Marketing And Sales Data Linear Regression

August 5, 2023

1 Marketing And Sales Data Linear Regression

1.1 Step 1: Imports

[1]:
```
# Import relevant libraries and packages.
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

1.1.1 Load the dataset

[2]:
```
data = pd.read_csv('marketing_and_sales_data.csv')

# Display the first five rows.
data.head()
```

[2]:

	TV	Radio	Social_Media	Sales
0	16.0	6.566231	2.907983	54.732757
1	13.0	9.237765	2.409567	46.677897
2	41.0	15.886446	2.913410	150.177829
3	83.0	30.020028	6.922304	298.246340
4	15.0	8.437408	1.405998	56.594181

1.2 Step 2: Data exploration

[3]:
```
# Display the shape of the data.

data.shape
```

[3]:
```
(4572, 4)
```

1.2.1 Explore the independent variables

[4]:
```python
# Generate descriptive statistics about independent variables.

data[["TV", "Radio", "Social_Media"]].describe()
```

[4]:

	TV	Radio	Social_Media
count	4562.000000	4568.000000	4566.000000
mean	54.066857	18.160356	3.323956
std	26.125054	9.676958	2.212670
min	10.000000	0.000684	0.000031
25%	32.000000	10.525957	1.527849
50%	53.000000	17.859513	3.055565
75%	77.000000	25.649730	4.807558
max	100.000000	48.871161	13.981662

1.2.2 Explore the dependent variable

[5]:
```python
# Calculate the total number of missing values in the sales column.
data["Sales"].isna().sum()
```

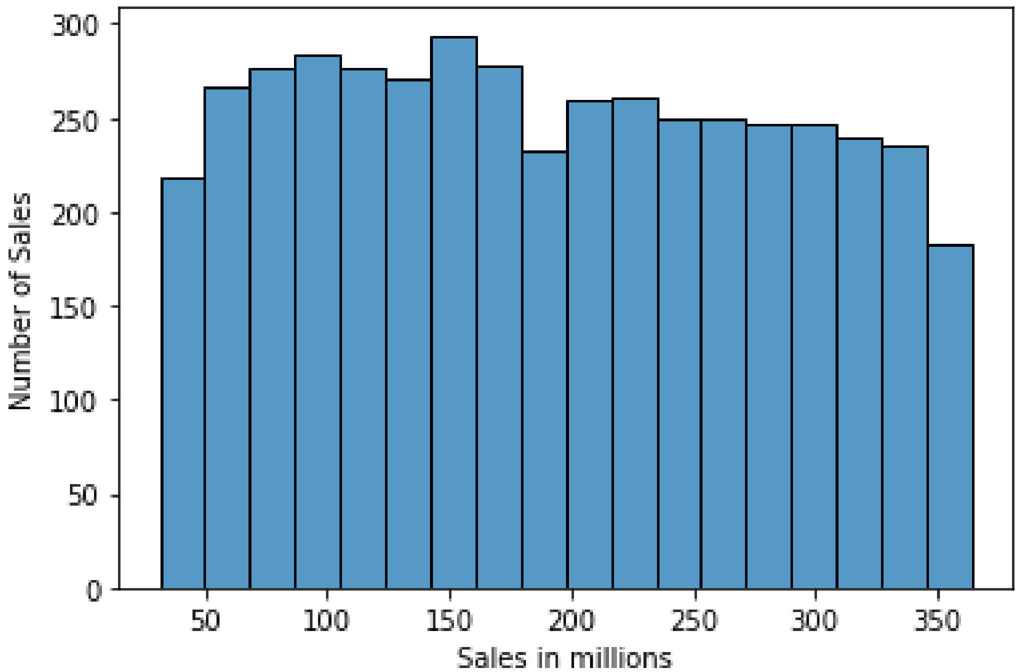
[5]: 6

1.2.3 Remove the missing data

[6]:
```python
# Subset the data to include rows where Sales is present.
data.dropna(subset=["Sales"], axis=0, inplace=True)
```

1.2.4 Visualize the sales distribution

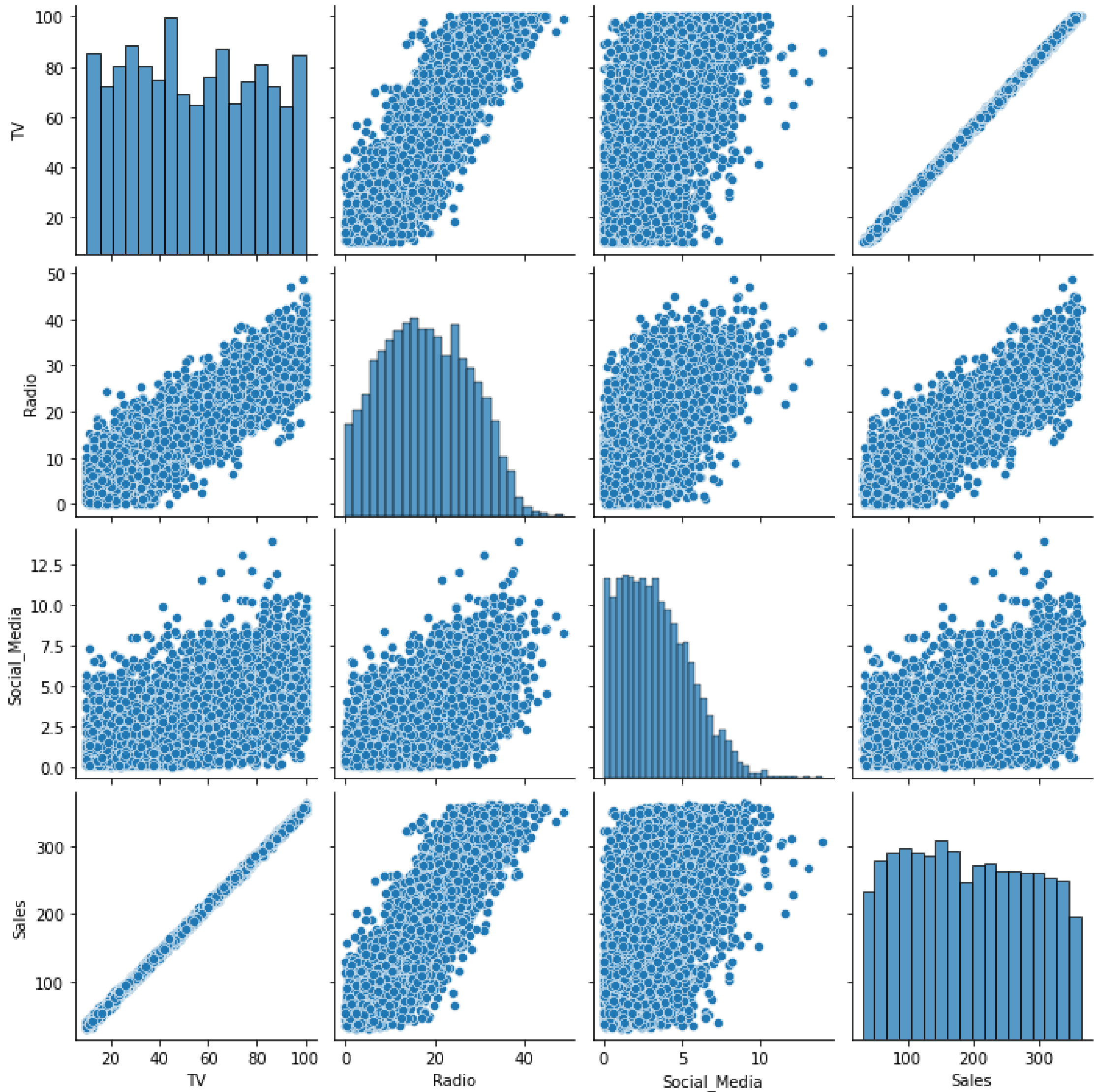
[7]:
```python
# Create a histogram of the Sales.
fig=sns.histplot(data["Sales"])

# Add a title
fig.set_xlabel("Sales in millions")
fig.set_ylabel("Number of Sales")
plt.show()
```



1.3 Step 3: Model building

[8]:
```
# Create a pairplot of the data.
sns.pairplot(data)
```

[8]: <seaborn.axisgrid.PairGrid at 0x7f1bba294c50>



1.3.1 Build and fit the model

[9]:
```
# Define the OLS formula.
ols_formula="Sales ~ TV"

# Create an OLS model.
OLS=ols(data=data, formula=ols_formula)

# Fit the model.
model=OLS.fit()

# Save the results summary.
```

```python
model_result=model.summary()

# Display the model results.
model_result
```

[9]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 4.527e+06
Date:                Sat, 05 Aug 2023   Prob (F-statistic):               0.00
Time:                        14:58:58   Log-Likelihood:                -11393.
No. Observations:                4556   AIC:                         2.279e+04
Df Residuals:                    4554   BIC:                         2.280e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.1263      0.101     -1.257      0.209      -0.323       0.071
TV             3.5614      0.002   2127.776      0.000       3.558       3.565
==============================================================================
Omnibus:                        0.051   Durbin-Watson:                   2.002
Prob(Omnibus):                  0.975   Jarque-Bera (JB):                0.030
Skew:                           0.001   Prob(JB):                        0.985
Kurtosis:                       3.012   Cond. No.                         138.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

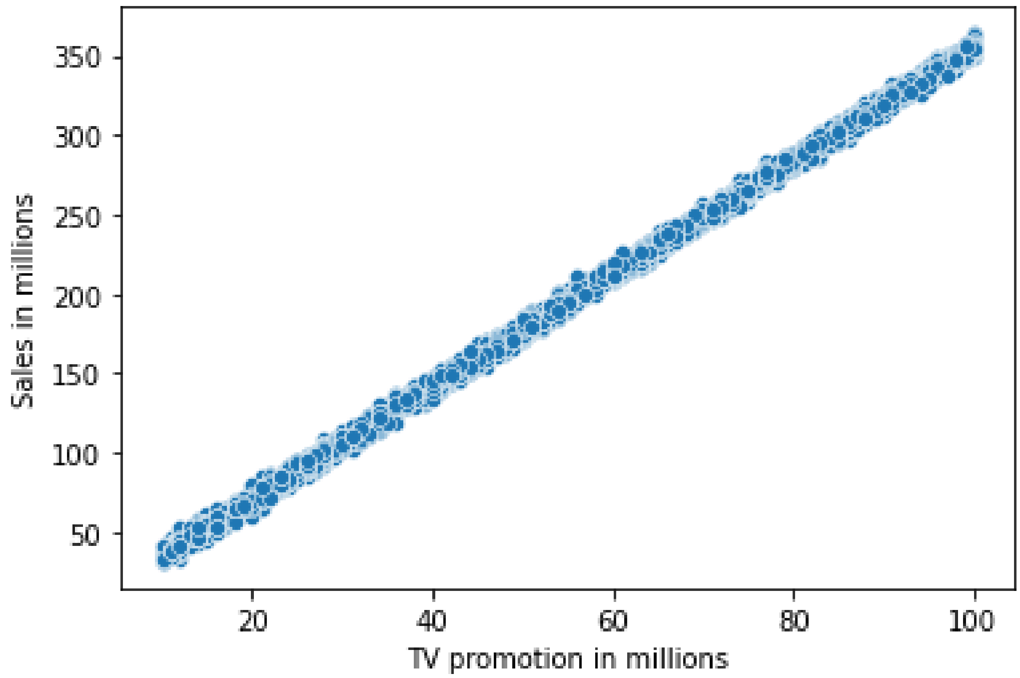
1.3.2 Check model assumptions

To justify using simple linear regression, we should check that the four linear regression assumptions are not violated.

These assumptions are: - Linearity requires a linear relationship between the independent and dependent variables. - Independent Observations states that each observation in the dataset is independent. - Normality states that the errors are normally distributed. - Homoscedasticity is that the residuals have a constant variance for all values of independent variables.

1.3.3 Model assumption: Linearity

[10]:
```
# Create a scatterplot comparing X and Sales (Y).
fig=sns.scatterplot(x=data["TV"], y=data["Sales"])
fig.set_xlabel("TV promotion in millions")
fig.set_ylabel("Sales in millions")
plt.show()
```



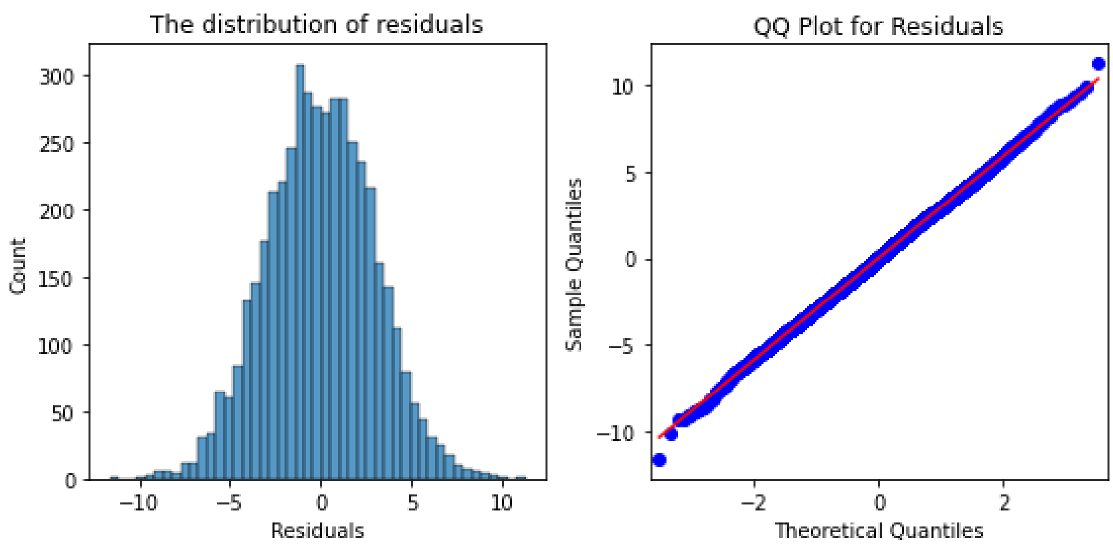
1.3.4 Model assumption: Independence

As each marketing promotion (i.e., row) is independent from one another, the independence assumption is not violated.

1.3.5 Model assumption: Normality

[11]:
```
# Calculate the residuals.
residuals= model.resid

# Create a 1x2 plot figures.
fig, axes = plt.subplots(1,2, figsize=(8,4))

# Create a histogram with the residuals.
sns.histplot(residuals, ax=axes[0])
```

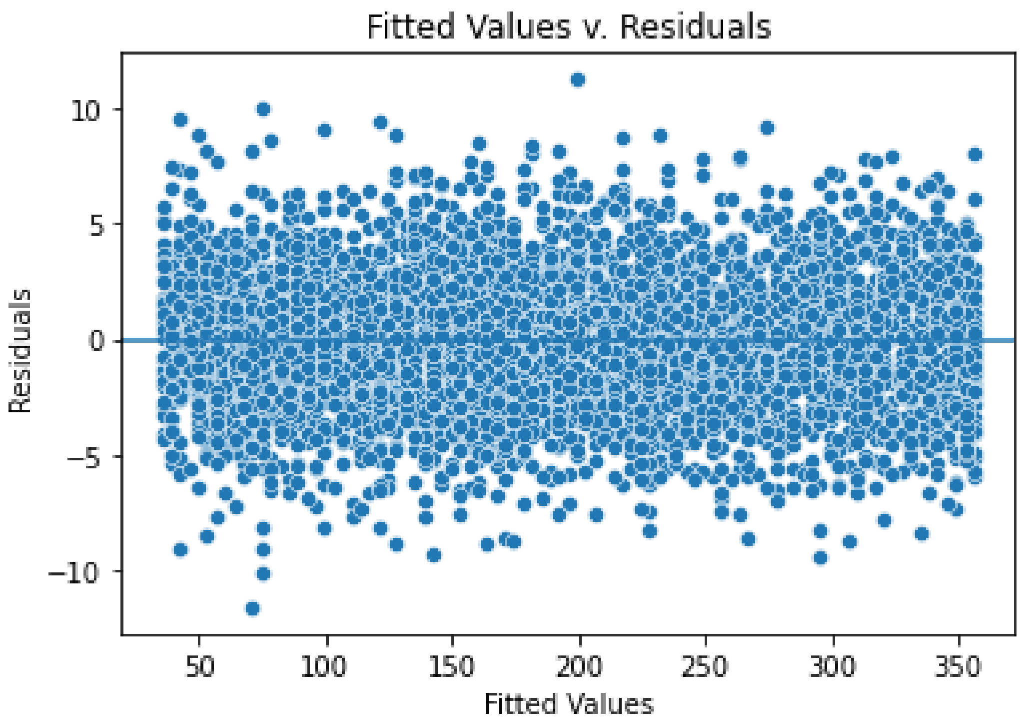
```
# Set the x label of the residual plot.
axes[0].set_xlabel("Residuals")

# Set the title of the residual plot.
axes[0].set_title("The distribution of residuals")

# Create a Q-Q plot of the residuals.
sm.qqplot(model.resid, line="s", ax=axes[1])

# Set the title of the Q-Q plot.
axes[1].set_title("QQ Plot for Residuals")

# Use tight_layout() function to add space between plots.
plt.tight_layout()

# Show the plot.
plt.show()
```



1.3.6 Model assumption: Homoscedasticity

[12]:
```
# Create a scatterplot with the fitted values from the model and the residuals.
fitted_values=model.predict(data["TV"])
fig=sns.scatterplot(x=fitted_values, y=residuals)

# Set the x-axis label.
fig.set_xlabel("Fitted Values")
```

```
# Set the y-axis label.
fig.set_ylabel("Residuals")

# Set the title.
fig.set_title("Fitted Values v. Residuals")

# Add a line at y = 0 to visualize the variance of residuals above and below 0.
fig.axhline(0)

# Show the plot.
plt.show()
```



1.4 Step 4: Results and evaluation

1.4.1 Display the OLS regression results

[13]:
```
# Display the model_results defined previously.
model_result
```

[13]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 4.527e+06
Date:                Sat, 05 Aug 2023   Prob (F-statistic):               0.00
Time:                        14:58:58   Log-Likelihood:                -11393.
No. Observations:                4556   AIC:                         2.279e+04
Df Residuals:                    4554   BIC:                         2.280e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.1263      0.101     -1.257      0.209      -0.323       0.071
TV             3.5614      0.002   2127.776      0.000       3.558       3.565
==============================================================================
Omnibus:                        0.051   Durbin-Watson:                   2.002
Prob(Omnibus):                  0.975   Jarque-Bera (JB):                0.030
Skew:                           0.001   Prob(JB):                        0.985
Kurtosis:                       3.012   Cond. No.                         138.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

1.5 Conclusions

- Sales are evenly distributed between 25 and 350 million.
- TV has the strongest positive linear relationship with sales compared to radio and social media.
- The model has a high R-squared value of 0.999, indicating that 99.9% of the variation in sales can be explained by the TV promotional budget.
- The p-value for the TV coefficient is 0.0000, and the 95% confidence interval is [3.558, 3.565], indicating high confidence in the impact of TV on sales.
- The coefficients for Intercept and TV are -0.1263 and 3.5614, respectively, meaning an increase of one million dollars in the TV promotional budget leads to an estimated increase of 3.5614 million dollars in sales.
- Increasing the TV promotional budget should be prioritized to boost sales.