Masters of Science (MSc) in Business Analytics

Forecasting Healthcare Demand for Emerging Infectious Diseases: A Data Driven Approach

Degree: MSc. Business Analytics

Department: Business and Management

Aston Business school

Name: Sayali Sanjay Shevkar

Student ID: 230284735

Candidate no: 749403

Supervisor: Dr Akram Dehnokhalaji.

Declaration:

Acknowledgments

I wish to sincerely thank my supervisor, Dr. Akram, for her invaluable guidance and feedback during this project. Her consistent support and expertise have been essential from the very beginning of this project, and I am incredibly grateful for her time and commitment.

I am also thankful to my friends for their constant support, which made this journey much more enjoyable. Your presence, whether during times of challenge or celebration, has been a source of strength and motivation.

Lastly, I owe a deep sense of gratitude to my family. To my parents, for their endless support, love, and belief in me from the start of my academic journey, and to my siblings for always being there with a listening ear or a word of encouragement. Your unwavering support has made all the difference.

Abstract

This research examines the effectiveness of time-series forecasting models—specifically ARIMA and LSTM (with and without LLM integration)—to predict key healthcare demand variables during the COVID-19 pandemic using data from NHS England. The study focuses on forecasting variables such as the number of patients, hospital admissions, mechanical ventilation bed usage, and registered deaths due to COVID.

The research employs ARIMA for capturing linear trends and LSTM models for handling complex, non-linear relationships. ARIMA showed strong and consistent performance across several variables, especially in stable and predictable trends, while LSTM with LLM, with a MAPE of 19.24% for the number of patients and 31.08% for patients in the hospital, highlighted its strengths in complex contexts but faced challenges with overfitting and computational complexity. While ARIMA showed stable performance for linear trends, LSTM without LLM models proved more effective in handling non-linear patterns, despite struggling with sudden spikes. The results suggest that ARIMA is suitable for predictable, stable trends like mechanical ventilation bed usage, while LSTM models should be applied to more dynamic datasets with irregular patterns. Practical recommendations include using ARIMA for simpler scenarios and LSTM for complex variables, with future research focusing on optimizing LSTM models to reduce errors and improve adaptability.

Table of Content:

Table of Figures:

List of Abbreviations and acronyms

- ADF Test: Augmented Dickey-Fuller Test
- ARIMA: Autoregressive Integrated Moving Average
- LSTM: Long Short-Term Memory
- LSTM with LLM: Long Short-Term Memory with Large Language Model
- LSTM without LLM: Long Short-Term Memory without Large Language Model
- GRU: Gated Recurrent Unit
- lb_stat: Ljung-Box Statistic
- lb_pvalue: Ljung-Box p-value
- AIC: Akaike Information Criterion, BIC: Bayesian Information Criterion
- MSE: Mean Squared Error
- RMSE: Root Mean Squared Error
- MAPE: Mean Absolute Percentage Error
- GDPR: General Data Protection Regulation
- MinMaxScaler: A scaling method from the scikit-learn library that normalizes data between 0 and 1.
- SEIR -Susceptible-Exposed-Infectious-Recovered
- RNNs: Recurrent neural networks
- EHR: Electronic Health Records
- CORD-19: COVID-19 Open Research Dataset.

1. Chapter: Introduction

This chapter outlines the study's background and core elements, providing an overview of its foundational context. It also highlights the significance of the study, details the project's objectives along with relevance and scope.

1.1 Background of the study

The global healthcare system has faced unprecedented challenges during pandemics, especially the COVID-19 pandemic, which exposed critical gaps in healthcare demand forecasting and resource management. Without accurate predictions, healthcare systems risk becoming overwhelmed, leading to shortages of vital resources such as ventilators, hospital beds, and staff, which can compromise patient care and increase mortality rates (Singh RK, 2020).

Traditional models like ARIMA have been widely used in healthcare forecasting but struggle to handle non-linear patterns under volatile conditions. Advanced models like Long Short-Term Memory (LSTM), particularly when combined with Large Language Models (LLM), offer the ability to incorporate real-time, unstructured data, enhancing accuracy and responsiveness of predictions((Sujata Dash, 2021); (Pal, 2022)).

1.2 Research Problem

Although predictive modeling has advanced, challenges persist in achieving accuracy and scalability, particularly with incomplete or inconsistent healthcare data (Singh RK, 2020). While ARIMA is effective for linear trends, it fails to capture complex, non-linear patterns seen during pandemics (Pal, 2022) .Conversely, LSTM models are better equipped to manage these complexities but require substantial computational resources and risk overfitting (Ma, 2021) . The lack of hybrid models combining ARIMA with advanced machine learning, like LSTM and LLM, limits the development of highly accurate real-time predictive models (Mohammad Fattahi, 2023). Addressing these gaps is crucial for developing reliable forecasting tools for healthcare systems like NHS England to manage resources efficiently during crises.

1.3 Research Focus

This study compares the application of three predictive modeling methods—ARIMA, LSTM without LLM (GRU), and LSTM with LLM—in forecasting healthcare demand during pandemics. By evaluating accuracy and computational efficiency, this research aims to provide insights into how healthcare systems, such as the NHS, can better prepare for future pandemics using advanced models. This study also explores how incorporating language models with LSTM enhances forecasting by integrating unstructured data from diverse sources (Elsheikh, et al., 2021).

1.4 Study Significance

The significance of this study extends beyond NHS England to healthcare systems globally, especially in regions that may lack robust data infrastructures but face similar challenges in managing healthcare resources during crises. Accurate forecasting is crucial for healthcare systems to avoid resource shortages, reduce staff burnout, and improve patient outcomes during crises (Rohit Salgotra, 2020). Advanced models like LSTM and LSTM with LLM offer a framework for more precise, context-aware healthcare demand forecasting. This research enhances the existing literature by integrating traditional statistical approaches with machine learning techniques to provide practical insights for policymakers and practitioners in optimizing healthcare resource allocation and enhancing pandemic preparedness (Kumar, et al., 2021). This research is particularly timely given the evolving nature of pandemics, with emerging variants and healthcare systems striving to build resilience for future global health threats.

1.5 Objectives of the Study

To fulfil the aims of this research, the following objectives have been established:

- ❖ Conduct a literature review of predictive modeling in healthcare, focusing on ARIMA, LSTM, and hybrid approaches to identify highlight deficiencies in current models and recommend potential improvements for future research.

- ❖ Implement and evaluate ARIMA, LSTM without LLM (GRU), and LSTM with LLM models to forecast demand during pandemics to determine the accuracy and computational efficiency of each mode.

- ❖ Provide recommendations for healthcare systems, such as the NHS, on utilizing predictive models for pandemic preparedness to propose practical solutions for integrating predictive models into healthcare decision-making.

1.6 Relevance and Scope

The research is significant for healthcare planners and policymakers preparing for future pandemics or medical emergencies. For healthcare systems to allocate resources effectively, accurate forecasting is essential. This study compares ARIMA and LSTM models to offer practical insights for NHS England and similar organizations in making data-driven decisions. Although the dataset is limited to NHS England between April 2020 and May 2023, the findings have broader implications for other healthcare systems globally. By integrating LLM, this research provides a foundation for advanced forecasting strategies that combine numerical data with external, context-aware information.

2. Chapter: Literature Review

2.1. Introduction

The COVID-19 pandemic has underscored the critical need for effective predictive modeling techniques in healthcare. Predictive models play a vital role in resource management and demand forecasting, which are essential for healthcare systems to navigate through crises effectively. Traditional time-series models such as ARIMA have been the mainstay for forecasting, but the rapidly changing dynamics of healthcare demand during a pandemic necessitate more sophisticated approaches that can account for complex, non-linear patterns and incorporate real-time external data. This literature review critically examines existing research on ARIMA and advanced machine learning models like Long Short-Term Memory (LSTM) networks and hybrid approaches that integrate external data sources. The review also addresses current gaps in the literature, particularly in model integration, real-time data usage, and practical deployment challenges in healthcare systems such as the NHS. Furthermore, the review aims to highlight the advantages and limitations of various predictive models and identify areas for future research, contributing to a deeper understanding of how healthcare systems can better prepare for unforeseen surges in demand.

2.2. Theme 1: Predictive Modelling Techniques

2.2.1 Traditional Models: ARIMA and SARIMA

Autoregressive Integrated Moving Average (ARIMA) has been extensively applied for short- and medium-term forecasting in healthcare due to its simplicity and effectiveness for linear time-series data. At the onset of the COVID-19 pandemic, ARIMA was employed to forecast daily infection rates, resource utilization, and mortality trends across different countries, providing critical inputs for public health planning((Singh RK, 2020); (Adiga, 2020)). Despite its proven utility, ARIMA's reliance on stationarity and inability to capture sudden, non-linear changes limit its applicability in more complex healthcare settings (Pal, 2022). Seasonal ARIMA (SARIMA) variants have been explored to address some of these limitations by incorporating seasonality and cyclic patterns into the model. Studies indicate that these models perform well in scenarios with predictable seasonal variations, such as influenza outbreaks, but struggle when applied to datasets with unpredictable seasonal patterns, such as those observed during the pandemic (Kumar, et al., 2021).

However, ARIMA and SARIMA models are limited in capturing the broader contextual influences that impact healthcare demand, such as government policy changes, social behaviour, and vaccination rollouts. As a result, while ARIMA remains a reliable tool for short-term forecasting of linear trends, it is increasingly being supplemented by machine learning techniques that can handle greater complexity and non-linearity in the data (Adiga, 2020). This transition reflects a growing recognition within the literature that traditional statistical models, though effective in stable environments, may not be sufficient for forecasting during highly dynamic and uncertain conditions, like those presented by COVID-19.

2.2.2 Machine Learning Models: LSTM and GRNN

The advent of machine learning models like Long Short-Term Memory (LSTM) networks has revolutionized time-series forecasting by enabling the capture of long-term dependencies and non-linear patterns in healthcare data (Sujata Dash, 2021). Unlike ARIMA, which primarily focuses on linear relationships, LSTM models can learn complex temporal patterns and adapt to sudden changes, perfectly suited for dynamic scenarios like pandemics. LSTM's recurrent neural network (RNN) structure allows it to retain memory over time, effectively handling large time-series datasets. During the COVID-19 pandemic, LSTM models were successfully deployed to predict ICU admissions, ventilator demand, and hospitalizations in real time, significantly outperforming ARIMA in both accuracy and adaptability (Ma, 2021).

Despite these advantages, LSTM models are not without challenges. They are computationally intensive, requiring substantial resources for training and deployment, which can be a barrier in real-world healthcare settings, especially those with limited infrastructure (Brownlee, 2019). Furthermore, LSTM models are prone to overfitting when trained on small or inconsistent datasets, which was a common issue during the initial stages of the pandemic when data availability and quality varied widely. To mitigate this, techniques such as dropout regularization and early stopping have been used, but these methods add additional layers of complexity to model development and tuning (Sepp Hochreiter, 1997).

In response to these challenges, alternative neural network-based models, such as the General Regression Neural Network (GRNN), have been explored. GRNNs are particularly suited for real-time forecasting as they require less computational power and can manage non-linear relationships efficiently (Specht, 1991).

In comparative studies, GRNN outperformed LSTM in predicting non-linear trends in emergency department visits during the COVID-19 pandemic, highlighting its potential as a more accessible alternative for real-world applications (Duarte, 2021). However, GRNNs may lack the flexibility of LSTM in capturing long-term dependencies, suggesting that the choice between these models should be guided by the specific requirements of the forecasting task.

2.2.3 Hybrid Models and Ensemble Approaches

Integrated models that merge ARIMA with machine learning methods, such as Genetic Programming (GP) or XGBoost, have emerged as a promising alternative to standalone models. These models leverage ARIMA's ability to capture linear trends while using machine learning to model non-linear patterns ((Kaushik Shruti, 2020). For example, a study in India combined ARIMA with GP to forecast COVID-19 cases, resulting in a 40% reduction in prediction errors compared to using ARIMA alone (Rohit Salgotra, 2020). Implementing hybrid models can address certain limitations of standalone models by leveraging their combined strengths , leading to better performance in complex forecasting scenarios.

Similarly, ensemble models that integrate LSTM with ARIMA or other machine learning techniques have shown promise in healthcare forecasting. Ensemble models are particularly effective because they aggregate predictions from multiple models, reducing the risk of overfitting and enhancing overall robustness (Elsheikh, et al., 2021). For instance, a hybrid model combining LSTM, ARIMA, and XGBoost was used to forecast hospital bed occupancy during the peak of the pandemic, achieving higher accuracy than any single model (Kumar, et al., 2021).

However, hybrid and ensemble models also come with their own set of challenges. They require extensive data integration and preprocessing to ensure that each component model is trained on high-quality, consistent data. Moreover, the computational cost of training multiple models simultaneously can be prohibitive, particularly in resource-constrained settings like some NHS Trusts. This necessitates a careful balance between model complexity and practical applicability, as overly complex models may not always justify the marginal gains in predictive performance.

2.3. Theme 2: Data Sources and Integration

The effectiveness of predictive models relies on the quality and integration of data. During the COVID-19 pandemic, healthcare systems faced challenges in accessing real-time, high-quality data for effective predictions. Combining data sources, such as electronic health records (EHR), online platforms, and open resources, is essential for improving healthcare demand forecasting. For example, integrating web-based data like Google Trends into predictive models significantly enhanced real-time forecasting, allowing healthcare systems to adapt to changing trends more dynamically (Rabiolo A, 2021). Similarly, the use of open data resources, such as the COVID-19 Open Research Dataset (Wang, 2020) enabled continuous refinement of models as new data became available, thus improving the accuracy of predictions in fast-changing conditions (Alamo T, 2020).

Integrating real-time EHR data further improved resource allocation, ensuring that hospital beds and ventilators were efficiently distributed during COVID-19 surges (Mohammad Fattahi, 2023). This sentiment was echoed by

(Singh RK, 2020), who emphasized that continuous updates to ARIMA models enhanced predictive accuracy as the pandemic evolved.

Time-series models, such as ARIMA, can be further optimized by incorporating diverse, real-time data streams, including infection rates, hospitalization numbers, and mobility data (Kumar, et al., 2021). However, a lack of robust data infrastructure can undermine even sophisticated models, as highlighted (Rohit Salgotra, 2020), who found that hybrid approaches, like ARIMA with Genetic Programming (GP), were effective only when supported by high-quality, real-time data.

Moving forward, it is critical for healthcare systems like the NHS to develop better data infrastructure that supports real-time integration of EHR, web-based data, and open platforms. This will enable more accurate and responsive predictive modeling, ensuring better preparedness for future pandemics and efficient resource management.

2.4. Theme 3: Model Evaluation and Validation

Evaluating and validating predictive models is crucial to ensuring their reliability in real-world healthcare settings, especially during pandemics. Studies comparing different models have highlighted both successes and limitations in forecasting healthcare demand. For instance, (Duarte, 2021) compared ARIMA, Prophet, and General Regression Neural Network (GRNN) models for predicting emergency department demand during COVID-19 and found that GRNN outperformed the other models, particularly in capturing non-linear trends. This underscores the limitations of ARIMA, which, despite its widespread use, may struggle with complex healthcare data patterns. Similarly, Haneen (Haneen Alabdulrazzaq, 2021) demonstrated ARIMA's high accuracy in predicting COVID-19 cases in Kuwait but noted that it heavily relies on data stationarity, making it less adaptable to non-stationary data.

Ensemble models, which combine predictions from different techniques such as LSTM and ARIMA, have proven more robust. For example, (Kaushik Shruti, 2020) found that ensemble approaches outperformed individual models in terms of accuracy and reliability. Likewise, hybrid models that integrate ARIMA with Genetic Programming (GP) showed significant reductions in prediction errors by leveraging the strengths of both statistical and machine learning techniques (Rohit Salgotra, 2020).

Validating models using appropriate metrics is equally important. (Ma, 2021) demonstrated that using RMSE to evaluate an LSTM-Markov hybrid model resulted in a 60% reduction in errors compared to traditional LSTM models. (Singh RK, 2020) used MAPE to validate ARIMA models, highlighting their effectiveness for short-term forecasts during the pandemic. However, (John P.A. Ioannidis, 2022) argued that many models used during the pandemic failed to incorporate real-time validation, leading to discrepancies between predictions and actual outcomes, particularly in the long term.

The importance of continuous validation is critical, especially in dynamic healthcare environments. (Mohamadou, 2020) emphasized that models need to adapt as new data becomes available, and static models that lack real-time updates are ineffective in fast-changing conditions. This is particularly relevant for the NHS, which requires predictive models that can respond to evolving circumstances and provide accurate, up-to-date forecasts for resource allocation.

Overall, no single model is universally superior. The most effective approaches are those that combine traditional methods with machine learning and are continually validated with real-time data to ensure high accuracy in rapidly changing healthcare environments.

2.5. Theme 4: Impact on Healthcare Systems

The COVID-19 pandemic exerted tremendous pressure on healthcare systems globally, necessitating rapid adaptations to manage surging demand. Predictive models played a pivotal role in resource allocation and informed healthcare policy decisions. By anticipating needs for hospital beds, ventilators, and other critical resources, these models helped healthcare systems respond more effectively to the crisis.

For instance, (Mohammad Fattahi, 2023) introduced a multi-stage stochastic program to optimize resource allocation during pandemics. Assessed through real-world case studies, this model ensured that critical resources, such as ventilators and ICU beds, were allocated where they were needed most. Similarly, (Bertsimas, 2021) developed a data-driven optimization model that dynamically adjusted ventilator distribution based on real-time data, helping healthcare systems avoid shortages. Both studies highlight the importance of predictive modeling in resource management, especially for systems like the NHS.

Another approach, a Bayesian SEIR (Susceptible-Exposed-Infectious-Recovered) model developed by (Paul Birrell, 2021) provided real-time predictions of infection rates and mortality in the initial wave of COVID-19 in England. This enabled the NHS to allocate resources more effectively, preventing system overload. The study underscored how predictive models can offer critical foresight for resource distribution.

However, the impact of COVID-19 was not limited to pandemic-related care. Non-COVID-19 services were also severely disrupted. For example, a qualitative study by (Alsaeed Dalal, 2023) in Kuwait highlighted the psychological toll on healthcare professionals and the disruption of routine healthcare services, including delayed surgeries and reduced access to non-emergency care. This imbalance exacerbated healthcare disparities, emphasizing the need for predictive models that consider both pandemic and routine healthcare needs to maintain overall system functionality.

Predictive models have shown their utility beyond pandemics. (Abdur Rais, 2010) reviewed operations research applications in healthcare, demonstrating how models can optimize resource allocation and treatment planning even during routine operations. Applying these insights to pandemic scenarios can better equip healthcare systems to manage both pandemic-related and everyday healthcare demands.

Long before COVID-19, (Duley, 2005) addressed many of the challenges that healthcare systems faced during the pandemic. Their work emphasized the need for capacity building, strategic response planning, and addressing resource shortages. Although published before the COVID-19 crisis, their recommendations remain relevant, illustrating the enduring need for predictive models to guide both immediate responses and long-term system adaptations.

In conclusion, predictive models significantly impacted healthcare systems during COVID-19 by guiding resource management and preventing overloads. However, there remains a need for models that can balance pandemic-related care with routine healthcare demands, ensuring that systems like the NHS remain resilient and adaptable in future crises.

2.6. Theme 5: Policy and Ethical Considerations

The use of predictive models in healthcare brings significant policy and ethical considerations, particularly concerning resource distribution and public health planning. These models guide crucial decisions, making it essential to ensure they uphold fairness and transparency, especially during crises like pandemics.

For instance, (Sullivan, 2024) highlighted the ethical dilemmas involved in resource allocation during COVID-19. Although predictive models were valuable for distributing critical resources such as ventilators and ICU beds, they often placed healthcare providers in difficult positions, requiring them to prioritize certain patients over others. This raises questions about equity and fairness, which must be central to any healthcare model used in crisis management.

Similarly, (John P.A. Ioannidis, 2022) critiqued many predictive models used during the pandemic, noting that flawed assumptions and inaccurate data inputs sometimes led to misguided public health decisions, resulting in overestimating or underestimating resource needs. The authors called for greater transparency and continuous validation of models to ensure that policy decisions are based on reliable, real-time data.

A comparative analysis by (Adiga, 2020) further reinforced the need for healthcare systems to carefully select models that align with ethical principles. Their study illustrated how different modeling approaches can produce vastly different outcomes, impacting decisions on resource allocation and public health measures. This underscores the importance of using ethical frameworks to guide the choice and application of predictive models.

In addition, (Matt J. Keeling, 2021) addressed the ethical trade-offs involved in balancing healthcare demand with economic recovery. Their model provided insights on navigating the challenges of reopening economies while controlling infection rates, emphasizing the need to consider both public health and economic interests. Such decisions have profound ethical implications, as reopening too early could overwhelm healthcare systems, while delayed reopening may harm vulnerable populations economically.

Expanding on this discussion, (Abdur Rais, 2010) explored how operations research can optimize healthcare resources even in non-crisis situations, highlighting its relevance for routine and emergency scenarios alike. In contrast, (Erwin J. Khoo, 2020) focused on the ethical dilemmas arising from imposing movement restrictions and prioritizing care during pandemics. Both studies emphasize the need to incorporate ethical considerations into the design and implementation of predictive models to ensure just and equitable outcomes.

In conclusion, ensuring that predictive models are grounded in ethical principles is essential for guiding healthcare systems like the NHS through crises. Fairness, transparency, and equity must be prioritized in the algorithms that shape public health policies and resource allocation, ensuring that vulnerable populations are protected, and healthcare decisions are just equitable.

2.7. Theme 6: Future Directions and Emerging Trends

The future of predictive modeling in healthcare is set for continued progress, with increasing incorporation of artificial intelligence (AI), machine learning (ML), and hybrid approaches providing promising pathways to enhance healthcare preparedness during pandemics. As healthcare systems like the NHS aim to strengthen their adaptability and resilience, adopting these emerging trends will be crucial for improving forecasting accuracy and optimizing resource management.

AI and ML models, such as those explored by (Elsheikh, et al., 2021) have already outperformed traditional models like ARIMA in healthcare demand forecasting. These AI-driven models are particularly effective in capturing non-linear trends and processing large volumes of data, which is critical in rapidly evolving scenarios like pandemics. As AI technology advances, its role in predictive modeling will continue to grow, providing more accurate forecasts essential for timely decision-making.

Ensemble models that combine machine learning techniques, such as LSTM and ARIMA, further enhance prediction accuracy. (Kaushik Shruti, 2020) found that combining the strengths of both statistical and machine learning models results in a more flexible and accurate forecasting method, particularly in complex healthcare settings. This highlights the increasing trend toward hybrid models that integrate traditional and modern techniques to better address the complexities of healthcare demand forecasting.

The importance of robust data infrastructure to support these advanced models cannot be overstated (Rohit Salgotra, 2020) emphasized that even sophisticated models will fail without access to real-time, high-quality data. Their study on COVID-19 in India demonstrated that the success of hybrid models, such as ARIMA combined with Genetic Programming (GP), relied on the availability of consistent data streams. Similarly, (Kumar, et al., 2021) noted that continuous updates of data sources—including health records, infection rates, and mobility patterns—are essential for improving forecast accuracy.

Integrating AI with real-time data is also vital for enhancing forecasting capabilities. (Mohamadou, 2020) argued that improving access to open data resources and refining AI techniques can enable healthcare systems to develop more adaptive models that respond to changing conditions. Such integration will be key in ensuring that healthcare systems can predict and manage future crises more effectively. Looking ahead, hybrid methods that merge AI with optimization strategies, as demonstrated by (Bertsimas, 2021) and (Mohammad Fattahi, 2023), are expected to pave the way for enhanced healthcare resource management during pandemics. These models have already shown their ability to dynamically allocate resources, such as ventilators and hospital beds, based on real-time demand. As healthcare systems like the NHS continue to face resource constraints, adopting these advanced hybrid models will be essential for ensuring preparedness and efficiency in future public health emergencies.

In conclusion, the future of healthcare predictive modeling lies in the incorporation of AI, machine learning, and hybrid approaches, supported by robust data infrastructure. By leveraging these emerging trends, healthcare systems will be better equipped to manage both pandemic-related and routine healthcare demand, ensuring they remain resilient in the face of future crises.

2.8. Summary Table: Themes and Papers:

Theme	Related Papers
Predictive Modelling Techniques	(Singh RK, 2020), (Ilie OD, 2020), (Pal, 2022), (Adiga, 2020) (Yuehui Chen, 2005), (Sujata Dash, 2021), (Ma, 2021), (Kaushik Shruti, 2020), (Elsheikh, et al., 2021), (Rohit Salgotra, 2020), (Kumar, et al., 2021)
Data Sources and Integration	(Rabiolo A, 2021), (Alamo T, 2020), (Mohammad Fattahi, 2023), (Singh RK, 2020), (Kumar, et al., 2021), (Rohit Salgotra, 2020)
Model Evaluation and Validation	(Duarte, 2021) (Haneen Alabdulrazzaq, 2021), (Kaushik Shruti, 2020), (Rohit Salgotra, 2020), (Ma, 2021), (Singh RK, 2020), (John P.A. Ioannidis, 2022)
Impact on Healthcare Systems	(Mohammad Fattahi, 2023), (Bertsimas, 2021), (Paul Birrell, 2021), (Alsaeed Dalal, 2023), (Abdur Rais, 2010) (Matt J. Keeling, 2021)
Policy and Ethical Considerations	(Sullivan, 2024), (John P.A. Ioannidis, 2022), (Adiga, 2020) (Grote, 2022), (Abdur Rais, 2010), (Erwin J. Khoo, 2020)
Future Directions and Emerging Trends	(Elsheikh, et al., 2021), (Kaushik Shruti, 2020), (Rohit Salgotra, 2020) (Kumar, et al., 2021), (Mohamadou, 2020), (Bertsimas, 2021), (Mohammad Fattahi, 2023)

Figure 1 Summary Table: Themes and Papers

3. Chapter: Research Methodology

3.1. Research Framework

This research adopts a comprehensive quantitative design, utilizing both statistical and machine learning techniques to forecast healthcare demand accurately. Quantitative methods were chosen due to the factual nature of the data and the need to provide measurable, data-driven predictions for NHS England. The primary focus is to assess the healthcare system's capacity by analysing past trends and projecting key variables, including the number of patients in hospitals, the usage of mechanical ventilation beds, and registered deaths due to COVID-19.

Time series forecasting has gained prominence across various fields, including healthcare, due to its ability to model dynamic trends over time (Kumar, 2022). In healthcare, accurate forecasting is crucial for resource management and planning, particularly during unpredictable events like pandemics ((Singh RK, 2020); (Pal, 2022). However, existing models have shown limitations in capturing the complex patterns typically observed in such scenarios, highlighting the need for more advanced approaches.

ARIMA (Auto-Regressive Integrated Moving Average) is broadly used in healthcare for modeling linear relationships and handling seasonality ((Box, 2013), (Zhang, 2003)). The model's ability to accommodate structured time-dependent data has made it a popular choice for scenarios where data is relatively stable, such as mechanical ventilation usage or routine hospital admissions (Singh RK, 2020). However, ARIMA's reliance on stationary data and its difficulty in handling non-linear patterns limit its applicability in volatile healthcare scenarios, such as sudden spikes in patient admissions during pandemic waves ((Pal, 2022); (Mukherjee, 2022)).

LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) models were incorporated to address ARIMA's limitations by identifying long-term patterns and relationships in complex, sequential healthcare data (Sepp Hochreiter, 1997). These models are particularly effective in scenarios where abrupt changes and non-linear patterns dominate, as seen in dynamic patient admissions during the COVID-19 pandemic (Zachary C. Lipton, 2015). LSTM models have demonstrated success in predicting patient outcomes and managing healthcare data, making them suitable for non-linear forecasting (Pham, 2016).

The integration of LLM (Large Language Models) with LSTM enables a richer contextual understanding by incorporating unstructured data like public health announcements and policy changes, which significantly impact healthcare trends (Tom B. Brown, 2020). This hybrid approach leverages external, unstructured data sources, allowing the model to better adapt to fluctuating conditions and provide more accurate predictions during periods of high uncertainty (Paki, 2021).

Research Gap and Objective

Although ARIMA is effective for stable trends, and LSTM models are suited for complex patterns, the lack of hybrid models combining these approaches limits their predictive capacity across diverse healthcare variables (Mohammad Fattahi, 2023). This study addresses this gap by systematically comparing ARIMA, LSTM without LLM, and LSTM with LLM for healthcare demand forecasting, providing insights into which models are best suited for diverse types of healthcare data.

3.2. Data Collection Methods

Data for this research was sourced from NHS England's public database, covering the period from April 2020 to May 2023. The dataset underwent extensive preprocessing, including handling missing values and segmenting data based on distinct pandemic waves to ensure accurate trend analysis. Four key variables were chosen for modeling: number of patients, hospital admissions, mechanical ventilation usage, and registered deaths due to COVID-19. Data beyond December 2021 was excluded to mitigate the risk of overfitting due to inconsistencies, thus enhancing model robustness. This refined dataset includes comprehensive information from all NHS Trusts in England, making it highly applicable for developing reliable forecasting models.

Visualizations demonstrating key trends and the modified dataset are provided in Appendix A1 to A8(58). Additionally, the specific Python libraries utilized for building and evaluating ARIMA and LSTM models are detailed in Appendix B (65), offering a clear overview of the tools used throughout the analysis process.

3.3. Assumptions:

Below are the assumptions made to simplify the analysis and improve model performance:

- **Stationarity Assumption for ARIMA**: The dataset was transformed to achieve stationarity (Refer Appendix A8(64)), as ARIMA models require stable statistical attributes (Box, 2013).
- **Data Scaling for LSTM Models** Min-Max scaling was applied to standardize the range of variables for consistent model training (Brownlee, 2019).
- **No Seasonal Adjustments** The analysis did not apply seasonal adjustments, as pandemic-driven fluctuations showed short-term irregularities rather than clear seasonal patterns (Zohair Malki, 2020).
- **Residual Independence** Ljung-Box tests were conducted to confirm that residuals showed no significant autocorrelation, ensuring model adequacy (G. M. LJUNG, 1978).

3.4. Model evaluation methods:

To ensure the robustness and accuracy of the models, several evaluation techniques were employed:

3.4.1. Walk-Forward Testing

Walk-forward testing, also referred to as moving window testing, served as the primary approach for evaluating the predictive accuracy of the models. It is a commonly recognized method in time-series analysis, as it simulates real-world forecasting by partitioning the dataset into overlapping segments for training, testing, and validation. This approach updates the model progressively with new data, while reserving a portion for validation, thus offering a comprehensive and dynamic evaluation of model performance over time (Iebeling Kaastra, 1996).

3.4.2. Hyperparameter Tuning

Hyperparameter tuning was implemented to improve the performance of the LSTM and GRU models in this study. Variables such as learning rate, batch size, and the number of units in every layer are critical factors that influence the model's ability to generalize effectively. Since deep learning architectures like LSTM and GRU are highly sensitive to these settings, fine-tuning is necessary to prevent overfitting or underfitting and to achieve optimal performance (Ilemobayo, 2024).

According to (Ilemobayo, 2024), hyperparameter tuning is a crucial step in ensuring the accuracy and robustness of deep learning models used for time series forecasting. Their study highlights the importance of selecting appropriate values for key parameters to achieve optimal performance, including LSTM and GRU models. They also provide a detailed overview of the hyperparameter tuning process, outlining various methods such as grid search, random search, and Bayesian optimization, which help enhance model accuracy and efficiency by carefully adjusting critical parameters such as learning rate and batch size.

For ARIMA models, hyperparameter tuning involves selecting the optimal values for the model's three key parameters: the autoregressive term (p), the differencing term (d), and the moving average term (q). This process is typically achieved through methods like grid search, where various combinations of p, d, and q are evaluated to minimize error metrics such as the Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC). The aim of this tuning process is to balance model complexity and goodness of fit, ensuring that the ARIMA model can accurately capture both the trend and noise in the time series data without overfitting. Automated methods, such as Auto-ARIMA, are often used to streamline this process by systematically testing and selecting the best hyperparameter values based on predefined criteria (Brownlee, 2019).

3.4.3. Error Metrics and Model Validation

To assess and compare the models' performance, a range of error metrics were utilized, including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) (Botchkarev, 2019). These metrics provided a comprehensive view of the accuracy and reliability of each model in predicting key healthcare variables. Residual analysis, including the Ljung-Box test, was also performed on the ARIMA model to ensure the residuals were independent, validating the model's fitness (G. M. LJUNG, 1978).

This combination of techniques—walk-forward testing, hyperparameter tuning, and rigorous model evaluation through error metrics—ensures the reliability of the forecasting results and provides a detailed comparison between the three models. This structured approach guarantees that the findings are robust and suitable for practical application in healthcare resource planning.

3.5. Ethical Considerations

In terms of ethical considerations, the research adhered to GDPR regulations by anonymizing all data, ensuring patient privacy was maintained throughout the analysis (Office, 2020). Fairness in forecasting was a key concern, with efforts made to validate that the models did not introduce biases that could impact resource allocation (Grote, 2022). The use of LSTM with LLM raised additional concerns around AI accountability, addressed through maintaining human oversight in line with ethical AI guidelines ((Anna Jobin, 2019); (Luciano Floridi, 2018).

4. Chapter: Data Analysis Process

This chapter compares three different forecasting methods—ARIMA, LSTM without LLM, and LSTM with LLM—across four healthcare variables: Total Number of Patients, Number of Hospital Admissions, Number of Mechanical Ventilation Beds, and Registered Deaths due to COVID-19. Each method was evaluated using multiple error metrics, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE), to assess its forecasting precision and practical relevance. The goal is to identify which model performs best for each variable and derive actionable insights for healthcare planning.

4.1 ARIMA Analysis

This section focuses on the practical application of the ARIMA (Auto-Regressive Integrated Moving Average) model for forecasting healthcare demand during the COVID-19 pandemic. The ARIMA model was selected due to its capability to model linear dependencies and capture trends and seasonality in structured time-series data ((Singh RK, 2020); (Pal, 2022)). Given its flexibility in handling time-dependent data, ARIMA is particularly suitable for variables with stable trends, making it an ideal choice for predicting healthcare indicators like the Number of Patients, Patients in Hospital, Mechanical Ventilation Usage, and Registered Deaths due to COVID-19.

Using the Auto ARIMA function, the optimal parameters (p, d, q) were identified for each healthcare variable. Auto ARIMA automates the process of parameter selection, thereby reducing manual trial-and-error and ensuring that the model minimizes error metrics like AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) ((Timina Liu, 2020); (Mukherjee, 2022)). This automation also helps mitigate the risk of overfitting by selecting parameters that generalize well on unseen data. Furthermore, diagnostic tests like the Ljung-Box Test help ensure that the residuals of the model exhibit no significant autocorrelation, confirming the model's robustness (G. M. LJUNG, 1978).

The ARIMA analysis will be visually represented through various plots, including forecast plots, residual analysis, and walk-forward validation, to illustrate the model's performance in predicting the selected healthcare variables over time.

Following Figure 7 illustrates the entire workflow for implementing an ARIMA model, from initial data preparation to final forecasting.

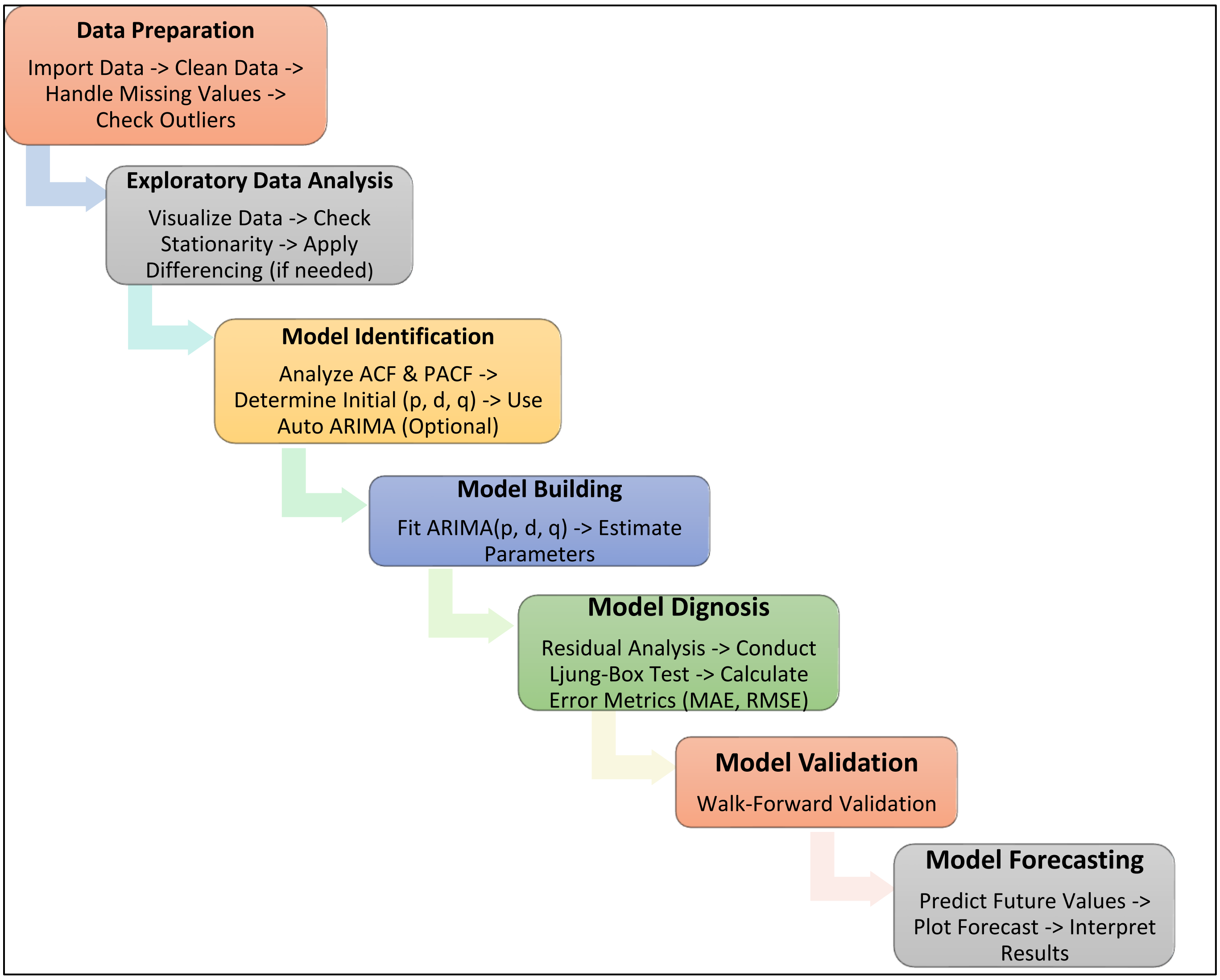

Data Preparation
Import Data -> Clean Data -> Handle Missing Values -> Check Outliers
Exploratory Data Analysis
Visualize Data -> Check Stationarity -> Apply Differencing (if needed)
Model Identification
Analyze ACF & PACF -> Determine Initial (p, d, q) -> Use Auto ARIMA (Optional)
Model Building
Fit ARIMA(p, d, q) -> Estimate Parameters
Model Dignosis
Residual Analysis -> Conduct Ljung-Box Test -> Calculate Error Metrics (MAE, RMSE)
Model Validation
Walk-Forward Validation
Model Forecasting
Predict Future Values -> Plot Forecast -> Interpret Results

Figure 2 ARIMA Analysis Workflow

Variable-Wise Analysis and Key Findings:

For each variable, a detailed analysis was conducted to evaluate ARIMA's strengths and limitations in capturing healthcare trends. Refer Appendix C (67)for analysis of one variable in python notebook.

The summary findings are presented below:

4.1.1. Number of Patients:

Optimal Parameters: (p=2, d=2, q=2)

Ljung-Box Test Results:

The test statistic (lb_stat = 2.9865) and high p-value (0.9817 > 0.05 shows the absence of significant autocorrelation in the residuals, validating the model's effectiveness with random noise patterns.

ARIMA Forecast Plot for No of Patients

The forecast plot presents the ARIMA model's predictions for future patient numbers, along with confidence intervals, highlighting a rising trend with some uncertainty in the predictions.

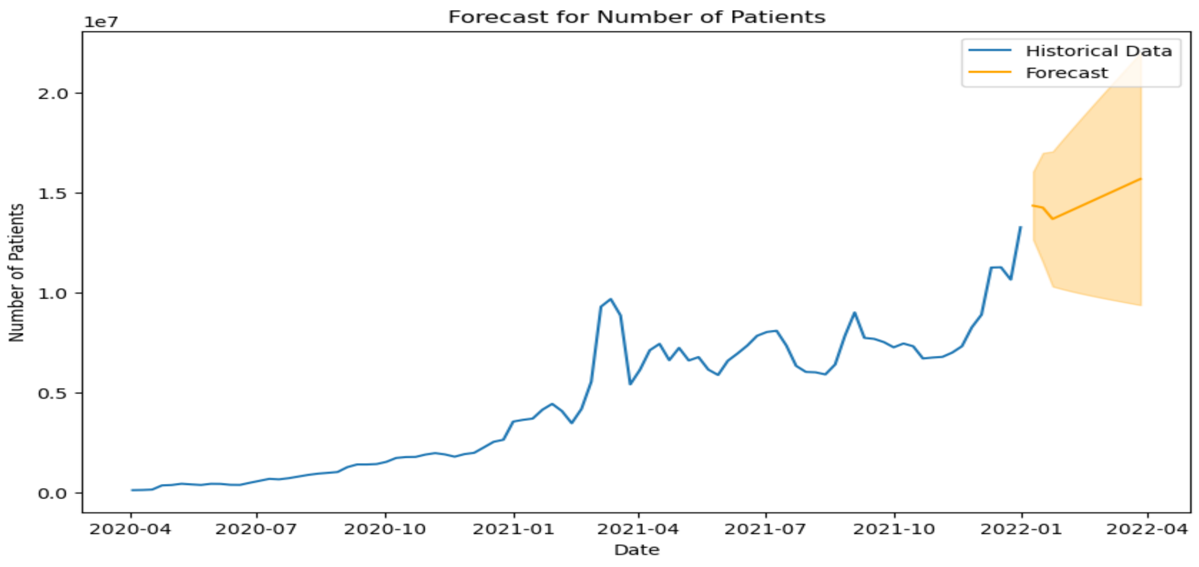


Figure 3 ARIMA Forecast for Variable 1: No of Patients

Walk-Forward Validation Plot:

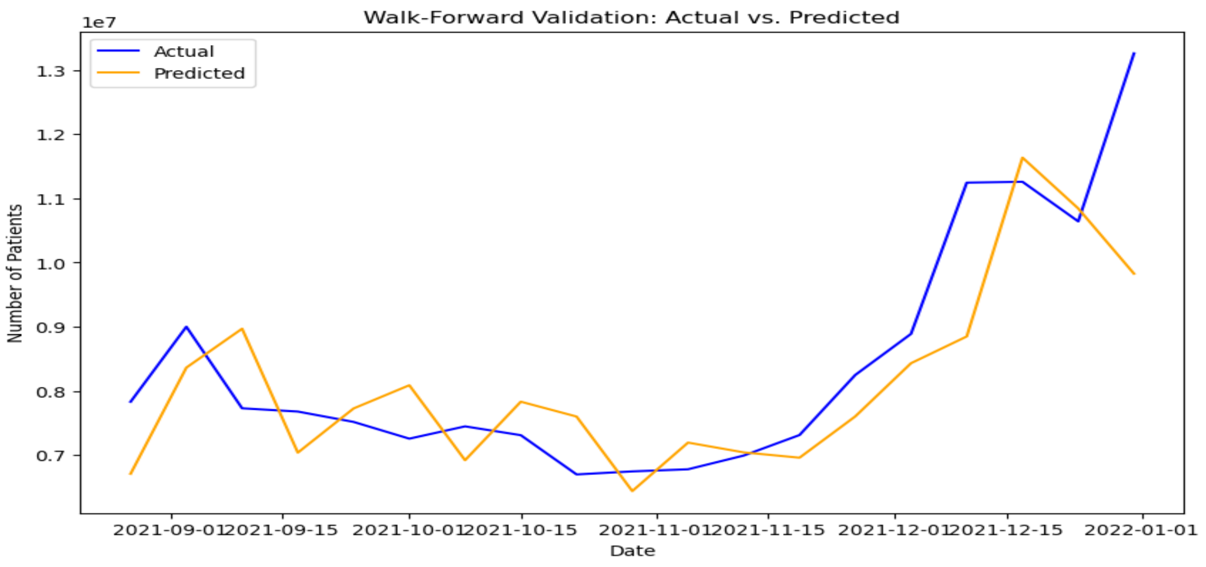


Figure 4 Walk-Forward Validation plot for Variable 1: No of Patients

This plot compares the actual number of patients with the predicted values from the ARIMA model over time, showing that the model underperforms during periods of sudden increases.

Residual Plot of Variable 1: No of Patients

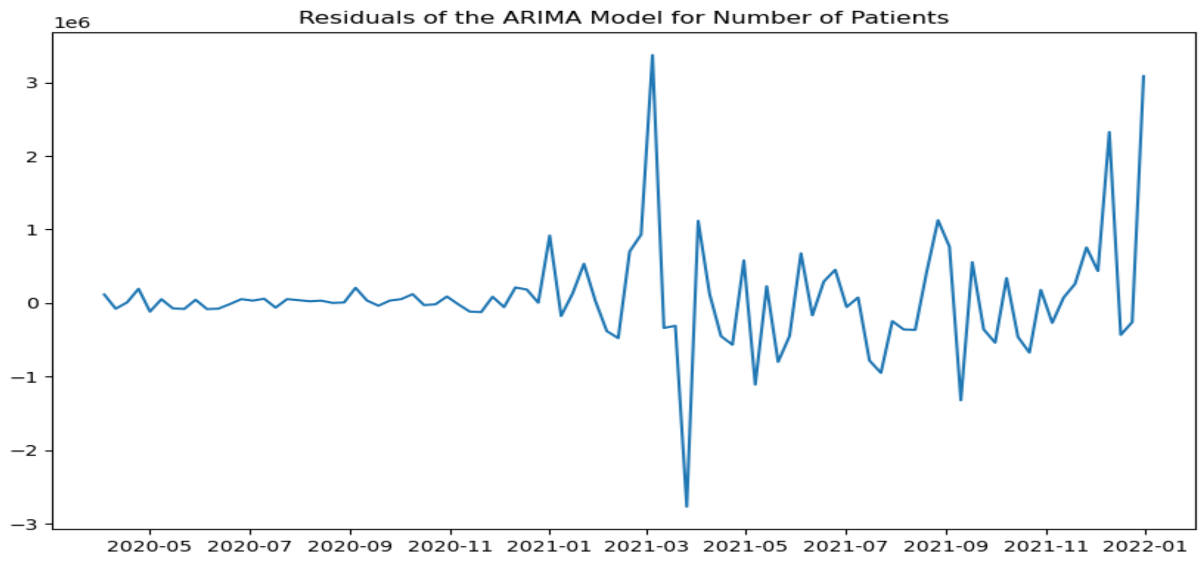


Figure 5 Residual Plot of Variable 1: No of Patients

This plot illustrates the residuals of the ARIMA model, representing the differences between the actual values and the model's predicted outputs with fluctuations suggesting the model's difficulty in capturing large spikes in the data.

4.1.2. No of Patients in Hospital

Optimal Parameters: (p=2, d=2, q=2)

Ljung-Box Test Results: With a test statistic of 9.3828 and a high p-value (0.496 > 0.05), there is no evidence of significant autocorrelation, indicating a good model fit with expected residual behaviour.

ARIMA Forecast Plot for No of Patients in Hospital

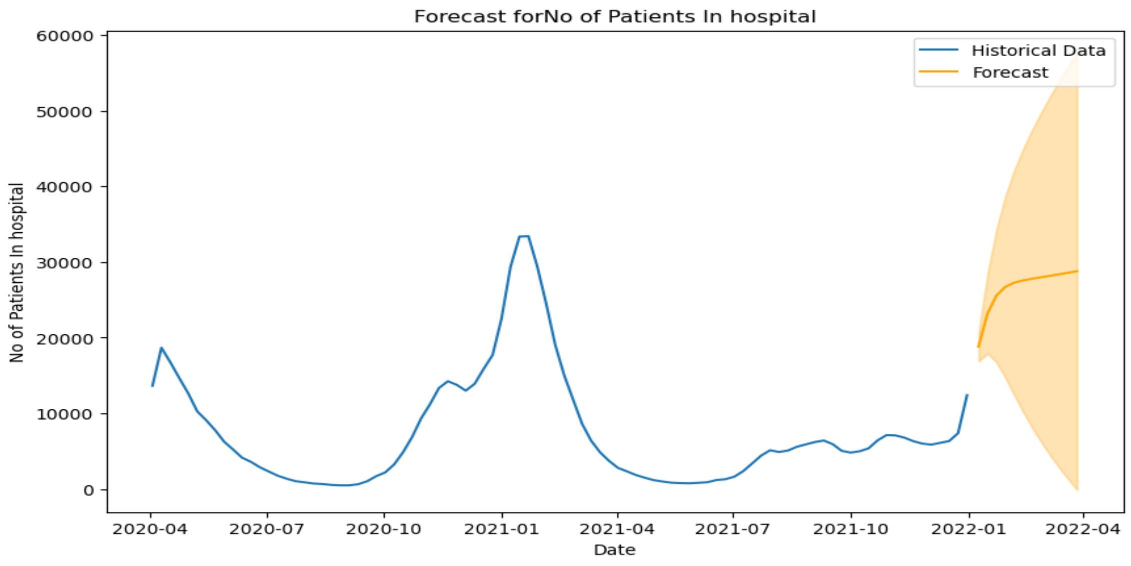


Figure 6: ARIMA Forecast plot for Variable 2: No of Patients in Hospital

The forecast plot shows a gradual rise in the number of hospitalized patients, indicating an increasing trend with widening uncertainty bands over time.

Walk-Forward Validation Plot:

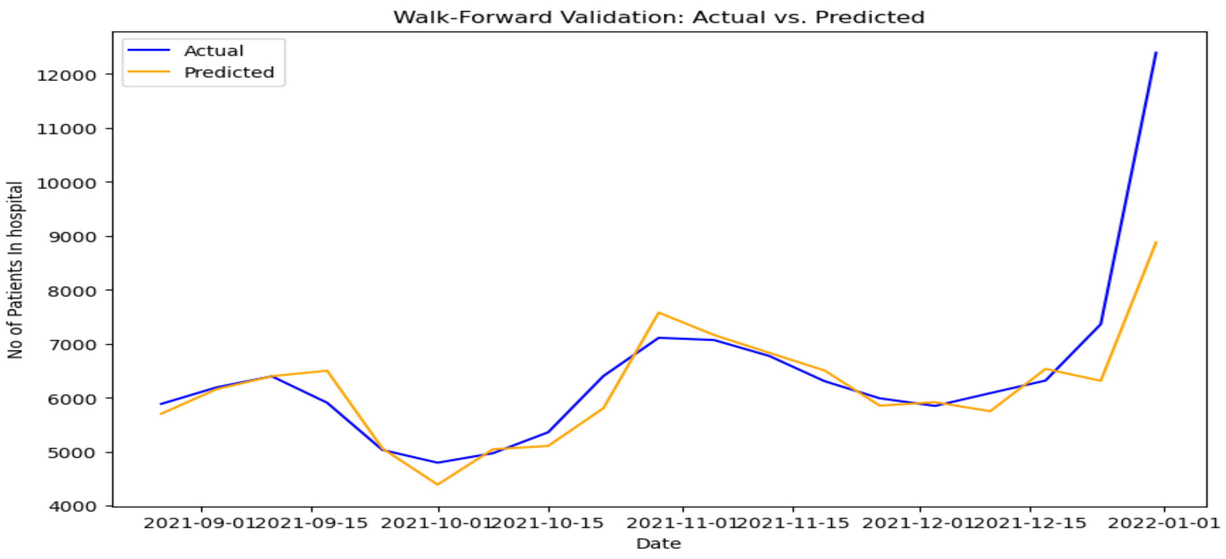


Figure 7 Walk-Forward Validation Plot for Variable 2: No of Patients in Hospital

This plot compares actual patient numbers with the ARIMA model's predicted values using walk-forward validation. The graph shows deviations between the actual and predicted values, especially during periods of sharp increases in patient numbers.

Residual Plot for Variable 2: No of Patients in Hospital

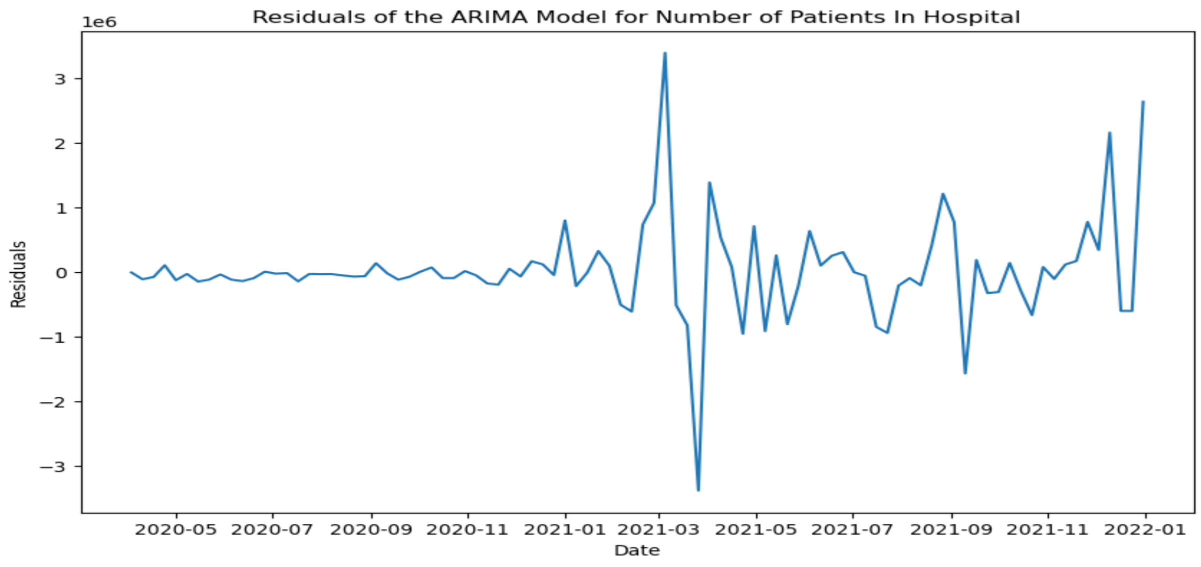


Figure 8 Residual Plot for Variable 2: No of Patients in Hospital

This residual plot shows the deviation among the ARIMA model's predicted values and the actual patient counts, highlighting any discrepancies in the forecasted results. The large fluctuations indicate areas where the model struggled, particularly during spikes in COVID-19 cases.

4.1.3. Patients in Mechanical Ventilation Beds:

Optimal Parameters: (p=1, d=1, q=9)

Ljung-Box Test Results: The test statistic (0.6567) and a p-value of 0.999 indicate a very high probability that the null hypothesis is true, confirming no significant autocorrelation in the residuals.

ARIMA Forecast Plot for No of Patients in Mechanical Ventilation Bed

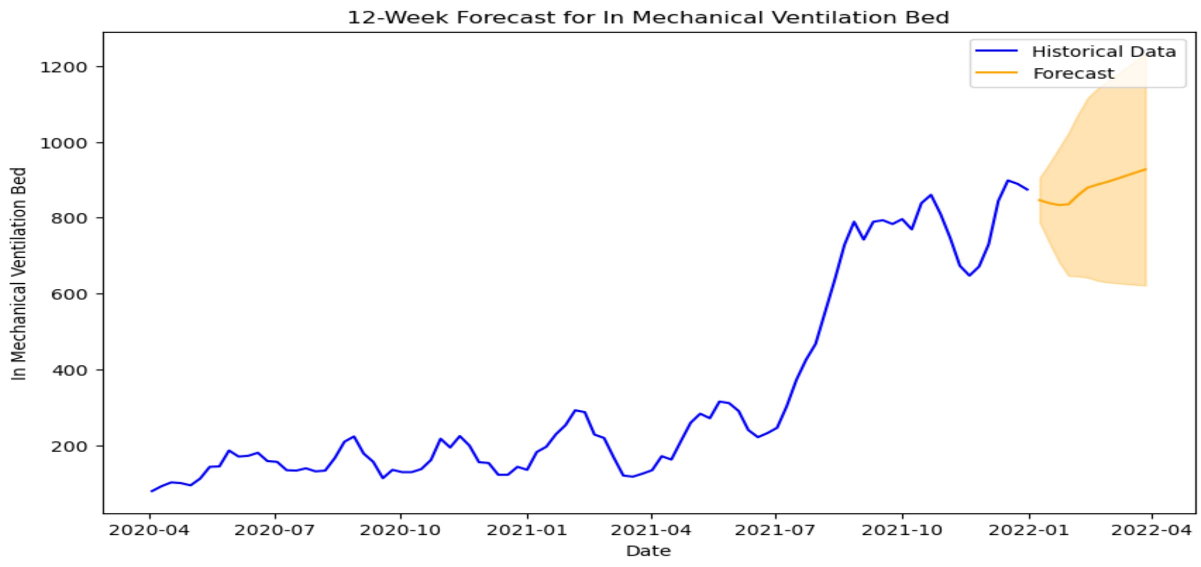


Figure 9 ARIMA Forecast Plot for Variable 3: No of Patients in Mechanical Ventilation Bed

The forecast predicts a stable increase in mechanical ventilation bed usage, reflecting the model's ability to capture the trend while accounting for fluctuations in demand.

Walk-Forward Validation Plot:

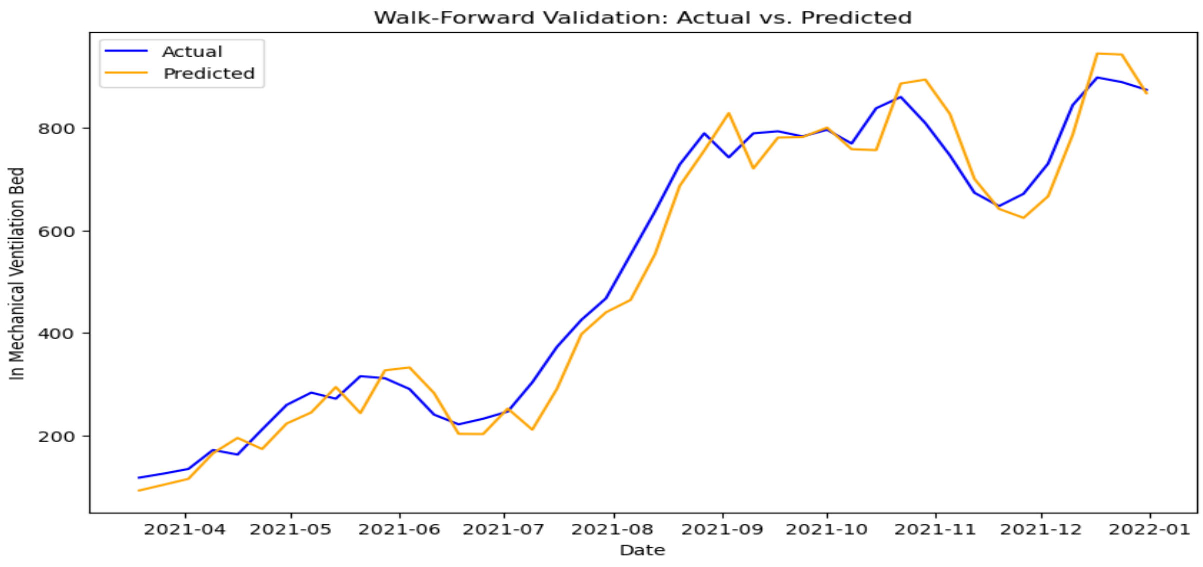


Figure 10 Walk-Forward Validation Plot for Variable 3: No of Patients in Mechanical Ventilation Bed

This graph illustrates a comparison between the actual data and the ARIMA model's predictions, indicating that the model closely tracks the observed values with only minor discrepancies observed at the highest and lowest points.

Residual Plot:

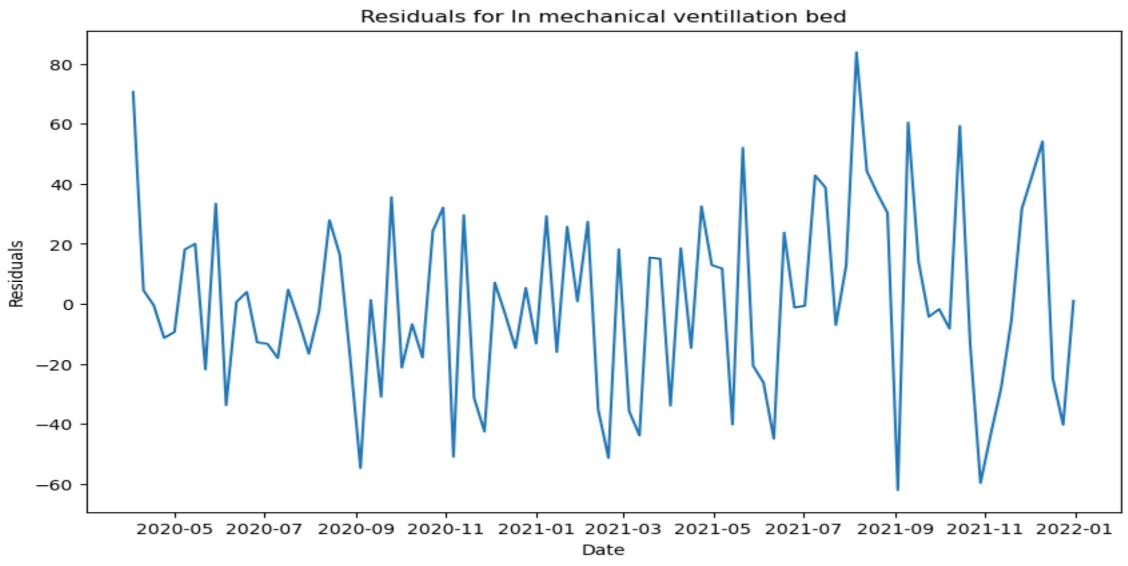


Figure 11 Residual Plot for Variable 3: No of Patients in Mechanical Ventilation Bed

This residual plot shows the differences between the predicted and actual values, indicating that while the model performs well, there are fluctuations and occasional larger errors during some periods.

4.1.4. Registered Deaths Due to COVID:

Optimal Parameters: (p=2, d=2, q=2).

Ljung-Box Test Results: With a test statistic of 9.18 and a p-value of 0.515, the results indicate no significant autocorrelation in the residuals, confirming the null hypothesis.

ARIMA Forecast Plot for Registered Deaths due to Covid:

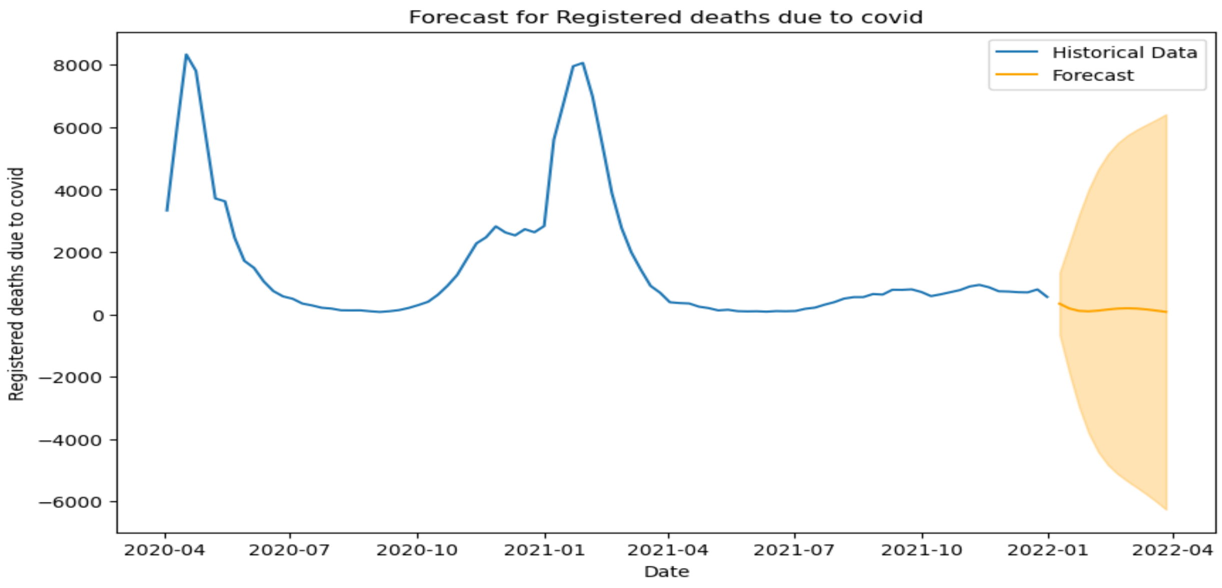


Figure 12 ARIMA Forecast Plot for Variable 4: Registered Deaths due to Covid.

The forecast plot shows a relatively flat trend for registered deaths with uncertainty bands, highlighting the model's limited accuracy in predicting sudden mortality changes.

Walk-Forward Validation Plot:

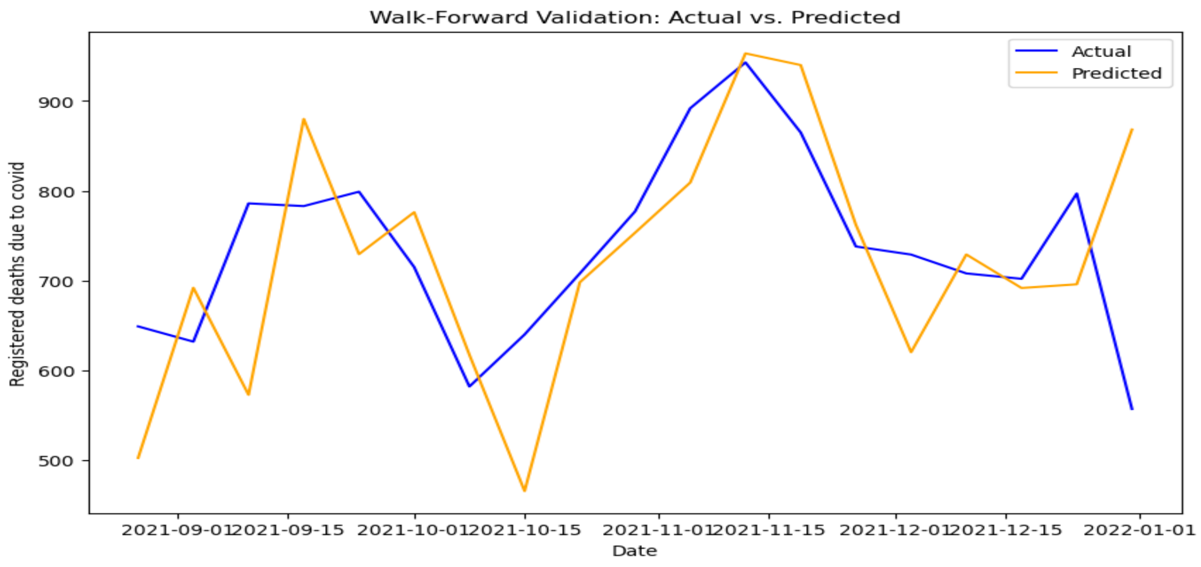


Figure 13 Walk-Forward Validation Plot for Variable 4: Registered Deaths due to Covid.

This plot compares actual and predicted registered deaths, showing that while the model generally captures the trend, it misses some peaks and troughs, especially during periods of sharp changes in death counts.

Residuals Plot:

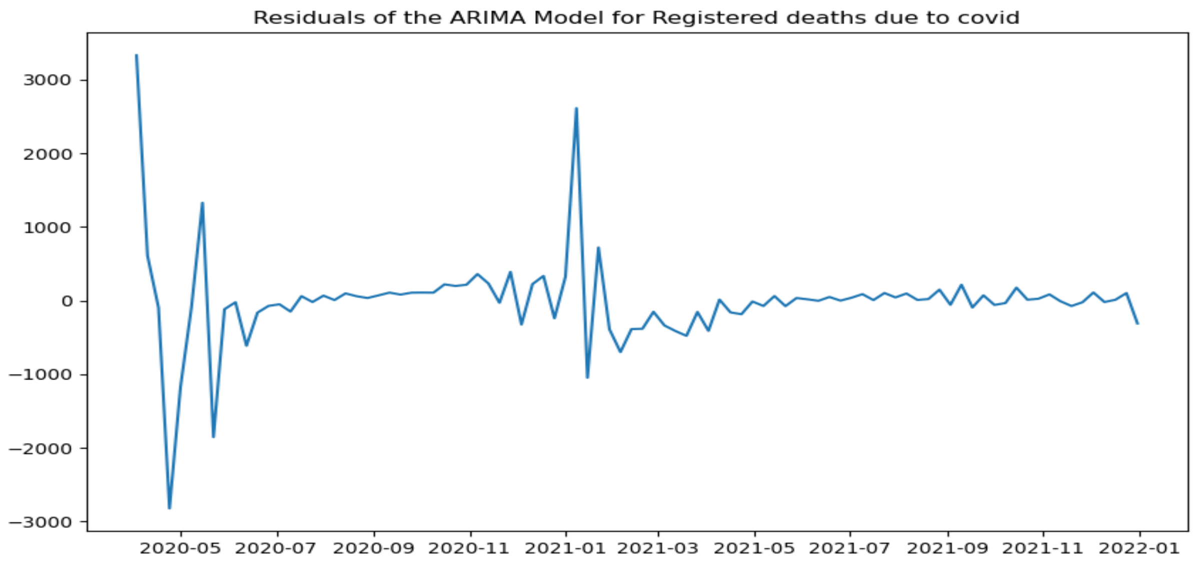


Figure 14 Residual Plot for Variable 4: Registered Deaths due to Covid.

This residual plot shows that the model had difficulty with large fluctuations early on, but residuals stabilized over time, with fewer large deviations in the later periods.

4.2. LSTM with LLM Forecasts

4.2.1. Introduction of the Workflow and Model Architecture

The following diagram outlines the structured approach used in implementing the LSTM with LLM model for healthcare demand forecasting. Each step, from data preparation to forecasting, is crucial for ensuring the model effectively captures temporal patterns and delivers reliable predictions.

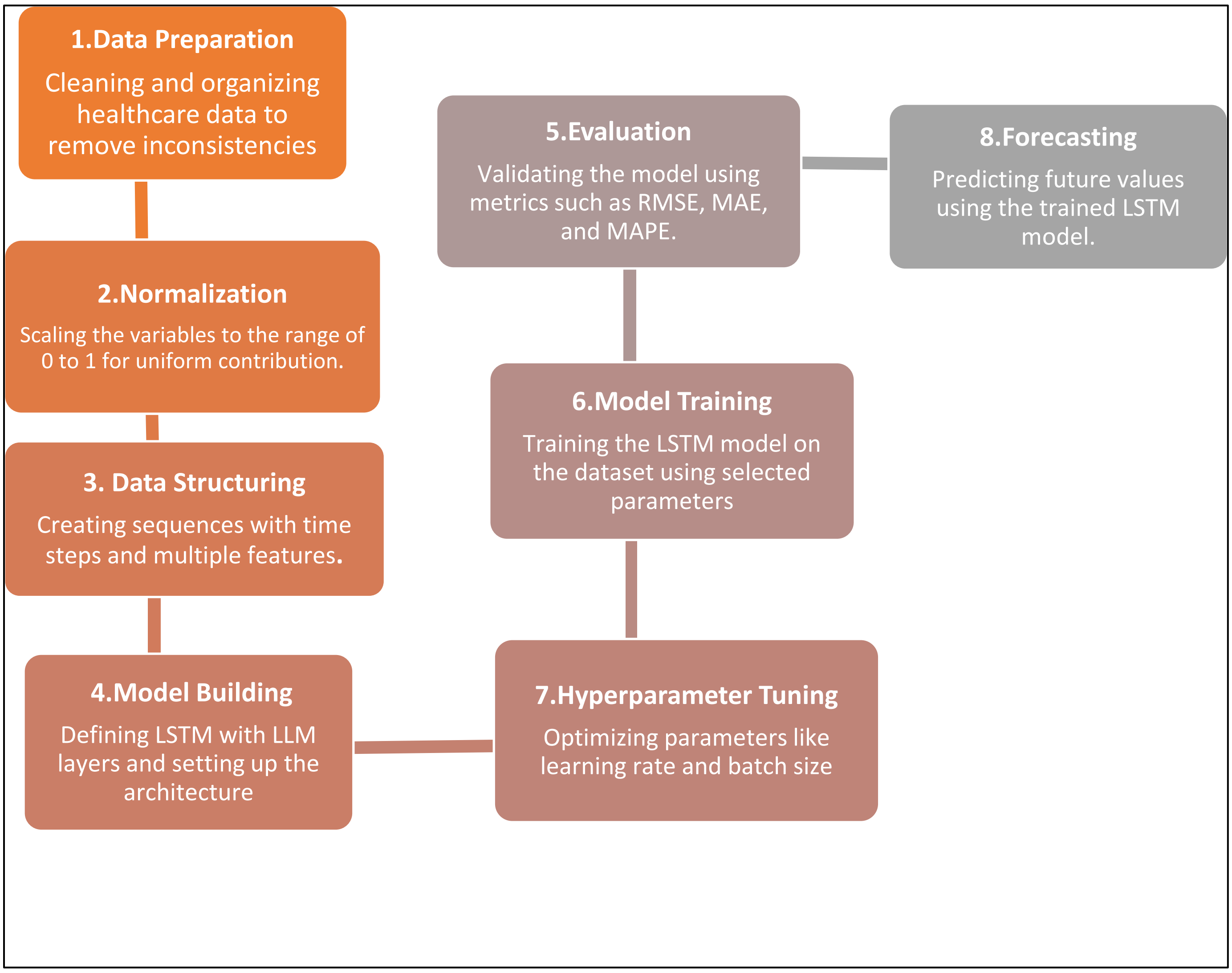

1.Data Preparation
Cleaning and organizing healthcare data to remove inconsistencies
2.Normalization
Scaling the variables to the range of 0 to 1 for uniform contribution.
3. Data Structuring
Creating sequences with time steps and multiple features.
4.Model Building
Defining LSTM with LLM layers and setting up the architecture
5.Evaluation
Validating the model using metrics such as RMSE, MAE, and MAPE.
6.Model Training
Training the LSTM model on the dataset using selected parameters
7.Hyperparameter Tuning
Optimizing parameters like learning rate and batch size
8.Forecasting
Predicting future values using the trained LSTM model.

Figure 15 Workflow for LSTM with LLM Implementation

The LSTM with LLM model was implemented to forecast healthcare demand, leveraging its capacity to capture extended temporal dependencies and incorporate complex, unstructured data sources. The structured workflow for this model is illustrated in Figure 15, detailing each stage from data preparation to final forecasting.

Data normalization was performed using the MinMaxScaler from the scikit-learn library, scaling all variables between 0 and 1 to ensure balanced contributions during training ((Pedregosa, 2012) ,(Courville, 2016)). Following normalization, the data was structured into sequences with multiple time steps to enable the LSTM to learn temporal patterns efficiently across the four key variables: Number of Patients, Number of Patients in Hospital, Mechanical Ventilation Usage, and Registered Deaths (Kalloubi Fahd, 2024).

The model architecture comprised a primary LSTM layer with 50 units, followed by a Dense layer to produce simultaneous predictions for all four variables (Ian Goodfellow, 2017). The final model was trained using the Adam optimizer over 100 epochs with a batch size of 32. To enhance generalization and minimize overfitting, a dropout rate of 0.2 was applied, which has been shown to improve model robustness (Diederik P. Kingma, 2014) (Srivastava, 2014).

4.2.3. Hyperparameter Tuning and Training

Hyperparameter tuning was performed using Keras Tuner to optimize critical parameters, including learning rate, dropout rate, sequence length, and batch size, which significantly impacted model performance (O'Malley, 2019). After multiple iterations, the final configuration was established with a sequence length of 5, learning rate of 0.001, batch size of 32, and dropout rate of 0.2. These settings balanced training efficiency and complexity, reducing overfitting and achieving optimal predictive accuracy((Siami-Namini, et al., 2018). (Srivastava, 2014); (Ian Goodfellow, 2017)).

Hyperparameter	Optimal Value	Justification
Sequence Length	5	Captures short-term dependencies (Siami-Namini, et al., 2018).
Learning Rate	0.001	Ensures steady convergence without overshooting (Ian Goodfellow, 2017).
Batch Size	32	Balances computational efficiency and stability (Srivastava, 2014).
Dropout Rate	0.2	Prevents overfitting by randomly dropping units during training (Srivastava, 2014).

Figure 16 Summary table of the optimal hyperparameter.

After fine-tuning the hyperparameters, noticeable improvements were seen in both training and validation loss, minimizing overfitting, and enhancing model stability. This optimization process led to a more reliable model, capable of generating accurate predictions on the test set, which included all four variables. The LSTM with LLM model moderately captured temporal patterns and external factors but struggled to align closely with actual values during sudden shifts in healthcare demand.

Refer Appendix D (72) for a detailed analysis and Python implementation.

4.2.4. Forecast Plots Analysis

LSTM with LLM Model Predicted vs Actual Values plot for All Variables Combined

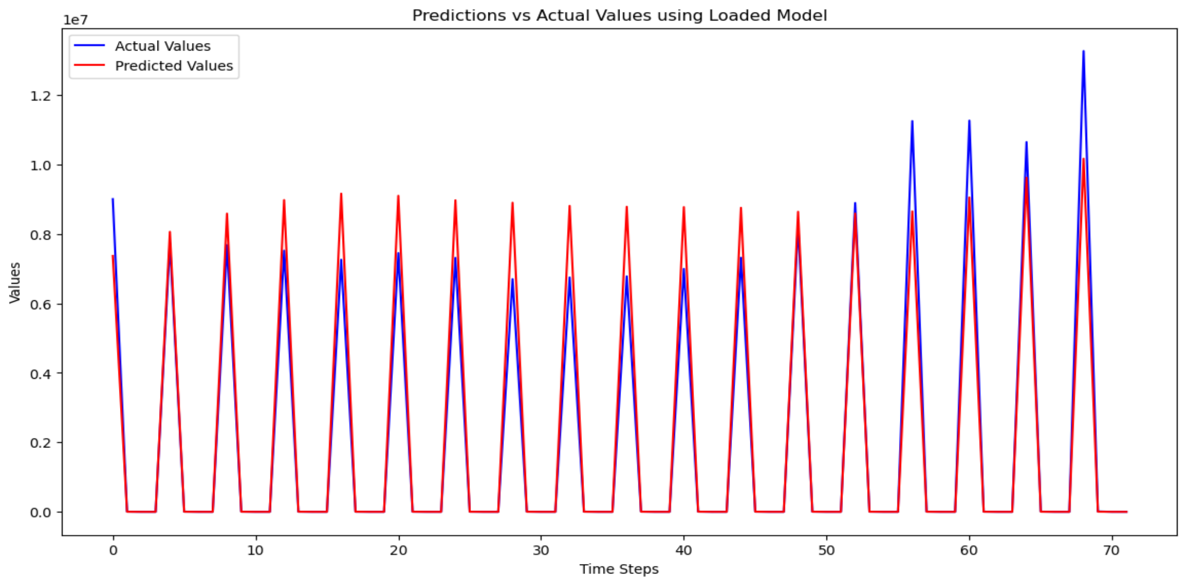


Figure 17 LSTM with LLM Model Predicted vs Actual Values plot foe All Variables Combined

Error Metrics for above model of all variables combined:

MAE: 398,350,

RMSE: 879,141

MAPE: 26.52%,

The LSTM with LLM Model Predicted vs. Actual Values plot shows the model's performance in predicting all combined variables. The error metrics indicate moderate accuracy, with an MAE of 398,350 and RMSE of 879,141, reflecting some deviation from actual values. The MAPE of 26.52% suggests varying prediction accuracy, likely due to complex trends across different variables.

Following plots shows the Predicted vs Actual Values plots for all variables:

Variable 1: No of Patients

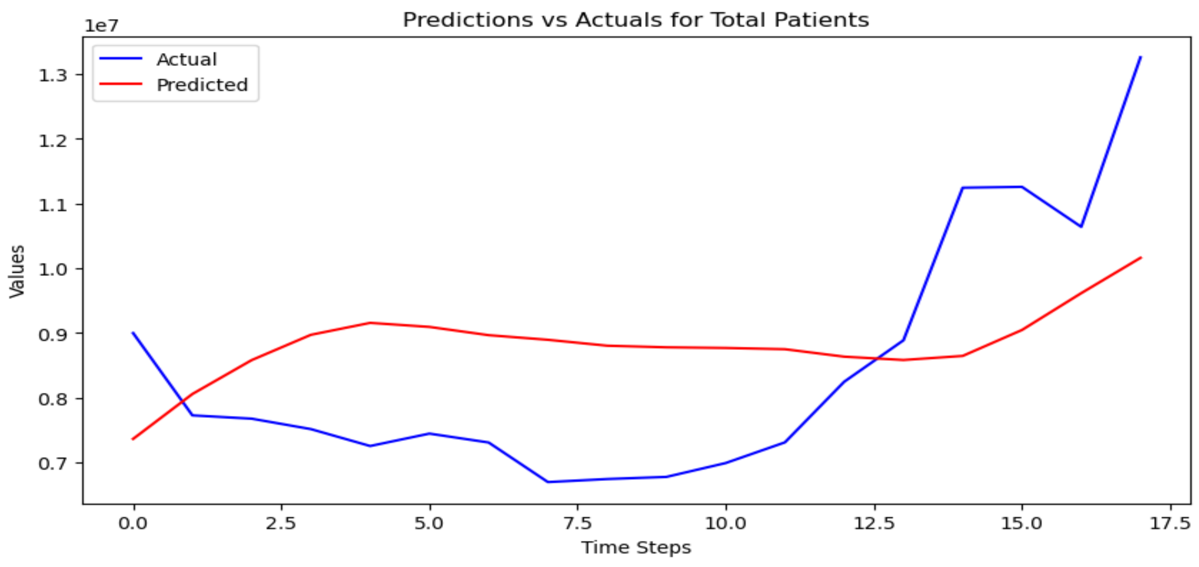


Figure 18 LSTM with LLM Forecast Plot for Variable 1: No of Patients

The model seems to smooth out the predictions, potentially due to averaging effects or insufficient sensitivity to rapid changes.

Variable 2: No of Patients in Hospital

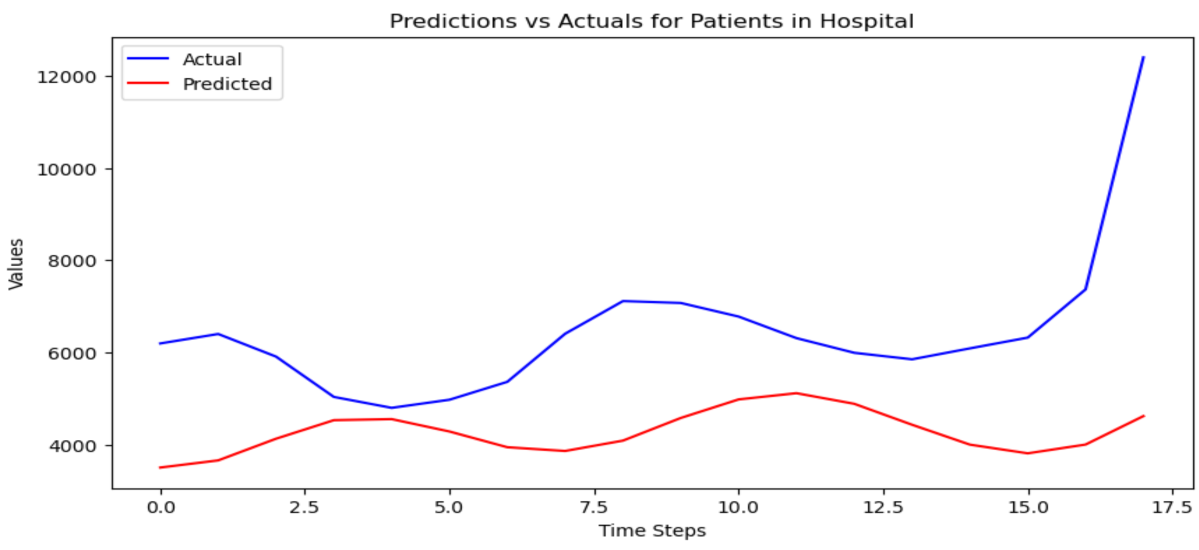


Figure 19 LSTM with LLM Forecast Plot for Variable 2: No of Patients in Hospital

The model might be underestimating the demand on hospitals, possibly missing out on key features or trends that drive sudden increases.

Variable 3: No of Patients in Mechanical Ventilation Bed

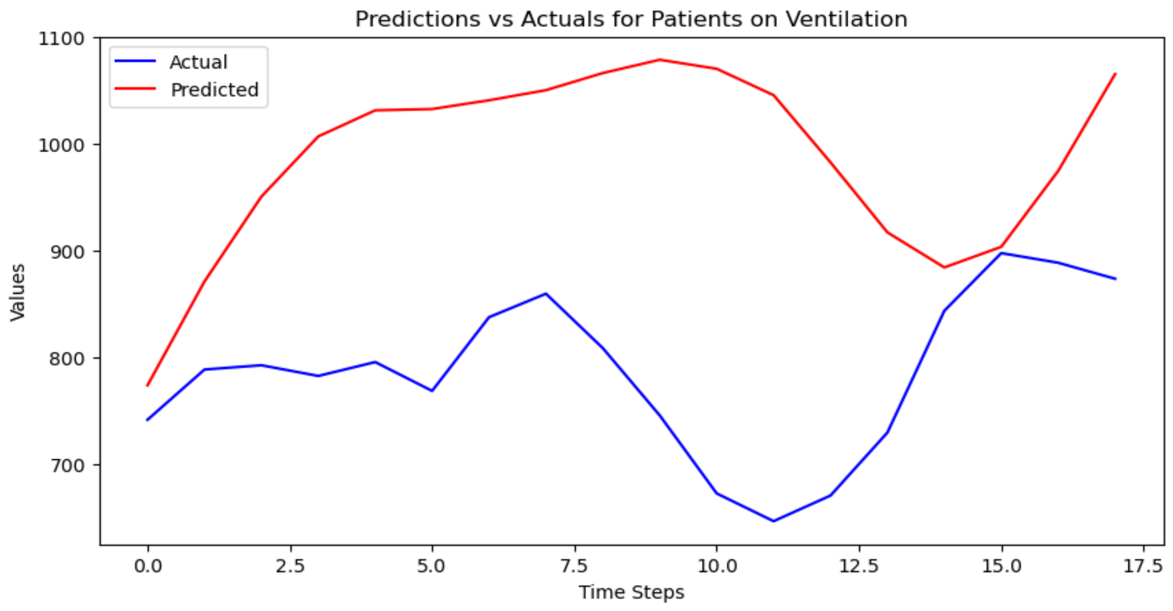


Figure 20 : LSTM with LLM Forecast Plot for Variable 3: No of Patients in Mechanical Ventilation Bed

This suggests that the model may need further tuning or that it is not fully capturing the factors that lead to lower numbers.

Variable 4: Registered Deaths Due to Covid

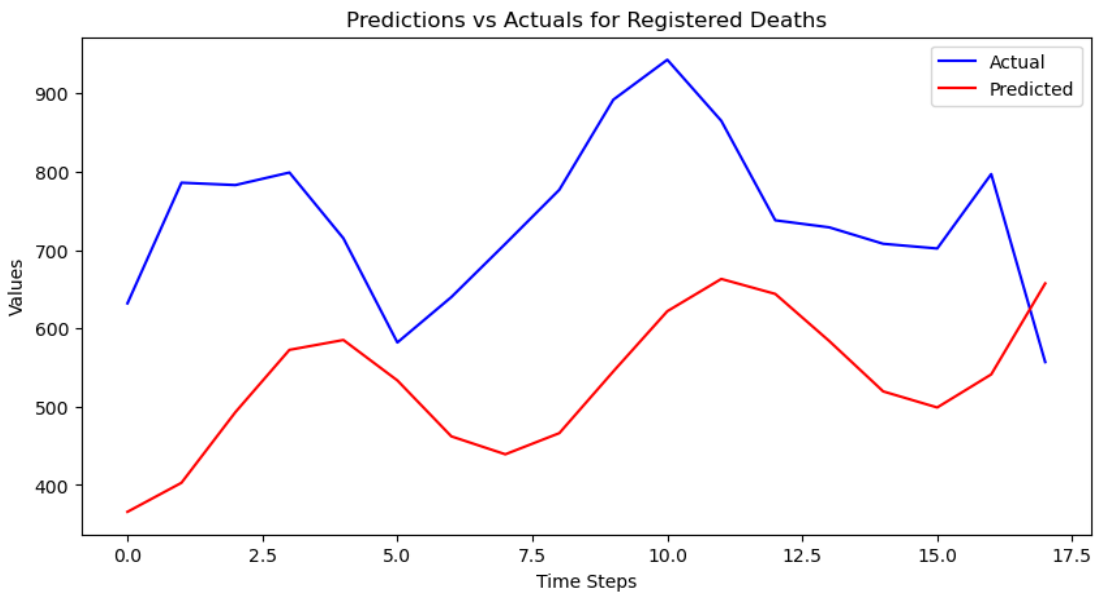


Figure 21 LSTM with LLM Forecast Plot for Variable 4: Registered Deaths Due to Covid

The model may be capturing the overall pattern but struggles with the more extreme variations in the data.

Summary of LSTM with LLM Performance

Despite using advanced LSTM techniques, the model showed significant deviations in predicting healthcare demand variables, primarily due to its inability to adjust to rapid changes. This suggests that integrating external data or using more sophisticated modeling approaches may be necessary to enhance accuracy in dynamic scenarios like the COVID-19 pandemic.

4.3. LSTM without LLM (GRU) Forecasts

4.3.1. Workflow of the GRU model

GRU (Gated Recurrent Unit) model follows a streamlined workflow, emphasizing efficient handling of smaller datasets and reduced computational overhead. Key steps include data preparation, feature scaling, model building, and evaluation. Unlike the LSTM with LLM, this leaner architecture adapts quickly to changes in temporal patterns. The following diagram (Figure 22) illustrates the complete process, from data preparation and feature scaling to model building and evaluation.

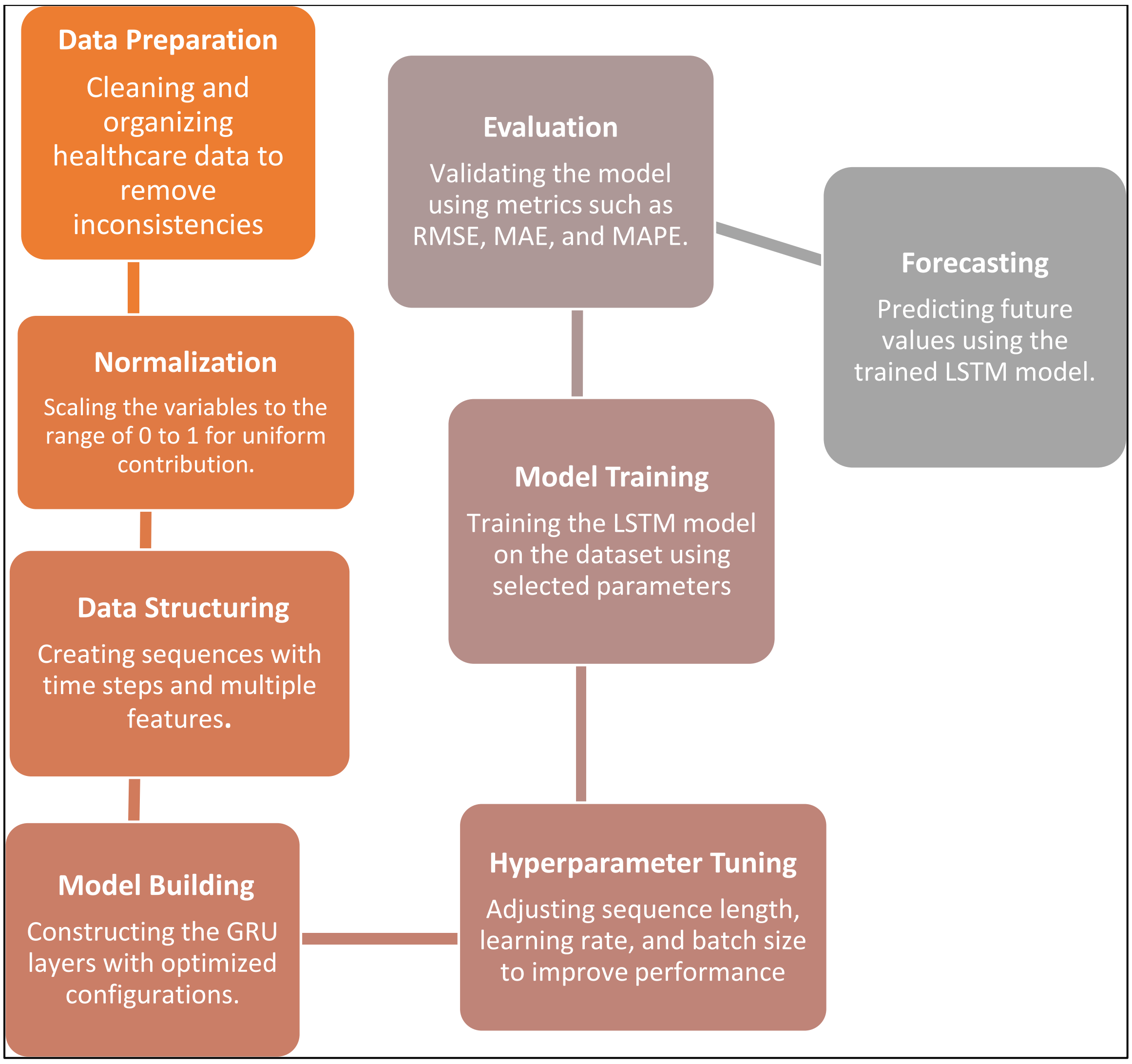

Data Preparation
Cleaning and organizing healthcare data to remove inconsistencies
Normalization
Scaling the variables to the range of 0 to 1 for uniform contribution.
Data Structuring
Creating sequences with time steps and multiple features.
Model Building
Constructing the GRU layers with optimized configurations.
Hyperparameter Tuning
Adjusting sequence length, learning rate, and batch size to improve performance
Model Training
Training the LSTM model on the dataset using selected parameters
Evaluation
Validating the model using metrics such as RMSE, MAE, and MAPE.
Forecasting
Predicting future values using the trained LSTM model.

Figure 22 Workflow for LSTM without LLM: GRU Implementation

4.3.2. Data Preparation and Feature Scaling

The data was normalized using MinMaxScaler to scale variables between 0 and 1, ensuring balanced contributions during training (Chung, 2014). The four variables—Number of Patients, Patients in Hospital, Mechanical Ventilation, and Registered Deaths—were structured into sequences using an 80-20 split for training and testing.

4.3.3.GRU Model Architecture

The GRU model architecture, consisting of 50 GRU units and a Dense layer for multi-variable output, was chosen for its efficiency over standard LSTM models (Chung, 2014). Key hyperparameters included a learning rate of 0.001, batch size of 32, and a sequence length of 5, with dropout (0.2) and early stopping to prevent overfitting

((Srivastava, 2014); (Diederik P. Kingma, 2014)). The loss function used was mean squared error (MSE), and a linear activation function was applied for the output layer (Ian Goodfellow, 2017).

4.3.4. Model Training and Hyperparameter Tuning

Initial training showed overfitting after 20 epochs, which was mitigated through hyperparameter optimization using Keras Tuner. Adjusted settings improved model stability, although the GRU still struggled with capturing sudden fluctuations. Despite these limitations, the GRU provided reliable short-term forecasts, closely following the general trends in patient data.

For a detailed analysis and Python implementation, refer to Appendix C (77).

The GRU model's predictions for each variable were compared against the actual values. The performance across the four variables showed the following results:

Forecast Evaluation:

Plot: Combined GRU Forecast Plot for All Variables

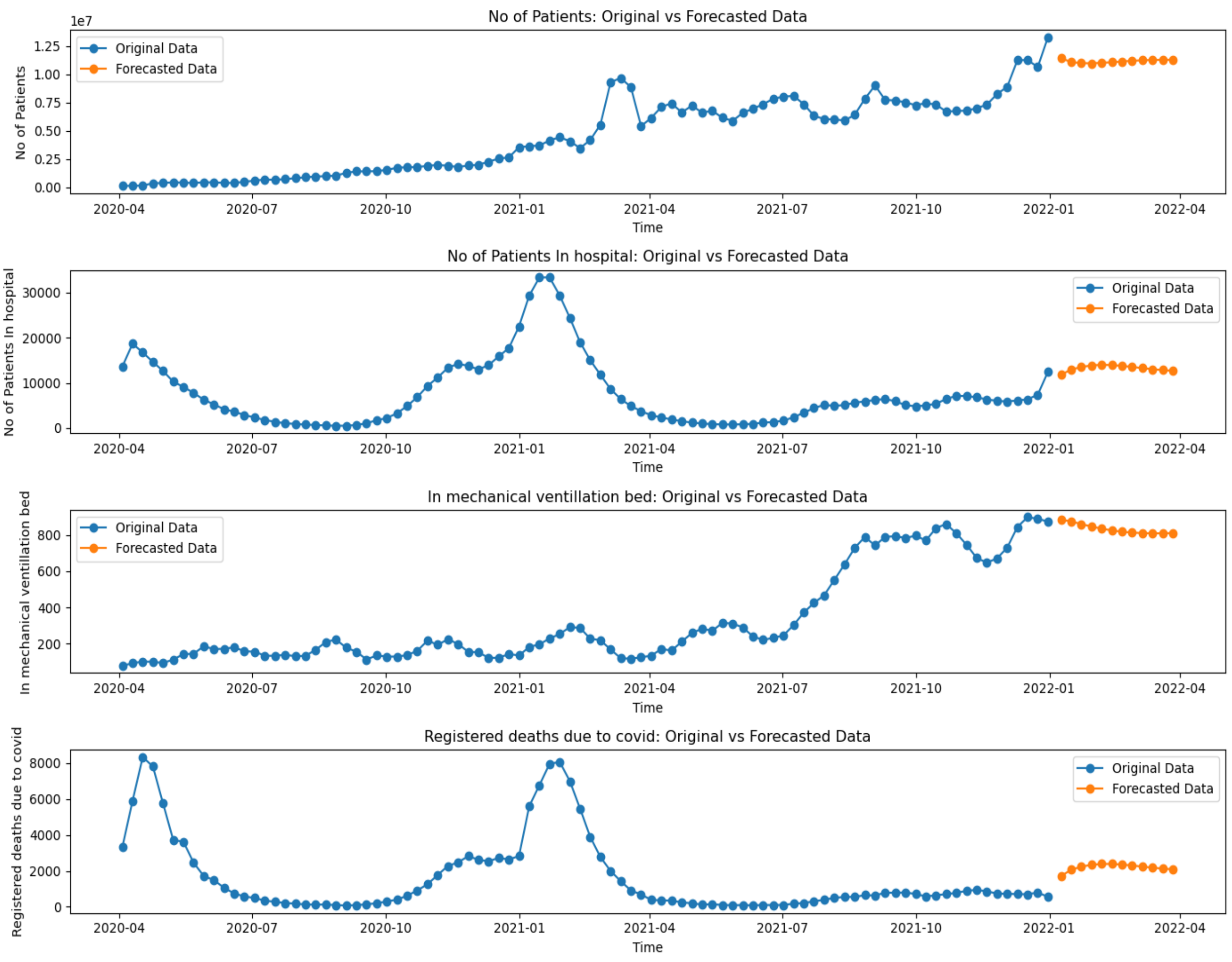


Figure 23 LSTM without LLM (GRU) Forecasts Plots for All variables.

The forecasted data across all four variables—No of Patients, No of Patients in Hospital, In Mechanical Ventilation Bed, and Registered Deaths Due to COVID—closely follows the original data trends, demonstrating the model's ability to predict healthcare demand efficiently. While the model captures the overall upward or stable trends accurately, it slightly smooths out extreme fluctuations, particularly during peaks in patient numbers and hospitalizations. Overall, the model provides reliable forecasts with minimal deviation from the actual data, making it a suitable tool for short-term healthcare demand forecasting.

Walk-Forward Validation Plots:

To further assess the robustness of the GRU model, walk-forward validation plots were generated for each of the four variables. This method evaluates the model’s performance over rolling time windows, allowing for a more comprehensive analysis of its predictive accuracy and adaptability across different time periods.

Variable 1: No of Patients

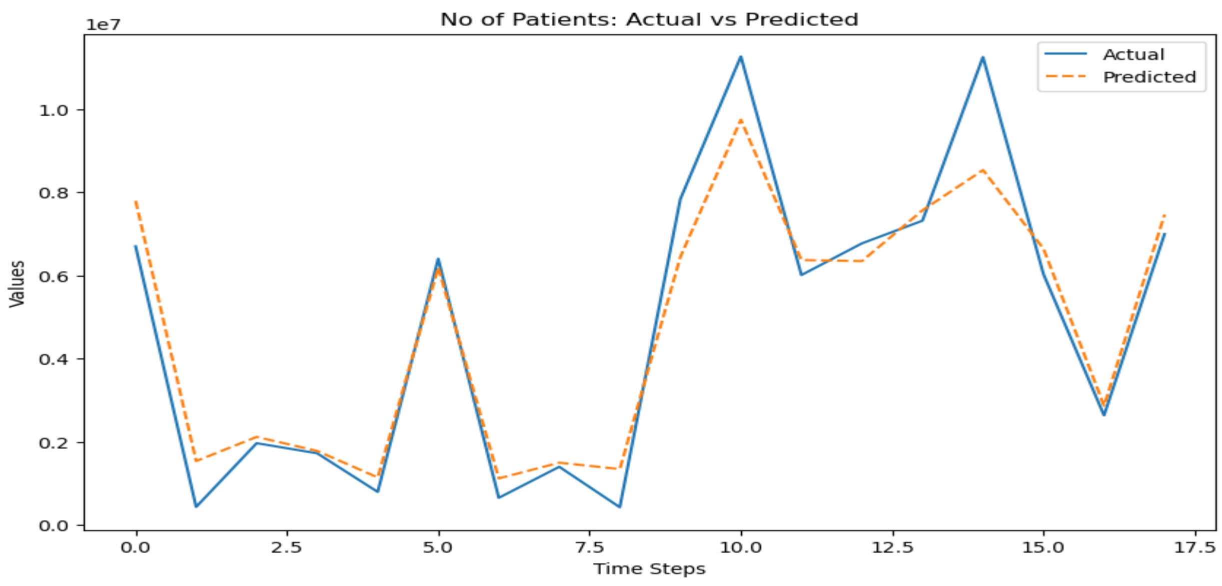


Figure 24 GRU Walk-Forward Validation Plot for Variable 1: No of Patients

The GRU model closely follows the actual patient numbers, capturing both the overall trend and major fluctuations with minimal deviations.

Variable 2: No of Patients in Hospital

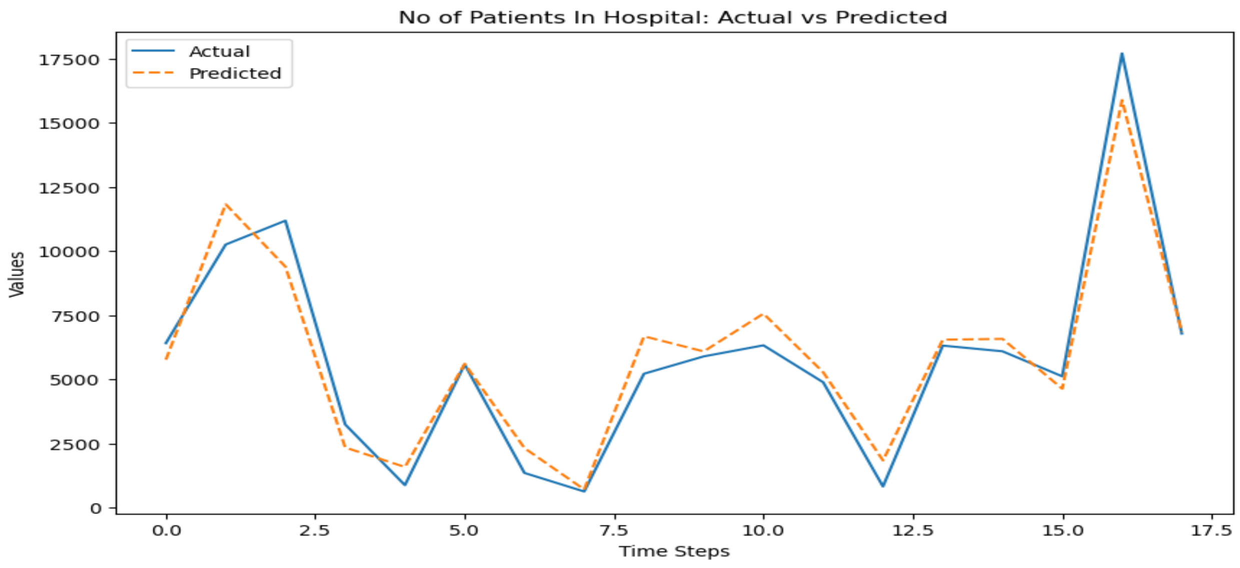


Figure 25 GRU Walk-Forward Validation Plot for Variable 2: No of Patients in Hospital

The GRU model successfully predicts hospital admissions, closely aligning with the actual values while slightly underestimating the highest peaks.

Variable 3: No of Patients in Mechanical Ventilation Bed

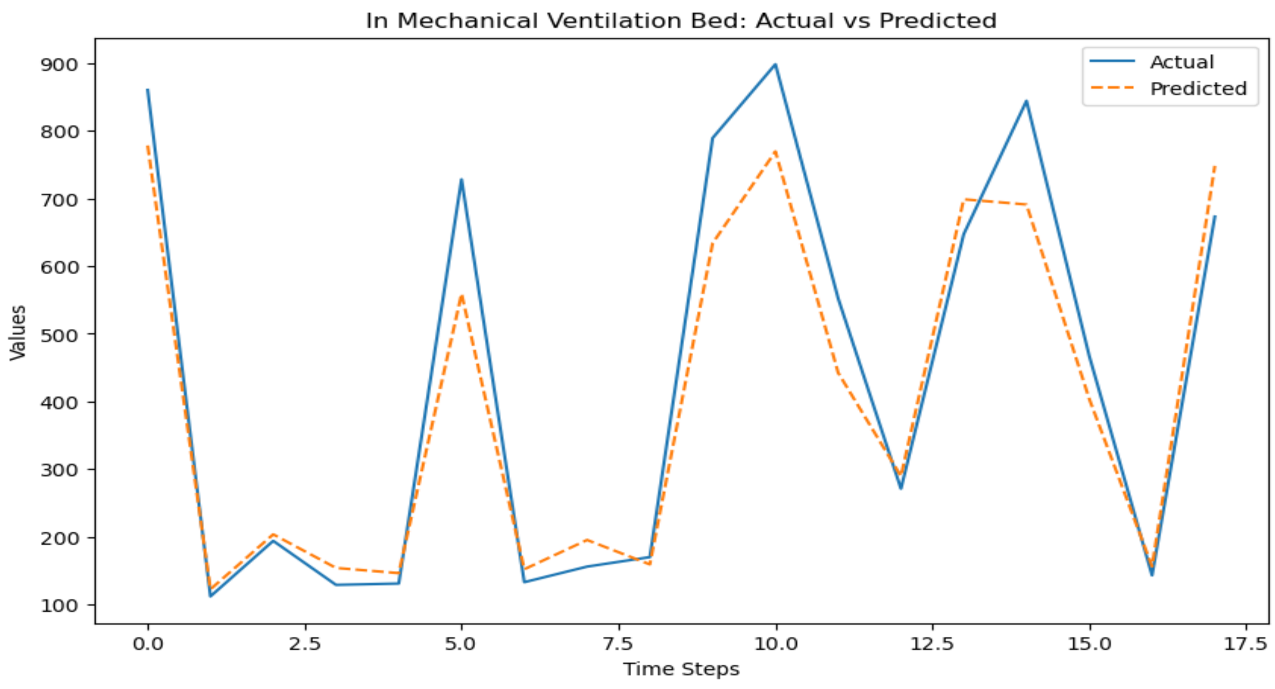


Figure 26 GRU Walk-Forward Validation Plot for Variable 3: No of Patients in Mechanical Ventilation Bed

The model accurately forecasts the demand for mechanical ventilation, with predicted values closely matching the actual data, including sharp spikes.

Variable 4: Registered Deaths due to Covid.

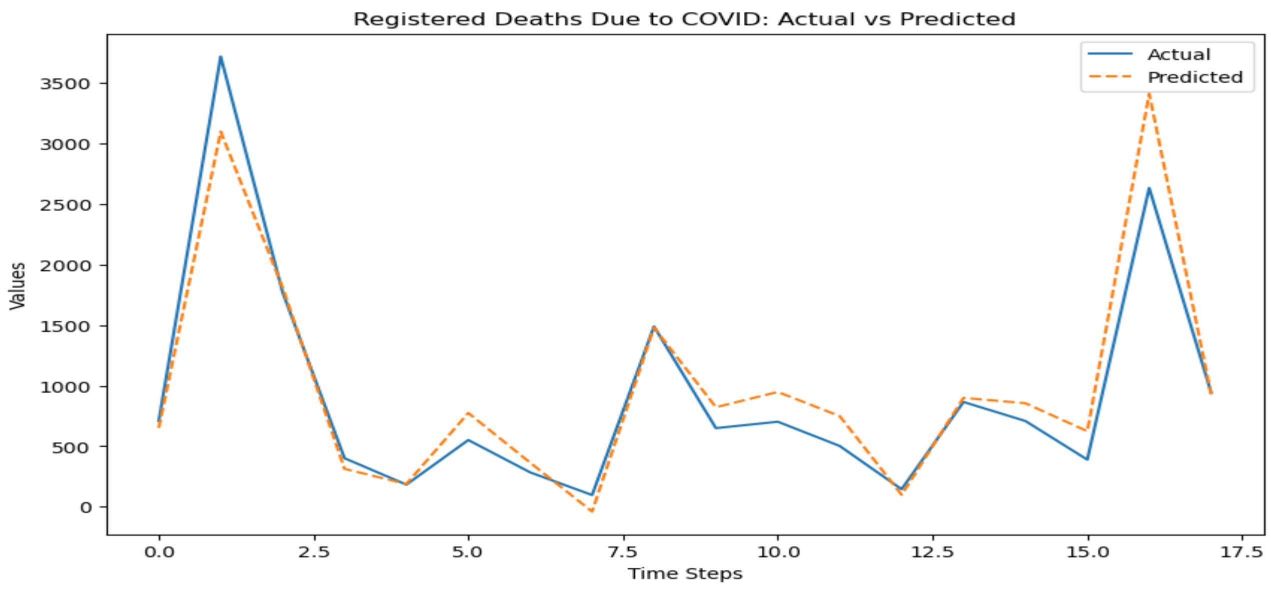


Figure 27 GRU Walk-Forward Validation Plot for Variable 4: Registered Deaths due to Covid.

The GRU model provides reliable predictions for registered deaths, aligning well with actual death counts while capturing the extreme peaks and troughs effectively.

Summary of GRU Performance

The GRU model exhibited moderate success in predicting general trends across all variables, particularly in scenarios where the data was less volatile (e.g., mechanical ventilation). However, the model struggled significantly with sudden fluctuations and peak periods, leading to large discrepancies between the predicted and actual values.

4.4. Dashboard: A Comparative Analysis

Dashboard Overview and Insights

To facilitate a comprehensive comparison of model performance, a dashboard was developed using Python Dash. This interactive dashboard presents visual comparisons of ARIMA, LSTM without LLM, and LSTM with LLM across key healthcare variables, including Number of Patients, Patients in Hospital, Mechanical Ventilation Bed Usage, and Registered Deaths Due to COVID-19. Each model's forecasts are plotted against actual historical data, allowing for a straightforward evaluation of their accuracy and error metrics.

The aim of the dashboard is to present a user-friendly and interactive way to analyse how each model performs under different scenarios. It includes features such as side-by-side comparisons, error metrics, and walk-forward validation for ARIMA, which enable more granular examination of the models' performance.

For a detailed view of the dashboard visuals and specific plots for each variable, please refer to Appendix F (84).

Visual Comparison and Summaries: Each variable’s section includes a side-by-side comparison of the three models, along with summaries highlighting which model performed best. The walk-forward validation plots for ARIMA provide additional granularity for understanding its temporal accuracy across different time periods. For a detailed breakdown of the visual comparisons and validation plots for each variable, see Appendix F (84).

5.Chapter: Results

This section evaluates the performance of three forecasting models—ARIMA, LSTM without LLM, and LSTM with LLM—on four healthcare variables: number of patients, hospitalizations, mechanical ventilation bed usage, and registered deaths due to COVID-19. Each model's accuracy was assessed by key error metrics such as MAE, RMSE, and MAPE.

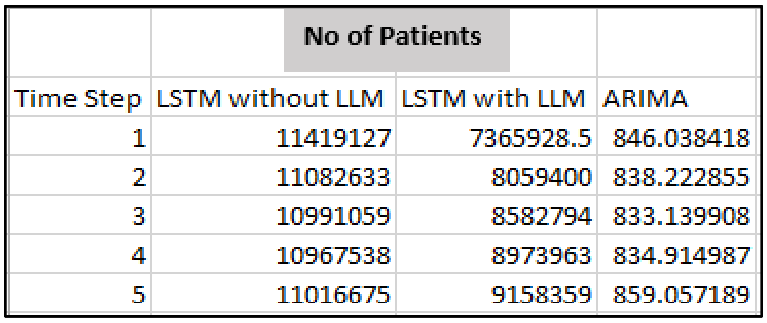
The LSTM with LLM model, leveraging external contextual data, was expected to perform best due to its ability to capture complex non-linear patterns. In contrast, ARIMA served as a benchmark, demonstrating stable results for linear trends but facing limitations with volatile data.

The comparison of forecasted values and error metrics provides a thorough overview of each model's strengths and weaknesses, aiding in the identification of the most suitable approach for healthcare demand forecasting during the pandemic.

5.1. Forecasting Results for each variable from each method:

To present a clear overview of the forecasted values for each variable, a subset of the predicted data is displayed in the following tables, showcasing the first five forecasted values. These initial values are intended to offer a concise representation of the model outputs, facilitating comparison, and highlighting any notable discrepancies between the models.

For Variable 1: No of Patients:

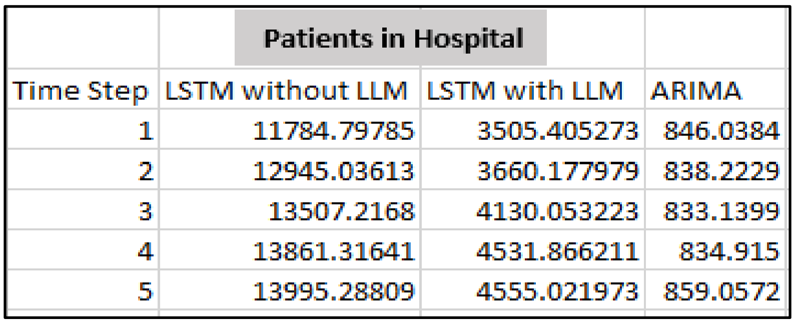


	No of Patients		
Time Step	LSTM without LLM	LSTM with LLM	ARIMA
1	11419127	7365928.5	846.038418
2	11082633	8059400	838.222855
3	10991059	8582794	833.139908
4	10967538	8973963	834.914987
5	11016675	9158359	859.057189

Figure 28 Combined Forecast table for Variable 1: NO of Patients

Here, LSTM without LLM consistently overestimated patient numbers, reaching up to 11 million, while LSTM with LLM forecasted more moderately around 7-9 million. ARIMA produced the most conservative estimates (830,000 to 850,000), indicating its limited capacity to capture rapid changes.

For Variable 2: No of Patients in Hospital:

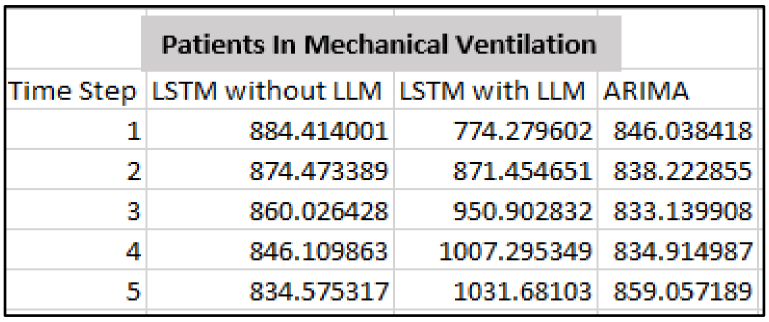


	Patients in Hospital		
Time Step	LSTM without LLM	LSTM with LLM	ARIMA
1	11784.79785	3505.405273	846.0384
2	12945.03613	3660.177979	838.2229
3	13507.2168	4130.053223	833.1399
4	13861.31641	4531.866211	834.915
5	13995.28809	4555.021973	859.0572

Figure 29 Combined Forecast table for Variable 2: NO of Patients in Hospital

In this table, LSTM without LLM again projected higher patient counts, whereas LSTM with LLM provided more realistic estimates (3,500 to 4,500), aligning closer to expected hospital admissions. ARIMA's forecast of 830-850 patients significantly underestimated actual hospitalizations.

For Variable 3: No of Patients in Mechanical Ventilation Bed:

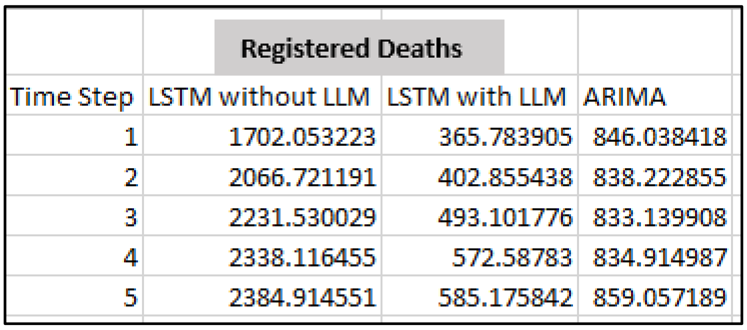


	Patients In Mechanical Ventilation		
Time Step	LSTM without LLM	LSTM with LLM	ARIMA
1	884.414001	774.279602	846.038418
2	874.473389	871.454651	838.222855
3	860.026428	950.902832	833.139908
4	846.109863	1007.295349	834.914987
5	834.575317	1031.68103	859.057189

Figure 30 Combined Forecast table for Variable 3: NO of Patients in Mechanical Ventilation Bed

Here, for mechanical ventilation, LSTM models delivered comparable forecasts, with LSTM with LLM predicting 774 to 1,031 ventilator requirements and LSTM without LLM at 834 to 884. ARIMA, while conservative, slightly overestimated the ventilator usage based on real-world data.

For Variable 4: Registered Deaths Due to Covid:



	Registered Deaths		
Time Step	LSTM without LLM	LSTM with LLM	ARIMA
1	1702.053223	365.783905	846.038418
2	2066.721191	402.855438	838.222855
3	2231.530029	493.101776	833.139908
4	2338.116455	572.58783	834.914987
5	2384.914551	585.175842	859.057189

Figure 31 Combined Forecast table for Variable 4: Registered Deaths Due to Covid

Here, LSTM without LLM greatly overestimated registered deaths (1,700 to 2,384), whereas LSTM with LLM forecasted closer to the actual figures (365 to 585). ARIMA's estimates were stable (830-850) but failed to capture sudden fluctuations in death counts.

5.2. Summary table of error metrics of performed models:

The table below summarizes the error metrics (MAE, RMSE, MAPE) for ARIMA and LSTM models (with and without LLM) across the key variables: Number of Patients, Number of Patients in Hospital, Mechanical Ventilation Bed Usage, and Registered Deaths due to COVID. These metrics highlight the performance of each model in capturing the underlying trends and provide a basis for comparing their effectiveness.

Variable	**Model**	**MAE**	**RMSE**	**MAPE**	**Observations**
Number of Patients	ARIMA	8108843.94	8223776.68	3658089.16%	Good for stable trends, ineffective for rapid changes.
	LSTM without LLM- GRU	579,971.96	843,926.88	19.66%	Handles non-linear patterns better but struggles during extreme spikes.
	LSTM with LLM	1,590,793.86	1,758,281.06	19.24%	Incorporates external factors, but high errors indicate overfitting in highly dynamic periods.
Number of Patients in Hospital	ARIMA	5940.14	6015.47	2676.73%	Stable but fails with sudden changes.
	LSTM without LLM- GRU	811.62	992.79	37.05%	Better accuracy for sudden increases but lacks adaptability to external changes.
	LSTM with LLM	2,186.86	2,716.09	31.08%	Complex patterns handled moderately well, but marginal performance gains compared to simpler models.

Mechanical Ventilation Bed Usage	ARIMA	40.71	49.08	9.93%	Best performance for linear and stable patterns, indicating suitability for predictable scenarios.
	LSTM without LLM- GRU	56.405	74.832	11.47%	Effective in modeling non-linear patterns, though slightly higher errors compared to ARIMA.
	LSTM with LLM	200.04	230.77	26.67%	High complexity with only marginal improvements, making it less practical for this variable.
Registered Deaths due to COVID	ARIMA	477.31	494.73	223.77%	Linear assumptions limit accuracy for volatile trends: high errors observed. Fails with volatile trends.
	LSTM without LLM	187.37	325.154	26.62%	Better for sudden peaks, still limited.
	LSTM with LLM	219.88	56,606.77	29.05	Marginally better, high errors persist.

Figure 32 Summary Table of Error Metrics for all variables for each method.

6.Chapter: Discussion

6.1. Comparison of Findings with Existing Literature and Theoretical Frameworks:

This study's results align with previous research on healthcare forecasting using time-series models. ARIMA, known for its efficiency in capturing linear and stable patterns, performed well for Mechanical Ventilation Bed Usage ((Ping-Feng Pai, 2005) and (Mehdi Khashei, 2011)). However, it struggled with highly volatile variables like Number of Patients and Registered Deaths, confirming its limitations in handling sudden shifts (Zohren, 2021).

In contrast, LSTM models outperformed ARIMA for non-linear variables, consistent with findings by (Siami-Namini, et al., 2018) and (Mohd Saqib, 2022). LSTM with LLM showed only marginal improvements for variables such as Number of Patients in Hospital (MAPE: 31.08%), reflecting its tendency to overfit and incur high computational costs (Kaushik Shruti, 2020). LSTM models, while effective for capturing complex patterns, may not always yield significant benefits in healthcare forecasting due to these inefficiencies (Kaushik Shruti, 2020).

6.2. Addressing Research Questions and Hypotheses:

The study tested two hypotheses:

(1) ARIMA models would perform better for linear variables with stable patterns, and

(2) LSTM models (with and without LLM) would outperform ARIMA for non-linear variables influenced by external factors.

The results confirmed both hypotheses. ARIMA excelled for stable patterns, such as Mechanical Ventilation Bed Usage (MAPE: 9.93%), while LSTM without LLM captured non-linear trends in variables like Number of Patients. For externally influenced variables such as Registered Deaths, LSTM with LLM provided marginal gains but required higher computational costs and showed a risk of overfitting.

6.3. Analysis of Patterns and Trends Observed

ARIMA was effective for stable variables like Mechanical Ventilation Bed Usage (MAPE: 9.93%) but struggled with volatile ones like Registered Deaths (MAPE: 223.77%) (Zohren, 2021). Meanwhile, LSTM with LLM offered limited gains for Number of Patients in Hospital (MAPE: 31.08%), indicating that increased model complexity does not always translate to improved accuracy (Rohitash Chandra, 2022). Interestingly, LSTM without LLM performed comparably well for Registered Deaths (MAPE: 26.62%), suggesting that simpler models are often sufficient when trends are primarily driven by historical data (Siami-Namini, et al., 2018). These findings reinforce that ARIMA is ideal for stable trends, while LSTM without LLM effectively captures non-linear patterns with lower computational costs (Brandon Wagner, 2023).

7. Chapter: Conclusions and Recommendations

7.1Key Findings

This study provides critical insights into the effectiveness of ARIMA, LSTM without LLM, and LSTM with LLM for forecasting healthcare demand during the COVID-19 pandemic. LSTM with LLM showed marginal improvements for variables like Registered Deaths but did not consistently outperform other models across all variables. ARIMA was effective for stable, linear patterns such as Mechanical Ventilation Bed Usage, demonstrating its suitability for predictable and steady trends. In contrast, LSTM without LLM handled non-linear patterns better than ARIMA, especially for variables like the Number of Patients, which had more irregular patterns.

7.2. Implications of the Results

The results underscore the importance of selecting models based on variable characteristics and operational needs. LSTM with LLM, though more complex, did not always result in significantly improved accuracy. In contrast, ARIMA and LSTM without LLM were more effective for variables with predictable patterns and moderate volatility.

This suggests that NHS England should adopt a strategic approach: leveraging LSTM with LLM for highly dynamic variables that require nuanced modeling and using simpler models like ARIMA or LSTM without LLM for routine planning. This would ensure efficient resource utilization during both crisis and non-crisis periods, balancing computational cost and forecast accuracy.

7.3 Theoretical Contributions

This research advances the understanding of time-series forecasting in healthcare by contrasting traditional methods like ARIMA with deep learning models such as LSTM. While ARIMA is reliable for capturing stable, linear trends, it struggles with complex patterns and abrupt changes, affirming prior research (Siami-Namini, et al., 2018) that deep learning models like LSTM excel at capturing non-linear trends. A key contribution of this study is in demonstrating that added model complexity does not always lead to better performance. For instance, LSTM with LLM showed only marginal gains in variables like Mechanical Ventilation Bed Usage, supporting the need to balance model complexity and forecast efficiency (Zohren, 2021).

This aligns with broader observations in the literature that suggest that more complex models should be employed only when necessary and where the cost of computational resources is justified by the complexity of the data patterns involved.

7.4 Recommendations

Recommendations for Future Pandemic Preparedness and Healthcare Forecasting:

The goal of this research was not just to assess predictive models but to provide actionable strategies that can help the NHS prepare for future pandemics more effectively. As a business analytics student with a deep commitment to supporting healthcare systems, I hope these recommendations lay the groundwork for transforming the way NHS England approaches resource planning and demand forecasting.

- **Adopt Advanced Context-Aware Forecasting Models for Strategic Planning**
  Use ARIMA for stable variables like Mechanical Ventilation Usage (Brandon Wagner, 2023) and LSTM for dynamic variables such as Hospital Admissions (Siami-Namini, et al., 2018).This hybrid approach ensures precise predictions based on variable characteristics.

- **Use Hybrid Models to Bridge Strategic and Operational Needs**
  Hybrid models, such as ARIMA-LSTM, combine the strengths of both linear and non-linear forecasting, making them suitable for managing complex healthcare scenarios (Sumalatha Saleti, 2024). This tiered system can switch between models based on demand complexity.

- **Enhance Forecasting for Routine Healthcare Management**
  Apply ARIMA for predicting routine events (e.g., seasonal flu) and LSTM for complex resource interactions, supporting balanced resource allocation and reducing staff burnout (Siami-Namini, et al., 2018).

- **Leverage Real-Time Data for Continuous Learning and Adaptation**
  Integrate real-time data and reinforcement learning to improve model responsiveness during rapidly evolving crises (Rohitash Chandra, 2022).This dynamic approach enhances decision-making, as demonstrated by Johns Hopkins' COVID-19 dashboard (University, 2023).

- **Plan for Long-Term Integration of AI in Healthcare Management**
  Expand AI use beyond crisis response to routine operations, optimizing clinical decision-making and resource management. By developing the NHS AI Lab, the NHS can stay at the forefront of predictive analytics for future health challenges (NHS England, 2024).

Implementing these recommendations will enable the NHS to build a resilient, adaptable healthcare system, capable of proactive resource management during both crises and routine operations.

8. Chapter: Limitations and Future Research Directions

Every research study has inherent limitations, and this research is no exception. Several constraints were identified, relating to the data, methods used, and the scope of the research.

8.1. Model Limitations

Data Availability and Consistency: The study used data from April 2020 to December 2021, excluding 2022 data due to inconsistencies, potentially missing out on long-term trends and the impact of new COVID-19 variants.

ARIMA's Limitation with Non-Linear Trends: ARIMA struggled to capture non-linear patterns and sudden changes, making it less suitable for complex healthcare scenarios.

Computational Complexity of LSTM: LSTM with LLM required significant computational resources, limiting its applicability for real-time forecasting in resource-constrained environments.

Overfitting Risks in LSTM Models: The complexity of LSTM models led to a higher risk of overfitting, even with regularization techniques applied.

8.3 Future Research Directions

While this research has offered valuable insights into healthcare demand forecasting using ARIMA, LSTM, and LSTM with LLM models, there are several opportunities for future studies to build upon these findings and address the limitations identified:

Hybrid Models: Future studies could explore combining ARIMA with LSTM or XGBoost to leverage both linear and non-linear pattern detection, improving performance in diverse contexts.

Expanded Datasets: Incorporate demographic data, regional breakdowns, and patient severity levels to capture variations across diverse groups and provide more accurate, generalizable forecasts.

Improved LLM Integration: Refine LLM models by using more specific external data (e.g., local health policies) to boost forecasting accuracy in dynamic environments.

By addressing these gaps, future research can contribute to more robust and adaptable healthcare forecasting models.

END

9. References

Abdur Rais, A. V., 2010. Operations Research in Healthcare: a survey. *International Transactions in Operational Research,* 18(1), pp. 1-31.

Adiga, A. D. D. L. B. e. a., 2020. Mathematical Models for COVID-19 Pandemic: A Comparative Analysis. *Journal of the Indian Institute of Science,* Volume 100, p. 793–807.

Alamo T, R. D. M. M. A. A., 2020. Covid-19: Open-Data Resources for Monitoring, Modeling, and Forecasting the Epidemic. *Electronics,* 9(5), p. 827.

Alsaeed Dalal, A.-O. A. ,. A. H. ,. A. F. ,. A.-O. E., 2023. Are we ready for the next pandemic? Lessons learned from healthcare professionals' perspectives during the COVID-19 pandemic. *Frontiers in Public Health,* Volume 11, pp. 2296-2565.

Anna Jobin, M. I. E. V., 2019. Artificial Intelligence: the global landscape of ethics guidelines. *Computer Science, Philosophy,* Volume abs/1906.11668.

Anon., n.d. *Python Software Foundation. Python Language Reference, version 3.x.* [Online]
Available at: http://www.python.org

Anon., n.d. *SciPy User Guide.* [Online]
Available at: https://docs.scipy.org/doc/scipy/tutorial/index.html

Bertsimas, D. B. L. C.-W. R. e. a., 2021. From predictions to prescriptions: A data-driven response to COVID-19. *Health Care Management Science,* Volume 24, p. 253–272.

Botchkarev, A., 2019. A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management,* Volume 14, pp. 045-076.

Box, G. a. J., 2013. Box and Jenkins: Time Series Analysis, Forecasting and Control. *A Very British Affair ,* p. 161–215.

Brandon Wagner, K. C., 2023. Using autoregressive integrated moving average models for time series analysis of observational data. *BMJ,* Volume 383.

Brownlee, J., 2019. How to Scale Data for Long Short-Term Memory Networks in Python. *Deep Learning for Time Series.*

Chung, J. &. G. C. &. C. K. &. B. Y., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.

Courville, I. G. a. Y. B. a. A., 2016. *Deep Learning.* s.l.:MIT Press.

Diederik P. Kingma, J. B., 2014. Adam: A Method for Stochastic Optimization. *Machine Learning.*

Duarte, D. W. C. &. R. N., 2021. A Comparison of Time-Series Predictions for Healthcare Emergency Department Indicators and the Impact of COVID-19. *Applied Sciences ,* 11(8), p. 3561.

Duley, M. G. K., 2005. The Next Pandemic: Anticipating an. *YALE JOURNAL OF BIOLOGY AND MEDICINE,* Volume 78, pp. 351-358.

Elsheikh, A. et al., 2021. Artificial Intelligence for Forecasting the Prevalence of COVID-19 Pandemic: An Overview. *Healthcare,* 9(12), p. 1614.

Erwin J. Khoo, J. D. L., 2020. Lessons learned from the COVID-19 pandemic. *Acta Paediatrica,* 109(7), pp. 1323-1325.

G. M. LJUNG, G. E. P. B., 1978. On a measure of lack of fit in time series models. *Biometrika,* 65(2), p. 297–303.

G. M. LJUNG, G. E. P. B., 1978. On a measure of lack of fit in time series models. *Biometrika,* 65(2), p. 297–303.

Grote, T. K. G., 2022. Enabling Fairness in Healthcare Through Machine Learning. *Ethics and Information Technology,* Volume 24, p. article number 39.

Haneen Alabdulrazzaq, M. N. A. Y. R. B. A. A. A. A. A.-H. F. S. A.-A., 2021. On the accuracy of ARIMA based prediction of COVID-19 spread. *Results in Physics,* Volume 27, pp. 2211-3797.

Harris, C. M. K. v. d. W. S. e. a., 2020. Array programming with NumPy. Nature. Volume 585, pp. 357-362.

Hunter, J. D., 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering,* 9(3), pp. 90-95.

Ian Goodfellow, Y. B. a. A. C., 2017. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. *Genetic Programming and Evolvable Machines,* 19(2018), p. 305–307.

Iebeling Kaastra, M. B., 1996. Designing a neural network for forecasting financial and economic time series. *Neurocomputing,* 10(3), pp. 215-236.

Ilemobayo, J. D. O. A. O. A. O. A. T. F. O. O. A. O. A. O. D. I. A. O. I. &. E. O., 2024. Hyperparameter Tuning in Machine Learning: A Comprehensive Review. *Journal of Engineering Research and Reports,* 26(6), pp. 388-395.

Ilie OD, C. R. C. A. T. S. M. I. D. B., 2020. Forecasting the Spreading of COVID-19 across Nine Countries from Europe, Asia, and the American Continents Using the ARIMA Models. *Microorganisms,* 8(8), p. 1158.

John P.A. Ioannidis, S. C. M. A. T., 2022. Forecasting for COVID-19 has failed. *International Journal of Forecasting,* 38(2), pp. 423-438.

Kalloubi Fahd, H. B. L. S. E. &. A. M., 2024. Forecasting stock market returns using deep learning and time series techniques: a comparative and empirical study using technical indicators. *Multimedia Tools and Applications .*

Kaushik Shruti, C. A. ,. S. P. K. ,. D. N. ,. N. S. ,. P. L. A. ,. D. V., 2020. AI in Healthcare: Time-Series Forecasting Using Statistical, Neural, and Ensemble Architectures. *Front. Big Data Sec. Medicine and Public Health,* Volume 3.

Kumar, R., Jain, A., Tripathi, A. K. & Tyagi, S., 2021. COVID-19 Outbreak: An Epidemic Analysis using Time Series Prediction Model. *11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2021,* pp. 1090-1094.

Kumar, R. K. P. &. K. Y., 2022. Multi-step time series analysis and forecasting strategy using ARIMA and evolutionary algorithms. *International Journal of Information Technology ,* Volume 14, p. 359–373.

Luciano Floridi, J. C. M. B. R. C. P. C. V. D. C. L. R. M. U. P. F. R. B. S. P. V. &. E. V., 2018. AI4People—An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations. *Minds and Machines,* Volume 28, p. 689–707.

Martín Abadi, A. A. P. B. E. B. Z. C. C. C. G. S. C. A. D. J. D. M. D. S. G. I. G. A. H. G. I. M. I. Y. J. R. J. L., 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *Computer Science > Distributed, Parallel, and Cluster Computing,* pp. 265-283.

Ma, R. Z. X. W. P. e. a., 2021. The prediction and analysis of COVID-19 epidemic trend by combining LSTM and Markov method.. *Scientific Reports,* Volume 11.

Matt J. Keeling, E. M. H. ,. E. G. P. G.-F. H. L. M. T. D. ,. J. T., 2021. Predictions of COVID-19 dynamics in the UK: Short-term forecasting and analysis of potential exit strategies. *Plos Computational Biology.*

McKinney, W., 2010. Data Structures for Statistical Computing in Python. *Computer Science, Mathematics.*

Mehdi Khashei, M. B., 2011. A novel hybridization of artificial neural networks and ARIMA models for time series forecasting. *Applied Soft Computing,* 11(2), pp. 2664-2675.

Mohamadou, Y. H. A. &. K. P., 2020. A review of mathematical modeling, artificial intelligence and datasets used in the study, prediction and management of COVID-19. *Artificial intelligence,* Volume 50, p. 3913–3925.

Mohammad Fattahi, E. K. D. K. K. G., 2023. Resource planning strategies for healthcare systems during a pandemic. *European Journal of Operational Research,* 304(1), pp. 192-206.

Mohd Saqib, E. Ş. S. A. S. &. M. A. A., 2022. Comparisons of autoregressive integrated moving average (ARIMA) and long short term memory (LSTM) network models for ionospheric anomalies detection: a study on Haiti (Mw = 7.0) earthquake. *Acta Geodaetica et Geophysica,* 57(2022), p. 195–213.

Mukherjee, S. B. &. S., 2022. A Comparative Study of Seasonal-ARIMA and RNN (LSTM) on Time Series Temperature Data Forecasting. *Pervasive Computing and Social Networking,* p. 315–326.

Newbold, P., 1983. ARIMA model building and the time series analysis approach to forecasting. *Journal of Forecasting,* 2(1), pp. 23-35.

NHS England, 2024. *NHS Artificial Intelligence Laboratory (AI Lab).* [Online]
Available at: https://transform.england.nhs.uk/ai-lab/
[Accessed June 2024].

Nur Izzati Ab Kader, U. K. Y. M. N. A. K. &. N. R. N. H., (2023. *A Review of Long Short-Term Memory Approach for Time Series Analysis and Forecasting.* s.l., s.n., p. 12–21.

Nurhafiza Md Hamzah, M.-M. Y. &. K. F. S., 2021. Assessing the efficiency of Malaysia health system in COVID-19 prevention and treatment response. *Health Care Management Science,* p. 273–285.

Office, I. C., 2020. *Information Commissioner's Office.* [Online]
Available at: https://ico.org.uk/for-organisations/data-protection-and-the-eu/data-protection-and-the-eu-in-detail/the-uk-gdpr/

O'Malley, T. a. B. E. a. L. J. a. C. F. a. J. H. a. I. L. a. o., 2019. *KerasTuner.* [Online]
Available at: \url{https://github.com/keras-team/keras-tuner

Paki, H. A. &. R., 2021. Improving time series forecasting using LSTM and attention models. *Journal of Ambient Intelligence and Humanized Computing,* 13(2022), p. 673–691.

Pal, V. C. S., 2022. Application of machine learning time series analysis for prediction COVID-19 pandemic. *Research on Biomedical Engineering,* 38(March 2022), p. 35–47.

Paul Birrell, J. B. E. v. L. N. G. a. D. D. A., 2021. Real-time nowcasting and forecasting of COVID-19 dynamics in England: the first wave. *Royal Society,* 376(1829).

Pedregosa, F. V. G. G. A. e. a., 2012. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research,* pp. 2825-2830.

Pham, T. T. T. P. D. V. S., 2016. DeepCare: A Deep Dynamic Memory Model for Predictive Medicine.. *Advances in Knowledge Discovery and Data Mining Conference paper,* Volume 9652, p. 30–41.

Ping-Feng Pai, C.-S. L., 2005. A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega,* 33(6), pp. 497-505.

Rabiolo A, A. E. M. E. M. A. B. F. A. A. M. A., 2021. Forecasting the COVID-19 Epidemic by Integrating Symptom Search Behavior Into Predictive Models: Infoveillance Study. *Infoveillance Study. J Med Internet Res. 2021,* 23(8), p. e28876.

Rohit Salgotra, M. G. A. H. G., 2020. Time Series Analysis and Forecast of the COVID-19 Pandemic in India using Genetic Programming. *Chaos, Solitons & Fractals,* Volume 138, p. 109945.

Rohitash Chandra, A. J. ,. S. C., 2022. Deep learning via LSTM models for COVID-19 infection forecasting in IndiaDeep learning via LSTM models for COVID-19 infection forecasting in India. *PLOS ONE,* 17(1).

Said, A. E. A. A. H. e. a., 2021. Predicting COVID-19 cases using bidirectional LSTM on multivariate time series.. *Environmental Science and Pollution Research ,* Volume 28, p. 56043–56052.

Seabold, S. &. P. J., 2010. *Statsmodels: Econometric and Statistical Modeling with Python..* s.l., Proceedings of the 9th Python in Science Conference.

Sepp Hochreiter, J. S., 1997. Long Short-term Memory. *Neural Computation,* 9(8), pp. 1735 - 1780.

Shaharudin SM, I. S. H. N. T. M. a. S. N., 2021. Short-Term Forecasting of Daily Confirmed COVID-19 Cases in Malaysia Using RF-SSA Model. *Front. Public Health Sec. Infectious Diseases – Surveillance, Prevention and Treatment,* 9(2021).

Siami-Namini, S., Tavakoli, N. & Namin, A. S., 2018. A Comparison of ARIMA and LSTM in Forecasting Time Series. *17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA,* pp. 1394-1401.

Singh RK, R. M. B. A. S. R. R.-M. A. K. H. N. C. S. S. S. Y. R. A. R. J. K. P., 2020. Prediction of the COVID-19 Pandemic for the Top 15 Affected Countries: Advanced Autoregressive Integrated Moving Average (ARIMA) Model. *JMIR Public Health Surveill.*

Smith, T. G. e. a., 2017. pmdarima: ARIMA estimators for Python.

Specht, D., 1991. A general regression neural network. *IEEE Transactions on Neural Networks,* 2(6), pp. 568-576.

Srivastava, N., 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research.,* 15(1), pp. 1929-1958.

Sujata Dash, C. C. S. K. G. S. K. P., 2021. Intelligent computing on time-series data analysis and prediction of COVID-19 pandemics. *Pattern Recognition Letters,* 151(0167-8655), pp. 69-75.

Sullivan, T., 2024. Lessons (Machine) Learned From COVID-19. *The Journal of Infectious Diseases,* 229(1), p. 7–9.

Sumalatha Saleti, L. Y. P. Y. R. K. L. P. &. S. J., 2024. Enhancing Forecasting Accuracy with a Moving Average-Integrated Hybrid ARIMA-LSTM Model. *SN Computer Science,* Volume 5.

Technologies, P., n.d. [Online]
Available at: https://dash.plotly.com/introduction)

Timina Liu, S. L. &. L. S., 2020. ARIMA Modelling and Forecasting. *Time Series Analysis Using SAS Enterprise Guide,* p. 61–85.

Tom B. Brown, B. M. N. R. M. S. J. K. P. D. A. N. P. S. G. S. A. A. S. A. A. H.-V. G. K. T. H. R. C. A. R. D., 2020. *Language Models are Few-Shot Learners.* s.l., s.n.

University, C. f. S. S. a. E. (. a. J. H., 2023. *COVID-19 Dashboard.* [Online]
Available at: https://coronavirus.jhu.edu/map.html
[Accessed 2024].

Virtanen, P. G. R. O. T. H. M. R. T. C. D. B. E. P. P. W. W. B. J. v. d. W. S. B. M. W. J. M. K. M. N. N. A. J. E. K. R. L., 2020. Fundamental Algorithms for Scientific Computing in Python.. *Nature methods,* 17(3), pp. 261-272.

Yuehui Chen, B. Y. J. D. A. A., 2005. Time-series forecasting using flexible neural tree model. *Information Sciences,* 174(3-4), pp. 219-235.

Zachary C. Lipton, D. C. K. C. E. R. W., 2015. Learning to Diagnose with LSTM Recurrent Neural Networks. *Computer Science > Machine Learning.*

Zhang, G., 2003. Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model.. *Neurocomputing,* 50(17), pp. 159-175.

Zohair Malki, E.-S. A. A. E. G. D. A. R. A. G. E. M. A. E. A. E. H. &. I. G., 2020. ARIMA models for predicting the end of COVID-19 pandemic and the risk of second rebound. *Neural Computing and Applications,* Volume 33, p. 2929–2948.

Zohren, B. L. a. S., 2021. Time-series forecasting withdeep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences,* 379(2194).

10. Appendix:

A. Data processing:

Following are the visualization of the complete dataset:

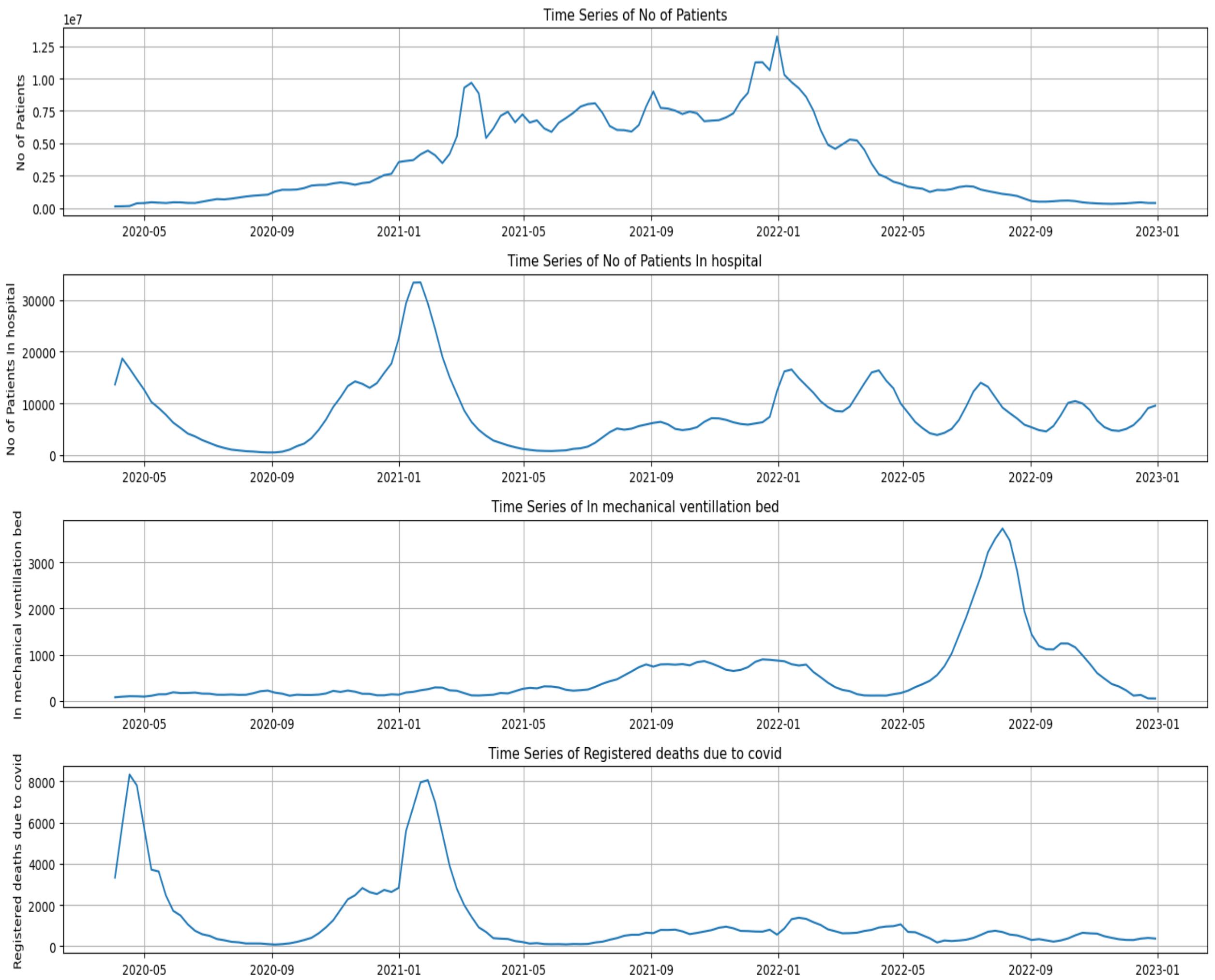


Figure 33 Time Series Plot of Original Dataset

The time series plots for key healthcare variables visualize significant trends observed throughout the COVID-19 pandemic. The Number of Patients graph shows major peaks around early 2021, representing major pandemic waves. The Number of Patients in Hospital plot reveals corresponding peaks but with more granular fluctuations, highlighting specific surges in hospital admissions. For Patients in Mechanical Ventilation Beds, a critical rise in mid-2021 indicates the peak of the crisis. The Registered Deaths Due to COVID graph shows distinct mortality waves, with prominent peaks also observed in early 2021.

A1: Distribution plot of complete dataset

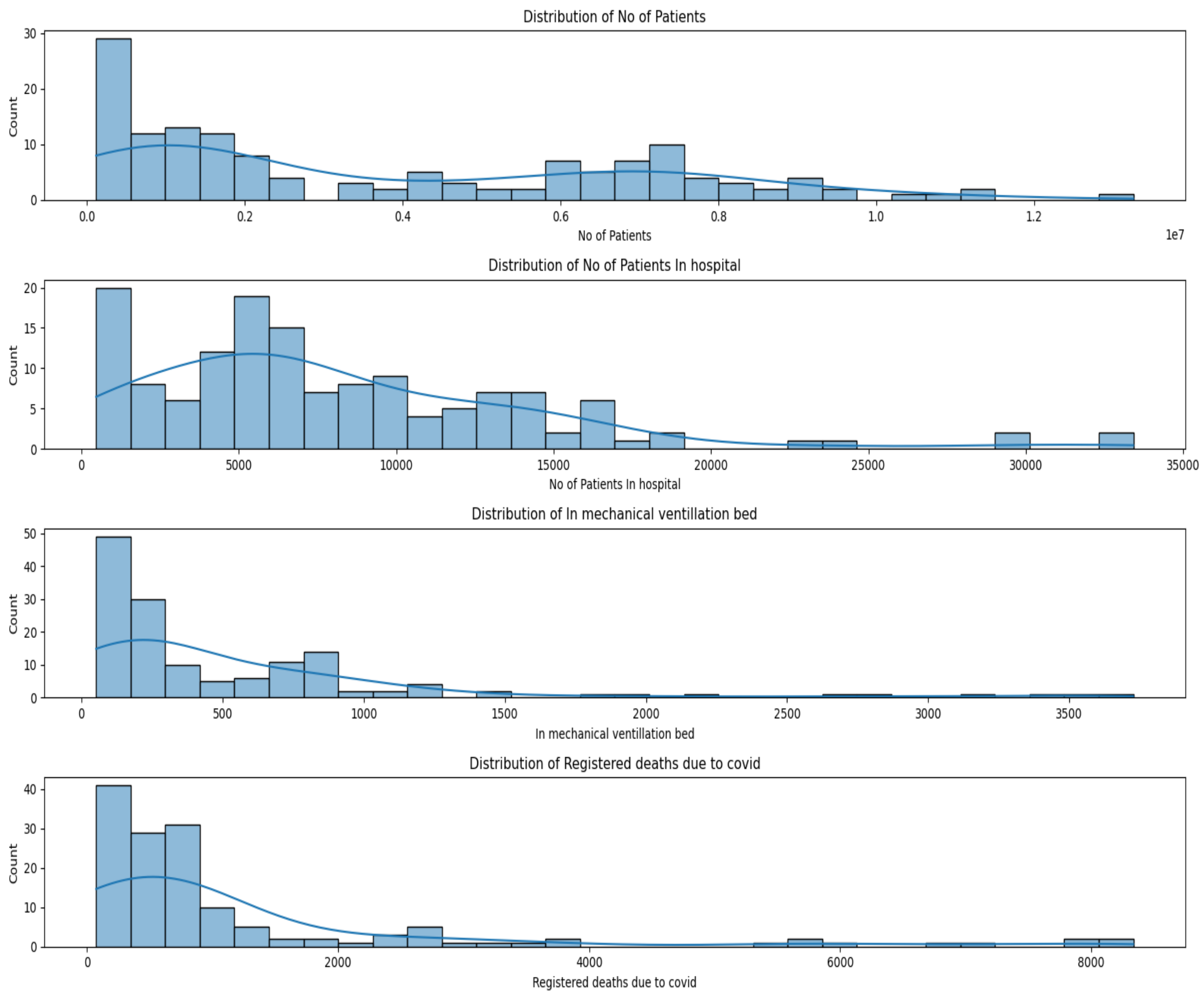


Figure 34 Distribution of Original dataset.

Distribution of No of Patients: This histogram illustrates the frequency distribution of the total number of patients, indicating a higher concentration of lower patient counts. The distribution displays a long right tail, which is attributed to the presence of a few high-value outliers.

Distribution of No of Patients in Hospital: The hospital admissions distribution shows a similar pattern to the total patients, with a right-skewed distribution, suggesting most values are clustered towards the lower range.

Distribution of Patients in Mechanical Ventilation Bed: The histogram of mechanical ventilation usage shows many cases with low to moderate usage, with fewer instances of higher utilization.

Distribution of Registered Deaths Due to COVID: This histogram shows that most death counts are on the lower end, with a few high outliers indicating periods of particularly severe pandemic impact.

A2. Boxplot of complete dataset

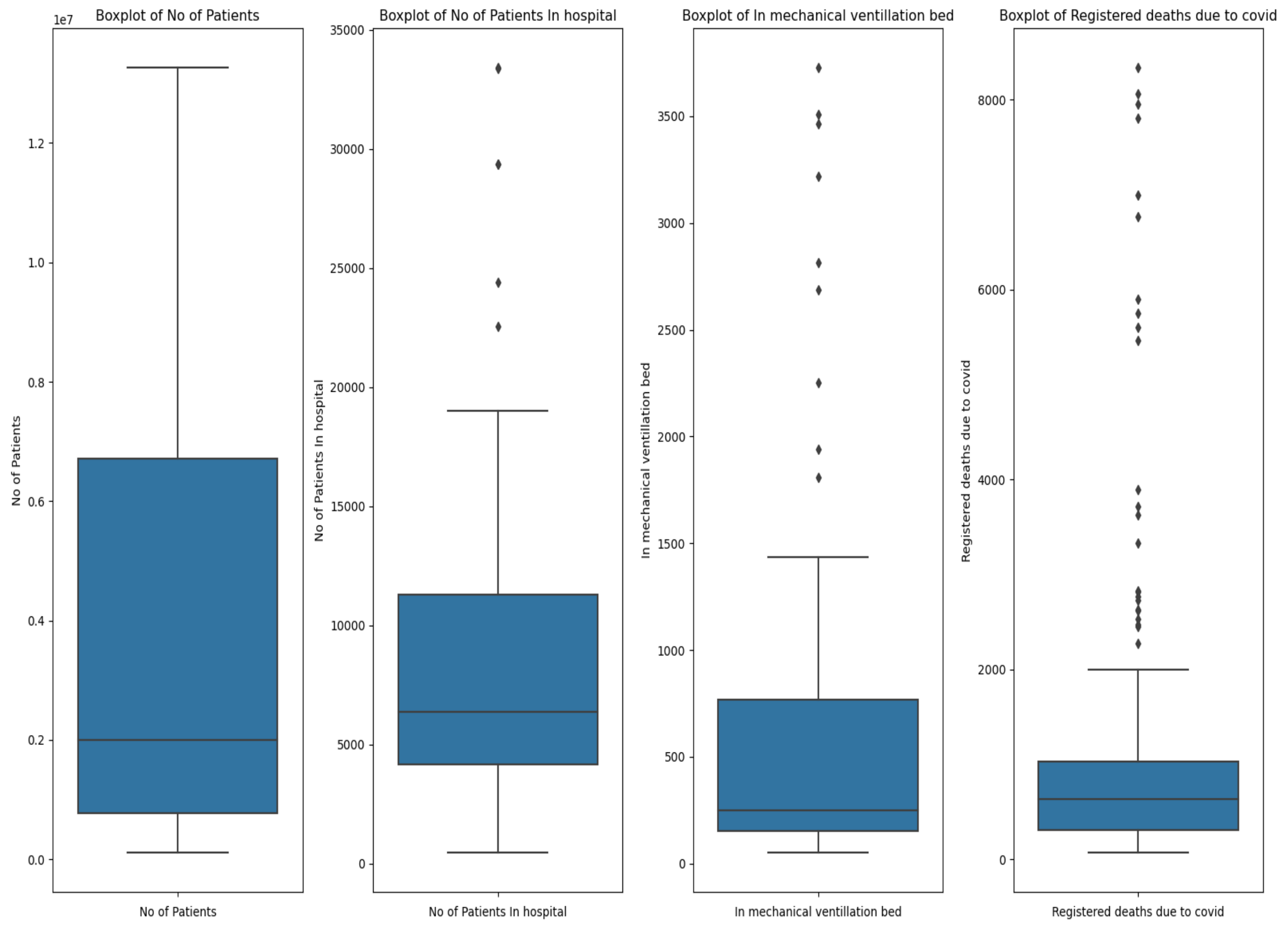


Figure 35 Boxplots of Original Dataset

Boxplot of No of Patients: This boxplot highlights the distribution and spread of total patient numbers, showing the interquartile range and presence of a few significant outliers in the upper range.

Boxplot of No of Patients in Hospital: The hospital admissions boxplot reveals a relatively narrower interquartile range, with several higher outliers corresponding to periods of increased hospital demand.

Boxplot of Patients in Mechanical Ventilation Bed: This boxplot shows that while most values lie within a moderate range, a few extreme values highlight surges in ventilator bed usage during peak pandemic times.

Boxplot of Registered Deaths Due to COVID: This boxplot indicates that while most recorded deaths are clustered in the lower range, there are a few high-value outliers, representing significant mortality surges during key pandemic periods.

A3. Time series plot of truncated dataset

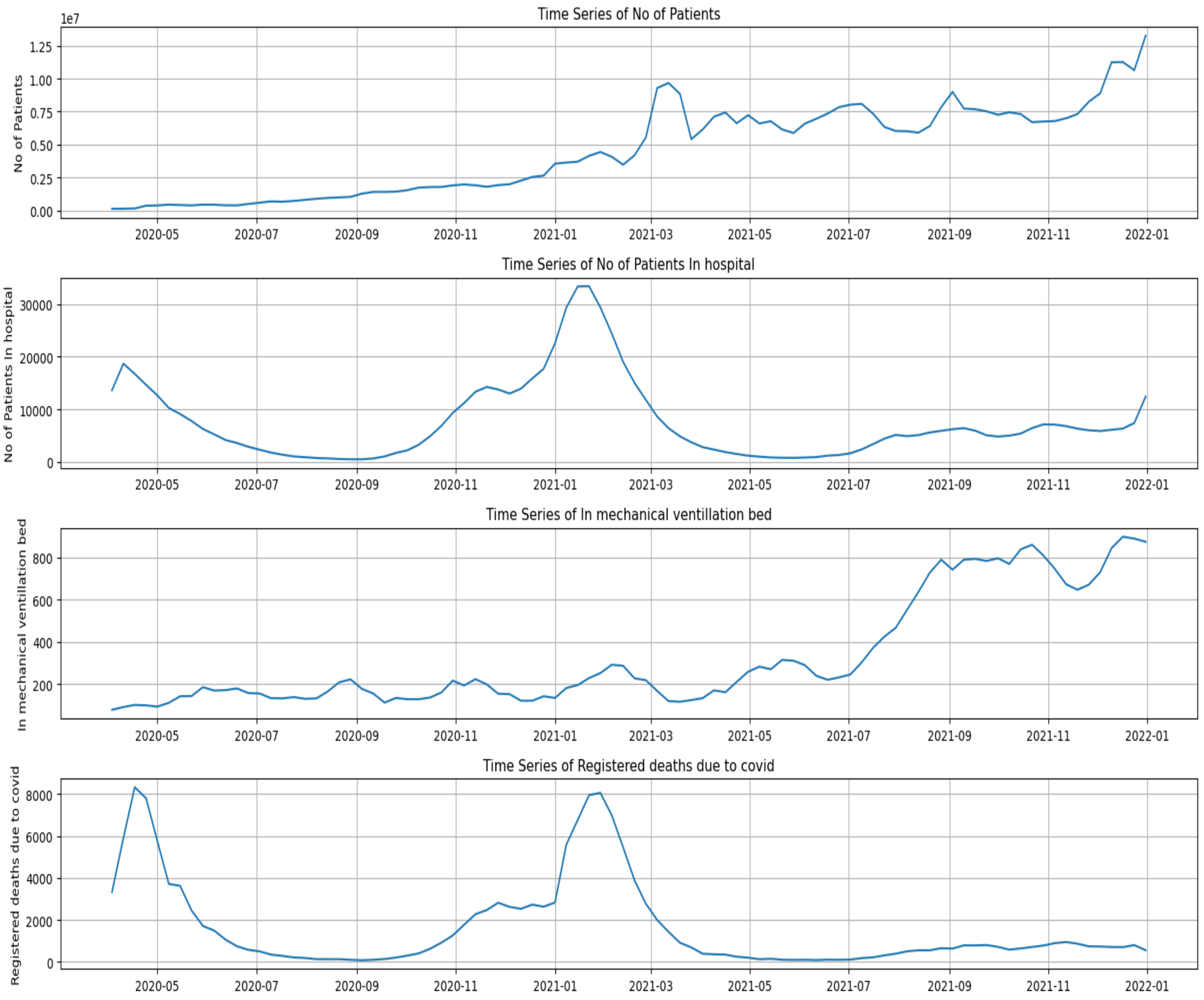


Figure 36 Time Series Plots of Truncated Dataset

No of Patients: There is a visible upward trend over time, with spikes around early 2021. This indicates an overall increase in the number of patients over the period.

No of Patients In hospital: This series shows a seasonal pattern with peaks in early 2021 and a sudden drop afterward, followed by another gradual increase. This could be due to several waves of the pandemic.

In mechanical ventilation bed: This variable shows a relatively stable trend until mid-2021 with a gradual increase over time.

Registered deaths due to COVID: There is a clear peak in early 2021, corresponding to the rise in cases and hospitalization. After the peak, the overall trend remains stable at lower values.

A4 Distribution Plot of truncated dataset:

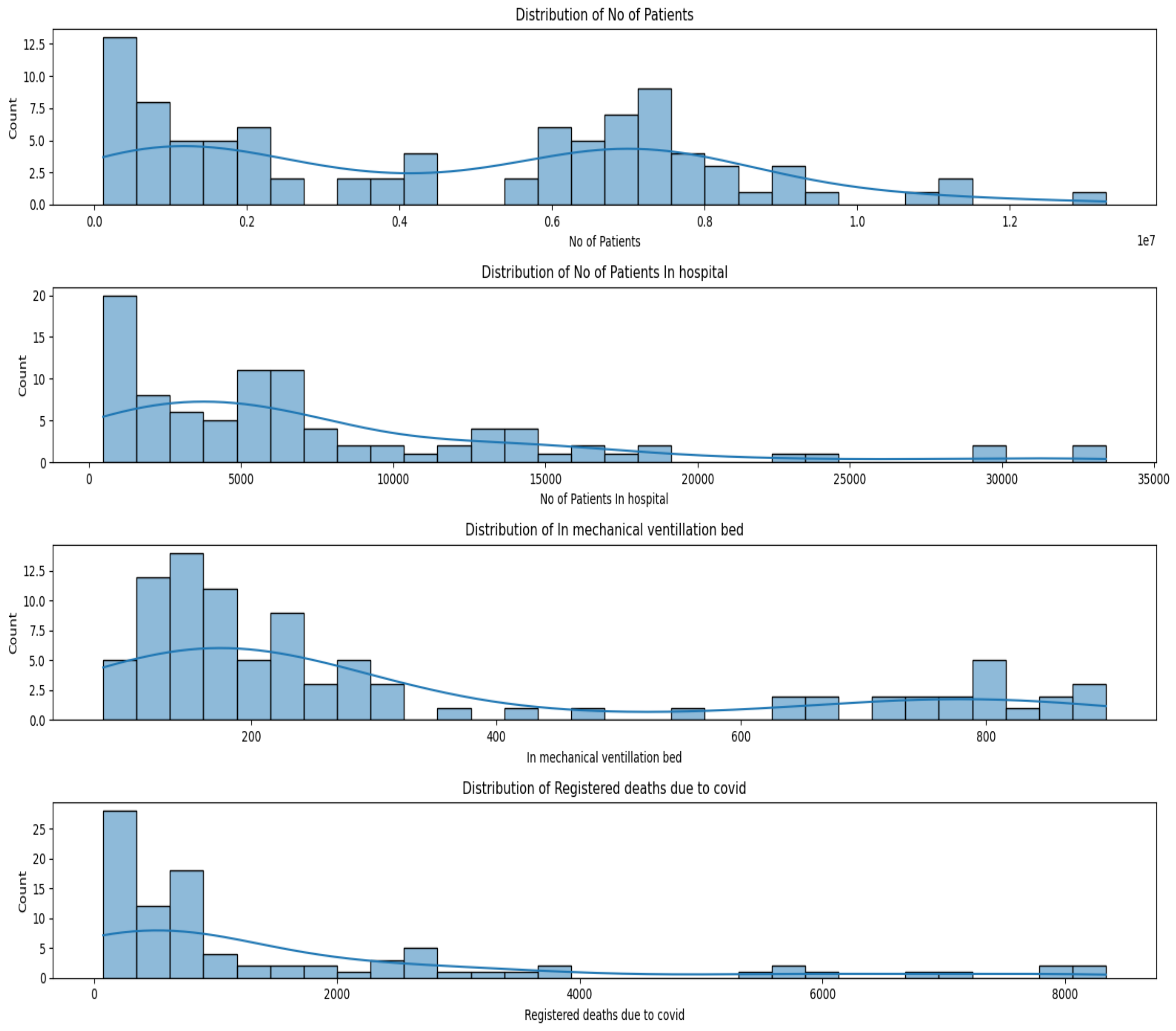


Figure 37 Distribution Plots of Truncated Dataset

The distribution graphs similarly reflect this adjusted timeframe, providing a cleaner and more accurate representation of patient counts, hospitalizations, mechanical ventilation usage, and deaths due to COVID. By removing later data, these graphs concentrate on the core pandemic periods when patient volumes were higher and the patterns more predictable. This adjustment helps maintain a focus on the most impactful periods without the interference of anomalous data from the later stages of the pandemic.

A5. Boxplot of truncated Dataset:

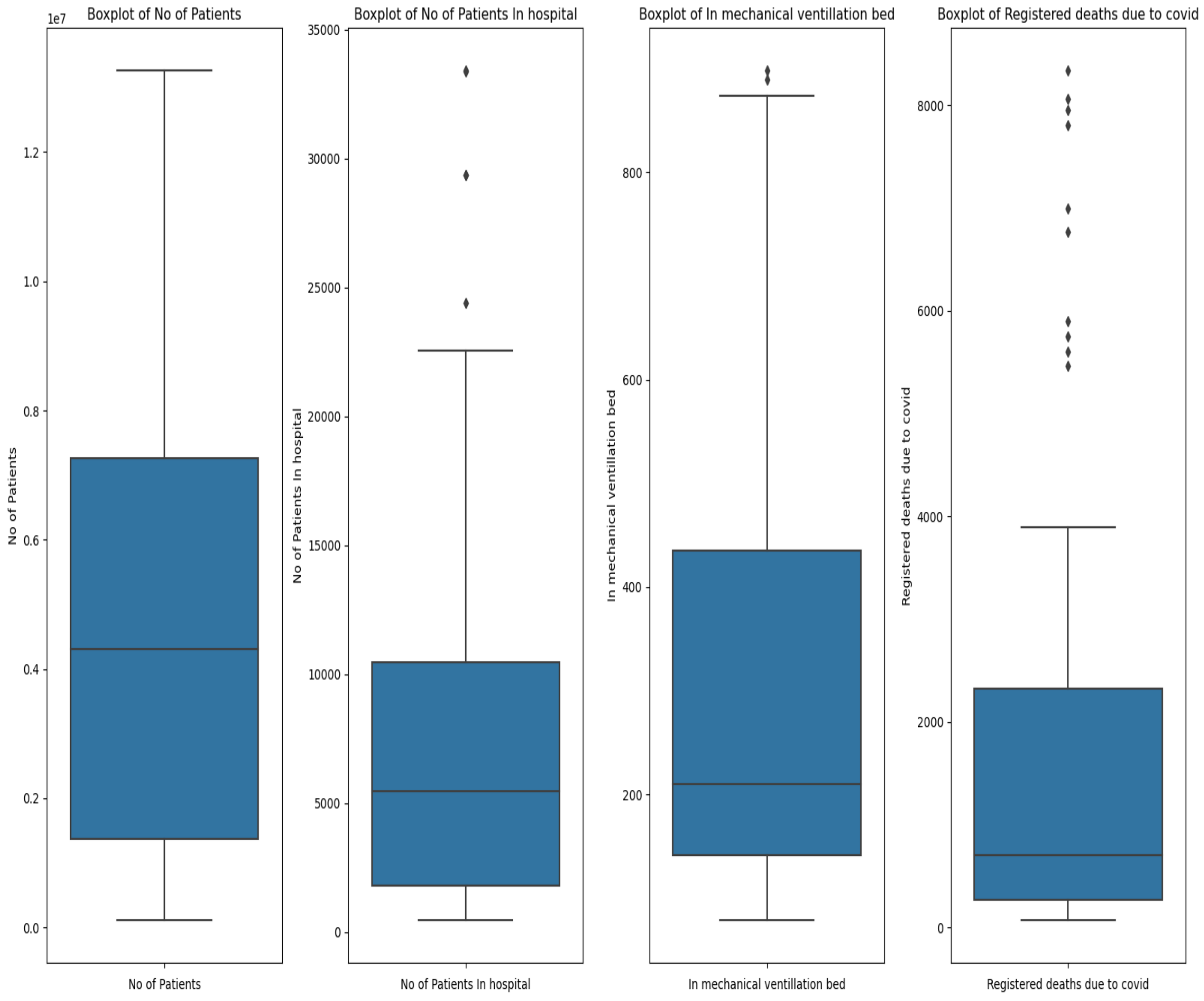


Figure 38 Boxplots of Truncated Dataset

Finally, the boxplots summarize the spread of the data up to December 2021, offering a visual representation of key measures like central tendency and variability. By excluding data points from later periods that introduced potential outliers, these boxplots provide a more accurate reflection of trends during the pandemic's peak periods, ensuring the analysis remains robust and the conclusions derived from the data are dependable.

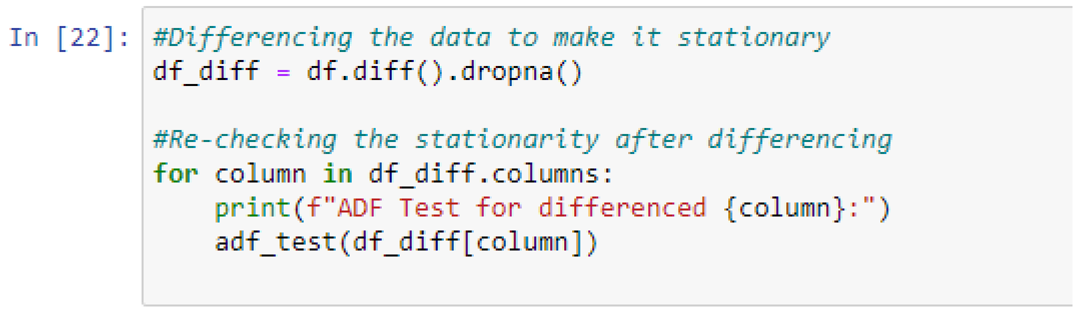
A6. The truncated dataset until 2021

https://liveastonac-my.sharepoint.com/:x:/g/personal/230284735_aston_ac_uk/EakFKuIoFWJBq60zQGjovS4Bsf6rT3m7YqjvvxxLMdgJpQ?e=FoPIey

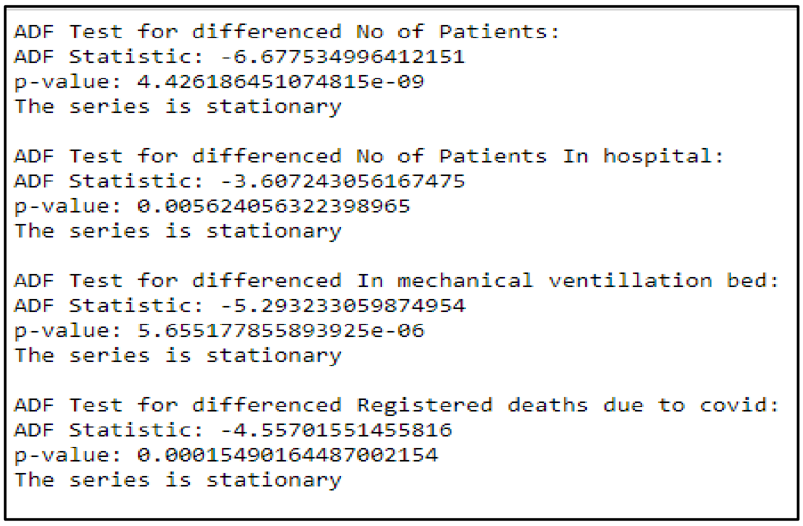
A7. Sampling Methods:

The dataset was not sampled in the traditional sense, as all available data for the relevant time frame was included to maintain the integrity of the time series analysis. Rather than sampling, the focus was on data segmentation and ensuring that each variable used in the forecasting models (e.g., number of patients in hospitals, mechanical ventilation use, registered deaths) was well-represented across different pandemic waves. Data from the initial phases of the pandemic, where the spread of COVID-19 was exponential, was weighted to emphasize the extreme healthcare demands during this time. Variables were segmented based on these pandemic waves to analyse both peak and trough periods of demand.

A8. ADF Test for Differencing the data:



```
In [22]: #Differencing the data to make it stationary
         df_diff = df.diff().dropna()

         #Re-checking the stationarity after differencing
         for column in df_diff.columns:
             print(f"ADF Test for differenced {column}:")
             adf_test(df_diff[column])
```



```
ADF Test for differenced No of Patients:
ADF Statistic: -6.677534996412151
p-value: 4.426186451074815e-09
The series is stationary

ADF Test for differenced No of Patients In hospital:
ADF Statistic: -3.607243056167475
p-value: 0.005624056322398965
The series is stationary

ADF Test for differenced In mechanical ventillation bed:
ADF Statistic: -5.293233059874954
p-value: 5.655177855893925e-06
The series is stationary

ADF Test for differenced Registered deaths due to covid:
ADF Statistic: -4.55701551455816
p-value: 0.00015490164487002154
The series is stationary
```

B Appendix: Software Tools and Libraries Used

This section outlines the software tools and libraries utilized for data analysis, modeling, and visualization. Each tool and library served a specific purpose in facilitating the research workflow.

Tool	**Purpose**
Excel	Initial data cleaning, organization, and preliminary data exploration.
Python	Main programming language for modeling, analysis, and visualization, utilizing various libraries.
Jupyter Notebooks	Facilitated the execution and documentation of code, enabling transparency and reproducibility.
ChatGPT by OpenAI	Providing guidance, feedback, and structured support for organizing content and refining sections of the research.

Table 1 Software Tools and Libraries Used

- **Python Libraries for ARIMA Analysis**

The ARIMA models were developed using the following Python libraries:

Library	**Functionality**	**Reference**
Pandas	Data manipulation and preparation, particularly for handling time-series data.	(McKinney, 2010)
Statsmodels	Building and evaluating ARIMA models, performing parameter tuning, and statistical tests.	(Seabold, 2010)
Matplotlib	Visualization of time series, residuals, and forecast results.	(Hunter, 2007)
SciPy	Conducting statistical tests and mathematical evaluations for model analysis.	(Virtanen, 2020)

Scikit-learn	Calculation of evaluation metrics, including MAE, MSE, and RMSE.	(Pedregosa, 2012)
pmdarima	Automating the ARIMA parameter selection and fitting processes (Auto ARIMA).	(Smith, 2017)

Table 2 Python Libraries for ARIMA Analysis

- **Python Libraries for LSTM (GRU) Models**

The LSTM and GRU models were built and optimized using the following libraries:

Library	Functionality	Reference
TensorFlow/Keras	Framework for developing, training, and optimizing deep learning models.	(Martín Abadi, 2016)
NumPy	Numerical calculations, handling arrays, and matrix operations essential for LSTM/GRU models.	(Harris, 2020)
Matplotlib	Visualizing model performance, including loss curves and forecast outputs.	(Hunter, 2007)
SciPy	Statistical evaluations and mathematical functions used in model optimization.	(Virtanen, 2020)
Scikit-learn	Performance metrics calculation (RMSE, MAE, MAPE).	(Pedregosa, 2012)

Table 3 Python Libraries for LSTM (GRU) Models

- **LSTM with LLM Integration**

The integration of Large Language Models (LLMs) into the LSTM architecture was achieved using:

Library	Functionality	Reference

TensorFlow/Keras	Seamless integration of LLMs with deep learning models for enhanced time-series forecasting.	(Martín Abadi, 2016)

Table 4 LSTM with LLM Integration

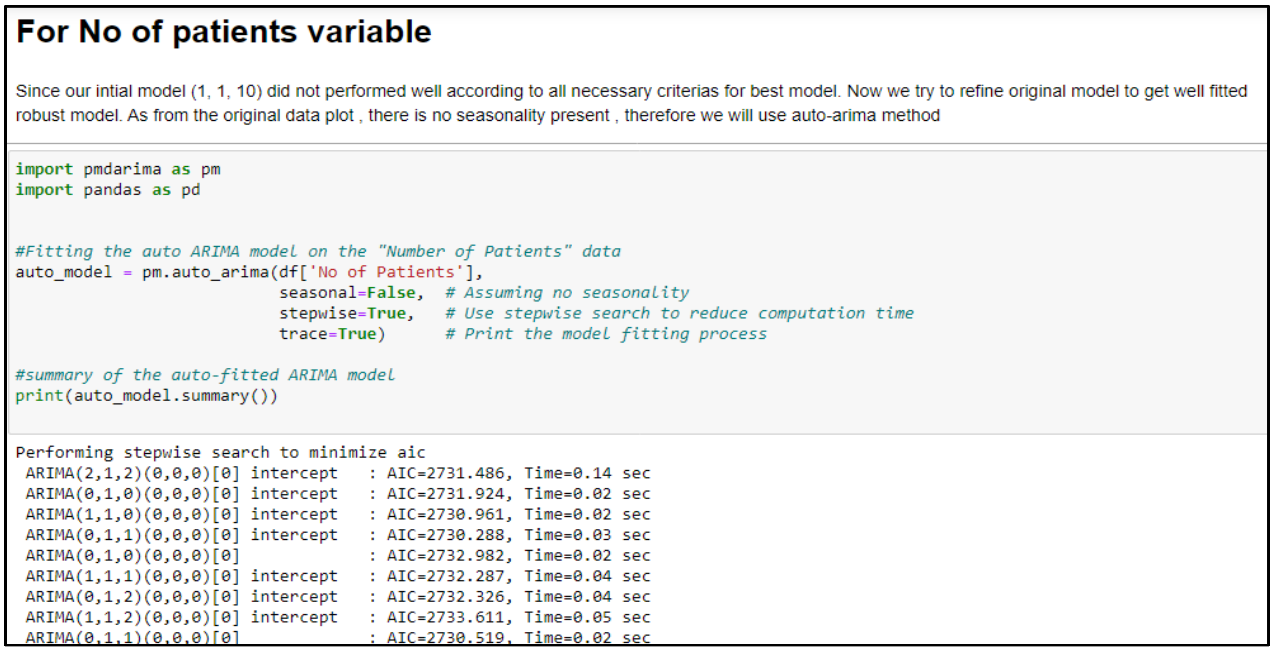
- **Dashboard Development for Visualization:**

The performance and forecasts of the models were visualized using a customized dashboard, which was built using Python libraries:

Library	Purpose	Reference
Matplotlib	Visualization of comparison plots and trends.	(Hunter, 2007)
Seaborn	Enhanced data visualizations for complex relationships and trends.	(Technologies, n.d.)

Table 5 Dashboard Development for Visualization libraries used.

C. ARIMA analysis for one variable code snippets

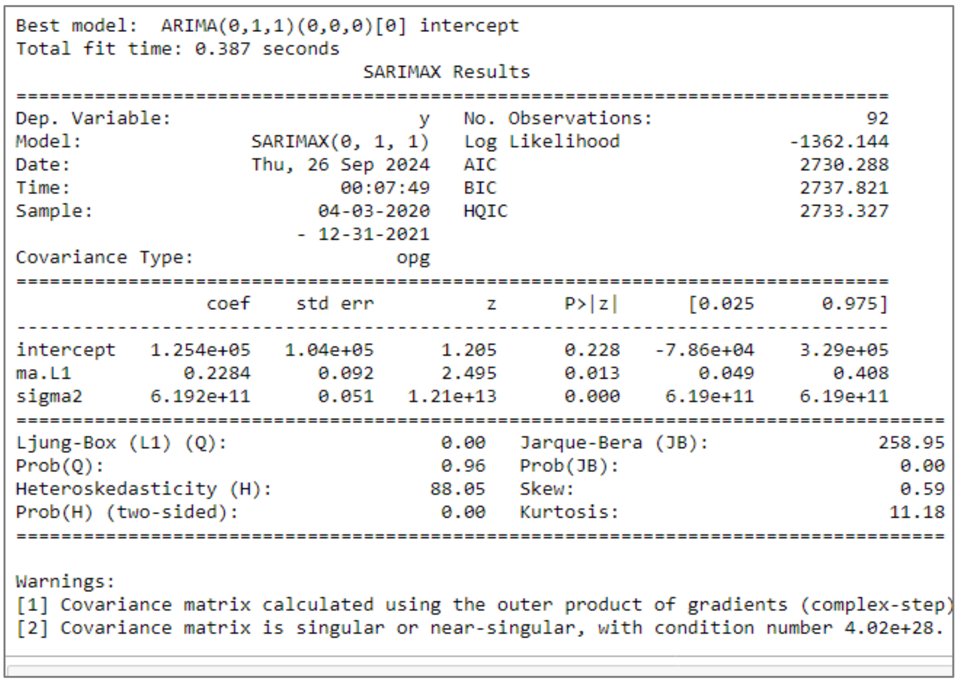


For No of patients variable

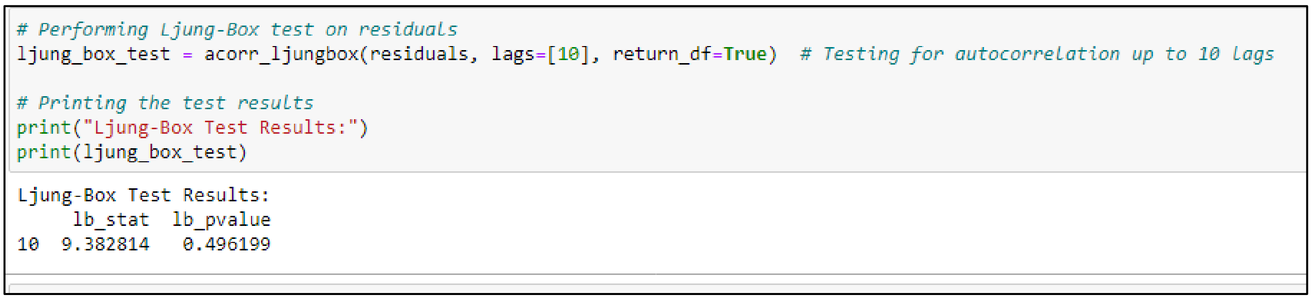
Since our intial model (1, 1, 10) did not performed well according to all necessary criterias for best model. Now we try to refine original model to get well fitted robust model. As from the original data plot , there is no seasonality present , therefore we will use auto-arima method

```
import pmdarima as pm
import pandas as pd


#Fitting the auto ARIMA model on the "Number of Patients" data
auto_model = pm.auto_arima(df['No of Patients'],
                           seasonal=False,  # Assuming no seasonality
                           stepwise=True,   # Use stepwise search to reduce computation time
                           trace=True)      # Print the model fitting process

#summary of the auto-fitted ARIMA model
print(auto_model.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=2731.486, Time=0.14 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=2731.924, Time=0.02 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=2730.961, Time=0.02 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=2730.288, Time=0.03 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=2732.982, Time=0.02 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=2732.287, Time=0.04 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=2732.326, Time=0.04 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=2733.611, Time=0.05 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=2730.519, Time=0.02 sec
```

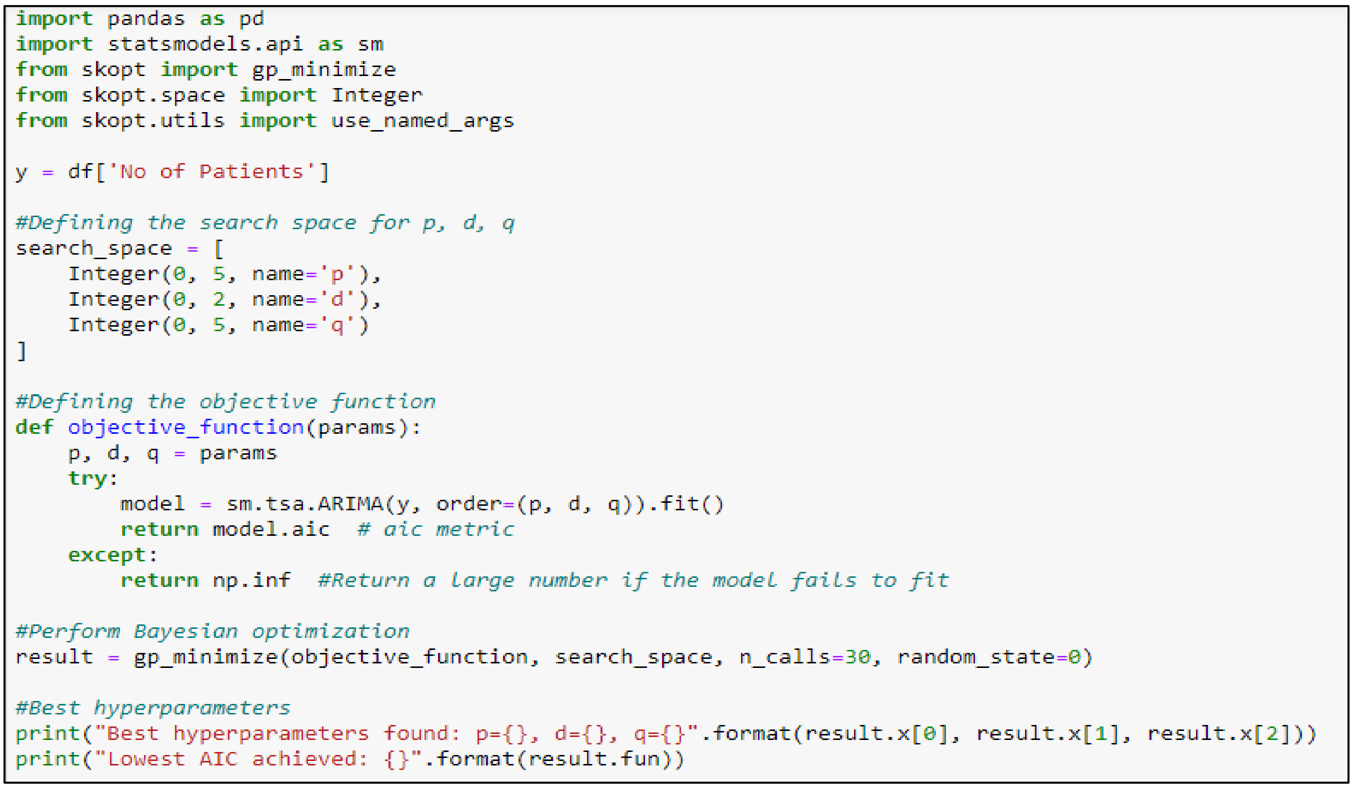


```
Best model:  ARIMA(0,1,1)(0,0,0)[0] intercept
Total fit time: 0.387 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   92
Model:               SARIMAX(0, 1, 1)   Log Likelihood               -1362.144
Date:                Thu, 26 Sep 2024   AIC                           2730.288
Time:                        00:07:49   BIC                           2737.821
Sample:                    04-03-2020   HQIC                          2733.327
                         - 12-31-2021
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept   1.254e+05   1.04e+05      1.205      0.228   -7.86e+04    3.29e+05
ma.L1          0.2284      0.092      2.495      0.013       0.049       0.408
sigma2      6.192e+11      0.051   1.21e+13      0.000    6.19e+11    6.19e+11
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):               258.95
Prob(Q):                              0.96   Prob(JB):                         0.00
Heteroskedasticity (H):              88.05   Skew:                             0.59
Prob(H) (two-sided):                  0.00   Kurtosis:                        11.18
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step)
[2] Covariance matrix is singular or near-singular, with condition number 4.02e+28.
```

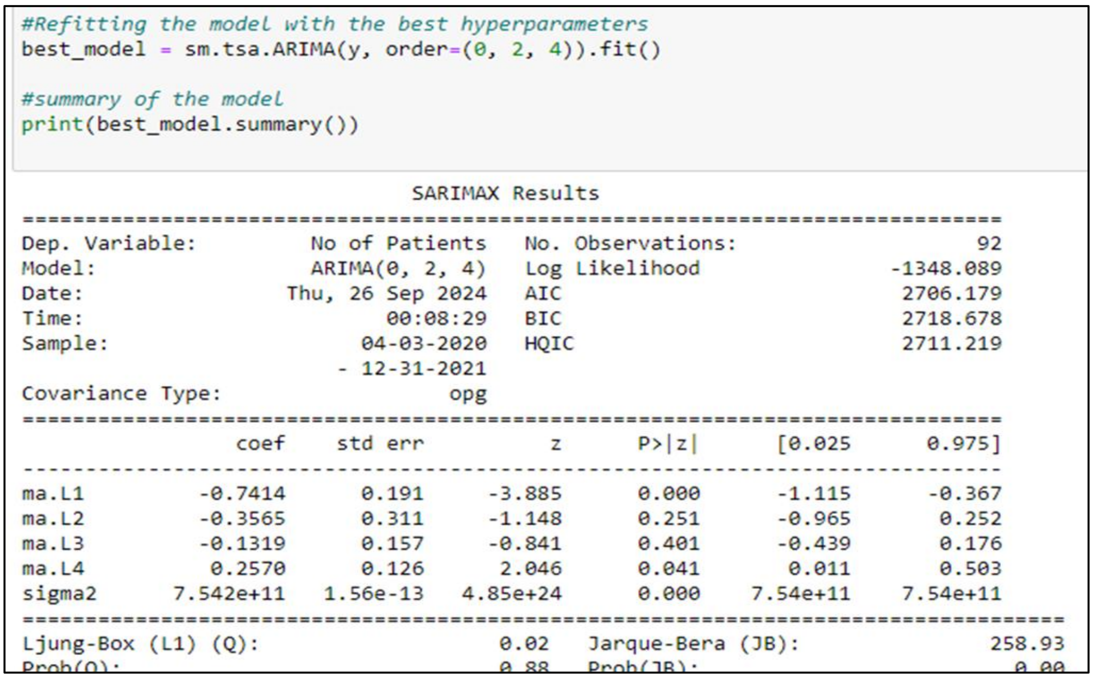


```python
# Performing Ljung-Box test on residuals
ljung_box_test = acorr_ljungbox(residuals, lags=[10], return_df=True)  # Testing for autocorrelation up to 10 lags

# Printing the test results
print("Ljung-Box Test Results:")
print(ljung_box_test)
```

```
Ljung-Box Test Results:
     lb_stat  lb_pvalue
10  9.382814   0.496199
```

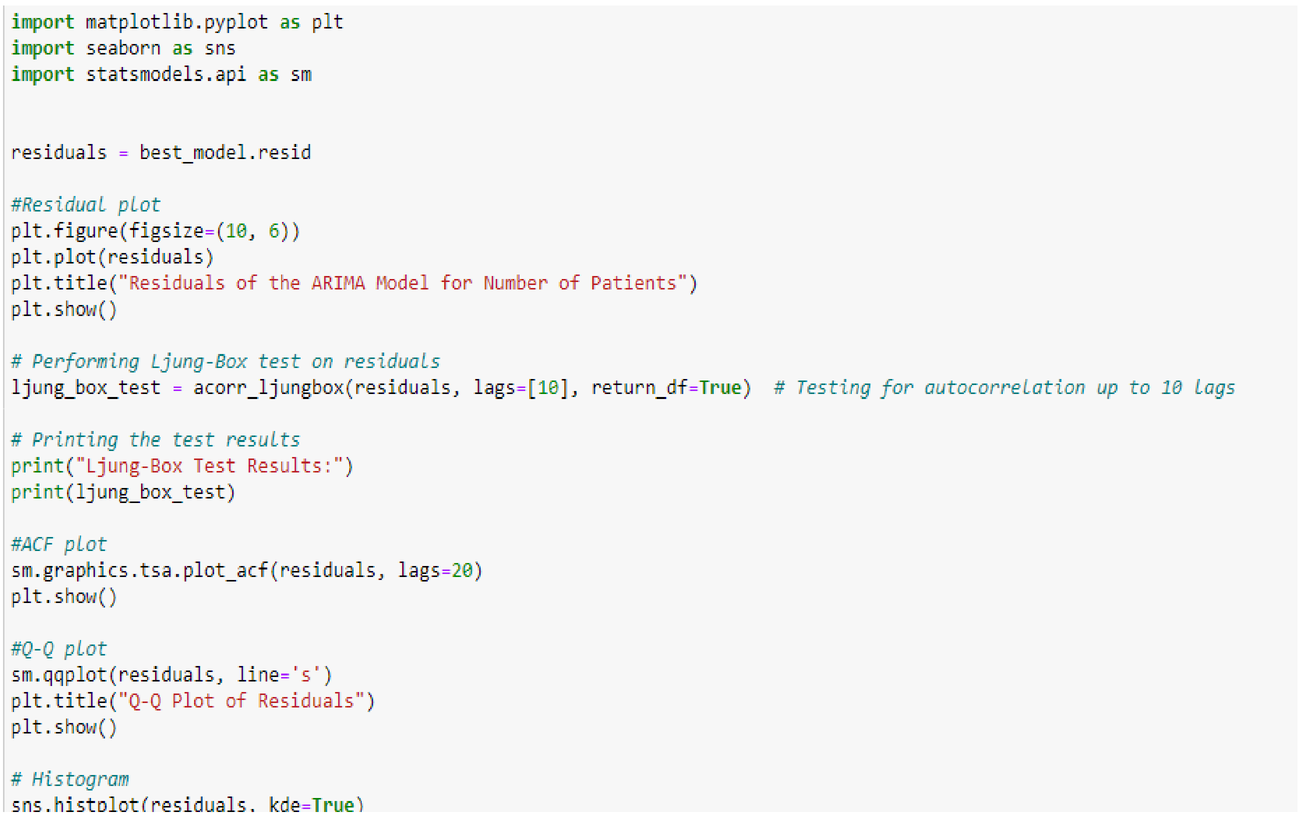


```python
import pandas as pd
import statsmodels.api as sm
from skopt import gp_minimize
from skopt.space import Integer
from skopt.utils import use_named_args

y = df['No of Patients']

#Defining the search space for p, d, q
search_space = [
    Integer(0, 5, name='p'),
    Integer(0, 2, name='d'),
    Integer(0, 5, name='q')
]

#Defining the objective function
def objective_function(params):
    p, d, q = params
    try:
        model = sm.tsa.ARIMA(y, order=(p, d, q)).fit()
        return model.aic  # aic metric
    except:
        return np.inf  #Return a large number if the model fails to fit

#Perform Bayesian optimization
result = gp_minimize(objective_function, search_space, n_calls=30, random_state=0)

#Best hyperparameters
print("Best hyperparameters found: p={}, d={}, q={}".format(result.x[0], result.x[1], result.x[2]))
print("Lowest AIC achieved: {}".format(result.fun))
```

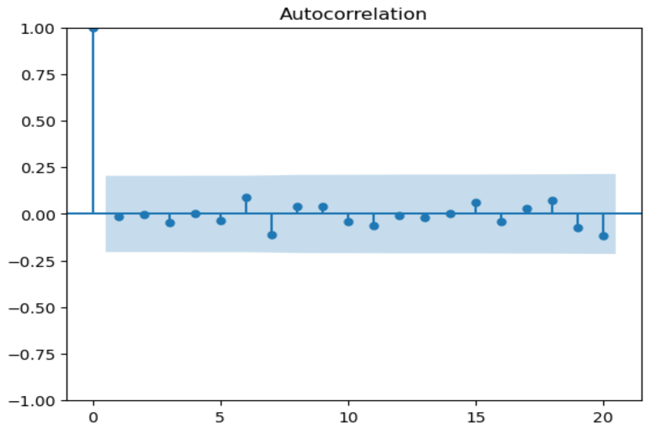


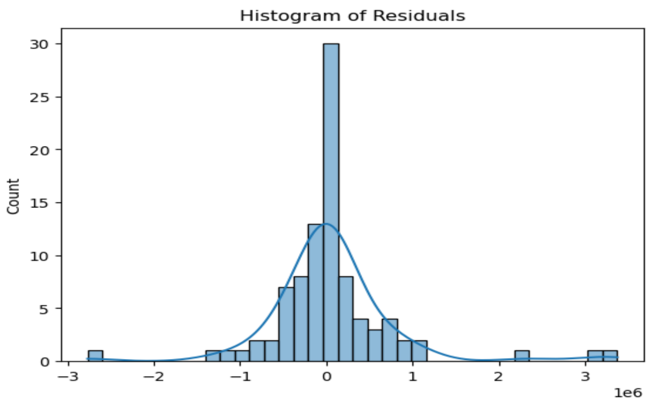
```
#Refitting the model with the best hyperparameters
best_model = sm.tsa.ARIMA(y, order=(0, 2, 4)).fit()

#summary of the model
print(best_model.summary())
```

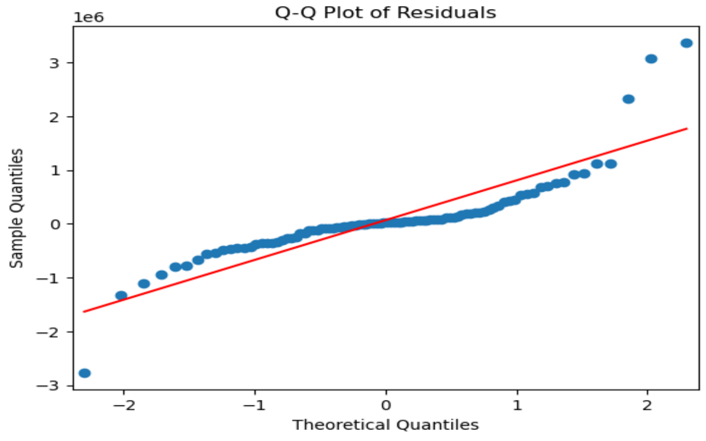
```
                               SARIMAX Results
==============================================================================
Dep. Variable:          No of Patients   No. Observations:                   92
Model:                  ARIMA(0, 2, 4)   Log Likelihood               -1348.089
Date:                 Thu, 26 Sep 2024   AIC                           2706.179
Time:                         00:08:29   BIC                           2718.678
Sample:                     04-03-2020   HQIC                          2711.219
                          - 12-31-2021
Covariance Type:                   opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.7414      0.191     -3.885      0.000      -1.115      -0.367
ma.L2         -0.3565      0.311     -1.148      0.251      -0.965       0.252
ma.L3         -0.1319      0.157     -0.841      0.401      -0.439       0.176
ma.L4          0.2570      0.126      2.046      0.041       0.011       0.503
sigma2      7.542e+11   1.56e-13   4.85e+24      0.000    7.54e+11    7.54e+11
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):               258.93
Prob(Q):                              0.88   Prob(JB):                         0.00
```



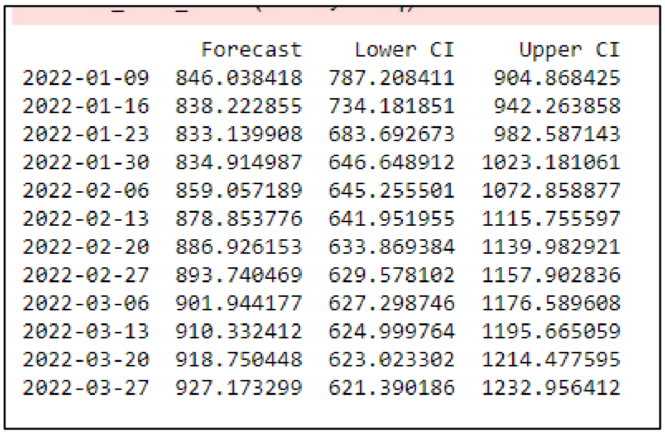
```
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm


residuals = best_model.resid

#Residual plot
plt.figure(figsize=(10, 6))
plt.plot(residuals)
plt.title("Residuals of the ARIMA Model for Number of Patients")
plt.show()

# Performing Ljung-Box test on residuals
ljung_box_test = acorr_ljungbox(residuals, lags=[10], return_df=True)  # Testing for autocorrelation up to 10 lags

# Printing the test results
print("Ljung-Box Test Results:")
print(ljung_box_test)

#ACF plot
sm.graphics.tsa.plot_acf(residuals, lags=20)
plt.show()

#Q-Q plot
sm.qqplot(residuals, line='s')
plt.title("Q-Q Plot of Residuals")
plt.show()

# Histogram
sns.histplot(residuals, kde=True)
```



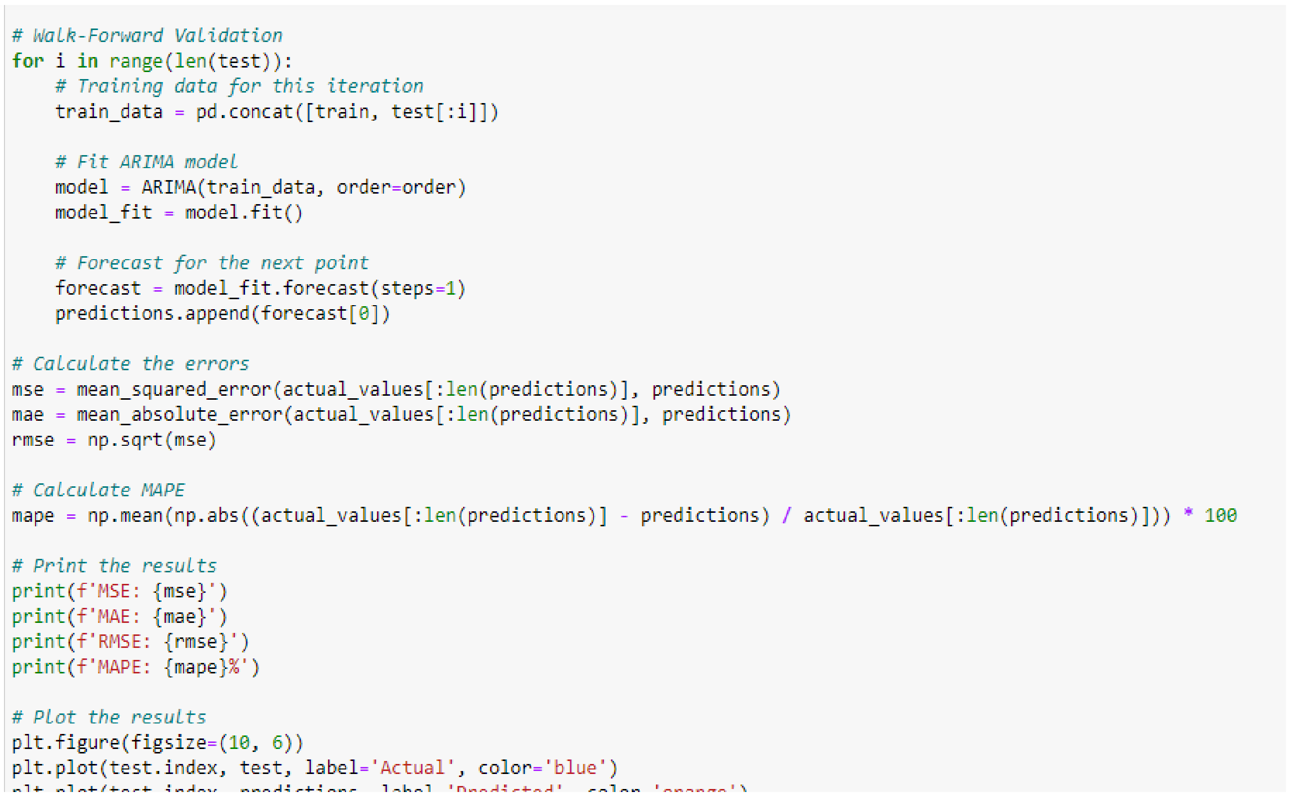




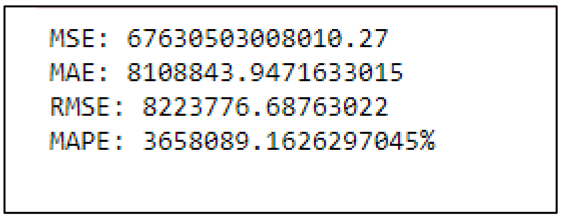
```
import pandas as pd
import matplotlib.pyplot as plt


#Fitting the ARIMA model to the full dataset
model = ARIMA(df['No of Patients'], order=(0, 2, 4))
model_fit = model.fit()

#Forecast for the next 12 weeks
forecast_steps = 12
forecast = model_fit.get_forecast(steps=forecast_steps)
forecast_index = pd.date_range(start=df.index[-1], periods=forecast_steps + 1, freq='W')[1:]

#the forecasted values and confidence intervals
forecast_values = forecast.predicted_mean
confidence_intervals = forecast.conf_int()

print(forecast_df)

# Save the forecast to a CSV file
forecast_df.to_csv('arima_forecast_variable_1.csv')


# Plot the forecast along with the historical data
plt.figure(figsize=(10, 6))
plt.plot(df.index, df['No of Patients'], label='Historical Data')
plt.plot(forecast_index, forecast_values, label='Forecast', color='orange')
plt.fill_between(forecast_index, confidence_intervals.iloc[:, 0], confidence_intervals.iloc[:, 1], color='orange', alpha=0.3)
plt.title('Forecast for Number of Patients')
plt.xlabel('Date')
plt.ylabel('Number of Patients')
```



```
            Forecast    Lower CI     Upper CI
2022-01-09  846.038418  787.208411   904.868425
2022-01-16  838.222855  734.181851   942.263858
2022-01-23  833.139908  683.692673   982.587143
2022-01-30  834.914987  646.648912  1023.181061
2022-02-06  859.057189  645.255501  1072.858877
2022-02-13  878.853776  641.951955  1115.755597
2022-02-20  886.926153  633.869384  1139.982921
2022-02-27  893.740469  629.578102  1157.902836
2022-03-06  901.944177  627.298746  1176.589608
2022-03-13  910.332412  624.999764  1195.665059
2022-03-20  918.750448  623.023302  1214.477595
2022-03-27  927.173299  621.390186  1232.956412
```

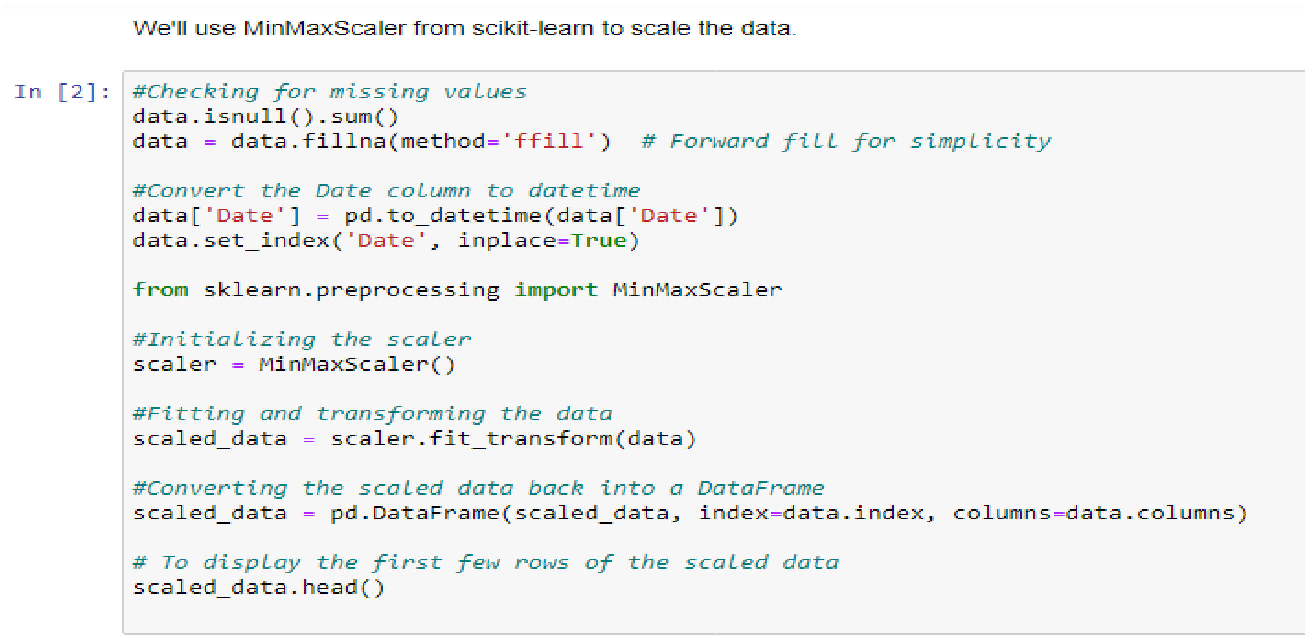


```python
# Walk-Forward Validation
for i in range(len(test)):
    # Training data for this iteration
    train_data = pd.concat([train, test[:i]])

    # Fit ARIMA model
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()

    # Forecast for the next point
    forecast = model_fit.forecast(steps=1)
    predictions.append(forecast[0])

# Calculate the errors
mse = mean_squared_error(actual_values[:len(predictions)], predictions)
mae = mean_absolute_error(actual_values[:len(predictions)], predictions)
rmse = np.sqrt(mse)

# Calculate MAPE
mape = np.mean(np.abs((actual_values[:len(predictions)] - predictions) / actual_values[:len(predictions)])) * 100

# Print the results
print(f'MSE: {mse}')
print(f'MAE: {mae}')
print(f'RMSE: {rmse}')
print(f'MAPE: {mape}%')

# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(test.index, test, label='Actual', color='blue')
plt.plot(test.index, predictions, label='Predicted', color='orange')
```



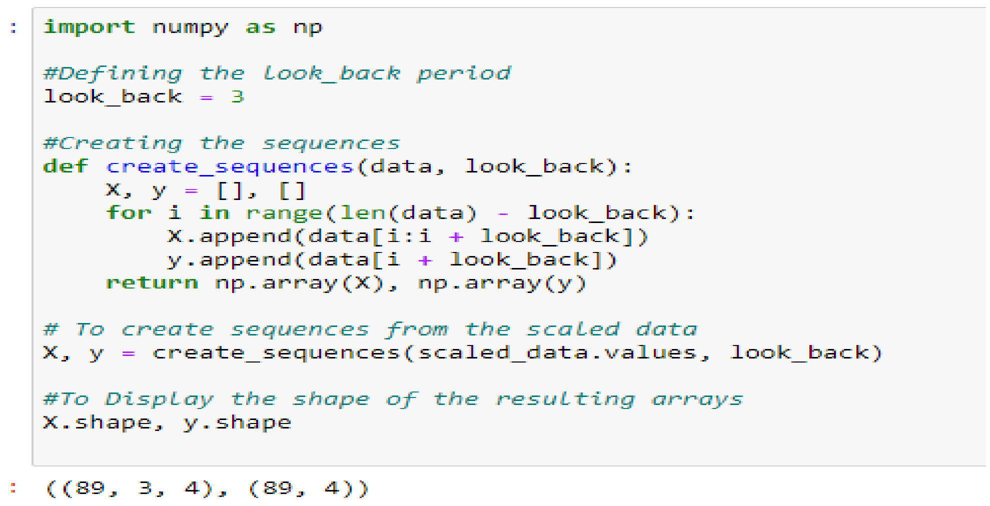
```
MSE: 67630503008010.27
MAE: 8108843.9471633015
RMSE: 8223776.68763022
MAPE: 3658089.1626297045%
```

D. LSTM with LLM important code snippets

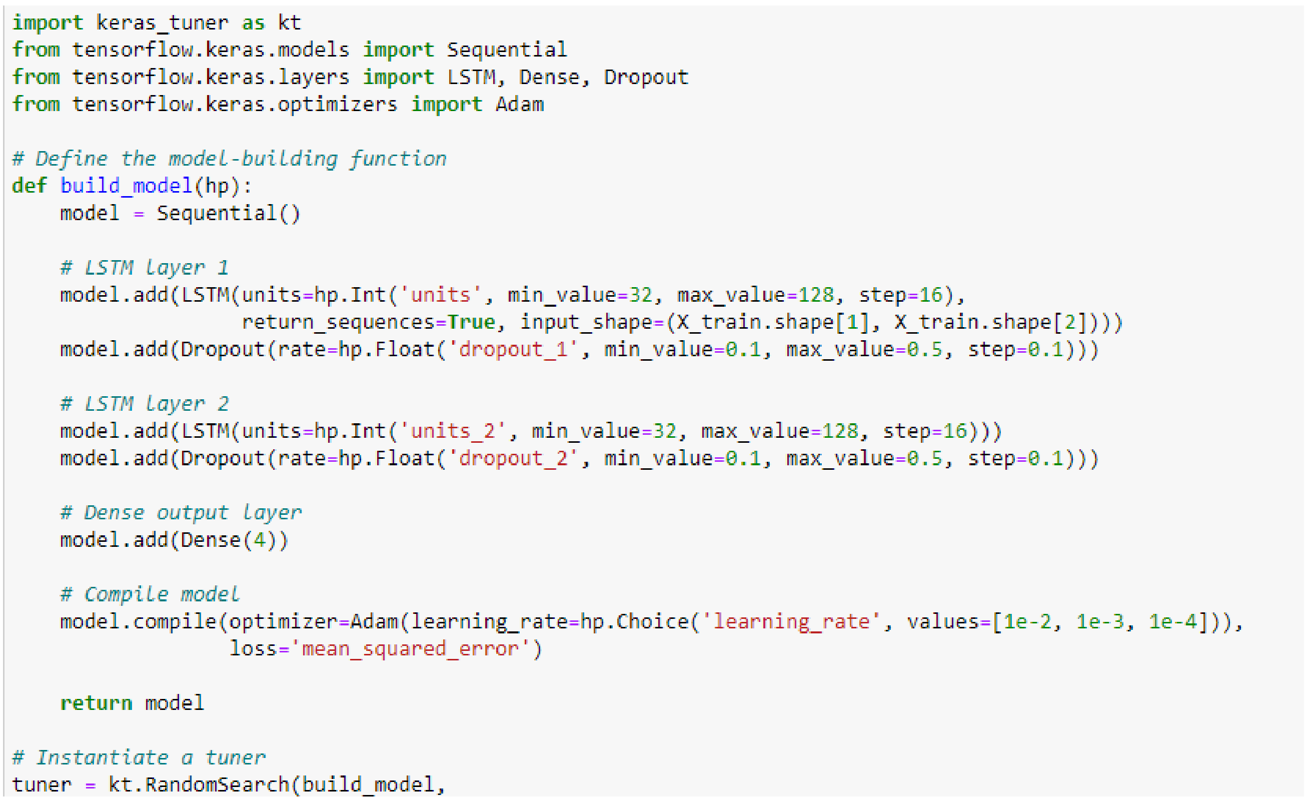


We'll use MinMaxScaler from scikit-learn to scale the data.

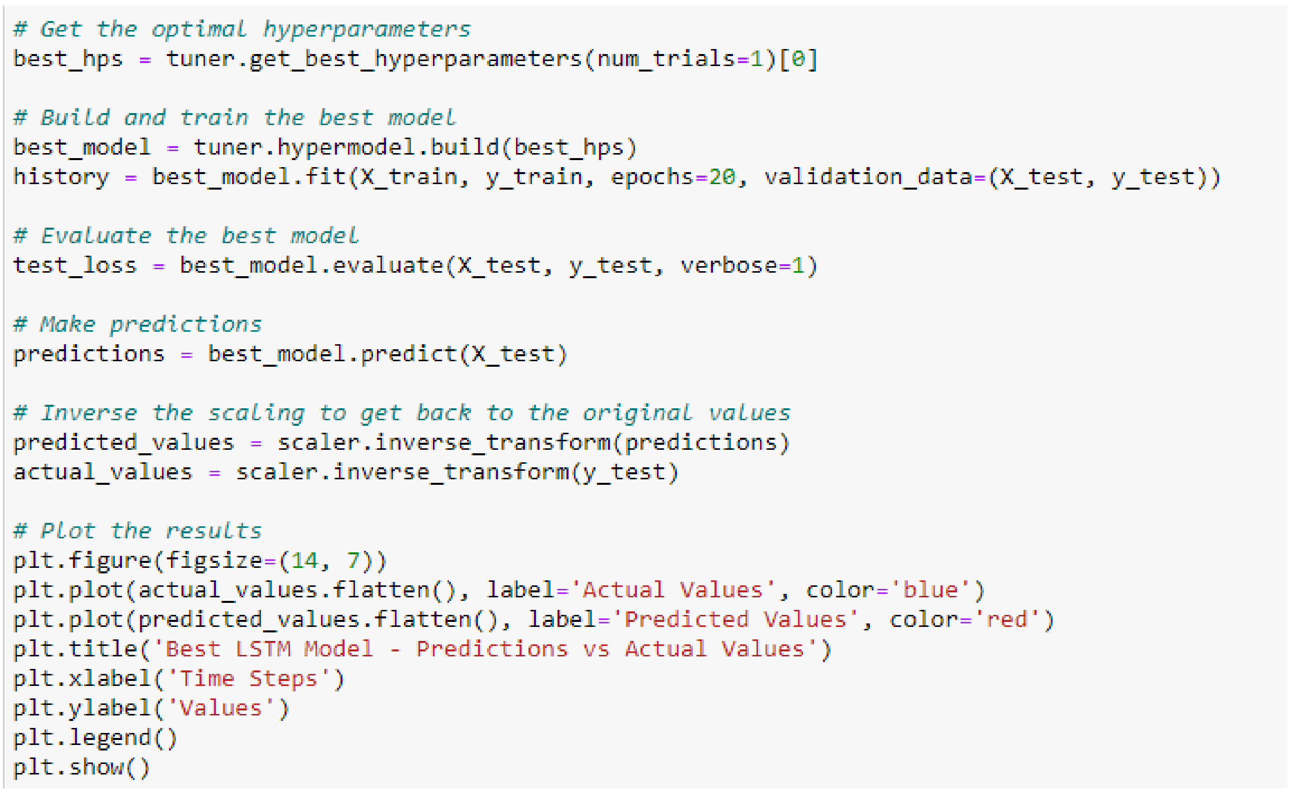
```
In [2]: #Checking for missing values
        data.isnull().sum()
        data = data.fillna(method='ffill')  # Forward fill for simplicity

        #Convert the Date column to datetime
        data['Date'] = pd.to_datetime(data['Date'])
        data.set_index('Date', inplace=True)

        from sklearn.preprocessing import MinMaxScaler

        #Initializing the scaler
        scaler = MinMaxScaler()

        #Fitting and transforming the data
        scaled_data = scaler.fit_transform(data)

        #Converting the scaled data back into a DataFrame
        scaled_data = pd.DataFrame(scaled_data, index=data.index, columns=data.columns)

        # To display the first few rows of the scaled data
        scaled_data.head()
```



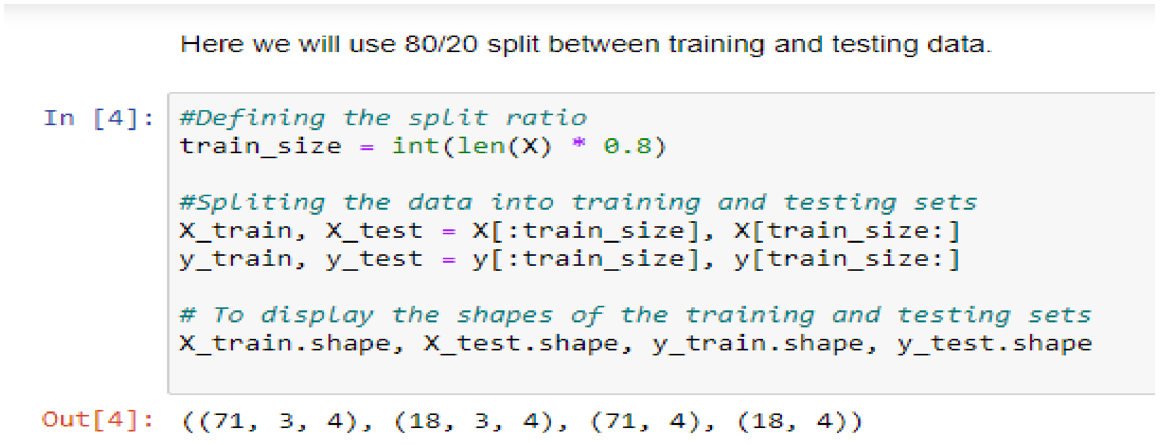
```
: import numpy as np

  #Defining the look_back period
  look_back = 3

  #Creating the sequences
  def create_sequences(data, look_back):
      X, y = [], []
      for i in range(len(data) - look_back):
          X.append(data[i:i + look_back])
          y.append(data[i + look_back])
      return np.array(X), np.array(y)

  # To create sequences from the scaled data
  X, y = create_sequences(scaled_data.values, look_back)

  #To Display the shape of the resulting arrays
  X.shape, y.shape

: ((89, 3, 4), (89, 4))
```



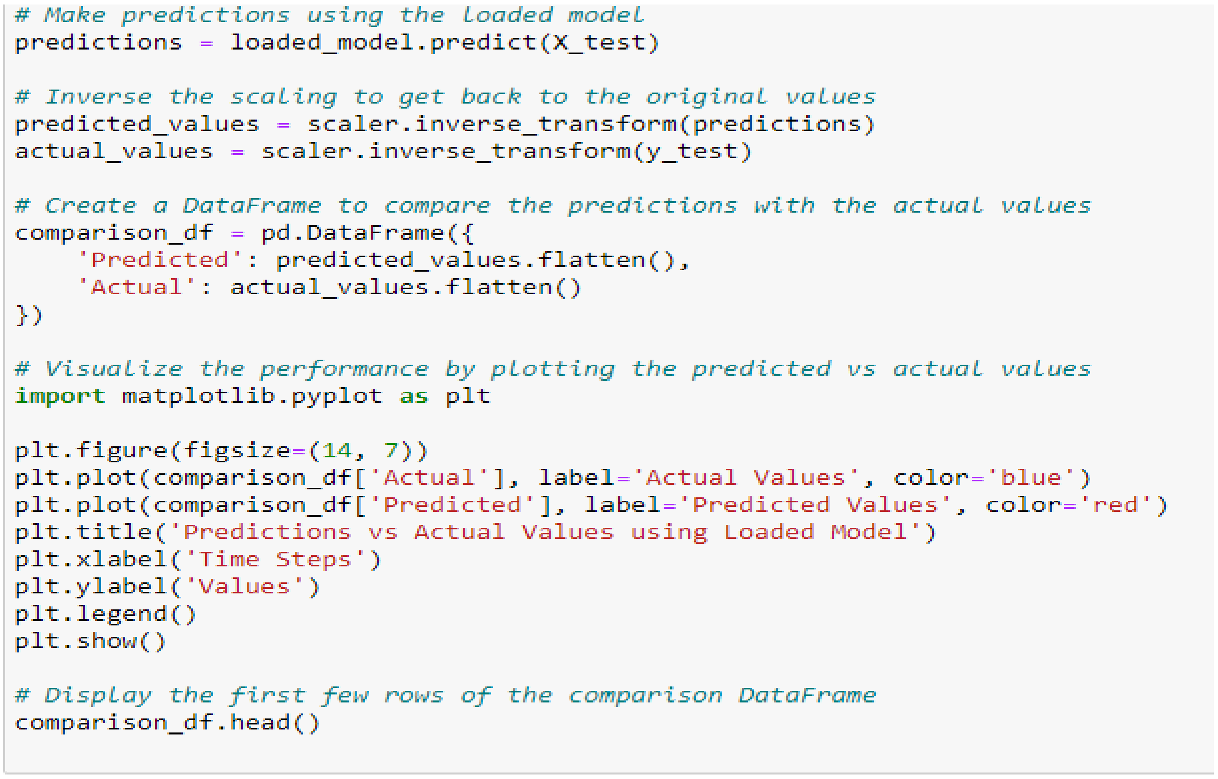
```python
import keras_tuner as kt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam

# Define the model-building function
def build_model(hp):
    model = Sequential()

    # LSTM layer 1
    model.add(LSTM(units=hp.Int('units', min_value=32, max_value=128, step=16),
                   return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
    model.add(Dropout(rate=hp.Float('dropout_1', min_value=0.1, max_value=0.5, step=0.1)))

    # LSTM layer 2
    model.add(LSTM(units=hp.Int('units_2', min_value=32, max_value=128, step=16)))
    model.add(Dropout(rate=hp.Float('dropout_2', min_value=0.1, max_value=0.5, step=0.1)))

    # Dense output layer
    model.add(Dense(4))

    # Compile model
    model.compile(optimizer=Adam(learning_rate=hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])),
                  loss='mean_squared_error')

    return model

# Instantiate a tuner
tuner = kt.RandomSearch(build_model,
```

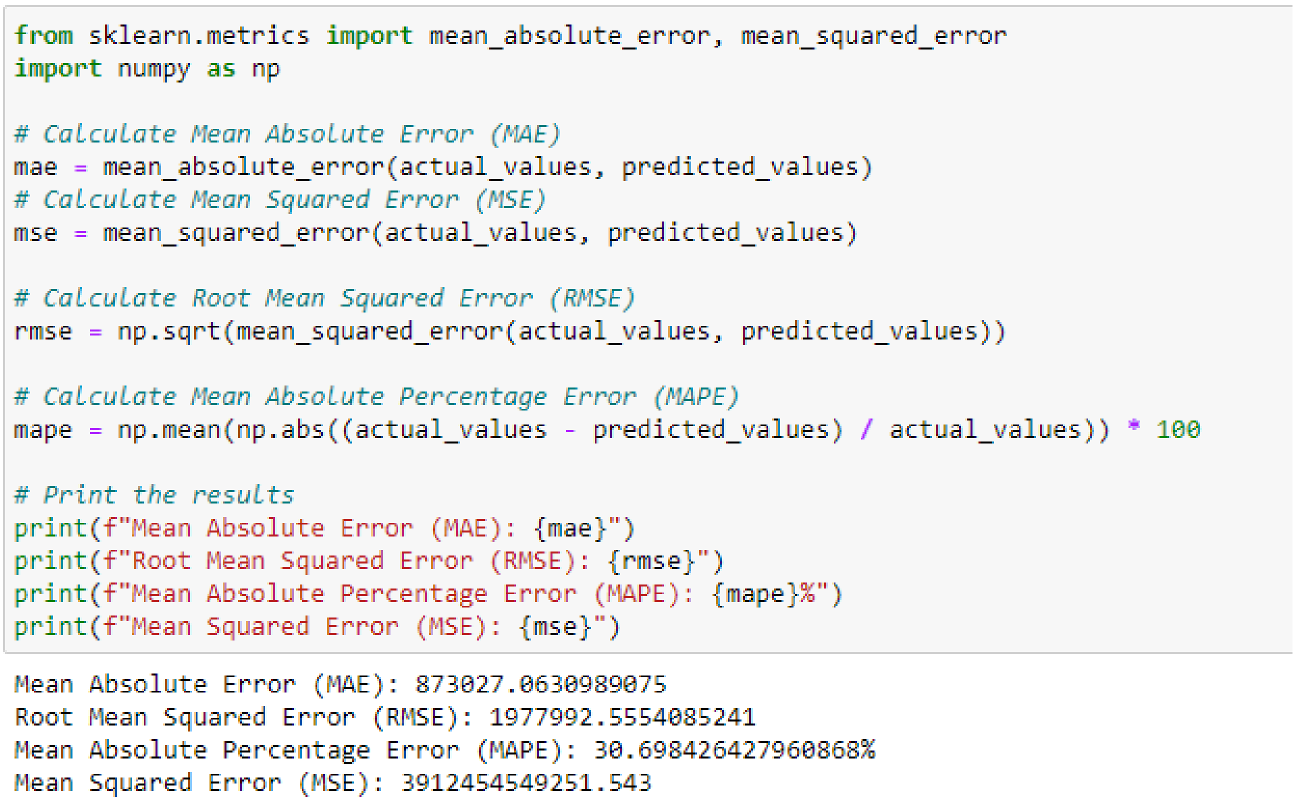


```python
# Get the optimal hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]

# Build and train the best model
best_model = tuner.hypermodel.build(best_hps)
history = best_model.fit(X_train, y_train, epochs=20, validation_data=(X_test, y_test))

# Evaluate the best model
test_loss = best_model.evaluate(X_test, y_test, verbose=1)

# Make predictions
predictions = best_model.predict(X_test)

# Inverse the scaling to get back to the original values
predicted_values = scaler.inverse_transform(predictions)
actual_values = scaler.inverse_transform(y_test)

# Plot the results
plt.figure(figsize=(14, 7))
plt.plot(actual_values.flatten(), label='Actual Values', color='blue')
plt.plot(predicted_values.flatten(), label='Predicted Values', color='red')
plt.title('Best LSTM Model - Predictions vs Actual Values')
plt.xlabel('Time Steps')
plt.ylabel('Values')
plt.legend()
plt.show()
```


Here we will use 80/20 split between training and testing data.

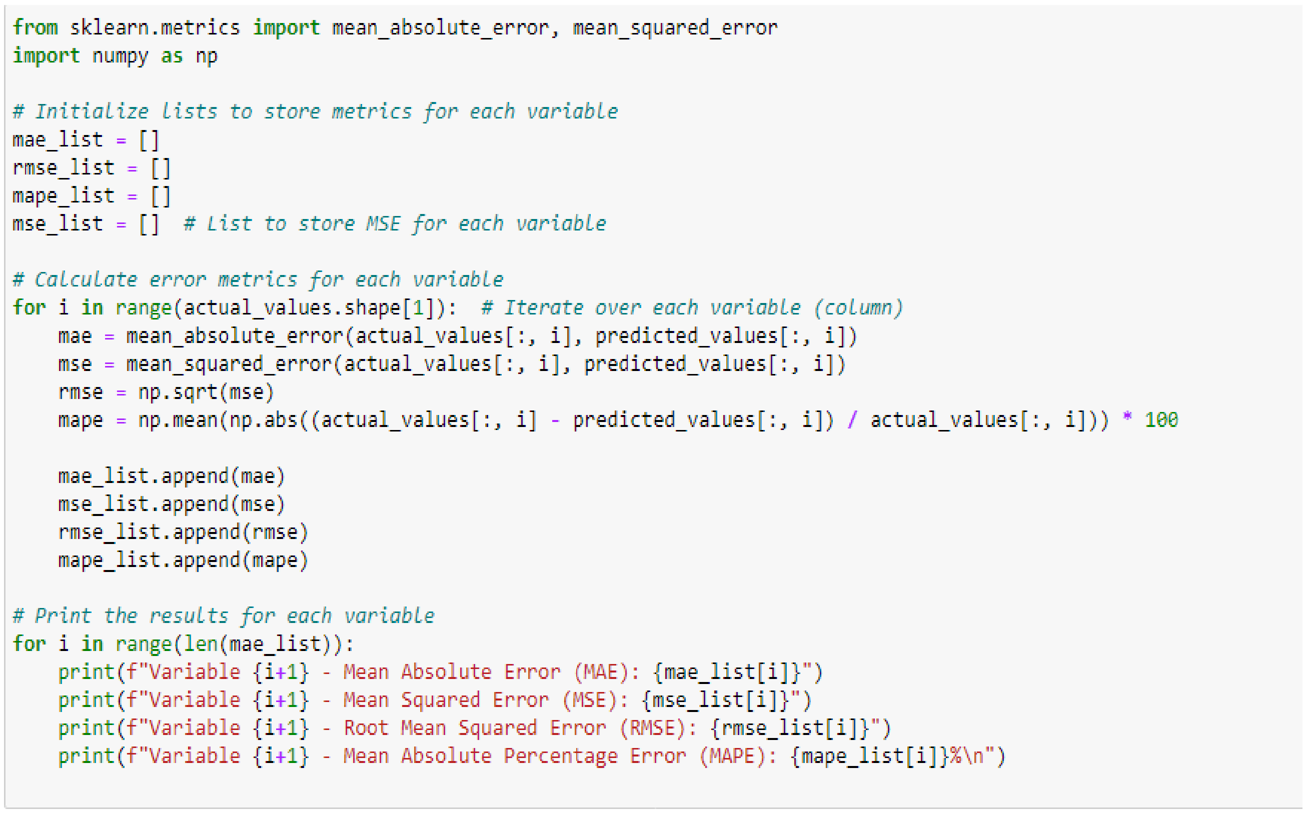
```
In [4]: #Defining the split ratio
        train_size = int(len(X) * 0.8)

        #Spliting the data into training and testing sets
        X_train, X_test = X[:train_size], X[train_size:]
        y_train, y_test = y[:train_size], y[train_size:]

        # To display the shapes of the training and testing sets
        X_train.shape, X_test.shape, y_train.shape, y_test.shape

Out[4]: ((71, 3, 4), (18, 3, 4), (71, 4), (18, 4))
```


```
# Make predictions using the loaded model
predictions = loaded_model.predict(X_test)

# Inverse the scaling to get back to the original values
predicted_values = scaler.inverse_transform(predictions)
actual_values = scaler.inverse_transform(y_test)

# Create a DataFrame to compare the predictions with the actual values
comparison_df = pd.DataFrame({
    'Predicted': predicted_values.flatten(),
    'Actual': actual_values.flatten()
})

# Visualize the performance by plotting the predicted vs actual values
import matplotlib.pyplot as plt

plt.figure(figsize=(14, 7))
plt.plot(comparison_df['Actual'], label='Actual Values', color='blue')
plt.plot(comparison_df['Predicted'], label='Predicted Values', color='red')
plt.title('Predictions vs Actual Values using Loaded Model')
plt.xlabel('Time Steps')
plt.ylabel('Values')
plt.legend()
plt.show()

# Display the first few rows of the comparison DataFrame
comparison_df.head()
```

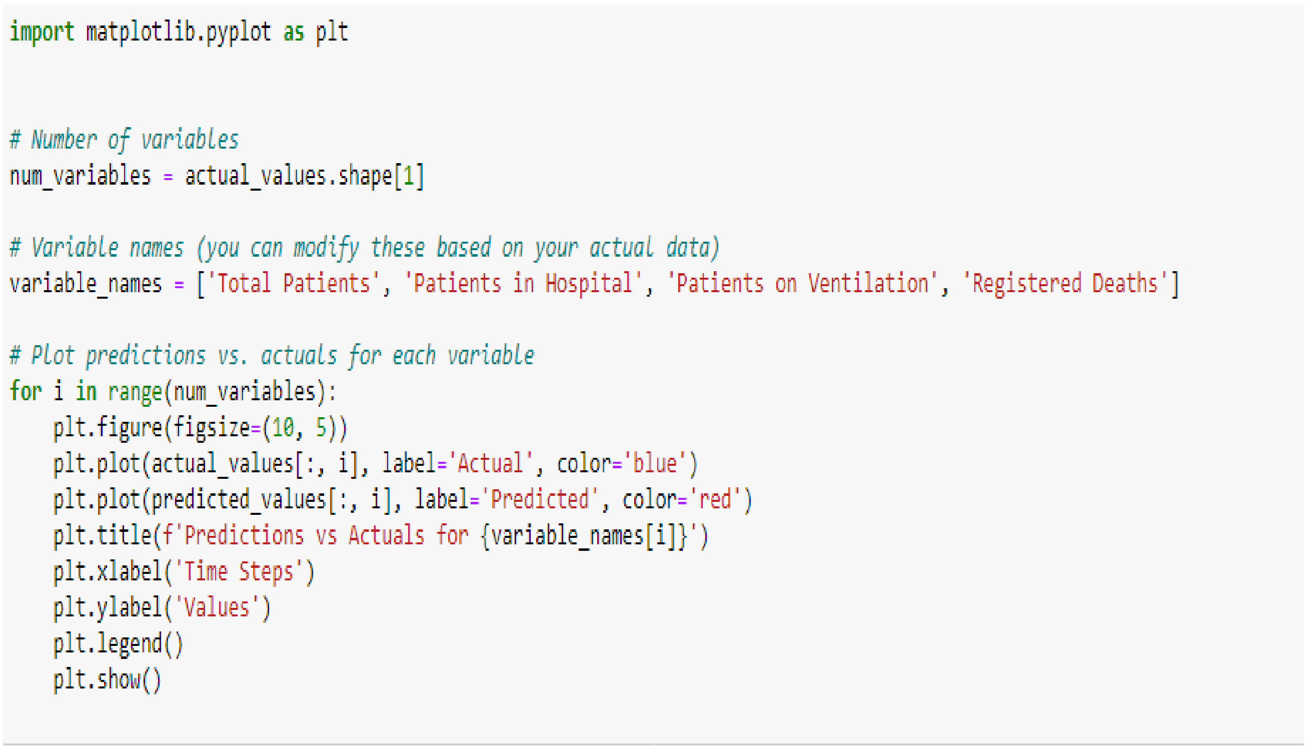


```python
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(actual_values, predicted_values)
# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(actual_values, predicted_values)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mean_squared_error(actual_values, predicted_values))

# Calculate Mean Absolute Percentage Error (MAPE)
mape = np.mean(np.abs((actual_values - predicted_values) / actual_values)) * 100

# Print the results
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Absolute Percentage Error (MAPE): {mape}%")
print(f"Mean Squared Error (MSE): {mse}")
```

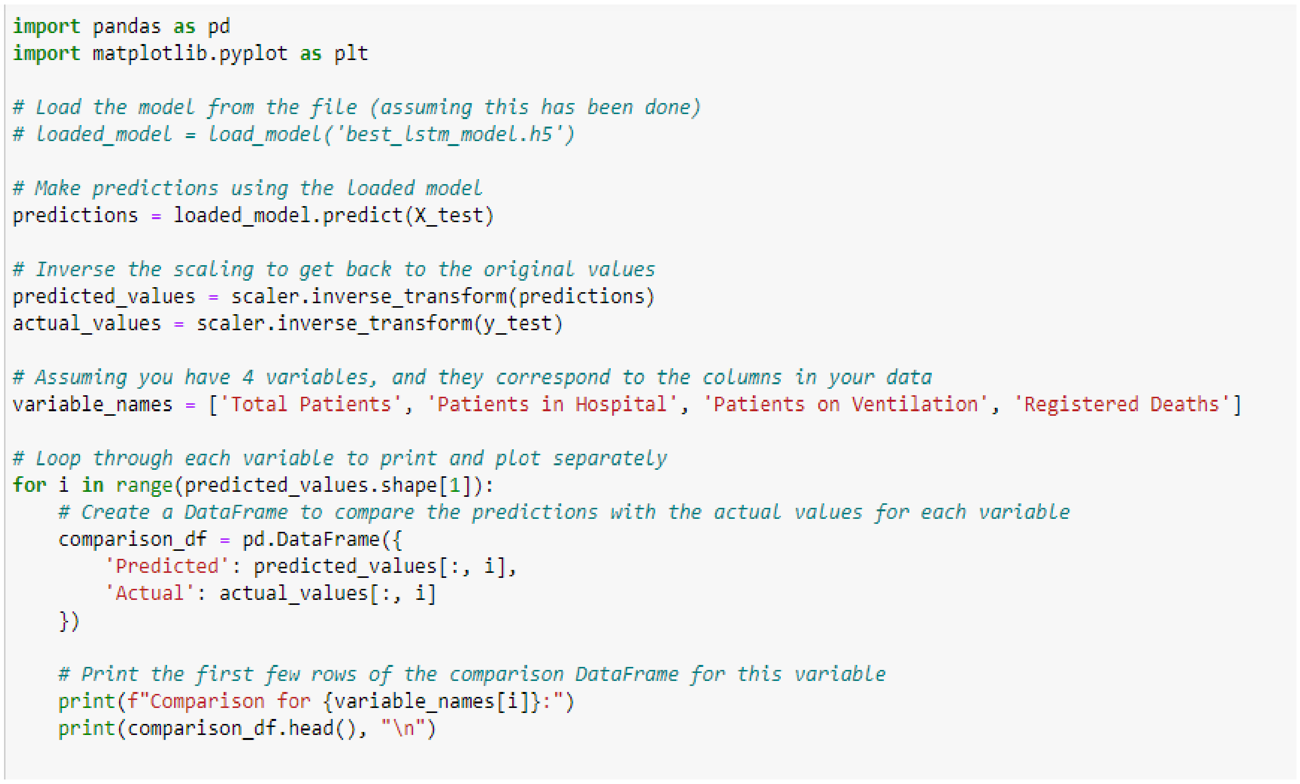
```
Mean Absolute Error (MAE): 873027.0630989075
Root Mean Squared Error (RMSE): 1977992.5554085241
Mean Absolute Percentage Error (MAPE): 30.698426427960868%
Mean Squared Error (MSE): 3912454549251.543
```



```python
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Initialize lists to store metrics for each variable
mae_list = []
rmse_list = []
mape_list = []
mse_list = []  # List to store MSE for each variable

# Calculate error metrics for each variable
for i in range(actual_values.shape[1]):  # Iterate over each variable (column)
    mae = mean_absolute_error(actual_values[:, i], predicted_values[:, i])
    mse = mean_squared_error(actual_values[:, i], predicted_values[:, i])
    rmse = np.sqrt(mse)
    mape = np.mean(np.abs((actual_values[:, i] - predicted_values[:, i]) / actual_values[:, i])) * 100

    mae_list.append(mae)
    mse_list.append(mse)
    rmse_list.append(rmse)
    mape_list.append(mape)

# Print the results for each variable
for i in range(len(mae_list)):
    print(f"Variable {i+1} - Mean Absolute Error (MAE): {mae_list[i]}")
    print(f"Variable {i+1} - Mean Squared Error (MSE): {mse_list[i]}")
    print(f"Variable {i+1} - Root Mean Squared Error (RMSE): {rmse_list[i]}")
    print(f"Variable {i+1} - Mean Absolute Percentage Error (MAPE): {mape_list[i]}%\n")
```



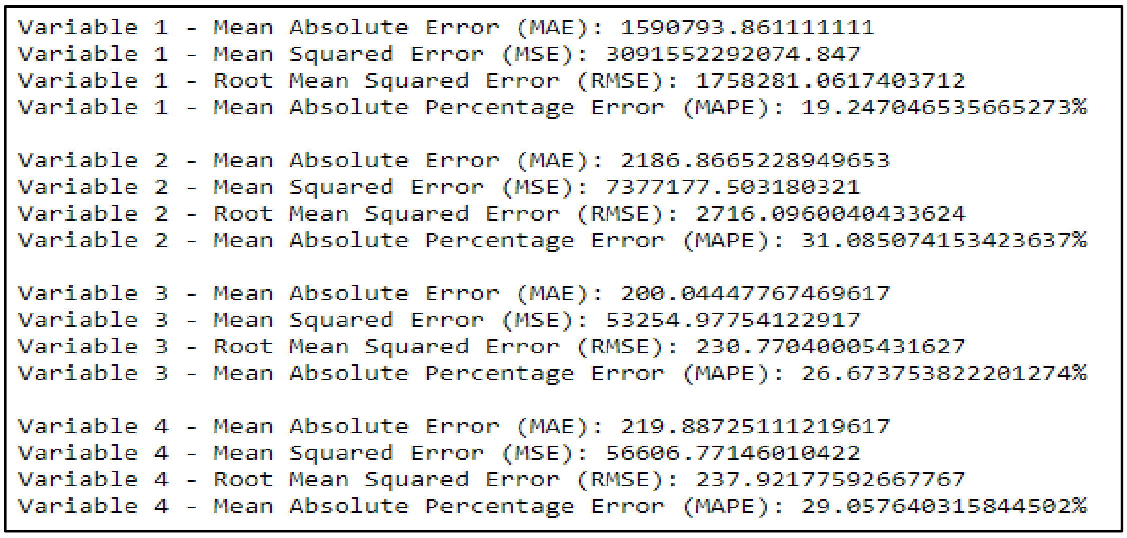
```python
import matplotlib.pyplot as plt


# Number of variables
num_variables = actual_values.shape[1]

# Variable names (you can modify these based on your actual data)
variable_names = ['Total Patients', 'Patients in Hospital', 'Patients on Ventilation', 'Registered Deaths']

# Plot predictions vs. actuals for each variable
for i in range(num_variables):
    plt.figure(figsize=(10, 5))
    plt.plot(actual_values[:, i], label='Actual', color='blue')
    plt.plot(predicted_values[:, i], label='Predicted', color='red')
    plt.title(f'Predictions vs Actuals for {variable_names[i]}')
    plt.xlabel('Time Steps')
    plt.ylabel('Values')
    plt.legend()
    plt.show()
```



```python
import pandas as pd
import matplotlib.pyplot as plt

# Load the model from the file (assuming this has been done)
# loaded_model = load_model('best_lstm_model.h5')

# Make predictions using the loaded model
predictions = loaded_model.predict(X_test)

# Inverse the scaling to get back to the original values
predicted_values = scaler.inverse_transform(predictions)
actual_values = scaler.inverse_transform(y_test)

# Assuming you have 4 variables, and they correspond to the columns in your data
variable_names = ['Total Patients', 'Patients in Hospital', 'Patients on Ventilation', 'Registered Deaths']

# Loop through each variable to print and plot separately
for i in range(predicted_values.shape[1]):
    # Create a DataFrame to compare the predictions with the actual values for each variable
    comparison_df = pd.DataFrame({
        'Predicted': predicted_values[:, i],
        'Actual': actual_values[:, i]
    })

    # Print the first few rows of the comparison DataFrame for this variable
    print(f"Comparison for {variable_names[i]}:")
    print(comparison_df.head(), "\n")
```

Error Metrics Evaluation:

Following is the screenshot of the error metrics for all the variables:

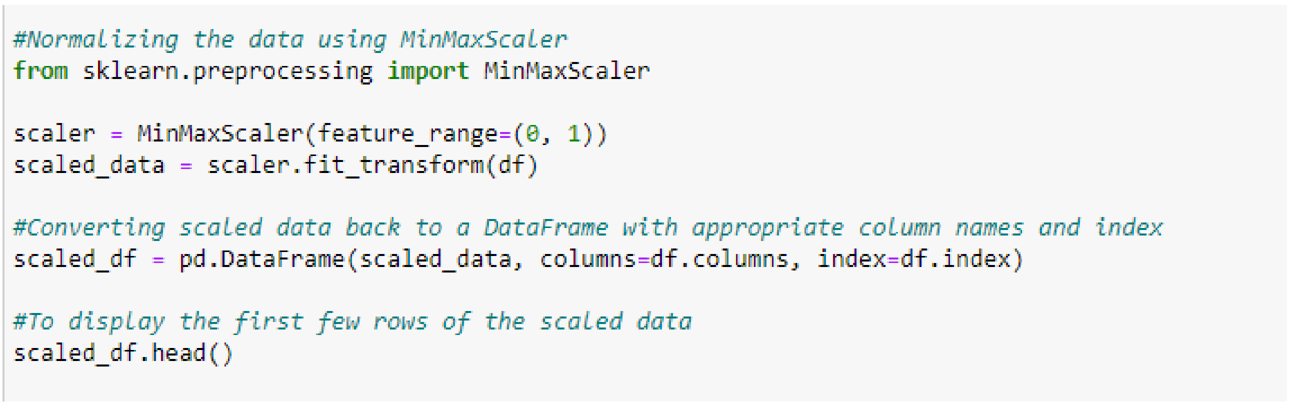


```
Variable 1 - Mean Absolute Error (MAE): 1590793.861111111
Variable 1 - Mean Squared Error (MSE): 3091552292074.847
Variable 1 - Root Mean Squared Error (RMSE): 1758281.0617403712
Variable 1 - Mean Absolute Percentage Error (MAPE): 19.247046535665273%

Variable 2 - Mean Absolute Error (MAE): 2186.8665228949653
Variable 2 - Mean Squared Error (MSE): 7377177.503180321
Variable 2 - Root Mean Squared Error (RMSE): 2716.0960040433624
Variable 2 - Mean Absolute Percentage Error (MAPE): 31.085074153423637%

Variable 3 - Mean Absolute Error (MAE): 200.04447767469617
Variable 3 - Mean Squared Error (MSE): 53254.97754122917
Variable 3 - Root Mean Squared Error (RMSE): 230.77040005431627
Variable 3 - Mean Absolute Percentage Error (MAPE): 26.673753822201274%

Variable 4 - Mean Absolute Error (MAE): 219.88725111219617
Variable 4 - Mean Squared Error (MSE): 56606.77146010422
Variable 4 - Root Mean Squared Error (RMSE): 237.92177592667767
Variable 4 - Mean Absolute Percentage Error (MAPE): 29.057640315844502%
```

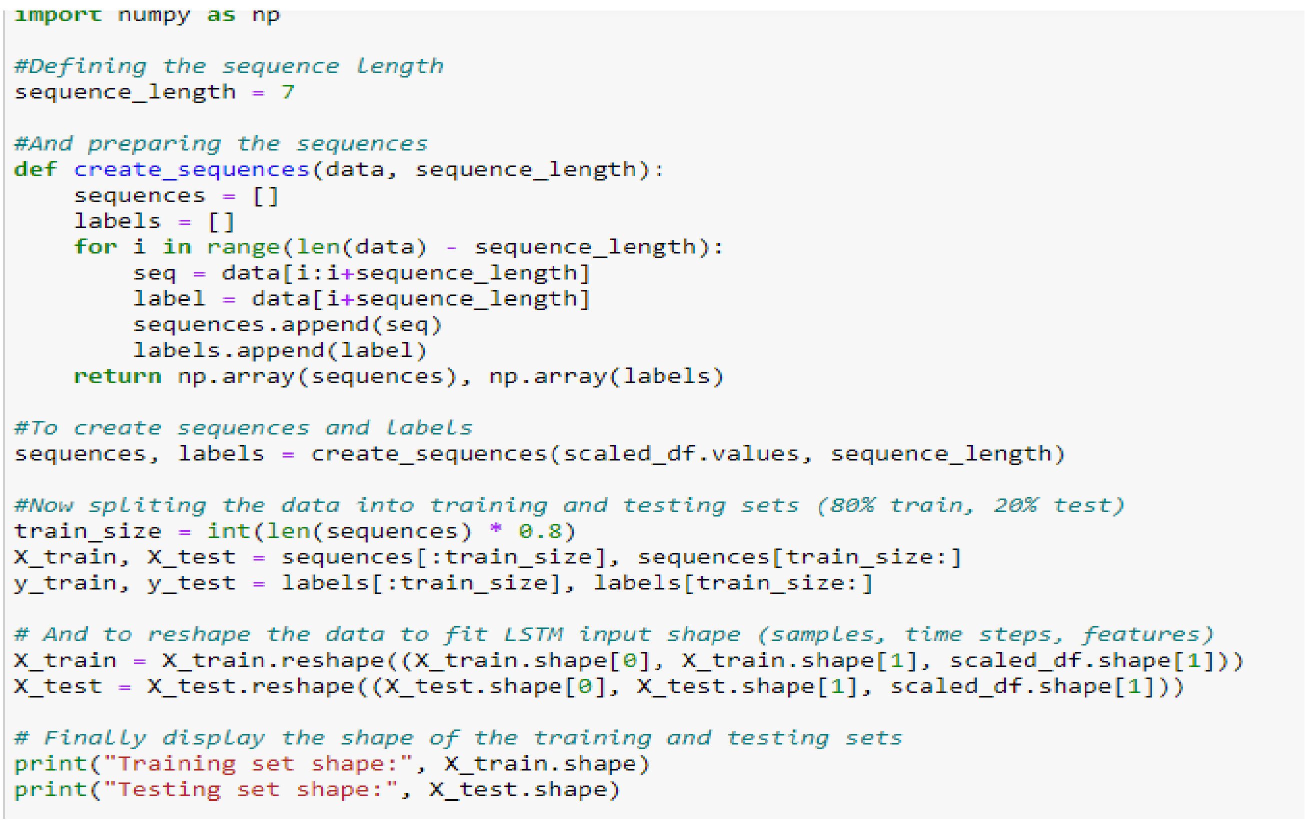
Figure 39 Error Metrics for all the Variables for LSTM with LLM method.

From the error metrics, the model shows varying performance across variables, with the highest errors for the total number of patients and the lowest for registered deaths. This suggests that certain pandemic trends are more complex and challenging for the model to capture accurately.

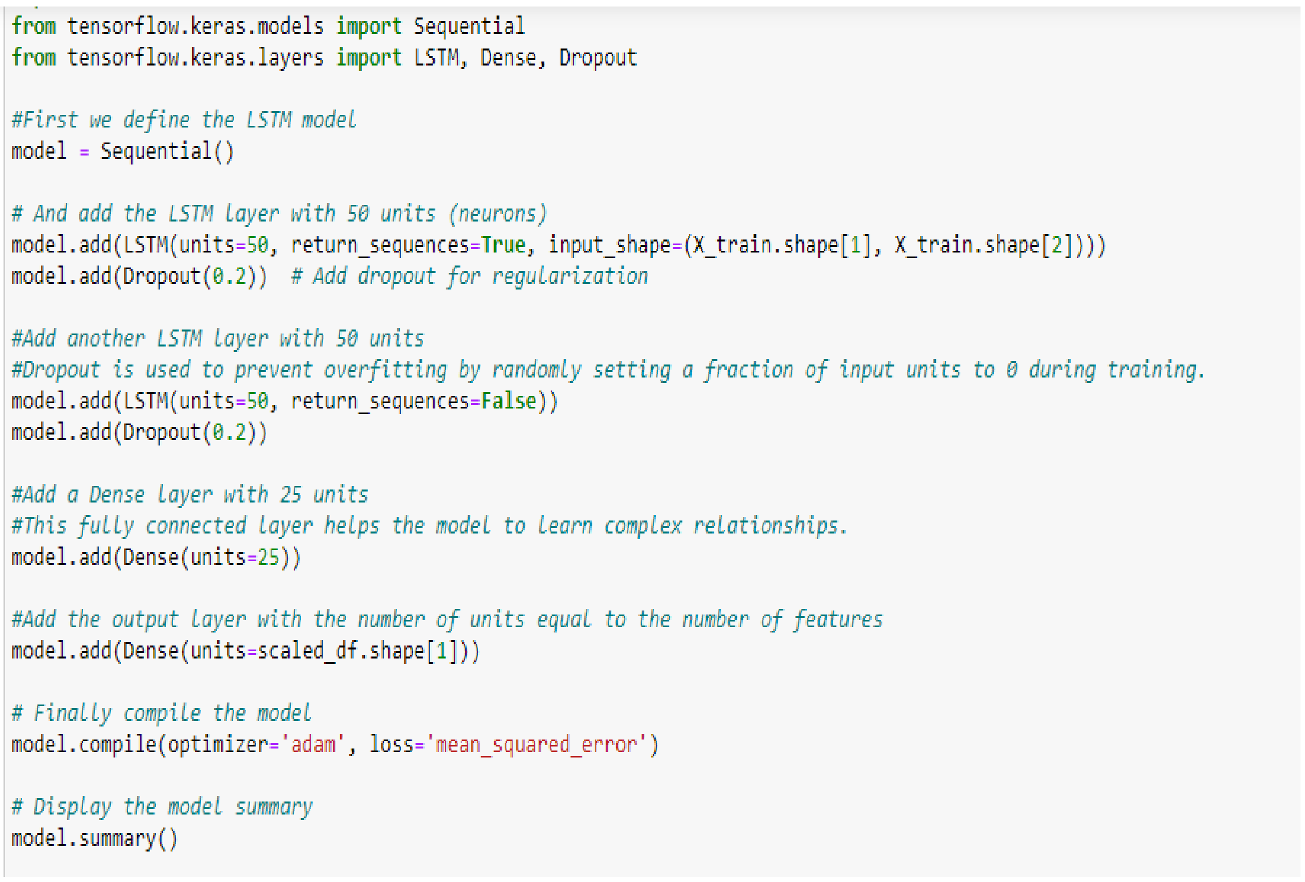
E. LSTM without LLM: GRU Model code snippets



```python
#Normalizing the data using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(df)

#Converting scaled data back to a DataFrame with appropriate column names and index
scaled_df = pd.DataFrame(scaled_data, columns=df.columns, index=df.index)

#To display the first few rows of the scaled data
scaled_df.head()
```



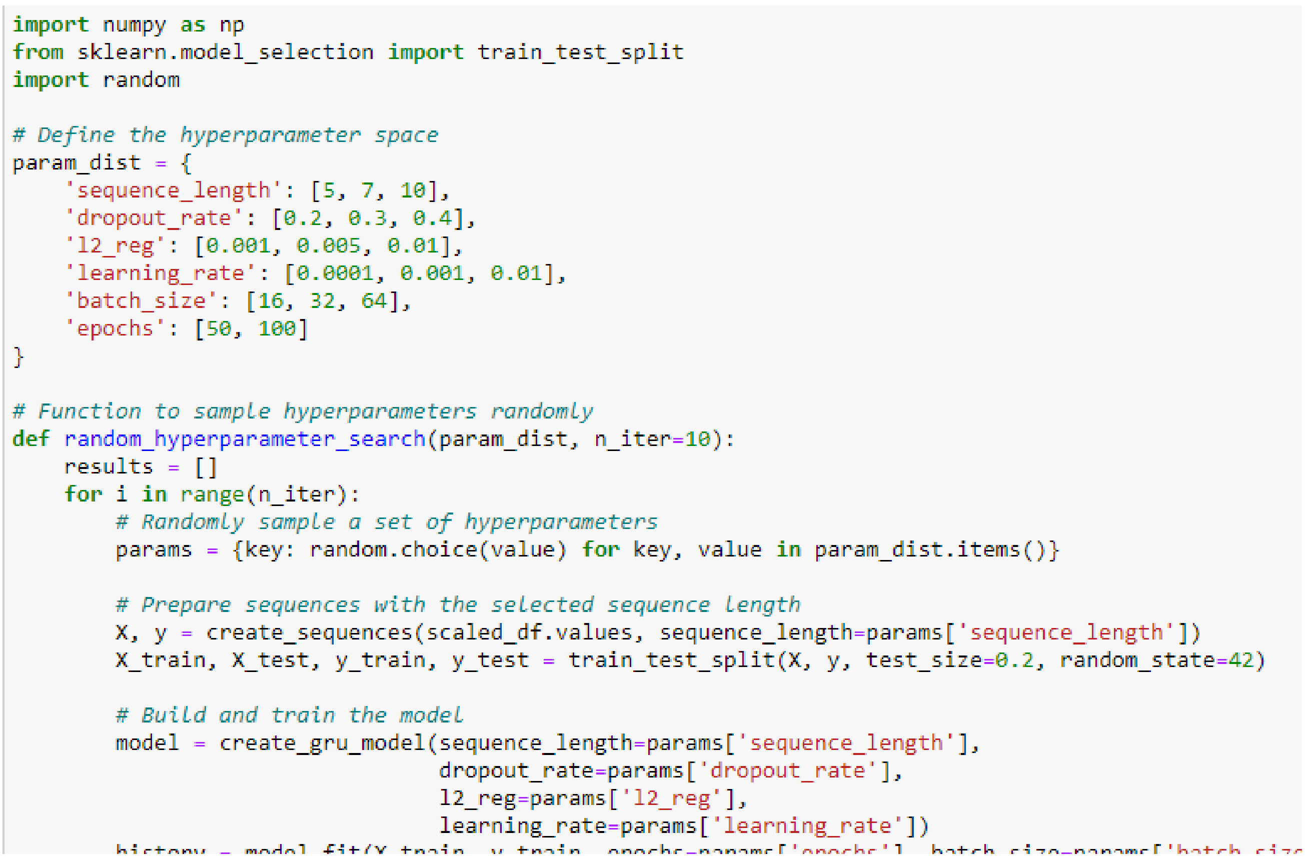
```python
import numpy as np

#Defining the sequence length
sequence_length = 7

#And preparing the sequences
def create_sequences(data, sequence_length):
    sequences = []
    labels = []
    for i in range(len(data) - sequence_length):
        seq = data[i:i+sequence_length]
        label = data[i+sequence_length]
        sequences.append(seq)
        labels.append(label)
    return np.array(sequences), np.array(labels)

#To create sequences and labels
sequences, labels = create_sequences(scaled_df.values, sequence_length)

#Now spliting the data into training and testing sets (80% train, 20% test)
train_size = int(len(sequences) * 0.8)
X_train, X_test = sequences[:train_size], sequences[train_size:]
y_train, y_test = labels[:train_size], labels[train_size:]

# And to reshape the data to fit LSTM input shape (samples, time steps, features)
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], scaled_df.shape[1]))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], scaled_df.shape[1]))

# Finally display the shape of the training and testing sets
print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
```



```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

#First we define the LSTM model
model = Sequential()

# And add the LSTM layer with 50 units (neurons)
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))  # Add dropout for regularization

#Add another LSTM layer with 50 units
#Dropout is used to prevent overfitting by randomly setting a fraction of input units to 0 during training.
model.add(LSTM(units=50, return_sequences=False))
model.add(Dropout(0.2))

#Add a Dense layer with 25 units
#This fully connected layer helps the model to learn complex relationships.
model.add(Dense(units=25))

#Add the output layer with the number of units equal to the number of features
model.add(Dense(units=scaled_df.shape[1]))

# Finally compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Display the model summary
model.summary()
```

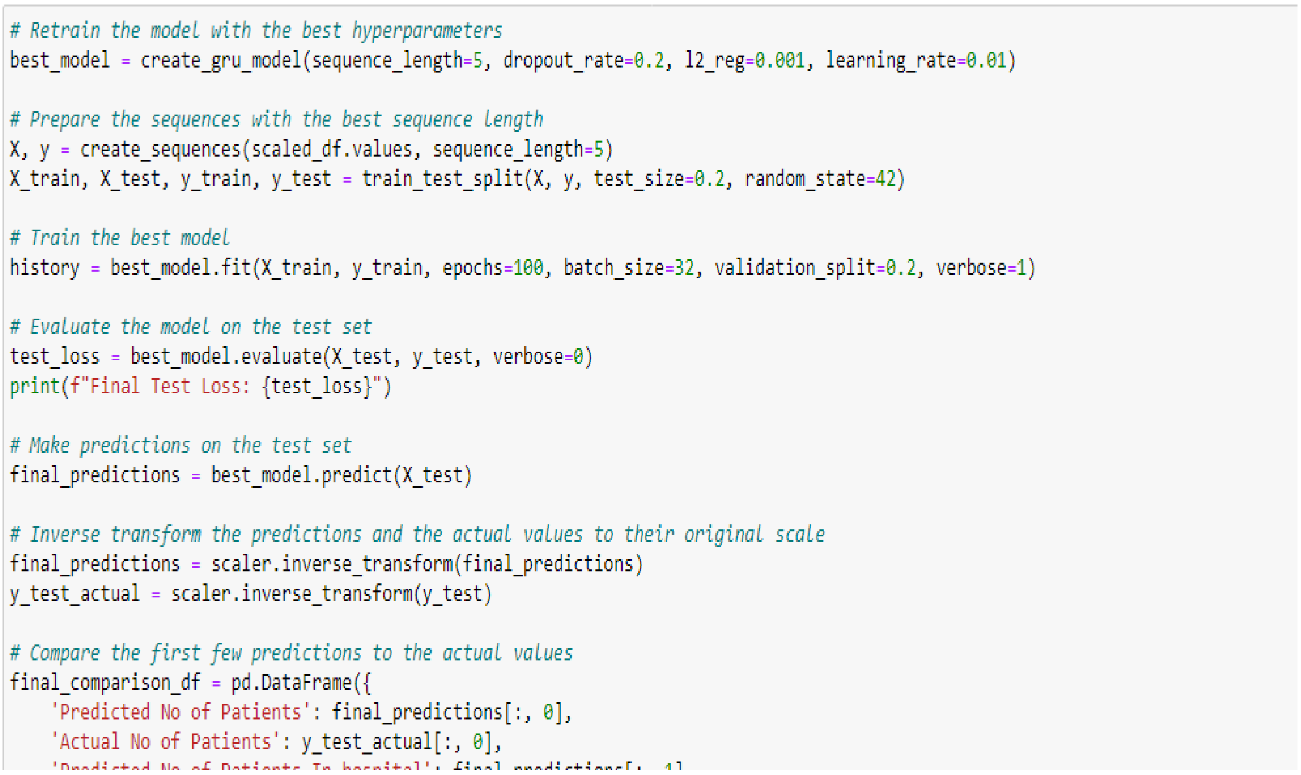
```python
from tensorflow.keras.layers import GRU

def create_and_train_gru_model(sequence_length, batch_size, learning_rate):
    #First prepare the sequences
    X, y = create_sequences(scaled_df.values, sequence_length)

    # Then split the data into training and testing sets
    train_size = int(len(X) * 0.8)
    X_train, X_test = X[:train_size], X[train_size:]
    y_train, y_test = y[:train_size], y[train_size:]

    # Reshape the data to fit GRU input shape
    X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], scaled_df.shape[1]))
    X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], scaled_df.shape[1]))

    # Here we define the GRU model
    model = Sequential()
    model.add(GRU(units=50, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
    model.add(Dropout(0.2))
    model.add(GRU(units=50, return_sequences=False))
    model.add(Dropout(0.2))
    model.add(Dense(units=25))
    model.add(Dense(units=scaled_df.shape[1]))

    # Compile the model with the specified learning rate
    optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer, loss='mean_squared_error')
```

```python
# Train the GRU model with the best hyperparameters found earlier
gru_model, gru_history = create_and_train_gru_model(sequence_length=5, batch_size=32, learning_rate=0.001)

# Make predictions on the test set
gru_predictions = gru_model.predict(X_test)

# Inverse transform the predictions and the actual values to their original scale
gru_predictions = scaler.inverse_transform(gru_predictions)
y_test_actual = scaler.inverse_transform(y_test)

# Compare the first few predictions to the actual values to see the difference
gru_comparison_df = pd.DataFrame({
    'Predicted No of Patients': gru_predictions[:, 0],
    'Actual No of Patients': y_test_actual[:, 0],
    'Predicted No of Patients In hospital': gru_predictions[:, 1],
    'Actual No of Patients In hospital': y_test_actual[:, 1],
    'Predicted In mechanical ventillation bed': gru_predictions[:, 2],
    'Actual In mechanical ventillation bed': y_test_actual[:, 2],
    'Predicted Registered deaths due to covid': gru_predictions[:, 3],
    'Actual Registered deaths due to covid': y_test_actual[:, 3]
})

# Display the comparison
gru_comparison_df.head()
```

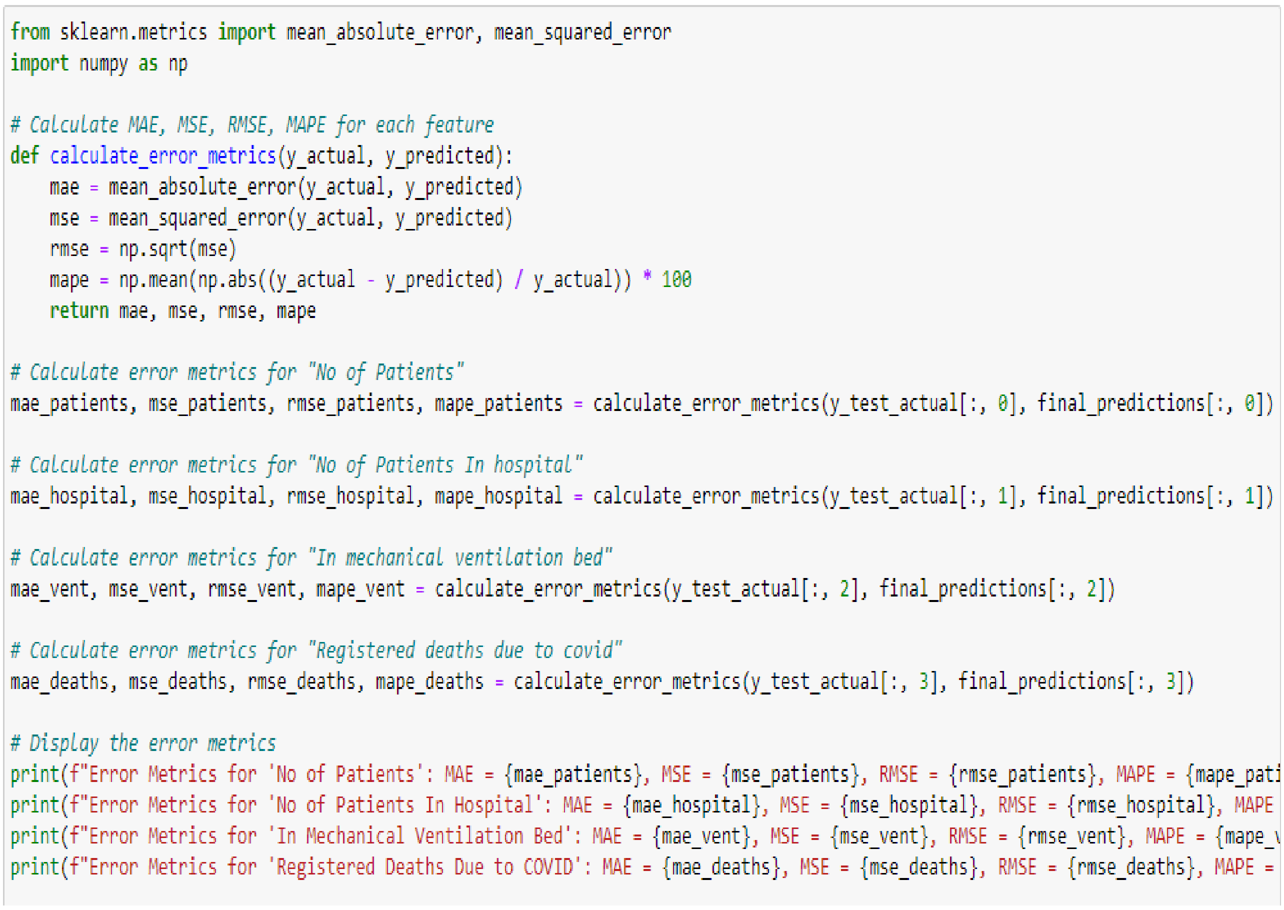


```python
import numpy as np
from sklearn.model_selection import train_test_split
import random

# Define the hyperparameter space
param_dist = {
    'sequence_length': [5, 7, 10],
    'dropout_rate': [0.2, 0.3, 0.4],
    'l2_reg': [0.001, 0.005, 0.01],
    'learning_rate': [0.0001, 0.001, 0.01],
    'batch_size': [16, 32, 64],
    'epochs': [50, 100]
}

# Function to sample hyperparameters randomly
def random_hyperparameter_search(param_dist, n_iter=10):
    results = []
    for i in range(n_iter):
        # Randomly sample a set of hyperparameters
        params = {key: random.choice(value) for key, value in param_dist.items()}

        # Prepare sequences with the selected sequence length
        X, y = create_sequences(scaled_df.values, sequence_length=params['sequence_length'])
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

        # Build and train the model
        model = create_gru_model(sequence_length=params['sequence_length'],
                                 dropout_rate=params['dropout_rate'],
                                 l2_reg=params['l2_reg'],
                                 learning_rate=params['learning_rate'])
        history = model.fit(X_train, y_train, epochs=params['epochs'], batch_size=params['batch_siz
```

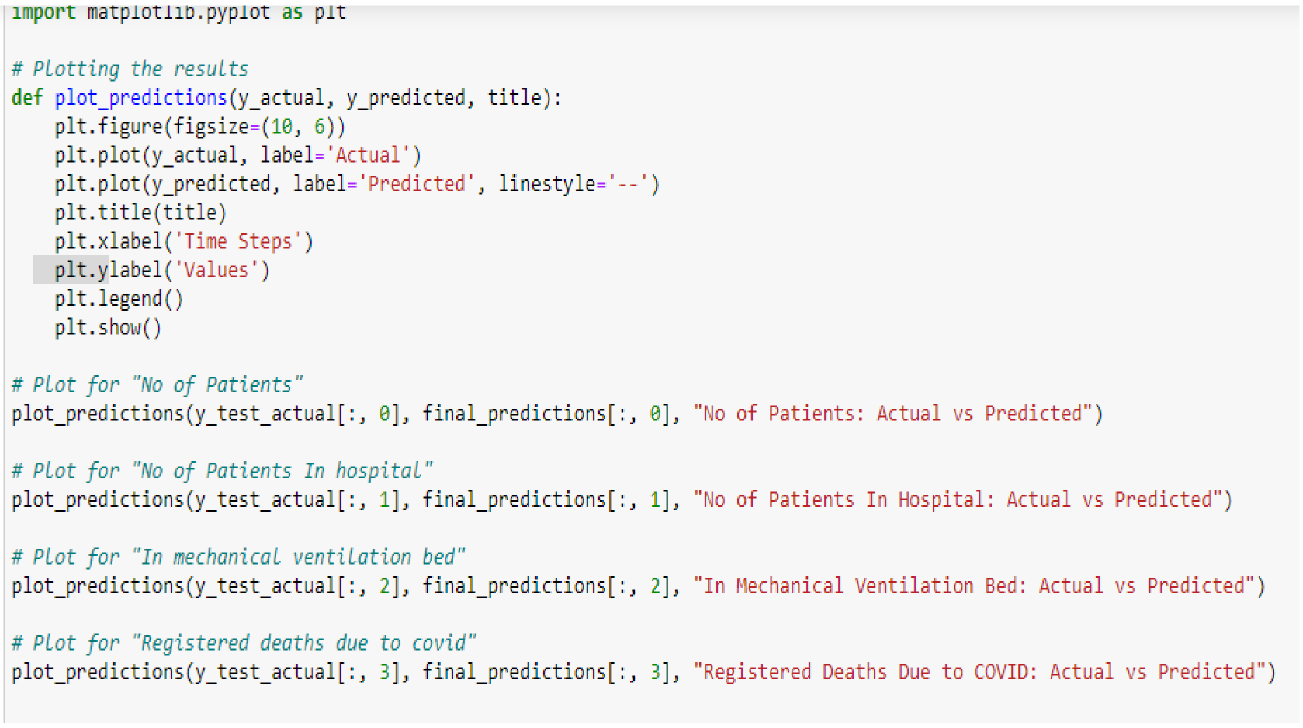
Sequence Length: 5 Dropout Rate: 0.2 L2 Regularization: 0.001 Learning Rate: 0.01 Batch Size: 32 Epochs: 100 Now we can proceed to retrain the model using these settings and then evaluate its performance on the test set.



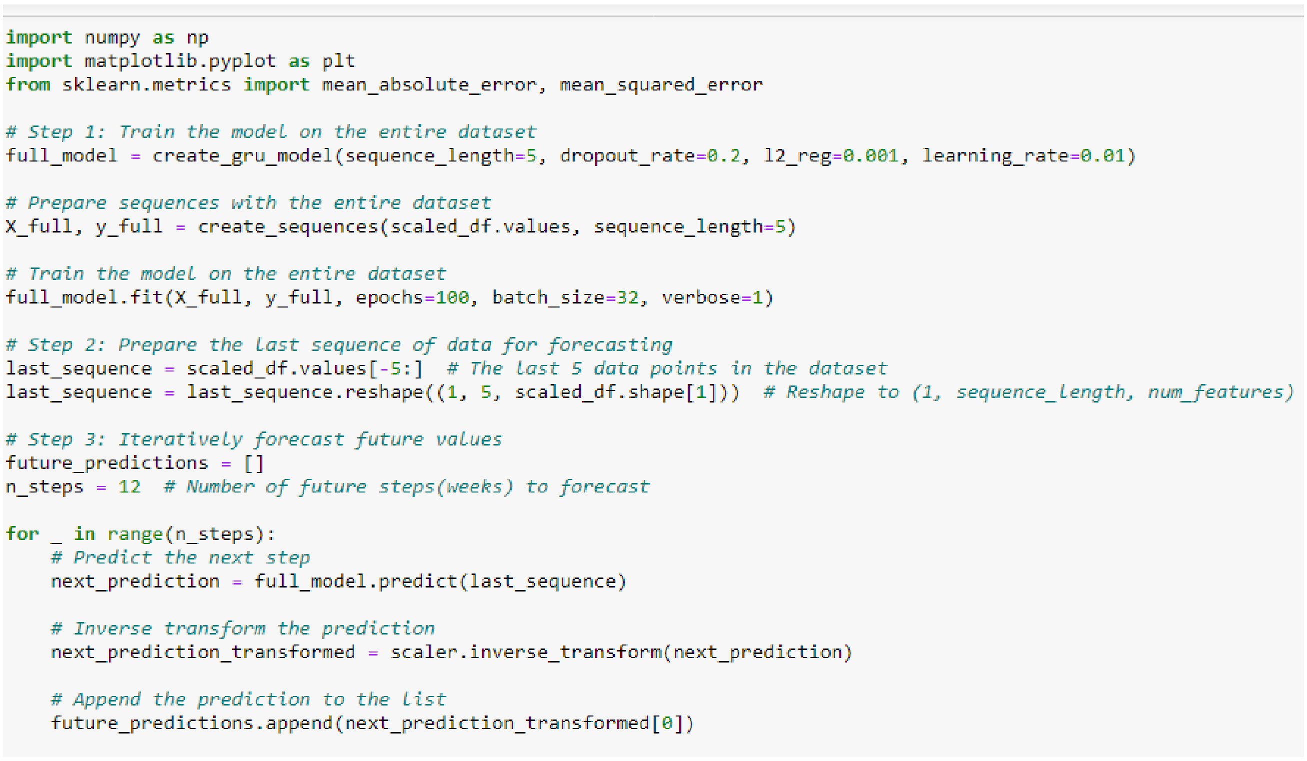
```python
# Retrain the model with the best hyperparameters
best_model = create_gru_model(sequence_length=5, dropout_rate=0.2, l2_reg=0.001, learning_rate=0.01)

# Prepare the sequences with the best sequence length
X, y = create_sequences(scaled_df.values, sequence_length=5)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the best model
history = best_model.fit(X_train, y_train, epochs=100, batch_size=32, validation_split=0.2, verbose=1)

# Evaluate the model on the test set
test_loss = best_model.evaluate(X_test, y_test, verbose=0)
print(f"Final Test Loss: {test_loss}")

# Make predictions on the test set
final_predictions = best_model.predict(X_test)

# Inverse transform the predictions and the actual values to their original scale
final_predictions = scaler.inverse_transform(final_predictions)
y_test_actual = scaler.inverse_transform(y_test)

# Compare the first few predictions to the actual values
final_comparison_df = pd.DataFrame({
    'Predicted No of Patients': final_predictions[:, 0],
    'Actual No of Patients': y_test_actual[:, 0],
    'Predicted No of Patients In hospital': final_predictions[:, 1]
```



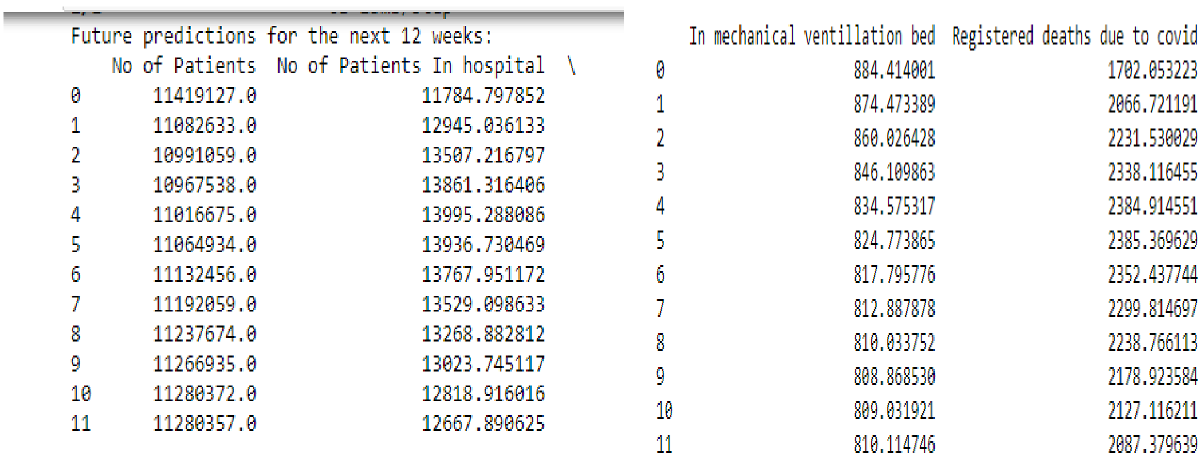
```python
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Calculate MAE, MSE, RMSE, MAPE for each feature
def calculate_error_metrics(y_actual, y_predicted):
    mae = mean_absolute_error(y_actual, y_predicted)
    mse = mean_squared_error(y_actual, y_predicted)
    rmse = np.sqrt(mse)
    mape = np.mean(np.abs((y_actual - y_predicted) / y_actual)) * 100
    return mae, mse, rmse, mape

# Calculate error metrics for "No of Patients"
mae_patients, mse_patients, rmse_patients, mape_patients = calculate_error_metrics(y_test_actual[:, 0], final_predictions[:, 0])

# Calculate error metrics for "No of Patients In hospital"
mae_hospital, mse_hospital, rmse_hospital, mape_hospital = calculate_error_metrics(y_test_actual[:, 1], final_predictions[:, 1])

# Calculate error metrics for "In mechanical ventilation bed"
mae_vent, mse_vent, rmse_vent, mape_vent = calculate_error_metrics(y_test_actual[:, 2], final_predictions[:, 2])

# Calculate error metrics for "Registered deaths due to covid"
mae_deaths, mse_deaths, rmse_deaths, mape_deaths = calculate_error_metrics(y_test_actual[:, 3], final_predictions[:, 3])

# Display the error metrics
print(f"Error Metrics for 'No of Patients': MAE = {mae_patients}, MSE = {mse_patients}, RMSE = {rmse_patients}, MAPE = {mape_pati
print(f"Error Metrics for 'No of Patients In Hospital': MAE = {mae_hospital}, MSE = {mse_hospital}, RMSE = {rmse_hospital}, MAPE
print(f"Error Metrics for 'In Mechanical Ventilation Bed': MAE = {mae_vent}, MSE = {mse_vent}, RMSE = {rmse_vent}, MAPE = {mape_v
print(f"Error Metrics for 'Registered Deaths Due to COVID': MAE = {mae_deaths}, MSE = {mse_deaths}, RMSE = {rmse_deaths}, MAPE =
```



```python
import matplotlib.pyplot as plt

# Plotting the results
def plot_predictions(y_actual, y_predicted, title):
    plt.figure(figsize=(10, 6))
    plt.plot(y_actual, label='Actual')
    plt.plot(y_predicted, label='Predicted', linestyle='--')
    plt.title(title)
    plt.xlabel('Time Steps')
    plt.ylabel('Values')
    plt.legend()
    plt.show()

# Plot for "No of Patients"
plot_predictions(y_test_actual[:, 0], final_predictions[:, 0], "No of Patients: Actual vs Predicted")

# Plot for "No of Patients In hospital"
plot_predictions(y_test_actual[:, 1], final_predictions[:, 1], "No of Patients In Hospital: Actual vs Predicted")

# Plot for "In mechanical ventilation bed"
plot_predictions(y_test_actual[:, 2], final_predictions[:, 2], "In Mechanical Ventilation Bed: Actual vs Predicted")

# Plot for "Registered deaths due to covid"
plot_predictions(y_test_actual[:, 3], final_predictions[:, 3], "Registered Deaths Due to COVID: Actual vs Predicted")
```



```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Step 1: Train the model on the entire dataset
full_model = create_gru_model(sequence_length=5, dropout_rate=0.2, l2_reg=0.001, learning_rate=0.01)

# Prepare sequences with the entire dataset
X_full, y_full = create_sequences(scaled_df.values, sequence_length=5)

# Train the model on the entire dataset
full_model.fit(X_full, y_full, epochs=100, batch_size=32, verbose=1)

# Step 2: Prepare the last sequence of data for forecasting
last_sequence = scaled_df.values[-5:]  # The last 5 data points in the dataset
last_sequence = last_sequence.reshape((1, 5, scaled_df.shape[1]))  # Reshape to (1, sequence_length, num_features)

# Step 3: Iteratively forecast future values
future_predictions = []
n_steps = 12  # Number of future steps(weeks) to forecast

for _ in range(n_steps):
    # Predict the next step
    next_prediction = full_model.predict(last_sequence)

    # Inverse transform the prediction
    next_prediction_transformed = scaler.inverse_transform(next_prediction)

    # Append the prediction to the list
    future_predictions.append(next_prediction_transformed[0])
```

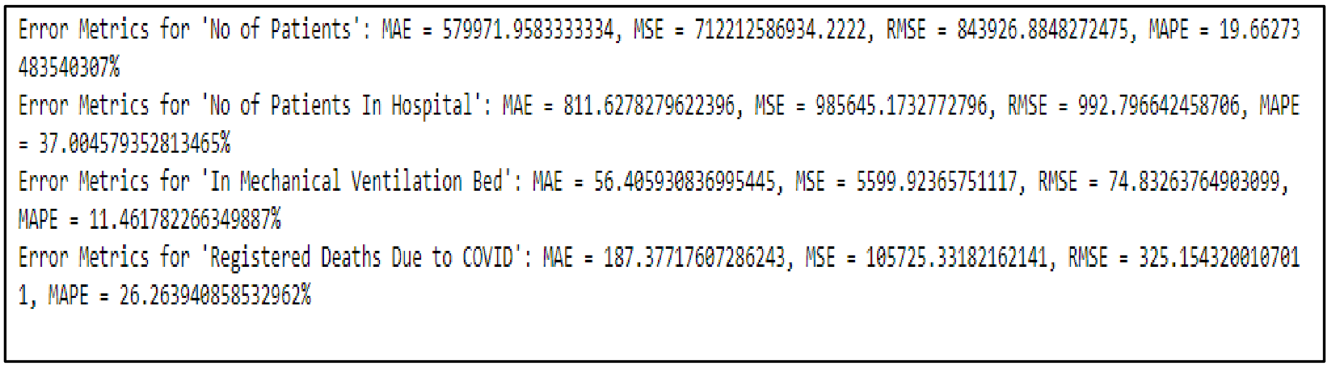


```
Future predictions for the next 12 weeks:
    No of Patients  No of Patients In hospital  \
0       11419127.0                  11784.797852
1       11082633.0                  12945.036133
2       10991059.0                  13507.216797
3       10967538.0                  13861.316406
4       11016675.0                  13995.288086
5       11064934.0                  13936.730469
6       11132456.0                  13767.951172
7       11192059.0                  13529.098633
8       11237674.0                  13268.882812
9       11266935.0                  13023.745117
10      11280372.0                  12818.916016
11      11280357.0                  12667.890625
```

```
    In mechanical ventillation bed  Registered deaths due to covid
0                       884.414001                     1702.053223
1                       874.473389                     2066.721191
2                       860.026428                     2231.530029
3                       846.109863                     2338.116455
4                       834.575317                     2384.914551
5                       824.773865                     2385.369629
6                       817.795776                     2352.437744
7                       812.887878                     2299.814697
8                       810.033752                     2238.766113
9                       808.868530                     2178.923584
10                      809.031921                     2127.116211
11                      810.114746                     2087.379639
```

Error Metric Evaluation:

Following is the screenshot of the error metrics for all the variables:



```
Error Metrics for 'No of Patients': MAE = 579971.9583333334, MSE = 712212586934.2222, RMSE = 843926.8848272475, MAPE = 19.66273
483540307%
Error Metrics for 'No of Patients In Hospital': MAE = 811.6278279622396, MSE = 985645.1732772796, RMSE = 992.796642458706, MAPE
= 37.004579352813465%
Error Metrics for 'In Mechanical Ventilation Bed': MAE = 56.405930836995445, MSE = 5599.92365751117, RMSE = 74.83263764903099,
MAPE = 11.461782266349887%
Error Metrics for 'Registered Deaths Due to COVID': MAE = 187.37717607286243, MSE = 105725.33182162141, RMSE = 325.154320010701
1, MAPE = 26.263940858532962%
```

Figure 40 Error Metrics for all Variables for GRU model.

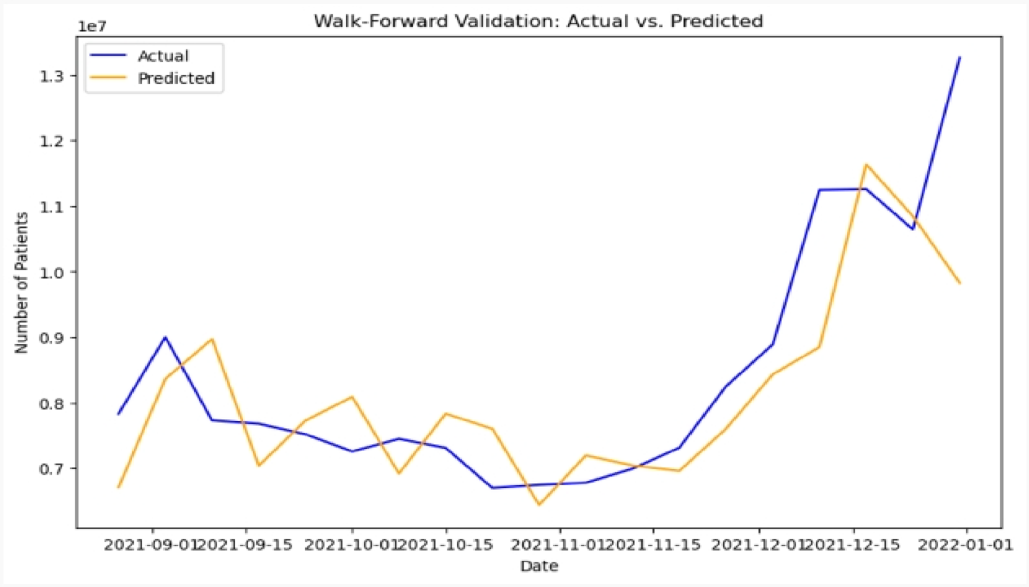
From the error metrics, for Total Patients, it underestimated counts during peak periods (RMSE: 8.4 million; MAPE: 19.66%), indicating difficulty in capturing sudden spikes. While it captured general trends for Patients in Hospital, it faltered during surges (RMSE: 992.79; MAPE: 37%). The model performed better for Mechanical Ventilation (RMSE:74.83; MAPE: 11.46%), managing smoother patterns with fewer fluctuations. However, for Registered Deaths, it generated unrealistic negative values (RMSE: 325.15; MAPE: 26.26%), reflecting poor handling of extreme variations.

F: Dashboard Visuals:

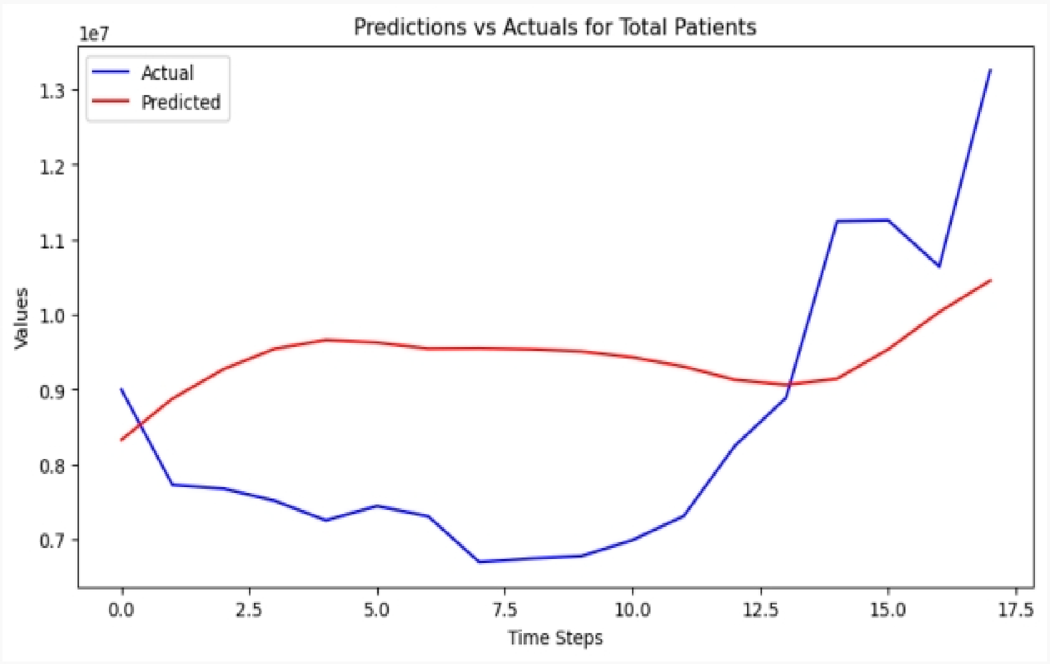
Following are the visuals of the dashboard:

Forecasting Techniques Comparison

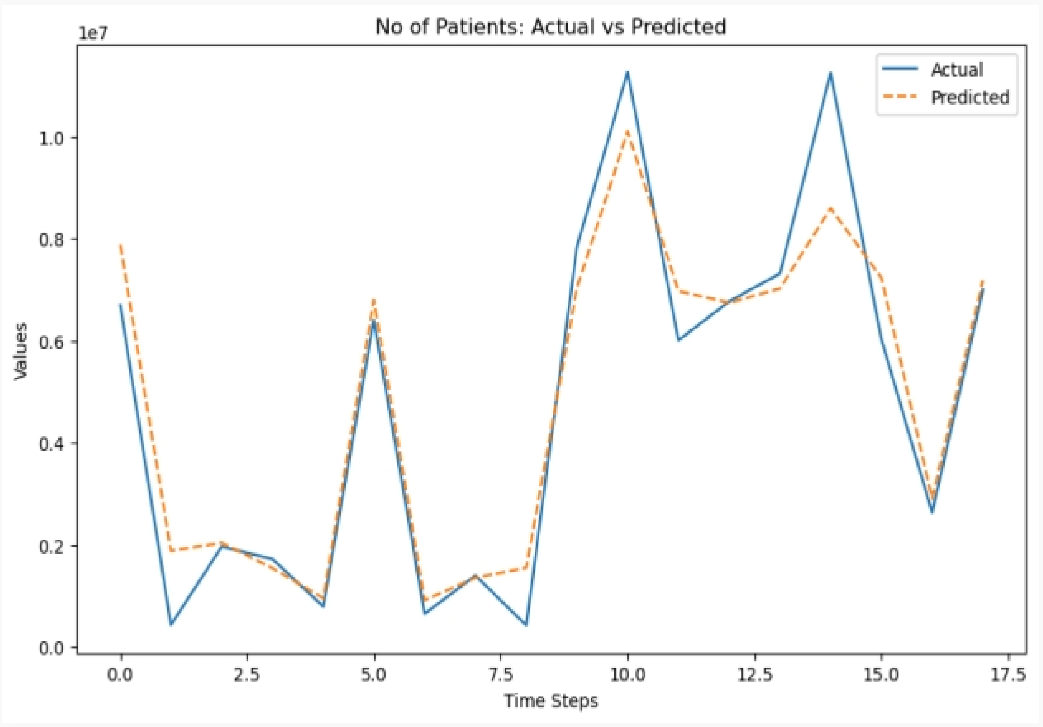
Comparison for No of Patients



ARIMA Walk-Forward Validation



LSTM with LLM

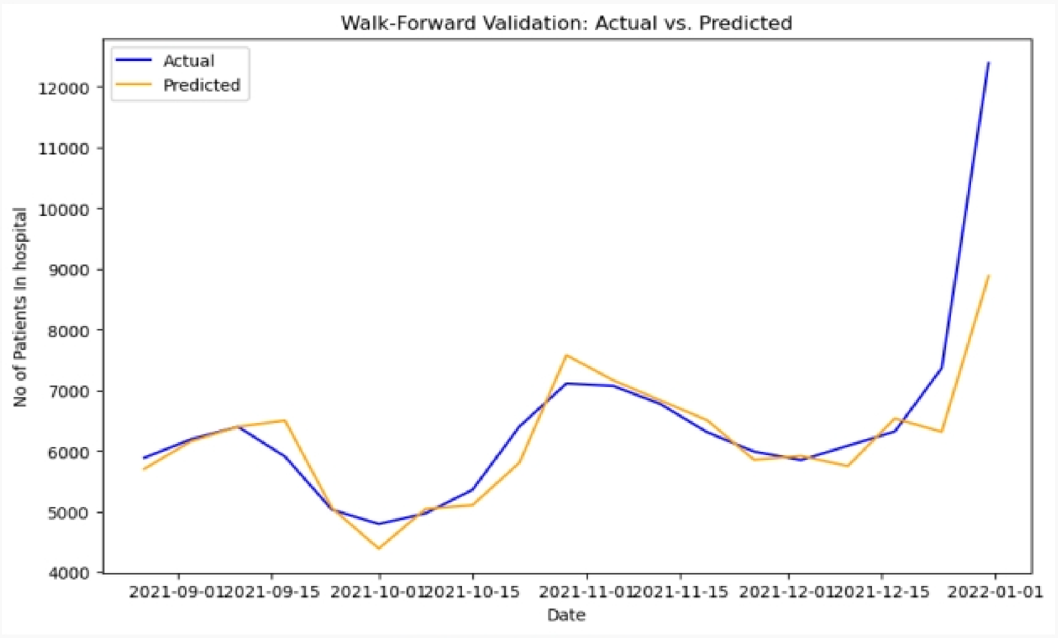


LSTM without LLM

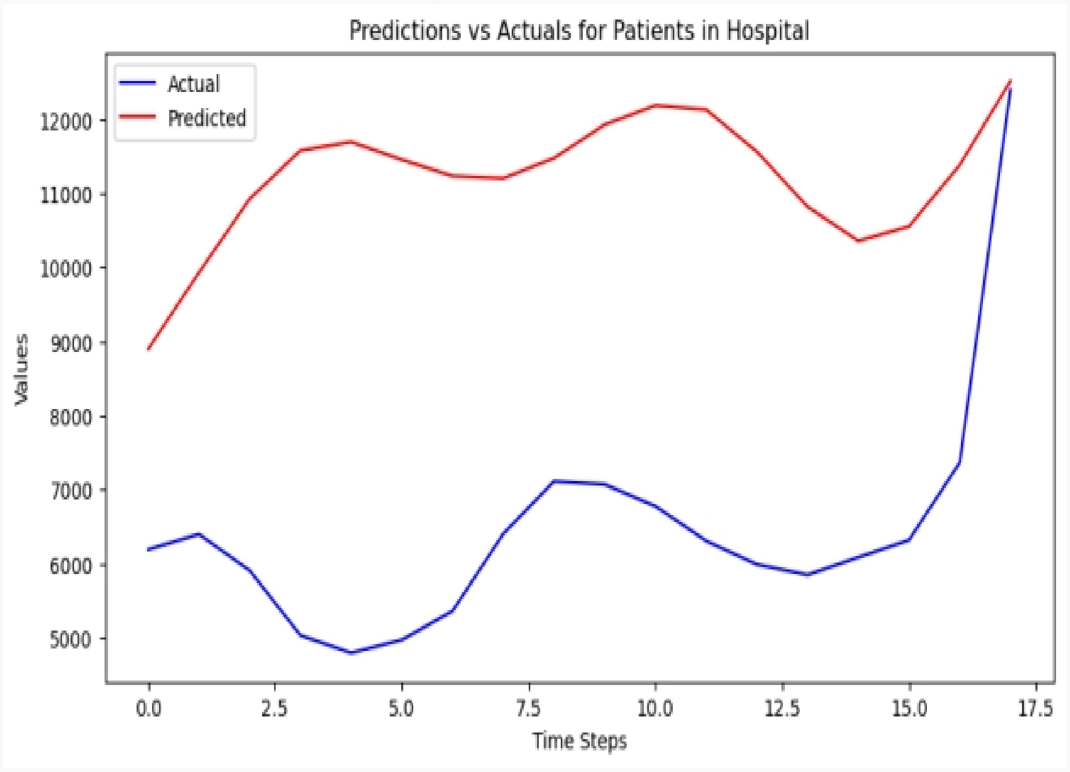
Summary

For 'No of Patients', ARIMA shows a consistent trend in its predictions, though it slightly underestimates rapid increases. LSTM with LLM captures more of the sudden changes, while LSTM without LLM offers the closest alignment with actual values overall.

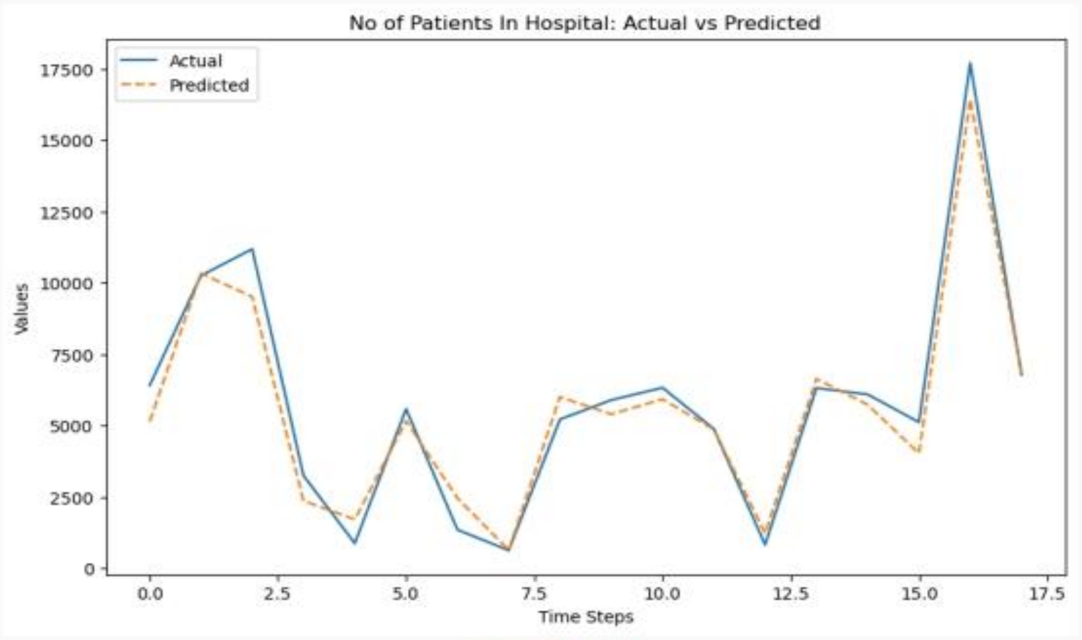
Comparison for No of Patients In hospital



ARIMA Walk-Forward Validation



LSTM with LLM



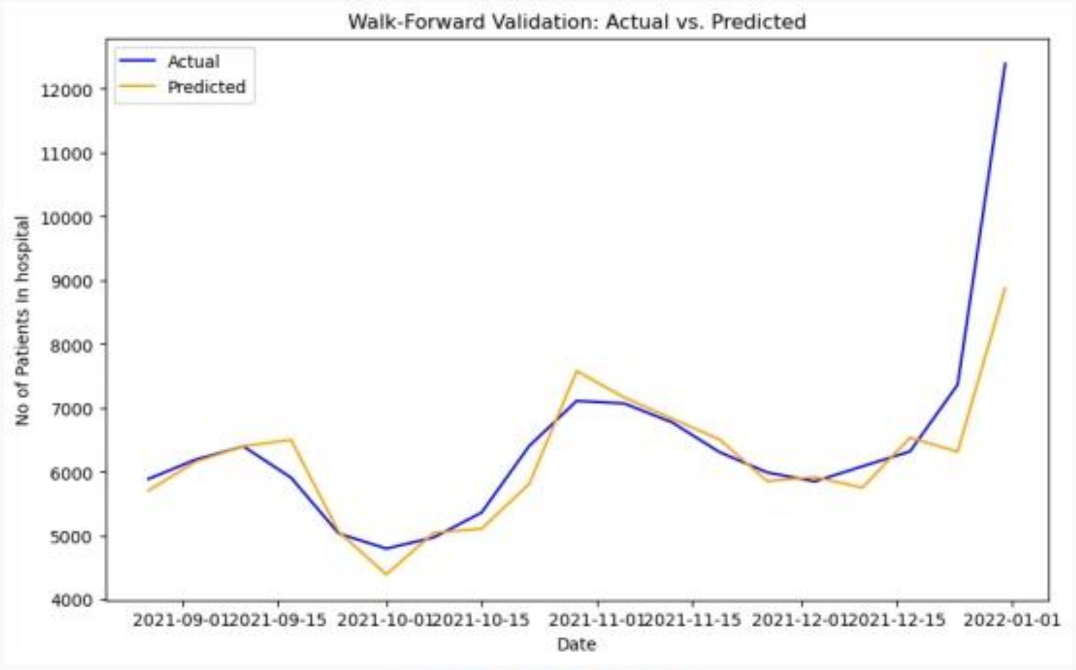
LSTM without LLM

Summary

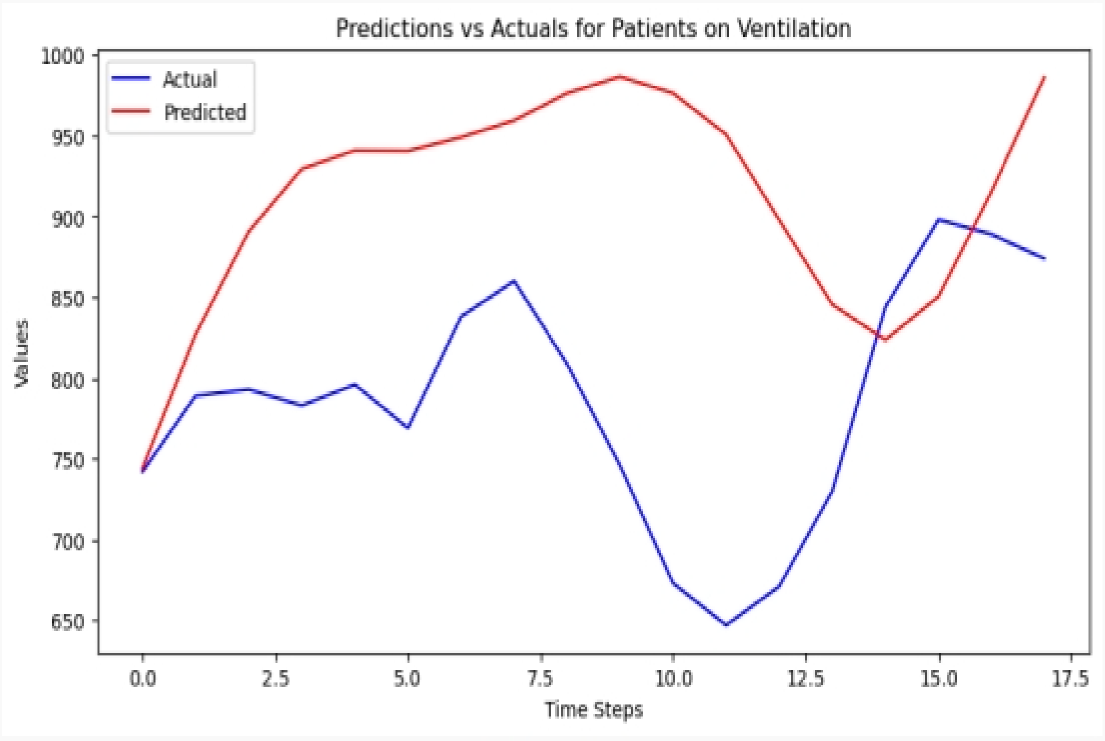
For 'No of Patients In hospital', ARIMA struggles with some peaks, whereas LSTM with LLM adjusts more smoothly. LSTM without LLM performs the best in handling the significant fluctuations.

Comparison for In Mechanical Ventilation Bed

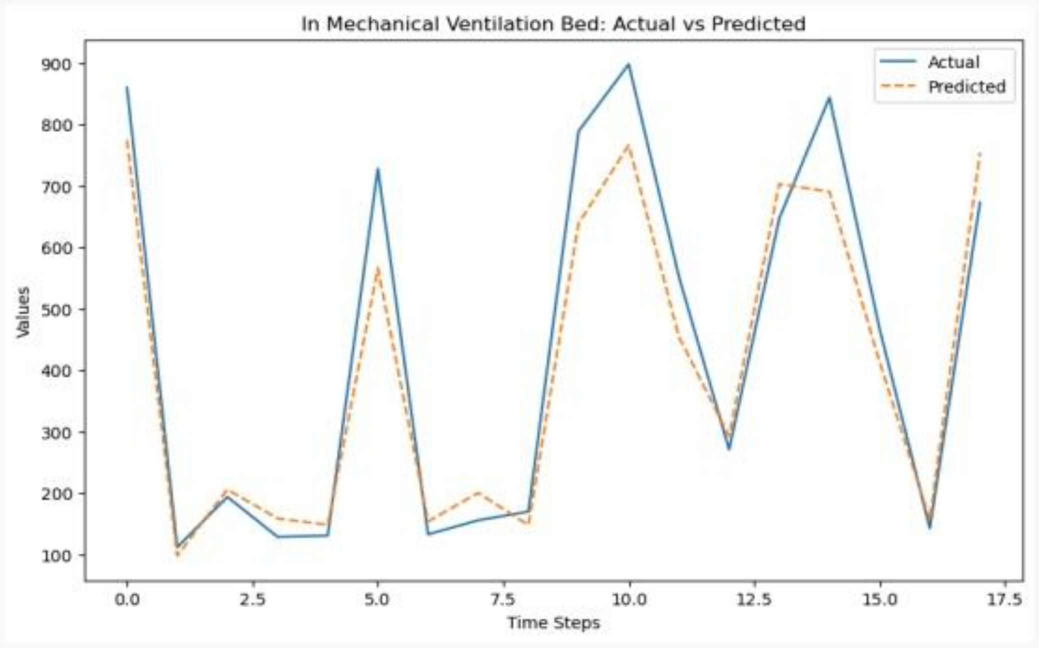
19/09/2024, 21:53 Dash



ARIMA Walk-Forward Validation



LSTM with LLM

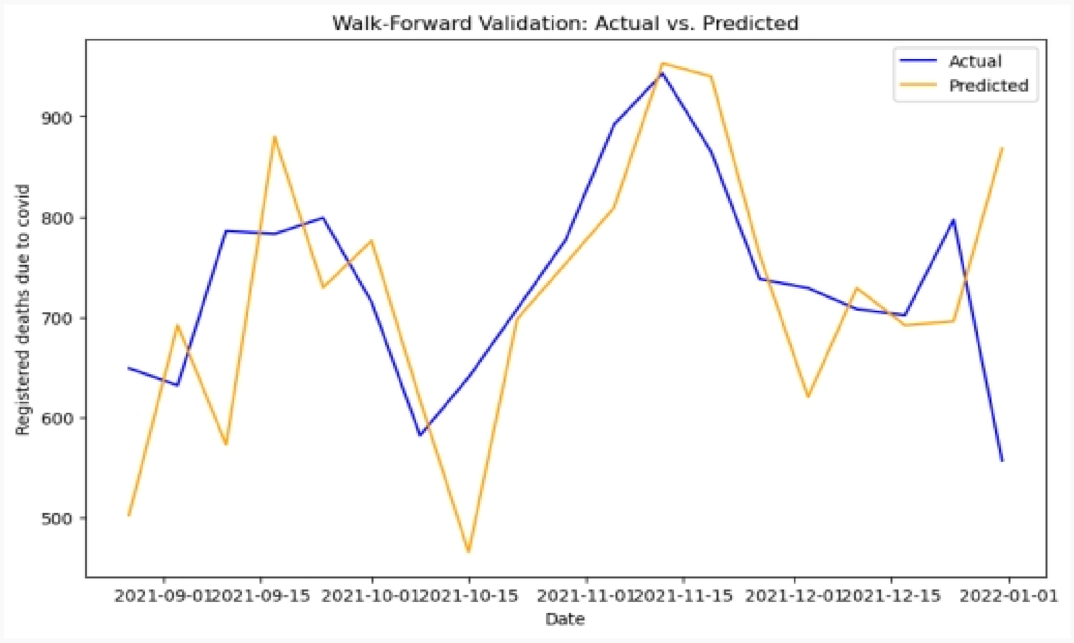


LSTM without LLM

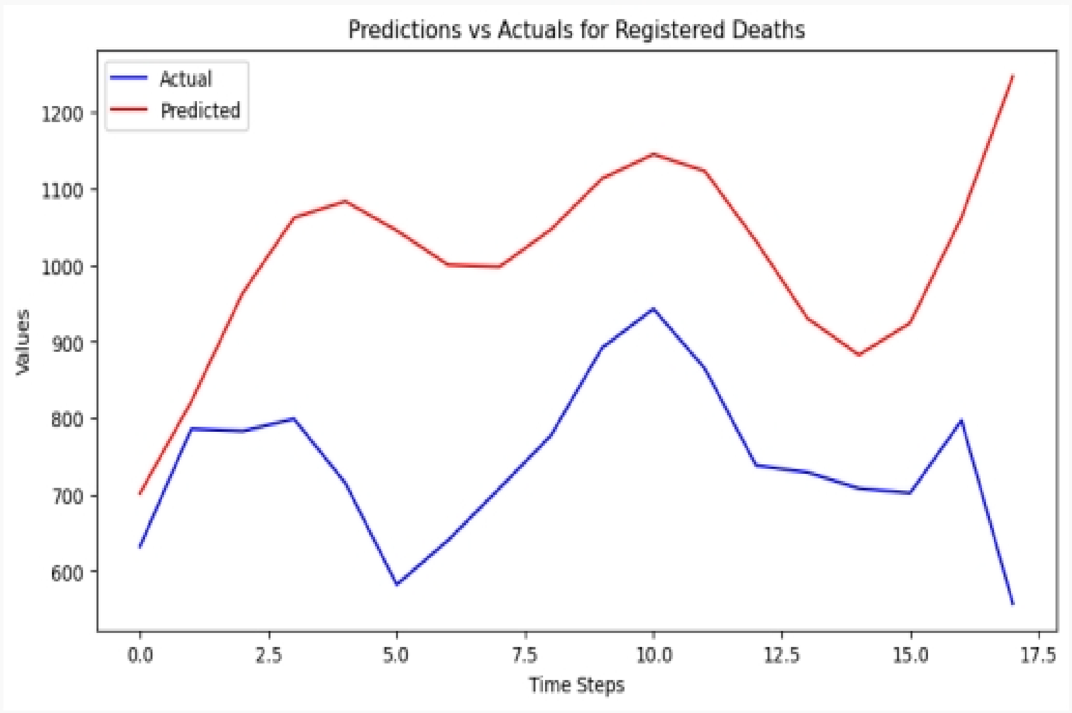
Summary

In predicting 'Mechanical Ventilation Bed' usage, LSTM with LLM provides a slightly better fit compared to ARIMA, but LSTM without LLM shows the best accuracy, particularly for sharp spikes in the data.

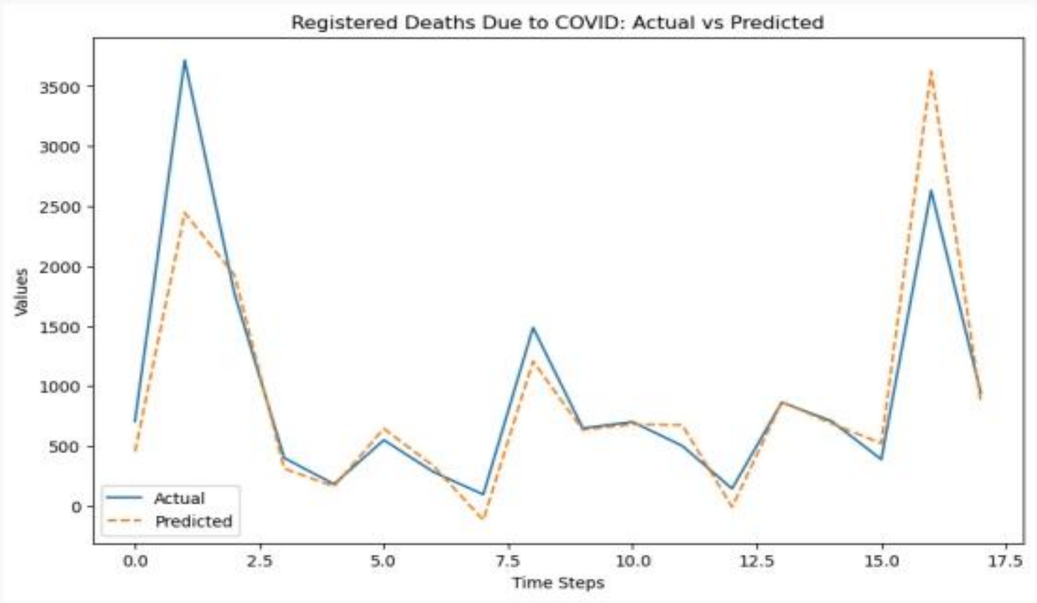
Comparison for Registered Deaths due to Covid



ARIMA Walk-Forward Validation



LSTM with LLM



LSTM without LLM

Summary

For 'Registered Deaths due to Covid', all models follow the general trend, but LSTM without LLM has the smallest error in tracking the actual death toll variations, especially during sudden rises.