Sales Data Analysis
Submitted for

BUSINESS FORECASTING METHODS AND APPLICATIONS

Submitted by:

(502304202) Manmeet Kour

(502304205) Raj Kansal

Submitted to-

Dr. Nitin Arvind Shelke

L M Thapar School of Management

Jan-May 2024

INDEX

LIST OF FIGURES

LIST OF TABLES

Note: List of Figures and List of Tables shall be on separate pages.

1. ABSTRACT

This report analyzes sales data for Amazon for three consecutive years to identify key trends and optimize sales performance.. By leveraging statistical analysis, we revealed that sales increased in next consecutive year and. These results suggest that amazon sales have the potential to boost revenue growth for the coming years .This report offers valuable insights for sales managers to improve coaching and tailor strategies for different customer segments.

2. INTRODUCTION AND RELATED WORK

The competitive landscape of today's business world demands a deep understanding of customer behaviour and sales performance. In this context, the core challenge lies in **unlocking actionable insights** from vast quantities of sales data. By effectively analyzing this data, we can gain valuable knowledge to optimize sales strategies, improve rep performance, and ultimately drive revenue growth.

Initial Goals

Our initial goals for this analysis are:

- To identify key trends and patterns within the sales data.
- To assess the effectiveness of current sales strategies.
- To uncover potential areas for improvement in the sales process.
- To formulate data-driven recommendations to enhance sales performance.

Building on existing research, we recognize the importance of key sales metrics, customer segmentation, and statistical analysis. These elements empower data-driven decision making to enhance sales performance.

This initial groundwork paves the way for a deep dive into our sales data analysis. We'll leverage these insights to formulate actionable recommendations for sales success.

3. SOFTWARE USED

We've chosen Python for data analysis because it's flexible and user-friendly. We also leverage some specialized helpers called libraries: Pandas, Plotly, Seaborn, and Scikit-learn. These libraries act like extra tools that supercharge Python's capabilities.

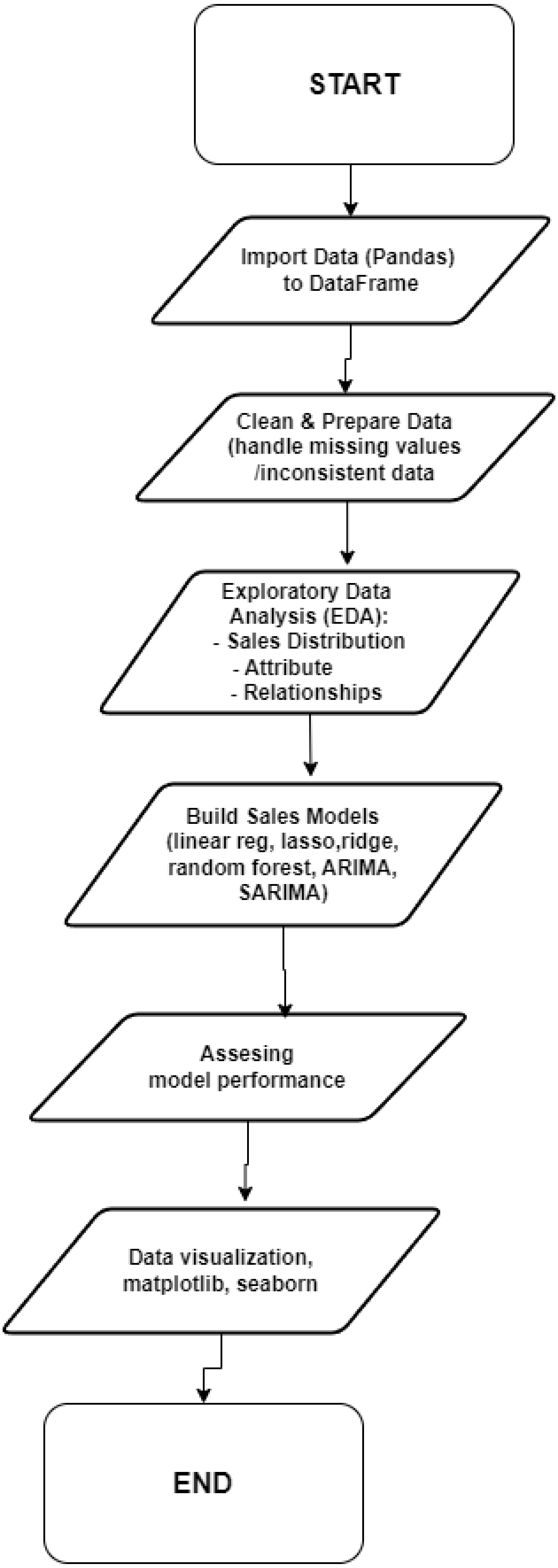
Pandas lets us effortlessly organize and manipulate our data, like meticulously arranging it into well-defined categories. Plotly helps us transform our data into vibrant charts and graphs, allowing us to readily identify trends. Seaborn is another valuable tool for creating visually appealing and professional-looking charts. Finally, Scikit-learn acts as our secret weapon for building intelligent models that forecast sales.

By utilizing these tools together, we ensure our analysis is robust and accurate. They assist us in processing massive amounts of data, uncovering crucial sales patterns, and making informed predictions about future sales.

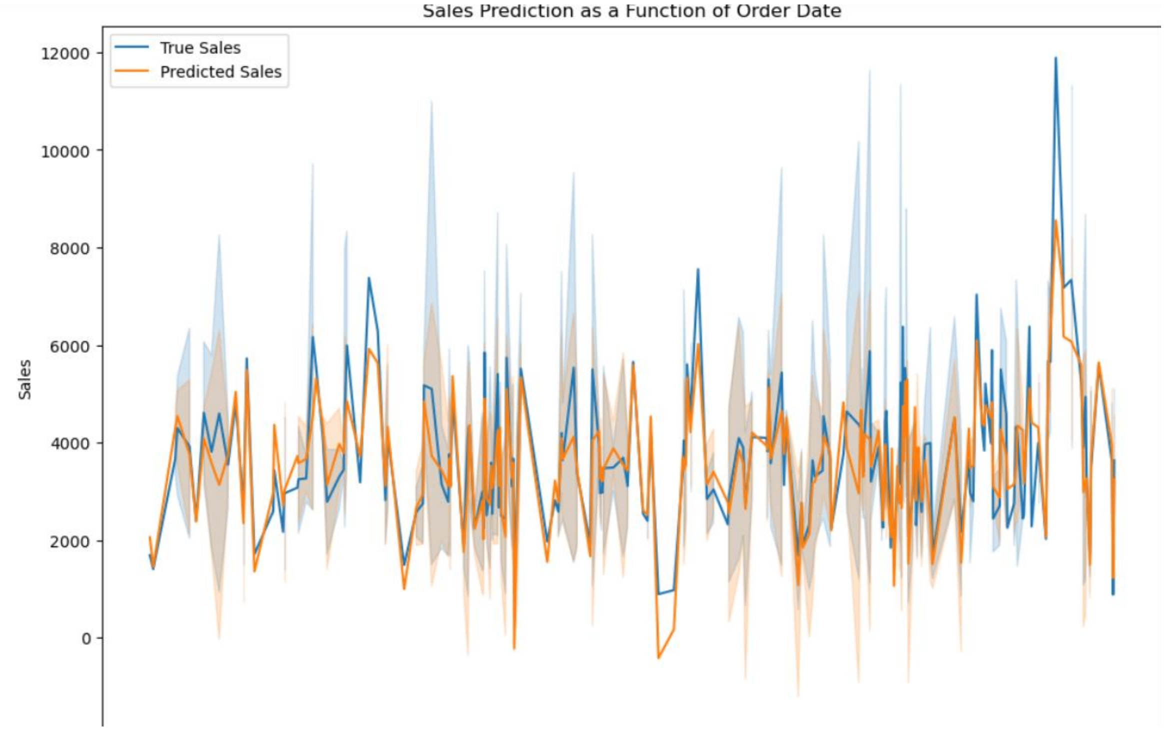
4. METHODOLOGY --- Flowchart is compulsory.

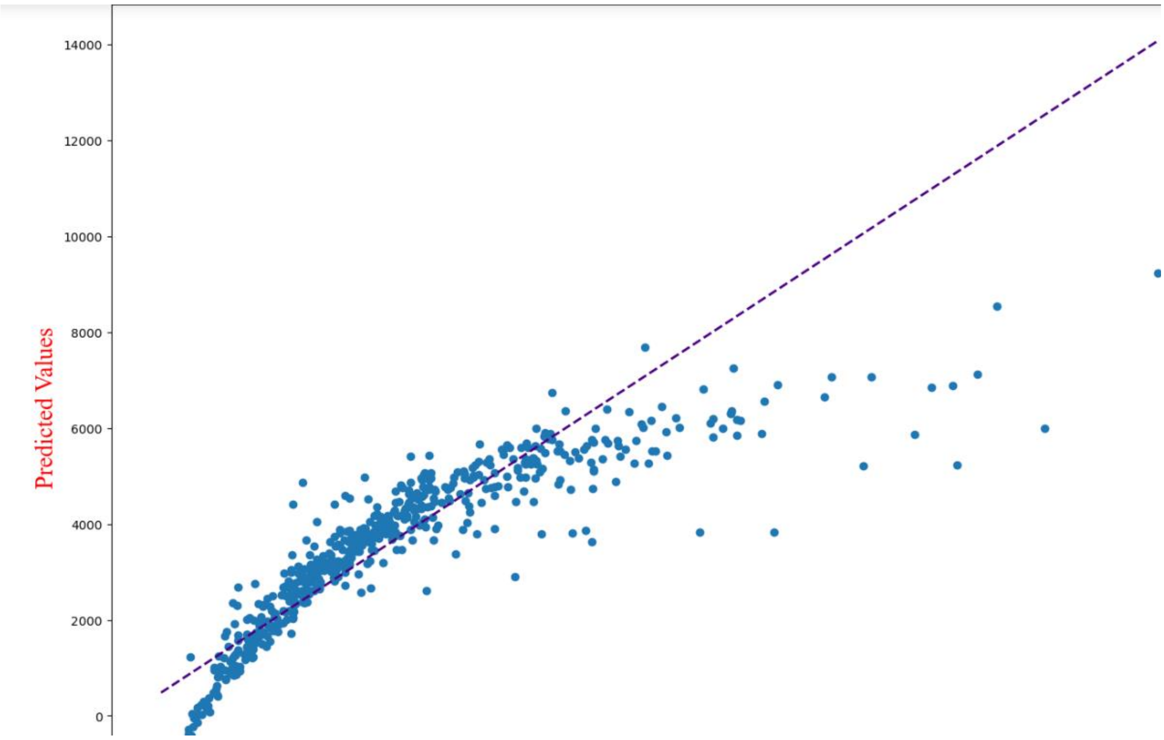
Our analysis follows a structured approach with several steps:

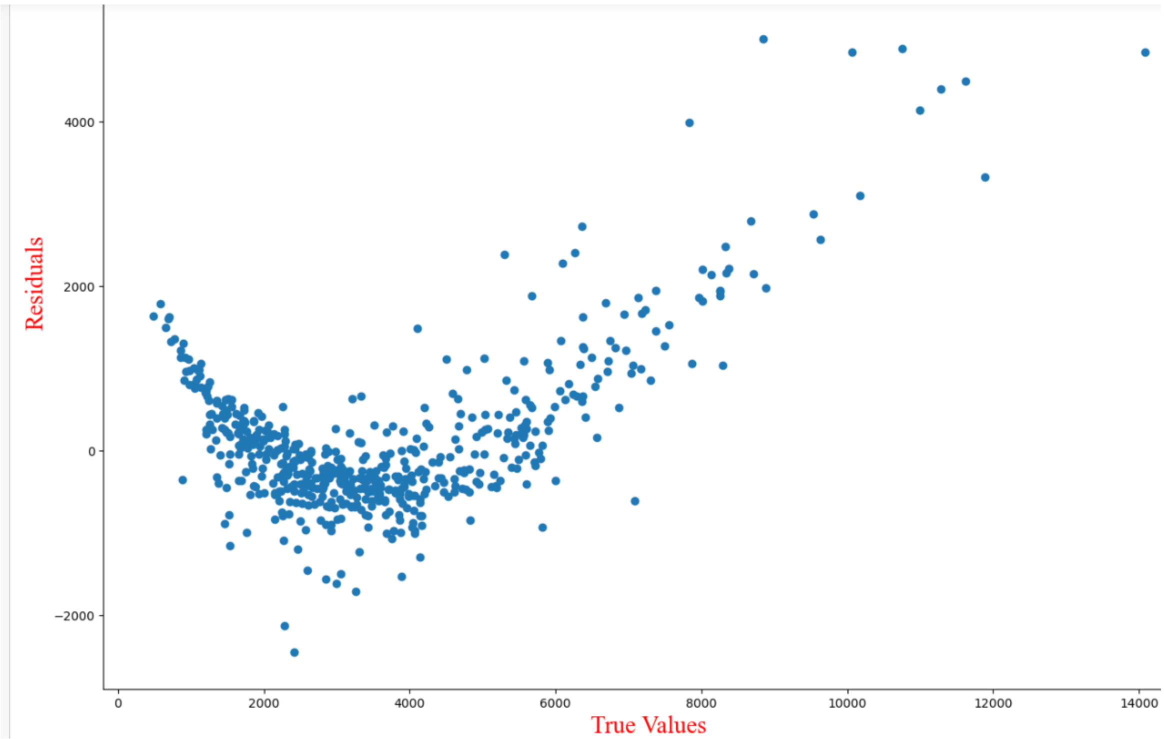
1. **Data Gathering and Cleaning:** We start by importing sales data into a Pandas DataFrame and prepare it for analysis. This might involve handling missing values or inconsistencies.
2. **Exploring the Data:** We then perform Exploratory Data Analysis (EDA) to understand the distribution of sales figures and how different sales attributes relate to each other.
3. **Building Sales Models:** We employ multiple linear regression techniques uncover the relationships between sales attributes and actual sales figures.
4. **Assessing Model Performance:** We evaluate the effectiveness of these sales models using appropriate metrics to determine which model performs best.
5. **Data Visualization:** Finally, we leverage tools like Seaborn, and Matplotlib to create clear and informative charts and graphs that help us interpret the results of our analysis.

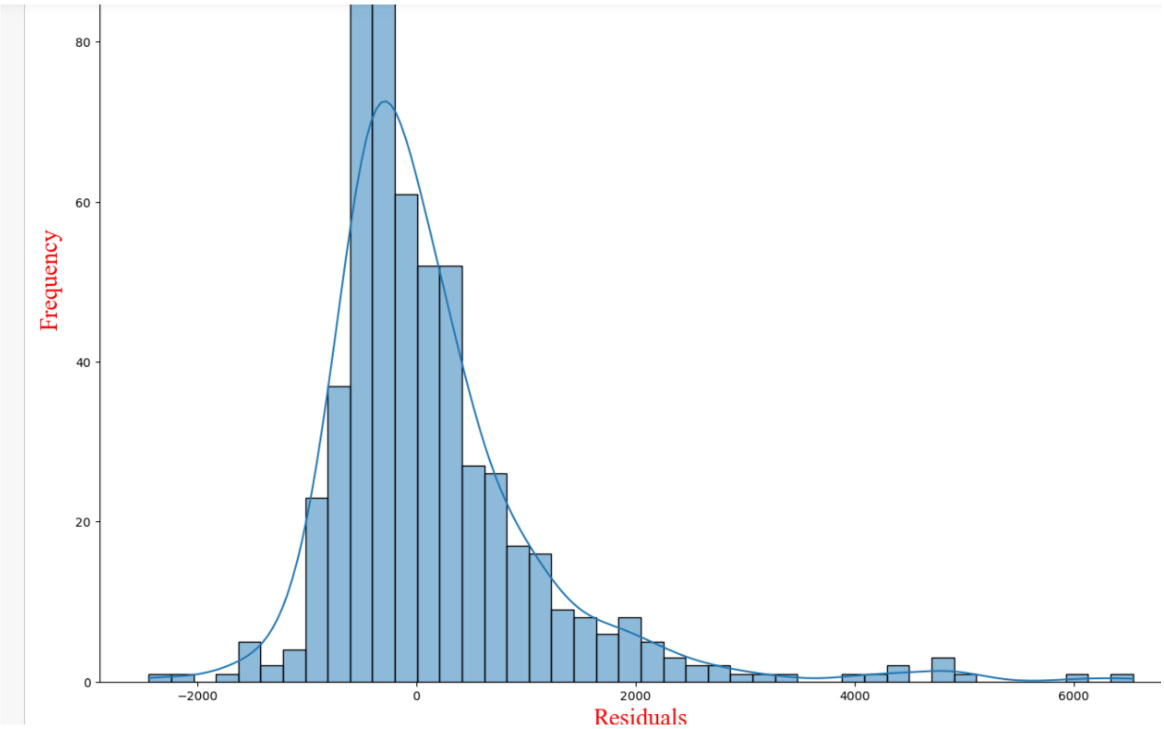

START
Import Data (Pandas)
to DataFrame
Clean & Prepare Data
(handle missing values
/inconsistent data
Exploratory Data
Analysis (EDA):
- Sales Distribution
- Attribute
- Relationships
Build Sales Models
(linear reg, lasso,ridge,
random forest, ARIMA,
SARIMA)
Assesing
model performance
Data visualization,
matplotlib, seaborn
END

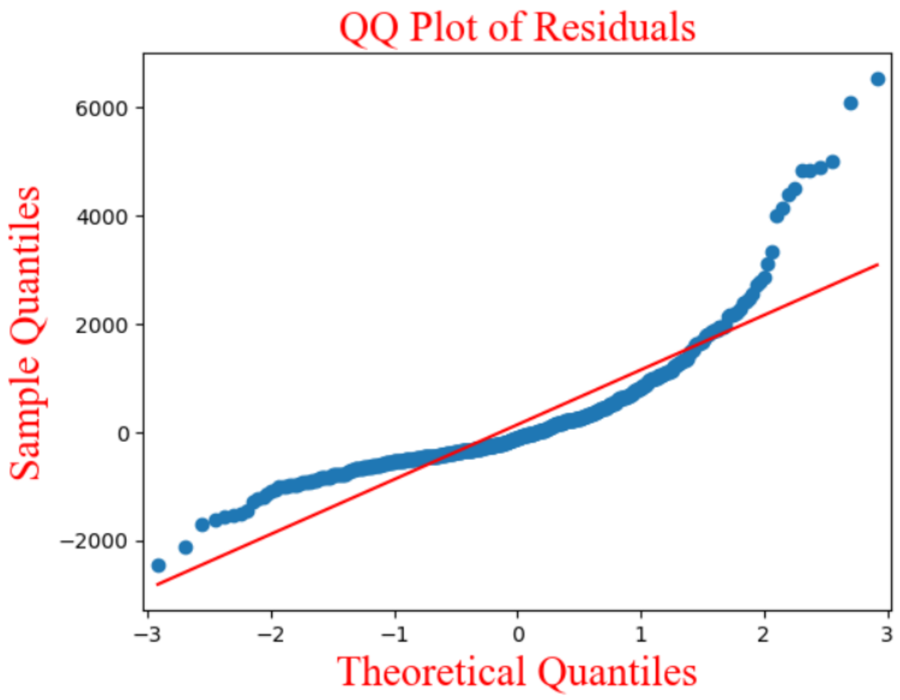
5. EXPERIMENTAL RESULTS

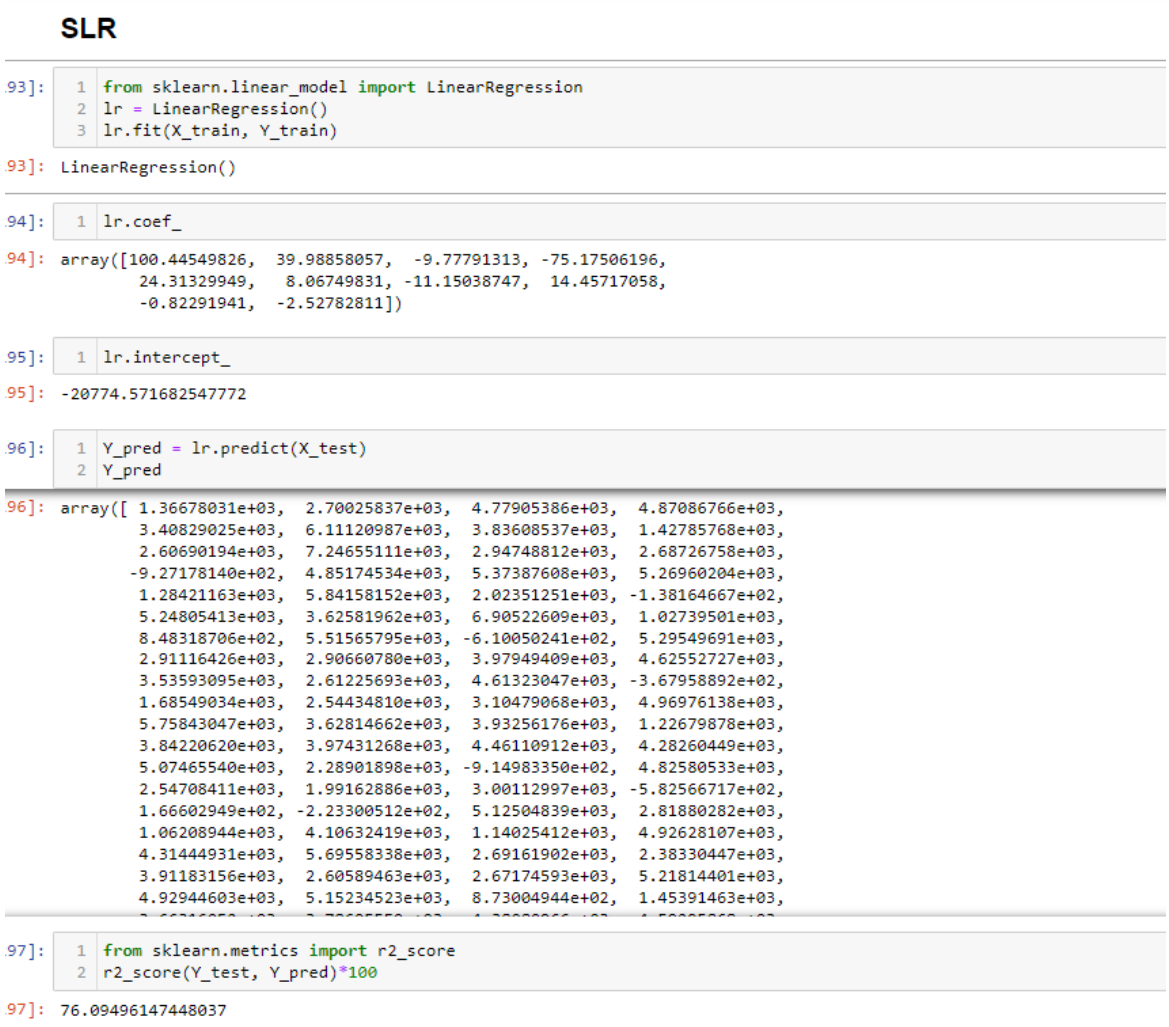





Residuals
4000
2000
0
−2000
0
2000
4000
6000
8000
10000
12000
14000
True Values


Frequency
80
60
40
20
0
−2000
0
2000
4000
6000
Residuals





SLR

```
93]: from sklearn.linear_model import LinearRegression
     lr = LinearRegression()
     lr.fit(X_train, Y_train)

93]: LinearRegression()
```

```
94]: lr.coef_

94]: array([100.44549826,  39.98858057,  -9.77791313, -75.17506196,
         24.31329949,   8.06749831, -11.15038747,  14.45717058,
         -0.82291941,  -2.52782811])
```

```
95]: lr.intercept_

95]: -20774.571682547772
```

```
96]: Y_pred = lr.predict(X_test)
     Y_pred

96]: array([ 1.36678031e+03,  2.70025837e+03,  4.77905386e+03,  4.87086766e+03,
         3.40829025e+03,  6.11120987e+03,  3.83608537e+03,  1.42785768e+03,
         2.60690194e+03,  7.24655111e+03,  2.94748812e+03,  2.68726758e+03,
        -9.27178140e+02,  4.85174534e+03,  5.37387608e+03,  5.26960204e+03,
         1.28421163e+03,  5.84158152e+03,  2.02351251e+03, -1.38164667e+02,
         5.24805413e+03,  3.62581962e+03,  6.90522609e+03,  1.02739501e+03,
         8.48318706e+02,  5.51565795e+03, -6.10050241e+02,  5.29549691e+03,
         2.91116426e+03,  2.90660780e+03,  3.97949409e+03,  4.62552727e+03,
         3.53593095e+03,  2.61225693e+03,  4.61323047e+03, -3.67958892e+02,
         1.68549034e+03,  2.54434810e+03,  3.10479068e+03,  4.96976138e+03,
         5.75843047e+03,  3.62814662e+03,  3.93256176e+03,  1.22679878e+03,
         3.84220620e+03,  3.97431268e+03,  4.46110912e+03,  4.28260449e+03,
         5.07465540e+03,  2.28901898e+03, -9.14983350e+02,  4.82580533e+03,
         2.54708411e+03,  1.99162886e+03,  3.00112997e+03, -5.82566717e+02,
         1.66602949e+02, -2.23300512e+02,  5.12504839e+03,  2.81880282e+03,
         1.06208944e+03,  4.10632419e+03,  1.14025412e+03,  4.92628107e+03,
         4.31444931e+03,  5.69558338e+03,  2.69161902e+03,  2.38330447e+03,
         3.91183156e+03,  2.60589463e+03,  2.67174593e+03,  5.21814401e+03,
         4.92944603e+03,  5.15234523e+03,  8.73004944e+02,  1.45391463e+03,
```

```
97]: from sklearn.metrics import r2_score
     r2_score(Y_test, Y_pred)*100

97]: 76.09496147448037
```

MLR

In [298]:
```
from sklearn.preprocessing import PolynomialFeatures
pf = PolynomialFeatures(degree=3)

X_poly = pf.fit_transform(X)
X_poly
```

Out[298]:
```
array([[1.0000e+00, 3.0000e+01, 9.5700e+01, ..., 3.6450e+04, 1.4580e+04,
        5.8320e+03],
       [1.0000e+00, 3.4000e+01, 8.1350e+01, ..., 2.6136e+04, 2.3760e+03,
        2.1600e+02],
       [1.0000e+00, 4.1000e+01, 9.4740e+01, ..., 1.2696e+04, 1.6560e+03,
        2.1600e+02],
       ...,
       [1.0000e+00, 4.3000e+01, 1.0000e+02, ..., 1.5246e+04, 6.4680e+03,
        2.7440e+03],
       [1.0000e+00, 3.4000e+01, 6.2240e+01, ..., 6.0000e+00, 3.6000e+01,
        2.1600e+02],
       [1.0000e+00, 4.7000e+01, 6.5520e+01, ..., 2.4642e+04, 1.1988e+04,
        5.8320e+03]])
```

In [299]:
```
from sklearn.linear_model import LinearRegression

poly_reg = LinearRegression()
poly_reg.fit(X_poly, Y)
```

Out[299]:
```
LinearRegression()
```

In [300]:
```
# X_train_poly = pf.fit_transform(X_train)
# X_test_poly = pf.transform(X_test)
```

In [301]:
```
# from sklearn.linear_model import LinearRegression
# pr = LinearRegression()
# pr.fit(X_train_poly, Y_train)
```

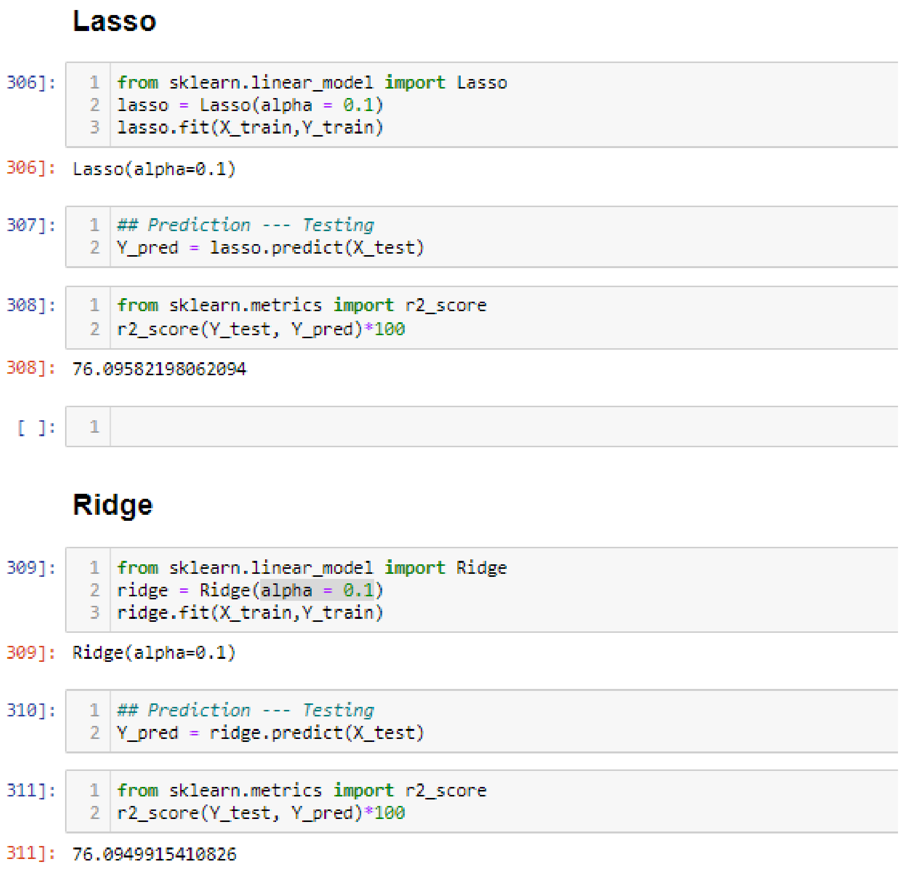
In [302]:
```
# Y_pred = pr.predict(X_test_poly)
# Y_pred
```

In [303]:
```
Y_pred_2 = poly_reg.predict(X_poly)
Y_pred_2
```

Out[303]:
```
array([3061.4851515 , 2719.66423914, 3929.89551422, ..., 6108.25005743,
       1870.94479248, 2928.16789123])
```

In [304]:
```
from sklearn.metrics import r2_score
r2_score(Y_test, Y_pred)*100
```

Out[304]:
```
76.09496147448037
```

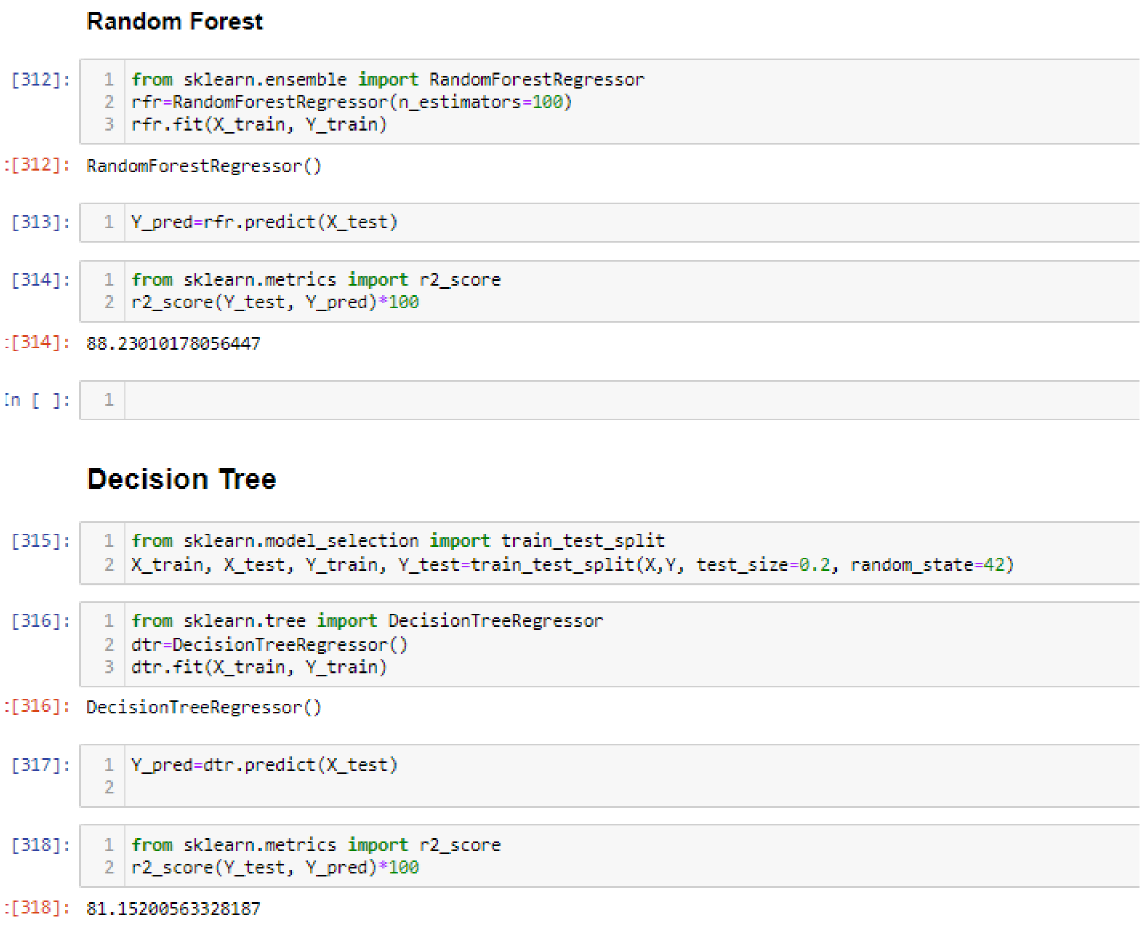


Lasso

```
306]: from sklearn.linear_model import Lasso
      lasso = Lasso(alpha = 0.1)
      lasso.fit(X_train,Y_train)

306]: Lasso(alpha=0.1)

307]: ## Prediction --- Testing
      Y_pred = lasso.predict(X_test)

308]: from sklearn.metrics import r2_score
      r2_score(Y_test, Y_pred)*100

308]: 76.09582198062094

[ ]:
```

Ridge

```
309]: from sklearn.linear_model import Ridge
      ridge = Ridge(alpha = 0.1)
      ridge.fit(X_train,Y_train)

309]: Ridge(alpha=0.1)

310]: ## Prediction --- Testing
      Y_pred = ridge.predict(X_test)

311]: from sklearn.metrics import r2_score
      r2_score(Y_test, Y_pred)*100

311]: 76.0949915410826
```



Random Forest

[312]:
```
from sklearn.ensemble import RandomForestRegressor
rfr=RandomForestRegressor(n_estimators=100)
rfr.fit(X_train, Y_train)
```

:[312]:
```
RandomForestRegressor()
```

[313]:
```
Y_pred=rfr.predict(X_test)
```

[314]:
```
from sklearn.metrics import r2_score
r2_score(Y_test, Y_pred)*100
```

:[314]:
```
88.23010178056447
```

In [ ]:

Decision Tree

[315]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test=train_test_split(X,Y, test_size=0.2, random_state=42)
```

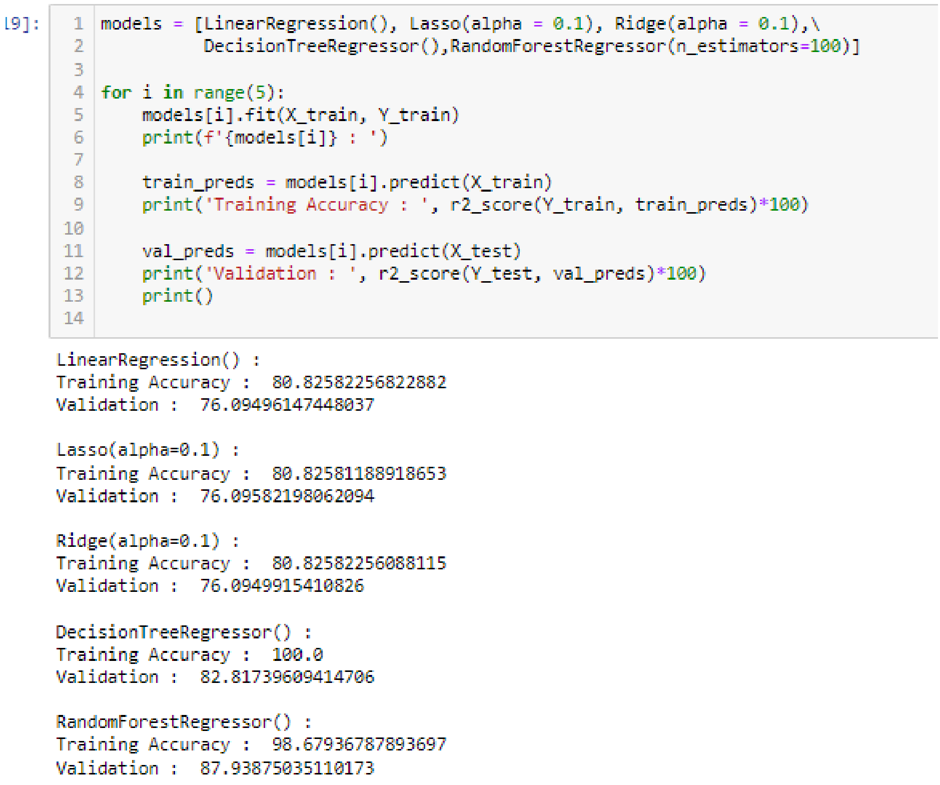
[316]:
```
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor()
dtr.fit(X_train, Y_train)
```

:[316]:
```
DecisionTreeRegressor()
```

[317]:
```
Y_pred=dtr.predict(X_test)

```

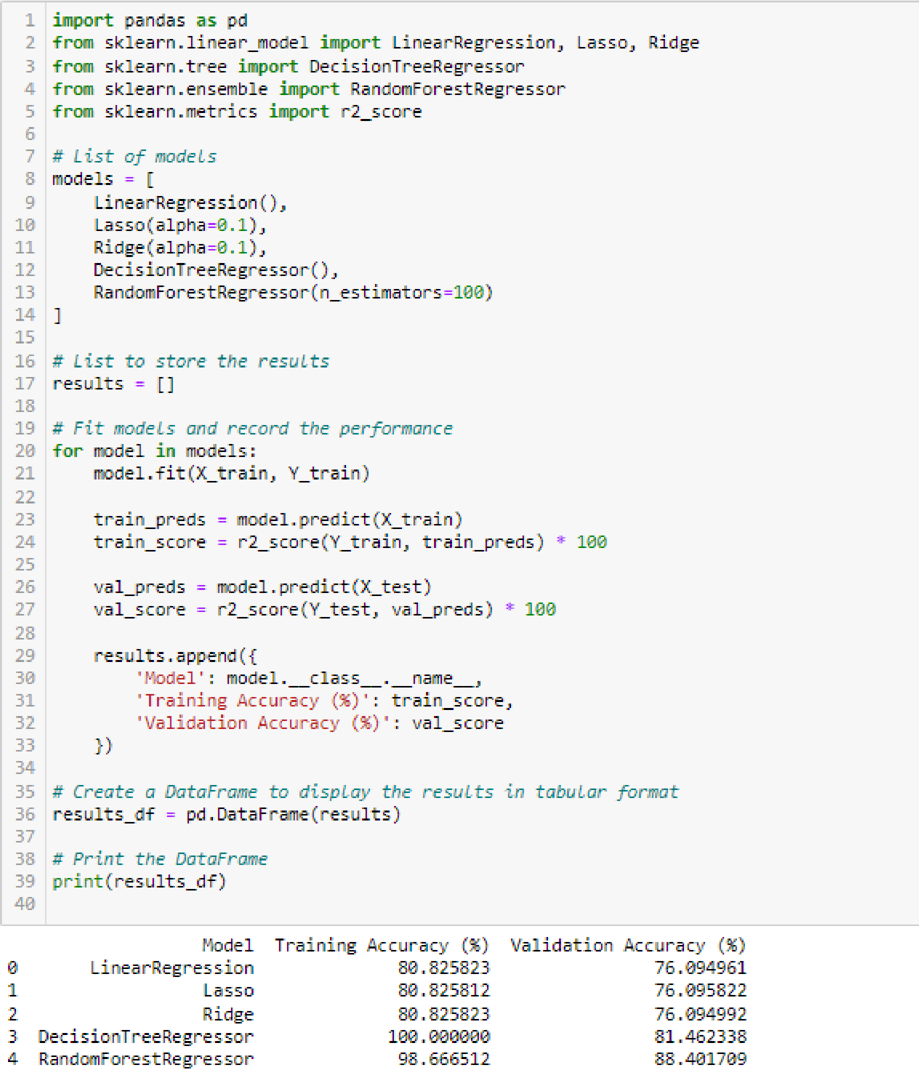
[318]:
```
from sklearn.metrics import r2_score
r2_score(Y_test, Y_pred)*100
```

:[318]:
```
81.15200563328187
```



19]:
```
models = [LinearRegression(), Lasso(alpha = 0.1), Ridge(alpha = 0.1),\
          DecisionTreeRegressor(),RandomForestRegressor(n_estimators=100)]

for i in range(5):
    models[i].fit(X_train, Y_train)
    print(f'{models[i]} : ')

    train_preds = models[i].predict(X_train)
    print('Training Accuracy : ', r2_score(Y_train, train_preds)*100)

    val_preds = models[i].predict(X_test)
    print('Validation : ', r2_score(Y_test, val_preds)*100)
    print()

```

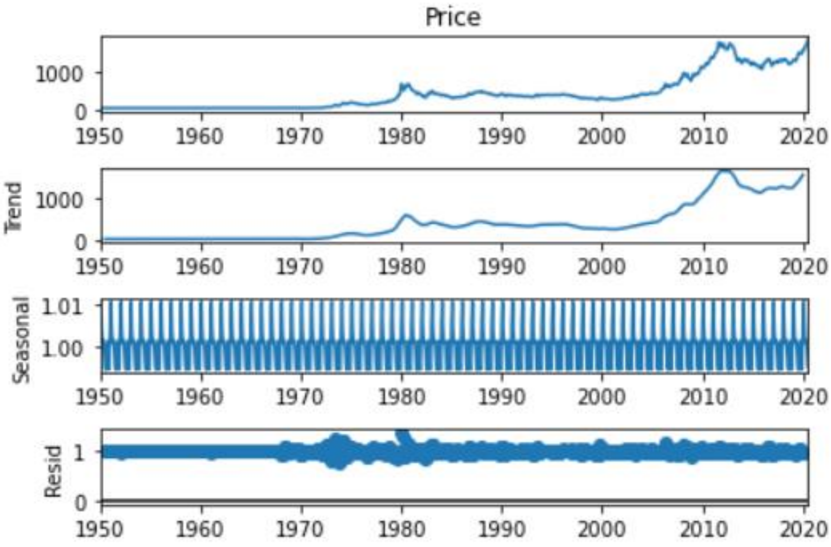
```
LinearRegression() :
Training Accuracy :  80.82582256822882
Validation :  76.09496147448037

Lasso(alpha=0.1) :
Training Accuracy :  80.82581188918653
Validation :  76.09582198062094

Ridge(alpha=0.1) :
Training Accuracy :  80.82582256088115
Validation :  76.0949915410826

DecisionTreeRegressor() :
Training Accuracy :  100.0
Validation :  82.81739609414706

RandomForestRegressor() :
Training Accuracy :  98.67936787893697
Validation :  87.93875035110173
```



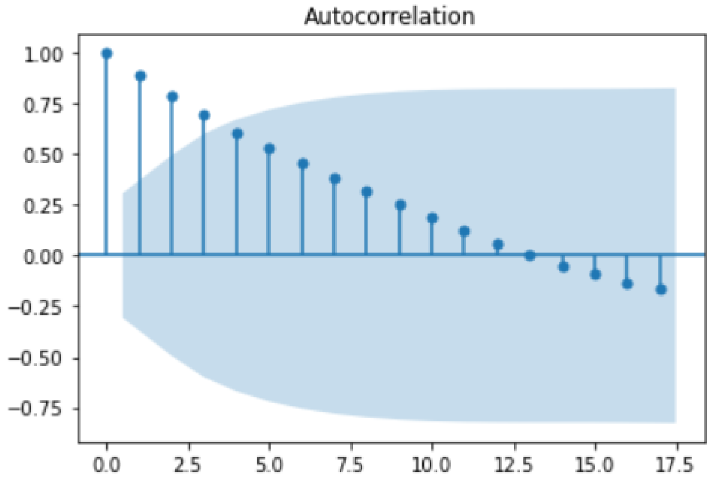
```python
import pandas as pd
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score

# List of models
models = [
    LinearRegression(),
    Lasso(alpha=0.1),
    Ridge(alpha=0.1),
    DecisionTreeRegressor(),
    RandomForestRegressor(n_estimators=100)
]

# List to store the results
results = []

# Fit models and record the performance
for model in models:
    model.fit(X_train, Y_train)

    train_preds = model.predict(X_train)
    train_score = r2_score(Y_train, train_preds) * 100

    val_preds = model.predict(X_test)
    val_score = r2_score(Y_test, val_preds) * 100

    results.append({
        'Model': model.__class__.__name__,
        'Training Accuracy (%)': train_score,
        'Validation Accuracy (%)': val_score
    })

# Create a DataFrame to display the results in tabular format
results_df = pd.DataFrame(results)

# Print the DataFrame
print(results_df)

```

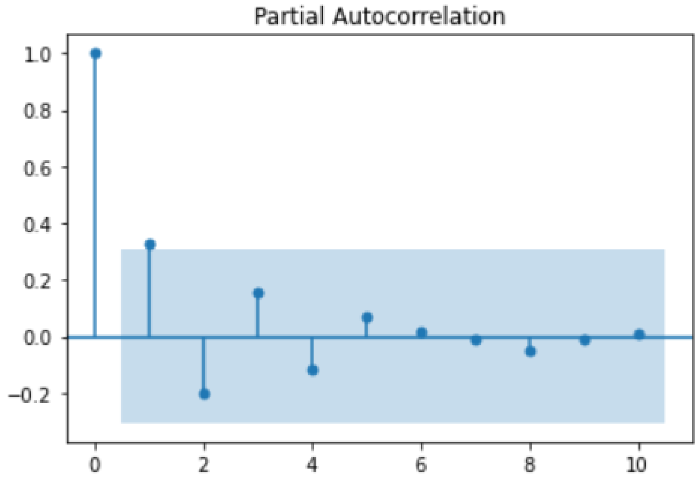
```
                   Model  Training Accuracy (%)  Validation Accuracy (%)
0       LinearRegression              80.825823                76.094961
1                  Lasso              80.825812                76.095822
2                  Ridge              80.825823                76.094992
3  DecisionTreeRegressor             100.000000                81.462338
4  RandomForestRegressor              98.666512                88.401709
```

Time Series Results



ACF AND PACF





6. CONCLUSION

Through the application of data analysis techniques, our investigation has yielded significant insights into the dynamics of sales data**.** By leveraging regression analysis and visualization tools, we have been able to elucidate previously unknown relationships between various sales attributes and overall sales performance. These findings contribute to the development of a more comprehensive understanding of sales trends within the market.

By acknowledging and comprehending the identified patterns, businesses are empowered to make data-driven decisions that optimize their sales strategies. This optimization, in turn, leads to demonstrably improved sales performance.

However, the pursuit of knowledge in this domain remains an ongoing endeavor. We posit that further exploration utilizing advanced methodologies and incorporating a broader range of sales-related factors has the potential to yield even more refined sales predictions.

7. REFERENCES

https://www.youtube.com/watch?v=2XGSIlgUBDI&ab_channel=KrishNaik

https://www.geeksforgeeks.org/ml-linear-regression/

https://colab.research.google.com/github/jesperdramsch/skillshare-data-science/blob/book/book/notebooks/33%20-%20Machine%20learning%20classification.ipynb#:~:text=By%20analyzing%20the%20relationships%20between,efficient%20and%20effective%20decision%2Dmaking.

https://colab.research.google.com/github/TannerGilbert/Tutorials/blob/master/Scikit-Learn-Tutorial/5.%20Classification%20Algorithms.ipynb#scrollTo=umKHCwXB_OPr

8. GitHub Repository Link

https://github.com/RajK19/BFM_2024_Sales-Data-Analysis