Retail Sales Trend & Seasonality Analysis

Abstract

In this project, a comprehensive time series analysis was performed on daily retail sales data collected from multiple outlets of a nationwide retail company. The primary objective was to analyze the underlying structure of the sales data by identifying long-term trends, recurring seasonal patterns, and random fluctuations. The analysis included time series visualization, seasonal decomposition, autocorrelation and partial autocorrelation analysis (ACF and PACF), and stationarity testing using the Augmented Dickey-Fuller (ADF) test. The results revealed the presence of a clear upward trend, strong weekly seasonality, and non-stationarity in the original series. This statistical validation prepares the dataset for accurate future forecasting using time series models such as ARIMA and SARIMA.

Problem Statement

Retail sales data is influenced by various time-dependent factors such as weekends, seasonal shopping behavior, festivals, and promotional campaigns. Without understanding whether the data exhibits long-term trends, repeating seasonal patterns, or randomness, forecasting models may produce misleading predictions.

The main objective of this project is to statistically analyze retail sales data in order to:

- Identify trends and seasonal effects
- Measure temporal dependence using autocorrelation
- Test whether the dataset satisfies stationarity assumptions
- Prepare the data for reliable forecasting models

Table of Contents

1. Dataset Loading and Preprocessing

The retail sales dataset was loaded using the Pandas library. The Date column was converted into datetime format and set as the index to ensure compatibility with time series analysis techniques.

Key Preprocessing Steps:

- Converted the Date column to datetime format
- Set Date as the time index
- Checked for missing values and data consistency
- Verified continuity of observations

These preprocessing steps ensured that the dataset was properly structured for time-dependent statistical analysis.

2. Time Series Visualization

A line plot was generated to visualize daily sales behavior over time.

Observations:

- Sales values display an overall increasing pattern
- Regular fluctuations indicate recurring seasonal behavior
- Short-term variations suggest the presence of noise and promotional effects

This visualization provided an initial understanding of sales dynamics and revealed the existence of trend and seasonality in the data.

3. Seasonal Decomposition (Trend, Seasonality, Residuals)

Seasonal decomposition was performed using an additive model with a weekly seasonal period.

The time series was decomposed into four components:

- **Observed:** Original sales data
- **Trend:** Long-term upward movement in sales
- **Seasonal:** Repeating weekly pattern showing higher sales on specific days
- **Residual:** Random noise not explained by trend or seasonality

Key Insight:

The presence of a strong trend and consistent seasonal pattern indicates that sales behavior follows a systematic structure rather than being purely random.

4. Autocorrelation Function (ACF) Analysis

The Autocorrelation Function (ACF) plot was used to analyze the relationship between current sales values and their previous values.

Findings:

- High autocorrelation at lower lags
- Gradual decay of correlations
- Repeating spikes at regular intervals indicating seasonality

These results suggest that past sales values strongly influence future sales, confirming the non-stationary nature of the data.

5. Partial Autocorrelation Function (PACF) Analysis

Partial Autocorrelation Function (PACF) analysis was performed to examine direct relationships between observations at specific time lags.

Observations:

- Significant spikes at early lags
- Reduced influence at higher lags

This information helps in identifying suitable autoregressive (AR) terms for building forecasting models such as ARIMA.

6. Stationarity Testing using Augmented Dickey-Fuller (ADF) Test

The Augmented Dickey-Fuller (ADF) test was conducted to statistically evaluate the stationarity of the time series.

Results:

- ADF Statistic ≈ −0.41
- p-value ≈ 0.90

Since the p-value is greater than the significance level of 0.05, the null hypothesis of non-stationarity cannot be rejected.

Conclusion:

The time series is non-stationary due to the presence of trend and seasonal components.

7. Interpretation of Results

The analysis confirms that the retail sales data exhibits:

- A strong upward trend indicating growth in sales

- Weekly seasonality showing higher sales on weekends
- High autocorrelation reflecting dependence on past values
- Non-stationarity requiring transformation before forecasting

To make the data suitable for predictive modeling, differencing and seasonal adjustments are necessary.

8. Summary

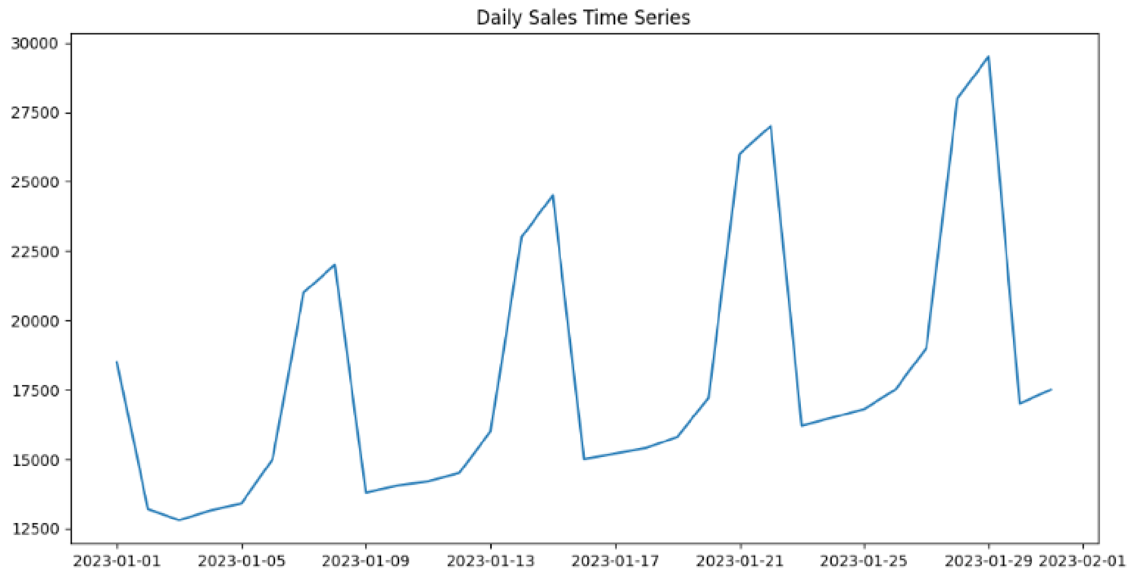
This project successfully analyzed the statistical properties of retail sales time series data. Through visualization, decomposition, correlation analysis, and stationarity testing, key patterns such as trend, seasonality, and randomness were identified.

Implementation:

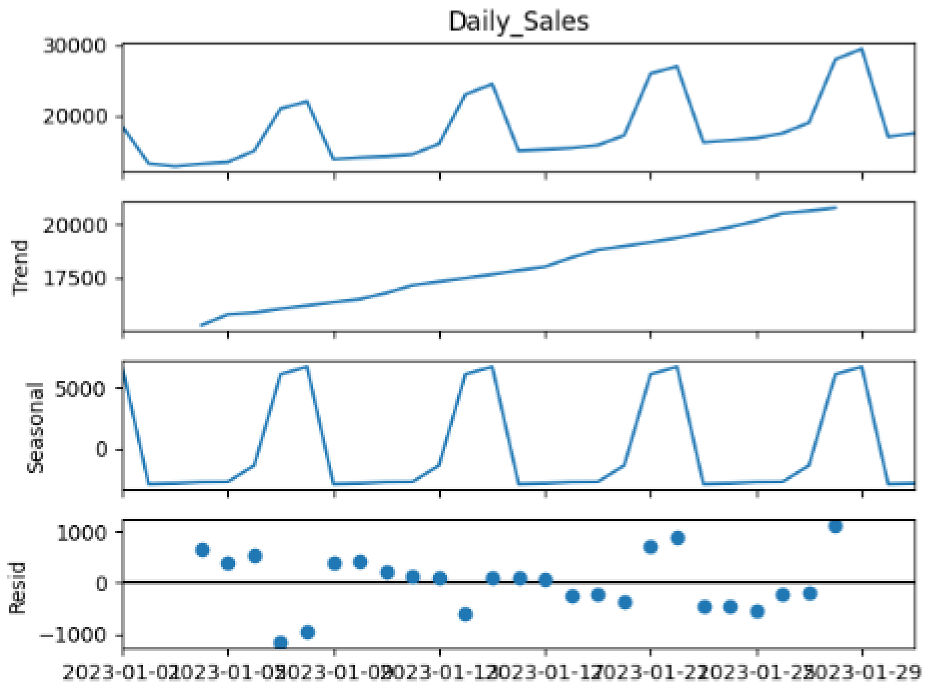
```
[2]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import warnings
     warnings.filterwarnings("ignore")
     from statsmodels.tsa.seasonal import seasonal_decompose
     from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
     from statsmodels.tsa.stattools import adfuller
```

```
[3]: data = pd.read_csv(r"C:\Users\santh\Downloads\retail_sales_dataset\retail_sales.csv")
```

```
[4]: data['Date'] = pd.to_datetime(data['Date'])

     data.set_index('Date', inplace=True)

     ts = data['Daily_Sales']

     plt.figure(figsize=(12,6))
     plt.plot(ts)
     plt.title("Daily Sales Time Series")
     plt.show()
```

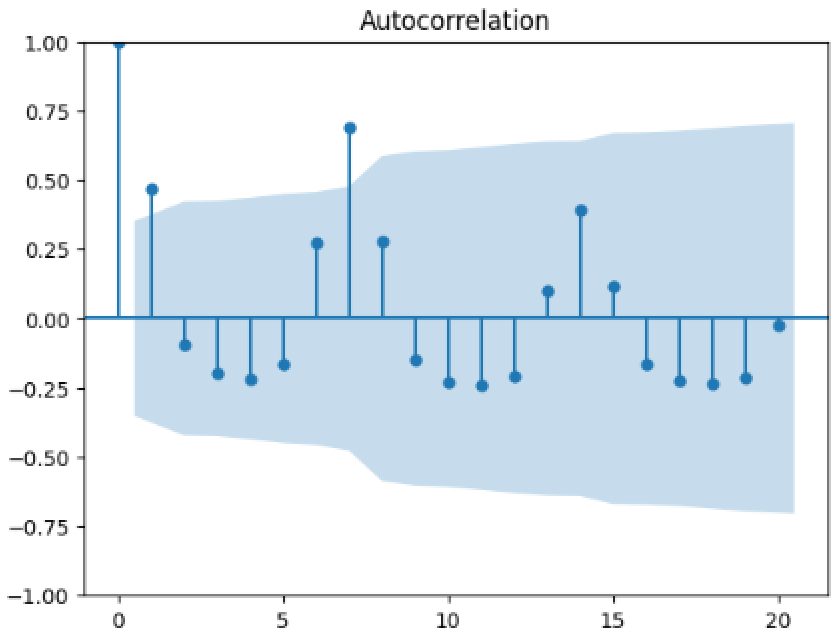


[5]:
```
decomposition = seasonal_decompose(ts, model='additive', period=7)

decomposition.plot()
plt.show()
```



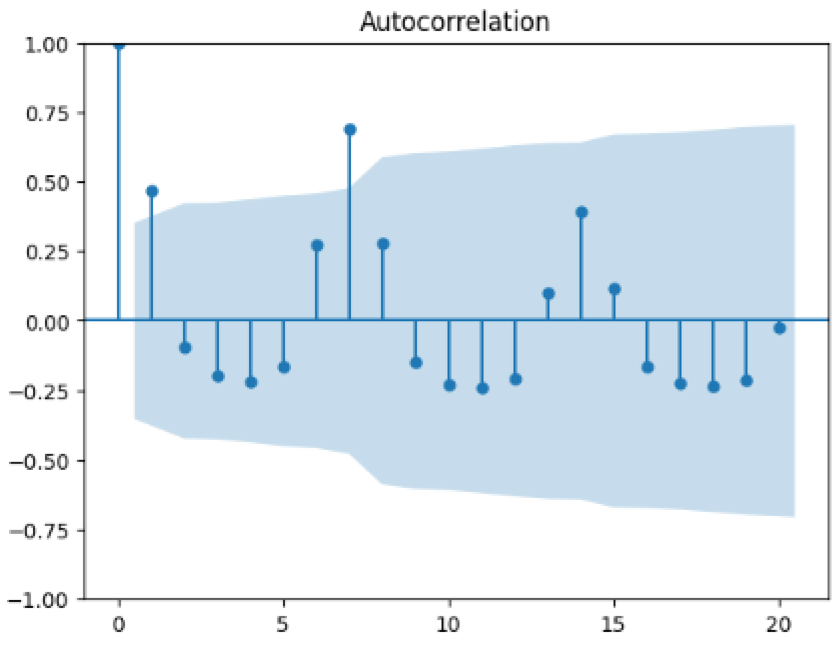
[6]:
```
plt.figure(figsize=(10,4))
plot_acf(ts, lags=20)
plt.show()
```

```
<Figure size 1000x400 with 0 Axes>
```



[7]:
```
plt.figure(figsize=(10,4))
plot_acf(ts, lags=20)
plt.show()
```

```
<Figure size 1000x400 with 0 Axes>
```

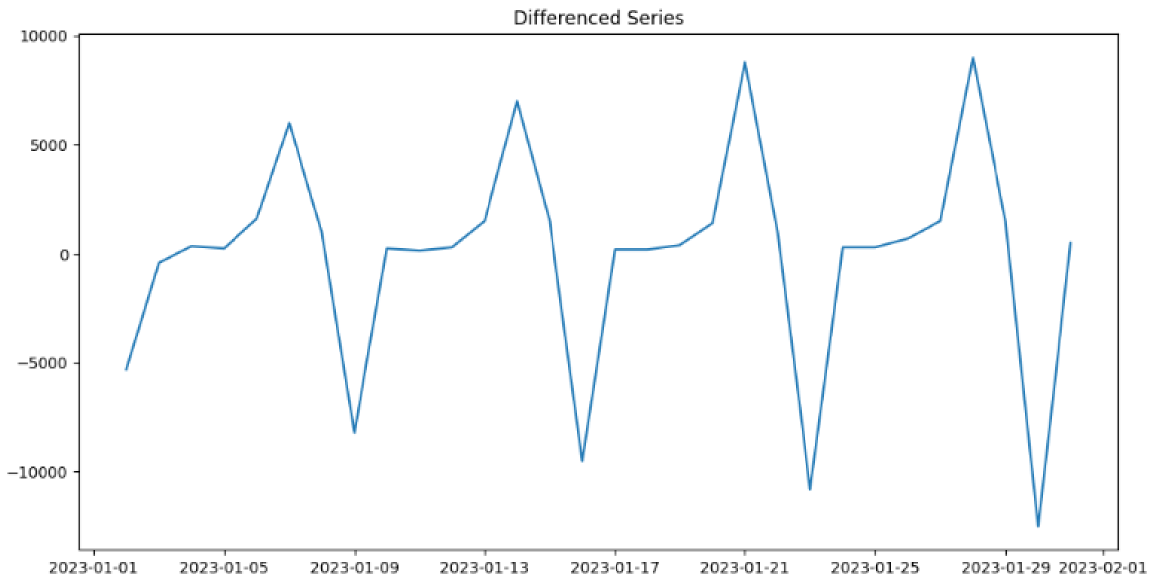


[8]:
```
result = adfuller(ts)

adf_stat = result[0]
p_value = result[1]
critical_values = result[4]

print("ADF Statistic:", adf_stat)
print("p-value:", p_value)
```

```
ADF Statistic: -0.01226606993787269
p-value: 0.9575058587612261
```

[9]:
```python
for key, value in critical_values.items():
    print("Critical Value", key, ":", value)

diff_ts = ts.diff().dropna()

plt.figure(figsize=(12,6))
plt.plot(diff_ts)
plt.title("Differenced Series")
plt.show()
```

```
Critical Value 1% : -3.7377092158564813
Critical Value 5% : -2.9922162731481485
Critical Value 10% : -2.635746736111111
```

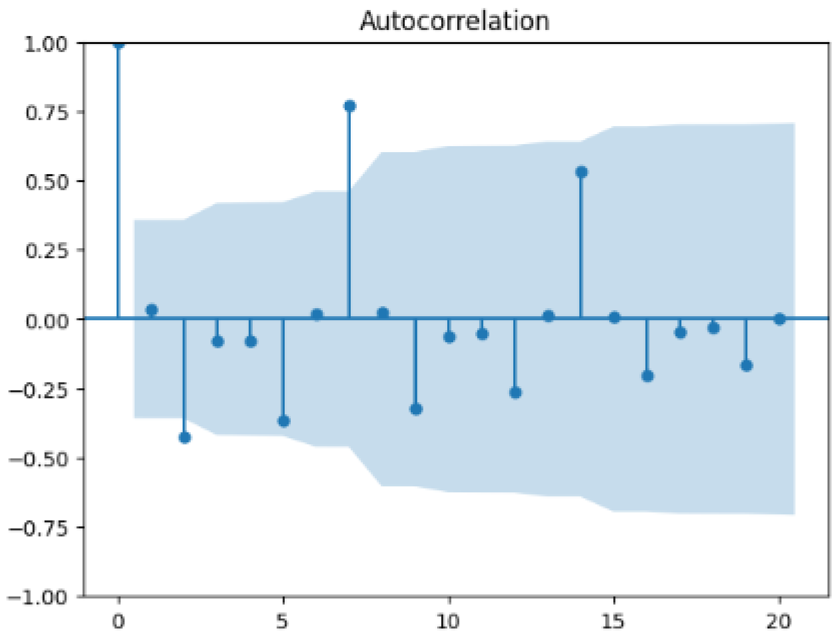


[10]:
```python
result_diff = adfuller(diff_ts)

print("ADF Statistic After Differencing:", result_diff[0])
print("p-value After Differencing:", result_diff[1])
```

```
ADF Statistic After Differencing: -49.49101445793401
p-value After Differencing: 0.0
```

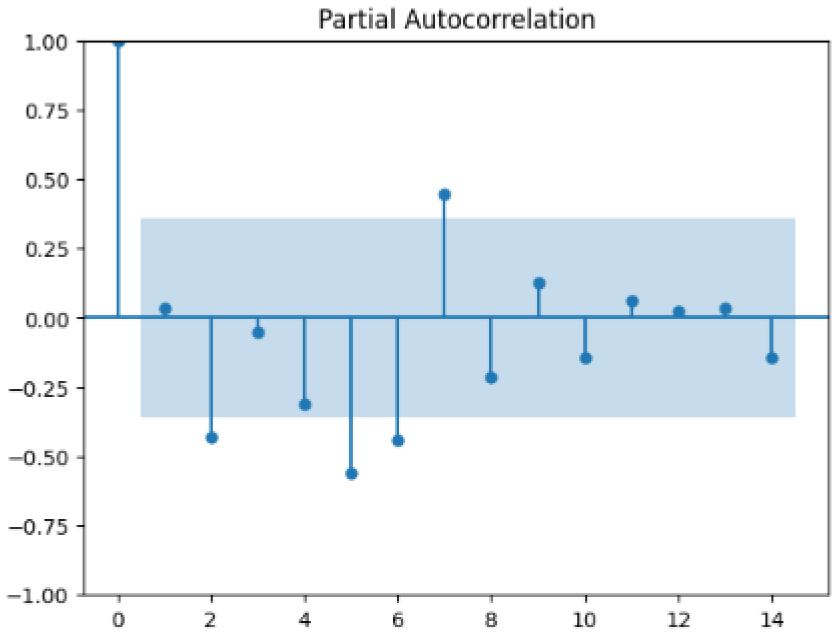
```
[11]: plt.figure(figsize=(10,4))
      plot_acf(diff_ts, lags=20)
      plt.show()
```

```
<Figure size 1000x400 with 0 Axes>
```



```
[13]: plt.figure(figsize=(10,4))
      plot_pacf(diff_ts, lags=14)
      plt.show()
```

```
<Figure size 1000x400 with 0 Axes>
```


Partial Autocorrelation