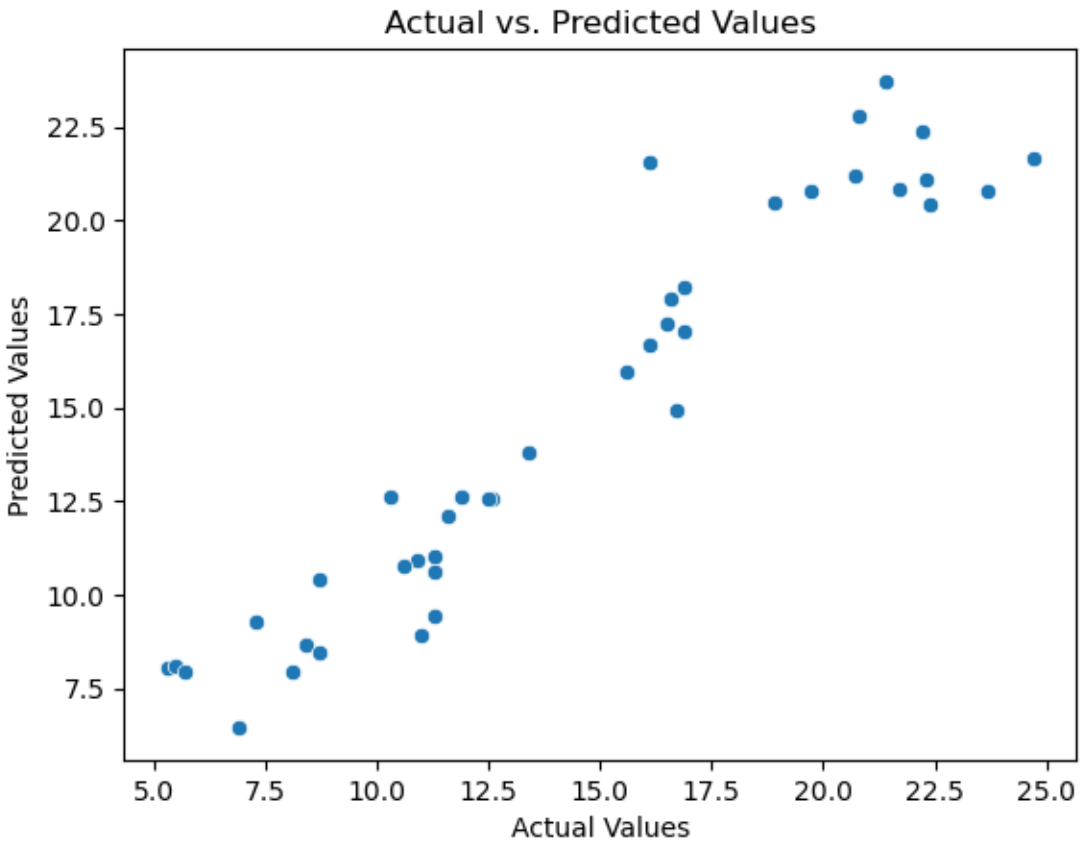
1.1.1 Performance of the Model

[14]:
```
print('Mean Absolute Error (MAE) :', mae)
print('Mean Squared Error (MSE) :', mse)
print('Root Mean Absolute Error (RMSE) :', rmse)
print('R-squared (R²) Error (R²) :', r2)
```

```
Mean Absolute Error (MAE) : 1.274826210954934
Mean Squared Error (MSE) : 2.907756910271091
Root Mean Absolute Error (RMSE) : 1.7052146229349228
R-squared (R²) Error (R²) : 0.9059011844150826
```

[15]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[16]:
```
sns.scatterplot(x=ytest, y=predicted)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted Values')
plt.show()
```


Actual vs. Predicted Values
Predicted Values
Actual Values
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0