Sales Data Analysis

This project, titled “Sales Data Analysis,” focuses on analyzing Walmart’s sales dataset to derive actionable insights that can support better decision-making. The dataset includes information such as store details, weekly sales, holiday indicators, temperature, fuel price, and economic factors like CPI and unemployment rate. By performing detailed data cleaning, preprocessing, and exploratory analysis, the project aims to identify sales patterns, seasonal fluctuations, and the impact of external variables on sales performance.

Through the use of Python (VS Code) the project transforms raw data into meaningful visualizations and metrics. The insights gained from this analysis help highlight top-performing stores and products, understand seasonal demand, and identify business opportunities for improving inventory management and sales forecasting.

Objective or Problem Statement

The goal of this project is to analyze and visualize Walmart sales data to uncover meaningful business insights. It aims to identify sales trends, seasonal patterns, and product performance across different stores and categories.

The project seeks to solve key business challenges such as recognizing top and low-performing products**,** understanding factors that influence sales (like holidays and discounts), and improving inventory management and sales forecasting through data-driven insights.

Tools and Technologies Used

Python (VS Code Environment): Used for data cleaning, preprocessing, and exploratory data analysis (EDA).

- **Libraries:**
  - **Pandas** – for data manipulation and analysis
  - **Matplotlib / Seaborn** – for creating visualizations and identifying sales trends

Database Used

The dataset used in this project is the Walmart Sales Data, obtained from Kaggle**.** It contains detailed sales information from multiple Walmart stores.

The dataset includes approximately 10,000+ records of sales transactions collected over a specific time period.

It consists of various fields such as:

- Store ID
- Date of Sale
- Weekly Sales
- Holiday Flag
- Temperature
- Fuel Price
- CPI (Consumer Price Index)
- Unemployment Rate

This dataset provides both numerical and categorical data, enabling analysis of sales trends, store performance, and external factors affecting sales.

Methodology

1. Data Collection

- The Walmart Sales Dataset was imported from Kaggle.
- The dataset was loaded into Python (VS Code) using the Pandas library for analysis.
- Data integrity was verified by checking for duplicates and missing values.

2. Data Preprocessing

- Missing values were handled appropriately, and duplicate records were removed.
- Date columns were converted to a standard format for time-series analysis.
- Data types were standardized to ensure consistency for visualization and calculations.

3. Exploratory Data Analysis (EDA)

- Conducted exploratory analysis using Pandas, Matplotlib, and Seaborn to identify:
  - Sales trends over time
  - Seasonal and regional variations
  - High-performing and low-performing stores and products
- Visualizations such as line charts, bar graphs, and heatmaps were created to interpret patterns.

4. Data Visualization

- Total Sales by Store and Category
- Impact of holidays and temperature on weekly sales
- Correlations between economic indicators and sales performance

5. Insights and Recommendations

- Key insights were summarized to help management understand:
  - Which stores and products drive the most revenue
  - Seasonal demand fluctuations
  - The impact of external factors like holidays and unemployment on sales

Features Implemented/ Key Visuals

1. Data Cleaning and Preprocessing

- Removed duplicate and null values to ensure data accuracy.
- Standardized date and numerical formats for consistent analysis.

2. Trend Analysis

- Created time-series visualizations to show overall sales performance over months and years.
- Identified seasonal patterns and holiday impacts on weekly sales.

3. Product and Store Performance

- Built charts to highlight top-performing stores and product categories.
- Compared sales across different regions and time periods using bar and line graphs.

4. Correlation Analysis

- Used visual tools to examine the relationship between economic indicators (e.g., CPI, fuel price, unemployment) and sales trends.

6. Insights Presentation

- Combined all findings into a comprehensive visual summary to support business decision-making in forecasting, inventory planning, and promotional strategies.

Results/ Key Insights

1. Sales Performance Trends

- Steady growth was observed during specific months, indicating seasonal demand spikes, particularly around holiday periods.
- Significant sales dips occurred during off-season months, suggesting potential opportunities for promotions during those times.

2. Store and Regional Insights

- A few stores consistently contributed to the highest revenue, showing strong customer demand and operational efficiency.
- Some regions displayed lower sales figures, pointing to the need for localized marketing efforts or revised pricing strategies.

3. Product-Level Insights

- Top-performing product categories accounted for a large share of total revenue.
- Low-selling products were identified, helping management focus on optimizing inventory and discontinuing underperforming items.

4. Impact of External Factors

- External variables such as holidays**,** temperature changes, and unemployment rate had a noticeable impact on weekly sales.
- Holiday weeks showed a clear increase in sales volume, confirming the importance of seasonal campaigns.

5. Business Recommendations

- Focus marketing and stock efforts on high-demand months and holiday periods.
- Strengthen supply chain and logistics for top-performing stores.
- Use predictive analytics for better sales forecasting and demand planning in future operations.

Challenges Faced

1. Data Quality Issues

Challenge: The dataset contained missing values, duplicate records, and inconsistent date formats, which affected data accuracy.

Solution: Used Python (Pandas) functions to clean the data duplicates were removed, missing values were handled using imputation techniques, and date formats were standardized for time-series analysis.

2. Large and Unstructured Data

Challenge: Managing and analyzing large amounts of raw sales data made it difficult to quickly generate insights.

Solution: Utilized data filtering and grouping techniques in Python to organize and summarize the dataset efficiently.

3. Identifying Seasonal Trends

Challenge: Detecting clear seasonal sales patterns was complex due to overlapping product categories and time periods.

Solution: Applied line charts and time-series visualizations in Matplotlib to clearly observe and interpret monthly and seasonal variations.

4. Interpretation of Correlation Factors

Challenge: Understanding the influence of external factors like fuel price and CPI on sales performance required careful statistical interpretation.

Solution: Used correlation heatmaps and pair plots to visually examine relationships and draw accurate conclusions.

Conclusion

The Sales Data Analysis project successfully provided meaningful insights into the sales performance of Walmart stores using data-driven techniques. Through data cleaning, exploration, and visualization, the project identified key trends, high-performing products, and the impact of external factors such as holidays and economic conditions on weekly sales.

The findings revealed that a small number of stores and products contribute to the majority of total sales, highlighting areas of strong business performance. Seasonal and holiday periods were found to significantly boost revenue, confirming the importance of targeted marketing and inventory planning during these times. Overall, the project demonstrates how effective data analysis and visualization can play a crucial role in understanding business performance and supporting data-driven decision-making in the retail sector.

Screenshots

walmart_analysis.py > ...

```python
import os
import warnings
warnings.filterwarnings("ignore")

import matplotlib
import sys

# Try to force an interactive backend commonly available.
# If this fails, we'll still try to show plots and provide a helpful message.
for preferred in ("TkAgg", "Qt5Agg", "MacOSX"):
    try:
        matplotlib.use(preferred, force=True)
        break
    except Exception:
        pass

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.seasonal import seasonal_decompose
import joblib

DATA_PATH = "Walmart.csv"   # put your file here


def load_data(path):
    if not os.path.exists(path):
        raise FileNotFoundError(f"{path} not found.")
    sample = pd.read_csv(path, nrows=5)
    date_cols = [c for c in sample.columns if "date" in c.lower() or "week" in c.lower()]
    if date_cols:
        df = pd.read_csv(path, parse_dates=[date_cols[0]])
        df.rename(columns={date_cols[0]: "Date"}, inplace=True)
    else:
        df = pd.read_csv(path)
        df.rename(columns={df.columns[0]: "Date"}, inplace=True)
        df["Date"] = pd.to_datetime(df["Date"], errors="coerce")
    return df


def clean_data(df):
    df = df.copy().drop_duplicates()
    sales_col = next((c for c in df.columns if "sale" in c.lower()), None)
    if not sales_col:
        raise ValueError("No column with 'sales' in the name was found.")
```

walmart_analysis.py > explore_and_show

```python
def clean_data(df):
        raise ValueError("No column with 'sales' in the name was found.")
    df.rename(columns={sales_col: "Weekly_Sales"}, inplace=True)
    df["Weekly_Sales"] = pd.to_numeric(df["Weekly_Sales"], errors="coerce")
    df["Weekly_Sales"].fillna(df["Weekly_Sales"].median(), inplace=True)

    for col in list(df.columns):
        if "store" in col.lower() and col != "Store":
            df.rename(columns={col: "Store"}, inplace=True)
        if "dept" in col.lower() and col != "Dept":
            df.rename(columns={col: "Dept"}, inplace=True)

    df.sort_values(by=["Date"], inplace=True)
    df.reset_index(drop=True, inplace=True)
    return df


def explore_and_show(df):
    print("Matplotlib backend:", matplotlib.get_backend())
    df["Date"] = pd.to_datetime(df["Date"], errors="coerce")
    ts = df.set_index("Date").resample("W")["Weekly_Sales"].sum()

    # 1) Total sales over time
    fig, ax = plt.subplots(figsize=(12, 5))
    ax.plot(ts.index, ts.values, linewidth=1.5)
    ax.set_title("Total Weekly Sales Over Time")
    ax.set_xlabel("Date")
    ax.set_ylabel("Weekly Sales")
    plt.show(block=True)   # <--- blocking: window appears and stays until closed

    # 2) Top 10 stores (if available)
    if "Store" in df.columns:
        top_stores = df.groupby("Store")["Weekly_Sales"].sum().nlargest(10)
        fig, ax = plt.subplots(figsize=(10, 5))
        sns.barplot(x=top_stores.index.astype(str), y=top_stores.values, ax=ax)
        ax.set_title("Top 10 Stores by Total Sales")
        ax.set_xlabel("Store")
        ax.set_ylabel("Total Sales")
        plt.setp(ax.get_xticklabels(), rotation=45)
        plt.show(block=True)

    # 3) Seasonal decomposition if enough points
    if len(ts) >= 104:
        try:
            result = seasonal_decompose(ts, model="additive", period=52, extrapolate_trend='freq')
            # result.plot() creates subplots; show them
            result.plot()
            plt.show(block=True)
```

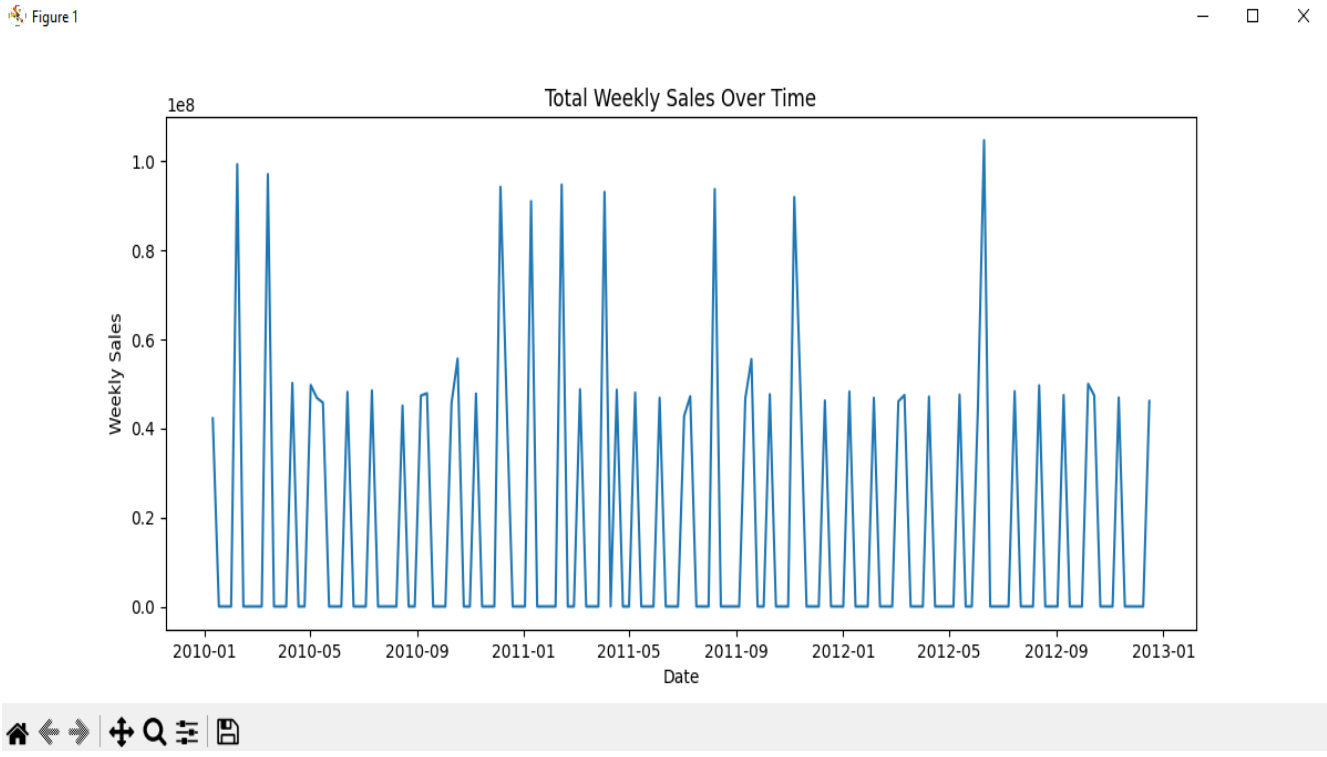
walmart_analysis.py > ...

```
64  def explore_and_show(df):
95          except Exception as e:
96              print("Seasonal decomposition error:", e)
97      else:
98          print("Not enough data for seasonal decomposition (need >=104 weekly points).")
99
100
101 def forecasting_and_show(df):
102     ts = df.set_index("Date").resample("W")["Weekly_Sales"].sum()
103     if len(ts) < 10:
104         print("Too few data points for forecasting.")
105         return
106
107     def create_lags(series, lags=[1, 2, 3, 4, 52]):
108         df_lag = pd.DataFrame({'y': series})
109         for lag in lags:
110             df_lag[f'lag_{lag}'] = df_lag['y'].shift(lag)
111         df_lag["rolling_4"] = df_lag['y'].shift(1).rolling(4).mean()
112         return df_lag.dropna()
113
114     df_feat = create_lags(ts)
115     X = df_feat.drop("y", axis=1)
116     y = df_feat["y"]
117     split = int(len(X) * 0.8)
118     X_train, X_test = X.iloc[:split], X.iloc[split:]
119     y_train, y_test = y.iloc[:split], y.iloc[split:]
120
121     model = RandomForestRegressor(n_estimators=100, random_state=42)
122     model.fit(X_train, y_train)
123     preds = model.predict(X_test)
124     preds_series = pd.Series(preds, index=y_test.index)
125
126     try:
127         rmse = mean_squared_error(y_test, preds_series, squared=False)
128     except TypeError:
129         rmse = np.sqrt(mean_squared_error(y_test, preds_series))
130     print(f"Forecast RMSE: {rmse:.2f}")
131
```

```
    # Plot actual vs predicted (show only)
    fig, ax = plt.subplots(figsize=(10, 5))
    ax.plot(y_test.index, y_test.values, label="Actual", linewidth=2)
    ax.plot(preds_series.index, preds_series.values, label="Predicted", linewidth=2)
    ax.set_title("Actual vs Predicted Sales")
    ax.set_xlabel("Date")
    ax.set_ylabel("Weekly Sales")
    ax.legend()
    ax.grid(True, linestyle="--", alpha=0.6)
    plt.show(block=True)


def main():
    print("Running show-only script. Make sure you run this in a desktop environment.")
    df = load_data(DATA_PATH)
    df = clean_data(df)
    explore_and_show(df)
    forecasting_and_show(df)
    print("Finished. All plots were displayed (no files saved).")


if __name__ == "__main__":
    main()
```



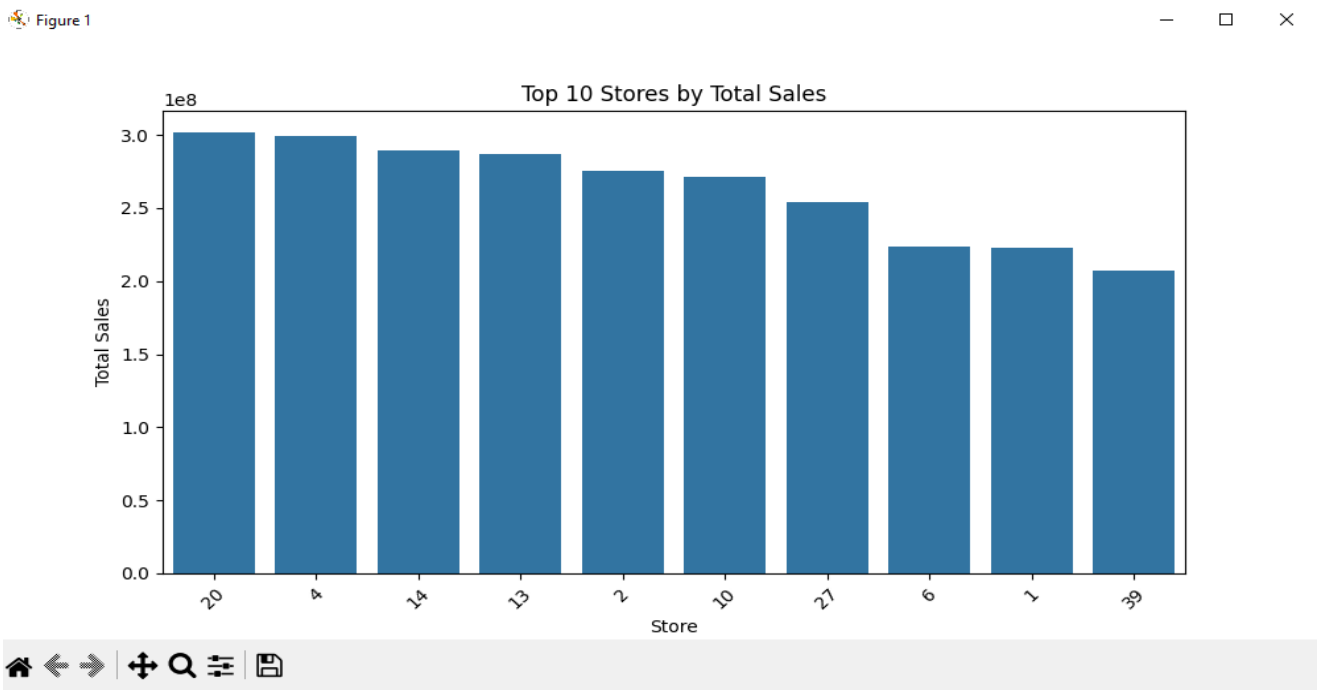


Figure 1
Top 10 Stores by Total Sales
1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Total Sales
20
4
14
13
2
10
27
6
1
39
Store

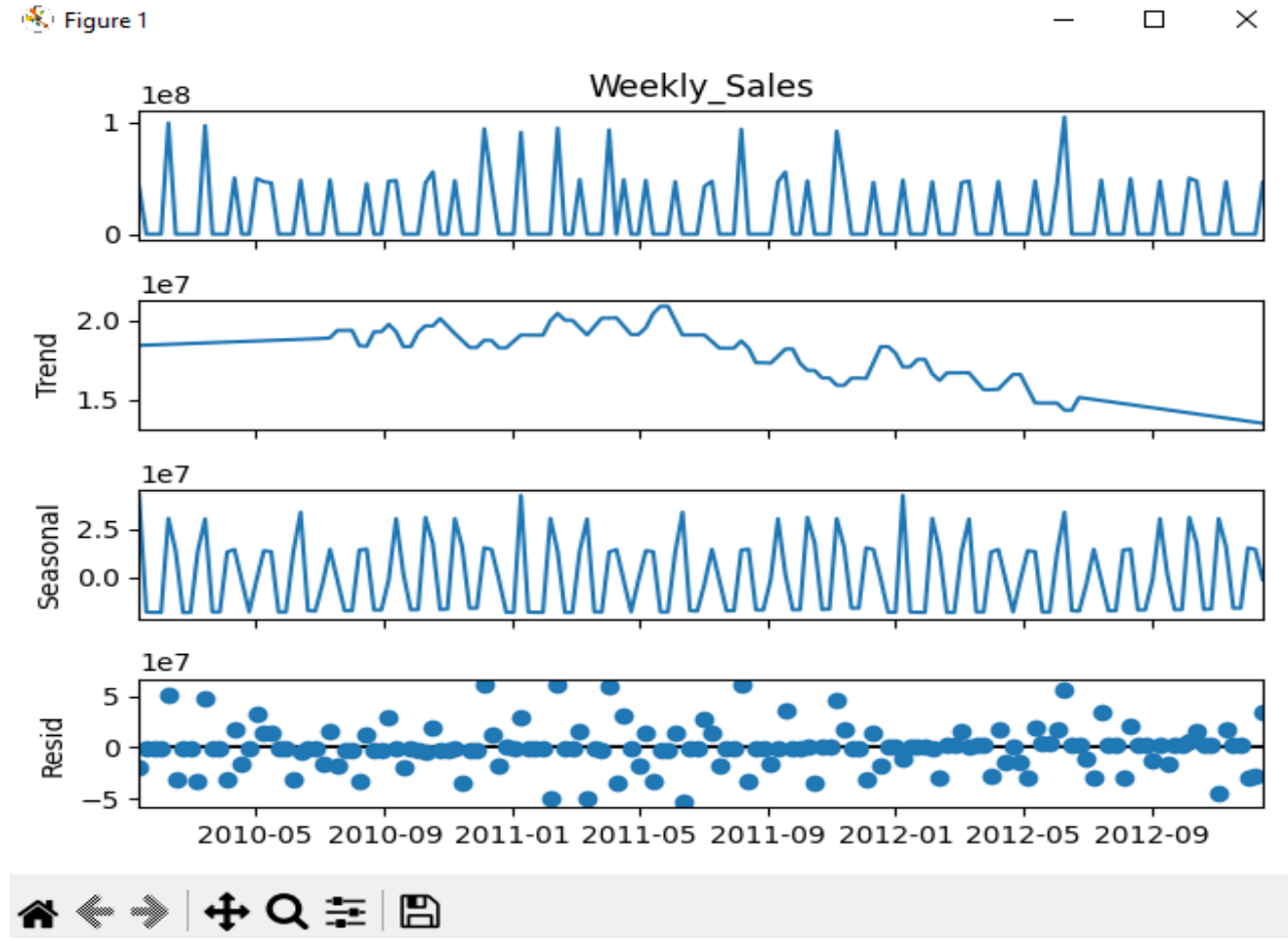
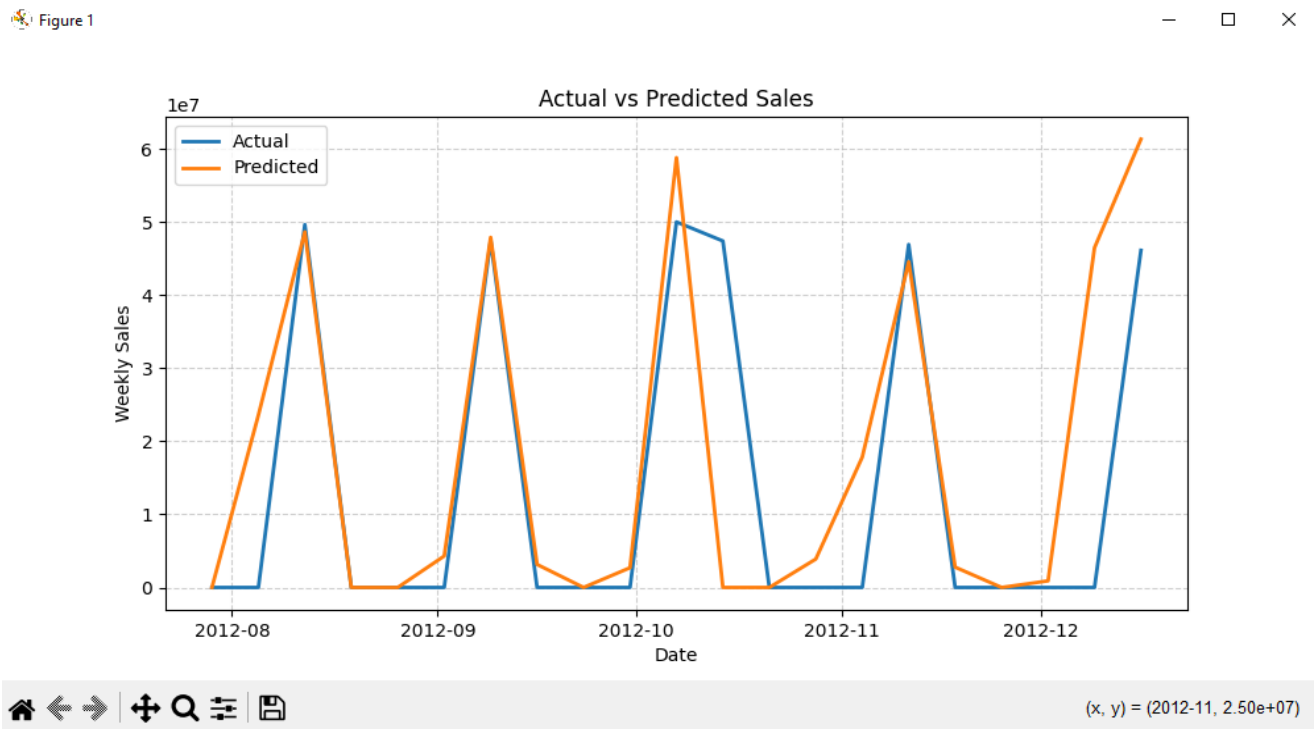


Figure 1
Weekly_Sales
1e8
1
0
1e7
2.0
1.5
Trend
1e7
2.5
0.0
Seasonal
1e7
5
0
−5
Resid
2010-05
2010-09
2011-01
2011-05
2011-09
2012-01
2012-05
2012-09



Link to GitHub

GitHub: https://github.com/Yashu-teach/Sales-Data-Analysis-Project