National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Arpitha Bhaskara Rao Ghat
Student ID: 22123318

School of Computing
National College of Ireland

Supervisor: Cristina Hava Muntean

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Arpitha Bhaskara Rao Ghat……. ……
Student ID:	22123318……..……
Programme:	MSc Data Analytics…… **Year:** 2023……...
Module:	MSc Research Project……
Lecturer:	14th Decmber 2023……
Submission Due Date:	……..……
Project Title:	Future Evolution of Telemedicine: Enhancing Healthcare Accessibility and Reliability through the Integration of Machine Learning Techniques
Word Count:	**1661**…… **Page Count: 11**……..…….……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……

Date: ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Arpitha Bhaskara Rao Ghat
Student ID: 22123318

1 Introduction

This document gives a detailed description of the steps to be followed to replicate the results of the research that have been enlisted in the project report. The document covers the different tools required to simulate the experiments and provides a detailed explanation of the steps to be followed for code execution.

2 Environment Setup

2.1 Hardware Specifications

For successful telemedicine research project implementation, we ensure hardware configurations match these specifications:

- **Processor:** Used a bespoke M2 chip with an 8-core CPU. The robust processor architecture is ideal for LSTM and Bi-LSTM machine learning models used in telemedicine forecasting.
- **Memory (RAM):** For smooth machine learning algorithm execution and effective time series analysis of huge datasets, 16 GB of unified memory is recommended.
- **Storage:** Chose SSD storage from 256 GB to 2 TB for the project's needs. Working with large datasets requires ample storage for datasets, codebase, and model files.
- **Graphics:** The embedded M2 chip's GPU for graphics can aid machine learning and visualization in the telemedicine research project.

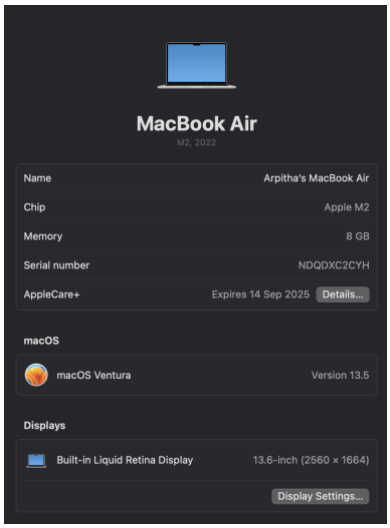

Figure 1: Hardware Specifications

2.2 Software Requirements

We ensure the following software components are installed in the system:

- **Anaconda Navigator (Version 2.4.2):** Anaconda provides a comprehensive platform for Python-based data science and machine learning, streamlining package management and environment setup.
- **Python (Version 3.7.6):** The Python programming language serves as the foundation for the codebase, ensuring compatibility with machine learning libraries and tools.

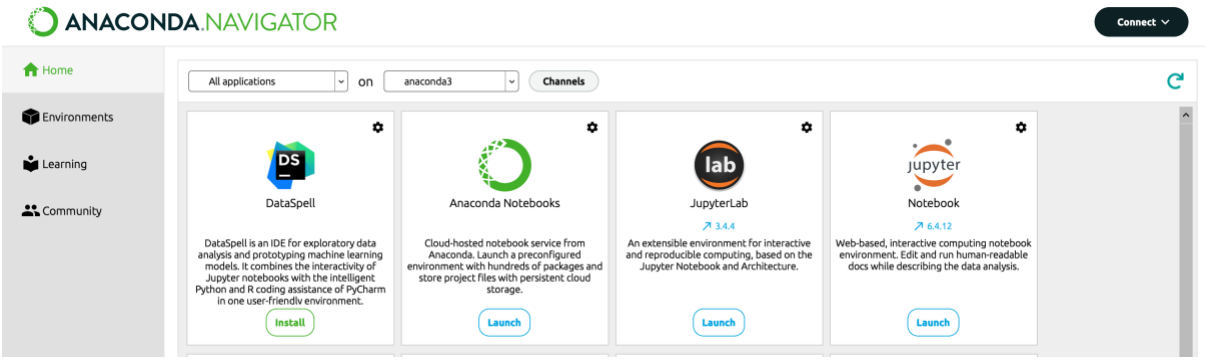

Figure 2: Anaconda Navigator for software requirements

2.3 Code Execution

The python code used to run the technical aspect of the research can be found in the artefact zip file saved as **"22123318 | Python Code and Dataset".**

Following these steps on JupyterLab, we execute the code:

1. Launch JupyterLab 3.4.4 from Anaconda Navigator – anaconda 3.
2. The interface opens in web browser, displaying the system's folder structure.
3. Navigate to the folder containing the code file.
4. Open the code file within the notebook.
5. To run the code, navigate to the Kernel menu and select "Run all cells."

3 Data Collection

The dataset, "Telemedicine Use in the Last 4 Weeks" is sourced from the National Center for Health Statistics (NCHS) and the Health Resources and Services Administration's Maternal and Child Health Bureau (HRSA MCHB). Collected through the Household Pulse Survey, an experimental data system, this 20-minute online survey monitors recent changes in telemedicine use, providing crucial insights into the social and economic impacts of the COVID-19 pandemic on American households.

3.1 Data Exploration

We use various libraries to enhance our code functionality. Pandas and NumPy assist in handling and analyzing data efficiently. The pandas_datareader is employed for fetching financial data from the web, while matplotlib aids in creating visualizations. Scikit-learn

provides metrics for evaluating model performance, and statsmodels supports time series analysis. The Keras and TensorFlow libraries are essential for building and training neural network models, and warnings are managed using the warnings library. These tools collectively streamline tasks such as data manipulation, statistical analysis, machine learning, and deep learning, contributing to the overall effectiveness of our code.

```
#Import relevant libraries
import pandas as pd
import numpy as np
import pandas_datareader.data as web
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from statsmodels.tsa.stattools import kpss
from statsmodels.graphics.tsaplots import plot_pacf, plot_acf
from statsmodels.tsa.api import ARIMA, SARIMAX
from sklearn.metrics import mean_squared_error,  mean_absolute_error
import functools as ft
from keras.models import Sequential
import tensorflow as tf
from keras.layers import Dense, LSTM, Input, Bidirectional, Flatten, Dropout
import warnings
warnings.filterwarnings("ignore")
```

Figure 3: Necessary Python Libraries required to run the time series models for forecasting accuracy

3.2 Data Analysis and Model Preparation

The code pd.to_datetime(data['Time Period Start Date']) is employed to specifically convert the 'Time Period Start Date' column from an object data type to a datetime data type. This conversion is crucial in time series modelling, particularly when analysing temporal patterns and trends.

```
# converting column to datetime type from ojbect type
data['Time Period Start Date'] = pd.to_datetime(data['Time Period Start Date'])
data
```

	Indicator	Group	State	Subgroup	Phase	Time Period	Time Period Start Date	Value	Low CI	High CI
0	Adults Who Had Appointment with Health Profess...	National Estimate	United States	United States	3.1	28	2021-04-14	25.7	25.0	26.4
1	Adults Who Had Appointment with Health Profess...	By Age	United States	18 - 29 years	3.1	28	2021-04-14	21.6	19.2	24.1
2	Adults Who Had Appointment with Health Profess...	By Age	United States	30 - 39 years	3.1	28	2021-04-14	23.1	21.7	24.5
3	Adults Who Had Appointment with Health Profess...	By Age	United States	40 - 49 years	3.1	28	2021-04-14	25.7	24.2	27.3
4	Adults Who Had Appointment with Health Profess...	By Age	United States	50 - 59 years	3.1	28	2021-04-14	26.3	24.6	28.1
...	...	...	...	...	...	...	...	...	...	...
3338	Households With Children Where Any Child Had A...	By State	Vermont	Vermont	3.5	48	2022-07-27	23.1	14.6	33.6
3339	Households With Children Where Any Child Had A...	By State	Virginia	Virginia	3.5	48	2022-07-27	18.8	12.9	26.1
3340	Households With Children Where Any Child Had A...	By State	Washington	Washington	3.5	48	2022-07-27	18.9	15.1	23.3
3342	Households With Children Where Any Child Had A...	By State	Wisconsin	Wisconsin	3.5	48	2022-07-27	13.8	8.8	20.3
3343	Households With Children Where Any Child Had A...	By State	Wyoming	Wyoming	3.5	48	2022-07-27	10.7	6.6	16.2

Figure 4: Converting Time Period Start Date to datetime format as it is crucial for time series analysis

The KPSS test for stationarity on a given time series calculates the test statistic, p-value, and critical values, then prints the results and determines the stationarity of the series based on the significance level. Additionally, the code generates and displays autocorrelation function (ACF) and partial autocorrelation function (PACF) plots for further analysis.

```
kpss_test(data['Value'])  # Test indicate that the data follows trend and seasonality
```

```
KPSS Statistic: 3.023199187659166
p-value: 0.01
num lags: 28
Critial Values:
   10% : 0.347
   5% : 0.463
   2.5% : 0.574
   1% : 0.739
Result: The series is non-stationary
```

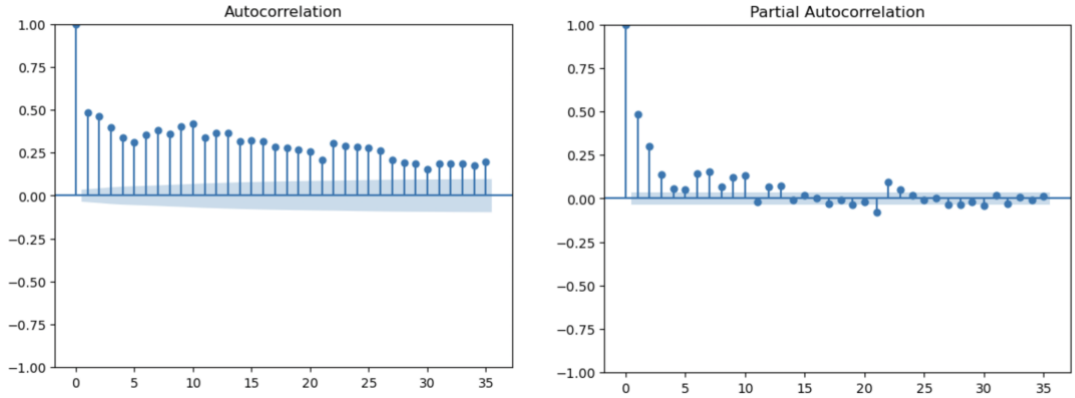

Figure 5: Autocorrelation Function (ACF) Plot and Partial Autocorrelation Function (PACF) Plot of Differenced Time Series

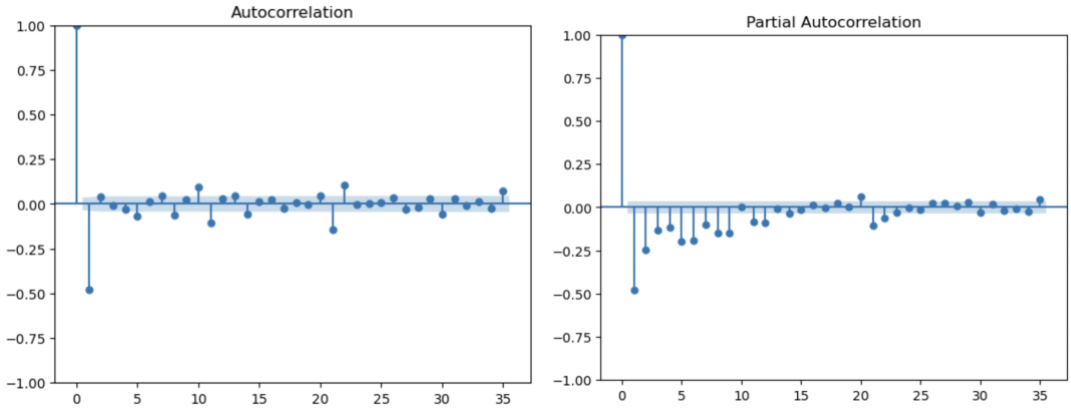

Figure 6: Autocorrelation Function (ACF) Plot and Partial Autocorrelation Function (PACF) Plot for Stationarised Time Series

3.3 Model Import and Model Forecasting

- The **ARIMA** model is configured with parameters (28, 1, 20) for autoregression, differencing, and moving average. Here choosing p=28 for the number of lags, d=1

for differencing achieving stationarity, and q=20 (p, d, q) for the moving average, considers lags, stationarity through differencing, and changes in plots after 20 lags, reflecting the moving average impact in the time series model.

- In the **SARIMAX** model, the seasonal component is explicitly introduced to capture recurring patterns that occur at regular intervals. The seasonal order (0, 1, 2, 28) indicates that there is a seasonal moving average component (2) with a seasonal period of 28. The value of 0 for the seasonal autoregressive order (P) indicates that there is no seasonal autoregressive component. So, while the autoregressive and differencing components may resemble those in ARIMA, the focus is on seasonal differencing (D), seasonal moving average (Q), and the seasonal period (S) of 28-time units. so SARIMAX suggests a pattern that repeats every 28 observations.

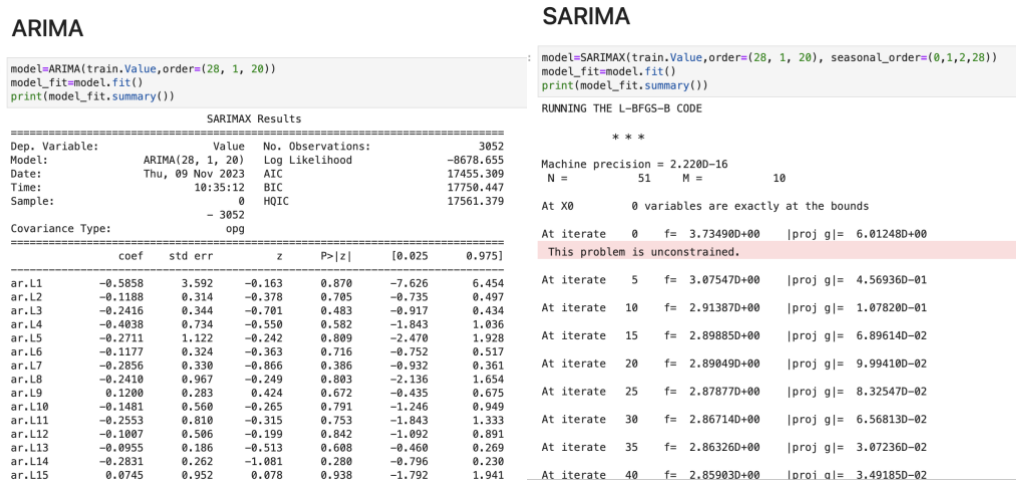

Figure 7: ARIMA and SARIMAX Model Forecasting

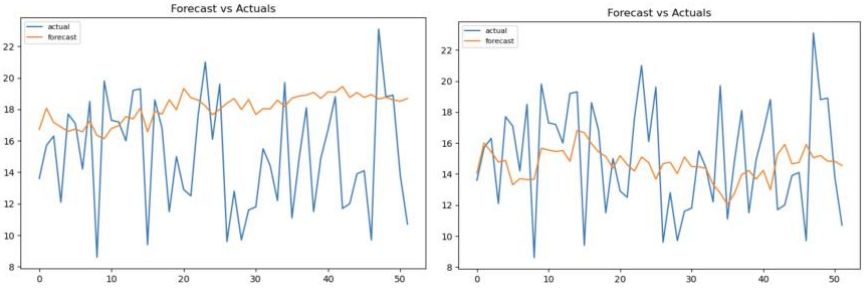

Figure 8: ARIMA and SARIMA Model Forecast versus Actual Time Series Data

- The reshaping of input data, such as X_train and X_test, is crucial for the **LSTM** model to process sequential information. The LSTM architecture is configured with multiple layers, specifying units, activation functions, and dropout rates. The model is compiled with mean squared error as the loss function and Adam optimizer. It is trained on the training set (X_train, Y_train) for 10 epochs. The LSTM forecasts are then compared with actual values, and the model's performance is evaluated using metrics such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). The LSTM's significance lies in its ability to capture complex temporal patterns, as evidenced by its lower MSE compared to ARIMA and SARIMA models.

- **Bidirectional LSTM** processes input sequences in both forward and backward directions, capturing more extensive temporal dependencies compared to unidirectional LSTM. The Bidirectional LSTM is configured with multiple layers, including bidirectional ones, specifying units, activation functions, and dropout rates. By processing input sequences in both forward and backward directions, Bidirectional LSTM captures more comprehensive temporal dependencies. The model is trained on the training set (X_train, Y_train) for 10 epochs. The Bidirectional LSTM's forecasts are then compared with actual values, and its performance is evaluated using metrics such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). The addition of bidirectional processing allows the model to potentially capture more complex patterns, demonstrating its superior ability, as evidenced by its competitive MSE compared to other models.

LSTM

```
X_train = X_train.to_numpy().reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.to_numpy().reshape((X_test.shape[0], X_test.shape[1], 1))
X_train.shape,  X_test.shape
```

```
((3052, 8, 1), (52, 8, 1))
```

```
model.fit(X_train, Y_train, epochs=10)
```

```
Epoch 1/10
96/96 [==============================] - 2s 9ms/step - loss: 196.7917 - mean_squared_error: 196.7917 - mean_absolute_error: 1
1.5409
Epoch 2/10
96/96 [==============================] - 1s 8ms/step - loss: 76.8431 - mean_squared_error: 76.8431 - mean_absolute_error: 6.96
76
Epoch 3/10
96/96 [==============================] - 1s 8ms/step - loss: 55.9310 - mean_squared_error: 55.9310 - mean_absolute_error: 5.92
20
Epoch 4/10
96/96 [==============================] - 1s 8ms/step - loss: 45.5986 - mean_squared_error: 45.5986 - mean_absolute_error: 5.28
44
Epoch 5/10
96/96 [==============================] - 1s 8ms/step - loss: 41.3179 - mean_squared_error: 41.3179 - mean_absolute_error: 5.02
47
Epoch 6/10
96/96 [==============================] - 1s 8ms/step - loss: 34.3282 - mean_squared_error: 34.3282 - mean_absolute_error: 4.52
50
Epoch 7/10
96/96 [==============================] - 1s 8ms/step - loss: 29.2332 - mean_squared_error: 29.2332 - mean_absolute_error: 4.16
70
Epoch 8/10
96/96 [==============================] - 1s 8ms/step - loss: 27.2490 - mean_squared_error: 27.2490 - mean_absolute_error: 4.06
23
Epoch 9/10
96/96 [==============================] - 1s 8ms/step - loss: 27.6247 - mean_squared_error: 27.6247 - mean_absolute_error: 4.03
13
Epoch 10/10
96/96 [==============================] - 1s 8ms/step - loss: 25.6535 - mean_squared_error: 25.6535 - mean_absolute_error: 3.89
22
<keras.callbacks.History at 0x7f8dd0565730>
```

Figure 9: LSTM Model Forecasting

Bidirectional LSTM

```
#Applying the LSTM model
model = Sequential()

# Configuring the parameters
model.add(Bidirectional(LSTM(units=128,activation = 'relu', return_sequences = True, input_shape = (X_train.shape[0],X_train.sh
# Adding a dropout layer
model.add(Dropout(0.2))

model.add(Bidirectional(LSTM(units=64,activation = 'relu', return_sequences = True)))
model.add(Dropout(0.5))

model.add(Bidirectional(LSTM(units=32,activation = 'relu', return_sequences = True)))
model.add(Dropout(0.5))

model.add(Bidirectional(LSTM(units=16,activation = 'relu')))
model.add(Dropout(0.2))

model.add(Dense(units=1,activation = 'relu'))
```

```
model.fit(X_train, Y_train, epochs=10)
```

```
Epoch 1/10
96/96 [==============================] - 5s 14ms/step - loss: 108.1225 - mean_squared_error: 108.1225 - mean_absolute_error:
8.1354
Epoch 2/10
96/96 [==============================] - 1s 14ms/step - loss: 35.8630 - mean_squared_error: 35.8630 - mean_absolute_error: 4.6
497
Epoch 3/10
96/96 [==============================] - 1s 15ms/step - loss: 24.1945 - mean_squared_error: 24.1945 - mean_absolute_error: 3.8
083
Epoch 4/10
96/96 [==============================] - 1s 15ms/step - loss: 17.1183 - mean_squared_error: 17.1183 - mean_absolute_error: 3.1
773
Epoch 5/10
96/96 [==============================] - 1s 14ms/step - loss: 15.8872 - mean_squared_error: 15.8872 - mean_absolute_error: 3.0
756
Epoch 6/10
96/96 [==============================] - 1s 14ms/step - loss: 13.4224 - mean_squared_error: 13.4224 - mean_absolute_error: 2.8
489
Epoch 7/10
96/96 [==============================] - 1s 14ms/step - loss: 11.6034 - mean_squared_error: 11.6034 - mean_absolute_error: 2.6
251
Epoch 8/10
96/96 [==============================] - 1s 14ms/step - loss: 12.2463 - mean_squared_error: 12.2463 - mean_absolute_error: 2.6
711
Epoch 9/10
96/96 [==============================] - 1s 15ms/step - loss: 11.2254 - mean_squared_error: 11.2254 - mean_absolute_error: 2.5
740
Epoch 10/10
96/96 [==============================] - 2s 16ms/step - loss: 11.4677 - mean_squared_error: 11.4677 - mean_absolute_error: 2.6
173
<keras.callbacks.History at 0x7f8bb01aed00>
```

Figure 10: Bi-LSTM Model Forecasting

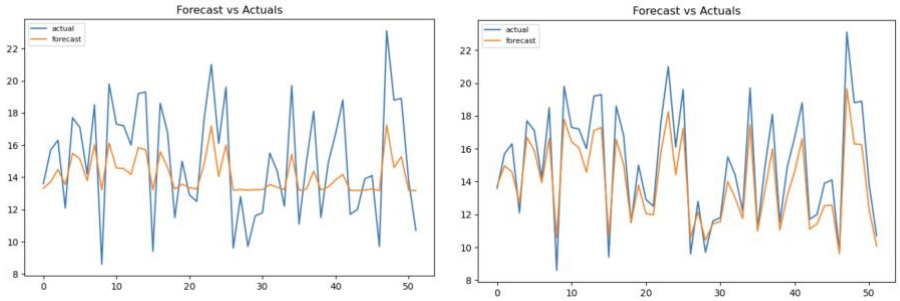

Figure 11: LSTM Model and Bi-LSTM Model Forecast versus Actual Time Series Data

Comparitive Analysis

```
score.columns = ['Model', 'Mean Squared Error', 'RMSE', "Mean Absolute Error"]
score
```

	Model	Mean Squared Error	RMSE	Mean Absolute Error
0	ARIMA	22.280399	4.720212	3.823641
0	SARIMA	12.781661	3.575145	2.974683
0	LSTM	7.171037	2.677879	2.348136
0	Bidirectional LSTM	2.390244	1.546042	1.323637

Figure 12: Comparative Analysis of 4 time series models considering the error metrics – MSE, RMSE, MAE

With the comparison of the performance of four forecasting models against actual telemedicine metrics, the ARIMA model shows the largest deviation, while SARIMA improves on this, suggesting better handling of seasonality. The LSTM model's predictions are closer to the actual data, and the Bi-LSTM model's forecasts align most closely, confirming its superior accuracy as indicated by its lowest error metrics (MSE, RMSE, MAE). The graph solidifies the quantitative analysis, showing that the Bi-LSTM model is not only statistically superior but also practically more aligned with the actual data trends. This comparative analysis is essential for healthcare practitioners and policymakers who rely on accurate forecasts to make informed decisions in the telemedicine domain.

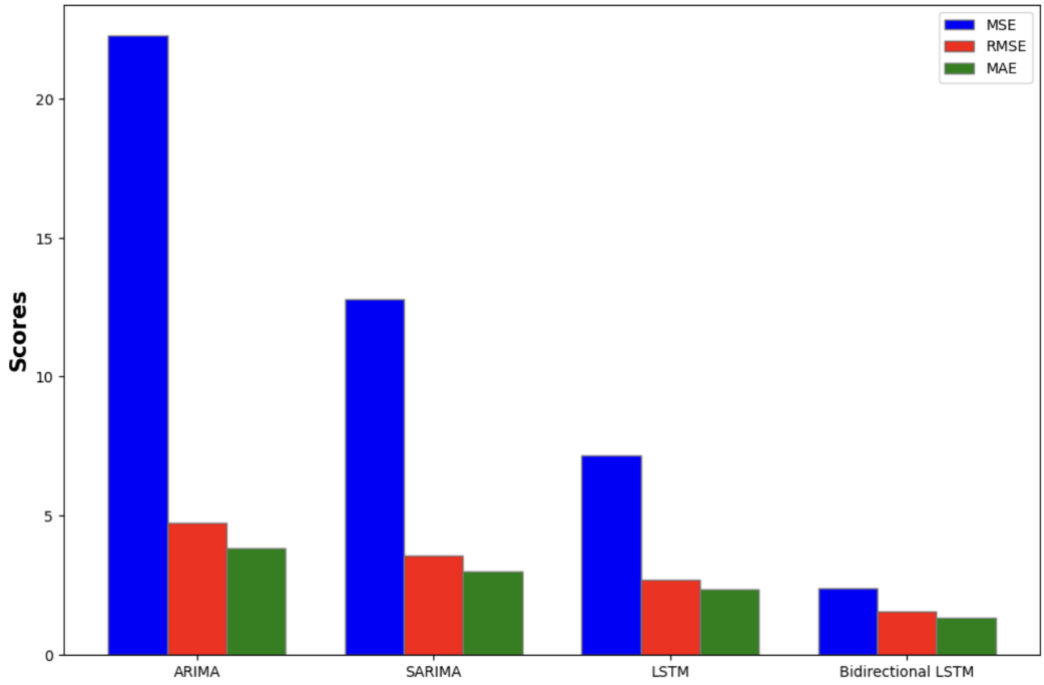

Figure 13: MSE, RMSE, MAE comparison for the 4 time series models implemented.

The Mean Squared Error (MSE) sees a reduction of approximately 89.3% from ARIMA to SARIMA, 44.5% from SARIMA to LSTM, and a substantial 66.6% decrease from LSTM to Bidirectional LSTM. Although there are changes in MAE and RMSE, we prioritize MSE as the primary error metric. The transition from ARIMA to SARIMA resulted in a roughly 20.2% increase in accuracy, while the shift from SARIMA to LSTM marked an approximately 59.8% improvement. Moving from LSTM to Bi-LSTM saw a significant

jump, yielding around 66.6% higher accuracy. The cumulative accuracy improvement from ARIMA to Bi-LSTM is approximately **89.42%.**

The plot reveals a maximum coincidence signifies that Bidirectional LSTM has outperformed, exhibiting superior accuracy in predicting telemedicine trends. The alignment underscores the model's exceptional performance, with a near-perfect match between predicted and actual values.

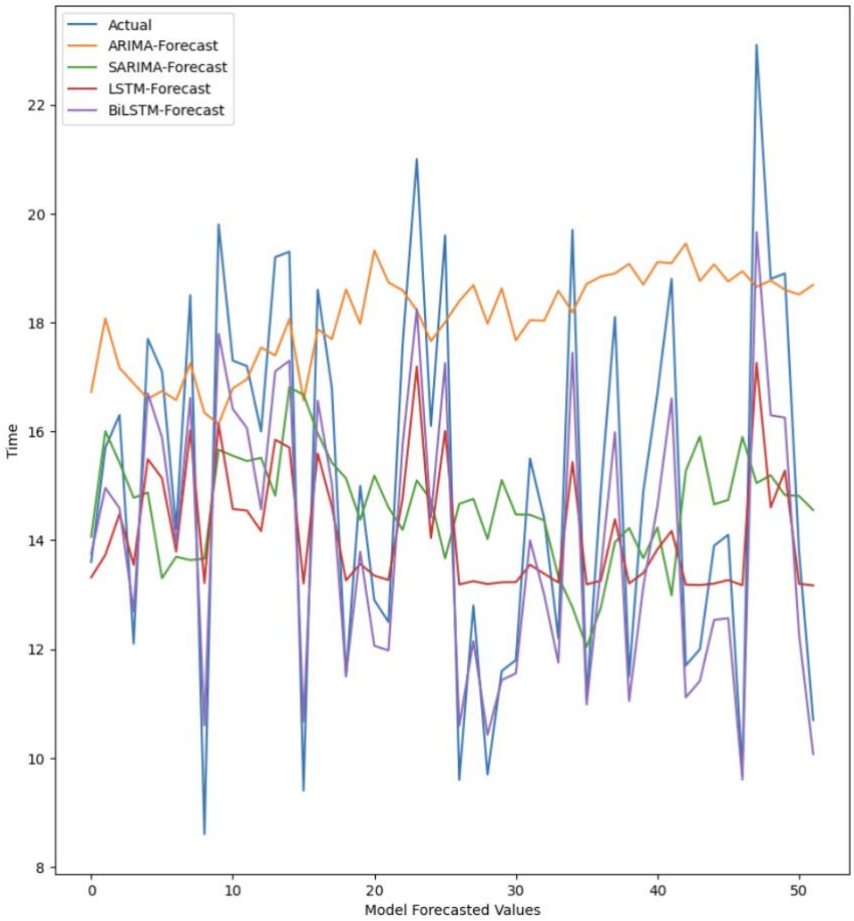

Figure 14: Comparative Forecasting Accuracy of ARIMA, SARIMA, LSTM, and Bi-LSTM Models Against Actual Telemedicine Data