MoveInSync Assignment

Case Study Task1

Somesh Awasthi(MT2023172)

Date: 08-12-2024

Introduction

Objective: Design a **time series forecasting model** to predict values for the next **three months** based on the provided historical data Output should be filled csv file(test1.csv). Date range for testing - 01-10-2017 to 31-12-2017

Dataset Overview:

Training Data

- **Size**: The training dataset consists of **1,369 entries**.
- **Date Range**:
- **Features**:
    - **`date`**: The date column in `Object` format later converted to `datetime64[ns]` format.
    - **`price`**: The price column with integer values, representing the target variable.
- **Data Quality**:
    - All 1,369 entries in both columns are non-null, ensuring complete data availability for training.

Testing Data

- **Size**: The testing dataset consists of **92 entries**.
- **Date Range**: Covers the period from **01-10-2017 to 31-12-2017**.
- **Features**:
    - **`date`**: The date column in `datetime64[ns]` format.
    - **`price`**: The price column, is an empty column where we have to do prediction (all are `NaN`).

2. Data Preprocessing

2.1 Time Features Creation

To enhance the predictive power of the model, we engineer additional time-based features from the date column. These features are essential in capturing seasonal patterns, trends, and other temporal dynamics in the data.

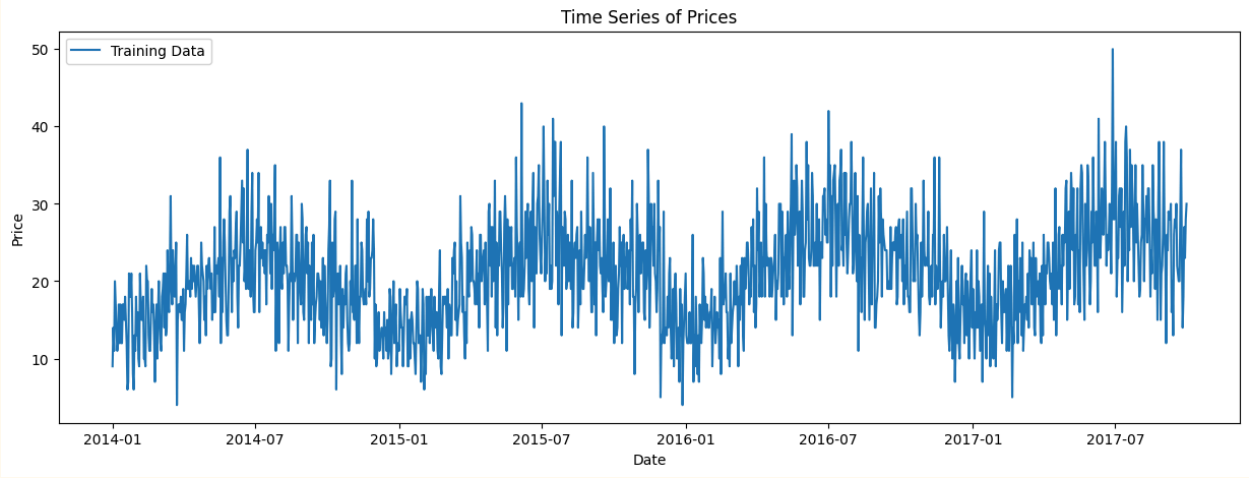
The `create_time_features` function extracts the following time-related features from the `date` column (which is set as the DataFrame index):

- **year**: The year extracted from the date.
- **month**: The month extracted from the date.
- **day**: The day of the month extracted from the date.
- **day_of_week**: The weekday as an integer (0 = Monday, 6 = Sunday).
- **day_of_year**: The day of the year (1 to 365/366).
- **quarter**: The fiscal quarter of the year (1 to 4).

Data Visualization

Time Series Plot

The time series plot visualizes the historical price data for the training period, showing the price trend over time.

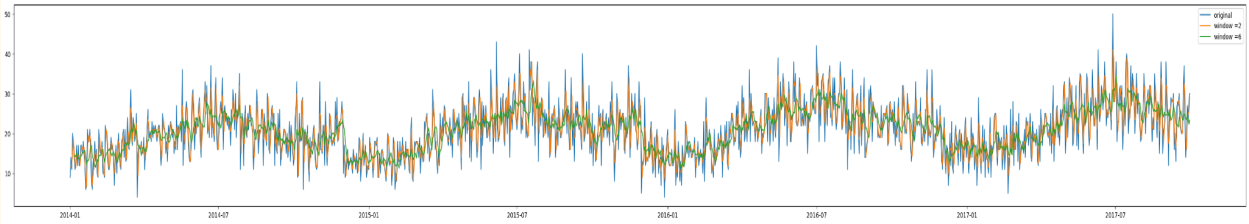


Price Distribution Plot

The distribution plot of the price data is created to understand the frequency of different price values in the training dataset.

Rolling Mean

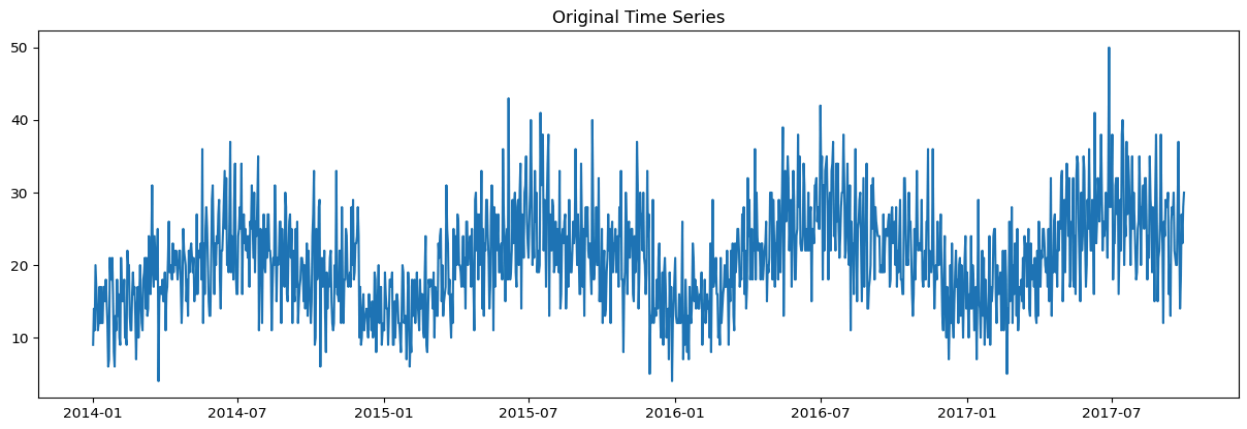
To understand the trend in the price data, rolling means with window sizes of 2 and 6 are computed and plotted alongside the original price data.



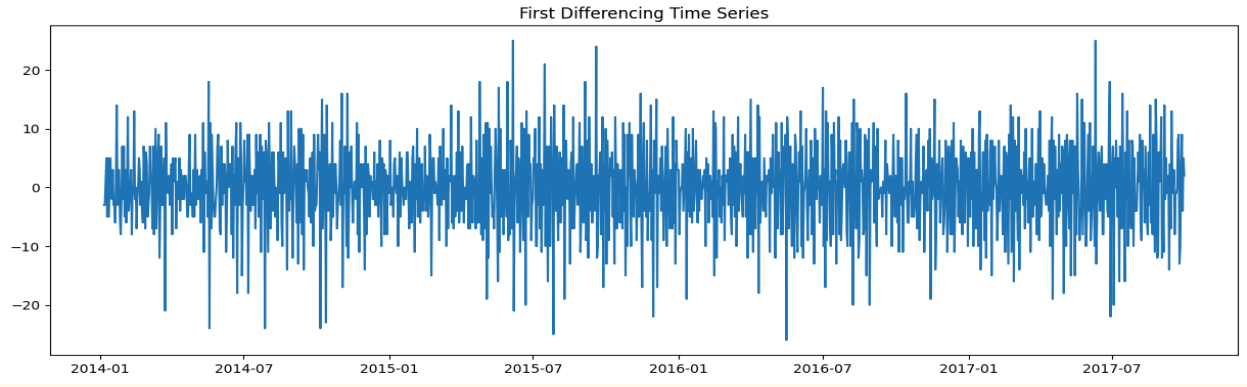
Stationarity Check:

Stationarity is a critical assumption in many time series forecasting models. A time series is considered stationary if its statistical properties (mean, variance, and autocorrelation) do not change over time. Non-stationary data can lead to inaccurate forecasting results.

- **ADF Test (Augmented Dickey-Fuller Test):** The **ADF test** is commonly used to check for stationarity. It tests the null hypothesis that a unit root is present (i.e., the time series is non-stationary).
    - **ADF Statistic**: A test statistic for the null hypothesis.
    - **p-value**: If this value is less than a chosen significance level (e.g., 0.05), the null hypothesis is rejected, suggesting the series is stationary.
    - **Critical Values**: These are threshold values for the test statistic. If the test statistic is less than the critical value for a given confidence level, it suggests rejecting the null hypothesis (i.e., the series is stationary).



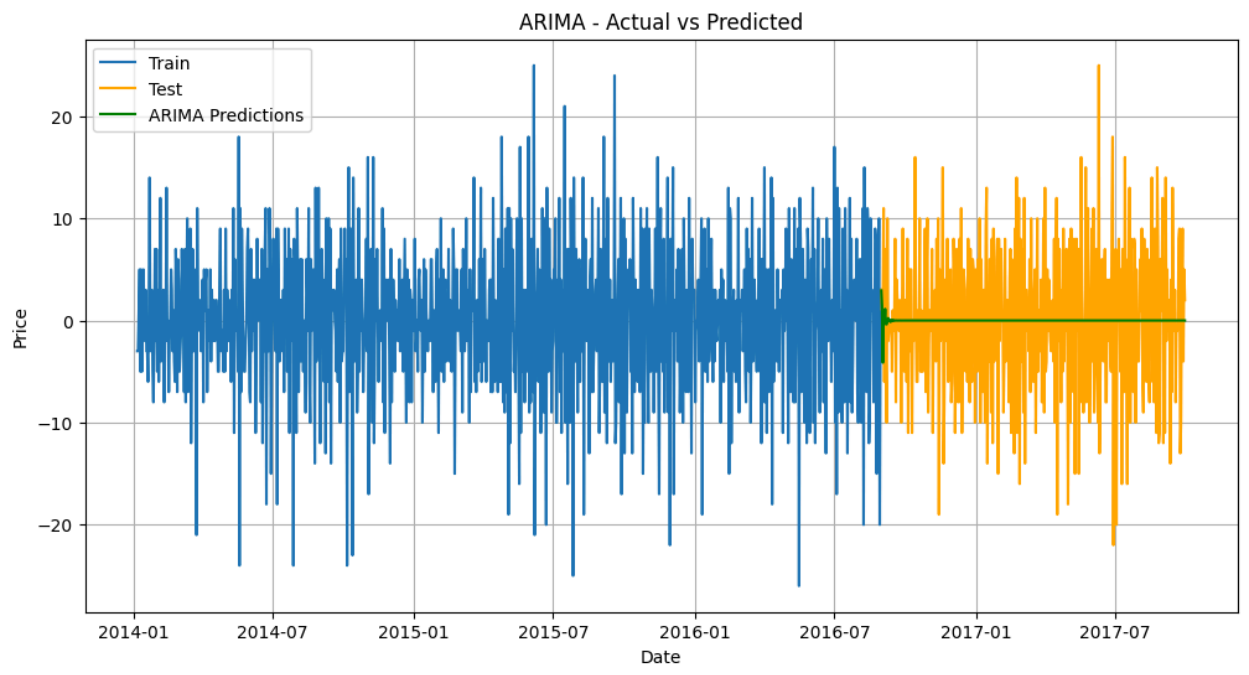
- **Differencing**: If the data is non-stationary, differencing is applied to make it stationary.
    - **First-order differencing**: Subtract the previous value from the current value to remove trends.
    - **Seasonal differencing**: This involves subtracting the value from the same period in the previous cycle (e.g., subtract the value from the same month or season in the previous year).



Model Selection:

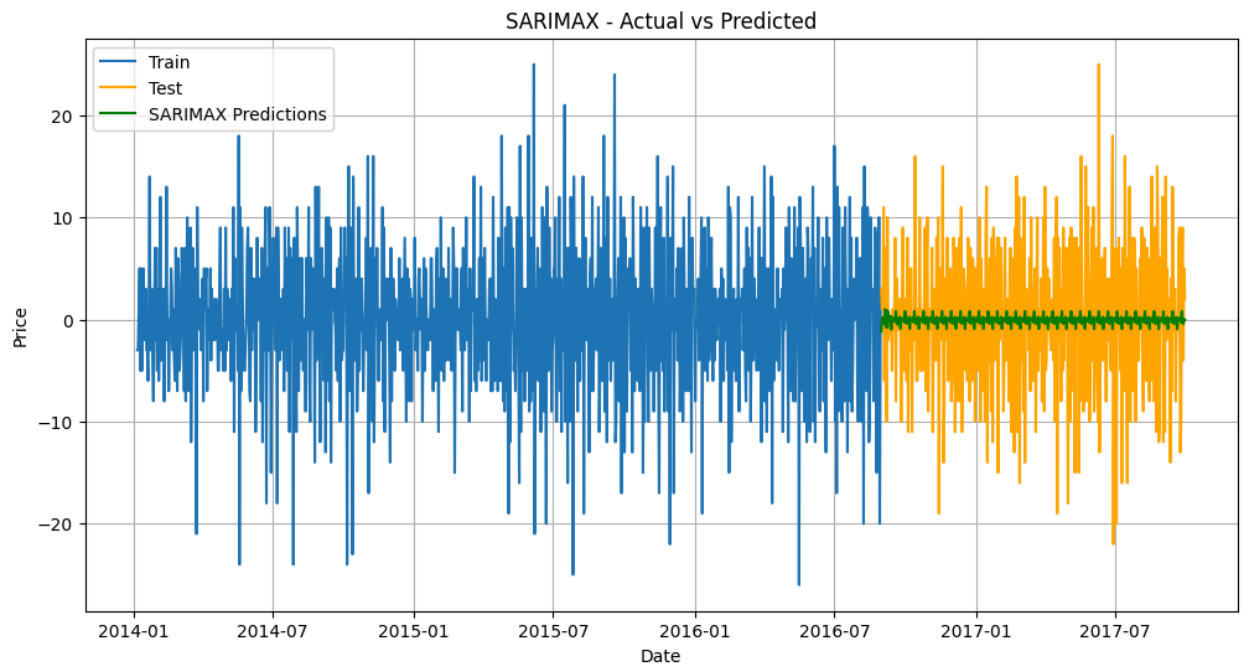
I've used several models for this assignment, each with its strengths in handling different patterns in the data:

1. **ARIMA (AutoRegressive Integrated Moving Average)**:
    - **Components**: AR (AutoRegressive), I (Integrated), and MA (Moving Average) are combined to predict values.
    - **Use**: Best for non-seasonal data with a linear trend.
    - **Pros**: Simple and interpretable; effective for short-term forecasting of stationary data.
    - I have try to implement it but it didn't work

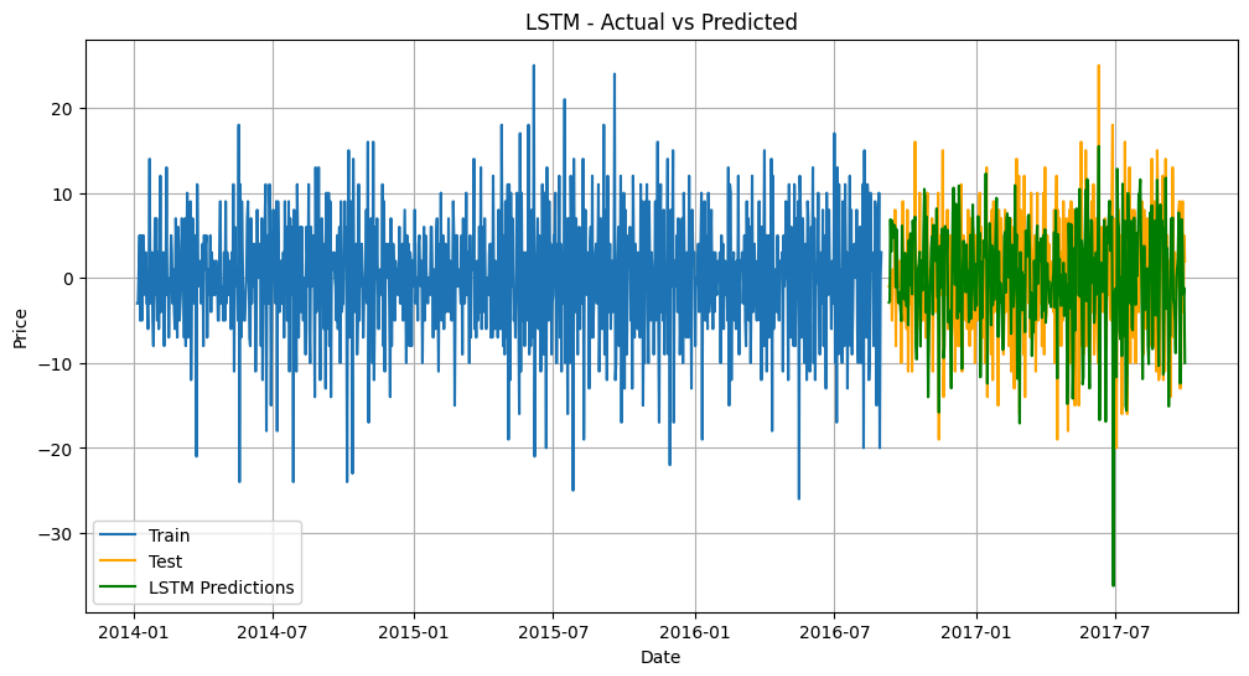


2. **SARIMAX (Seasonal ARIMA with Exogenous Regressors)**:
    - **Enhancement**: Extends ARIMA by adding seasonal components and external regressors.
    - **Use**: Best for seasonal data with trends or additional influencing factors (e.g., economic indicators).

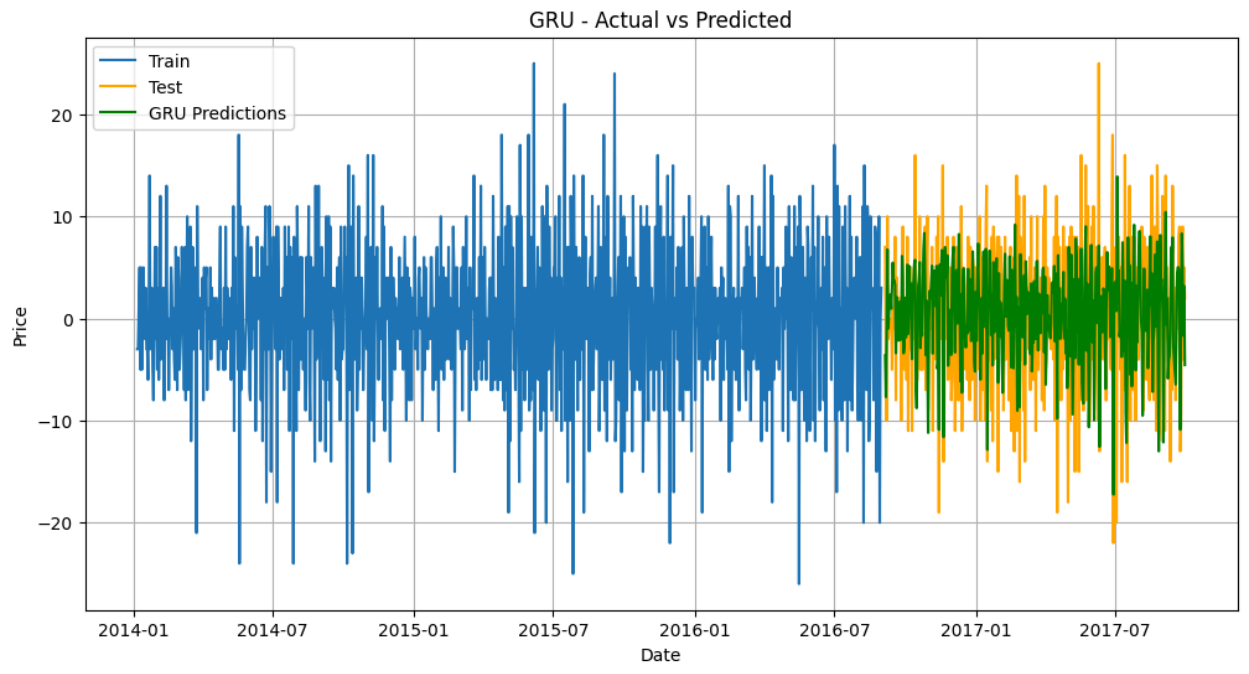
- **Pros**: Handles both trend and seasonality, making it suitable for more complex datasets.
- I have try to implement it but it didn't work



3. **LSTM (Long Short-Term Memory)** and **GRU (Gated Recurrent Unit)**:
   - **Type**: These are deep learning models from the Recurrent Neural Networks (RNN) family.
   - **Use**: Ideal for capturing long-term dependencies and non-linear patterns in sequential data.
   - **Pros**: Well-suited for time series data with complex patterns, trends, and seasonality. LSTM, in particular, handles long-range dependencies better.

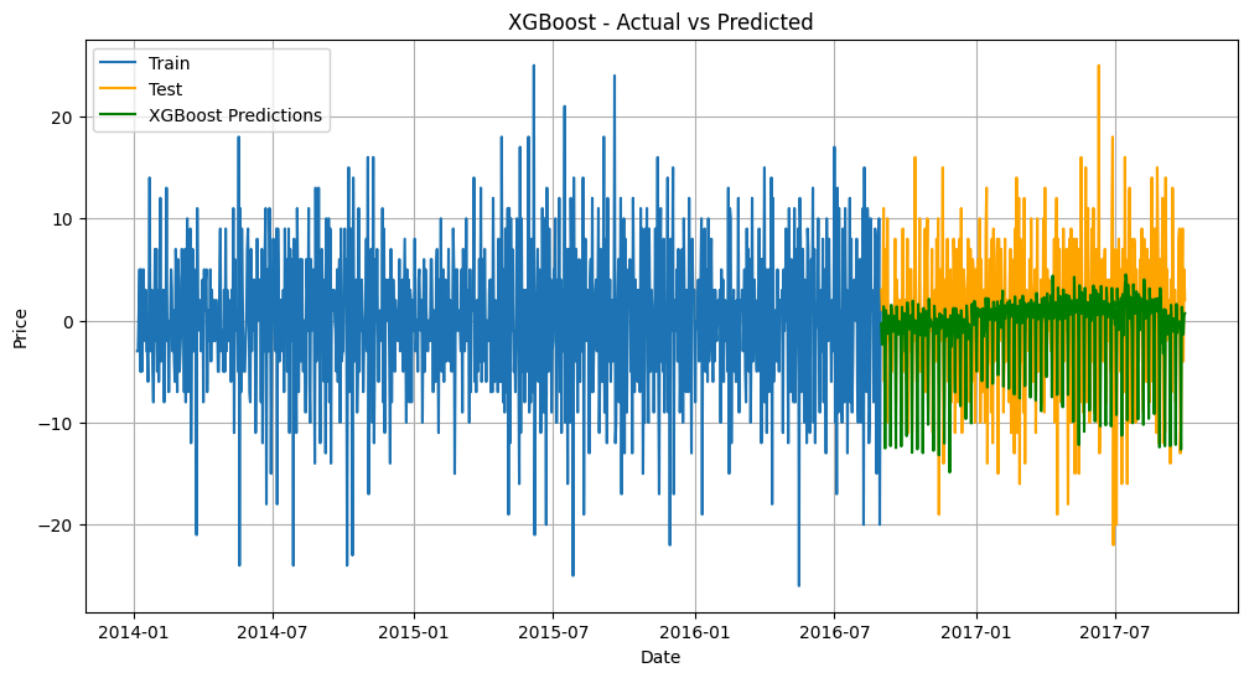


4. **GRU** is a simplified version of LSTM with fewer parameters and faster training, making it suitable for less complex tasks.



5. **XGBoost**:
    - **Type**: A gradient boosting machine learning algorithm.

- **Use**: Often used in regression tasks, including time series forecasting, by treating it as a regression problem.
- **Pros**: High performance, robustness to overfitting, and good at handling various data types.
- **Customization**: Grid Search can be used to tune hyperparameters for optimal performance.



Model Evaluation:

Once the models are trained, they are evaluated based on how well they predict the test data. Evaluation metrics include:

- **MSE (Mean Squared Error)**: Measures the average of the squared differences between the predicted and actual values. Lower values are better.
- **RMSE (Root Mean Squared Error)**: The square root of MSE. It gives a measure of the average magnitude of errors in the same units as the data.
- **MAE (Mean Absolute Error)**: Measures the average of the absolute differences between the predicted and actual values. Again, lower values are better.

```
Model Evaluation Metrics:
ARIMA: {'MSE': 54.46813326374315, 'RMSE': 7.380252926813765, 'MAE': 5.935967905478987}
SARIMAX: {'MSE': 54.58501481041401, 'RMSE': 7.388167215921281, 'MAE': 5.956220963255269}
LSTM: {'MSE': 36.826567175802516, 'RMSE': 6.068489694792479, 'MAE': 4.878295377538651}
GRU: {'MSE': 36.201402302340846, 'RMSE': 6.016760116735655, 'MAE': 4.796634562604148}
XGBoost: {'MSE': 45.7754409069244, 'RMSE': 6.765755013812162, 'MAE': 5.432223346027913}
```

Conclusion:

This project evaluated several time series forecasting models: ARIMA, SARIMAX, LSTM, GRU, and XGBoost, using MSE, RMSE, and MAE as performance metrics.

- **ARIMA** performed poorly with high MSE, RMSE, and MAE, struggling to capture the non-linear patterns in the data.
- **SARIMAX** showed some improvement but still struggled with accuracy, failing to fully capture the data's variability.
- **LSTM** outperformed the other models, capturing non-linear relationships and long-term dependencies with lower error metrics, making it the most effective choice.
- **GRU**, with a simpler architecture, performed similarly to LSTM, offering competitive results and faster training.
- **XGBoost** provided reasonable predictions but underperformed compared to LSTM and GRU, suggesting it may not be optimal for this time series task.

Reference:

Colab notebook: click here