Sales Prediction using Python

1) IMPORTING THE LIBRARIES

In [1]:
```
# Import necessary libraries

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
import statsmodels.formula.api as sm
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
import warnings
```

In [2]:
```
warnings.simplefilter(action='ignore', category=FutureWarning)
os.getcwd()
```

Out[2]:
```
'C:\\Users\\Aniket\\Aniket JupyterFiles\\Oasisi Infobyte - Data Science Intern'
```

2) LOADING THE DATASET:

In [4]:
```
# Load dataset
df = pd.read_csv("Advertising.csv")
```

3) EXPLORATORY DATA ANALYSIS:

In [5]:
```
# View the first few rows of the dataset
df.head()
```

Out[5]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [6]:
```
# Get the column names of the dataset
df.columns
```

Out[6]:
```
Index(['Unnamed: 0', 'TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [7]:
```
# To rename the column 'Unnamed: 0' to 'Index'
df.rename(columns={'Unnamed: 0': 'Index'}, inplace=True)
```

In [11]:
```
df
```

Out[11]:

	Index	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
...	...	...	...	...	...
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

200 rows × 5 columns

In [8]:
```
# Get the shape of the dataset (rows, columns)
df.shape
```

Out[8]:
```
(200, 5)
```

In [9]:
```
# Check information about the dataset, data types, and missing values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Index      200 non-null    int64  
 1   TV         200 non-null    float64
 2   Radio      200 non-null    float64
 3   Newspaper  200 non-null    float64
 4   Sales      200 non-null    float64
dtypes: float64(4), int64(1)
memory usage: 7.9 KB
```

In [10]:
```
# Get statistical summary of the numerical columns
df.describe().T
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
Index	200.0	100.5000	57.879185	1.0	50.750	100.50	150.250	200.0
TV	200.0	147.0425	85.854236	0.7	74.375	149.75	218.825	296.4
Radio	200.0	23.2640	14.846809	0.0	9.975	22.90	36.525	49.6
Newspaper	200.0	30.5540	21.778621	0.3	12.750	25.75	45.100	114.0
Sales	200.0	14.0225	5.217457	1.6	10.375	12.90	17.400	27.0

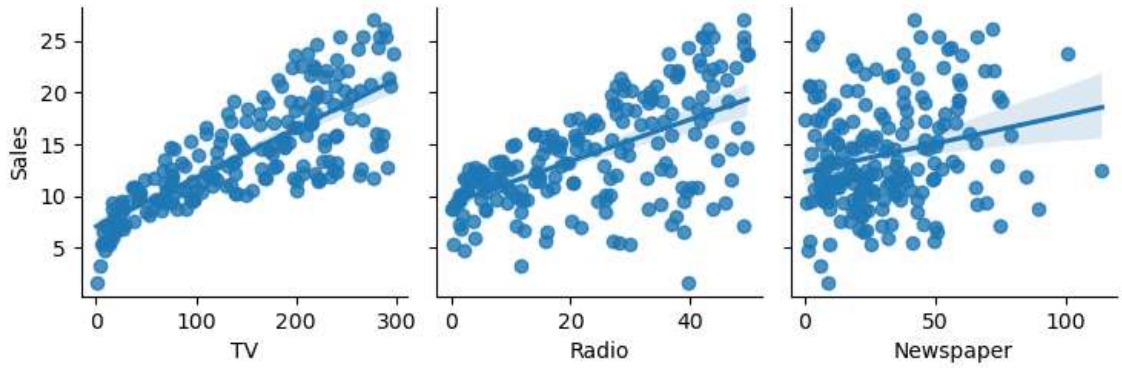
In [11]:
```
# Check for missing values in the dataset
df.isnull().values.any()
df.isnull().sum()
```

Out[11]:
```
Index        0
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

4) Data Visualization:

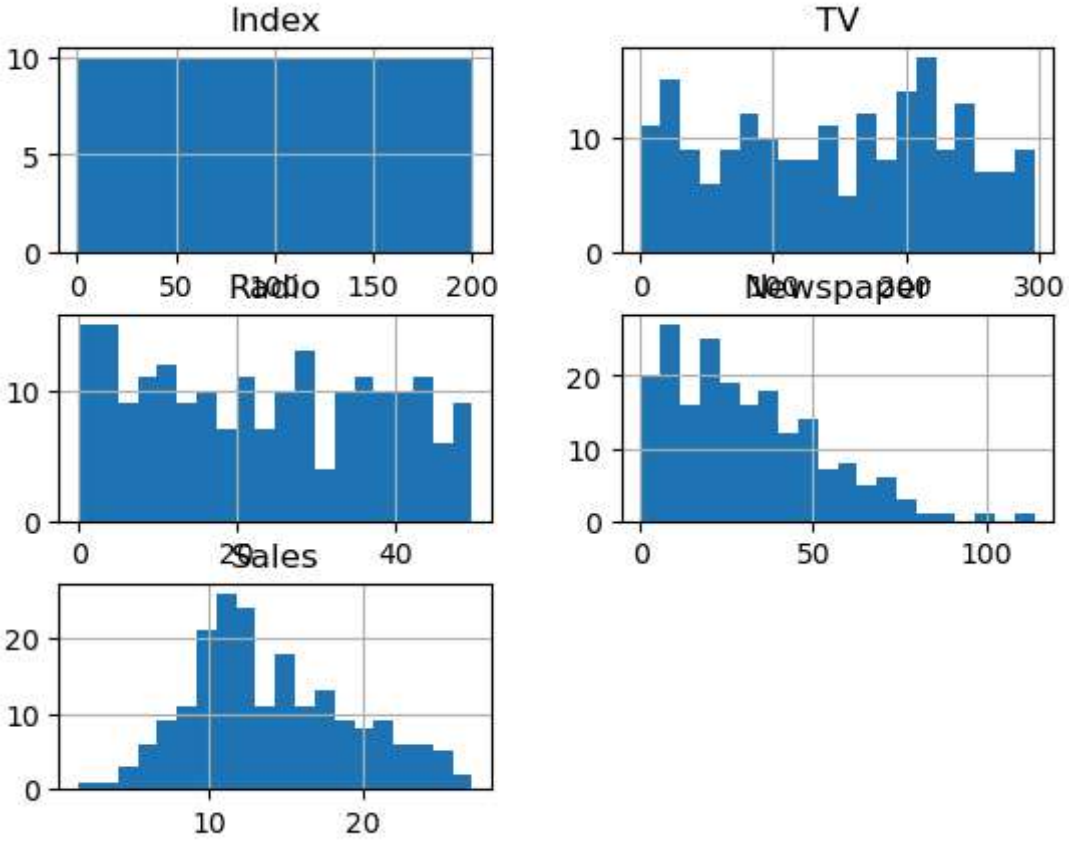
In [12]:
```
# Scatter plots to check the linearity assumption between each independent
sns.pairplot(df, x_vars=["TV", "Radio", "Newspaper"], y_vars="Sales", kind=
```

Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x21bee9031d0>
```



In [13]:
```
# Histograms to check the normality assumption of the dependent variable (S
df.hist(bins=20)
```

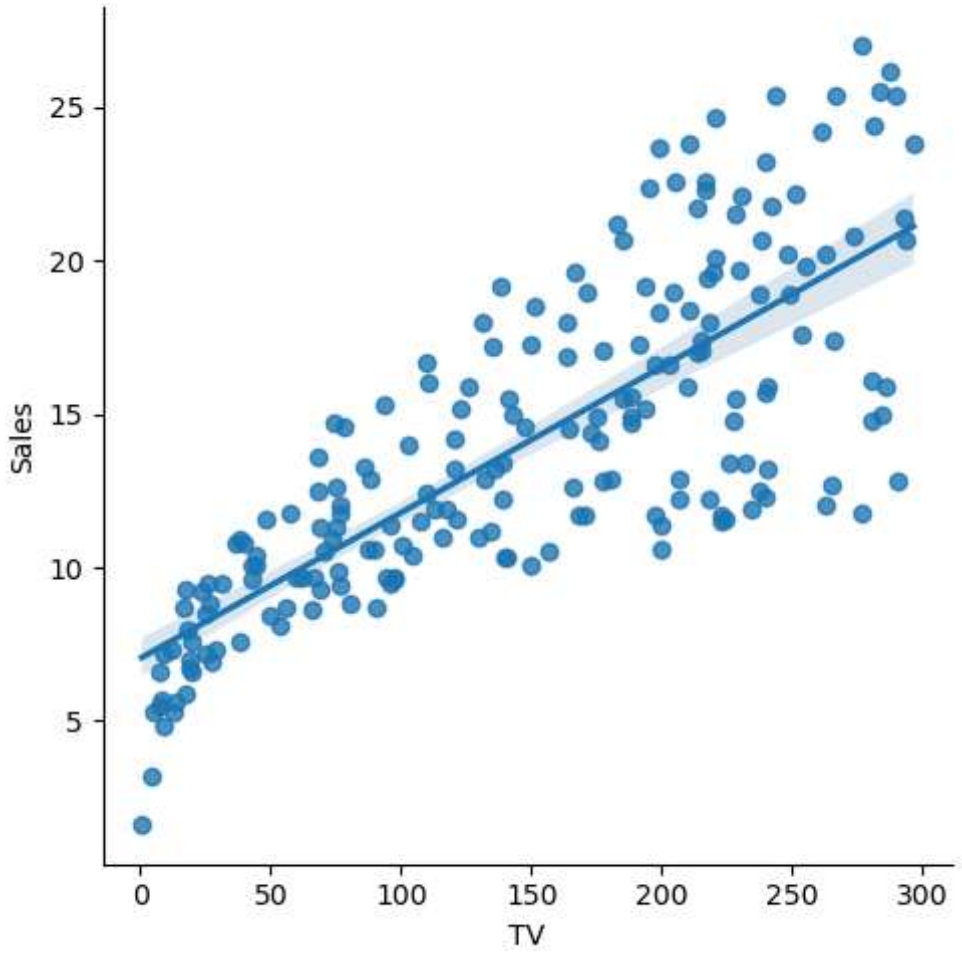
Out[13]:
```
array([[<Axes: title={'center': 'Index'}>,
        <Axes: title={'center': 'TV'}>],
       [<Axes: title={'center': 'Radio'}>,
        <Axes: title={'center': 'Newspaper'}>],
       [<Axes: title={'center': 'Sales'}>, <Axes: >]], dtype=object)
```

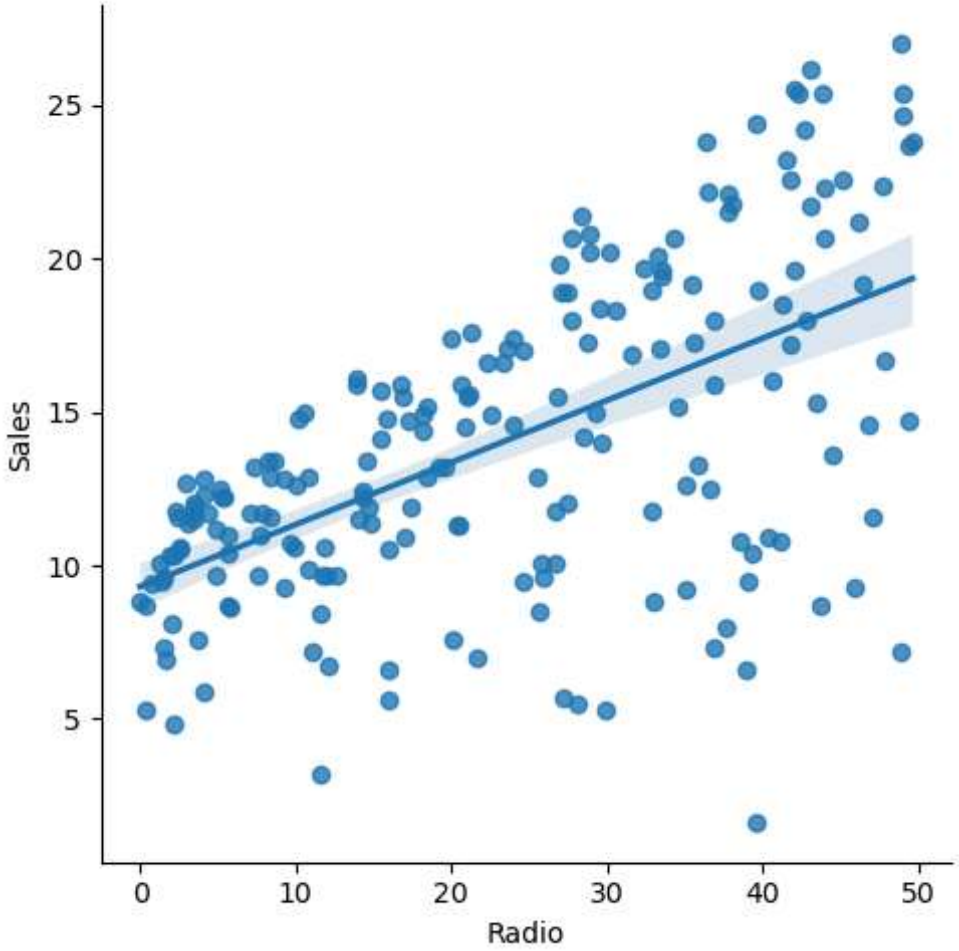


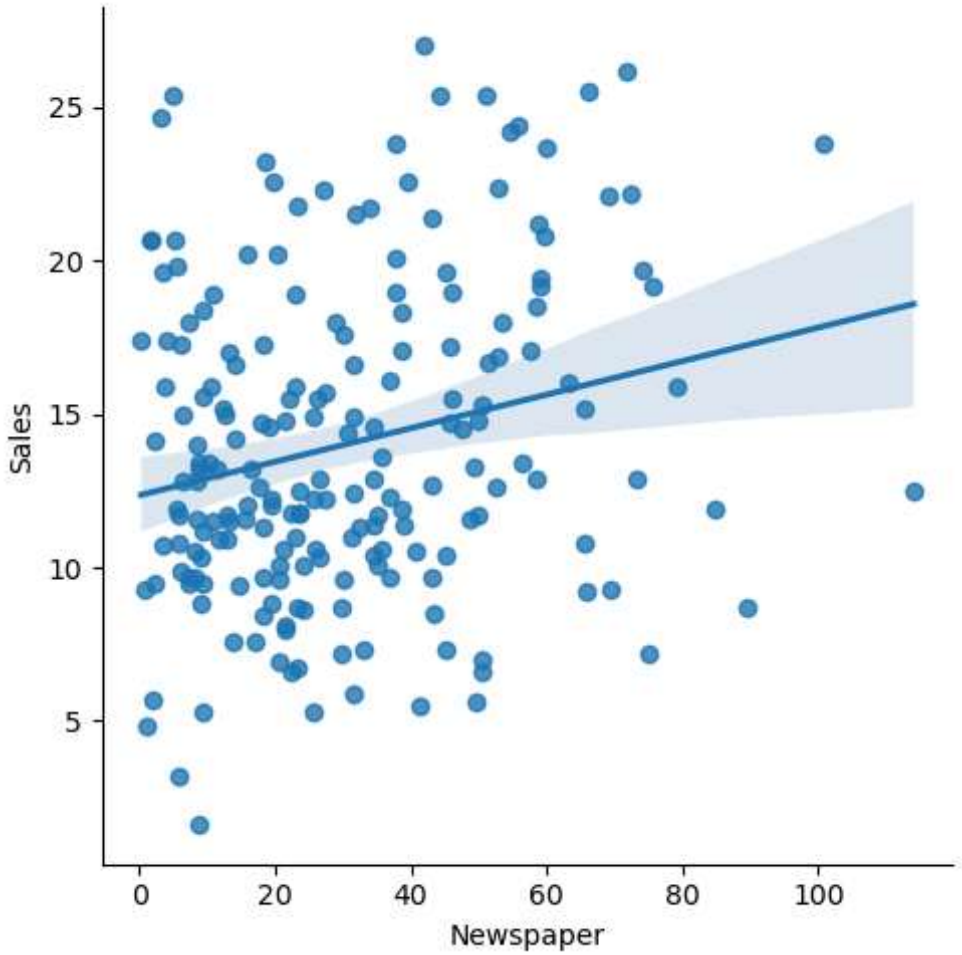
In [14]:
```
# Linear regression plots to visualize the relationship between each indepe

sns.lmplot(x='TV', y='Sales', data=df)
sns.lmplot(x='Radio', y='Sales', data=df)
sns.lmplot(x='Newspaper',y= 'Sales', data=df)
```

Out[14]: <seaborn.axisgrid.FacetGrid at 0x21bf20f3050>




Sales
25
20
15
10
5
0
10
20
30
40
50
Radio

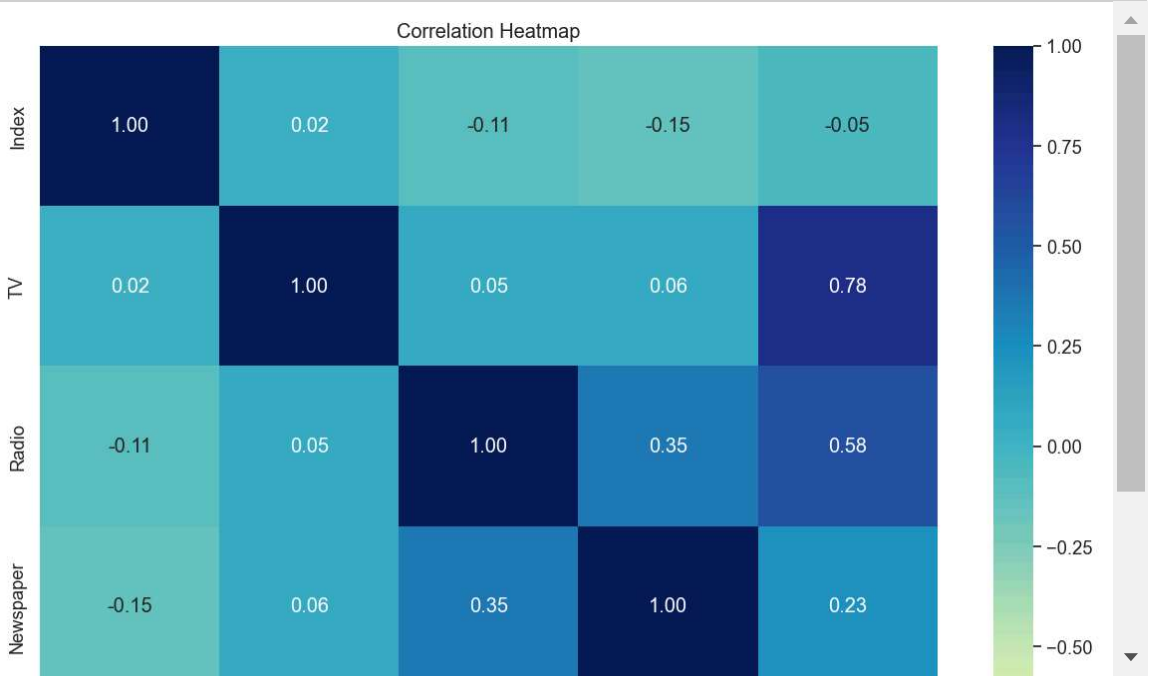

25
20
15
10
5
Sales
0
20
40
60
80
100
Newspaper

In [15]:
```
corrmat = df.corr()
# Set up the figure and axis for the heatmap
plt.figure(figsize=(12, 9))
sns.set(font_scale=1.2)  # Adjust font size for better readability

# Create a heatmap using Seaborn
sns.heatmap(corrmat, annot=True, cmap="YlGnBu", fmt=".2f", vmin=-1, vmax=1,

# Add title and adjust layout
plt.title("Correlation Heatmap")
plt.tight_layout()

# Display the heatmap
plt.show()
```



5) MODEL PREPARATION:

In [16]:
```
# Model Preparation
X = df.drop('Sales', axis=1)
y = df[["Sales"]]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, r
```

6) LINEAR REGRESSION MODEL:

In [17]:
```
lin_model = sm.ols(formula="Sales ~ TV + Radio + Newspaper", data=df).fit()
```

In [18]:
```
# Print the coefficients and summary of the linear model
print("Coefficients:\n", lin_model.params, "\n")
print("Summary:\n", lin_model.summary(), "\n")
```

```
Coefficients:
 Intercept    2.938889
TV           0.045765
Radio        0.188530
Newspaper   -0.001037
dtype: float64

Summary:
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                  Sales   R-squared:
0.897
Model:                            OLS   Adj. R-squared:
0.896
Method:                 Least Squares   F-statistic:
570.3
Date:                Thu, 22 Aug 2024   Prob (F-statistic):
1.58e-96
```

7) EVALUATE THE MODEL:

In [19]:
```
results = []
names = []
models = [('LinearRegression', LinearRegression())]

for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    result = np.sqrt(mean_squared_error(y_test, y_pred))
    results.append(result)
    names.append(name)
    msg = f"{name}: {result:.4f}"
    print(msg)
```

```
LinearRegression: 1.7036
```

8) MAKE REDICTIONS ON NEW DATA:

In [20]:
```
# Make predictions on new data
new_data = pd.DataFrame({'TV': [100], 'Radio': [50], 'Newspaper': [25]})
predicted_sales = lin_model.predict(new_data)
print("Predicted Sales for new data:", predicted_sales)
```

```
Predicted Sales for new data: 0    16.915917
dtype: float64
```

In [28]:

```
new_data = pd.DataFrame({'TV': [25], 'Radio': [63], 'Newspaper': [80]})
predicted_sales = lin_model.predict(new_data)
print("Predicted Sales for new data:", predicted_sales)
```

```
Predicted Sales for new data: 0    15.877397
dtype: float64
```

Project Outcome

- Machine learning empowers businesses to make informed decisions about marketing strategies and resource allocation.
- The linear regression model reveals the impact of advertising expenses on sales across different platforms.
- Model evaluation metrics like RMSE quantify predictive accuracy and help optimize performance.
- Predicting sales on new data scenarios enables revenue forecasting based on various advertising strategies.

Benefits

- Sales prediction enhances marketing campaigns, budget allocation, and revenue forecasting.
- Data-driven decisions improve resource utilization, reduce risks, and maximize profitability.
- Updating the model with new data ensures adaptability to market changes and trends.

Conclusion

This project demonstrates the application of machine learning techniques in sales prediction using Python. Analyzing historical data and building predictive models empower businesses to make data-driven decisions, optimize marketing strategies, allocate resources effectively, and achieve better performance in a competitive market. Sales prediction is a powerful tool for businesses seeking to maximize revenue potential and stay competitive in their industry.