**Forecasting Covid-19 Mortality Rate in West Africa**



**AMADEVBORHO, ABUSOMWAN**

**Master of Science in Data Analytics**

**2021/2022**

# DECLARATION

I hereby declare that this material, which I have now summited for assessment on the program of study leading to the award of Master of Science in Data Analytics is my work and to the best of my knowledge and belief, it contains no material previously published or written by another person, only to the extent that such work has been cited within the text of my work.

Furthermore, future students may use the material contained in this thesis provided that the source is acknowledged or cited in full.

**K00266018**

**Amadevborho, Abusomwan**

# ABSTRACT

COVID-19, a new acute respiratory disease, has hit the planet since 2019, killing millions and affecting the economy, politics, and civilization. As a result, an accurate prediction of COVID-19's mortality becomes critical in such a circumstance. Four different models were investigated in this comparative study, Hence, these include two classical time series models focusing on the Autoregressive Integrated Moving Average (ARIMA) model, the PROPHET model, and two Machine Learning (ML) models focusing on Random Forest (RF) model, and the Light Gradient Boosting Machine model(LGBM). This study sets to determine which has the best performance when predicting the future case trends of COVID-19 in West African countries. However, After the collection of publicly available COVID-19 mortality case data from the Johns Hopkins University Center for Systems Science and Engineering database, repeated experiments were carried out that used the data to forecast future trends for all models. Nevertheless, The Mean Squared Error (MSE) measure is then used to assess performance. The results reveal that the ARIMA model has the highest overall performance for all nations with exceptionally low MSE. Hence, making the ARIMA model the champion model for forecasting. Also, as a result, in comparison to the PROPHET, RF, and LGBM, the statistical model ARIMA performed better for forecasting COVID-19 mortality rate. Nigeria was also discovered to have the highest mortality rate in West Africa. This study demonstrates the great accuracy of the ARIMA model, which can be used to anticipate the spread of COVID-19 so that West African countries and their various government authorities may be more prepared and alert when managing the spread in a long/short run.

# ACKNOWLEDGEMENT

I would like to extend special gratitude to my lecturer and supervisor Dr Matt Horrigan for providing support and guidance throughout the project research/completion process.

Also, special thanks to my family and friends for their heart-warming encouragements and supports during this time.

# Table of Contents

[DECLARATION i](#_Toc113218275)

[ABSTRACT ii](#_Toc113218276)

[ACKNOWLEDGEMENT iii](#_Toc113218277)

[Table of Contents iv](#_Toc113218278)

[Table of Tables vii](#_Toc113218279)

[Table of Figures vii](#_Toc113218280)

[Tables of Equations viii](#_Toc113218281)

[ABBREVIATION AND NOTATION ix](#_Toc113218282)

[CHAPTER 1 10](#_Toc113218283)

[INTRODUCTION 10](#_Toc113218284)

[1.1 BACKGROUND 10](#_Toc113218285)

[1.2 AIMS AND OBJECTIVE 12](#_Toc113218286)

[1.3 RESEARCH QUESTIONS 12](#_Toc113218287)

[1.4 DEFINITION OF TERMS 13](#_Toc113218288)

[1.5 THESIS STRUCTURE 13](#_Toc113218289)

[CHAPTER 2 14](#_Toc113218290)

[LITERATURE REVIEW 14](#_Toc113218291)

[2.1 TIME-SERIES ANALYSIS 14](#_Toc113218292)

[2.1.1 ARIMA 14](#_Toc113218293)

[2.1.2 PROPHET 15](#_Toc113218294)

[2.2 MACHINE LEARNING 16](#_Toc113218295)

[2.2.1 RANDOM FOREST 17](#_Toc113218296)

[2.3 RELATED WORK 18](#_Toc113218297)

[2.4 IDENTIFICATION OF KNOWLEDGE GAPS 24](#_Toc113218298)

[CHAPTER 3 26](#_Toc113218299)

[RESEARCH METHOD 26](#_Toc113218300)

[3.1 INTRODUCTION TO CRISP-DM 26](#_Toc113218301)

[3.2 Business Understanding 27](#_Toc113218302)

[3.2.1 Determine Business objectives 27](#_Toc113218303)

[3.2.2 Research Goals 28](#_Toc113218304)

[3.2.3 Produce Project Plan 28](#_Toc113218305)

[3.3 Data Understanding 29](#_Toc113218306)

[3.3.1 Data Collection 29](#_Toc113218307)

[3.3.2 Data Description 30](#_Toc113218308)

[3.3.3 Data Exploration 31](#_Toc113218309)

[3.3.4 Verify Data Quality 32](#_Toc113218310)

[3.4 Data Preparation 33](#_Toc113218311)

[3.4.1 Data Selection 33](#_Toc113218312)

[3.4.2 Data Cleaning 33](#_Toc113218313)

[3.4.3 Data Construction 33](#_Toc113218314)

[3.5 Modelling 34](#_Toc113218315)

[3.5.1 Model Selection 34](#_Toc113218316)

[3.5.2 Generate Train and Test Data 38](#_Toc113218317)

[3.6 Evaluation 39](#_Toc113218318)

[3.7 CROSS-VALIDATION 39](#_Toc113218319)

[3.8 Deployment 40](#_Toc113218320)

[CHAPTER 4 41](#_Toc113218321)

[ANALYSIS OF FINDINGS 41](#_Toc113218322)

[4.1.1 ARIMA MODEL 41](#_Toc113218323)

[4.1.2 PROPHET MODEL 44](#_Toc113218324)

[4.2 MACHINE LEARNING 48](#_Toc113218325)

[4.2.1 Random Forest 48](#_Toc113218326)

[4.2.2 Light Gradient Boosting Machine 51](#_Toc113218327)

[4.3 MODEL SUMMARY 53](#_Toc113218328)

[CHAPTER 5 56](#_Toc113218329)

[CONCLUSION 56](#_Toc113218330)

[5.1 CONCLUSION 56](#_Toc113218331)

[5.1.1 Forecast horizon 56](#_Toc113218332)

[5.2 LIMITATIONS 57](#_Toc113218333)

[5.3 FUTURE WORKS 57](#_Toc113218334)

[Bibliography 58](#_Toc113218335)

[APPENDIX 64](#_Toc113218336)

[============NOTEBOOK FOR DATA EXPLORATION AND MODELLING ========= 64](#_Toc113218337)

[5.3.1 Business Understanding 65](#_Toc113218338)

[5.3.2 Data Prepartion 69](#_Toc113218339)

[5.3.3 Data selection 69](#_Toc113218340)

[5.3.4 Modelling 77](#_Toc113218341)

[5.3.5 list of MSE errors 117](#_Toc113218342)

[5.4 =================Deployment=================== 117](#_Toc113218343)

[5.4.1 File structure 117](#_Toc113218344)

# Table of Tables

[Table 2.1 - RELATED WORKS 23](#_heading=h.49x2ik5)

[Table 3.1 - CRISP-DM process description (Alliance A. et al 2021) 28](#_heading=h.ihv636)

[Table 3.2 - Datasets Description 35](#_heading=h.3l18frh)

[Table 4.1 - Summary table for ARIMA times series model 61](#_heading=h.2ce457m)

[Table 4.2 - Summary table for PROPHET times series model 65](#_heading=h.338fx5o)

[Table 4.3 - RF hyper-parameters 66](#_heading=h.2hio093)

[Table 4.4 - Summary table for Random Forest model 68](#_heading=h.2uxtw84)

[Table 4.5 - LightGBM hyper-parameters 68](#_heading=h.3u2rp3q)

[Table 4.6 - Summary table for LightGBM model 71](#_heading=h.1nia2ey)

[Table 4.7 - Overview of the models errors 71](#_heading=h.2mn7vak)

# Table of Figures

[Figure 2.1 - Prediction of daily page views using PROPHET (screenshot from PROPHET official website) 13](#_heading=h.1ksv4uv)

[Figure 2.2 - Random Forest Diagram (Dietterich T.G 2010) 17](#_heading=h.1ci93xb)

[Figure 2.3 - The SIRD compartmental model 23](#_heading=h.1pxezwc)

[Figure 3.1 - The CRISP-DM process model(Alliance A. et al 2021) 28](#_heading=h.23ckvvd)

[Figure 3.2 - line chart showing death rate of West Africa from 2020 till date 37](#_heading=h.4k668n3)

[Figure 3.19 - Statistical forecasting procedure 48](#_heading=h.48pi1tg)

[Figure 3.20 - Characteristics of Lightgbm 50](#_heading=h.1gf8i83)

[Figure 3.21 - Growth Approach of Lightgbm 51](#_heading=h.40ew0vw)

[Figure 4.1 - Cross Validation Kfold (Arlot et al 2016) 55](#_heading=h.meukdy)

[Figure 4.2 - Rolling-origin accumulated (McDermott et al, 2019) 56](#_heading=h.36ei31r)

[Figure 4.3 - Rolling-origin constant (McDermott et al, 2019) 57](#_heading=h.1ljsd9k)

[Figure 4.4 - Autocorrelation plot 58](#_heading=h.2koq656)

[Figure 4.5 - Partial Correlation 58](#_heading=h.zu0gcz)

[Figure 4.6 - Actual historical data vs Predicted 59](#_heading=h.1yyy98l)

[Figure 4.7 - Actual historical data vs Predicted for 200 days 60](#_heading=h.4iylrwe)

[Figure 4.8 -Actual historical data vs Predicted for last 100days and 14 days 60](#_heading=h.2y3w247)

[Figure 4.9 - Actual historical data vs Predicted for last 200 days and 14 days 61](#_heading=h.1d96cc0)

[Figure 4.10 - Deaths Scaled 62](#_heading=h.4anzqyu)

[Figure 4.11 - Trends plot 63](#_heading=h.2pta16n)

[Figure 4.12 - Actual historical data vs Predicted 63](#_heading=h.14ykbeg)

[Figure 4.13 - Actual historical data vs Predicted for 200 days 64](#_heading=h.3oy7u29)

[Figure 4.14 - Actual historical data vs Predicted for the last 100days plus 14 days 64](#_heading=h.243i4a2)

[Figure 4.15 - Actual historical data vs Predicted for last 200days plus 14 days 65](#_heading=h.j8sehv)

[Figure 4.16 - Actual daily historical data vs Predicted 67](#_heading=h.wnyagw)

[Figure 4.17 - Actual daily historical data vs Predicted for 200 days 67](#_heading=h.1vsw3ci)

[Figure 4.18 - Actual daily historical data vs Predicted last 100 days plus 14 days 68](#_heading=h.4fsjm0b)

[Figure 4.19 - Actual daily historical data vs Predicted 69](#_heading=h.2981zbj)

[Figure 4.20 - Actual daily historical data vs Predicted for 200 days 70](#_heading=h.odc9jc)

[Figure 4.21 - Actual daily historical data vs Predicted last 100 days plus 14 days 70](#_heading=h.38czs75)

# Tables of Equations

[Equation 1 – The PROPHET model 16](#_Toc110693832)

[Equation 2 - Cumulative result of the forward and backward layer outputs 22](#_Toc110693833)

[Equation 3 - ETS(MMN) 23](#_Toc110693834)

[Equation 4 - 1-step differencing 46](#_Toc110693835)

[Equation 5 – AR model 49](#_Toc110693836)

[Equation 6 – MA model 49](#_Toc110693837)

[Equation 7 – ARIMA model 49](#_Toc110693838)

[Equation 8 – The PROPHET Model 49](#_Toc110693839)

[Equation 10 – Random Forest predicton 52](#_Toc110693841)

[Equation 11 – MSE and MAPE 53](#_Toc110693842)

# ABBREVIATION AND NOTATION

|  |  |
| --- | --- |
| ACF | Autocorrelation Function |
| AI | Artificial Intelligence |
| ARIMA | Autoregressive Integrated Moving Average |
| COVID-19 | Coronavirus Disease Of 2019 |
| CRISP-DM | Cross -Industry Standard Process for Data Mining |
| CUDA | Compute Unified Device Architecture |
| CV | Cross-Validation |
| DL | Deep Learning |
| DM | Data Mining |
| EV | Explained Variance |
| GRU | Gated Recurrent Unit |
| GRUs | Gated Recurrent Units |
| JHU CSSE | Johns Hopkins University Center for Systems Science and Engineering |
| LGBM | Light Gradient Boosting Machine |
| LSTM | Long Short-Term Memory |
| MAPE | Mean Absolute Percentage Error |
| ML | Machine Learning |
| MSE | Mean Square Error |
| MSLE | Mean Squared Log Error |
| NCDC | Nigeria Centre For Disease Control |
| PACF | Partial Autocorrelation Function |
| RF | Random Forest |
| RMSLE | Root Mean Squared Log Error |
| RNNs | Recurrent Neural Networks |
| SARIMA | Seasonal Autoregressive Integrated Moving Average |
| SIRD | Susceptible, Infected, Recovered and Dead |

# 

# INTRODUCTION

## BACKGROUND

Over the years, major pandemic and epidemic crises have occasionally affected the entire globe. Since the Spanish Influenza Pandemic of 1918-1920, the Covid-19 pandemic has posed the single greatest danger to worldwide public health and, as a result, the global economy. This gets us to the main topic of the discussion if the world was more prepared in 2020 than in prior years. The next 100 years, society should have made significant advances in forecasting pandemics using historical data, controlling, reducing, and managing pandemics (Morens and Fauci, 2007). Public health experts were concerned even before 2020, stressing the hazards of "hubris, isolationism, and mistrust" (Parmet and Rothstein, 2018)

The history of epidemics began in the 13th century when the Black Death killed tens of millions of people or half the population of western Europe with a particular focus on the elderly and those who were subjected to psychological stressors (DeWitte SN, 2020). The world worst demographic disaster for humanity occurred during the following plague, which struck in the early 1500s. In Mexico, the smallpox outbreak of 1519 resulted in the deaths of about 8 million individuals. However, in terms of epidemic awakening, the same century for the same nation turned devastating. This causes the outbreak of the cocoliztli epidemic in 1545, which results in the deaths of almost 15 million people and roughly 80% of Mexico indigenous population (Acuna-Soto R *et al.*, 2009).

Furthermore, with a pandemic of new coronavirus (also known as Covid-19) triggered by the SARS-CoV-2, this has nearly slowed down the pace of the world. Hence, In December 2019, Wuhan, Hubei Province, China, was the epicenter of this deadly virus (Salim N *et al.,* 2020). Nevertheless, due to the Covid-19 outbreaks progression into a deadly pandemic, the entire world is now protected by it. Early in January, the first death was recorded, indicating the possibility of human-to-human transmission through close or direct contact with an infected person. Johns Hopkins University reports that as of July 12, 2020, there had been a total of 12,745,734 confirmed cases of Covid-19, 556,036 fatalities worldwide, and 7,030,749 recovered cases. With a case-fatality rate of 4.2 percent and a death rate per 100,000 of 41.20, the United States became the most affected nation in the world. Moreover, with 1.68 deaths per 100,000, India has a 2.7 percent Covid-19 case fatality rate (Huang C., *et al* 2020). It was seen that COVID-19 outbreak has an influence on economic production and, as a result, family economics through two-sided pathways reflecting supply and demand adverse shocks. The most important route is health (hence the term "health channel") because sick personnel results in lower economic output. Buyers, on the other hand, may reduce their demand for goods and services that require human connection because of the pandemic (Wren-Lewis, 2020). The second approach is the government's movement restrictions, specifically the introduction of a curfew to contain the pandemic. At the same time, these policies caused havoc on the economy. Hence, due to stay-at-home orders and other mobility limitations, businesses in almost every industry are constrained. Most nations, for example, closed their borders and whole businesses such as restaurants and retail have been shut down for a certain period (Akim et al., 2021).

The globe is constantly battling COVID-19 by implementing various preventative measures, yet there is currently no vaccination for COVID-19. Several studies were published to further the development of vaccines and the mathematical modeling of diagnosis solutions (Huang C *et al.,* 2020). By resolving multiple extraordinarily difficult and sophisticated real-world problems, ML has established itself as a distinct study area in the last ten years. In this study, ML is used to forecast the number of new cases and deaths (Gorriz JM, 2020). There is already literature that has demonstrated daily mortality predictions using deep learning models such as Long Short-Term Memory (LSTM), Bidirectional-LSTM, Convolutional-LSTM, Bidirectional-Conv-LSTM, Gated Recurrent Unit (GRU), and Bidirectional-GRU, the mortality rate and new cases are predicted every day, every three days, and every seven days (Bi-GRU).

Data that has been collected and indexed over time can be modeled using time series models. Real-world practical issues have been effectively modeled, estimated, forecasted, and predicted using time series analysis. An important requirement is the symmetry of the error distribution. However, the assumption of symmetrically distributed error terms is not always accurate in real-world situations (Mahmoudi M.R., *et al* 2017). Therefore, time series models are taken into account based on two-piece distributions in our methodology. The proposed time series models were initially fitted to the historical COVID-19 datasets and include the PROPHET by Facebook and ARIMA time series models. Then, the time series that fits each dataset the best is chosen. The chosen models are applied to forecast the number of confirmed cases and COVID-19 mortality rate in West Africa.

Finally, concerns have been voiced about the lack of participation and fairness in the governance organizations overseeing crisis management and public actions (Rajan D et al., 2020). As a result, the West African government has taken use of government funds set aside to combat the virus. Similarly, amid reports of the virus being used for political goals and the ineffectiveness of public health measures and/or treatment, these concerns were passionately argued in public. In Niger, for example, some suspected that the outbreak was being utilized for financial gain: "politicians are distorting data to portray more good instances in the goal of attracting donor assistance" (IFRC, 2020). People in Cameroon are similarly skeptical of the statistics: "The COVID19 mortality figures are incorrect" (IFRC, 2020). Although the situation differed by country, and our previous analyses demonstrated that routine data could be useful for evaluating public health interventions in West Africa (Zombré D et al., 2017), the quality of health data in this region of the world is frequently debated and called into question (Naudet J-D, 2002). Public response against COVID-19 remains understudied, sometimes overshadowed by arguments about epidemiological statistics (Rice BL et al.,2020). A initial study in Kenya with a sample size of 213 persons revealed the efficiency of the policy package on the epidemic's reproductive rate in early June (Quaife M et al., 2020), however there has been a lack of follow-up study.

## AIMS AND OBJECTIVE

Our aim is to create a system with the use of ML and classical times series that can predict the mortality rate over a consistent timeframe that could be beneficial to authorities in West Africa. To achieve the aim of the study the following objectives have been drawn:

1. Develop a ML and classical time series model that could make accurate predictions of the COVID-19 mortality rate in West Africa.
2. Identifying the most suitable ML technique and time series techniques for prediction.
3. Identifying the features that affect the prediction of COVID-19 mortality rate in West Africa.
4. Develop a champion model which would enable the government of various West Africa nations make feasible short-term policies.

## RESEARCH QUESTIONS

Some research questions have been developed to attain the goals of our thesis:

1. What suitable ML and time series techniques may be utilized to forecast COVID-19 mortality in West Africa? The purpose for the research topic is to undertake a conjunctive literature review and experiment to determine which ML algorithms and time series may be best applied to the supplied data, as well as which approach offers us the greatest results in forecasting COVID-19.
2. What predicted COVID-19 mortality model is more accurate for West Africa government officials to make a feasible short-term policy? The purpose for this research is to perform an experiment to discover best model prediction of the Coronavirus death rate in West Africa and in turn help to make short term policy to tackle the spread.

## DEFINITION OF TERMS

Covid-19 is coronavirus-caused acute respiratory infection in humans, capable of causing severe symptoms and, in some cases, death, particularly in the elderly and those with underlying health issues. It was discovered in China in 2019 and became pandemic in 2020. Machine Learning (ML) is a subfield of artificial intelligence that focuses on developing systems that learn or improve performance depending on the data they ingest.

Classical time series: A time series is a set of well-defined data observations produced by repeated measurements over time. A time series might be tracking the value of retail sales made each month of the year, for example. Although traditional time series forecasting methods are focused on linear correlations, they are sophisticated and perform well.

## THESIS STRUCTURE

This thesis is structured as follows.

**Chapter 2**: Literature review and related work. Forecasting from current studies on the Covid 19 mortality rate will be evaluated critically within the part on literature reviews. The time resolution of the forecasted data will be the focus of this section. This section provides a quick overview of the time series and ML techniques. Based on the literature review that was done, knowledge gaps will be identified.

In this context, ML techniques are thoroughly explained, addressing their advantages and applications. However, there is also a review of various techniques for Covid 19 mortality rate prediction and their classification in detail. Following this, the architecture and working procedures of the proposed techniques and data description are described in detail in **Chapter 3.**

**Chapter 4** reviews the proposed ML and time series framework along with the data preprocessing in detail and presents the Covid 19 mortality rate prediction results obtained in the PYTHON software along with a comparison against different methods.

Finally, **Chapter 5** provides the concluding remarks and insights on future research opportunities.

# 

# LITERATURE REVIEW

This chapter offers the most essential information for readers to comprehend this work. It begins with a brief overview of the time-series analytic methods employed in this study, including ARIMA, Prophet, and ML approaches. The second portion includes an examination of related work, followed by a summary at the end of this chapter.

## TIME-SERIES ANALYSIS

Time-series data are a series of data that are indexed by timestamps or organized in time order. The data are often collected in a sequence with regular gaps in time, for example, at the same time every day. Time-series data are frequently utilized in statistics, economics, and finance for forecasting as well as in many other fields where there is a need to deal with temporal information (Shastri, K. *et al,* 2020). The term "time-series analysis" refers to analytical techniques that identify such temporal components in the data and identify significant statistics and patterns within them. Typically, time-series models are used to fit the data and highlight its peculiarities (Mahmoudi M.R *et al,* 2017).

### ARIMA

ARIMA is an acronym that stands for Autoregressive Integrated Moving Average. It is a well-known time-series model used extensively in statistics, econometrics, and data analytics. It is a generalization of the Autoregressive Moving Average (ARMA) model that introduces differencing into the model to address the problem that ARMA is only suitable for stationary time-series data. The ARIMA model is a type of regression analysis that is widely used to analyse data and generate future predictions based on past values (Ziegel, Box, Jenkins and Reinsel, 1995).

Furthermore, ARIMA is an efficient tool for small dataset, authors like (Fong Simon et al., 2019) proposes a model based on early predictions from a small dataset. The authors of (Domenico Benvenuto et al., 2020) advocated using ARIMA models to anticipate the spread of COVID- 19 over the world, namely ARIMA (1,2,0) and ARIMA (1,2,0). (1,0,4). (Guorong Ding et al., 2020), (Lutfi Bayyurt and Burcu Bayyurt, 2020) developed ARIMA models to forecast cases and mortality in three countries which include Italy, Turkey, and Spain. (Hiteshi Tandon et al., 2020) employ an ARIMA (2,2,2) model to estimate the spread in India. Models for several Italian areas are proposed by (Gaetano Perone, 2020). (Xingde Duan, Xiaolei Zhang, 2020) analyzes two data sets and predicts daily new confirmed cases during a seven-day period using the (ARIMA) model. The authors of (R.K. Singh et al., 2020) investigated 15 nations (the United States, Spain, Italy, France, Germany, the United Kingdom (UK), Turkey, Iran, China, Russia, Brazil, Canada, Belgium, the Netherlands, and Switzerland) to forecast the expansion of the Coronavirus in these countries. Sophisticated prediction models employed ARIMA (Box et al., 2015) to anticipate Coronavirus spread, because ARIMA models provide outcomes in terms of predictive performance. The models allow us to forecast viral behaviour, which might be utilized to make long term plans

### PROPHET

PROPHET is a forecasting technique presented in the publication "Forecasting at Scale" by the Facebook Core Data Science team in 2017 ". It is free and open-source software that is accessible in both Python and R. (S. J. Taylor and B. Letham, 2017). It is "based on an additive model where non-linear trends are combined with annual, weekly, and daily seasonality, including holiday impacts," according to its official website." The PROPHET formulation is as follows:

**y(t) = g(t) + s(t) + h(t) + et**

Equation 1 – The PROPHET model

where y(t) represents the predicted value, g(t) the trend function, s(t) seasonal variations, h(t) the influence of vacations, and is the normally distributed error PROPHET performs best when the data has substantial seasonal influences, and it handles missing data or series changes quite well. PROPHET is precise and quick, with several user-friendly settings for tuning the model and adapting to unique domains to optimize performance. Figure 2.1 shows a visualization of PROPHET 's prediction, with the black dots representing observed values, the dark blue representing projected values, and the light blue representing the upper and lower bounds of the uncertainty interval.

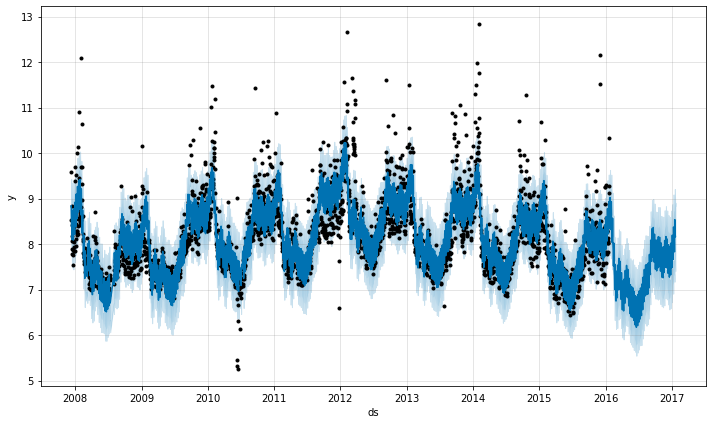


Figure 2.1 – Daily page prediction views from PROPHET (screenshot from PROPHET official website)

PROPHET is incredibly adaptable and has been employed in numerous applications across several disciplines in recent years because to its great accuracy in forecasting/prediction. For instance, (Işil Yenidoan et al., 2018). utilized PROPHET to do Bitcoin predictions. They created a 90-day projection using Bitcoin prices from May 3rd, 2016, to August 30th, 2018. The findings indicated that PROPHET could reach a high R2 value of 94.5%, suggesting that the model can successfully forecast. (Georgia A. and Hristos Tyralis, 2018) forecasted daily stream flow using PROPHET. When the prediction horizon is longer than three days, PROPHET outperforms RF in terms of accuracy. The PROPHET model was used by (Cong Xie et al., 2021) to assess and forecast the daily reported cases of hand, foot, and mouth disease (HFMD). The model detects substantial seasonality and holiday impacts of the disease in the paper, which is very useful in taking measures against the disease. Overall, PROPHET has been used effectively in finance, climate, medicine, and a variety of other scientific domains.

## MACHINE LEARNING

ML is a subset of Artificial Intelligence (AI) that originated from pattern recognition, where data can be arranged for user comprehension. Many applications have recently been created utilizing ML in domains such as healthcare, finance, military equipment, and space. ML is now a highly dynamic and constantly changing discipline. It optimizes computer performance by programming them with data (Taiwo, 2010). It learns the parameters to optimize computer programs by using training data or previous experiences. It can also forecast the future using the data. ML also assists us in developing a mathematical model based on data statistics. The basic goal of ML is that it learns from the fed data without human intervention, that is, it automatically learns from supplied data and produces the required output by searching for trends/patterns in the data (Rao C and Venkat N, 2018).

### RANDOM FOREST

RF is a technique of machine learning that uses a series of decision trees to examine complicated relationships between clinical parameters and deliver excellent classification accuracy (Touw et al., 2013). The RF method constructs a "forest," which is trained via bagging or bootstrap aggregation. Bagging is a meta-algorithm that increases the accuracy of machine learning methods (S. Janitza, 2012).

Although an improved classification model may typically be generated by optimizing only a few parameters, the RF method can be employed without adjustment of algorithm parameters (Breiman L., 2021). The RF is a machine learning method that has showed promise in properly simulating COVID-19 results. To create predictions, the RF method employs an ensemble of weak decision tree classifiers. The RF method has been used to simulate the COVID-19 epidemic. (Tang et al., 2019) developed an RF model from chest CT images to predict the severity of a COVID-19 diagnosis. (Barbosa et al., 2020) used an RF classifier to detect COVID-19 in blood samples. (Gupta et al., 2021) used an RF model to forecast COVID-19 cases in India. (Yesilkanat, 2020) also used the RF model to forecast COVID-19 case numbers at the national level. The authors of (An et al., 2020) attempted to forecast COVID-19 health effects in Korea using demographic and medical data. RF techniques were used in a similar endeavor to predict COVID-19 health outcomes for patients in Wuhan (Wang et al.,2020). A RF model was also used by (Majhi et al., 2020) to forecast COVID-19 case numbers. (Iwendi et al., 2020) set out with the objective of determining the optimal modeling strategy for COVID-19 patient health outcome prediction.

#### LIGHT GBM

LGBM is a gradient boosting system that employs tree-based learning algorithms, which are thought to be extremely strong in terms of computing. It is considered a fast-processing algorithm. Unlike other algorithms, which expand horizontally, the LGBM method climbs vertically, which means it grows leaf-wise whereas other algorithms grow level-wise. LGBM selects the leaf with the greatest growth loss. When expanding the same leaf, it can reduce loss more than a level-wise strategy (Shi, X et al 2019).

## RELATED WORK

The advent of COVID-19, researchers are committed to examining the virus trajectory and making predictions about it using a variety of techniques and models. Additionally, several works compare various time-series models and ML algorithms. This paper will give a summary of the prior endeavor in various aspects of the related work.

Recent research (Ayoobi et al., 2021) concentrated on applying deep learning models to forecast morbidity and mortality rates in two countries which are Iran and Australia. In deep learning, the researchers examined four variations of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) Neural Network Model (LSTM), Bidirectional LSTM (Bi-LSTM), Convolutional-LSTM (Conv-LSTM), Bidirectional-Conv-LSTM (Bi-Conv-LSTM), GRU, Bidirectional-GRU (Bi-GRU), and themselves. Morbidity and mortality rates were forecasted for the following day, three days, and seven days for a total of 100 days for both nations, for a total of six forecast data. The training data for the two nations were got from World Health Organization. According to (Ayoobi et al., 2021), the dataset was divided into two halves, with 70% of each used for training and the remainder preserved for testing. The researchers also validated 20% of the training data. The models were evaluated using four metrics, Mean Squared Log Error (MSLE), Mean Absolute Percent Error (MAPE), Root Mean Squared Log Error (RMSLE), and explained variance (EV). To evaluate the models' performance, the average of the four models was utilized as the error rate. It should be mentioned that (Ayoobi et al., 2021) did not scale the assessment measures before obtaining their averages. As a result, the best model was chosen based on the average rank of each model over the six forecasts. Bi-GRU had the highest average rank of 2.33, while Conv-LSTM had the lowest average rank of 4.83.

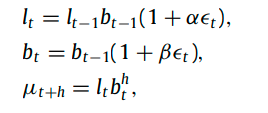
Furthermore, in Predicting the Spread of COVID-19 in Indonesia, (Christophorus Beneditto et al., 2021), data was gathered for confirmed cases, death cases, and recovered patients beginning on March 2nd, 2020, and continuing for a total of 81 days. They then used ARIMA and PROPHET to anticipate cases in a 30-day forecast window and observed that PROPHET had 91 percent accuracy for the whole confirmed case prediction, but ARIMA had only minor differences between the actual true values. Nonetheless, there was insufficient training data to yield clear conclusions or show that PROPHET is superior to ARIMA.

(ArunKumar et al., 2021) obtained a COVID-19 60-day forecast using deep layer RNNs. The models were used to rank the ten countries with the most confirmed cases. A tailored RNN model was recommended for each country to increase performance. As a result, they projected the confirmed cases, recovered cases, and death cases in these nations to compare the accuracy of the Gated Recurrent Units (GRUs) and LSTM units. The data indicated that LSTM did not always outperform GRU since GRU outpaced LSTM in certain countries. A thorough analysis of the reasons for the case patterns in these countries was undertaken, and it was eventually determined that a DL model should be constructed utilizing confirmed, recovered, and death cases as input to boost prediction accuracy.

According to (Shastri et al., 2020), who also explored variants of the LSTM models in predicting morbidity and death rates in India and the United States of America for one month (USA). Bi-LSTM, Conv-LSTM, and Stacked-LSTM were the three LSTM versions employed. The researchers utilized four datasets, the number of COVID-19 cases and the number of COVID-19 death cases datasets for each nation. The COVID-19 dataset for the United States was received from the Center for Disease Control and Prevention, while the COVID-19 dataset for India was collected from the Ministry of Health and Family Welfare, Government of India. (Shastri et al., 2020), employed MAPE as an assessment metric. The Minmax Scaler was used by the researchers to scale the four datasets. Each dataset was utilized for training 80% of the time and tested 20% of the time. The Stacked-LSTM model was trained using two layers. The researchers select the best model based on MAPE Conv-LSTM. All four datasets have a MAPE ranging from 2.0% to 3.3%. Bi-LSTM comes in second with a MAPE range of 3.33% to 6.66%. With a MAPE range of 4.0% to 10.0, the Stacked-LSTM model fared the worst. According to their results, the number of confirmed cases and deaths in both India and the United States would rise in the following month. Nonetheless, based on the best model (Conv-LSTM), the authors projected that by the end of July to mid-August 2020, India might have 1,282,346 cumulative daily confirmed COVID-19 cases and 24,333 total deaths. Similarly, in the United States, there were 58,244,656 cumulative daily cases and 171,806 deaths.

A similar viewpoint may be seen in (Omran et al., 2021), which examined COVID-19 data from three Middle Eastern nations. The authors proposed utilizing GRU and LSTM to anticipate the number of mortality cases and confirmed cases linked to COVID-19 in Egypt, Saudi Arabia, and Kuwait from May 1st to July 6th, 2020. Kaggle was utilized to collect the data. As a result, a total of 12 models were developed, with each country having two models to forecast the number of COVID-19 cases and two models to estimate the rate of mortality caused by COVID-19. Each model has a distinct setup. This may be due to each model being optimized to match their underlying data well. However, (Omran et al., 2021) maintained the number of layers identical across all models, with each model having a one-layer and a two-layer configuration. Each model's number of neurons varied greatly. MAPE, RMSE, and MAE were used as assessment metrics to do a comparative study of all the models for each nation. The error levels of the models were modest, with MAPE values ranging from 0.45 to 5.38. Based on their experiment, (Omran et al., 2021) determined that LSTM had obtained the best performance in the confirmed instances for the three nations, and GRU had achieved the best performance in death cases for Egypt and Kuwait.

So far in this study, researchers have favored deep learning models such as GRU and LSTM, as well as versions of these models. Forecasting timeseries can also be accomplished using mathematical and statistical models. In a recent comprehensive study (Petropoulos et al., 2022), suggested a simple statistical model (ETS(MMN)) to estimate confirmed coronavirus infections and mortality over four months, as well as to evaluate its accuracy and utility. This was achieved by forecasting both variables 10 days ahead of time and repeating the process 12 times. The projection was originally prepared for the entire world, then for three European nations, including Denmark, Norway, and Sweden. The dataset used was obtained from the Johns Hopkins University Centre for Systems Science and Engineering and contains confirmed cases, deaths, and recovered cases by country as of January 22nd, 2020. Following a detailed analysis of the issue under discussion and an overview of exponential smoothing approaches, the authors discovered that the non-seasonal multiplicative error and multiplicative trend exponential smoothing model, typically abbreviated as ETS, fits their requirements (MMN). The model mathematical formulation is given in Equation (3) below.



Equation 3 - ETS(MMN)

*where***:**

🡺 estimation of the level component at period t,

🡺 estimation of the trend component for the same period,

**α, β** 🡺 smoothing parameters for the level and the trend, respectively,

**🡺** forecast horizon,

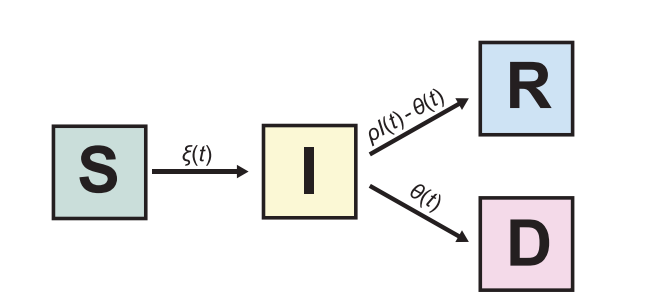
and 🡺 Mean estimate (point forecast) and the error for period t, respectively.

(Petropoulos et al., 2022) used a rolling-origin evaluation method. They began with ten data points available at the time (from January 22nd to January 31st, 2020) and created forecasts and prediction intervals for the next ten days (1st of Feb. 2020 to 10th of Feb. 2020). They increased the in-sample sets to include 20 data points (from January 22nd to February 10th, 2020) on February 11th, 2020 and made estimates for the next 10 days. This method was repeated ten times to get 12 sets of non-overlapping 10-step forecasts spanning four months from February to May 2020. MAPE is the evaluating metric for both models. Each model has 12 APE values since it went through 12 rounds of forecasting. The MAPE for each model was calculated by taking the average of the APE values for each model. The MAPE value for the cumulative death model was 30.5%, whereas the MAPE value for the cumulative confirmed cases model was 42.1%. This was partly due to the extremely high APE values seen in the initial round of forecasting for each model, with values of 132.7% for the mortality model and 384.8% for the confirmed cases.

In Nigeria, (Kabir Abdulmajeed, 2020) forecasted Covid-19 cases on a daily basis using a relative small dataset of Covid-19 cases. The researchers demonstrated how ensemble forecasting models may be used in a data-constrained context. Their ensemble model predicts a lower and upper constraint on the number of covid-19 instances that might occur every day. The COVID-19 dataset was obtained from the Nigeria Centre for Disease Control (NCDC) and consisted of just 22 rows, each indicating a covid-19 case for a single day. The ensemble model was built using three statistical time-series forecasting algorithms: ARIMA, PROPHET from Facebook, and Holt-Winters Exponential smoothing. ARIMA, possibly the most well-known classical time-series model, was the topic of another study (Hiteshi Tandon et al., 2021). The organization predicted the following 20 days' worth of COVID-19 cases in India. The ARIMA model was compared against Linear Trend, Quadratic Trend, S-Curve Trend, Moving Average, Single Exponential, and Double Exponential models. The data utilized came from Johns Hopkins University, which kept a daily record of instances from January 22nd to April 13th, 2020. The ARIMA model performed the best, with a MAPE value of 4.1%.

According to a recent study (Watson et al., 2021), predicting COVID-19 growth cases and mortality is limited due to the virus's novelty, limited data, and dynamic political and societal responses. However, it is critical to the decisions of political leaders, businesses, and individuals dealing with the pandemic. The researchers used a Bayesian time series model and an RF algorithm inside an epidemiological compartmental model to anticipate COVID-19 new cases and deaths for each state in the United States for the next 21 days. The results from New York, Colorado, and West Virginia were chosen for further comparison. The groups model is made up of three major components. First, consider the velocity model for forecasting new confirmed cases. Second, there is the death model, which predicts how many cases will result in death, and third, there is a four-compartment epidemiological model (SIRD), which combines these to produce joint projections of cases, deaths, and recoveries. Within the compartmental model, the case and death models were transformed into transition functions.

The SIRD compartmental model presented by (Watson et al., 2021) used the COVID-19 case and mortality models to estimate the transmission and progression of COVID-19 through U.S. state populations. The SIRD compartmental model was called after the four compartments it divides the population into, S for susceptible, I for infected, R for recovered, and D for dead, as shown in fig 2.3 below. The compartmental model enables combined forecasting of multiple values, which is a significant benefit over many techniques, including most ML and deep learning models, which normally only simulate a single result. The compartmental model also enables COVID-19 cases to be forecasted to be employed as variables in the COVID-19 death model, which would otherwise not offer predictions beyond one day after the observed data.



**Figure 2.3 - The SIRD compartmental model**

Mean Absolute Scaled Error was the evaluation statistic used to determine how successful the SIRD model was (MASE) (Watson et al., 2021). MASE is a scale-free error measure that presents each error as a ratio to the average error. The group calculated MASE by dividing the Mean Absolute Prediction Error (MAPE) by the in-sample MAE of a naïve random walk forecast. A MASE of 1 implies that the predictions in the training were on average equally accurate to the Mean accuracy of a random walk forecast, while MASE values less than 0.5 are regarded good forecasts. Death case projections were more accurate than COVID-19 case estimates because death cases in each state were often less than a MASE value of 1.

**Table 2.1 - RELATED WORKS**

| **Reference** | **Country** | **Horizon** | **Models (best)** | **Morbidity** | **Mortality** | **Evaluation** |
| --- | --- | --- | --- | --- | --- | --- |
| (Watson et al., 2021) | USA | Day | SIRD model (Bayes time series + Random Forest) | Morbidity | Mortality | MASE (best = 0)  Mortality: < 1  Morbidity: < 2.5 |
| (Omran, et al., 2021) | Egypt | Day | GRU (1 layer) | Morbidity |  | MAPE (best = 0%): 0.47  RMSE (best = 0): 670.30  MAE (best = 0): 531.86 |
| (Omran et al., 2021) | Egypt | Day | LSTM (1 layer) |  | Mortality | MAPE (best = 0%): 0.45  RMSE (best = 0): 29.86  MAE (best = 0): 28.59 |
| (Omran et al., 2021) | Kuwait | Day | GRU (2 layers) | Morbidity |  | MAPE (best = 0%): 0.73  RMSE (best = 0): 1150.10  MAE (best = 0): 957.22 |
| (Omran et al., 2021) | Kuwait | Day | LSTM (2 layers) |  | Mortality | MAPE (best = 0%): 1.28  RMSE (best = 0): 11.08  MAE (best = 0): 10.56 |
| (Omran et al., 2021) | Saudi Arabia | Day | LSTM (1 layer) | Morbidity |  | MAPE (best = 0%): 0.07  RMSE (best = 0): 292.78  MAE (best = 0): 259.11 |
| (Omran et al., 2021) | Saudi Arabia | Day | LSTM (1 layer) |  | Mortality | MAPE (best = 0%): 0.14  RMSE (best = 0): 12.36  MAE (best = 0): 8.07 |
| (Ayoobi et al., 2021) | Iran | Day | Bi-GRU | Morbidity |  | avg. of Metrics  (best = 0): 0.60 |
| (Ayoobi et al., 2021) | Iran | Day | Bi-GRU |  | Mortality | avg. of Metrics  (best = 0): 1.09 |
| (Ayoobi et al., 2021) | Australia | Day | LSTM | Morbidity |  | avg. of Metrics  (best = 0): 0.49 |
| (Ayoobi et al., 2021) | Australia | Day | Bi-GRU |  | Mortality | avg. of Metrics  (best = 0): 0.70 |
| (Ayoobi et al., 2021) | Iran | 3-day | Bi-GRU | Morbidity |  | avg. of Metrics  (best = 0): 1.09 |
| (Ayoobi et al., 2021) | Iran | 3-day | Bi-Conv-LSTM |  | Mortality | avg. of Metrics  (best = 0): 0.82 |
| (Ayoobi et al., 2021) | Australia | 3-day | Conv-LSTM | Morbidity |  | avg. of Metrics  (best = 0): 0.66 |
| (Ayoobi et al., 2021) | Australia | 3-day | GRU |  | Mortality | avg. of Metrics  (best = 0): 1.26 |
| (Ayoobi et al., 2021) | Iran | Week | Bi-Conv-LSTM | Morbidity |  | avg. of Metrics  (best = 0): 1.48 |
| (Ayoobi et al., 2021) | Iran | Week | Bi-Conv-LSTM |  | Mortality | avg. of Metrics  (best = 0): 1.23 |
| (Ayoobi et al., 2021) | Australia | Week | Bi-Conv-LSTM | Morbidity |  | avg. of Metrics  (best = 0): 1.09 |
| (Ayoobi et al., 2021) | Australia | Week | LSTM |  | Mortality | avg. of Metrics.  (best = 0): 0.33 |
| (Shastri et al., 2020) | USA | Month | Conv-LSTM | Morbidity |  | MAPE (best = 0%): 2.00 |
| (Shastri et al., 2020) | USA | Month | Conv-LSTM |  | Mortality | MAPE (best = 0%): 2.50 |
| (Shastri et al., 2020) | India | Month | Conv-LSTM | Morbidity |  | MAPE (best = 0%): 2.17 |
| (Shastri et al., 2020) | India | Month | Conv-LSTM |  | Mortality | MAPE (best = 0%): 3.33 |
| (Petropoulos et al., 2022) | Worldwide | 10-day | ETS(MMN) | Morbidity |  | MAPE (best = 0%): 42.1 |
| (Petropoulos et al., 2022) | Worldwide | 10-day | ETS(MMN) |  | Mortality | MAPE (best = 0%): 30.5 |
| (Kabir Abdulmajeed, 2020) | Nigeria | Day | ARIMA + PROPHET + Holt-Winters Exponential smoothing | Morbidity |  | - |
| (Hiteshi Tandon *et al.*, 2021) | India | Day | ARIMA | Morbidity |  | MAPE (best = 0%): 4.1 |

## IDENTIFICATION OF KNOWLEDGE GAPS

Given the detailed Literature reviews on COVID-19 pandemic variables forecasting that have been carried out in the previous section, some knowledge gaps have been observed. These knowledge gaps entail where no studies have been carried out or where more studies need to be made. They are as follows:

1. Very limited research has been carried out on forecasting the long-term impact of COVID-19. (Omran, et al., 2021), (Watson, et al., 2021) and others have made extensive studies on forecasting the mortality and morbidity caused by Covid-19 on daily and weekly scales. However, just a few research on the monthly or yearly impact have been conducted.
2. A very limited amount of data has been used by most researchers in making their forecasts. Many studies were carried out on COVID-19 and had limited historical data. Models are only as good as their underlying data and given the limited data used by most researchers, more complex relationships and trends would not have been obtained by their model.
3. Geographically, few studies have been carried out concerning Covid-19 forecasting in West Africa and her countries. Asia, Europe, and the Americas have been the focal point of study by researchers for Covid-19 forecasting. This may be due to the availability of more resources in these areas.
4. No test has been carried out to see how classical time-series models fair with the more invoked deep learning models. In the previous section, many researchers favored deep learning models in carrying out their forecasts. Deep learning models are known to need a relatively large amount of data compared to classical time-series models.
5. Limited testing was carried out in most of the research. While most researchers in the previous section favored the hold-out style (train and test) of validating their models, it is not a very robust choice. Other methods need to be explored to see clearly how the forecasting models are performing.

**RESEARCH FOCUS**

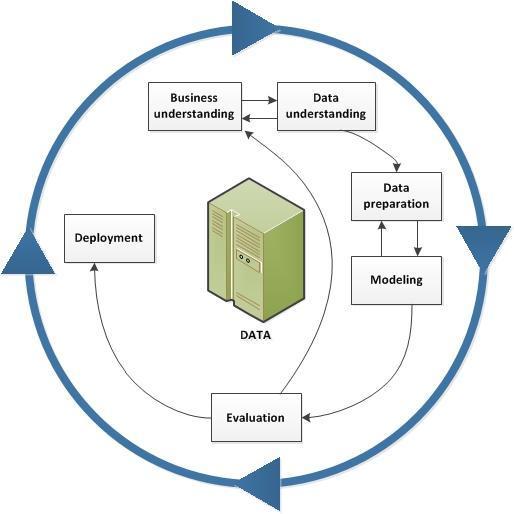
To begin, the study will perform thorough research and comparisons of several journals that will be utilized in estimating the long and short term impact of Covid 19 mortality rate in West Africa. Nonetheless, two classical time series models (ARIMA and PROPHET) and two ML models (RF and LGBM) will be used. As a result, building a champion model with less error makes it easier for people and governments to devise a short/long-term policy to combat disease transmission. Second, this study will obtain data from JHU CSSE, where the mortality rate is updated daily, resulting in a more accurate forecast and model. Third, because just a few studies have been conducted in African countries, this study will concentrate on West African countries. Fourth, in contrast to the deep learning model, which was chosen by many researchers, this study would employ traditional time series models such as ARIMA and PROPHET, which provide good predictions for a smaller dataset. Finally, other methodologies will be investigated to see how forecasting model performance. In this study, accumulated rolling origin cross validation would be used.

# 

# RESEARCH METHOD

## INTRODUCTION TO CRISP-DM

The most widely used framework for carrying out data science initiatives is CRISP-DM, which was first published in 1999. It offers a natural explanation of the data science life cycle (the process utilized in initiatives with a data focus) (Alliance A. et al, 2021). However, this project execution strategy that is task-focused ignores teamwork and communication problems. CRISP-DM should therefore be used in conjunction with other team coordination frameworks. Recent technological advancements, data can be collected, stored, and processed massive amounts. Data scientists utilize analytical and mathematical models to extract useful insights from data using data-driven techniques like Data Mining (DM) and ML. CRISP-DM consists of six phases, Figure 3.1 shows how these six steps and how they are sequenced in a typical data analytics project.



**Figure 3.1 - The CRISP-DM process model (Alliance A. et al., 2021)**

Every project, team, and organization is unique. So, to evaluate CRISP-DM for the next project, first review its key concepts. Then, assess its strengths and weaknesses. Finally, consider some key phases for its use.

**Table 3.1 - CRISP-DM process description (Alliance A. et al 2021)**

|  |  |
| --- | --- |
| Phases | Description |
| Business understanding | What does the business require? |
| Data understanding | What data is required? Is it clean? |
| Data preparation | How is the data organized for modeling? |
| Modeling | What modeling approaches are suitable? |
| Evaluation | What fits the business objectives the best? |
| Deployment | How do stakeholders get their hands on the results? |

CRISP-DM will be used as the base template for each phase (Alliance et al., 2021).

## Business Understanding

The initial step in this phase is to choose the research objectives, which were already discussed in the thesis introduction. The second step is to comprehend and access the existing situation. In the thesis literature review portion analyzed some of the pertinent works that have been done in the field, and in the final section of the literature review, the study objectives were established.

### Determine Business objectives

#### Business objectives

Creating a forecasting model that can accurately predict mortality rate in West Africa. This will help the many hospitals and diseases control agencies in the country, allowing the agencies to have accurate predictions of the COVID-19 mortality rate in any country in West Africa. Nevertheless, choosing the best ML approach for prediction and time series techniques for statistical analysis will aid in better decision making in a long or short run.

#### Business Success Criteria

The degree to which the model can predict the Covid - 19 mortality rate in West Africa with accuracy will serve as a barometer for the success of the thesis. The disease authorities can use this information to plan manufacturing, maintenance, and operation in their various countries. Forecasting Covid -19 Mortality rate in West Africa will help in the reduction of death rate in the countries.

### Research Goals

The goal of this research thesis is to make a weekly forecast on the number of deaths caused by COVID-19 in West Africa based on historic records of weekly COVID-19 deaths in the same region. The historic data will be explored to find insights and patterns in covid-19 fatality. Comparison of two statistical models (ARIMA and PROPHET) and two ML models (RF and LGBM) will be carried out to select the best model. Based on the forecasted values the models will be evaluated using MAE to select the best performing model. The goals can be summarized as below:

1. Exploratory data analysis
2. Using cross-validation
   1. Generate the statistical models - ARIMA and PROPHET using the train data.
   2. Generate the ML models – RF and LGBTM using the train data.
   3. Validate the models using the test data.

### Produce Project Plan

The project plan will lay out the stages to be taken to complete the research method, and the second section will address the basic tools and approaches. The project plan will also outline the activities that will take place over the remainder of the study process.

#### Project Plan

The data used in this experiment is available in the repository from the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE). It collects information from several sources and has been updated daily since January 22, 2020. It gives reliable information on case reports in the United States and many other nations, as well as a variety of options. In addition, the dataset includes a time-series summary of COVID-19 instances, which is ideal for our project research. Finally, a description of the selected data is provided, and data pre-processing will be discussed in the rest of this section.

#### Initial Assessment of tools and techniques

Since the tools and procedures selected could have an effect on the entire project, it is crucial to evaluate them early in the process. Python and R are two of the often-used languages for data mining projects. The both languages are open source, anyone can download and use them. The language delivers both strengths and limitations through data-driven innovation, AI and ML. These languages are compatible with data science tasks like automation and data manipulation as well as business analysis and large-scale data exploration. They are similar in many aspects. One of the primary differences between the two languages is that Python is a general-purpose language while R has its roots in statistical analysis which is one of the main distinctions between the two languages. Python will be used in this project, as the research is predominantly focused on statistical models and forecasting techniques. Forecast package in Python is used to forecast time series models several summary measurements of forecast accuracy are included in the forecast package. Nevertheless, Packages like Matplotlib and plotly can be used together to create interactive plots in Python. Numpy & Pandas for data wrangling.

Python Streamlit module offers a web application framework that may be used to build reactive web applications that can be published online. For the entire data mining process flow, including data purification, wrangling, modeling, analyzing, visualizing, and deployment. Hence Python offers a strong basis. Exploratory data analysis can reveal underlying patterns in the data, improving our comprehension of the information and enabling the development of dynamic models. Jupyter Notebook will be used for data exploration. It is a piece of interactive business intelligence software for data visualization. Large volumes of data can be processed with Notebook. It is quicker and offers more options for data display. This data can be viewed from every viewpoint with Jupyter Notebook because there are no constraints on the number of data points, rows, or sizes.

## Data Understanding

### Data Collection

The data needed for this study is the covid-19 death records in West Africa. Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE) provides a Github repository containing daily records of COVID-19 variables like cases, deaths and recoveries which is used to run their covid-19 dashboard. The datasets contain daily records of COVID-19 data for 199 countries. Nevertheless, all west African countries are included. These datasets are updated daily to account for the new cases recorded [csse\_covid\_19\_daily\_reports](https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_daily_reports).

.

#### Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE)

Modern, complicated issues necessitate the creation of a new scientific method for analyzing and addressing them, one that emphasizes the links between many fields. The Center for Systems Science and Engineering (CSSE) mission is to advance the science of interconnectivity. CSSE, which is housed in the Department of Civil and Systems Engineering at Johns Hopkins, takes an interdisciplinary approach to modeling, analyzing, and optimizing systems of local, national, and global relevance. Medicine, health care delivery, national infrastructure, information security, disaster response, and education are examples of these. CSSE draws on the experience of researchers from the schools of Medicine, Public Health, Nursing, Arts and Sciences, Business, and Education, as well as JHU Applied Physics Laboratory, which is already one of the nation top centers of systems engineering.

### Data Description

Under this section will investigate the dataset and identify attributes in the dataset. Thetable 3.2 below gives a better understanding of the dataset, the attributes, types and description.

**Table 3.2 - Datasets Description**

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Attribute | Type | Description |
| Timeseries covid19 confirmed global.csv | Province/State | String | state or province for countries that are listed by states or provinces |
| Country/Region | String | Country in focus |
| Lat | Float | Latitude of the state or country |
| Long | Float | Longitude of the state or country |
| daily records | Int | A progressive list of time-series columns each representing the record of the confirmed cases for a day starting from 22nd of January 2020 |
| Timeseries covid19 deaths global.csv | Province/State | String | state or province for countries that are listed by states or provinces |
| Country/Region | String | Country in focus |
| Lat | Float | Latitude of the state or country |
| Long | Float | Longitude of the state or country |
| daily records | Int | A progressive list of time-series columns each representing the record of the confirmed cases for a day starting from 22nd of January 2020 |

### Data Exploration

This study gives a better understanding through data exploration process of West Africa countries and its confirmed death rate. Understanding the data can help identify how different factors are affecting the death rate. The **time\_series\_covid19\_confirmed\_global data** set is aggregated to obtain the total daily confirmed cases for the world and **time\_series\_covid19\_deaths\_global dataset** obtains the total daily confirmed cases which will be detailed in the following section. The aggregated dataset as seen in fig 3.3 starts from 22nd January 2020 to 2nd July 2022. The dataset has five columns, and the rows will continue to add each day. Nigeria being the largest economy and country by populatuion in Africa also has the highest mortality rate as seen in fig 3.3. with total deaths of 3143. The second and the third country being Senegal and Ghana respectively had total death tolls of 1988 and 1445 respectively. Ten West African countries had a death toll below 500 with Sierra Leone having the lowest at 125 death case as shown in fig 3.3.

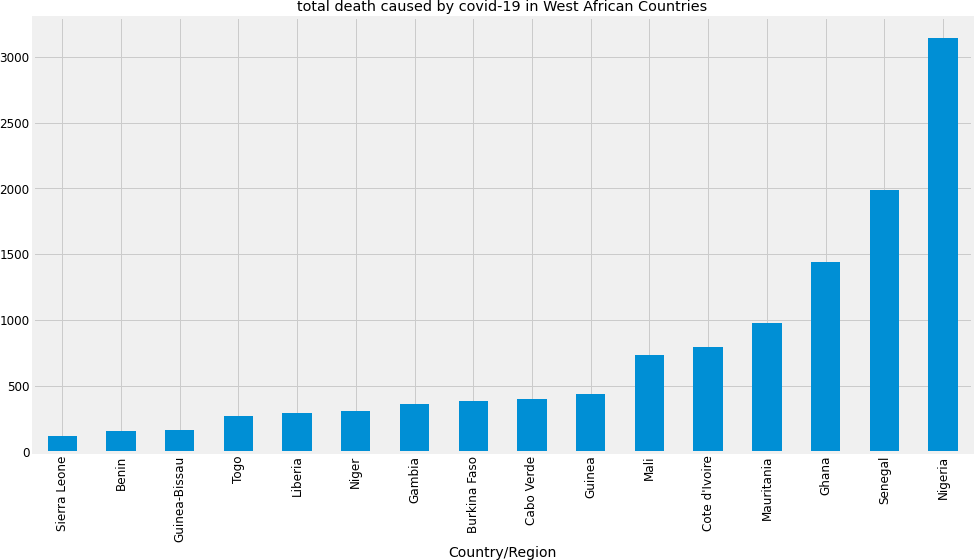


Figure 3.3 – Bar Chart Showing COVID-19 Mortality Rate in West Africa from 1st of March 2020 to 2nd of July 2022

**Sample Dataset of COVID-19 Death Cases from 1st August 2020 to 21st August 2020**

|  |  |
| --- | --- |
| **Date** | **Accumulated Covid-19 Death Rate in West Africa** |
| 01/08/2020 | 12 |
| 02/08/2020 | 5 |
| 03/08/2020 | 22 |
| 04/08/2020 | 30 |
| 05/08/2020 | 35 |
| 06/08/2020 | 9 |
| 07/08/2020 | 24 |
| 08/08/2020 | 16 |
| 09/08/2020 | 21 |
| 10/08/2020 | 12 |
| 11/08/2020 | 19 |
| 12/08/2020 | 13 |
| 13/08/2020 | 27 |
| 14/08/2020 | 20 |
| 15/08/2020 | 22 |
| 16/08/2020 | 17 |
| 17/08/2020 | 14 |
| 18/08/2020 | 14 |
| 19/08/2020 | 31 |
| 20/08/2020 | 21 |
| 21/08/2020 | 8 |

The death rate in the western part of Africa is depicted in the graph in Fig. 3.3 through data exploration process. Some nations have a low mortality rate; however, Nigeria has the highest mortality rate in West Africa.

### Verify Data Quality

JHU CSSE in their COVID-19 dataset GitHub repository listed all sources which they aggregate their data from and are mostly official government sources. The full list can be checked in their GitHub repository - [csse\_covid\_19\_daily\_reports](https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_daily_reports).

## Data Preparation

### Data Selection

The row entries containing West African states data are selected from the cases dataset and death dataset. The columns needed are only the time-series record which started from 22nd of January 2020 keeps updating daily with real time. Hence, from the cases dataset, West Africa did not have any cases till 28th of February 2020. Nevertheless, the column selection started from 28th of February and keeps updating daily.

### Data Cleaning

Given the forecast will be weekly, the last time series column would be the last daily entry that ends the week, and the others will be discarded. This will be useful during internal model development and testing but will not be useful after deployment. However, the programme will fetch the time series data at the end of every week. Also, There are no missing values in the selected columns

### Data Construction

#### Times Series

The time series data available for both cases and deaths are daily records. The data are Aggregated, to sum up all records for each week to produce weekly records that can be used for modeling. The time series reports on both datasets are cumulative i.e the total number of cases (same as death) is the total cases for that country/Region at that time. a 1-step differencing was done to get only the new cases for that day. 1-step differencing is done by subtracting the new time series record (t) from the previous time-series record (t-1) for all-time series record (tn)

**t = t -**

Equation 4 - 1-step differencing

The daily cumulative number of confirmed cases may prove useful when modelling with ML models, but it was not used for the statistical models.

#### Machine Learning Models

The data for ML models is constructed for modelling in stages. Firstly, the daily number of confirmed cases and deaths data was used when modelling with ML models. Since the number of confirmed deaths is influenced by the number of confirmed cases, it is useful to add the confirmed cases data as a feature in the ML models. Secondly, both datasets summed daily record by the West African countries. Hence, having West African daily record. Thirdly, the first order differencing was done on both data to get the new cases instead of the cumulative cases and also the two dataset was merged with each taking one column. Fourthly, a 1 day and 7days lagged result for both and death column was made and merged with the data as they are useful important features during experimentation. Lastly, the deaths column is separated out to be used as the dependent variable (y) while the other four columns are the independent variables(X)

## Modelling

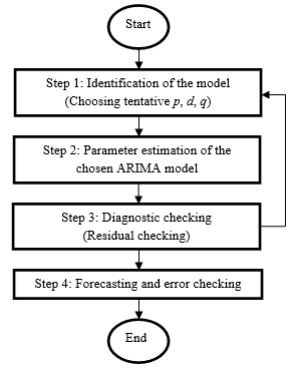
### Model Selection

The primarily focus of this research is applying statistical models and ML approaches to forecast the death rate, as it is a time series problem. The ARIMA and PROPHET, a more contemporary model for forecasting time series data that Facebook announced in 2018, are two of the most common statistical models used for time series analysis. This research will look into the various models under consideration, comprehend how they operate and see how they may be applied to forecast the Covid 19 Mortality rate.

#### Model Study

##### I ARIMA

The autocorrelation function (ACF) and partial autocorrelation function (PACF) must be analyzed while creating the ARIMA model. The study must then generate an estimation of the parameter for the specified ARIMA model. Diagnostics checks must be created in order to validate the model. The residual is the discrepancy between the observed and estimated quantities of interest (sample mean). The residual should be uncorrelated, having a Mean and variance of zero. The forecasting and error checking stages can then be carried out. Hence, fig 3.20 gives a better understanding of the ARIMA process.



***Figure 3.20 - Statistical forecasting procedure***

Box and Jenkins pioneered the ARIMA technique, and ARIMA models are sometimes referred to as Box-Jenkins models. The ARIMA procedure broad transfer function model was described by (Box and Tiao1975). When additional time series are included as input variables in an ARIMA model, the model is referred to as an ARIMAX model. (Pankratz,1991) refers to the ARIMAX model as dynamic regression. Firstly, this study describes the derivation of the Autoregressive (AR) method. Thus, the model is in the form of a stochastic difference equation. The notation AR (p) indicates an autoregressive model of order p. The AR (p) model is defined as:

***(AR)***

Equation 5 – AR model

In a regression-like model, a moving average model uses prior prediction errors rather than historical values of the forecast variable. The Moving Average (MA) can be represented as:

MA

Equation 6 – MA model

A non-seasonal ARIMA model is created by combining differencing with autoregression and a moving average model. The ARIMA model can be represented as:

(ARIMA)

Equation 7 – ARIMA model

where the differenced series, an ARIMA model is represented as ARIMA(p,d,q).

*p*: order of the autoregressive part

*d*: degree of first differencing involved

*q*: order of the moving average part

##### II. PROPHET

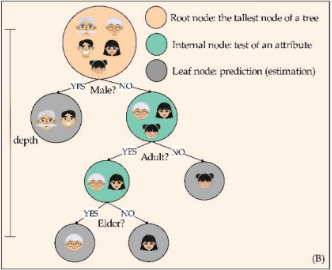
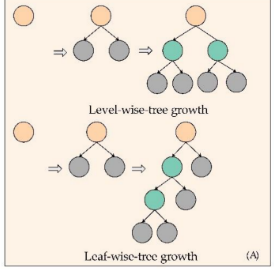
The PROPHET model was introduced by Facebook in 2018, S. J. Taylor & Letham originally created the model for forecasting daily data with weekly and yearly seasonality, plus holiday effects. Later the package was extended to cover more seasonal data types. The model can be represented as:

Equation 8 – The PROPHET Model

Where represents the linear trend or growth term, describes the seasonal patterns and represents the holiday effect. describes the white noise. PROPHET allows for multivariant analysis and defines holidays. The package also has a built-in function to use holidays based on the country code. The capability of PROPHET to factor in trend, seasonal patterns, and holiday make PROPHET an ideal choice to use for forecasting seeing a clear-up trend, so many problems, the performance of RF is very similar to boosting, and they are simpler to train and tune. Consequently, RF are popular and are implemented in a variety of packages. Bagging or bootstrap aggregation is a technique for reducing the variance of an estimated prediction seasonal usage pattern, and the effects of holidays.

##### III LightGBM

Microsoft first suggested the open-source framework known as the LGBM paradigm. It is a decision-tree-based technique that creates the mapping relationship between the inputs and outputs by segmenting the parameters in the input layer into separate portions (GuolinKe *et al* 2012). The leaf-wise-tree growth strategy is thus more avaricious than the level-wise-tree growth technique. When the tree depth is the same, the leaf-wise-tree algorithms tree nodes are typically less than those of the level-wise-tree approach.



**Figure 3.21 - Characteristics of Lightgbm Figure 3.22 - Growth Approach of Lightgbm**

Figure 3.22 provides a typical illustration of the leaf-wise-tree growth for breast cancer patients using age and gender as input parameters (Taifeng Wang *et al.,* 2013). The leaf-wise tree growth approach can drastically minimize the number of tree nodes when compared to the level-wise tree growth technique, which can significantly speed up training when the dataset is huge. The level-wise tree growth algorithm is comparatively more stable, but the leaf-wise tree growth algorithm tends to over-fit when the dataset is limited due to its greedier approach.

##### IV Random Forest

RF are a significant variation of bagging in which a huge sample of de-correlated trees is built and then averaged (Breiman, 2001). On the job. Bagging appears to perform particularly well for high-variance, low-bias techniques like trees. The same regression tree is fitted several times to bootstrap sampled copies of the training data for regression and the outcome is averaged. For categorization, a group of trees voted on the anticipated class.

Trees benefit immensely from averaging since they are typically loud. Furthermore, because each bagged tree is identically distributed (id), the expectation of an average of B such trees is the same as the expectation of any one of them (Amit and Geman 1997). This indicates that the bias of bagged trees is the same as that of individual trees, and the only way to improve is to reduce variation. This contrasts with boosting, where the trees are grown in an adaptive way to remove bias, and hence are not i.d. An average of B i.i.d. random variables, each with variance σ2, has variance σ2. If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pairwise correlation ρ, the variance of the average is:

Equation 9

As B increases, the second term disappears, but the first remains, and hence the size of the correlation of pairs of bagged trees limits the benefits of averaging. The idea in RF (eqn 3.4) is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much. This is achieved in the tree-growing process through random selection of the input variables. Specifically, when growing a tree on a bootstrapped dataset:

Before each split, select m ≤ p of the input variables at random as candidates for splitting

Typically values for m are √p or even as low as 1.

After B such trees {T(x; Θb)} are grown, the RF (regression) predictor is:

Equation 10 – Random Forest prediction

RF are popular. Leo Breiman’s collaborator Adele Cutler maintains a RF website where the software is freely available, with more than 3000 downloads reported by 2002(Friedman and Hall, 2003).

### Generate Train and Test Data

In ML, the train/test split divides the data randomly because there is no dependence between observations. This is not true for time series data. The datasets bottom values is used for testing, and the remaining values for training. Data in time series have a built-in temporal ordering. The model’s performance is tested against the test data once they have been trained using the train data. The time series is divided at a ratio of 0.9; 90% of the data will be used for model training and 10% for model performance validation. To ensure that the time series are split in sequence, the models are split using the initial time split() function.

## Evaluation

Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE) are the evaluation metrics employed in this project. The following mathematical expressions describe them:

MSE =

MAPE =

where is the predicted value and yi is the actual value.

is the actual value and is the forecast value

Equation 11 – MSE and MAPE

These two metrics can represent the accuracy of models well and can be used to evaluate their performance. For each model, these two metrics is calculated for the last 50 predicted values in the results and compare with each other, determining the model with best accuracy. Other evaluation metrics (e.g., R2, F-score) can also be used in this scenario, but to the simplicity of evaluation MSE will only be applied.

## CROSS-VALIDATION

Cross-validation is a statistical tool for evaluating robust models. Cross-validation gives the researcher a good estimate of the error of a model by training the model on different portions of the dataset and providing an average error. This study will employ the rolling origin cross validation. Rolling origin is a model evaluation technique where the forecast origin rolls forward in time and forecasts are produced from each origin (Svetunkov et al, 2018). This approach begins with a limited selection of data for training purposes (without shuffling), forecasts later data points, and then calculates the error for the forecasted data points. Following that, the same forecasted data points are incorporated in the next training dataset and following data points are forecasted.

For example, if the data set contains 12 months of data, 3 months can be allocated to the training set and 1 month to the validation set. The origin is then rolled forward a month, and the model is retrained and evaluated in the next month. The process is repeated until the last month, giving a total of 9 test errors produced over the year. The average of the errors gives the overall error estimate. The training set can be set to accumulate over time (i.e., increasing in-sample size). fig 4.2. For example, by month 12, the training set size maybe 11 months with a single month as a validation set. Alternatively, the training set can be kept constant (i.e., constant in-sample size) fig 4.3. below. In this work, the accumulated rolling-origin is used, the origin is rolled forward 14 days at a time with new models built at each origin. The "Timeseries Split" class in the python "Sci-Kit Learn" package was used to develop the rolling origin cross-validation strategy with the accumulative training set used in this work.

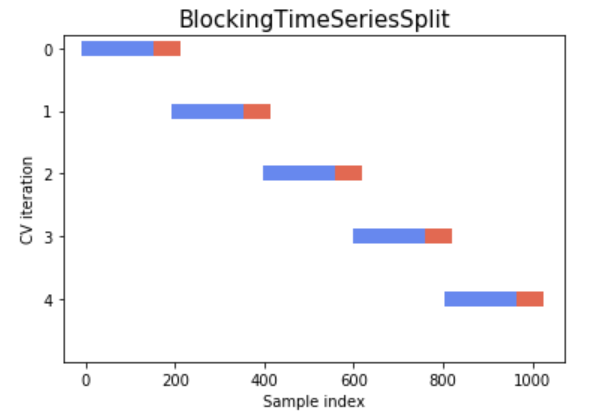
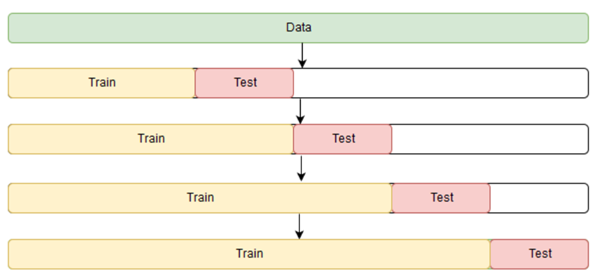


Figure 4.2 - Accumulated rolling-origin Figure 4.3 - Rolling-origin constant

(McDermott et al, 2019) (McDermott et al, 2019)

## Deployment

This is the procedure for utilizing the information from the aforementioned stages to enhance decision-making and enhance the client experience. Forecasting models is created for mortality rate in west Africa. The prediction models generated can be used to forecast the mortality rate of a country in western region of Africa.

Furthermore, to showcase the work done and to make available the prediction model making weekly covid-19 deaths prediction to stakeholders, a webapp was built [West Africa Covid-19 Mortality Prediction Web App](https://amade2233-myproject-deploy-vljy3v.streamlitapp.com/) with a python open-source app framework, Streamlit. Streamlit is an easy-to-use library to create simple web apps for reporting by turning python scripts into webapps "https://streamlit.io". With Streamlit little frontend web development knowledge is needed as the library abstracts it all away as python functions that can be called to perform the same task. Streamlit also provides a means to deploy the built webapp for free on their cloud infrastructure which makes it a suitable choice for this task. The ARIMA model which is the best model ascertained after this research work will be deployed using Streamlit. The webapp will also make predictions weekly after fetching new data from John Hopkins repo and retraining the ARIMA model with the same parameters used in this research work. The procedures are as follows.

Every Sundays:

1. Download the latest covid-19 deaths result from John Hopkins
2. Retrain the model including the new datapoints
3. Make a forecast for each day of the upcoming week.

# 

# ANALYSIS OF FINDINGS

### ARIMA MODEL

**PROCEDURE**

The ARIMA methods were carried out in stages. To begin, two weeks of data were utilized for validation. As a result, the ARIMA differencing term was calculated. Also "d" on the remaining data using the pmdarima library's ndiffs class. Second, using the AUTOARIMA class, the optimal parameter for p and q was obtained. The model testing, data was split using the accumulated rolling – origin cross validation using 15 split. As a result, the accumulated rolling origin cross validation was employed for model testing. Finally, the ARIMA model with the estimated p, d, and q was trained and the MSE was determined using the test data and saved in a list. The model's error was calculated using the average MSE value from the MSE list. The model is evaluated on held-out data to see how well it would perform with actual data, and the MSE is determined. As a result, if the error becomes sufficiently, the model will no longer be stable to real-world data.

**Hyperparameter values for ARIMA (ARIMA order)**

|  |  |
| --- | --- |
| Hyperparameter | Value |
| P | 1 |
| D | 1 |
| Q | 3 |

Chart, histogram

Description automatically generated

Figure 4.6 - Actual vs Forecasted (ARIMA) daily Covid-19 mortality rate from March 2020 to June 2022

Chart

Description automatically generated

Figure 4.7 - Actual vs Forecasted (ARIMA) daily Covid-19 mortality rate for the last 200 days (Dec 2021 – Jul 2022)

Chart

Description automatically generated

Figure 4.8 – ARIMA 14 days forecast (19th of Jun – 2nd of Jul 2022)

Figure 4.6 depicts a line chart visualization of the daily COVID-19 mortality rate from March 2020 to June 2022. This depiction in Fig 4.6 reveals that the model was able to match the underlying data extremely well and disregard noise, with the blue trend representing the real covid-19 mortality rate and the red trend representing the anticipated result. Nonetheless, fig. 4.7 shows a similar outcome of high performance in which the model was able to fit in well with dates extending from December 2021 to July 2022 (200 days) the actual and projected trend line was upward sloping and reached its apex during the months of January and February 2022.However, the mortality rate was at its lowest in Fig 4.7 from the month of April 2022 to June 2022.

Figure 4.8 also shows a future prediction using the ARIMA model with a 14-day forecast (19th of Jun – 2nd of Jul 2022). As a result, a trend pattern identical to the 14-day actual and anticipated trend was achieved. As a result, we could observe that the ARIMA model had a decent forecast that was close to the actual mortality rate, and there was no overfitting.

**Table 4.1 - Summary table for ARIMA times series model**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ARIMA | Cross-validation | Held out data | Mean | STD | Overfit |
| MSE | 38.3406 | 1.1557 | 38.3406 | 44.1156 | False |

The MSE, which assesses the amount of error in the model, is summarized in Table 4.1. This approach calculates the average squared difference between the actual and forecasted death rates. The findings in Table 4.1 also demonstrate that there is no overfitting using the ARIMA.

### PROPHET MODEL

**PROCEDURE**

The PROPHET model went through several stages. Initially, a two-week period of data was held for model validation. Using cumulative rolling origin cross validation, the data was divided into training and testing, 15 split rolling origin cross validation was used for the model testing. Because the MSE was calculated using the test data, it was saved in a list. Finally, the average MSE value from the list was used to determine the model error. As a result, the model is evaluated on held-out data to see how well it would perform with actual data, and the MSE is determined. If the error increases significantly, the model will no longer be stable with real-world data.

Chart, line chart

Description automatically generated

Figure 4.11 - Trends plot using PROPHET model

Chart, histogram

Description automatically generated

Figure 4.12 - Actual vs Forecasted (PROPHET) daily Covid-19 mortality rate from Mar 2020 to Jun 2022

Chart, histogram

Description automatically generated

Figure 4.13 Actual vs Forecasted (PROPHET) daily Covid-19 mortality rate for the last 200 days (Dec 2021 – Jul 2022)

Chart

Description automatically generated

Figure 4.14 - PROPHET 14 days forecast (19th of Jun – 2nd of Jul 2022)

A line graph of the PROPHET model is shown in Figure 4.11. This trend line provides a quick overview of the data from March 2020 to June 2022. As a result, the trend graph indicates an upward sloping increase in the death rate from March 2020 to September 2021, when the mortality rate peaked. However, following September 2021, when Covid 19 death was at its peak, there was a distinct downward slope considerable decline in the mortality rate caused by Covid 19, and the death rate was at its lowest in June 2022, when there was no death. The remarkable fall in the death rate here might be attributed to increased knowledge and strong steps to combat the disease. Figure 4.11 also depicts the weekly chart, which reveals that Covid 19 was at its highest on Tuesdays. As a result, most people die of disease on Tuesdays, with fewer deaths on Mondays, Thursdays, and Fridays. The annual trend in Fig. 4.11 demonstrates that September has the highest Covid-19 mortality rate among West African nations.

In addition, figure 4.12 above depicts a line chart depiction of the daily COVID 19 mortality rate from March 2020 to June 2022. The actual line mortality rate trend is shown in blue, while the forecasted line trend is shown in red. Looking closely at the visualisation, it is clear that the actual and anticipated data did not blend in properly. As a result, there are wide uncertainty intervals that do not correspond to the actual data point. Nonetheless, because the anticipated trend line has a negative value, we may conclude that the PROPHET model was not a good model. Figure 4.13, which depicts the COVID-19 mortality rate from December 2021 to July 2022, provides a deeper look at more recent data. As a result, the PROPHET model plainly did not fit the data well and predicted negative values beginning in APRIL 2022, when the trend continued to steeply decline.

Finally, the PROPHET 14-day prediction from the 19th of June 2022 to the 2nd of July 2022 was projected, and the outcome, as shown in Fig 4.14, demonstrates that the PROPHET model anticipated zero death instances for the majority of days, which contrasts with the actual data.

**Table 4.2 - Summary Table for PROPHET Times Series Model**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROPHET** | Cross-validation | Held out data | Mean | STD | Overfit |
| MSE | 97.4101 | 16.0341 | 97.4101 | 75.0071 | False |

The MSE, which measures the amount of error in the model, is summarized in Table 4.2. As a result, PROPHET numbers for Cross validation, Held-out data, Mean, and Standard deviation is shown above.

## MACHINE LEARNING

**PROCEDURE**

The ML technique was broken down into phases using the Scikit-Learn package. The data from the previous weeks was saved for validation. The ideal hyperparameter values were chosen via a random search with cumulative rolling origin cross validation. As a result, for the selected hyperparameter, a list of values was selected, and a 15 split rolling origin cross validation using two weeks of data test set, A,B,C, and D, was performed. Nevertheless, the model is evaluated on held-out data to see how well it would perform with actual data, and the MSE is determined. If the error becomes sufficiently, the model will no longer be stable with real-world data.

### Random Forest

The hyper-parameters used to tune the model and their final values are as follows:

**Table 4.3 - RF hyper-parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Description | Value |
| n\_estimators | The number of decision trees in the forest | 1000 |
| max\_depth | The maximum depth each tree is allowed to reach even if its leaves are not pure yet | 10 |

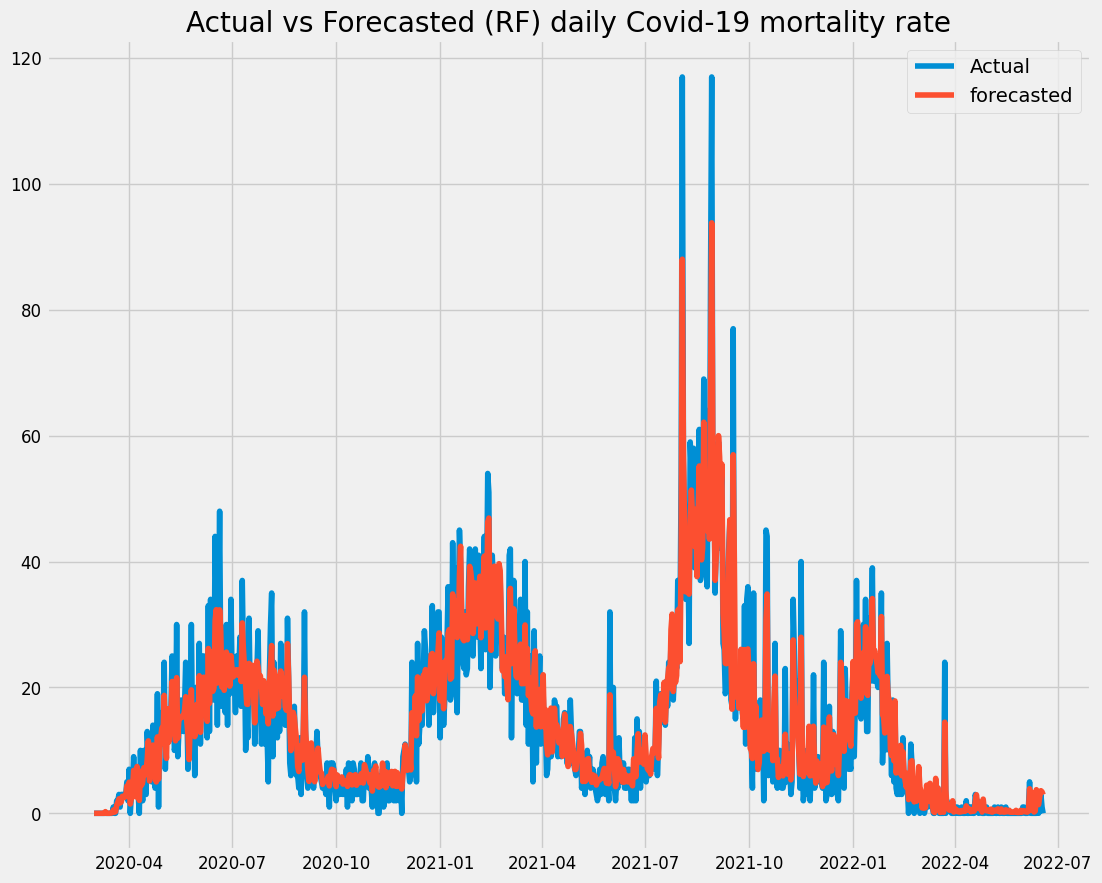


Figure 4.16 - Actual vs Forecasted (RF) daily Covid-19 mortality rate from 1st Mar 2020 to 18th Jun 2022

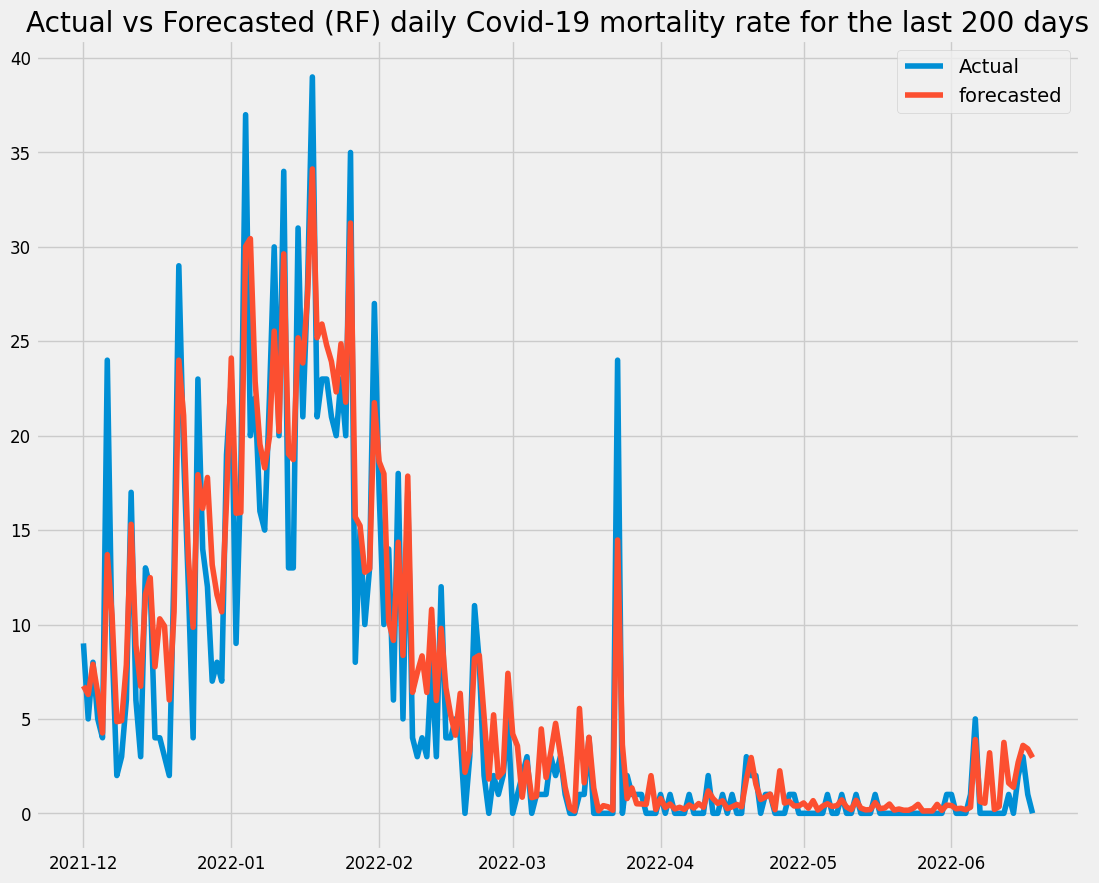


Figure 4.17 - Actual vs Forecasted (RF) daily Covid-19 mortality rate for the last 200 days (Dec 2021 – Jul 2022)

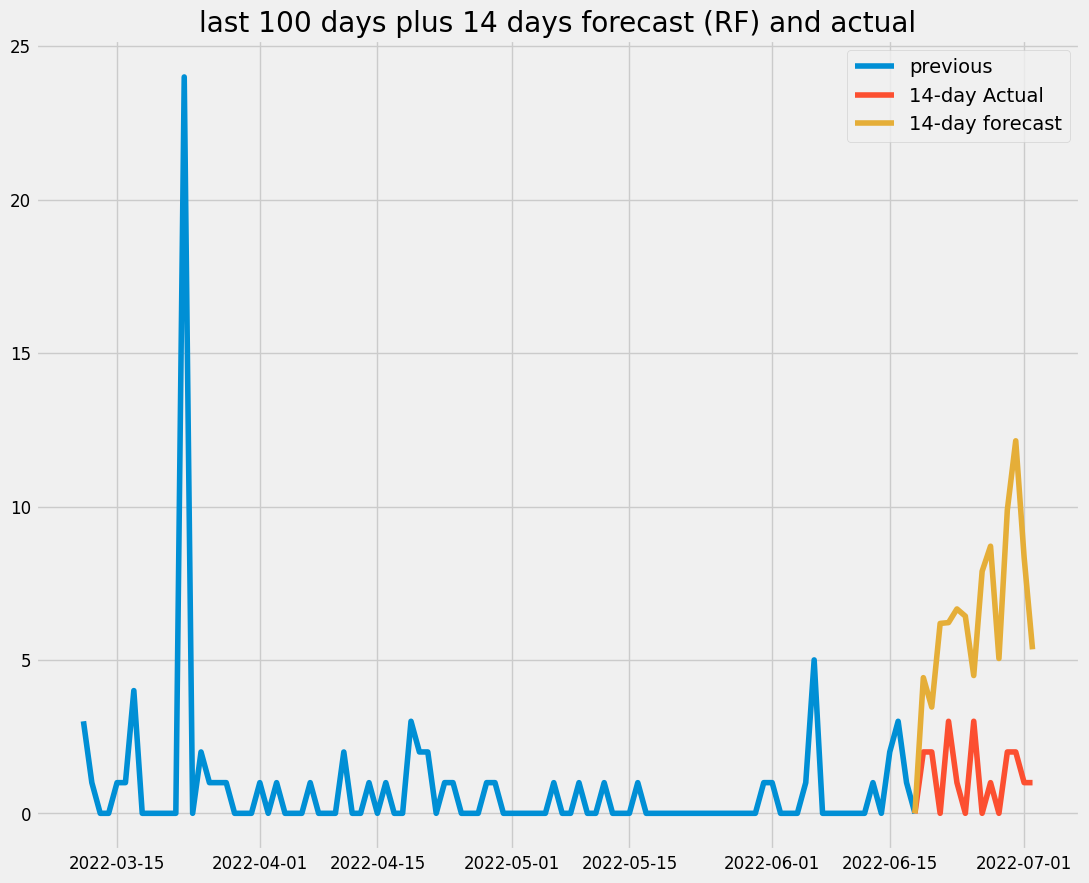


Figure 4.18 - RF 14 days forecast (19th of Jun – 2nd of Jul 2022)

The RF model's hyper parameter is shown in Table 4.3 above. Figure 4.16 depicts a line chart representation of the daily COVID 19 mortality rate using the RF model from March 2020 to June 2022. The actual line mortality rate trend is shown in blue, while the forecasted line trend is shown in red. The RF model in Fig 4.16 demonstrates the existence of overfitting in the data, with the model capturing the noise in the data. As a result, there are wide uncertainty intervals that do not correspond to the actual data point. Figure 4.17 depicts the COVID-19 death rate from December 2021 to July 2022, providing a more recent look at the data. Hence, there is the presence of overfitting of the actual and forecasted datapoint.

Figure 4.18 depicts a 14-day prediction from June 19th to July 2nd, 2022. Data was collected with a 100-day prior data point. However, the RF model forecasted greater values, which were not a reasonable approximation to the actual data pattern. The MSE, which measures the amount of error in the model, is summarized in Table 4.4. As a result, we can observe RF results for Cross validation, Hold-out data, Mean, and Standard deviation.

**Table 4.4 - Summary table for Random Forest model**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Random Forest | Cross-validation | Held out data | Model Error | Overfit |
| MSE | 13.5719 | 36.9391 | 36.94 | True |

### Light Gradient Boosting Machine

The hyper-parameters used to tune the model and their final values are as follows:

**Table 4.5 - LGBM hyper-parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Description | Value |
| n\_estimators | The number of decision trees to use for boosting | 670 |
| max\_depth | The maximum depth each tree is allowed to reach even if its leaves are not pure yet | 20 |
| learning\_rate | How fast the model should learn from the boosting trees | 0.01 |

Chart, histogram

Description automatically generated

Figure 4.19 - Actual vs Forecasted (LGBM) daily Covid-19 mortality rate from March 2020 to June 2022

Chart

Description automatically generated

Figure 4.20 - Actual vs Forecasted (LGBM) daily Covid-19 mortality rate for the last 200 days (Dec 2021 – Jul 2022)

Chart, histogram

Description automatically generated

Figure 4.21 - LGBM 14 days forecast (19th of Jun – 2nd of Jul 2022)

Table 4.5 displays the hyper-parameters that are utilized to modify the LGBM model, as well as their final values. Figure 4.19 depicts a line chart representation of the daily COVID 19 mortality rate using the LGBM model from March 2020 to June 2022. The actual line mortality rate trend is shown in blue, while the forecasted line trend is shown in red. The LGBM model, as shown in Fig 4.19, has overfitting underneath the data, with the model incorporating noise in the data from March 2020 to June 2022. Nonetheless, there are huge uncertainty ranges that do not correspond to the actual data point. Figure 4.20 depicts the COVID-19 mortality rate from December 2021 to July 2022, providing a more recent look at the data. As a result, there is overfitting of the actual and predicted datapoints.

Figure 4.21 depicts a 14-day prediction from June 19th to July 2nd, 2022. Data was collected with a 100-day prior data point. However, the LGBM model forecasted greater values, which were not a reasonable approximation to the actual data pattern. Table 4.6 summarizes the MSE, which measures the amount of inaccuracy in the model. As a result, LGBM results for Cross validation, Held-out data, Mean, and Standard deviation is shown.

**Table 4.6 - Summary table for LGBM model**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LightGBM | Cross-validation | Held out data | Model Error | Overfit |
| MSE | 4.6515 | 32.3548 | 32.35 | True |

## MODEL SUMMARY

The study's main purpose was to predict death rates in West African nations such as Benin, Burkina Faso, Cabo Verde, Gambia, Ghana, Guinea, Guinea-Bissau, Liberia, Mali, Mauritania, Niger, Nigeria, Senegal, Sierra Leone, and Togo. This study made use of statistical and ML approaches. As a result, in comparison to the PROPHET, RF, and LGBM, the statistical model ARIMA performed better, making it the champion model for forecasting COVID-19 mortality. First, historical data was analysed to determine each country's death rate, and Nigeria was found to have the highest mortality rate in West Africa. The visualization also shows how the death rate increases over time, demonstrating that Nigeria has the highest mortality rate in West Africa.

To maintain the sequential ordering of the time series data, the test and train data were generated using a time series split function. The models were trained using the train set, and they were evaluated using MSE. The ARIMA model was picked as the champion model based on the evaluation parameters. The finished models and their errors are summarized in Tables 4.7 and 4.8 below.

Table 4.7 - Overview of the models’ accuracy errors from Mar 2020 to Jun 2022

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Year** | **Month** | **ARIMA** | **PROPHET** | **RF** | **LGBM** |
| **2020** | **Mar** | 13.3957 | -219.325 | 2.7612 | 6.035 |
| **Apr** | 29.1184 | 2.6799 | 13.1096 | 24.7513 |
| **May** | 6.2735 | 227.0498 | 37.154 | 39.4875 |
| **Jun** | 26.2364 | 346.3501 | 51.1496 | 58.5563 |
| **Jul** | -39.1408 | 149.2084 | -43.3836 | -51.3396 |
| **Aug** | -31.7336 | -418.709 | -52.7279 | -70.4807 |
| **Sep** | 9.4228 | -236.179 | 2.9705 | -2.1992 |
| **Oct** | 4.9993 | -63.7761 | -23.4293 | -23.4973 |
| **Nov** | -0.4777 | -79.4542 | -48.3241 | -38.2792 |
| **Dec** | 80.9261 | 110.6749 | 13.6532 | 18.0705 |
| **2021** | **Jan** | 6.1469 | 29.5996 | -12.1929 | 11.5769 |
| **Feb** | -36.406 | 275.2926 | 14.1357 | -5.0802 |
| **Mar** | -29.4217 | 250.1374 | 6.7433 | -15.6935 |
| **Apr** | -12.1676 | -53.0714 | -20.36 | -25.5396 |
| **May** | 12.9408 | -280.403 | 1.6143 | 4.3448 |
| **Jun** | -7.8007 | -392.349 | -5.0818 | -30.2766 |
| **Jul** | 74.6019 | -210.685 | -15.2191 | -23.7609 |
| **Aug** | 111.1412 | 492.1148 | 125.415 | 202.3597 |
| **Sep** | -160.218 | 244.4622 | 0.7973 | 22.4637 |
| **Oct** | -50.7175 | 29.4364 | -1.0087 | 8.8217 |
| **Nov** | 7.963 | 84.5061 | 25.0827 | 54.6751 |
| **Dec** | 29.591 | -92.8461 | -39.3174 | -66.0995 |
| **2022** | **Jan** | 2.5065 | -29.3457 | -29.2086 | -5.3248 |
| **Feb** | -45.5099 | -261.169 | -44.8003 | -46.21 |
| **Mar** | 1.9542 | -47.9243 | -16.8232 | -19.1796 |
| **Apr** | -2.0748 | 44.3452 | -3.7923 | -8.5876 |
| **May** | -1.1008 | 64.0434 | -5.4156 | -5.9109 |
| **Jun** | 3.1192 | 35.4525 | -15.6728 | -13.6834 |

Table 4.7 illustrates the accuracy, which is the difference between the actual and forecasted value. The PROPHET fared the poorest of the four models, while ARIMA did marginally better than RF and LGBM from March 2020 to June 2022. Hence, ARIMA fits in well as the champion model.

**Table 4.8 - Overview of the models’ errors (MAE) for the 14-days held-out data**

|  |  |  |  |
| --- | --- | --- | --- |
| **MSE/ MODEL** | **Held-out Data Error** | **Cross-validation Error** | **Overfit** |
| **ARIMA** | 1.1557 | 38.3406 | False |
| **PROPHET** | 16.0341 | 97.4100 | False |
| **RANDOM FOREST** | 36.9391 | 13.5719 | True |
| **LIGHTGBM** | 32.3548 | 4.6515 | True |

According to Table 4.8, the ARIMA times series model has the lowest error when compared to PROPHET, RF, and LGBM. As a result, ARIMA, which performs better with smaller datasets, is the preferred model for forecasting COVID-19 in West Africa.

# 

# CONCLUSION

This chapter acts as the final chapter of the thesis project. The first section outlines the study's methodologies as well as the findings from the tests. As a result, the limitations of the investigations are highlighted, and some potential changes are suggested. Based on it, provide some future suggestions for this issue, and lastly, summarize this chapter and this thesis report with some project reflections.

## CONCLUSION

This study compares the performance of two distinct timeseries analysis approaches and two ML algorithms in predicting COVID-19 cases in West Africa. To begin, information on COVID19 mortality was gathered from the Johns Hopkins University Centre for Systems Science and Engineering (JHU CSSE), with an emphasis on West African nations. After then, new cases were received daily for each nation, and the time-series data was smoothed using pre-processing processes. After normalizing the data, the model, notably the ARIMA, PROPHET, LGBM, and RF, became appropriate to fit and predict the future trend in these nations.

When the measure of prediction outcomes was compared, the PROPHET, RF, and LGBM did not equal the statistical model ARIMA, which performed better, making it the champion model for forecasting COVID-19 mortality with the lowest MSE.

### Forecast horizon

The prediction period is seven days (a week). This will assist West African government officials, non-governmental organizations, the World Health Organization, and people in combating the spread of COVID-19 in these countries. A weekly prediction is also beneficial in creating short-term measures to combat the spread, as anybody can simply monitor the rate of mortality from their mobile phone, just like a weather forecast. Nonetheless, understanding how serious the Covid-19 death forecast is can enable the government to create short-term efforts to counteract it. This will also be important information for people to prepare for social events depending on the prediction.

## LIMITATIONS

As previously stated, where the confirmed case data, death case data, and recovered case data were downloaded. As a result, our statistic time series model relied solely on data from death cases. This is owing to the limits of the methodologies used, since ARIMA and PROPHET models are univariate timeseries models that rely solely on existing mortality case data to forecast future instances. On the one hand, the workstation used in this study is a conventional laptop computer with limited processing capability and no Compute Unified Device Architecture support (CUDA).

Data from certain countries is not smooth, with occasional extreme movements (rapid spikes or dips) on specific days. This is due to the fact that certain testing facilities or government agencies are closed or do not keep records of new cases on specific days (for example, weekends). Such information will lead to poor performance. Because of the late start time of this Covid-19, data were limited, which may have influenced model accuracy marginally.

## FUTURE WORKS

The basic aims were reached given the extent of the study, however the issues and restrictions faced while doing the research went unexplored. These issues are highlighted below and should be addressed in future work on improvements. To forecast and compare the COVID-19 case trend, only two time-series analytic techniques and two ML Algorithms are used in this work. In comparable time-series analysis situations, several more models, such as Convolutional Neural Network (CNN), other deep learning models, and ML models, can be used. Such models might be used in the future to broaden the comparisons.

In this study, only univariate data is evaluated. Many studies have shown that having more data sources can significantly improve prediction accuracy. In this situation, data on deaths and recovered cases, vaccination rates, and other COVID-19-related data can be used to better estimate the disease trend. The experiment can also be split into two sections: univariate prediction for all models and multivariate prediction for those that supported it.

# Bibliography

Acuna-Soto R, Stahle DW, *et al*. (2009) “Megadrought and megadeath in 16th century Mexico. Emerg Infect” Dis 2009;8(4):360–2. doi:10.3201/eid0804.010175.

Aditya S., Darmawan W., et al (2021) “Time series analysis and forecasting of coronavirus disease in Indonesia using ARIMA model and PROPHET,” *Procedia Computer Science*, vol. 179, pp. 524–532. doi:10.1016/j.procs.2021.01.036.

Akim, Al-mouksit and Firmin Ayivodji. 2020. “Interaction effect of lockdown with economic and fiscal measures against COVID-19 on social-distancing compliance: Evidence from Africa.” Available at SSRN 3621693

Alizadehsani R, Gorriz JM*, et al.(*2021*) “*Uncertainty-aware semi-supervised method using large unlabelled and limited labeled COVID-19 data.” *arXiv preprint arXiv:210206388. 2021.*

An, C.; Lim, H.; Kim, D.W.; Chang, J.H.; Choi, Y.J.; Kim, S.W. Machine learning prediction for mortality of patients diagnosed with COVID-19: A nationwide Korean cohort study. *Sci. Rep.* **2020**, *10*, 18716.

Arlot S., and Lerasle M.,(2016) “Choice of V for V-fold cross-validation in least-squares density estimation.” *J. Mach. Learn. Res. 17(208), 1–50*

ArunKumar K., Kalaga C., *et al* (2021), “Forecasting of COVID-19 using deep layer Recurrent Neural Networks (RNNs) with Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) cells,” *Chaos, Solitons & Fractals*, vol. 146, p. 110861, May 2021. doi: 10.1016/j.chaos.2021.110861.

Boulesteix A., Janitza S., et al (2012) . “Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics.” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(6):493–507.*

Box, G.E., Jenkins, G.M., Reinsel, G.C. and Ljung, G.M., 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

Breiman L. Random Forests. *Mach Learn.*2001;45:5–32.

C. Xie, H. Wen, W. Yang, J. Cai, P. Zhang, R. Wu, M. Li, and S. Huang, “Trend analysis and forecast of daily reported incidence of hand, foot and mouth disease in Hubei, China by PROPHET model,” Scientific Reports, vol. 11, no. 1, p. 1445, Jan. 2021. doi: 10.1038/s41598-021-81100- 2. [Online]. Available: https://doi.org/10.1038/s41598-021-81100-2

Contreras J, Espinola R., *et al* (2003), “ARIMA models to predict next-day electricity prices,” *IEEE Transactions on Power Systems*, vol. 18, no. 3, pp. 1014–1020, Aug. 2003. doi: 10.1109/TPWRS.2002.804943 Conference Name: IEEE Transactions on Power Systems.

Cornelius, E.; Akman, O.; Hrozencik, D. COVID-19 Mortality Prediction Using Machine Learning-Integrated Random Forest Algorithm under Varying Patient Frailty. Mathematics **2021**, 9, 2043. <https://doi.org/10.3390/math9172043>

CSSEGISandData, “COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University,” Sep. 2021, original-date: 2020-02-04T22:03:53Z. [Online]. Available: <https://github.com/CSSEGISandData/COVID-19>

de Freitas Barbosa, V.A.; Gomes, J.C.; de Santana, M.A.; de Lima, C.L.; Calado, R.B.; Bertoldo, C.R., Jr.; de Almeida Albuqurque, J.E.; de Souza, R.G.; de Araujo, R.J.E.; de Souza, R.E.; et al. Covid-19 rapid test by combining a random forest based web system and blood tests. *medRxiv* **2020**.

DeWitte SN (2020). “Mortality risk and survival in the aftermath of the medieval black death.” *PLoS ONE 2014;9(5) e96513. doi:10.1371/JOURNAL.PONE.0096513.*

Dietterich T.G.,(2010) “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization.” *Machine learning, 40(2):139–157.*

Domenico Benvenuto, Marta Giovanetti, Lazzaro Vassallo, Silvia Angeletti, Massimo Ciccozzi, Application of the ARIMA model on the COVID-2019 epidemic dataset, in: Data in Brief, Vol. 29, 2020, 105340, http://dx.doi. org/10.1016/j.dib.2020.105340, (ISSN 2352-3409).

Doornik, J. A., Castle, J. L *et al*. (2020). “Short-term forecasting of the coronavirus pandemic.” *International Journal of Forecasting*, <http://dx.doi.org/10.1016/j.ijforecast.>2020.09.003

Fong Simon, Gloria Li, Nilanjan Dey, Ruben Gonzalez Crespo, Enrique Herrera-Viedma, Finding an accurate early forecasting model from small dataset: A case of 2019-nCoV novel coronavirus outbreak, Int. J. Interact. Multimed. Artif. Intell. 6 (2020) 132–140, <http://dx.doi.org/10.9781/ijimai>. 2020.02.002.

G. A. Papacharalampous and H. Tyralis, “Evaluation of random forests and PROPHET for daily streamflow forecasting,” in Advances in Geosciences, vol. 45. Copernicus GmbH, Aug. 2018. doi: 10.5194/adgeo-45-201-2018 pp. 201–208, iSSN: 1680-7340. [Online]. Available: <https://adgeo.copernicus.org/articles/45/201/2018/>

G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, Time Series Analysis: Forecasting and Control. John Wiley & Sons, May 2015. ISBN 978-1-118-67492-5 Google-Books-ID: rNt5CgAAQBAJ.

G. P. Zhang, “Time series forecasting using a hybrid ARIMA and neural network model,” Neurocomputing, vol. 50, pp. 159–175, Jan. 2003. doi: 10.1016/S0925-2312(01)00702-0. [Online]. Available: https://www. sciencedirect.com/science/article/pii/S0925231201007020

Gaetano Perone, An ARIMA model to forecast the spread and the final size of COVID-2019 epidemic in Italy, 2020, arXiv:2004.00382. Guorong Ding, Xinru Li, Yang Shen, Brief Analysis of the ARIMA model on the COVID-19 in Italy, medRxiv 2020.04.08.20058636. http://dx.doi.org/10.1101/2020.04.08.200586

Gupta, V.K.; Kumar, D.; Sardana, A. Prediction of COVID-19 confirmed, death, and cured cases in India using random forest model. *Big Data Min. Anal.* **2021**, *4*, 116–123.

H. Ritchie, E. Mathieu, L. Rodés-Guirao, C. Appel, C. Giattino, E. Ortiz-Ospina, J. Hasell, B. Macdonald, D. Beltekian, and M. Roser, “Coronavirus Pandemic (COVID-19),” Our World in Data, Mar. 2020. [Online]. Available: <https://ourworldindata.org/covid-vaccinations>

Heinmüller R, Dembélé YA, Jouquet G, Haddad S, Ridde V. Free healthcare provision with an NGO or by the Malian government – Impact on health center attendance by children under five. Field ACTions Science Reports. 2012; Connection on 07 December 2012. <http://factsreports.revues.org/1731>.

Hetal Bhavsar and Amit Ganatra et al (2012) “A comparative study of training algorithms for supervised machine learning.” *International Journal of Soft Computing and Engineering (IJSCE)*, 2(4):2231–2307.

Hiteshi Tandon, Prabhat Ranjan, Tanmoy Chakraborty, Vandana Suhag, Coronavirus (COVID-19): ARIMA based time-series analysis to forecast near future, 2020, arXiv:2004.07859.

Huang C, Wang Y, et al.,(2020) “Clinical features of patients infected with 2019 novel coronavirus in wuhan, china.” *Lancet North Am Ed*  2020;395(10223):497–506.

I. Yenidoğan, A. Çayir, O. Kozan, T. Dağ, and Arslan, “Bitcoin Forecasting Using ARIMA and PROPHET,” in 2018 3rd International Conference on Computer Science and Engineering (UBMK), Sep. 2018. doi: 10.1109/UBMK.2018.8566476 pp. 621–624.

IFRC. COVID-19: rapport sur les retours d’information de la communauté. #7. Dakar: IFRC; 2020.

Iwendi, C.; Bashir, A.K.; Peshkar, A.; Sujatha, R.; Chatterjee, J.M.; Pasupuleti, S.; Mishra, R.; Pillai, S.; Jo, O. COVID-19 Patient Health Prediction Using Boosted Random Forest Algorithm. *Front. Public Health* **2020**, *8*, 357.

J. Contreras, R. Espinola, F. Nogales, and A. Conejo, “ARIMA models to predict next-day electricity prices,” IEEE Transactions on Power Systems, vol. 18, no. 3, pp. 1014–1020, Aug. 2003. doi: 10.1109/TPWRS.2002.804943 Conference Name: IEEE Transactions on Power Systems.

Kavasseri G. and K. Seetharaman,(2006) “Day-ahead wind speed forecasting using f-ARIMA models,” *Renewable Energy*, vol. 34, no. 5, pp. 1388–1393, May 2009. doi: 10.1016/j.renene.2008.09.006.

Lutfi Bayyurt, Burcu Bayyurt, Forecasting of COVID-19 Cases and Deaths Using ARIMA Models, medRxiv 2020.04.17.20069237.http://dx.doi.org/10.1101/2020.04.17.20069237

Mahmoudi M.R., Maleki M, *et al* (2017). “Testing the difference between two independent time series models.” *Iran J Sci Technol A (Sciences)*;41:665–9.

Majhi, R.; Thangeda, R.; Sugasi, R.P.; Kumar, N. Analysis and prediction of COVID-19 trajectory: A machine learning approach. *J. Public Aff.* **2020**, e2537.

McDermott MB, Wang S, and Marinsek N, et al. (2019) “Reproducibility in machine learning for health.” *Paper presented at: 2019 Reproducibility in Machine Learning, RML@ ICLR 2019 Workshop.*

Morens DM, Fauci AS. The 1918 influenza pandemic: insights for the 21st century. J Infect Dis. 2007 Apr 1;195(7):1018-28. doi: 10.1086/511989. Epub 2007 Feb 23. PMID: 17330793.

Naudet J-D. Les “guignols de l’info”. Réflexions sur la fragilité de l’information statistique. Les Nouveaux Cahiers de l’IUED; 2002. p. 31–55.

Papacharalampous G., and H. Tyralis (2018), “Evaluation of random forests and PROPHET for daily streamflflow forecasting,” in *Advances in Geosciences*, vol. 45. Copernicus GmbH, Aug. 2018. doi: 10.5194/adgeo

Parmet, W.E. and Rothstein, M.A., 2018. “The 1918 Influenza Pandemic: Lessons Learned and Not—Introduction to the Special Section.” American Journal of Public Health 108 (11) pp. 1435-1436.

“PROPHET.” [Online]. Available: <http://facebook.github.io/prophet/>

Quaife M, van Zandvoort K, Gimma A, Shah K, McCreesh N, Prem K, et al. The impact of COVID-19 control measures on social contacts and transmission in Kenyan informal settlements. Infect Dis (except HIV/AIDS). 2020. https://doi.org/10.1101/2020.06.06.20122689.

R. G. Kavasseri and K. Seetharaman, “Day-ahead wind speed forecasting using f-ARIMA models,” Renewable Energy, vol. 34, no. 5, pp. 1388–1393, May 2009. doi: 10.1016/j.renene.2008.09.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/ S0960148108003327

R.K. Singh, M. Rani, A.S. Bhagavathula, R. Sah, A.J. Rodriguez-Morales, H. Kalita, C. Nanda, S. Sharma, Y.D. Sharma, A.A. Rabaan, J. Rahmani, P. Kumar, Prediction of the COVID-19 pandemic for the top 15 affected countries:Advanced autoregressive integrated moving average (ARIMA) model, JMIR Public Health Surv. 6 (2) (2020) e19115, http://dx.doi.org/10.2196/19115.

Rajan D, Koch K, Rohrer K, Bajnoczki C, Socha A, Voss M, et al. Governance of the Covid-19 response: a call for more inclusive and transparent decision making. BMJ Glob Health. 2020;5(5):e002655. https://doi.org/10.1136/ bmjgh-2020-002655.

Rao C., and Venkat N., et al (2018) *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications. Elsevier.*

Rice BL, Annapragada AV, Baker RE, Bruijning M, Dotse-Gborgbortsi W, Mensah K, et al. High variation expected in the pace and burden of SARS CoV-2 outbreaks across sub-Saharan Africa. Public Global Health. 2020. <https://doi.org/10.1101/2020.07.23.20161208>.

S. J. Taylor and B. Letham, “Forecasting at scale,” PeerJ Inc., Tech. Rep. e3190v2, Sep. 2017, iSSN: 2167-9843. [Online]. Available: <https://peerj.com/preprints/3190>

Salim, N., Chan, W.H. *et al (*2020). “COVID-19 epidemic in Malaysia: impact of lockdown on infection dynamics.” *medRxiv preprint 2020 10.1101/2020.04.08.20057463*.

Shastri, K., S. Kumar *et al.* (2020)*. “*Time series forecasting of Covid-19 using deep learning models: India-USA comparative case study” .*Department of Computer Science & IT, University of Jammu, Jammu & Kashmir, India*

Sherstinsky A.(2020) “Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network.” *Phys D. doi:10.1016/j.physd.2019. 132306*

Shi, X., Wong, Y.D *et al* (2019) “A feature learning approach based On XGBoost for driving assessment and risk prediction.” *Accid. Anal. Prev., 129, 170–179.*

Sohrabi C, Alsafifi Z, *et al* (2020). “World health organization declares global emergency: a review of the 2019 novel coronavirus (COVID-19).” *Int J Surg 2020.* doi:10.1016/j.ijsu.2020.02.034.

Taiwo Oladipupo Ayodele (2010). “Types of machine learning algorithms”. *New advances in machine learning*, pages 19–48.

Tang, Z.; Zhao, W.; Xie, X.; Zhong, Z.; Shi, F.; Liu, J.; Shen, D. Severity Assessment of Coronavirus Disease 2019 (COVID-19) Using Quantitative Features from Chest CT Images. *arXiv* **2020**, arXiv:2003.11988.

Taylor J., and B. Letham, “Forecasting at scale,” P*eerJ Inc., Tech. Rep. e3190v2, Sep. 2017, iSSN: 2167-9843.* [Online]. Available:<https://peerj.com/preprints/3190>

Touw et al. (2013) Touw WG, Bayjanov JR, Overmars L, Backus L, Boekhorst J, Wels M, Van Hijum SA. Data mining in the Life Sciences with Random Forest: a walk in the park or lost in the jungle? *Briefings in Bioinformatics.*2013;14(3):315–326. doi: 10.1093/bib/bbs034.

V. Kotu and B. Deshpande, “Chapter 12 - Time Series Forecasting,” in *Data Science (Second Edition)*, V. Kotu and B. Deshpande, Eds. Morgan Kaufmann, Jan. 2019, pp. 395–445. ISBN 978-0-12-814761-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/ B9780128147610000125

Wang, J.; Yu, H.; Hua, Q.; Jing, S.; Liu, Z.; Peng, X.; Cao, C.; Luo, Y. A descriptive study of random forest algorithm for predicting COVID-19 patients outcome. *PeerJ* **2020**, *8*, e9945.

Wren-Lewis, Simon. 2020. “The Economic Effects of a Pandemic.” In Economics in the Time of COVID-19. Baldwin, R. and Weder di Mauro, B., ceprpress ed., 109–112.

Xie, H. Wen, W. *et al* (2021), “Trend analysis and forecast of daily reported incidence of hand, foot and mouth disease in Hubei, China by PROPHET model,” *Scientifific Reports*, vol. 11, no. 1, p. 1445, Jan. 2021. doi: 10.1038/s41598-021-81100.

Xingde Duan, Xiaolei Zhang, ARIMA modelling and forecasting of irregularly patterned COVID-19 outbreaks using Japanese and South Korean data, 2020, 105779, http://dx.doi.org/10.1016/j.dib.2020.105779, (ISSN 2352-3409)

XU, C., 2021. A Comparative Study: Time-Series Analysis Methods for Predicting COVID-19 Case Trend.

Yenidoğan I., A. Çayir, O. Kozan, T. Dağ, and Arslan,(2018) “Bitcoin Forecasting Using ARIMA and PROPHET,” in *2018 3rd InternationalConference on Computer Science and Engineering (UBMK)*, doi: 10.1109/UBMK.2018.8566476 pp. 621–624.

Yesilkanat, C.M. Spatio-temporal estimation of the daily cases of COVID-19 in worldwide using random forest machine learning algorithm. *Chaos Solitons Fractals* **2020**, *140*, 110210.

Zombré D, De Allegri M, Ridde V. Immediate and sustained effects of user fee exemption on healthcare utilization among children under five in Burkina Faso: a controlled interrupted time-series analysis. Soc Sci Med. 2017;179:27–35. <https://doi.org/10.1016/j.socscimed.2017.02.027>.

# 

# APPENDIX

**Links**

Modelling - [Amade2233/Forecasting-Covid-19-Mortality-Rate-in-West-Africa (github.com)](https://github.com/Amade2233/Forecasting-Covid-19-Mortality-Rate-in-West-Africa)

Data - [csse\_covid\_19\_daily\_reports](https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_daily_reports)

Web App Covid 19 Mortality Forecast - [West Africa Covid-19 Mortality Prediction Web App](https://amade2233-myproject-deploy-vljy3v.streamlitapp.com/)

Deployment - [Amade2233/MYPROJECT (github.com)](https://github.com/Amade2233/MYPROJECT)

## ============NOTEBOOK FOR DATA EXPLORATION AND MODELLING =========

1. :!pip install -r requirements.txt1> /dev/null2> /dev/null
2. :importnumpyasnp importpandasaspd

importmatplotlib.pyplotasplt frompylabimportrcParams

fromsklearn.model\_selectionimportTimeSeriesSplit, train\_test\_splitastts fromsklearn.model\_selectionimportGridSearchCV, RandomizedSearchCV fromsklearn.preprocessingimportMinMaxScaler fromsklearn.metricsimportmean\_squared\_errorasMSE

fromsklearn.ensembleimportRandomForestRegressor fromlightgbmimportLGBMRegressor

fromstatsmodels.tsa.stattoolsimportadfuller fromstatsmodels.tsa.statespace.toolsimportdiff fromstatsmodels.tsa.seasonalimportseasonal\_decompose fromstatsmodels.graphics.tsaplotsimportplot\_pacf, plot\_acf fromstatsmodels.tsa.arima.modelimportARIMA

importpmdarimaaspm frompmdarima.arimaimportndiffs frompmdarimaimportauto\_arima

fromprophetimportProphet

*# warnings.filterwarnings('ignore')*

seed=0

1. :plt.style.use( 'fivethirtyeight') rcParams['axes.labelsize']=14 rcParams['xtick.labelsize']=12 rcParams['ytick.labelsize']=12 rcParams['text.color']= 'k'

### Business Understanding

#### Research Goals

The goal of this research thesis is to make a weekly forecast on the number of deaths caused by COVID-19 in West Africa based on historic records of weekly COVID-19 deaths in the same region. The historic data will be explored to find insights and patterns in covid-19 fatality. Comparism of two statistical models - ARIMA and PROPHET - and two ML models – RF and LGBM will be carried out to select the best model. Based on the forecasted values the models will be evaluated using MSE to select the best performing model. The goals can be summarized as below: 1) Exploratory data analysis 2) Using cross-validation: 1) Generate the statistical models - ARIMA and PROPHET using the train data. 2) Generate the ML models – RF and LGBTM using the train data. 3) Validate the models using the test data.

#### Project Plan

**Assessment of tools and techniques**

* + - * Language: Python
      * Libraries:
        + statsmodels - timeseris analysis, statistical models
        + Numpy & Pandas - data wrangling
        + Matplotlib - Visualizations
        + Scikit Learn – RF model and other ML tool
        + LGBM model
        + PROPHET - PROPHET model by Facebook
        + Streamlit - Deployment

#### Data exploration

1. :wafr\_countries={

"Benin":"BEN",

"Burkina Faso":"BFA", "Cabo Verde":"CPV",

"Cote d'Ivoire":"CIV",

"Gambia":"GMB",

"Ghana":"GHA",

"Guinea":"GIN",

"Guinea-Bissau":"GIN",

"Liberia":"LBR",

"Mali":"MLI",

"Mauritania":"MRT",

"Niger":"NER",

"Nigeria":"NGA",

"Senegal":"SEN", "Sierra Leone":"SLE", "Togo":"TGO"

}

1. :defprepare\_data(path):

data=pd.read\_csv(path).set\_index(keys="Country/Region").drop(columns=␣

*‹→*["Province/State","Lat","Long"])

data.columns=pd.to\_datetime(data.columns) wafr\_data=data.loc[wafr\_countries.keys()] *#.sum()["2020-02-28":]* wafr\_data=wafr\_data.T["2020-03-01":"2022-05-26"]

returnwafr\_data

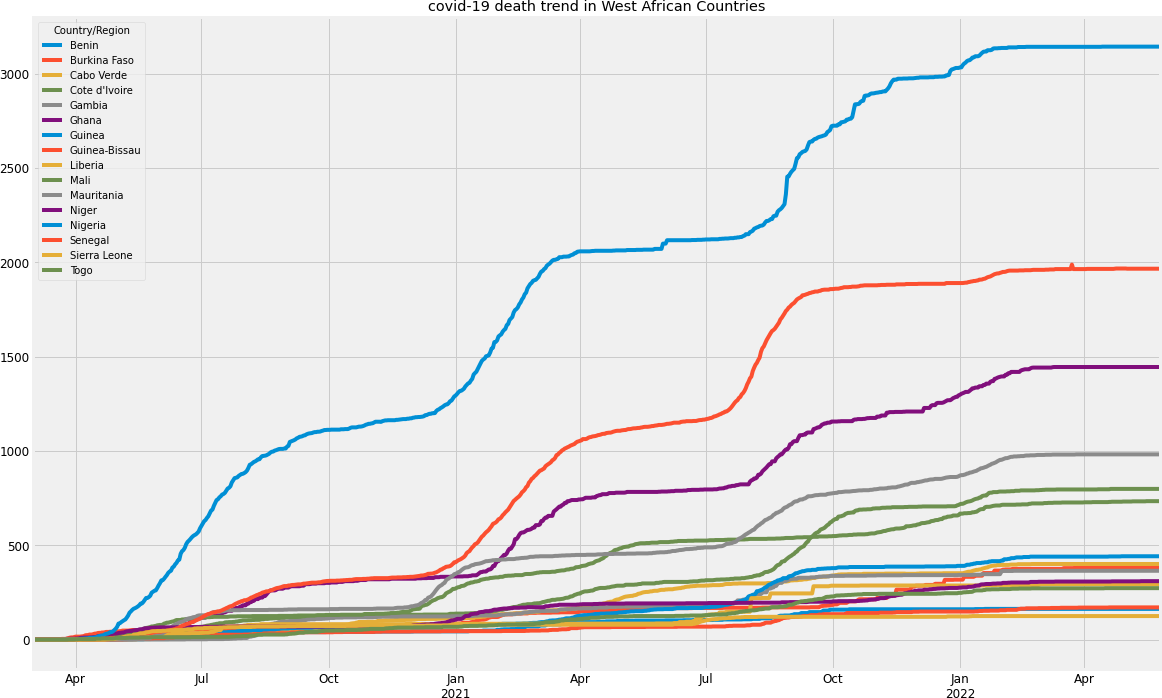
1. :cases=prepare\_data("../datasets/time\_series\_covid19\_confirmed\_global.csv") deaths=prepare\_data("../datasets/time\_series\_covid19\_deaths\_global.csv")
2. :ax=deaths.plot(title="covid-19 death trend in West African Countries",␣

*‹→*figsize=(18,12))

ax.figure.savefig("../plots/data\_exploration/"+"covid-19 death trend in West␣

*‹→*African Countries.png")

print("fig 1")

fig 1

|  |  |
| --- | --- |
| Sierra Leone | 125 |
| Benin | 163 |
| Guinea-Bissau | 171 |
| Togo | 273 |
| Liberia | 294 |
| Niger | 310 |
| Gambia | 365 |
| Burkina Faso | 384 |
| Cabo Verde | 401 |
| Guinea | 442 |
| Mali | 734 |
| Cote d'Ivoire | 799 |
| Mauritania | 982 |
| Ghana | 1445 |
| Senegal | 1988 |
| Nigeria | 3143 |
| dtype: int64 |  |

1. :ax=deaths.max().sort\_values().plot(kind="bar", title="total death caused␣

*‹→*by covid-19 in West African Countries", figsize=(15,8))

ax.figure.savefig("../plots/data\_exploration/"+"total death caused by covid-19␣

*‹→*in West African Countries.png")

print("fig 2")

fig 2

1. :print("fig 3.") deaths.max().sort\_values()

fig 3. [9]:Country/Region

The trend of the pandemic deaths in each country shown in fig 1. shows a regular pattern for all West African countries. For each country, there has been an uptrend in the number of deaths since the start of the pandemic until the year 2022, which has shown a rather flat trend with the number of deaths dropping significantly in all countries.

1. :print("cases and deaths trend for each country") cases and deaths trend for each country

[11]:

*# for i in deaths.columns:*

*# ax = cases[[i]].merge(deaths[[i]], left\_index = True, right\_index = True)\ # .rename(columns = {i+"\_x" : "cases", i+"\_y" : "deaths"}).plot(figsize =*␣

*‹→(12, 6), xlabel = i)*

*# ax.figure.savefig("../plots/data\_exploration/" + i + "\_cases+deaths.png")*

[12]:print("death trend for each country") death trend for each country

[13]:

*# for i in deaths.columns:*

*# ax = deaths[[i]].rename(columns = {i : "deaths"}).plot(figsize = (12, 6),*␣

*‹→xlabel = i)*

*# ax.figure.savefig("../plots/data\_exploration/" + i + "\_deaths.png")*

#### Data quality verification

JHU CSSE in their COVID-19 dataset[github repository](https://github.com/CSSEGISandData/COVID-19)listed all sources which they aggragate their data from and are mostly official government sources. The full list can be checked in their [github repository](https://github.com/CSSEGISandData/COVID-19)

### Data Prepartion

### Data selection

Only row entries containing west African states data are selected from the cases dataset and death dataset

The columns needed are only the time-series record which started from 22nd of Januuary 2020 till date. From the cases dataset, West Africa did not have any case till 28th of February 2020, so the column selection started from 28th of February till date

#### Data Cleaning

Given the forecast will be weekly, the last timeseries column would be the last daily entry that ends

the week and the others will be discarded. This will be useful during internal model development and testing but will not be useful after deployment as the programme will fetch the timeseries data at the end of every week

There are no missing values in the selected columns

#### Data construction

The timeseries data available for both cases and deaths are daily records.

#### For the timeseries models, the following was done to construct data for modelling

* + - * only the confirmed deaths dataset will be used as ARIMA and PROPHET are both Univariate timeseries models
      * To get total cases in West Africa for each day, the daily records from each country is summed up.
      * The timeseries report on both dataset are cummulative i.e the total number of cases (same as death) is the total cases for that country/Region at that time. a 1st order differencing was done to get only the new cases for that day. 1st order differencing is done by subtracting the new timeseries record (t) from the previous timeseries record (t-1) for all timeseries record (tn) > t = t - t-1

#### For the Machine Learning models, the following was done to construct data for modelling

* + - * The daily number of confirmed cases and deaths data will be used when modelling with ML models. Since the the number of confirmed deaths is influenced by the number of confirmed cases, it is useful to add the confirmed cases data as a feature in the ML models.
      * For both datasets the sum of each days record by the west African countries was done give the West African daily record
      * a 1st order differencing was done on both data to get the new cases instead of the cummula- tive cases
      * The two datsets are merged with each data taking one column
      * 1-day and 7-day Lagged results for both the cases and deaths column was made and merged with the data as they proved to be useful features during experimentation
      * The original cases column was removed since the data would not be available in real life use but its lags
      * The deaths column is seperated out to be used as the dependent variable (y) while the other four columns are the independent variables(X)

1. :path="../datasets/"
2. :wafr\_countries={

"Benin":"BEN",

"Burkina Faso":"BFA", "Cabo Verde":"CPV",

"Cote d'Ivoire":"CIV",

"Gambia":"GMB",

"Ghana":"GHA",

"Guinea":"GIN",

"Guinea-Bissau":"GIN",

"Liberia":"LBR",

"Mali":"MLI",

"Mauritania":"MRT",

"Niger":"NER",

"Nigeria":"NGA",

"Senegal":"SEN", "Sierra Leone":"SLE", "Togo":"TGO"

}

1. :defprepare\_data\_day(path):

data=pd.read\_csv(path).set\_index(keys="Country/Region").drop(columns=␣

*‹→*["Province/State","Lat","Long"])

data.columns=pd.to\_datetime(data.columns) wafr\_data=data.loc[wafr\_countries.keys()].sum()["2020-03-01":] wafr\_data=wafr\_data-wafr\_data.shift(1).fillna(0) wafr\_data[wafr\_data<0]=0

returnwafr\_data

defprepare\_ml\_data(confirmed\_data, deaths\_data): confirmed\_data.name="confirmed\_cases" deaths\_data.name="deaths"

ml\_data=pd.merge(confirmed\_data, deaths\_data, left\_index=True,␣

*‹→*right\_index=True)

returnml\_data

defprepare\_ml\_data\_with\_features(cases, deaths): ml\_data=prepare\_ml\_data(cases, deaths)

ml\_data.columns=["cases","deaths"] ml\_data.index.name="dates(daily)"

*# making lagged data for cases and deaths*

foriinml\_data.columns: forjin[1,7]:

ml\_data[f"pd{j}\_lagged\_{i}"]=ml\_data[i].shift(j).fillna(0)

ml\_data=ml\_data.drop(columns=["cases"])

X, y=ml\_data.drop(columns="deaths"), ml\_data.deaths ml\_data=X.copy()

ml\_data["deaths"]=y returnX, y

1. :defprepare\_data\_day\_noshift(path): data=pd.read\_csv(path).set\_index(keys="Country/Region").drop(columns=␣

*‹→*["Province/State","Lat","Long"])

data.columns=pd.to\_datetime(data.columns) wafr\_data=data.loc[wafr\_countries.keys()].sum()["2020-03-1":"2022-06-26"]

returnwafr\_data

defprepare\_ml\_data\_weekly(confirmed\_path, deaths\_path): cases=prepare\_data\_day\_noshift(confirmed\_path) deaths=prepare\_data\_day\_noshift(deaths\_path)

cases.name="cases" deaths.name="deaths"

ml\_data=pd.merge(cases, deaths, left\_index=True, right\_index=True) ml\_data=ml\_data.loc[ml\_data.index[::7]]

ml\_data.index.name="weekend"

ml\_data=ml\_data-ml\_data.shift(1).fillna(0) ml\_data[ml\_data<0]=0

*# making lagged data for cases and deaths*

foriinml\_data.columns: forjinrange(1,3):

ml\_data[f"pd{j}\_lagged\_{i}"]=ml\_data[i].shift(j).fillna(0)

ml\_data=ml\_data.drop(columns=["cases"])

X, y=ml\_data.drop(columns="deaths"), ml\_data.deaths ml\_data=X.copy()

ml\_data["deaths"]=y returnX, y, ml\_data

1. :paths=["time\_series\_covid19\_deaths\_global.csv",␣

*‹→*"time\_series\_covid19\_confirmed\_global.csv"]

1. :cases\_day=prepare\_data\_day(path+paths[1])
2. :cases\_day

|  |  |
| --- | --- |
| [20]:2020-03-01 | 1.0 |
| 2020-03-02 | 1.0 |
| 2020-03-03 | 1.0 |
| 2020-03-04 | 2.0 |
| 2020-03-05 | 0.0 |
|  | ... |
| 2022-06-28 | 198.0 |
| 2022-06-29 | 1486.0 |
| 2022-06-30 | 1079.0 |
| 2022-07-01 | 880.0 |
| 2022-07-02 | 760.0 |
| Length: 854, | dtype: float64 |

1. :deaths\_day=prepare\_data\_day(path+paths[0])

[22]:

*# prepare\_ml\_data(cases\_day, recovered\_day, deaths\_day).loc["2021"].style*

#### Test for Outliers using lag plots

By using lag plots the researcher made visible some outliers in the dataset. In the case of the con- firmed cases dataset values greater than hundred are far from other values which are concentrated below 80

1. :pd.plotting.lag\_plot(deaths\_day, lag=1)

plt.title("lag plot to show outliers in the death cases dataset") [23]:Text(0.5, 1.0, lag plot to show outliers in the death cases dataset)

#### Test if data is stationary - Augmented Dickey-Fuller test

The Augmented Dickey-Fuller test is a type of statistical test called a unit root test.

The intuition behind a unit root test is that it determines how strongly a time series is defined by a trend.There are a number of unit root tests and the Augmented Dickey-Fuller may be one of the more widely used. It uses an autoregressive model and optimizes an information criterion across mul- tiple different lag values.

The null hypothesis of the test is that the time series can be represented by a unit root, that it is not stationary (has some time-dependent structure). The alternate hypothesis (rejecting the null hypothesis) is that the time series is stationary.

Null Hypothesis (H0): If failed to be rejected, it suggests the time series has a unit root, meaning it is non-stationary. It has some time dependent structure. Alternate Hypothesis (H1): The null hypothesis is rejected; it suggests the time series does not have a unit root, meaning it is stationary. It does not have time-dependent structure.

**Visually** The stationarity of a data can also be roughly determined visually after seasonal de- composition of the data. But this is not an accurate way to tell for sure

**Result** From the adf test carried out the deaths dataset is stationary but the cases dataset is not.

This will be handled in the Modelling phase

1. :stationarity\_test=[] data=[cases\_day, deaths\_day]

names=["daily cases","daily deaths"] fori,jinzip(data, names):

X=i.values result=adfuller(X) ifresult[1]>0.05:

stationarity\_test.append((j, result[1],0.05,"reject")) else:

stationarity\_test.append((j, result[1],0.05,"accept"))

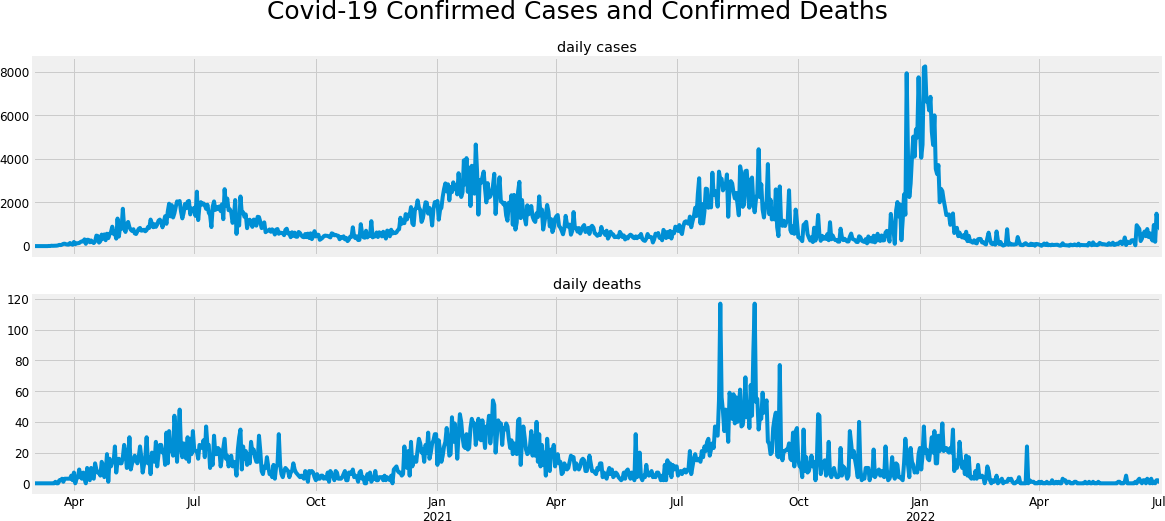
1. :test\_stationary=pd.DataFrame(stationarity\_test, columns=["variable",␣

*‹→*"p-value","threshold p-value","accept/reject"])

1. :test\_stationary [26]:variable p-value threshold p-value accept/reject
2. daily cases 0.000575 0.05 accept
3. daily deaths 0.113248 0.05 reject
4. :fig, axs=plt.subplots(2,1, figsize=(18,8), sharex="col") fordf, ax, nameinzip(data, axs.flatten(), names):

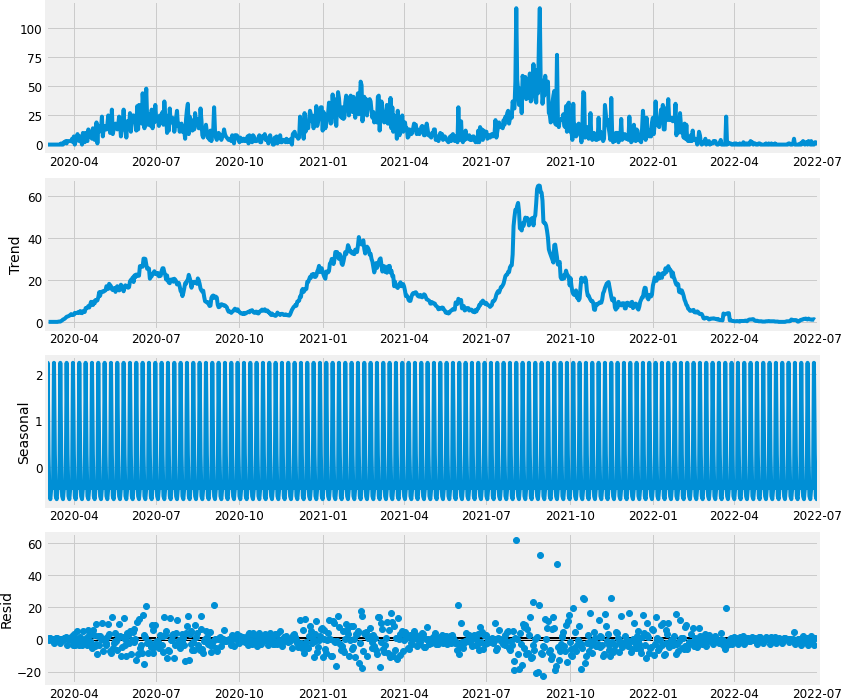
(df).plot(ax=ax, title=name)

plt.suptitle("Covid-19 Confirmed Cases and Confirmed Deaths", size=25);



1. :rcParams[ 'figure.figsize']=(12,10) n=seasonal\_decompose(deaths\_day).plot(); print("seasonal decomposition of the deaths dataset")

seasonal decomposition of the deaths dataset



#### showing lag correlation using acf and pacf (auto regression)

A time series data is said to be autoregressive if a current can be obtained by a previous value or previous value. Similarly a model is autoregressive if it predicts future values based on past values.

ACF (auto-correlation function) describes how well the present value of the series is related with its past values

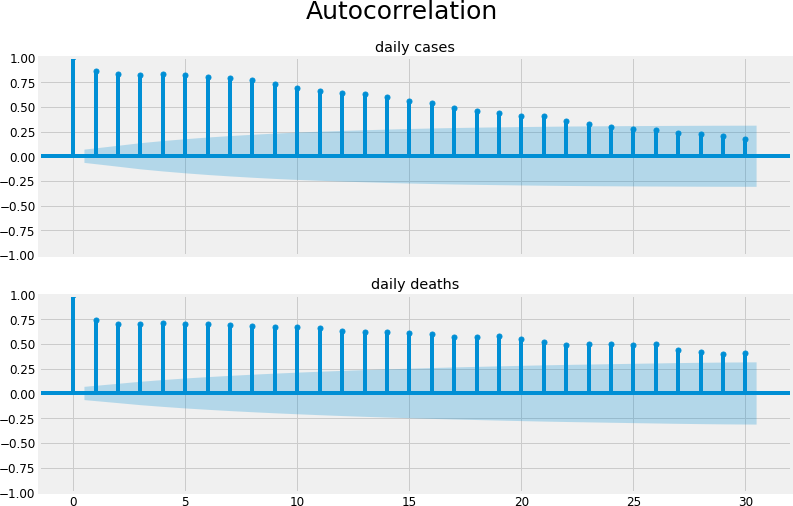
PACF (partial auto-correlation function) describes how a present value of a series is related to a past value after removing the effects which are already explained by the earlier lag(s) with the next lag

#### Result

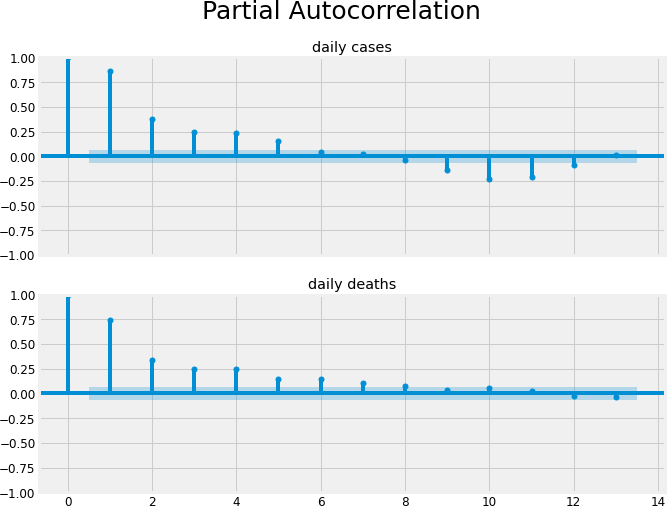
* + - * From the acf plot results, both the cases dataset and deaths dataset are highly autoregressive with correlations going farther than 30 lags for the deaths dataset and 22 for the cases dataset
      * The Pacf plot also showed partial autocorrelation till the 9th lag for the deaths dataset and 7th lag for the cases dataset

1. :fig, axs=plt.subplots(2,1, figsize=(12,8), sharex="col") fordf, ax, nameinzip(data, axs.flatten(), names):

plot\_acf(df, ax=ax, title=name) plt.suptitle("Autocorrelation", size=25);



1. :fig, axs=plt.subplots(2,1, figsize=(10,8), sharex="col") fordf, ax, nameinzip(data, axs.flatten(), names): plot\_pacf(df, lags=13, ax=ax, title=name) plt.suptitle("Partial Autocorrelation", size=25);



### Modelling

#### Rolling Origin Cross Validation

Cross-validation is a statistical method to perform robust model evaluation. Cross validation gives the researcher a good estimate of the error of a model by training the model on different portions of the dataset and providing and average error. For example, in K-fold-Cross-Validation, the dataset will be splitted into several folds (i.e 3, 5, 10, 15 folds), then training of model is carried out on all folds except one which will be used for testing. This stepp will be repeated until the model has been tested on all folds. The final metrics will be the average of error scores obtained in every fold. K-folds cross validation as well as other kinds prevents overfitting, and a more robust way to evaluate model performance than the traditional holdout/train-test format

In the case of time series, the cross-validation is not trivial. Random samples cannot be chosen at random and assigned to either the test set or the train set because it makes no sense to use the values from the future to forecast values in the past. Time series data is chronologically ordered, so models should avoid data leakage of future data through the training samples. The chronology of the Time series data must be preserved during modelling.

Rolling origin is a model evaluation technique where the forecast origin rolls forward in time and forecasts are produced from each origin (Svetunkov and Petropoulos, 2018). This method start with a small subset of data for training purpose (without shuffling), forecast for the later data

points is made and then the error is calculated for the forecasted data points. The same forecasted data points are then included as part of the next training dataset and subsequent data points are forecasted.

For example, if the data set contains 12 months of data, 3 months can be allocated the training set and 1 month to the validation set. The origin is then rolled forward a month, and model retrained and evaluated on the next month. The process is repeated until the last month is got too giving a total of 9 test errors produced over the year. The averaged of the errors give the overall error estimate. The training set can be set to accumulate over-time (i.e. increasing in-sample size) **fig**

1. **below**, for example by month 12, training set size may be 11 months with a single month as a validation set. Alternatively, the training set can be kept constant (i.e. constant in-sample size) **fig**
2. **below**. In this work the origin is rolled forward 14 days at a time with new models built at each origin. The “TimeSeriesSplit” class in the python “Sci-Kit Learn” package was used to develop the rolling origin cross validation strategy with accumulative training set used in this work
   * 1. **Statistical Models**
3. :days=14

train, holdout=deaths\_day.iloc[:-days], deaths\_day.iloc[-days:]

1. :split=TimeSeriesSplit(n\_splits=15, test\_size=14) cv\_ind=split.split(train) *# indices of the cv folds*

cv\_data=[] *# splits of the data into train-test folds* cv\_ts=[] *# splits of the time\_series index into folds* fortrn\_ind, tst\_indincv\_ind:

cv\_data.append((train.iloc[trn\_ind].values, train.iloc[tst\_ind].values)) cv\_ts.append((train.index[trn\_ind], train.index[tst\_ind]))

**ARIMA PROCEDURE** - Hold out 2 weeks worth of the data for validation - Estimate ARIMA differencing term, d on the remaining data using the ndiffs class in pmdarima library - use the autoarima class to find the best parameter for p and q - for model evaluation split the data using rolling origin cross validation of 15 folds with two weeks test set data. For each fold: - fit the ARIMA model with the calculated p, d and q on the train fold - calculate the MSE of the model using the test fold - store the MSE value in a list - calculate the average MSE value from the MSE list to calculate the error of the model - To have a test of how well the model will perform with real data, the model is tested on the held out data and the MSE is calculated. If the error increased significantly, then the model will not be stable to real life data

**Result** MSE mean after cross validation: 38.3406 MSE on heldout data: 1.1556779260950596

Plots are in code cells

1. :kpss\_diffs=ndiffs(train, alpha=0.05, test= 'kpss', max\_d=6) adf\_diffs=ndiffs(train, alpha=0.05, test= 'adf', max\_d=6) n\_diffs=max(adf\_diffs, kpss\_diffs)

print(n\_diffs)

1

1. :auto=pm.auto\_arima(train, d=n\_diffs, seasonal=False, stepwise=True,␣

*‹→*suppress\_warnings=True,

max\_p=6, max\_Q=6, trace=1, random\_state=seed)

Performing stepwise search to minimize aic

ARIMA(2,1,2)(0,0,0)[0] intercept : AIC=5953.007, Time=0.71 sec ARIMA(0,1,0)(0,0,0)[0] intercept : AIC=6306.631, Time=0.02 sec ARIMA(1,1,0)(0,0,0)[0] intercept : AIC=6145.477, Time=0.09 sec ARIMA(0,1,1)(0,0,0)[0] intercept : AIC=5951.855, Time=0.08 sec ARIMA(0,1,0)(0,0,0)[0] : AIC=6304.631, Time=0.01 sec ARIMA(1,1,1)(0,0,0)[0] intercept : AIC=5950.748, Time=0.10 sec ARIMA(2,1,1)(0,0,0)[0] intercept : AIC=5949.075, Time=0.17 sec ARIMA(2,1,0)(0,0,0)[0] intercept : AIC=6070.274, Time=0.10 sec ARIMA(3,1,1)(0,0,0)[0] intercept : AIC=5946.540, Time=0.33 sec ARIMA(3,1,0)(0,0,0)[0] intercept : AIC=6008.968, Time=0.13 sec ARIMA(4,1,1)(0,0,0)[0] intercept : AIC=5948.445, Time=0.50 sec ARIMA(3,1,2)(0,0,0)[0] intercept : AIC=5944.959, Time=1.09 sec ARIMA(4,1,2)(0,0,0)[0] intercept : AIC=5948.619, Time=1.08 sec ARIMA(3,1,3)(0,0,0)[0] intercept : AIC=5941.821, Time=1.61 sec ARIMA(2,1,3)(0,0,0)[0] intercept : AIC=5952.100, Time=1.28 sec ARIMA(4,1,3)(0,0,0)[0] intercept : AIC=5944.168, Time=1.83 sec ARIMA(3,1,4)(0,0,0)[0] intercept : AIC=5942.840, Time=2.17 sec ARIMA(2,1,4)(0,0,0)[0] intercept : AIC=5940.883, Time=1.80 sec ARIMA(1,1,4)(0,0,0)[0] intercept : AIC=5941.764, Time=2.00 sec ARIMA(2,1,5)(0,0,0)[0] intercept : AIC=5942.805, Time=3.30 sec ARIMA(1,1,3)(0,0,0)[0] intercept : AIC=5939.803, Time=1.02 sec ARIMA(0,1,3)(0,0,0)[0] intercept : AIC=5949.957, Time=0.30 sec ARIMA(1,1,2)(0,0,0)[0] intercept : AIC=5951.635, Time=0.38 sec ARIMA(0,1,2)(0,0,0)[0] intercept : AIC=5950.255, Time=0.13 sec ARIMA(0,1,4)(0,0,0)[0] intercept : AIC=5944.801, Time=0.47 sec ARIMA(1,1,3)(0,0,0)[0] : AIC=5937.804, Time=0.52 sec

ARIMA(0,1,3)(0,0,0)[0] : AIC=5947.958, Time=0.18 sec

ARIMA(1,1,2)(0,0,0)[0] : AIC=5949.635, Time=0.25 sec

ARIMA(2,1,3)(0,0,0)[0] : AIC=5950.101, Time=0.48 sec

ARIMA(1,1,4)(0,0,0)[0] : AIC=5939.764, Time=1.30 sec

ARIMA(0,1,2)(0,0,0)[0] : AIC=5948.255, Time=0.17 sec

ARIMA(0,1,4)(0,0,0)[0] : AIC=5942.801, Time=0.30 sec

ARIMA(2,1,2)(0,0,0)[0] : AIC=5951.007, Time=0.42 sec

ARIMA(2,1,4)(0,0,0)[0] : AIC=5938.884, Time=0.78 sec Best model: ARIMA(1,1,3)(0,0,0)[0]

Total fit time: 25.109 seconds

1. :order=auto.order print(order)

(1, 1, 3)

[36]:

*## cross validate*

err\_list=[]

fortrn, tstincv\_data:

model=ARIMA(trn, order=order).fit() err=MSE(tst, model.forecast(len(tst))) err\_list.append(err)

[37]:print(f"MSE mean:{np.mean(err\_list):.4f}, MSE std:{np.std(err\_list):.4f}") MSE mean: 38.3406, MSE std: 44.1156

[38]:

*## test on held out data* forecasts=auto.predict(len(holdout)) err=MSE(holdout, forecasts) print(err)

1.1556779260950596

1. :ARIMA\_err=(np.mean(err\_list), err)

#### Result

1. :ind=0

train\_slice=train.iloc[ind:] xticks=train\_slice.index.tolist()

plt.plot(xticks, train\_slice.values) plt.plot(xticks, auto.predict\_in\_sample()[ind:]) plt.legend(["Actual data","forecasted data"])

plt.title("Actual daily historical data vs Predicted")

[41]:ind=-200

train\_slice=train.iloc[ind:] xticks=train\_slice.index.tolist()

plt.plot(xticks, train\_slice.values) plt.plot(xticks, auto.predict\_in\_sample()[ind:]) plt.legend(["Actual data","forecasted data"])

plt.title(f"Actual daily historical data vs Predicted for{abs(ind)}days")

[42]:ind=-100

train\_slice=train.iloc[ind:] xticks=train\_slice.index.tolist()

n=len(holdout)+1

forecast=forecast=[train\_slice.iloc[-1]]+auto.predict(n-1).tolist() holdout\_=[train\_slice.iloc[-1]]+holdout.iloc[:n-1].tolist() holdout\_xticks=pd.date\_range(start=xticks[-1], periods=n)

plt.plot(xticks, train\_slice.values) plt.plot(holdout\_xticks, holdout\_) plt.plot(holdout\_xticks, forecast)

plt.legend(labels=["previous data",f"{n-1}-day Actual",f"{n-1}-day␣

*‹→*forecast"])

plt.title(f"last{abs(ind)}days plus{n-1}days forecast and actual")

[43]:ind=-200

train\_slice=train.iloc[ind:] xticks=train\_slice.index.tolist()

n=15

forecast=[train\_slice.iloc[-1]]+auto.predict(n-1).tolist() holdout\_xticks=pd.date\_range(start=xticks[-1], periods=n).tolist()

plt.plot(xticks, train\_slice.values) plt.plot(holdout\_xticks, forecast)

*# plt.plot(range(len(holdout.iloc[:14])), holdout.iloc[:14].values)*

plt.legend(labels=["previous data",f"{n-1}-day forecast"]) plt.title(f"last{abs(ind)}days plus{n-1}days forecast and actual")

[43]:Text(0.5, 1.0, 'last 200 days plus 14 days forecast and actual')

**PROPHET by Meta (previously Facebook)** PROPHET is an open source time-series forecasting soft- ware released by Meta (Facebook) Core Data Science team. It is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. PROPHET is robust to missing data and shifts in the trend, and typically handles outliers well[cite.](https://facebook.github.io/prophet/)

PROPHET parameter tuning is fully automatic making it easy to use.

The PROPHET model uses a decomposable time series model with three main model components: trend, seasonality, and holidays. They are combined in the following equation: > y(t) = g(t) + s(t)

+ h(t) + t[cite](https://doi.org/10.7287/peerj.preprints.3190v2)

Here g(t) is the trend function which models non-periodic changes in the value of the time series, s(t) represents periodic changes (e.g., weekly and yearly seasonality), and h(t) represents the ef- fects of holidays which occur on potentially irregular schedules over one or more days. The error term t represents any distinguishing changes which are not accommodated by the model.

**PROCEDURE** - Hold out 2 weeks worth of the data for validation - for model evaluation split the data using rolling origin cross validation of 15 folds with two weeks test set data. For each fold: - fit the prophet model with on the train fold - calculate the MSE of the model using the test fold - store the MSE value in a list - calculate the average MSE value from the MSE list to calculate the error of the model - To have a test of how well the model will perform with real data, the model is

tested on the held out data and the MSE is calculated. If the error increased significantly, then the model will not be stable to real life data

*## cross validate*

err\_list=[]

for(trn\_ts, tst\_ts), (trn, tst)inzip(cv\_ts, cv\_data): *## make data suitable for model* df=pd.DataFrame(trn, index=trn\_ts).reset\_index() df.columns=["ds","y"]

*## fit model* model=Prophet() model.fit(df)

*## make prediction* future=model.make\_future\_dataframe(periods=len(tst)) forecast=model.predict(future)[ 'yhat'].values

*## test prediction*

err=MSE(tst, forecast[-len(tst):]) err\_list.append(err)

[44]:

**Result** MSE mean after cross validation: 97.4101 MSE on heldout data: 16.034133074515896

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10:29:37 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:29:38 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:29:39 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:29:39 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:29:41 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:29:41 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:29:42 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:29:42 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:29:44 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:29:44 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:29:45 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:29:45 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:29:47 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:29:47 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:29:48 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:29:48 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:29:50 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:29:50 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:30:15 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:30:15 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:30:41 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:30:41 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:31:04 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:31:04 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:31:26 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:31:26 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:31:50 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:31:50 | - cmdstanpy | - INFO | - Chain | [1] | done processing |
| 10:32:15 | - cmdstanpy | - INFO | - Chain | [1] | start processing |
| 10:32:15 | - cmdstanpy | - INFO | - Chain | [1] | done processing |

1. :print(f"MSE mean:{np.mean(err\_list):.4f}, MSE std:{np.std(err\_list):.4f}")

MSE mean: 97.4101, MSE std: 75.0071

1. :df=train.to\_frame().reset\_index() df.columns="ds","y"

df.head()

1. :ds y

|  |  |
| --- | --- |
| 0 2020-03-01 | 0.0 |
| 1 2020-03-02 | 0.0 |
| 2 2020-03-03 | 0.0 |
| 3 2020-03-04 | 0.0 |
| 4 2020-03-05 | 0.0 |

1. :model=Prophet()

model.fit(df)

10:32:38 - cmdstanpy - INFO - Chain [1] start processing 10:32:38 - cmdstanpy - INFO - Chain [1] done processing

1. :<prophet.forecaster.Prophet at 0x7f0231f1b790>
2. :future=model.make\_future\_dataframe(periods=len(holdout)) forecasts=model.predict(future)

[49]:

*## test on held out data*

err=MSE(holdout, forecasts["yhat"][-len(holdout):]) print(err)

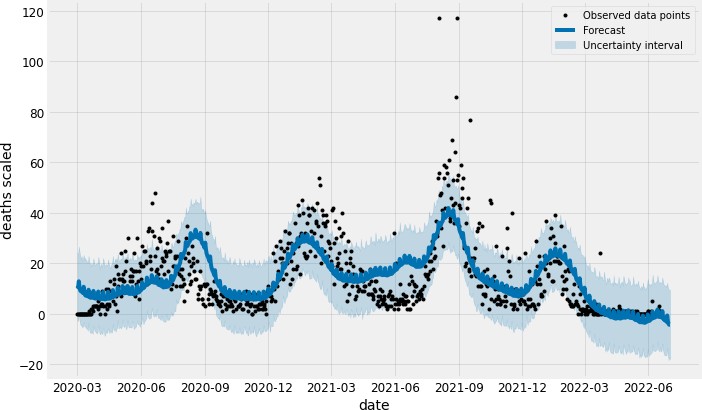
16.034133074515896

1. :prophet\_err=(np.mean(err\_list), err)

#### Results

1. :fig1=model.plot(forecasts, include\_legend=True, ylabel="deaths scaled",␣

*‹→*xlabel="date")



1. :fig2=model.plot\_components(forecasts, weekly\_start=0)
2. :ind=0

train\_slice=train.iloc[ind:] xticks=train\_slice.index.tolist() in\_forecast=model.predict()["yhat"]

plt.plot(xticks, train\_slice.values) plt.plot(xticks, in\_forecast[ind:]) plt.legend(["Actual data","forecasted data"])

plt.title("Actual daily historical data vs Predicted")

1. :Text(0.5, 1.0, 'Actual daily historical data vs Predicted')

[54]:ind=-200

train\_slice=train.iloc[ind:] xticks=train\_slice.index.tolist()

plt.plot(xticks, train\_slice.values) plt.plot(xticks, in\_forecast[ind:]) plt.legend(["Actual data","forecasted data"])

plt.title(f"Actual daily historical data vs Predicted for{abs(ind)}days")

1. :Text(0.5, 1.0, 'Actual daily historical data vs Predicted for 200 days')

[55]:ind=-100

train\_slice=train.iloc[ind:] xticks=train\_slice.index.tolist()

n=len(holdout)+1

*# future = model.make\_future\_dataframe(periods=n)*

*# forecast = model.predict(future)["yhat"].iloc[-n+1:]*

forecasts["yhat"][forecasts["yhat"]<0]=0

forecast=[train\_slice.iloc[-1]]+forecasts["yhat"].iloc[-n+1:].tolist() holdout\_=[train\_slice.iloc[-1]]+holdout.iloc[:n-1].tolist() holdout\_xticks=pd.date\_range(start=xticks[-1], periods=n)

plt.plot(xticks, train\_slice.values) plt.plot(holdout\_xticks, holdout\_) plt.plot(holdout\_xticks, forecast)

plt.legend(labels=["previous data",f"{n-1}-day Actual",f"{n-1}-day␣

*‹→*forecast"])

plt.title(f"last{abs(ind)}days plus{n-1}days forecast and actual")

[56]:ind=-200

train\_slice=train.iloc[ind:] xticks=train\_slice.index.tolist()

n=15

*# forecast = model.predict(future)["yhat"].iloc[-n+1:] # forecast[forecast < 0] = 0*

forecast=[train\_slice.iloc[-1]]+forecasts["yhat"].iloc[-n+1:].tolist() holdout\_xticks=pd.date\_range(start=xticks[-1], periods=n).tolist()

plt.plot(xticks, train\_slice.values) plt.plot(holdout\_xticks, forecast)

*# plt.plot(range(len(holdout.iloc[:14])), holdout.iloc[:14].values)*

plt.legend(labels=["previous data",f"{n-1}-day forecast"]) plt.title(f"last{abs(ind)}days plus{n-1}days forecast and actual")

#### ML Models

**PROCEDURE** The procedure used for both ML models are the same and are out- lined below: - Hold out 2 weeks worth of the data for validation - Carry out Random Search Cross validation (with rolling origin) to select the optimal hyperparameters values. The procedures are as below: - A) create the list of values for each selected hyperparameters - B) create a rolling ori- gin cross validation of 15 folds with two weeks test set data with the TimeSeriesSplit class in the Scikit-Learn package - C) instanstiate the ML model Class (RandomForestRegres- sor, RandomForestRegressor) - D) pass A, B, C, D to the “RandomizedSearchCV” class in Scikit- Learn package - E) fit d to start the random search algorithm, the algorithm calculates the MSE value for each fold and returns the best model and best MSE error

* + - * To have a test of how well the model will perform with real data, the model is tested on the held out data and the MSE is calculated. If the error increased significantly, then the model will not be stable to real life data

1. :days=14

X,y=prepare\_ml\_data\_with\_features(cases\_day, deaths\_day) X\_train, X\_holdout=X.iloc[:-days], X.iloc[-days:]

y\_train, y\_holdout=y.iloc[:-days], y.iloc[-days:]

#### Random Forest Regressor

**Hyperrameters** (n\_estimators, max\_depth) #### Result MSE Mean after cross validation: 13.5719 MSE for heldout data: 36.9391

{'n\_estimators': [100, 550, 1000, 1450, 1900], 'max\_depth': [10, 20, 30, 40, 50,

*# Number of trees in random forest* n\_estimators=[int(x)forxinnp.linspace(start=100, stop=1900, num=5)] *# Number of features to consider at every split* max\_depth=[int(x)forxinnp.linspace(10,100, num=10)]+[None]

*# Create the random grid*

random\_grid={ 'n\_estimators': n\_estimators,

'max\_depth': max\_depth

}

print(random\_grid)

60, 70, 80, 90, 100, None]}

1. :rf=RandomForestRegressor(criterion= 'squared\_error', random\_state=seed)␣

*‹→# base model to tune*

*# Random search of parameters, using 5 fold rolling basis cross validation, # search across 80 different combinations, and use all available cores* rf\_random=RandomizedSearchCV(estimator=rf, param\_distributions=␣

*‹→*random\_grid, n\_iter=40, cv=TimeSeriesSplit(n\_splits=5, test\_size=14),

verbose=1, random\_state=seed, n\_jobs=-1)

*# Fit the random search model*

rf\_random.fit(X\_train, y\_train)

Fitting 5 folds for each of 40 candidates, totalling 200 fits [59]:RandomizedSearchCV(cv=TimeSeriesSplit(gap=0, max\_train\_size=None, n\_splits=5,

test\_size=14),

estimator=RandomForestRegressor(random\_state=0), n\_iter=40, n\_jobs=-1,

param\_distributions={'max\_depth': [10, 20, 30, 40, 50, 60,

70, 80, 90, 100, None],

'n\_estimators': [100, 550, 1000, 1450,

1900]},

random\_state=0, verbose=1)

1. :rf\_random.best\_params\_ [60]:{ 'n\_estimators': 1000, 'max\_depth': 10}
2. :pd.DataFrame(rf\_random.cv\_results\_).iloc[:,6:].sort\_values(by=␣

*‹→*"rank\_test\_score").head()

1. :params split0\_test\_score \

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | {'n\_estimators': | 1000, | 'max\_depth': | 10} | -0.502466 | |
| 17 | {'n\_estimators': | 1000, | 'max\_depth': | 20} | -0.544937 | |
| 19 | {'n\_estimators': | 1000, | 'max\_depth': | 60} | -0.544937 | |
| 35 | {'n\_estimators': | 1000, | 'max\_depth': | 40} | -0.544937 | |
| 6 | {'n\_estimators': | 1000, | 'max\_depth': | 70} | -0.544937 | |
|  | split1\_test\_score | split2\_test\_score | | split3\_test\_score \ | | |
| 5 | -6.393773 | -23.472532 | | -27.440136 | | |
| 17 | -6.551951 | -22.943507 | | -27.906734 | | |
| 19 | -6.541412 | -22.943507 | | -27.906734 | | |
| 35 | -6.541412 | -22.943507 | | -27.906734 | | |
| 6 | -6.541412 | -22.943507 | | -27.906734 | | |
|  | split4\_test\_score | mean\_test\_score | | std\_test\_score | | rank\_test\_score |
| 5 | -10.050515 | -13.571884 | | 10.247792 | | 1 |
| 17 | -10.452258 | -13.679877 | | 10.217336 | | 2 |
| 19 | -10.480577 | -13.683433 | | 10.217026 | | 3 |
| 35 | -10.480577 | -13.683433 | | 10.217026 | | 3 |
| 6 | -10.480577 | -13.683433 | | 10.217026 | | 3 |

1. :best\_random=rf\_random.best\_estimator\_ random\_err=MSE(best\_random.predict(X\_holdout), y\_holdout)

print(f"model\_err:{random\_err:.2f}") model\_err: 36.94

1. :rf\_random.best\_score\_

[63]:-13.57188426643712

1. :RF\_err=(abs(rf\_random.best\_score\_), random\_err)
2. :pd.DataFrame(rf\_random.best\_estimator\_.feature\_importances\_, index=X.columns).

*‹→*sort\_values(by=0, ascending=1)\

.rename(columns={0:"importance"}).plot(kind="barh") [65]:<AxesSubplot:>

#### Result

1. :ind=0

train\_slice=y\_train.iloc[ind:] xticks=train\_slice.index.tolist()

plt.plot(xticks, train\_slice.values)

plt.plot(xticks, rf\_random.best\_estimator\_.predict(X\_train)[ind:]) plt.legend(["Actual data","forecasted data"])

plt.title("Actual daily historical data vs Predicted")

1. :Text(0.5, 1.0, 'Actual daily historical data vs Predicted')

[67]:ind=-200

train\_slice=y\_train.iloc[ind:] xticks=train\_slice.index.tolist()

plt.plot(xticks, train\_slice.values)

plt.plot(xticks, rf\_random.best\_estimator\_.predict(X\_train)[ind:]) plt.legend(["Actual data","forecasted data"])

plt.title(f"Actual daily historical data vs Predicted for{abs(ind)}days")

1. :Text(0.5, 1.0, 'Actual daily historical data vs Predicted for 200 days')

[68]:ind=-100

train\_slice=y\_train.iloc[ind:] xticks=train\_slice.index.tolist()

n=len(holdout)+1

forecast=forecast=[train\_slice.iloc[-1]]+rf\_random.best\_estimator\_.

*‹→*predict(X\_holdout).tolist()

holdout\_=[train\_slice.iloc[-1]]+holdout.iloc[:n-1].tolist() holdout\_xticks=pd.date\_range(start=xticks[-1], periods=n)

plt.plot(xticks, train\_slice.values) plt.plot(holdout\_xticks, holdout\_) plt.plot(holdout\_xticks, forecast)

plt.legend(labels=["previous data",f"{n-1}-day Actual",f"{n-1}-day␣

*‹→*forecast"])

plt.title(f"last{abs(ind)}days plus{n-1}days forecast and actual") [68]:Text(0.5, 1.0, 'last 100 days plus 14 days forecast and actual')

#### Light Gradient Boosting Machine (LGBM)

**Hyperparameters** (n\_estimators, max\_depth, learning\_rate) #### Result MSE Mean after cross validation: 4.6515

MSE for heldout data: 32.3548

1. :lgbm=LGBMRegressor(random\_state=seed)
2. :lgbm.get\_params()
3. :{ 'boosting\_type': 'gbdt', 'class\_weight': None, 'colsample\_bytree': 1.0, 'importance\_type': 'split', 'learning\_rate': 0.1,

'max\_depth': -1,

'min\_child\_samples': 20,

'min\_child\_weight': 0.001,

'min\_split\_gain': 0.0,

'n\_estimators': 100,

'n\_jobs': -1,

'num\_leaves': 31, 'objective': None, 'random\_state': 0,

'reg\_alpha': 0.0,

'reg\_lambda': 0.0, 'silent': 'warn',

'subsample': 1.0,

'subsample\_for\_bin': 200000,

'subsample\_freq': 0}

1. :n\_estimators=[int(x)forxinnp.linspace(start=10, stop=1000, num=10)] *# Number of features to consider at every split* max\_depth=[int(x)forxinnp.linspace(10,100, num=10)]+[None] learning\_rate=[0.01,0.1]

*# Create the random grid*

random\_grid={ 'n\_estimators': n\_estimators,

'max\_depth': max\_depth, 'learning\_rate': learning\_rate

}

print(random\_grid)

{'n\_estimators': [10, 120, 230, 340, 450, 560, 670, 780, 890, 1000],

'max\_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None], 'learning\_rate':

[0.01, 0.1]}

1. :lgbm\_random=RandomizedSearchCV(estimator=lgbm, param\_distributions=␣

*‹→*random\_grid,

*‹→*test\_size=14),

*# Fit the random search model*

lgbm\_random.fit(X\_train, y\_train)

n\_iter=80, cv=TimeSeriesSplit(n\_splits=5,␣

verbose=1, random\_state=seed, n\_jobs=-1, scoring="neg\_mean\_squared\_error")

Fitting 5 folds for each of 80 candidates, totalling 400 fits

[72]:RandomizedSearchCV(cv=TimeSeriesSplit(gap=0, max\_train\_size=None, n\_splits=5, test\_size=14),

estimator=LGBMRegressor(random\_state=0), n\_iter=80, n\_jobs=-1,

param\_distributions={'learning\_rate': [0.01, 0.1],

'max\_depth': [10, 20, 30, 40, 50, 60,

70, 80, 90, 100, None],

'n\_estimators': [10, 120, 230, 340, 450,

560, 670, 780, 890,

1000]},

random\_state=0, scoring='neg\_mean\_squared\_error', verbose=1) [73]:lgbm\_random.best\_params\_ [73]:{ 'n\_estimators': 670, 'max\_depth': 20, 'learning\_rate': 0.01}

1. :pd.DataFrame(lgbm\_random.cv\_results\_).iloc[:,5:].sort\_values(by=␣

*‹→*"rank\_test\_score").head()

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [74]:param\_max\_depth | | param\_learning\_rate | | | \ |  | | | | |
| 48 | | 20 | | | 0.01 |
| 3 | | 60 | | | 0.1 |
| 15 | | 90 | | | 0.1 |
| 16 | | 100 | | | 0.1 |
| 60 | | 40 | | | 0.1 |
|  |  | |  |  | | params | | split0\_test\_score | | \ |
| 48 | {'n\_estimators': | | 670, | 'max\_depth': | | 20, 'learni... | | -1.845234 | |  |
| 3 | {'n\_estimators': | | 120, | 'max\_depth': | | 60, 'learni... | | -1.693990 | |  |
| 15 | {'n\_estimators': | | 120, | 'max\_depth': | | 90, 'learni... | | -1.693990 | |  |
| 16 | {'n\_estimators': | | 120, | 'max\_depth': | | 100, 'learn... | | -1.693990 | |  |
| 60 | {'n\_estimators': | | 120, | 'max\_depth': | | 40, 'learni... | | -1.693990 | |  |
|  | split1\_test\_score | | split2\_test\_score | | | | split3\_test\_score | | \ | |
| 48 | -0.720517 | | -2.998532 | | | | -5.084756 | |  | |
| 3 | -0.925593 | | -2.925743 | | | | -6.604746 | |  | |
| 15 | -0.925593 | | -2.925743 | | | | -6.604746 | |  | |
| 16 | -0.925593 | | -2.925743 | | | | -6.604746 | |  | |
| 60 | -0.925593 | | -2.925743 | | | | -6.604746 | |  | |
|  | split4\_test\_score | | mean\_test\_score | | | std\_test\_score | | rank\_test\_score | | |
| 48 | -12.608317 | | -4.651471 | | | 4.232226 | | 1 | | |
| 3 | -11.229880 | | -4.675990 | | | 3.812997 | | 2 | | |
| 15 | -11.229880 | | -4.675990 | | | 3.812997 | | 2 | | |
| 16 | -11.229880 | | -4.675990 | | | 3.812997 | | 2 | | |
| 60 | -11.229880 | | -4.675990 | | | 3.812997 | | 2 | | |

1. :best\_lgbm=lgbm\_random.best\_estimator\_ lgbm\_err=MSE(best\_lgbm.predict(X\_holdout), y\_holdout)

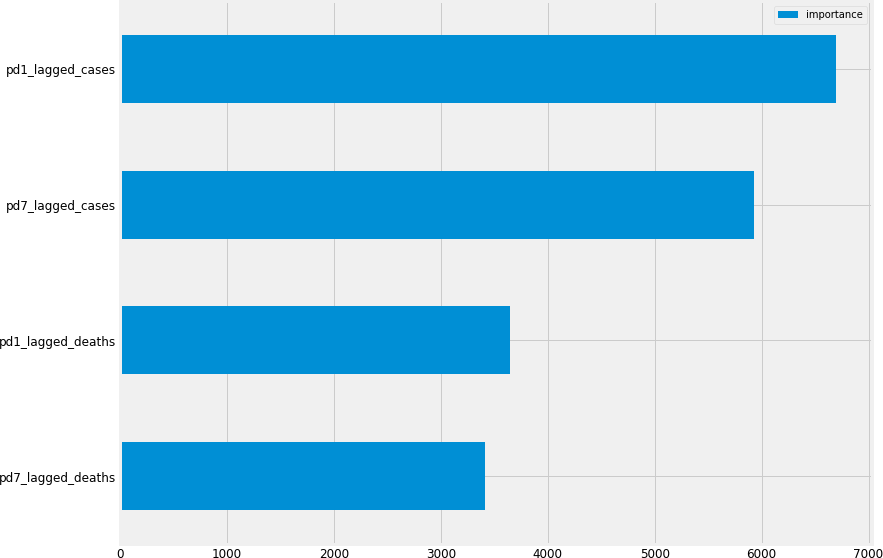
print(f"model\_err:{lgbm\_err:.2f}") model\_err: 32.35

1. :lgbm\_random.best\_score\_

[76]:-4.651470895367296

1. :LGBM\_err=(abs(lgbm\_random.best\_score\_), lgbm\_err)
2. :pd.DataFrame(lgbm\_random.best\_estimator\_.feature\_importances\_, index=X.

*‹→*columns).sort\_values(by=0, ascending=1)\

.rename(columns={0:"importance"}).plot(kind="barh") [78]:<AxesSubplot:>

#### Result

1. :ind=0

train\_slice=y\_train.iloc[ind:] xticks=train\_slice.index.tolist()

plt.plot(xticks, train\_slice.values)

plt.plot(xticks, lgbm\_random.best\_estimator\_.predict(X\_train)[ind:]) plt.legend(["Actual data","forecasted data"])

plt.title("Actual daily historical data vs Predicted")

1. :Text(0.5, 1.0, 'Actual daily historical data vs Predicted')

[80]:ind=-200

train\_slice=y\_train.iloc[ind:] xticks=train\_slice.index.tolist()

plt.plot(xticks, train\_slice.values)

plt.plot(xticks, lgbm\_random.best\_estimator\_.predict(X\_train)[ind:]) plt.legend(["Actual data","forecasted data"])

plt.title(f"Actual daily historical data vs Predicted for{abs(ind)}days")

1. :Text(0.5, 1.0, 'Actual daily historical data vs Predicted for 200 days')

[81]:ind=-100

train\_slice=y\_train.iloc[ind:] xticks=train\_slice.index.tolist()

n=len(holdout)+1

forecast=forecast=[train\_slice.iloc[-1]]+lgbm\_random.best\_estimator\_.

*‹→*predict(X\_holdout).tolist()

holdout\_=[train\_slice.iloc[-1]]+holdout.iloc[:n-1].tolist() holdout\_xticks=pd.date\_range(start=xticks[-1], periods=n)

plt.plot(xticks, train\_slice.values) plt.plot(holdout\_xticks, holdout\_) plt.plot(holdout\_xticks, forecast)

plt.legend(labels=["previous data",f"{n-1}-day Actual",f"{n-1}-day␣

*‹→*forecast"])

plt.title(f"last{abs(ind)}days plus{n-1}days forecast and actual") [81]:Text(0.5, 1.0, 'last 100 days plus 14 days forecast and actual')

[ ]:

### list of MSE errors

[82]:[ARIMA\_err, prophet\_err, RF\_err, LGBM\_err]

[82]:[(38.34063948227698, 1.1556779260950596),

(97.41006132803328, 16.034133074515896),

(13.57188426643712, 36.93914157488367),

(4.651470895367296, 32.3547937494093)]

## =================Deployment===================

To showcase the work done and to make available the prediction model making weekly covid-19 deaths prediction to stakeholders, a webapp will be built with a python open-source app frame- work, Streamlit. Streamlit is an easy to use library to create simple webapps for reporting by turning python scripts into webapps “https://streamlit.io”. With Streamlit little frontend webdevelopment knowledge is needed as the library abstracts it all away as python functions that can be called to perform the same task.

Streamlit also provides a means to deploy the built webapp for free on their cloud infrastructure which makes it a suitable choice for this task.

The ARIMA model which is the best model ascertained after this research work will be deployed using streamlit. The webapp will also make predictions weekly after fetching new data from John Hopkins repo and retraining the ARIMA model with the same parameters used in this research work. The procedures are as follows.

On Every Sundays: - download the latest covid-19 deaths result from John hopkins - retrain the model including the new datapoints - make forecast for each day of the upcoming week.

### File structure

deployment

utils.py

deploy.py

requirements.txt

#### ============ requirements.txt ==============

pandas

streamlit

statsmodels

#### ================= utils.py ===============================

from math import ceil

from datetime import timedelta, datetime as dt

import pandas as pd

import streamlit as st

import altair as alt

from statsmodels.tsa.arima.model import ARIMA

wafr\_countries = {

    "Benin": "BEN",

    "Burkina Faso": "BFA",

    "Cabo Verde": "CPV",

    "Cote d'Ivoire": "CIV",

    "Gambia": "GMB",

    "Ghana": "GHA",

    "Guinea": "GIN",

    "Guinea-Bissau": "GIN",

    "Liberia": "LBR",

    "Mali": "MLI",

    "Mauritania": "MRT",

    "Niger": "NER",

    "Nigeria": "NGA",

    "Senegal": "SEN",

    "Sierra Leone": "SLE",

    "Togo": "TGO",

}

ARIMA\_ORDER = (1, 1, 3)

DAYS = 7

def get\_sunday\_date():

    weekday = dt.today().weekday()

    diff = abs(-1 - weekday)

    sunday = dt.today() - timedelta(days=diff)

    return sunday.isoformat()[:10]

def get\_data(url, sunday):

    data = pd.read\_csv(url)

    data = data.set\_index(keys="Country/Region").drop(

        columns=["Province/State", "Lat", "Long"]

    )  # keep only the timeseries columns

    data.columns = pd.to\_datetime(data.columns)

    wafr\_data = data.loc[wafr\_countries.keys()].loc[:, "2020-03-01" : sunday()]

    return wafr\_data

@st.cache

def prepare\_model\_data(wafr\_data):

    wafr\_data = wafr\_data.sum()

    # timeseries are cummulative sum, make a 1-time difference to show actual amount per day

    wafr\_data = (wafr\_data - wafr\_data.shift(1)).fillna(0)

    wafr\_data[wafr\_data < 0] = 0

    return wafr\_data

@st.cache

def get\_last\_n\_days\_data(df, n=50, forecast=None):

    df = df.iloc[-n:].reset\_index()

    df.columns = ["date", "deaths"]

    df["kind"] = "previous"

    if forecast is None:

        return df

    forecast = forecast.to\_frame().reset\_index()

    forecast.columns = ["date", "deaths"]

    forecast["kind"] = "forecast"

    join = df.iloc[-1].copy()

    join["kind"] = "forecast"

    df = df.append(join).append(forecast, ignore\_index=True)

    return df

@st.cache

def make\_forecast(df):

    model = ARIMA(df, order=ARIMA\_ORDER).fit()

    forecast = model.forecast(DAYS)

    forecast = forecast.apply(lambda x: ceil(x))

    forecast[forecast < 0] = 0

    return forecast

# def altair\_plot(base, fields):

#     highlight = alt.selection(

#         type="single", on="mouseover", fields=fields, nearest=True

#     )

#     points = (

#         base.mark\_circle()

#         .encode(opacity=alt.value(0))

#         .add\_selection(highlight)

#         .properties(width=600)

#     )

#     lines = base.mark\_line().encode(

#         size=alt.condition(~highlight, alt.value(1), alt.value(3))

#     )

#     chart = points + lines

#     return st.altair\_chart(chart, use\_container\_width=True)

def altair\_plot(df, x, y, color):

    line = alt.Chart(df).mark\_line().encode(x=x, y=y, color=color)

    # Create a selection that chooses the nearest point & selects based on x-value

    nearest = alt.selection(

        type="single", nearest=True, on="mouseover", fields=[x[:-2]], empty="none"

    )

    # Transparent selectors across the chart. This is what tells us

    # the x-value of the cursor

    selectors = (

        alt.Chart(df)

        .mark\_point()

        .encode(x=x, opacity=alt.value(0),)

        .add\_selection(nearest)

    )

    # Draw points on the line, and highlight based on selection

    points = line.mark\_point().encode(

        opacity=alt.condition(nearest, alt.value(1), alt.value(0))

    )

    # Draw text labels near the points, and highlight based on selection

    text = line.mark\_text(align="left", dx=5, dy=-5).encode(

        text=alt.condition(nearest, y, alt.value(" "))

    )

    # Draw a rule at the location of the selection

    rules = alt.Chart(df).mark\_rule(color="gray").encode(x=x,).transform\_filter(nearest)

    # Put the five layers into a chart and bind the data

    chart = alt.layer(line, selectors, points, rules, text).properties(

        width=600, height=300

    )

    return st.altair\_chart(chart.interactive(), use\_container\_width=True)

def plot\_forecast(df):

    return altair\_plot(df, "date:T", "deaths:Q", "kind:N")

def plot\_history\_data(df):

    return altair\_plot(df, x="Date:T", y="Deaths:Q", color="Country:N")

=============== deploy.py ========================

from utils import \*

URL = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse\_covid\_19\_data/csse\_covid\_19\_time\_series/time\_series\_covid19\_deaths\_global.csv"

df = get\_data(URL, get\_sunday\_date)

df\_day = prepare\_model\_data(df).to\_frame()

forecast = make\_forecast(df\_day)

prev\_50 = get\_last\_n\_days\_data(df\_day, forecast=forecast)

st.title("Technological University of the Shannon: Midlands Midwest")

bar = st.sidebar

option = bar.selectbox(

    "Select an option", ("7-day death cases forecast", "death cases in West Africa"), 0

)

if option == "7-day death cases forecast":

    sunday = get\_sunday\_date()

    next\_sunday = (dt.fromisoformat(sunday) + timedelta(days=7)).isoformat()[:10]

    st.write("## A week forecast of death cases in West Africa")

    st.write(f"\_\_from {sunday} to {next\_sunday}\_\_")

    st.write("\_\_updated every sunday\_\_")

    plot\_forecast(prev\_50)

    st.write("zoom in on chart; double click to reset chart")

    st.write(f"### Total deaths forecast for the week is {forecast.sum()}")

if option == "death cases in West Africa":

    st.write(f"## Covid-19 death trend in West Africa by country")

    st.write("\_\_updated every sunday\_\_")

    countries = st.multiselect(

        "Choose countries", ["All"] + list(df.index), ["Nigeria", "Ghana"]

    )

    if not countries:

        st.error("Please select at least one country.")

    else:

        if "All" in countries:

            countries = df.index

        new\_df = df.loc[countries].reset\_index()

        new\_df = pd.melt(new\_df, id\_vars=["Country/Region"])

        new\_df.columns = ["Country", "Date", "Deaths"]

        plot\_history\_data(new\_df)

        st.write("zoom in on chart; double click to reset chart")