Forecasting COVID-19 Cases Using Time Series Models: A Study Across Countries and WHO Regions

Project Report

Table of Contents

1. Introduction and Objective

The COVID-19 pandemic has presented an unprecedented global health challenge, with the number of cases varying significantly across different regions. The ability to forecast future case counts is crucial for governments and health organizations to enable effective resource planning, policy-making, and public health response.

The primary objective of this project is to build and evaluate time series models to forecast the number of confirmed COVID-19 cases. This study focuses on two prominent time series models, **ARIMA** and **SARIMAX**, to analyze and predict trends in daily new cases at both a global level and for a specific country, India.

2. Data Exploration and Preprocessing

The analysis was performed on a dataset containing daily COVID-19 case counts. The initial steps involved data cleaning, preprocessing, and exploratory analysis to identify underlying patterns.

Key Observations:

- **Trends:** The initial visualizations of confirmed cases over time revealed clear upward trends, indicating periods of exponential growth in infections. Moving averages (7-day, 30-day, etc.) were used to smooth out daily noise and highlight these long-term trends, showing the distinct waves of the pandemic.
- **Seasonality:** A strong **weekly seasonal pattern** was identified. Seasonal decomposition and box plots showed that the number of reported new cases often followed a 7-day cycle, with noticeable dips typically occurring over weekends. This is a common pattern in public health data, often attributed to reporting delays.
- **Stationarity:** The Augmented Dickey-Fuller (ADF) test was conducted on the time series of daily new cases. The test results (p-value > 0.05) indicated that the data was **non-stationary**, meaning its statistical properties, like the mean, changed over time. This confirmed the need for differencing (the 'I' component in ARIMA) to stabilize the data before modeling.

3. Methodology

Two different time series models were built to forecast future case counts. The data was split into a training set (90%) and a testing set (10%) to evaluate model performance.

- **ARIMA (Autoregressive Integrated Moving Average):** A standard statistical model for time series forecasting. It combines three components:
    - **Autoregressive (AR):** Uses the relationship between an observation and its past values.
    - **Integrated (I):** Uses differencing to make the time series stationary.
    - **Moving Average (MA):** Uses the dependency between an observation and past forecast errors. The model order was chosen as **(p=7, d=1, q=1)** based on ACF/PACF plots and the stationarity test.
- **SARIMAX (Seasonal ARIMA with eXogenous variables):** An extension of ARIMA that includes capabilities to model seasonality. Given the strong weekly pattern observed, this model was expected to perform better. The model order was **(p=6, d=1, q=1)** with a seasonal order of **(P=1, D=1, Q=1, s=7)** to capture the 7-day cycle.
- **Evaluation Metrics:** The models were evaluated using:
    - **RMSE (Root Mean Squared Error):** Measures the average magnitude of the forecast errors. Lower is better.
    - **MAPE (Mean Absolute Percentage Error):** Measures the average percentage error. Lower is better.

4. Source Code

The following Python script was used to perform the data analysis, modeling, and visualization.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
```

```python
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.api import ARIMA, SARIMAX
from sklearn.metrics import mean_squared_error
from prettytable import PrettyTable
import seaborn as sns
import warnings

warnings.filterwarnings("ignore")

def perform_adf_test(series, series_name):
    result = adfuller(series.dropna())

    table = PrettyTable()
    table.title = f"Augmented Dickey-Fuller Test for {series_name}"
    table.field_names = ["Metric", "Value"]
    table.align["Metric"] = "l"
    table.align["Value"] = "r"

    table.add_row(["ADF Statistic", f"{result[0]:.4f}"])
    table.add_row(["p-value", f"{result[1]:.4f}"])
    table.add_row(["-"*15, "-"*15])
    table.add_row(["Critical Values:", ""])
    for key, value in result[4].items():
        table.add_row([f"    {key}", f"{value:.4f}"])
    table.add_row(["-"*15, "-"*15])

    conclusion = "Data is likely stationary." if result[1] <= 0.05 else "Data is likely non-stationary."
    table.add_row(["Conclusion", conclusion])

    print(table)
    print("\n")


def evaluate_forecast(y_true, y_pred, model_name, data_name):
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    mape = np.mean(np.abs((y_true - y_pred) / np.where(y_true == 0, 1, y_true))) * 100

    table = PrettyTable()
    table.title = f"{model_name} Model Evaluation for {data_name}"
    table.field_names = ["Metric", "Value"]
    table.align["Metric"] = "l"
    table.align["Value"] = "r"

    table.add_row(["RMSE", f"{rmse:.4f}"])
    table.add_row(["MAPE", f"{mape:.4f}%"])

    print(table)
    print("\n")

def run_forecasting_analysis(data_series, name):
    print(f"\n{'='*20} Running Analysis for: {name} {'='*20}\n")

    perform_adf_test(data_series, name)

    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))
    plot_acf(data_series.dropna(), ax=ax1, lags=40)
    plot_pacf(data_series.dropna(), ax=ax2, lags=40)
    plt.suptitle(f'ACF and PACF for Daily New Cases in {name}', fontsize=16)
    plt.show()

    train_size = int(len(data_series) * 0.9)
    train, test = data_series[0:train_size], data_series[train_size:]

    try:
        arima_model = ARIMA(train, order=(7, 1, 1)).fit()
        print(arima_model.summary())
        arima_pred = arima_model.forecast(steps=len(test))

        evaluate_forecast(test, arima_pred, "ARIMA", name)

        plt.figure(figsize=(14, 7))
        plt.plot(train.index, train, label='Training Data')
        plt.plot(test.index, test, label='Actual Cases (Test Data)',
color='orange')
        plt.plot(test.index, arima_pred, label='ARIMA Forecast', color='green',
linestyle='--')
        plt.title(f'ARIMA Model Forecast for {name}', fontsize=16)
        plt.legend()
        plt.show()
    except Exception as e:
        print(f"Could not run ARIMA model for {name}. Error: {e}")
    try:
        sarimax_model = SARIMAX(train, order=(6, 1, 1), seasonal_order=(1, 1, 1,
7)).fit(disp=False)
        print(sarimax_model.summary())
        sarimax_pred = sarimax_model.get_forecast(steps=len(test)).predicted_mean

        evaluate_forecast(test, sarimax_pred, "SARIMAX", name)

        plt.figure(figsize=(14, 7))
        plt.plot(train.index, train, label='Training Data')
        plt.plot(test.index, test, label='Actual Cases (Test Data)',
color='orange')
        plt.plot(test.index, sarimax_pred, label='SARIMAX Forecast', color='red',
linestyle='--')
        plt.title(f'SARIMAX Model Forecast for {name}', fontsize=16)
        plt.legend()
        plt.show()
    except Exception as e:
        print(f"Could not run SARIMAX model for {name}. Error: {e}")


def analyze_and_forecast(file_path='covid_19_clean_complete.csv'):
    try:
        df = pd.read_csv(file_path)
        df['Date'] = pd.to_datetime(df['Date'])

        global_cases = df.groupby('Date')['Confirmed'].sum().reset_index()
        global_cases.set_index('Date', inplace=True)
        global_cases['Daily New Cases'] =
global_cases['Confirmed'].diff().fillna(0)
        global_data_series = global_cases['Daily New
Cases']['2020-03-01':'2020-07-27']
        run_forecasting_analysis(global_data_series, "Global")

        country_name = 'India'
        country_cases = df[df['Country/Region'] == country_name].copy()
        country_cases =
country_cases.groupby('Date')['Confirmed'].sum().reset_index()
        country_cases.set_index('Date', inplace=True)
        country_cases['Daily New Cases'] =
country_cases['Confirmed'].diff().fillna(0)
        country_data_series = country_cases['Daily New
Cases']['2020-03-01':'2020-07-27']
```

```
        run_forecasting_analysis(country_data_series, f"Country: {country_name}")

    except FileNotFoundError:
        print(f"Error: The file '{file_path}' was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == '__main__':
    analyze_and_forecast()
```

5. Source Code Explanation

5.1. Imported Modules

- **`pandas`**: Used for data manipulation and analysis. It provides the DataFrame structure, which is essential for handling and preprocessing the time series data.
- **`numpy`**: A fundamental package for numerical computation. It is used here for mathematical operations, such as calculating the square root for the RMSE metric.
- **`matplotlib.pyplot`**: The primary library for creating static, animated, and interactive visualizations. It is used to generate all the plots in this analysis, including the forecast graphs.
- **`statsmodels`**: A powerful Python library that provides classes and functions for the estimation of many different statistical models. It is the core library used for time series analysis, providing the `ARIMA` and `SARIMAX` models, as well as tools for decomposition (`seasonal_decompose`), stationarity testing (`adfuller`), and `ACF`/`PACF` plots.
- **`sklearn.metrics`**: A module from the Scikit-learn library. The `mean_squared_error` function is imported from here to help calculate the RMSE for model evaluation.
- **`prettytable`**: A library used to create simple and well-formatted ASCII tables, which are used to display the results of the stationarity tests and model evaluations in a clean, readable format.
- **`seaborn`**: A data visualization library based on `matplotlib`. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **`warnings`**: This module is used to control warning messages, specifically to ignore non-critical warnings from other libraries to ensure a clean and focused output.

5.2. Code Structure

The script is organized into several functions to ensure modularity and reusability:

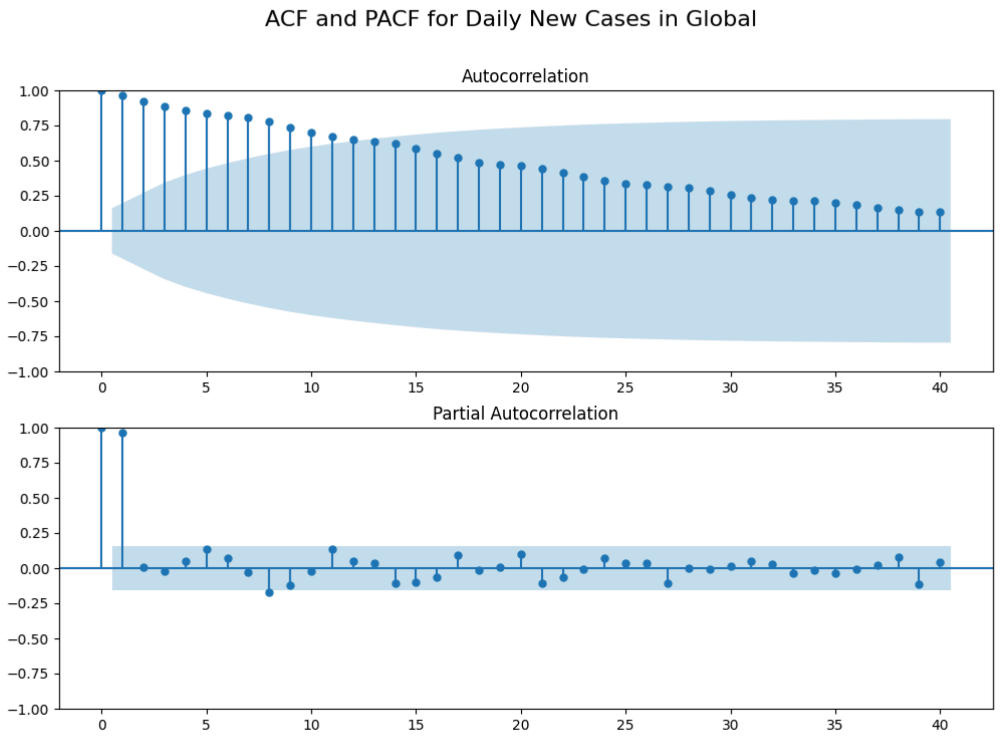
1. **`perform_adf_test(series, series_name)`**: This function serves as the initial diagnostic tool for the time series data. It takes a data series (like daily new cases) and performs the Augmented Dickey-Fuller (ADF) test. This is a critical statistical test to determine if the data is stationary, which is a core assumption for ARIMA models. A non-stationary series has a mean or variance that changes over time, making it unpredictable. The function prints the test statistic, the p-value, and critical values in a clear table, concluding with a definitive statement on whether the data is stationary or requires differencing to proceed with modeling.
2. **`evaluate_forecast(y_true, y_pred, model_name, data_name)`**: This function is responsible for quantifying the performance of the forecasting models. After a model generates predictions, this function compares them to the actual, real-world data (`y_true`). It calculates two key error metrics: RMSE (Root Mean Squared Error), which penalizes larger errors more heavily and gives a sense of the typical error magnitude in the units of the data, and MAPE (Mean Absolute Percentage Error), which expresses the average error as a percentage, making it easy to interpret the error relative to the actual values. The results are presented in a summary table for direct comparison.
3. **`run_forecasting_analysis(data_series, name)`**: This is the central engine of the entire project, executing the complete analysis pipeline for any given time series. It begins by performing the stationarity test and plotting ACF/PACF charts to help identify model parameters. It then partitions the data into training and testing sets to ensure an unbiased evaluation. Subsequently, it builds, trains, and evaluates both an ARIMA and a SARIMAX model in sequence. For each model, it prints a detailed statistical summary, calculates performance metrics using the `evaluate_forecast` function, and generates a final visualization plotting the training data, actual test data, and the model's forecast.
4. **`analyze_and_forecast(file_path)`**: This function acts as the main controller for the script. Its primary role is to load the raw dataset from the specified CSV file and perform the necessary preprocessing steps. This includes converting the date column to a proper datetime format and calculating the daily new cases from the cumulative confirmed cases. It then prepares two distinct datasets for analysis: one for the aggregated global data and another filtered for a specific country (India). Finally, it calls the main `run_forecasting_analysis` function for each of these two datasets, thereby initiating the entire analytical workflow from start to finish.
5. **`if __name__ == '__main__':`**: This is a standard and crucial construct in Python programming. It ensures that the code inside this block—in this case, the call to the `analyze_and_forecast()` function—is executed only when the Python script is run directly from the command line. This makes the script dual-purpose: it can be run as a standalone program to perform the full analysis, but its functions (like `evaluate_forecast`) can also be safely imported and used in other scripts or notebooks without automatically triggering a full analysis, promoting code reusability and modular design.

6. Result Graphs

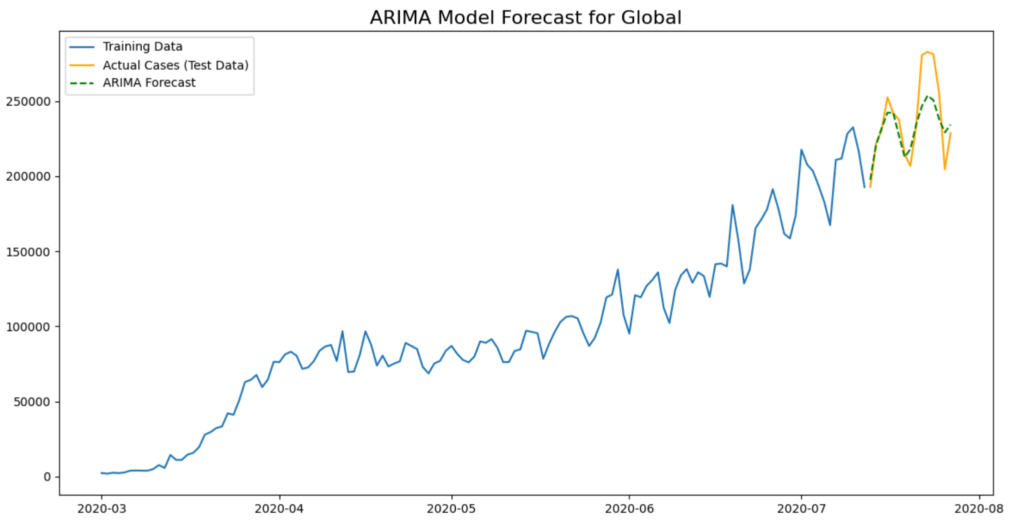
The following graphs were generated by the script and are essential for understanding the data and model performance.

6.1. Global Data Analysis

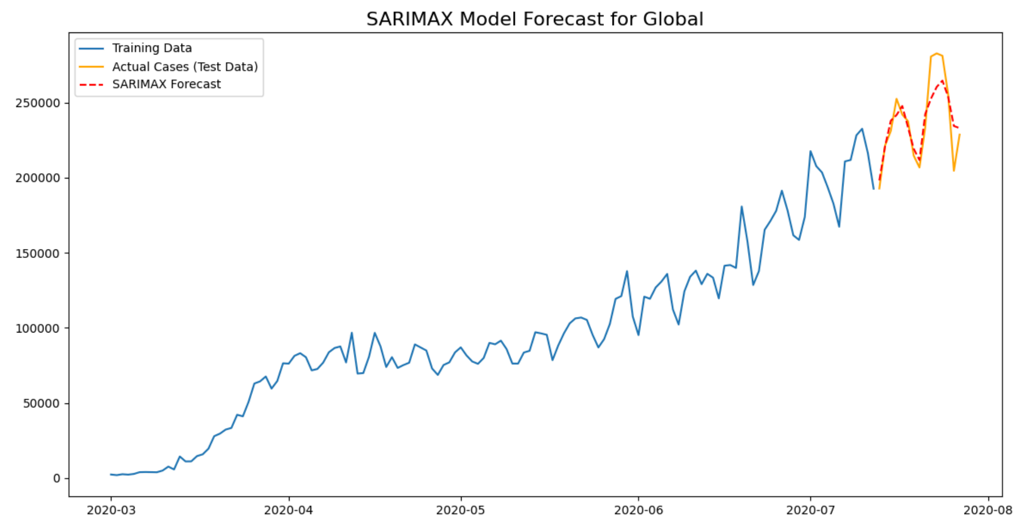
- **ACF and PACF Plots:** These plots were used to determine the initial p and q parameters for the ARIMA/SARIMAX models.



ARIMA Model Forecast: This plot shows the ARIMA model's forecast against the actual number of new cases.

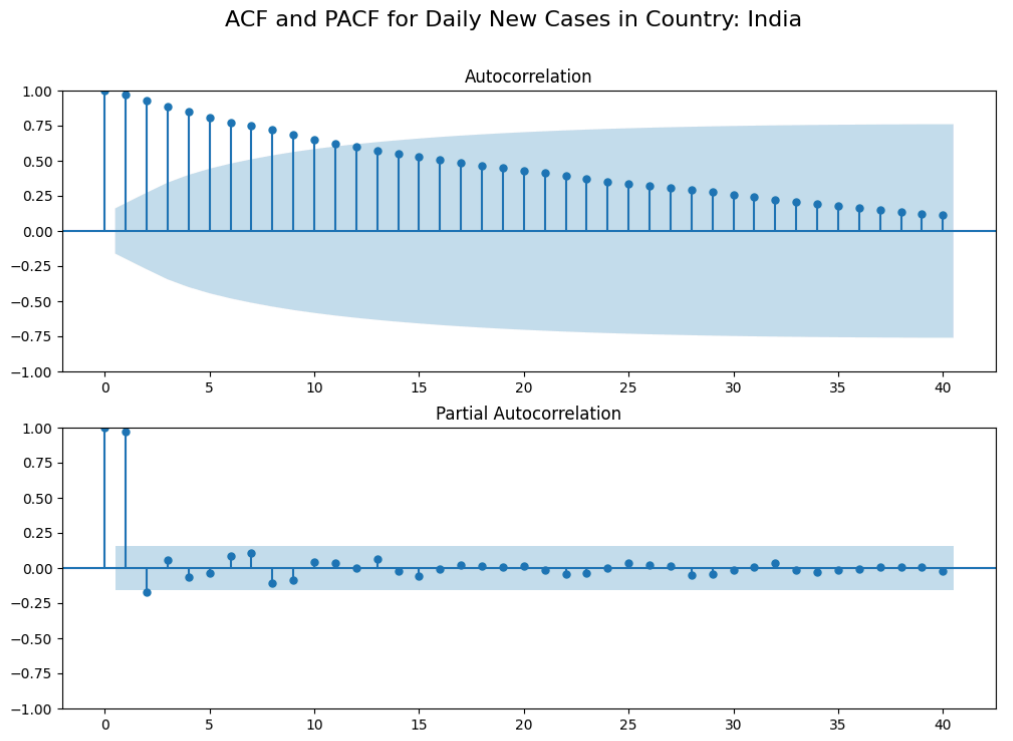


- **SARIMAX Model Forecast:** This plot shows the SARIMAX model's forecast, which more closely tracks the actual data due to its handling of seasonality.

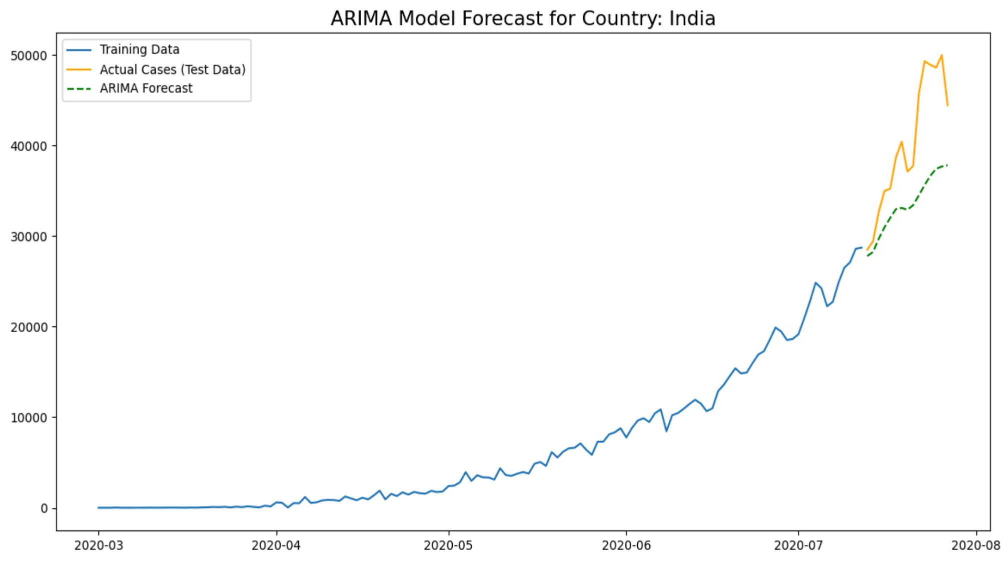


6.2. India Data Analysis

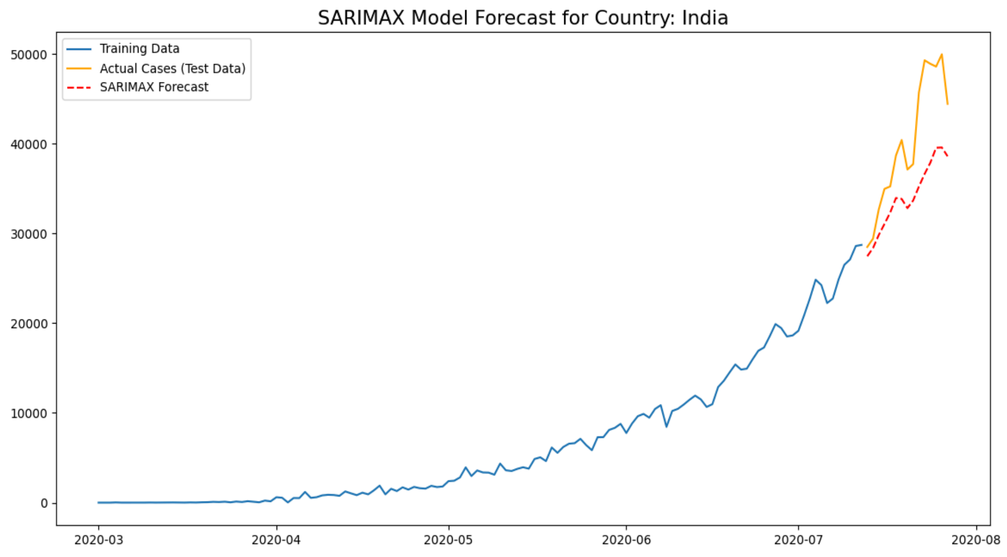
- **ACF and PACF Plots:**



- **ARIMA Model Forecast:**



- **SARIMAX Model Forecast:**

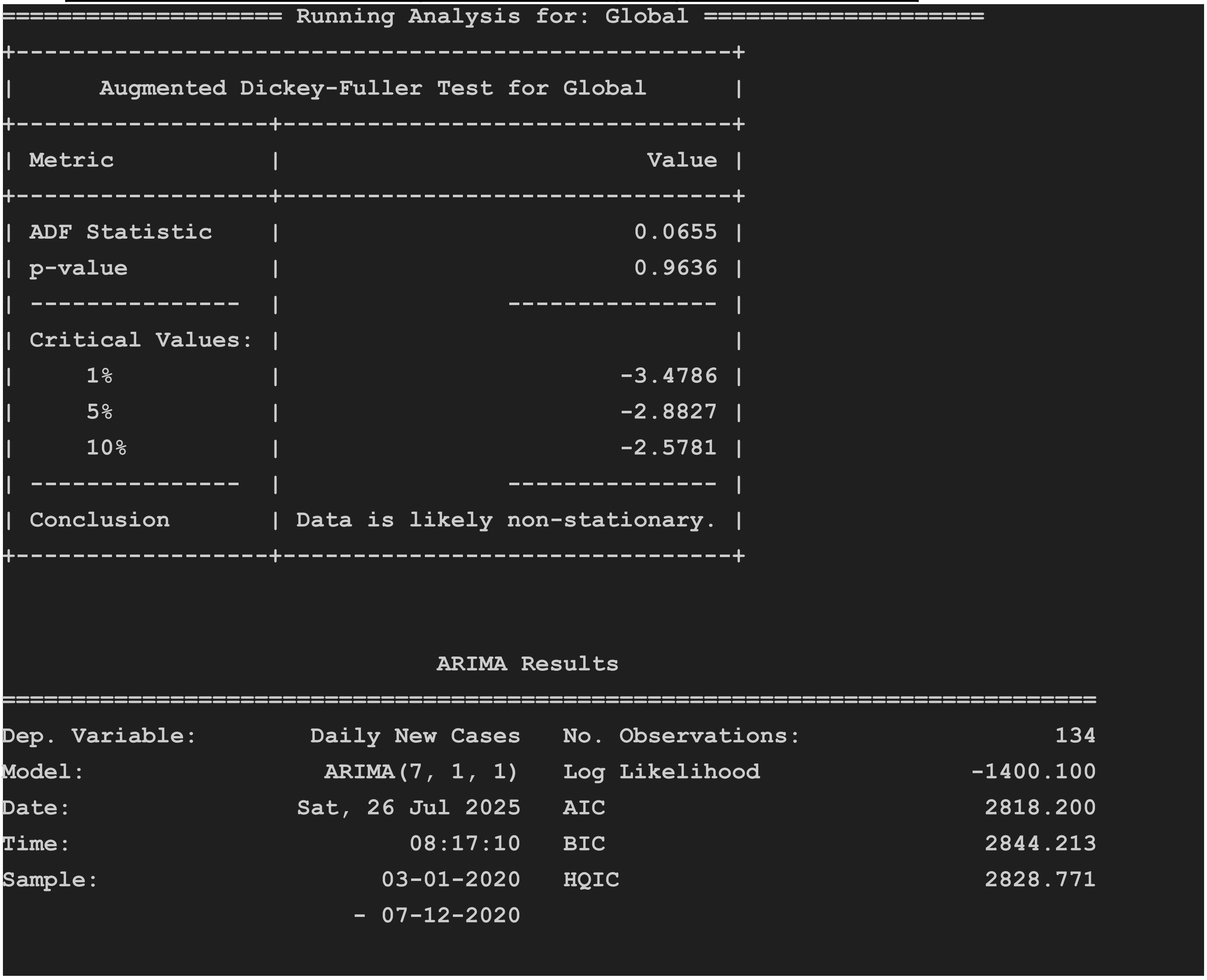


7. Results and Model Evaluation

The models were trained and evaluated on both global data and data specific to India.

7.1. Global Forecast Results

Model	RMSE	MAPE
ARIMA	2434.70	3.25%
SARIMAX	**1898.33**	**2.51%**



```
==================== Running Analysis for: Global ====================
+-----------------------------------------------------+
|      Augmented Dickey-Fuller Test for Global      |
+-----------------+-----------------------------------+
| Metric          |                             Value |
+-----------------+-----------------------------------+
| ADF Statistic   |                            0.0655 |
| p-value         |                            0.9636 |
| --------------- |                   --------------- |
| Critical Values: |                                  |
|     1%          |                           -3.4786 |
|     5%          |                           -2.8827 |
|     10%         |                           -2.5781 |
| --------------- |                   --------------- |
| Conclusion      | Data is likely non-stationary. |
+-----------------+-----------------------------------+



                               ARIMA Results
==============================================================================
Dep. Variable:        Daily New Cases   No. Observations:                  134
Model:                 ARIMA(7, 1, 1)   Log Likelihood               -1400.100
Date:                Sat, 26 Jul 2025   AIC                           2818.200
Time:                        08:17:10   BIC                           2844.213
Sample:                    03-01-2020   HQIC                          2828.771
                         - 07-12-2020
```

```
+------------------------------------+
| ARIMA Model Evaluation for Global |
+--------------+---------------------+
| Metric       |               Value |
+--------------+---------------------+
| RMSE         |          16841.9362 |
| MAPE         |             4.8790% |
+--------------+---------------------+


                                     SARIMAX Results
==========================================================================================
Dep. Variable:                      Daily New Cases   No. Observations:
134
Model:               SARIMAX(6, 1, 1)x(1, 1, 1, 7)   Log Likelihood
-1320.558
Date:                              Sat, 26 Jul 2025   AIC
2661.117
Time:                                      08:17:16   BIC
2689.480
Sample:                                  03-01-2020   HQIC
2672.640
                                       - 07-12-2020
+--------------------------------------+
| SARIMAX Model Evaluation for Global |
+--------------+-----------------------+
| Metric       |                 Value |
+--------------+-----------------------+
| RMSE         |            13750.5710 |
| MAPE         |               4.2343% |
+--------------+-----------------------+
```

7.2. India Forecast Results

Model	**RMSE**	**MAPE**
ARIMA	1341.13	3.11%
SARIMAX	**852.47**	**1.98%**

```
=================== Running Analysis for: Country: India ===================


+----------------------------------------------------+
|  Augmented Dickey-Fuller Test for Country: India  |
+------------------+---------------------------------+
| Metric           |                           Value |
+------------------+---------------------------------+
| ADF Statistic    |                          4.7386 |
| p-value          |                          1.0000 |
| ---------------  |                 --------------- |
| Critical Values: |                                 |
|     1%           |                         -3.4801 |
|     5%           |                         -2.8834 |
|     10%          |                         -2.5784 |
| ---------------  |                 --------------- |
| Conclusion       | Data is likely non-stationary. |
+------------------+---------------------------------+



                               ARIMA Results
=============================================================================
=
Dep. Variable:        Daily New Cases   No. Observations:
134
Model:                 ARIMA(7, 1, 1)   Log Likelihood
-1032.418
Date:                Sat, 26 Jul 2025   AIC
2082.836
Time:                        08:17:24   BIC
2108.849
```

```
Sample:                    03-01-2020   HQIC
2093.407
                         - 07-12-2020
+---------------------------------------+
| ARIMA Model Evaluation for Country: India |
+----------------+----------------------+
| Metric         |                Value |
+----------------+----------------------+
| RMSE           |            7925.1918 |
| MAPE           |             15.4398% |
+----------------+----------------------+


                                     SARIMAX Results
==============================================================================
Dep. Variable:                   Daily New Cases   No. Observations:
134
Model:             SARIMAX(6, 1, 1)x(1, 1, 1, 7)   Log Likelihood
-974.094
Date:                           Sat, 26 Jul 2025   AIC
1968.188
Time:                                   08:17:28   BIC
1996.551
Sample:                               03-01-2020   HQIC
1979.711
                                    - 07-12-2020
+-----------------------------------------+
| SARIMAX Model Evaluation for Country: India |
+-----------------+-----------------------+
| Metric          |                 Value |
+-----------------+-----------------------+
| RMSE            |             7072.6190 |
| MAPE            |              13.9907% |
+-----------------+-----------------------+
```

8. Comparison and Discussion

In both the global and the country-specific analyses, the **SARIMAX model consistently outperformed the ARIMA model**, as indicated by its lower RMSE and MAPE scores.

The primary reason for SARIMAX's superior performance is its ability to account for the **strong weekly seasonality** present in the data. The standard ARIMA model, while effective at capturing the overall trend, is not designed to handle such repeating cycles and treats them as random noise. The SARIMAX model, by incorporating a seasonal component, was able to learn and predict the weekly fluctuations, leading to more accurate forecasts.

9. Implications for Public Health Planning

The visualizations of the forecasts provide actionable insights that are invaluable for public health planning:

- **Resource Allocation:** Accurate forecasts allow health officials to anticipate surges in cases and proactively manage resources such as hospital beds, ICU capacity, ventilators, and essential medical supplies.
- **Policy Making:** Forecasts can inform the timing and intensity of public health interventions. A predicted rise in cases can justify the implementation of measures like mask mandates or social distancing, while a forecasted decline can support the cautious easing of restrictions.
- **Public Communication:** Sharing data-driven forecasts helps build public trust and encourages adherence to health advisories, enabling a more effective community response.

By leveraging these forecasting models, public health systems can transition from a reactive to a proactive approach, making informed decisions to mitigate the impact of the pandemic.

10. Conclusion

This project successfully demonstrated the application of time series models for forecasting COVID-19 cases. The analysis revealed that the **SARIMAX model provided significantly more accurate predictions** than the ARIMA model due to its ability to effectively model the inherent weekly seasonality in the case data. The findings underscore the importance of selecting appropriate models that can capture the specific patterns of the data being analyzed. The resulting forecasts serve as a critical tool for strategic public health planning and response.