Illinois Institute of Technology

Time Series – Math 546

Final Project

Sagar Katiyar – A20519417

Forecasting Pharmaceutical Sales Data

Introduction

Having the ability to forecast sales data is crucial across almost all sectors of the globe. This, naturally applies to the pharmaceutical industry. Since most medicines have a limited shelf life, being able to forecast specific sales helps in inventory management, resource allocation, financial and production planning and market research in line with the current consumer requirements. The importance is especially emphasized due to the increasing amounts of competitors in the pharmaceutical industry. The current approaches used in the industry are a combination of ARIMA time series, machine learning, statistical models. The objective of this study is to explore some of these methods in order to find some of the best performing models for forecasting pharmaceutical sales data. The dataset used in this study is obtained from Kaggle and it contains the sales information from a pharmacy over six years for eight different types of pharmaceutical products. This implies that we have eight different time series to forecast and draw insights from.

Problem and Dataset Description

As mentioned earlier, the aim of this study is to explore various forecasting techniques to see whether pharmaceutical sales data can be forecasted with increased accuracy.

The dataset is built from the initial dataset consisted of 600000 transactional data collected in 6 years (period 2014-2019), indicating date and time of sale, pharmaceutical drug brand name and sold quantity, exported from Point-of-Sale system in the individual pharmacy. Selected group of drugs from the dataset (57 drugs) is classified to the following Anatomical Therapeutic Chemical (ATC) Classification System categories:

- M01AB - Anti-inflammatory and antirheumatic products, non-steroids, Acetic acid derivatives and related substances
- M01AE - Anti-inflammatory and antirheumatic products, non-steroids, Propionic acid derivatives
- N02BA - Other analgesics and antipyretics, Salicylic acid and derivatives
- N02BE/B - Other analgesics and antipyretics, Pyrazolones and Anilides
- N05B - Psycholeptics drugs, Anxiolytic drugs
- N05C - Psycholeptics drugs, Hypnotics and sedatives drugs
- R03 - Drugs for obstructive airway diseases
- R06-Antihistamines for systemic use
  Sales data are resampled to the hourly, daily, weekly and monthly periods. Data is already pre-processed, where processing included outlier detection and treatment and missing data imputation.

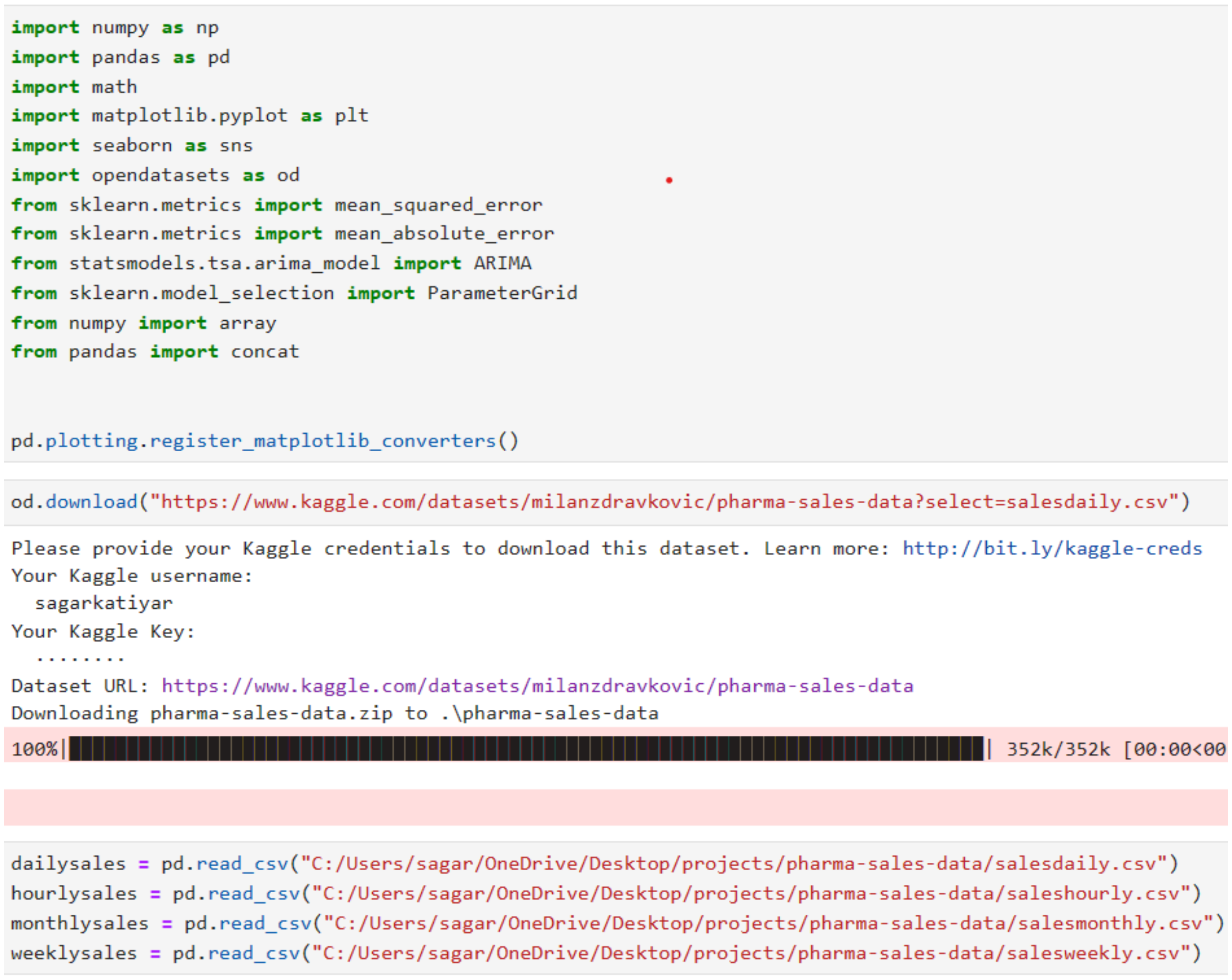
The models used to forecast the pharmaceutical sales data are as follows:

ARIMA: ARIMA (AutoRegressive Integrated Moving Average) is a popular time series forecasting model that combines autoregressive (AR), differencing (I), and moving average (MA) components. It is used to predict future values based on past observations and forecast errors. The model requires the time series data to be stationary, achieved through differencing if necessary. ARIMA is specified by three parameters: p (autoregressive order), d (degree of differencing), and q (moving average order).

LSTM Neural Networks: LSTM (Long Short-Term Memory) Neural Networks are a type of recurrent neural network (RNN) designed to capture long-range dependencies in sequential data. They excel in time series forecasting by learning from past observations and capturing complex patterns. LSTMs use memory cells to store and retrieve information over long periods, making them effective for modeling sequences with temporal dependencies. They are trained using backpropagation through time (BPTT) and can handle input features with varying time steps.

Methodology

Importing Libraries and Datasets



```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns
import opendatasets as od
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from statsmodels.tsa.arima_model import ARIMA
from sklearn.model_selection import ParameterGrid
from numpy import array
from pandas import concat


pd.plotting.register_matplotlib_converters()
```

```
od.download("https://www.kaggle.com/datasets/milanzdravkovic/pharma-sales-data?select=salesdaily.csv")
```

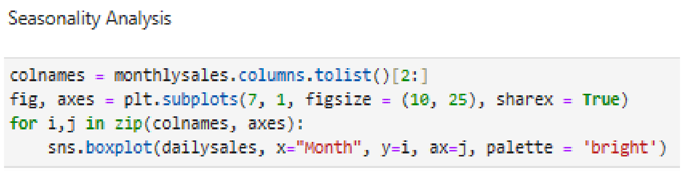
```
Please provide your Kaggle credentials to download this dataset. Learn more: http://bit.ly/kaggle-creds
Your Kaggle username:
  sagarkatiyar
Your Kaggle Key:
  ········
Dataset URL: https://www.kaggle.com/datasets/milanzdravkovic/pharma-sales-data
Downloading pharma-sales-data.zip to .\pharma-sales-data
100%|██████████████████████████████████████████████████| 352k/352k [00:00<00
```

```
dailysales = pd.read_csv("C:/Users/sagar/OneDrive/Desktop/projects/pharma-sales-data/salesdaily.csv")
hourlysales = pd.read_csv("C:/Users/sagar/OneDrive/Desktop/projects/pharma-sales-data/saleshourly.csv")
monthlysales = pd.read_csv("C:/Users/sagar/OneDrive/Desktop/projects/pharma-sales-data/salesmonthly.csv")
weeklysales = pd.read_csv("C:/Users/sagar/OneDrive/Desktop/projects/pharma-sales-data/salesweekly.csv")
```

Above are the libraries used for data importing processing, time series analysis and performance metrics. The daily, hourly and weekly sales data are stored in their corresponding dataframes.

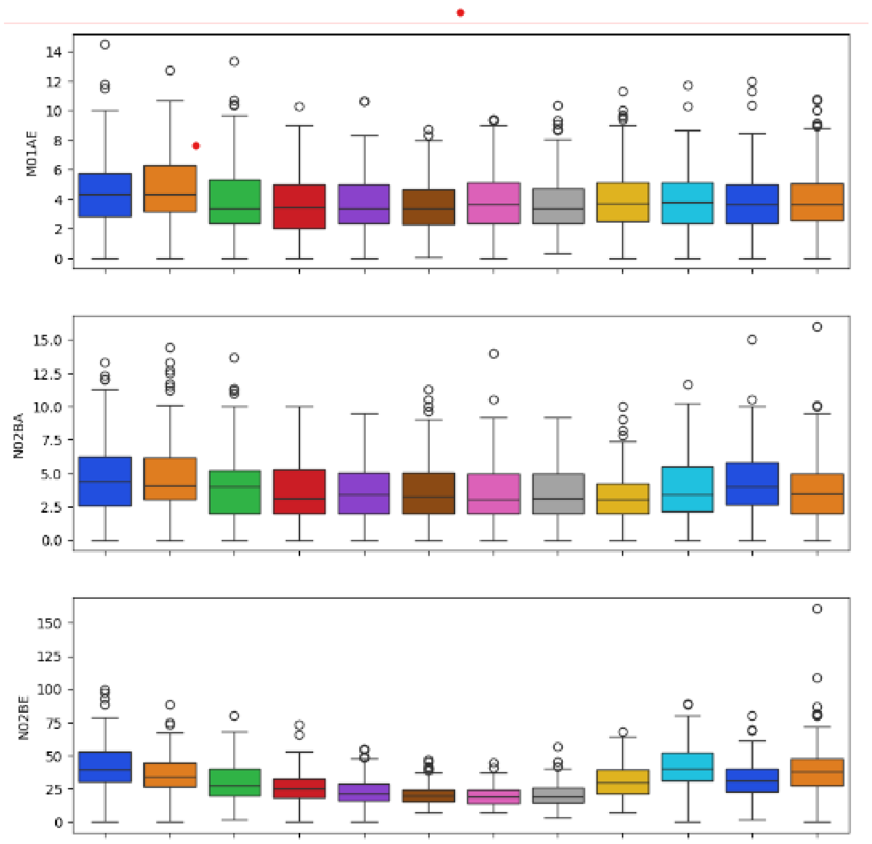
Seasonality Analysis

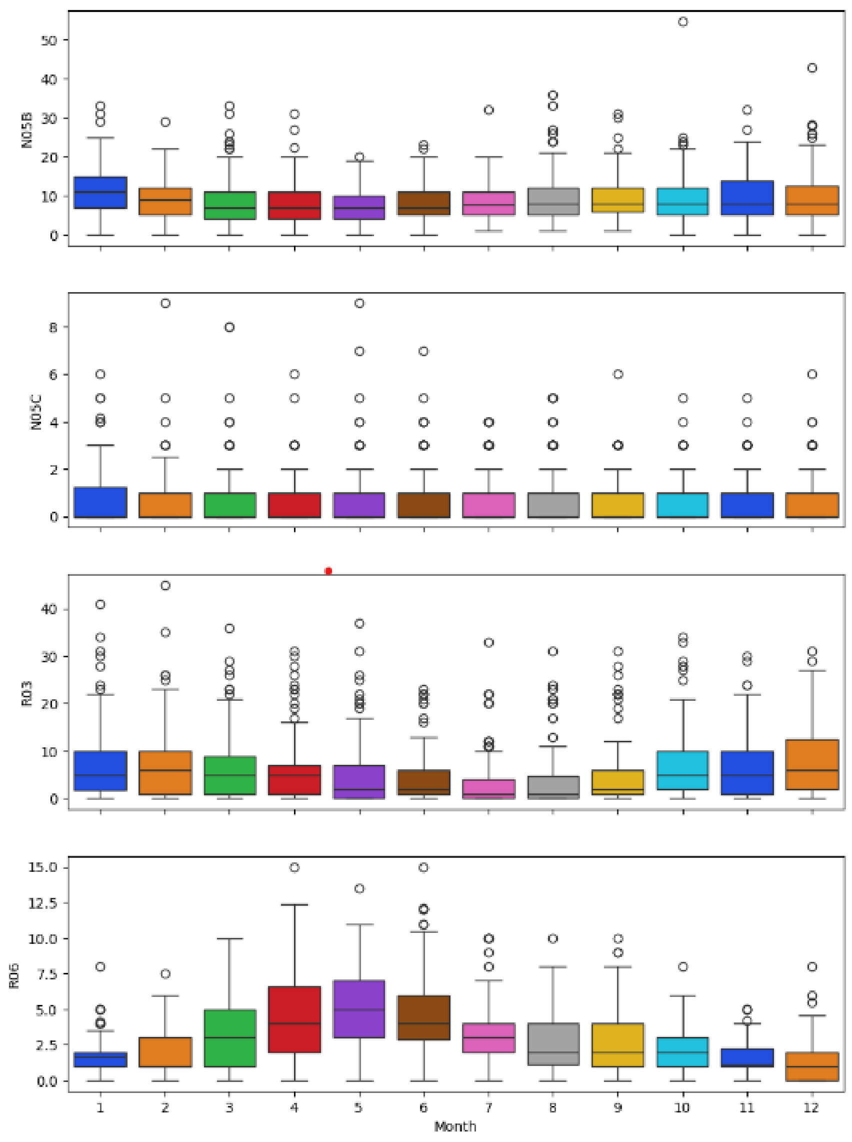
To analyze the seasonality of the data a boxplot visualization and rolling window visualization of the dataset was done.



Seasonality Analysis

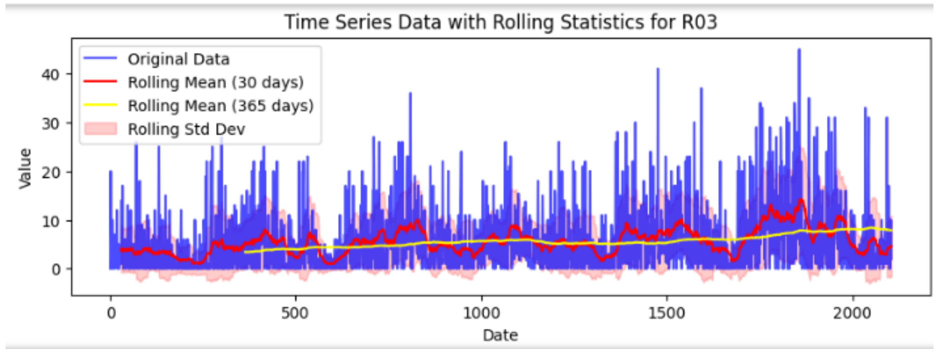
```
colnames = monthlysales.columns.tolist()[2:]
fig, axes = plt.subplots(7, 1, figsize = (10, 25), sharex = True)
for i,j in zip(colnames, axes):
    sns.boxplot(dailysales, x="Month", y=i, ax=j, palette = 'bright')
```

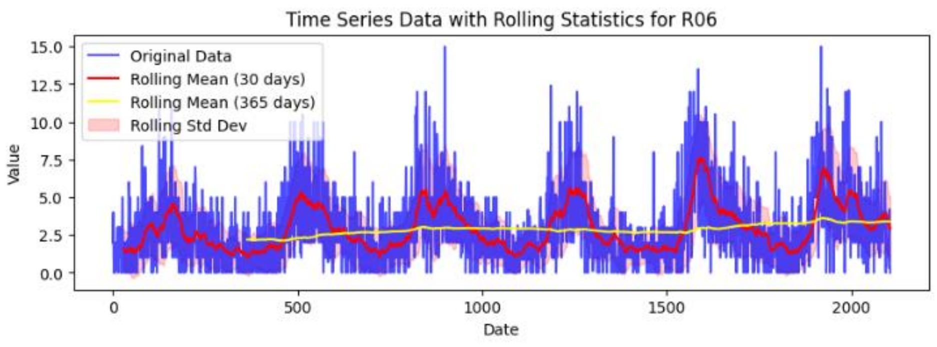


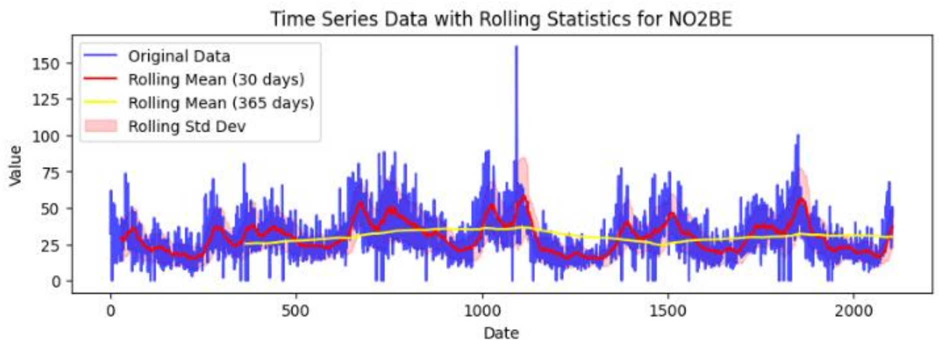


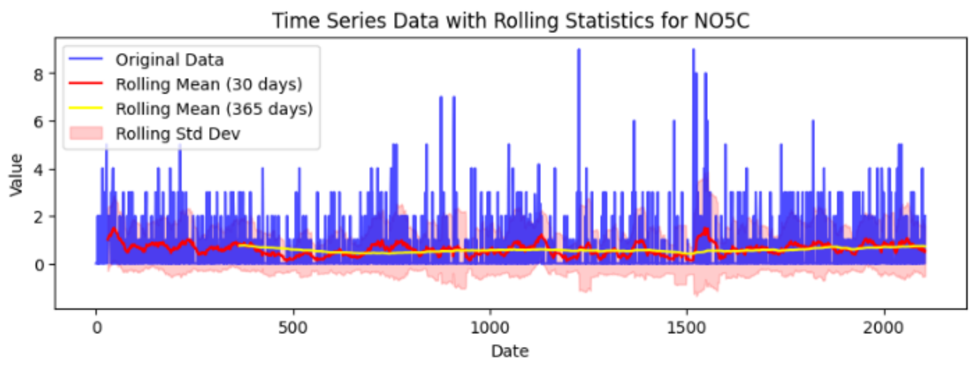
The boxplot visualization suggests the clear presence of seasonality in N02BE, R03 and R06.

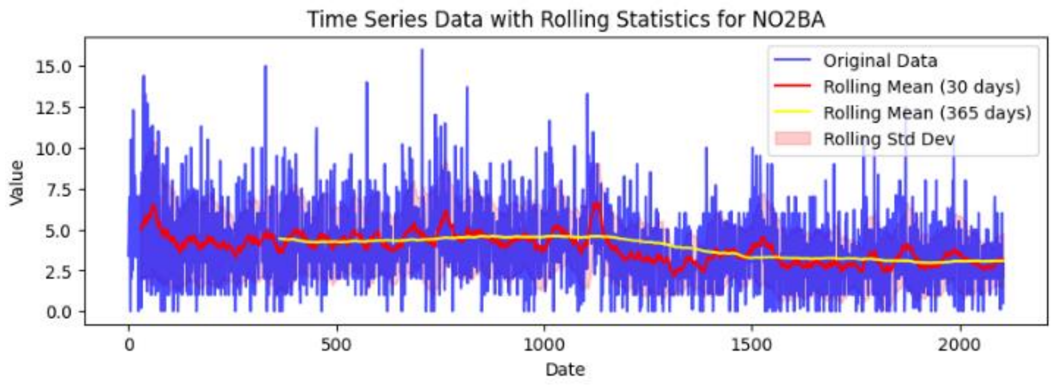
The next approach used to investigate seasonality was via the use of rolling means which essentially smooths out the fluctuations in the time series, revealing the underlying trends and patterns, if they exist, in the data.

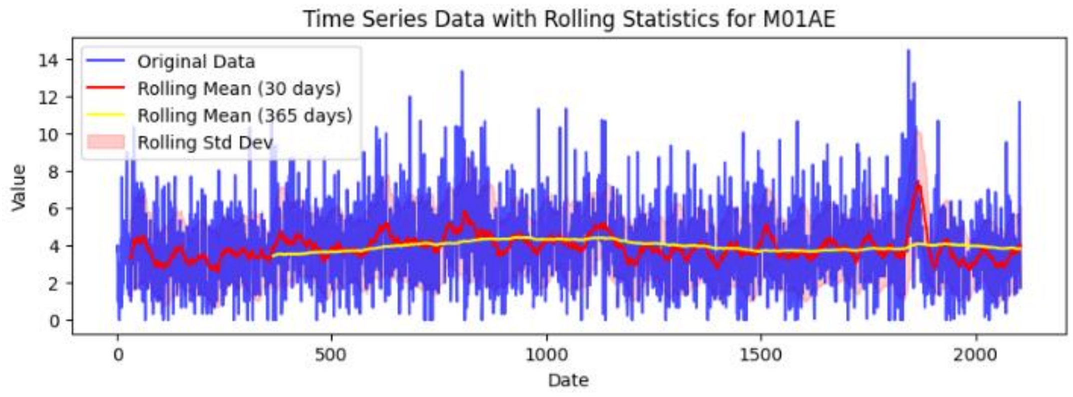


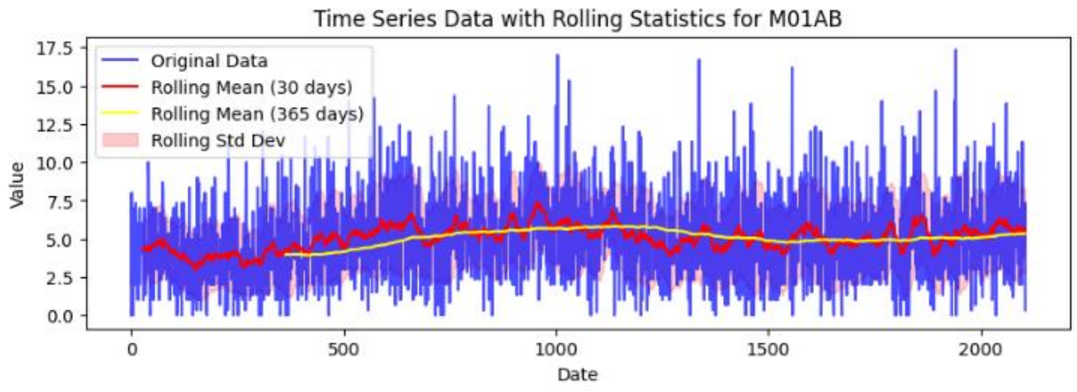


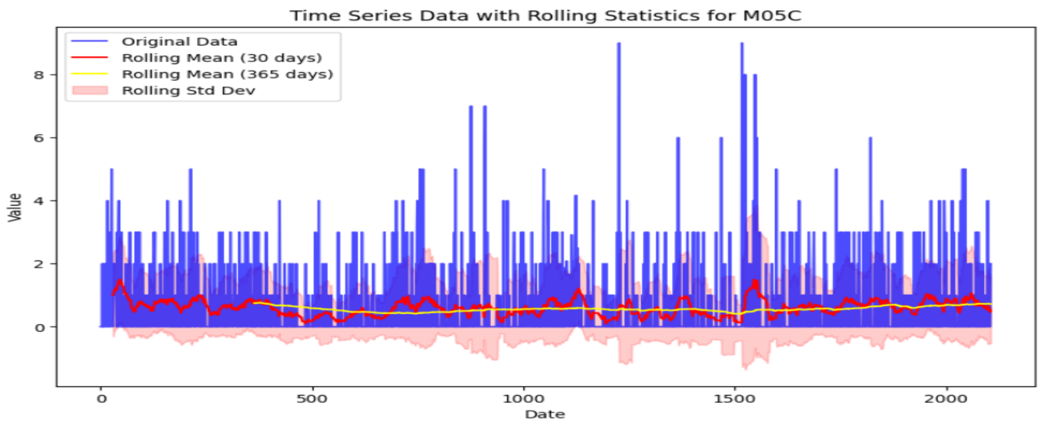





Time Series Data with Rolling Statistics for NO2BA
Original Data
Rolling Mean (30 days)
Rolling Mean (365 days)
Rolling Std Dev
Value
Date


Time Series Data with Rolling Statistics for M01AE
Original Data
Rolling Mean (30 days)
Rolling Mean (365 days)
Rolling Std Dev
Value
Date

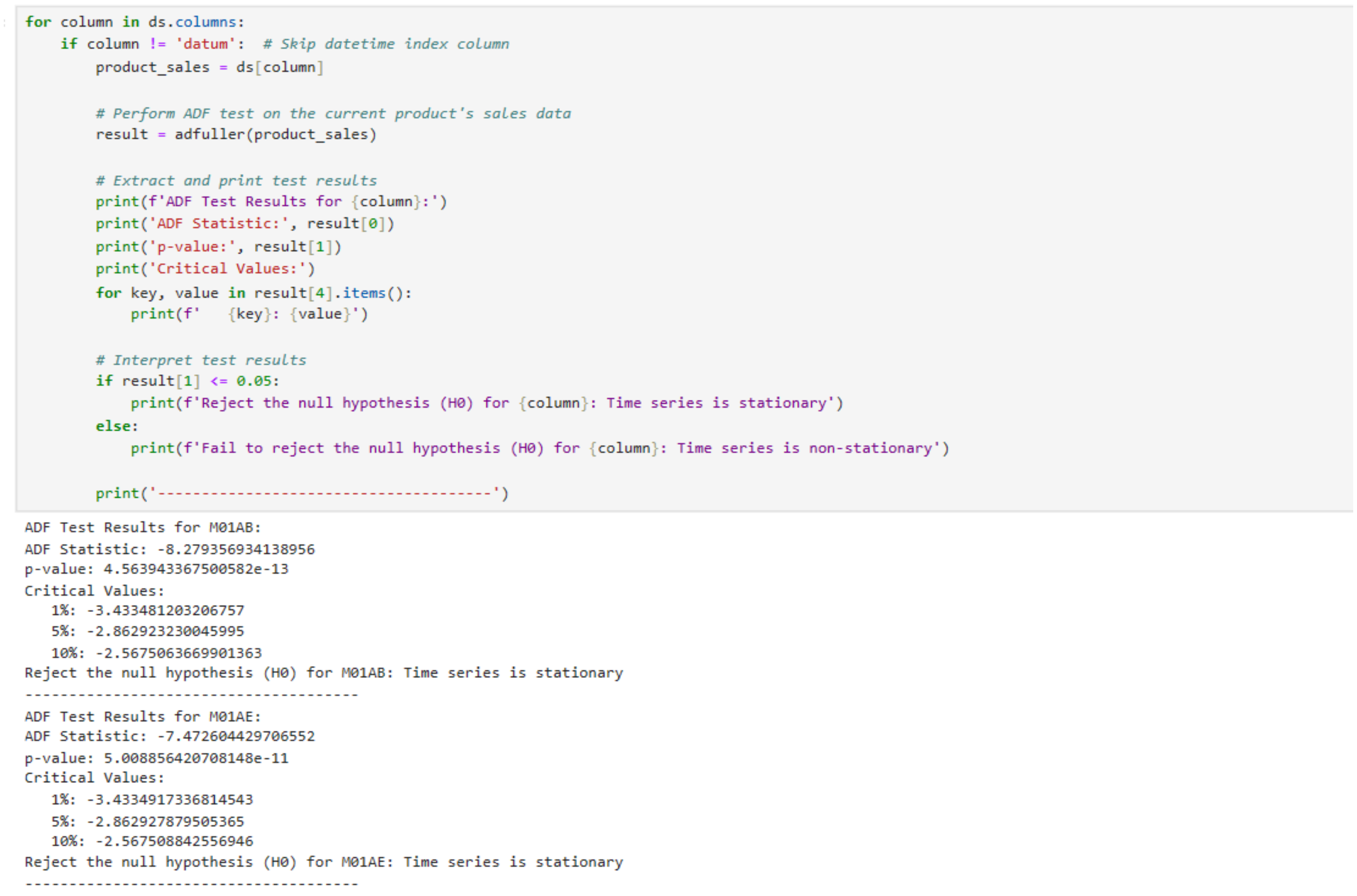

Time Series Data with Rolling Statistics for M01AB
Original Data
Rolling Mean (30 days)
Rolling Mean (365 days)
Rolling Std Dev
Value
Date


Time Series Data with Rolling Statistics for M05C
Original Data
Rolling Mean (30 days)
Rolling Mean (365 days)
Rolling Std Dev
Value
Date

Through the visualizations above we can even more clearly see the seasonal natures in some of the drug categories, especially in N02BE, R03 and R06 also displayed by the boxplot above.

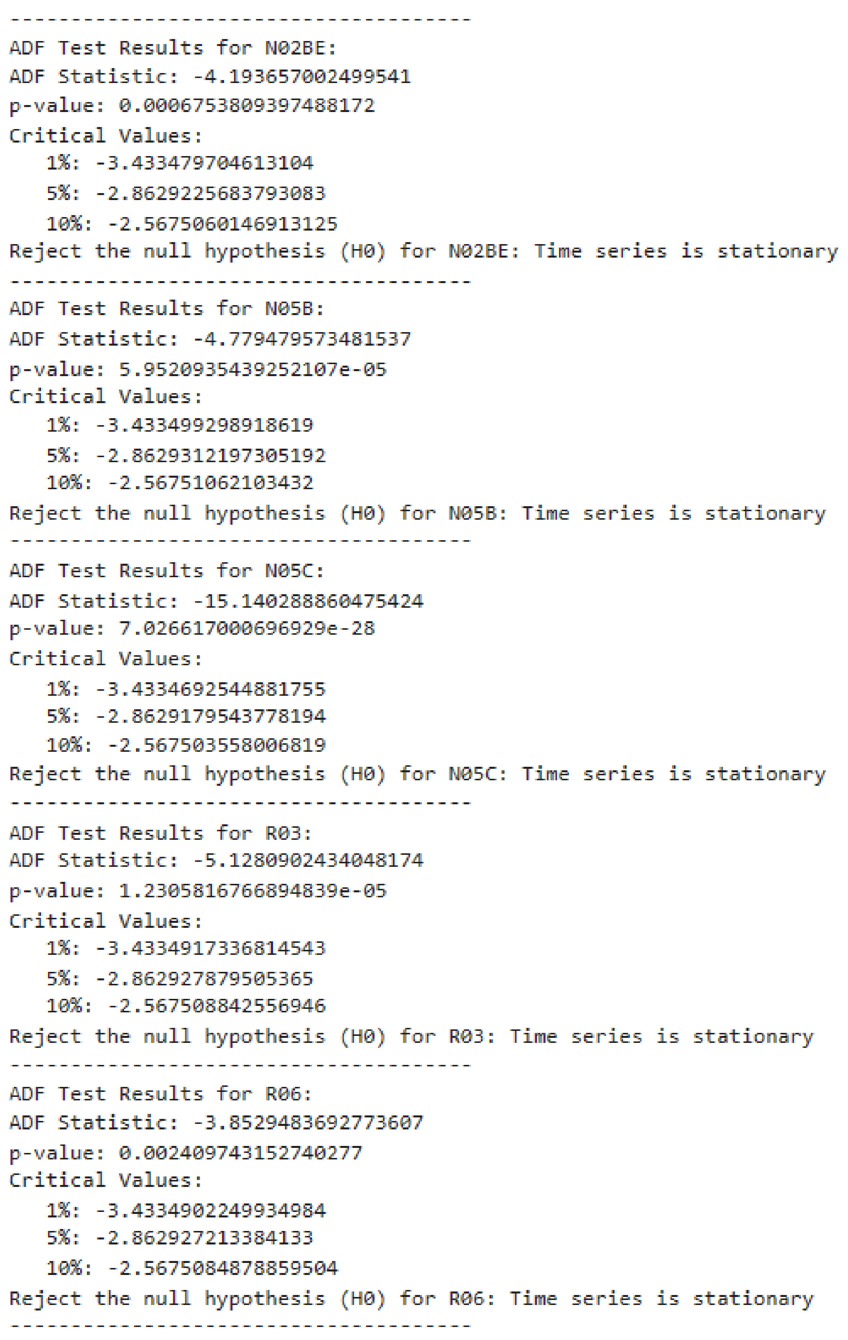
Stationarity Analysis

An Augmented Dickey Fuller (ADF) test was conducted for each column in order to detect stationarity. A time series being stationary implies that its mean, variance and autocorrelation are constant over time.



```python
for column in ds.columns:
    if column != 'datum':  # Skip datetime index column
        product_sales = ds[column]

        # Perform ADF test on the current product's sales data
        result = adfuller(product_sales)

        # Extract and print test results
        print(f'ADF Test Results for {column}:')
        print('ADF Statistic:', result[0])
        print('p-value:', result[1])
        print('Critical Values:')
        for key, value in result[4].items():
            print(f'   {key}: {value}')

        # Interpret test results
        if result[1] <= 0.05:
            print(f'Reject the null hypothesis (H0) for {column}: Time series is stationary')
        else:
            print(f'Fail to reject the null hypothesis (H0) for {column}: Time series is non-stationary')

        print('--------------------------------------')
```

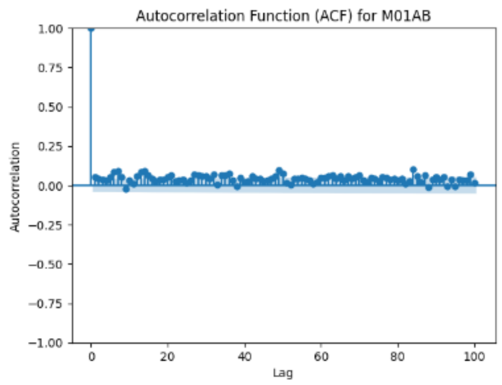
```
ADF Test Results for M01AB:
ADF Statistic: -8.279356934138956
p-value: 4.563943367500582e-13
Critical Values:
   1%: -3.433481203206757
   5%: -2.862923230045995
   10%: -2.5675063669901363
Reject the null hypothesis (H0) for M01AB: Time series is stationary
--------------------------------------
ADF Test Results for M01AE:
ADF Statistic: -7.472604429706552
p-value: 5.008856420708148e-11
Critical Values:
   1%: -3.4334917336814543
   5%: -2.862927879505365
   10%: -2.567508842556946
Reject the null hypothesis (H0) for M01AE: Time series is stationary
--------------------------------------
```

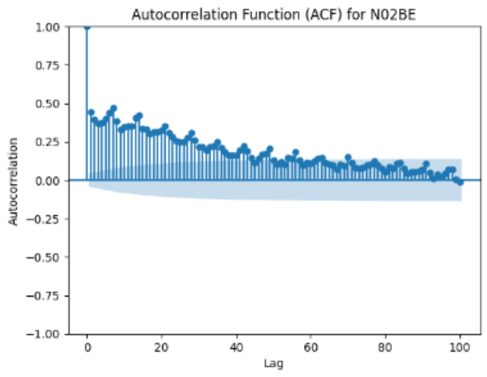


```
--------------------------------------
ADF Test Results for N02BE:
ADF Statistic: -4.193657002499541
p-value: 0.0006753809397488172
Critical Values:
   1%: -3.433479704613104
   5%: -2.8629225683793083
   10%: -2.5675060146913125
Reject the null hypothesis (H0) for N02BE: Time series is stationary
--------------------------------------
ADF Test Results for N05B:
ADF Statistic: -4.779479573481537
p-value: 5.9520935439252107e-05
Critical Values:
   1%: -3.433499298918619
   5%: -2.8629312197305192
   10%: -2.56751062103432
Reject the null hypothesis (H0) for N05B: Time series is stationary
--------------------------------------
ADF Test Results for N05C:
ADF Statistic: -15.140288860475424
p-value: 7.026617000696929e-28
Critical Values:
   1%: -3.4334692544881755
   5%: -2.8629179543778194
   10%: -2.567503558006819
Reject the null hypothesis (H0) for N05C: Time series is stationary
--------------------------------------
ADF Test Results for R03:
ADF Statistic: -5.1280902434048174
p-value: 1.2305816766894839e-05
Critical Values:
   1%: -3.4334917336814543
   5%: -2.862927879505365
   10%: -2.567508842556946
Reject the null hypothesis (H0) for R03: Time series is stationary
--------------------------------------
ADF Test Results for R06:
ADF Statistic: -3.8529483692773607
p-value: 0.002409743152740277
Critical Values:
   1%: -3.4334902249934984
   5%: -2.862927213384133
   10%: -2.5675084878859504
Reject the null hypothesis (H0) for R06: Time series is stationary
--------------------------------------
```

As can be seen from the ADF tests, all of the series were stationary, which makes it more viable for modelling the time series.

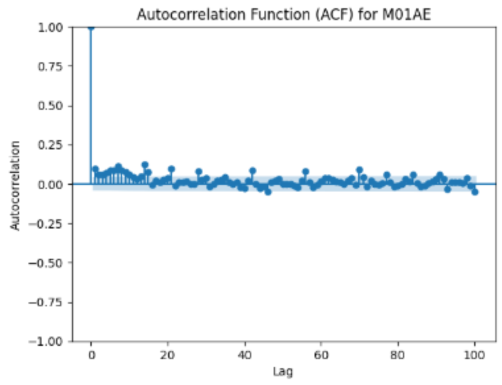
Auto-Correlation Analysis

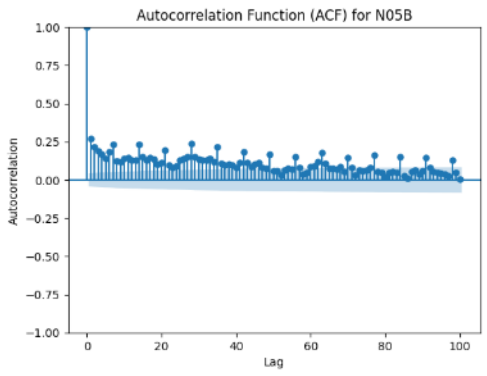




```
<Figure size 1200x600 with 0 Axes>
```

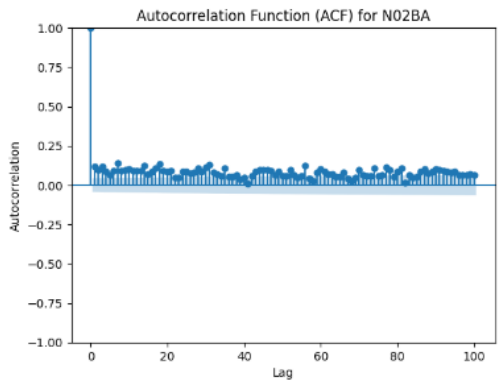
```
<Figure size 1200x600 with 0 Axes>
```

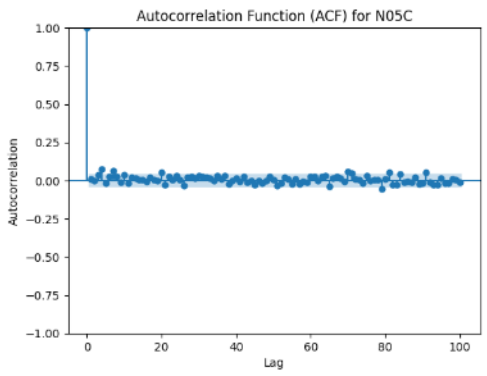


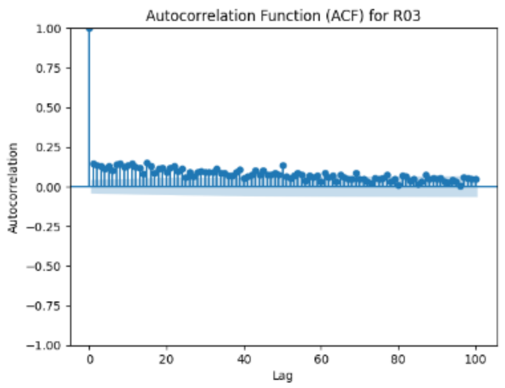


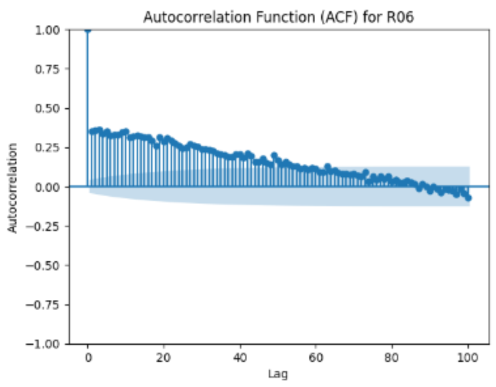
```
<Figure size 1200x600 with 0 Axes>
```

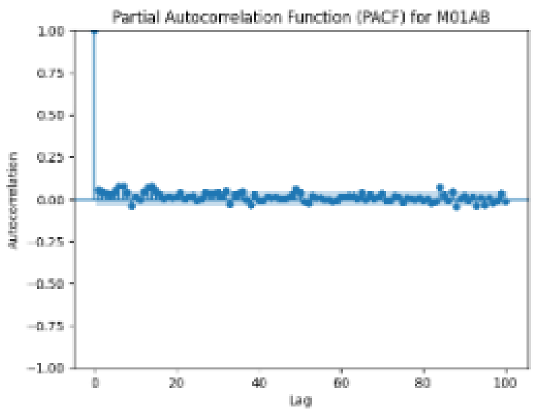
```
<Figure size 1200x600 with 0 Axes>
```



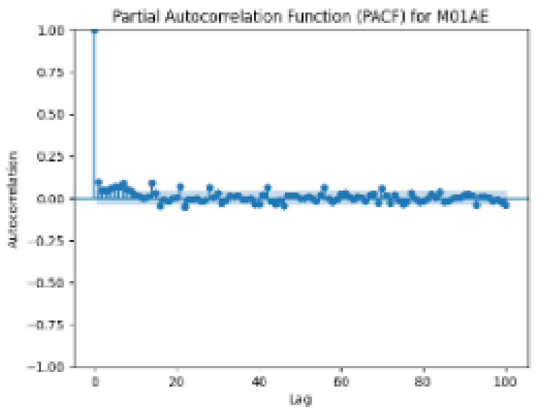




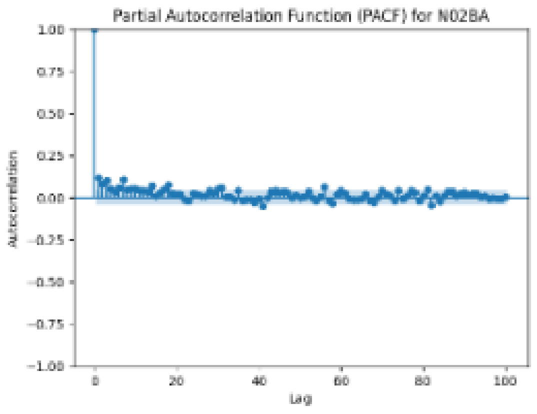



Partial Autocorrelation Function (PACF) for M01AB
Autocorrelation
Lag

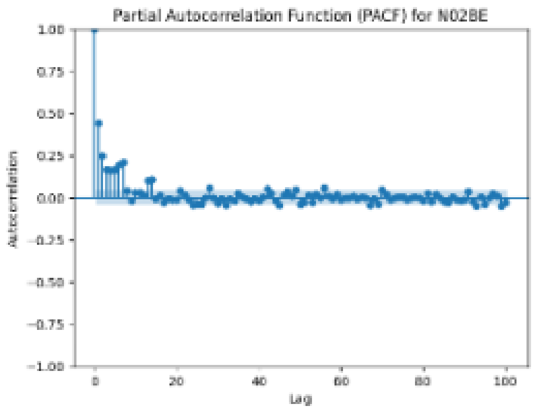
```
<Figure size 1200x600 with 0 Axes>
```


Partial Autocorrelation Function (PACF) for M01AE
Autocorrelation
Lag

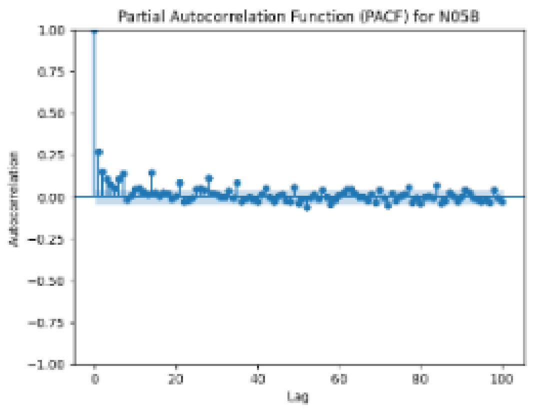
```
<Figure size 1200x600 with 0 Axes>
```


Partial Autocorrelation Function (PACF) for N02BA
Autocorrelation
Lag

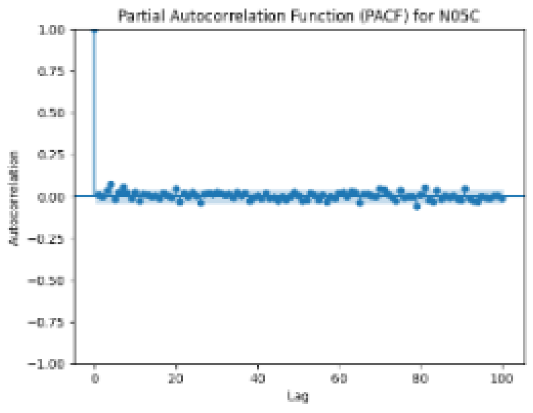
```
<Figure size 1200x600 with 0 Axes>
```


Partial Autocorrelation Function (PACF) for N02BE
Autocorrelation
Lag

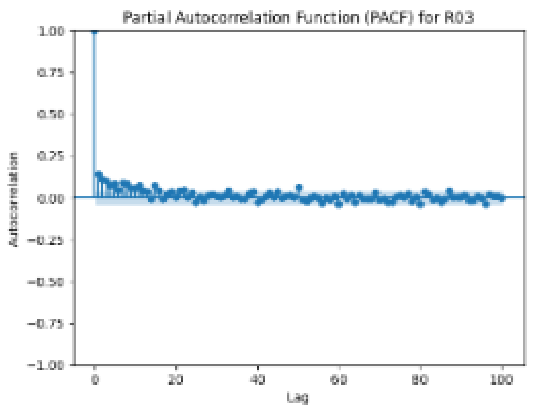
```
<Figure size 1200x600 with 0 Axes>
```


Partial Autocorrelation Function (PACF) for N05B
Autocorrelation
Lag

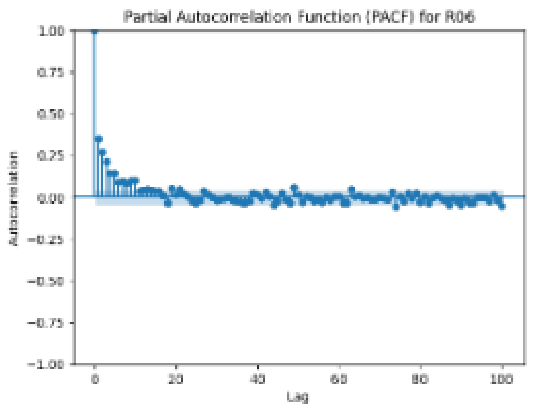
```
<Figure size 1200x600 with 0 Axes>
```


Partial Autocorrelation Function (PACF) for N05C
Autocorrelation
Lag

```
<Figure size 1200x600 with 0 Axes>
```


Partial Autocorrelation Function (PACF) for R03
Autocorrelation
Lag

```
<Figure size 1200x600 with 0 Axes>
```


Partial Autocorrelation Function (PACF) for R06
Autocorrelation
Lag

Based on the ACF and PACF plots, most of the series have weak autocorrelation. However for N02BE and R06, there appears to be a significant decrease in spikes with increasing lag, providing strong evidence for the presence of seasonality in the data.

ARIMA Forecasting

The ARIMA method was first used to do forecasting of the weekly data after determining the optimal parameters that gave the lowest training Mean Squared Error (MSE).

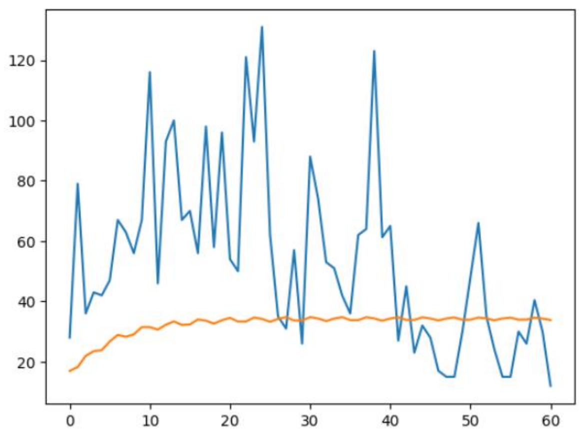
```python
def eval_ar_model(data, train_size=0.8, order=(1, 0, 1), test_size=None):
    """
    Split the time series data into training and test sets, fit an ARIMA model on the training data,
    make predictions on the test data, and calculate the Mean Squared Error (MSE).

    Parameters:
    - data (array-like): The time series data.
    - train_size (float): The proportion of data to use for training (default: 0.8).
    - order (tuple): The (p, d, q) order of the ARIMA model (default: (1, 0, 1)).
    - test_size (int or None): The number of data points to use for testing.
                               If None, use the remainder of the data after training size.

    Returns:
    - float: The Mean Squared Error (MSE) between predicted and actual values.
    """

    # Determine the split index based on train_size
    split_index = int(len(data) * train_size)

    # Split the data into training and test sets
    train_data = data[:split_index]
    if test_size is None:
        test_data = data[split_index:]
    else:
        test_data = data[split_index:split_index + test_size]

    # Fit ARIMA model on the training data
    model = ARIMA(train_data, order=order)
    fitted_model = model.fit()

    # Make predictions on the test data
    predictions = fitted_model.forecast(steps=len(test_data))

    # Calculate Mean Squared Error (MSE)
    mse = mean_squared_error(test_data, predictions)

    return mse
```

```python
def eval_model(colname, data, p_vals, d_vals, q_vals):
    best_mse, best_par = float("inf"), None
    for p in p_vals:
        for d in d_vals:
            for q in q_vals:
                o = (p, d, q)
                try:
                    mse = eval_ar_model(data, order=o)
                    if mse < best_mse:
                        best_mse, best_par = mse, o
                except:
                    continue
    print(colname+'-Best ARIMA params are %s with a MSE of %.3f' % (best_par, best_mse))
```

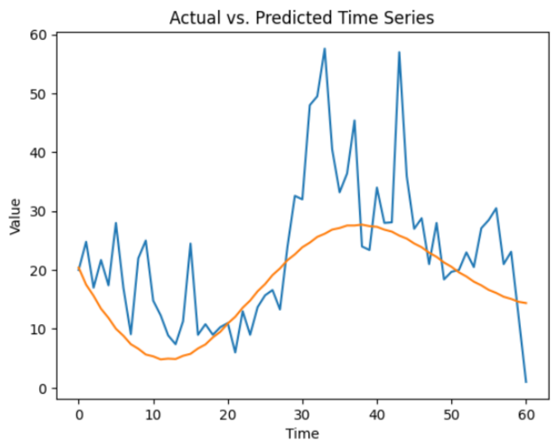
```python
for colname in colnames:
    eval_model(colname, weeklysales[colname].values, range(0,6), range(0,6), range(0,6))
```

```
M01AB-Best ARIMA params are (1, 2, 1) with a MSE of 60.322
M01AE-Best ARIMA params are (3, 4, 5) with a MSE of 83.407
N02BA-Best ARIMA params are (4, 2, 2) with a MSE of 28.940
N02BE-Best ARIMA params are (0, 0, 2) with a MSE of 6111.095
N05B-Best ARIMA params are (4, 1, 1) with a MSE of 146.322
N05C-Best ARIMA params are (1, 4, 3) with a MSE of 7.040
R03-Best ARIMA params are (0, 0, 1) with a MSE of 1208.085
R06-Best ARIMA params are (3, 0, 5) with a MSE of 110.513
```

```python
plot_ar_model(weeklysales['R03'].values, order = R06ops)
```

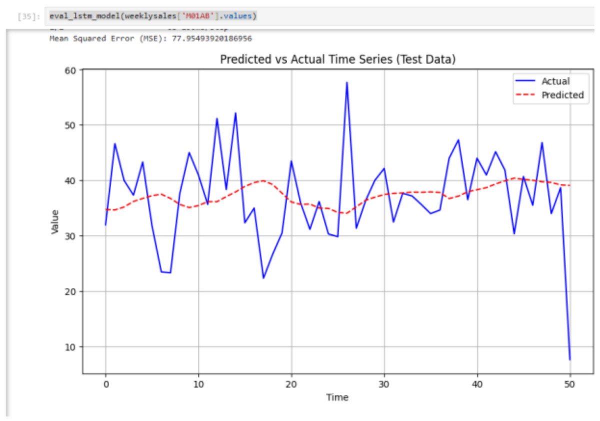


```python
plot_ar_model(weeklysales['R06'].values, order = R06ops)
```

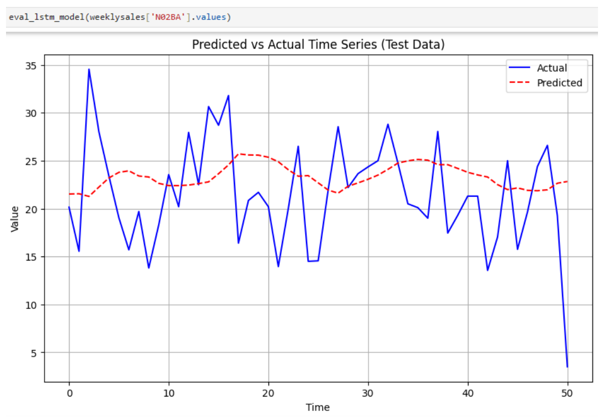


LSTM Forecasting

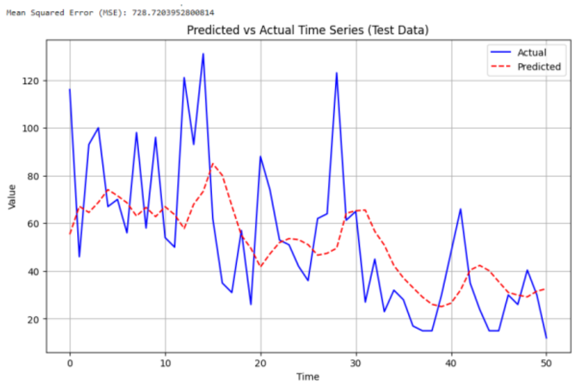
Next, an LSTM Neural Networks model was used for weekly forecasting. The data was normalized, trained over 50 epochs with a sequential model, an Adam optimizer and a sequence length of 10. After training the model, the MSE and plots of the predicted vs actual sales for the test data was displayed. Below is a subset of the plots obtained (from left to right: M01AB, N02BA, R03, R06)



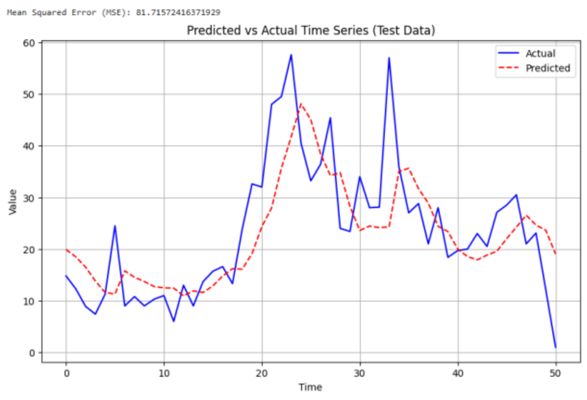
```
[35]: eval_lstm_model(weeklysales['M01AB'].values)
Mean Squared Error (MSE): 77.95493920186956
```



```
eval_lstm_model(weeklysales['N02BA'].values)
```



```
Mean Squared Error (MSE): 728.7203952800814
```



```
Mean Squared Error (MSE): 81.71572416371929
```

Model Evaluation

As can be seen through the acquired MSE's and plots of the testing data and the predicted data we got for our models, the LSTM model is more accurate than the ARIMA model for the seasonal categories of N02BE, R03 and R06 that were initially identified during the exploratory data analysis, both in terms of having a lower test MSE and more closely following the actual trends of the time series depicted by the plots. On the other hand, the ARIMA model performed better in terms of having lower MSE's for the rest of the time series.

The MSE's for all the different time series for both models are displayed below.

	M01AB	M01AE	N02BA	N02BE	N05B	N05C	R03	R06
ARIMA	60.32	83.41	28.94	6111.09	146.32	7.04	1208.09	110.51
LSTM	77.95	94.44	37.80	3372.19	142.72	7.73	728.72	81.72

With this, it can be evaluated that both the ARIMA models and the LSTM models have their own merits when it comes to dealing with specific types of data. LSTM models perform better when the data is seasonal in nature, whereas ARIMA performs better when it is not.

Conclusion

Both the ARIMA and the LSTM models proved useful in forecasting the pharmaceutical data and can be aimed to be generalized towards more pharmacies, thereby expanding the scope of the study and investigating these models on a larger scale. The LSTM models appear to be better for forecasting when it comes to seasonal data whereas ARIMA appears to be better for non seasonal data. However, there were a few caveats that need to be considered before making any final conclusions. Firstly, the SARIMA model could have been used instead of the ARIMA model and might have had a better performance since it is especially designed for seasonal data. Secondly, the LSTM model was also did not have their parameters optimized as were there for the ARIMA model and might also have boosted the performance on the model. In future studies, these could be accounted for in order to have more conclusive studies.

Furthermore, more models like different types of forecasting models and the inclusion of explanatory variables in the model could also be included in future studies to have more accurate forecasting.

References

Pharma sales data. (2020, January 5). Kaggle. https://www.kaggle.com/datasets/milanzdravkovic/pharma-sales-data

News-Medical. (2023, July 10). *What is pharmaceutical forecasting?* https://www.news-medical.net/news/20230113/What-is-pharmaceutical-forecasting.aspx

Husein, A. M., Arsyal, M., Sinaga, S., & Syahputa, H. (2019). Generative adversarial networks time series models to forecast medicine daily sales in hospital. *Sinkron*, *3*(2), 112. https://doi.org/10.33395/sinkron.v3i2.10044

Galkin, D., Dudkina, T. A., & Mamedova, N. (2022). Forecasting time series using neural networks on the example of primary sales of a pharmaceutical company. *SHS Web of Conferences*, *141*, 01014. https://doi.org/10.1051/shsconf/202214101014

Drug Sales Prediction with ACF and PACF Supported ARIMA Method. (2020, September 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9219448/references#references