TIME SERIES FORECAST OF CALL ARRIVALS USING MACHINE LEARNING METHODS

PEERAWIT SURASAI

Graduate School Srinakharinwirot University

2023

การคาดการณ์จำนวนสายที่ติดต่อโดยใช้แบบจำลองการเรียนรู้ของเครื่องด้วยข้อมูลเชิงอนุกรมเวลา

พีรวิทญ์ สุระสาย

สารนิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการข้อมูล
คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ
ปีการศึกษา 2566

TIME SERIES FORECAST OF CALL ARRIVALS USING MACHINE LEARNING METHODS

PEERAWIT SURASAI

A Master's Project Submitted in Partial Fulfillment of the Requirements

for the Degree of MASTER OF SCIENCE

(Data Science)

Faculty of Science, Srinakharinwirot University

2023

THE MASTER’S PROJECT TITLED

TIME SERIES FORECAST OF CALL ARRIVALS USING MACHINE LEARNING METHODS

BY

PEERAWIT SURASAI

HAS BEEN APPROVED BY THE GRADUATE SCHOOL IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE MASTER OF SCIENCE
IN DATA SCIENCE AT SRINAKHARINWIROT UNIVERSITY

(Assoc. Prof. Dr. Chatchai Ekpanyaskul, MD.)

Dean of Graduate School

ORAL DEFENSE COMMITTEE

.... Major-advisor	.... Chair
(Vera Sa-Ing, Ph.D.)	(Assoc. Prof.Supatana Auethavekiat, Ph.D.)
	.... Committee
	(Subhorn Khonthapagdee, Ph.D.)

Title	TIME SERIES FORECAST OF CALL ARRIVALS USING MACHINE LEARNING METHODS
Author	PEERAWIT SURASAI
Degree	MASTER OF SCIENCE
Academic Year	2023
Thesis Advisor	Vera Sa-Ing , Ph.D.

This study focuses on enhancing workforce management in the Citizen Service Request (CSR) Call Center dataset of the government of Cincinnati, Ohio, by improving the accuracy of call arrival forecasts. Recognizing the pivotal role of precise call arrival predictions in optimizing call center operations, this the study conducts experiments by utilizing a range of forecasting models, including statistical, machine learning, and neural network approaches. Feature engineering was proposed to broaden the scope of features for forecasting. The top-performing models are evaluated based on key metrics such as Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), and R-Squared (R^2) forecasting performance. The experimental results highlighted the comparative performance of various models, such as SARIMAX, Light Gradient Boosting Machine (Light GBM), Gradient Boosting Regressor (GBR), eXtreme Gradient Boosting (XGBoost), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). Among these, Support Vector Regression (SVR) leads in accuracy with an MAE of 25.13, an MAPE of 6.15%, an RMSE of 34.46, and an R^2 of 90.56%. The features of abandon rate, answer speed, service level calls, and the 1st and 5th lags, were identified as the most importance feature in this research. These findings provide valuable insights for the improvement of workforce management strategies in call center operations, emphasizing the effectiveness of machine learning algorithms in achieving more accurate call arrival forecasts.

Keyword : Workforce Manangent, Machine Learning, Support Vector Machine

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude and appreciation to Dr. Vera Sa-Ing, whose unwavering support, guidance, and mentorship have been instrumental throughout the entire research process. Dr. Sa-Ing's expertise and commitment to excellence have not only made her an exceptional teacher but also a source of inspiration.

Special thanks to Associate Professor Supatana Auethavekiat, Ph.D., for graciously accepting the role of chairman during the oral defense examination. I am truly honored to have benefited from Dr. Auethavekiat's invaluable insights, sound advice, and constructive feedback of the research to the final revisions. Her dedication to the refinement of ideas and lessons shared have greatly enriched the quality of this work.

I would also like to extend my sincere appreciation to Dr. Subhorn Khonthapagdee for her insightful guidance and continuous support as a committee member. From the initial proposal to the final oral examination, Dr. Khonthapagdee's expertise has played a crucial role in shaping the research into its final form.

To my family, your hope and unwavering encouragement have been my pillar of strength throughout this journey. Your support has meant the world to me, and I am profoundly grateful for the love and understanding you have provided.

Lastly, I acknowledge myself for the dedication and perseverance that have brought me to this significant milestone. This journey has been challenging, but the personal growth and resilience gained along the way have been invaluable.

Thank you to everyone who has contributed to this endeavor, directly or indirectly. Your support has been a driving force behind the successful completion of this research.

PEERAWIT SURASAI

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background

Workforce Management (WFM) is a strategic approach that organizations employ to optimize the efficiency and productivity of their workforce. It encompasses various key components, including workforce planning, scheduling, time and attendance management, task assignment, performance management, and training and development. Workforce planning involves forecasting future needs and identifying talent gaps, while scheduling ensures that the right employees, with the right skills, are in the right place at the right time. Time and attendance management tracks working hours and ensures compliance with labor regulations. Task assignment optimizes productivity by matching employees to tasks based on skills and availability. Performance management aligns individual performance with organizational goals, and training and development enhance workforce skills. Technology, such as workforce management software, plays a crucial role in automating and streamlining these processes. The overarching goal of Workforce Management is to create a dynamic and agile workforce that can adapt to changing business needs, ultimately contributing to enhanced operational performance and organizational success.

A Call Center Operation is a specialized business unit designed to handle large volumes of incoming and outgoing customer communications, typically via telephone, email, or chat. These operations play a pivotal role in customer service, technical support, sales, and various other functions, serving as a primary point of contact between an organization and its customers. Call centers employ a team of agents who are trained to address customer inquiries, resolve issues, and provide assistance. Efficient call center operations involve workforce management for optimal staffing levels, robust training programs to equip agents with the necessary skills, and the implementation of technology, such as interactive voice response (IVR) systems and customer relationship management (CRM) software, to streamline processes. Key performance indicators like average handling time, first-call resolution, and customer satisfaction are often used to gauge and improve the effectiveness of call center operations. The success of a call center is not only measured by its ability to handle inquiries effectively but also by its contribution to customer loyalty and overall organizational success.

Forecasting in a call center is a critical component of effective workforce management, aiming to predict and plan for future customer interaction volumes. It involves a comprehensive analysis of historical data, trends, and various factors influencing call volumes to make accurate predictions. By examining past call patterns and considering external variables such as marketing initiatives or economic factors, organizations can anticipate peak times and plan staffing levels accordingly. Regular monitoring and adjustment are essential for ensuring that forecasts remain aligned with real-time data and changing conditions. Accurate forecasting contributes to improved customer service by minimizing wait times, reducing the likelihood of abandoned calls, and enhancing overall operational efficiency.

In this study, diverse groups of forecasting models have been selected to enhance workforce management without relying on expensive software. Traditional time series models, such as SARIMAX, leverage statistical methods to capture complex patterns. Machine learning models, including Support Vector Regression (SVR) and LightGBM, harness the power of algorithms to predict non-linear relationships and handle large datasets efficiently. Ensemble models like Gradient Boosting Regressor and XGBoost combine multiple weak learners to create robust predictive models. Additionally, deep learning models, namely Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM), specialize in capturing dependencies and patterns over time. Notably, these chosen models are informed by insights from a former study on Call Center Time Series Forecasting, adding practical relevance and empirical grounding to the experiment. By exploring this comprehensive range of forecasting techniques, the study aims to identify the most effective and cost-efficient solution for optimizing workforce management in call center operations.

In evaluating the performance of the selected forecasting models, this study employs several key metrics, namely Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared (R2). These metrics serve as quantitative measures to assess the accuracy and reliability of the models in predicting workforce management outcomes. MAE represents the average magnitude of errors between predicted and actual values, providing insight into the forecasting precision. RMSE measures the square root of the average squared differences between predicted and actual values, offering a comprehensive assessment of prediction errors. MAPE calculates the average percentage difference between predicted and actual values, providing a relative measure of forecasting accuracy. R2, or the coefficient of determination, assesses the proportion of the variance in the dependent variable that is predictable from the independent variables, indicating the overall goodness of fit of the models. The utilization of these metrics enhances the study's ability to systematically evaluate and compare the forecasting performance of the diverse model groups in the context of call center workforce management.

The structure of this paper is outlined as follows: Chapter 2 provides literature review on research related to forecasting in call centers using machine learning methods. The methodology employed is detailed in Chapter 3. Subsequently, Chapter 4 presents the experimental results, while Chapter 5 engages in discussions and offers conclusions based on the findings.

1.2 Objectives of the study

The objectives of the study are as follows;

1.2.1 To use machine learning approaches based on regression techniques to forecast the call volume.

1.2.2 To study which feature influences call arrival predictions.

1.2.3 To understand seasonality patterns and trends in call volume, which can assist businesses in modifying their staffing and resource allocation plans appropriately.

1.2.4 To experiment how well various machine learning models perform when forecasting call volume.

1.3 Benefits of the study

The application of machine learning models for time series call volume forecasting has various advantages;

1.3.1 Enhanced Accuracy: Effective machine learning models are able to examine vast volumes of data and spot patterns that human analysts might overlook. Predictions of call volumes may therefore be less likely to underestimate or exaggerate.

1.3.2 Better Resource Planning: Accurate forecasting can help companies prepare for the amount of staff they will need, such as managers, call center agents, and other resources. This can save costs while maintaining service level agreements by avoiding either overstaffing or understaffing (SLAs).

1.3.3 Improving Customer Satisfaction: Wait times can be reduced and first call resolution rates can be increased with the help of accurate forecasting.

1.3.4 Reduced Costs: By accurately projecting call volumes, organizations can maximize staffing and other resources, reducing costs related to overstaffing or understaffing and decreasing the need for overtime.

1.4 Scope and limitations

1.4.1 Only daily data points are used to process the forecast.

1.4.2 The available models include XGBoost, RNN, GRU, LSTM, SARIMAX, SVR, LightGBM, Gradient Boosting Regressor, and XGBoost.

1.4.3 To avoid overfitting and enhance the generalization performance of RNNs, GRUs, and LSTMs, the available model architectures for RNNs, GRUs, and LSTMs are restricted to 32, 64, 128, 256, and 512 nodes, with a maximum epoch of 200

1.4.4 The tree-based model and correlation analysis are the only methods used in this work to determine feature importance.

1.5 Procedure of the research

1.5.1 Examine the workforce management and customer service organizational structures, together with any pertinent metrics.

1.5.2 Examine prior studies on forecasting, different time series models, machine learning, and their applications.

1.5.3 Compile call volume data, including historical figures, seasonal patterns, and associated indicators.

1.5.4 Examine the data to look for any trends or patterns in the number of calls made during the day, hour, month, and year.

1.5.5 Verify that the data is appropriately formatted for machine learning and improve it by handling any outliers or missing values.

1.5.6 Using the selected algorithm or algorithms and the ready data, create the machine learning models.

1.5.7 Evaluate the performance of the machine learning model by comparing it to previous call volume and models.

CHAPTER 2
LITERATURE REVIEW

2.1 Workforce Management and Call Center

Workforce Management (WFM) is a process used by call centers to maximize agent productivity. Meeting service level agreements requires completing a variety of responsibilities, including call volume predictions, agent scheduling, performance monitoring, and real-time changes. A well-implemented WFM process can lead to increased customer satisfaction, decreased operating costs, and increased agent productivity (Koole & Li, 2023). This article gives an overview of workforce planning for call centers and highlights its significance for call center operations optimization. Workforce planning and its essential elements—forecasting, scheduling, and real-time management—are defined at the outset of the article.

The significance of precise forecasting for call center workforce planning and gives a summary of popular forecasting techniques like regression and time series analysis. Along with emphasizing the value of high-quality data for forecasting, the article offers advice on how to make data better.

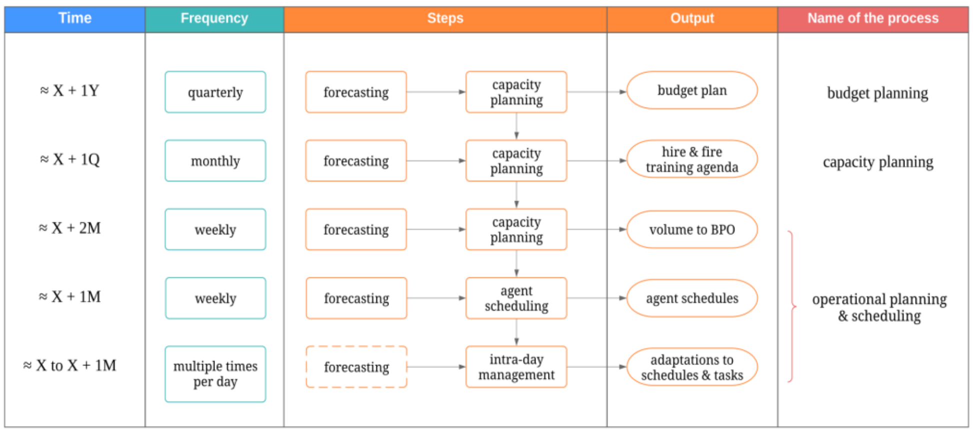


Figure 1 WFM process (Koole & Li, 2023)

A call center's performance can be evaluated using a number of critical metrics. The following list of frequently used metrics includes definitions for each. (Takwi, 2021)

2.1.1 Service Level: The proportion of calls that are returned within a predetermined window of time. A service level of 80/20, for instance, indicates that 80% of calls are returned in less than 20 seconds. Since service level has a direct impact on customer satisfaction, it is an important metric.

2.1.2 Average Handle Time (AHT): The mean duration of an agent's call handling, encompassing talk, hold, and any post-call tasks. AHT is a crucial indicator of agent productivity and efficiency.

Talk Time: The amount of time an agent actively converses with a customer, answering their questions, resolving their problems, or giving them information.

Hold Time: The duration of a customer's call, if any, while they are placed on hold.

Wrap Time: The period of time following the conclusion of the call in which the agent finishes up tasks like updating files, recording the discussion, or getting ready for the next call.

2.1.3 Abandonment Rate: The proportion of calls that clients hang up on before getting through to a representative. Excessive rates of abandonment may indicate extended wait periods or inadequate staffing.

2.1.4 First Call Resolution (FCR): The proportion of calls that get resolved without the need for escalation or follow-up calls on the first try. FCR is a crucial indicator for gauging client happiness and lowering call volumes.

2.1.5 Utilization rate : The percentage of time that agent is actually having a consersation with client, inputting the details of calls, and coordinations.

2.1.6 Average Speed of Answer (ASA): The average response time of an agent when they receive a call. An important metric for evaluating wait times and customer satisfaction is ASA.

2.1.7 Call Quality: is a metric that is subjectively used to evaluate how well customers and agents interact. Call quality can be measured by listening to calls to make sure that scripts and procedures are followed, or by using customer satisfaction surveys.

Effective workforce management is crucial for contact centers to maintain effectiveness and provide high quality customer service. In this context accurate call volume forecasting plays a role and time series models have proven to be tools, for this purpose. By analyzing data on call volumes and identifying patterns and trends these models can generate precise forecasts of future call volumes. This empowers contact centers to optimize their staffing levels and allocate resources efficiently.

One of the advantages of time series models is their ability to handle stationary and non-linear patterns in call volume data. This is especially crucial for contact centers because call arrivals can fluctuate over time owing to seasonality, holidays, and other special events. By recognizing these patterns and incorporating them into their forecasts time series models provide contact centers with a accurate understanding of call volume trends. Consequently, they can make informed decisions regarding staffing and resource allocation.

Depending on the type of data, the research question, and the analytical approach used, several steps in the data analysis process may be followed. Nonetheless, in general, the following primary processes can be separated out of data analysis:

Data Collection: The initial stage of the framework is gathering pertinent information from multiple sources. These data may originate from a number of sources, including databases, sensors, web crawlers, and social media platforms. They may also be structured or unstructured.

Data Preprocessing: To make sure the data is clear, consistent, and prepared for analysis, preprocessing is required after it has been gathered. This entails doing things like eliminating outliers, filling in the blanks, and eliminating duplicate records.

Data Exploration: Examining the data to discover more about its characteristics and to search for any trends or connections is the next step. Tasks that may be involved include statistical analysis, exploratory data analysis, and data visualization.

Feature Selection: The next stage is to choose the most pertinent features or variables for the analysis after the data has been thoroughly examined. This entails choosing the features that have the greatest predictive power and are most pertinent to the current issue.

Model Selection: The next stage after feature selection is to choose a suitable model or algorithm for data analysis. One could use a deep learning model, a machine learning algorithm, or a statistical model.

Training: To discover the connections between the input features and the output variable, the selected model needs to be trained using the available data.

Evaluation: After the model has been trained, it must be tested on an alternative validation dataset in order to assess its performance and make any required parameter modifications.

2.2 Time Series and Machine Learning Models

Forecasting is a practice in industries to make predictions about future trends and outcomes based on historical data. A collection of data on a variable that are logged over an extended period of time is called a time series. What distinguishes time series data from other forms of data is the temporal order of the data that are gathered at intervals. The various machine learning have been applied to model the in time series analysis to forecast call arrivals.

2.2.1 Autoregressive Integrated Moving Average (ARIMA)

ARIMA (Box & Jenkins, 1976) was initially proposed by Box and Jenkins in 1976. This model has been extensively employed for forecasting purposes over the years. It incorporates autoregressive differencing and moving components to capture the dependencies and trends within a time series.

The model is made up of three major parts: autoregression (AR), integration (I), and moving average (MA). ARIMA as following formula:

$$Y_t = c + \emptyset_1 Y_{t-1} + \emptyset_2 Y_{t-2} + \cdots + \emptyset_p Y_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Here's what each term represents:

- Yt : Observation at time t.
- c: A constant term (intercept).
- $\emptyset_1$, $\emptyset_2$,..., $\emptyset_p$: It considers the relationship between the observation and its preceding p observations, as part of the component.
- θ_1, θ_2,..., θ_q: The moving average component involves parameters θ_q that contribute to modeling aspects related to averaging. These show an association between the observation's error term and the q prior error terms.
- ε_t : he error term at time t is denoted as *t*.

Three elements can be used to describe the model.

Component (p) or Autoregressive (AR)

$$Z_t \ = c + \emptyset_1 Z_{t-1} + \ \emptyset_2 Z_{t-2} + \cdots + \emptyset_p Z_{t-p}$$

When Z_t represents the observation at time t it signifies that the current value of "available" is modeled based on its values. The number of values considered for modeling the current value is determined by the autoregression order denoted as (p).

Component (d) or Integrated (I): The *d* is a representation of the amount of differencing needed to make the time series stationary. If *d* =1, the 1st difference in the series (($Z_1 - \ Z_{t-1}$) is used. If *d* =2, take the second difference, and so on.

Component (q) or Moving Average (MA): The modeling of the data error term is the main goal of this component.

$$Z_t = c + \ \theta_1 \varepsilon_{t-1} + \ \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} + \ \varepsilon_t$$

The discrepancy between the variable's actual and expected values as determined by the autoregressive and integration components is represented by the error term. The error term is represented by the moving average component as a linear combination of previous error terms. The order of the moving average (q) dictates how many prior error terms were used for modeling the present error term.

With the use of ARIMA models, one can analyze time series data and determine the proper values for p, d, and q based on particular data characteristics. Values for p, d, and q are frequently determined through parameter tuning.

2.2.2 Seasonal ARIMA (SARIMA)

SARIMA, or Seasonal Autoregressive Integrated Moving Average, is used to make predictions, about values based on observations and their associated errors. Its goal is to determine which set of parameters, taking into account both short- and long-term trends and fluctuations, most effectively captures the inherent patterns within time series data:

$$Y_t = c + \emptyset_1 Y_{t-1} + \emptyset_2 Y_{t-2} + \cdots + \emptyset_p Y_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t +$$

$$+ \emptyset_1^s Y_{t-s} + \emptyset_1^s Y_{t-2s} + \cdots + \emptyset_P^s Y_{t-Ps} + \theta_1^s \varepsilon_{t-1s} + \theta_2^s \varepsilon_{t-2s} + \cdots + \theta_Q^s \varepsilon_{t-Qs} + \varepsilon_{t-s}$$

- The non-seasonal model parameters p, d, and q stand for the moving average, differencing, and autoregressive components.
- All three parameters—P, D, and Q—are comparable excluding the seasonal component.
- s stands for the seasonal cycle's duration.
- $\emptyset_1^s, \emptyset_2^s, \ldots, \emptyset_P^s$ are the seasonal autoregressive parameters representing the relationship between Y_t and its lagged values at seasonality s.
- $\varepsilon_{t-s}, \varepsilon_{t-2s}, \ldots, \varepsilon_{t-Qs}$ are the white noise error terms at seasonal lags.
- $\theta_1^s, \theta_2^s, \ldots, \theta_Q^s$ are the seasonal moving average parameters representing the relationship between Y_t and the white noise error terms at seasonal lags.

The SARIMA model predicts future values based on previous observations and their errors. The objective is to determine the optimal set of these parameters that accounts for both short- and long-term trends while capturing the inherent patterns in the time series data.

For time series data with recurring patterns, like sales data that exhibits a seasonal trend all year long, SARIMA makes forecasting easier. By choosing the right set of parameters depending on the features of the particular dataset under study, the model's efficacy is ascertained.

There is research on the application of SARIMA models to forecast emergency services call volume (Tunnicliffe Wilson, 2016). The authors collected call volume data from an Indian metropolitan area over a ten-year period. They then used several SARIMA models to forecast call volume for the following year. Researchers have explored the application of SARIMA models to predicting call volumes for emergency services. Their findings indicate that SA RIMA models provide predictions for call volume successfully capturing how holidays and other events impact phone usage.

2.2.3 Seasonal ARIMA (SARIMAX)

SARIMAX, an extension of SARIMA known as Seasonal AutoRegressive Integrated Moving Average with Exogenous Factors," allows for incorporating variables into the modeling process. The factors listed here are examples of exogenous variables that could have an impact on the time series under study. Three elements that are comparable to SARIMA make up the SARIMAX model;

Seasonal Component (S): Like SARIMA, the three components of the SARIMAX model take seasonality in the data into consideration. This method involves recognizing and reproducing repeating cycles or patterns across time.

Components of AutoRegressive (AR) and Moving Average (MA): There are two components, auto-regressive (AR) and moving average (MA) that represent the dependencies in a time series. Whereas the MA component models the relationship between the observation and the residual errors of earlier observations, the AR component illustrates the relationship between the observation and its prior values.

Exogenous Variables (X): In addition to this SARIMA includes a feature called variables (X). These are external variables that have the potential to affect the time series but are not included in it. For example, variables in sales forecasting could be things like holidays or advertising budget. These variables improve the model's ability to describe and capture changes in the time series.

The formula used to include variables, in the model is as follows: This formula shows how each variable impacts the time series.

$$\begin{aligned} Y_t = c + \emptyset_1 Y_{t-1} + \emptyset_2 Y_{t-2} + \cdots + \emptyset_p Y_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} \\ + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t + \\ + \emptyset_1^s Y_{t-s} + \emptyset_1^s Y_{t-2s} + \cdots + \emptyset_P^s Y_{t-Ps} + \theta_1^s \varepsilon_{t-1s} \\ + \theta_2^s \varepsilon_{t-2s} + \cdots + \theta_Q^s \varepsilon_{t-Qs} + \varepsilon_{t-s} + \beta_1 X_{1,t} \\ + \beta_2 X_{2,t} + \cdots + \beta_k X_{k,t} \end{aligned}$$

The differenced and stationary time series are represented by Y_t, and the exogenous variables are represented by $X_{1,t}, X_{2,t}, \ldots, X_{k,t}$. The coefficients $\beta_1 + \beta_2 + \cdots + \beta_k$, which represent the impact of each exogenous variable on the time series, are estimated during the model fitting process.

SARIMAX model is represented mathematically as SARIMAX (p,d,q) (P,D,Q)s, where P,D,Q represents seasonal AR, differencing, and MA orders with a seasonality of ss, and p,d,q represents non-seasonal AR, differencing, and MA orders.

In essence, SARIMAX is an effective time series forecasting tool that produces a more thorough and precise prediction by taking into account both the temporal structure of the data and the possible influence of outside factors. Determining appropriate values for the model's parameters and identifying pertinent exogenous variables present a challenge.

2.2.4 Support Vector Regression (SVR)

A machine learning technique called Support Vector Regression (SVR) was created especially for handling regression tasks involving complex and non-linear relationships between the target variable and the input features. In contrast to linear regression, support vector regression (SVR) uses a kernel trick to map the input features into a higher-dimensional space, which makes it possible to identify the best hyperplane for accurately representing the underlying relationships (see **Figure 2**).

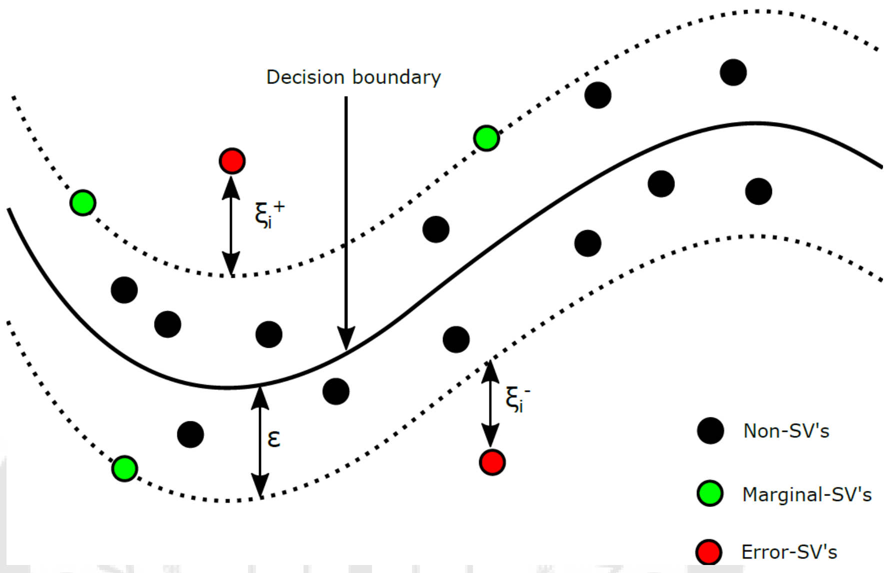


Figure 2 Visualization of hyperplane of SVR (Singh et al., 2020)

Important components of SVR include selecting the kernel function (such as linear, polynomial, sigmoid, or radial basis function) and putting in place an epsilon insensitive loss function. This loss function does not severely penalize SVR when there is a degree of deviation from the predicted value. As a result, SVR shows resilience to noise and anomalies in the data. The following briefly describes the essential elements of SVR:

Kernel Function : SVR creates a dimensional space from input features by applying a kernel function. Linear, polynomial, sigmoid, and Radial Basis Function (RBF) are typical examples of kernel functions. The kind of data being used and the intricacy of the relationship being modeled are two factors that factor into choosing the right kernel function.

Loss Function (Epsilon-Insensitive Loss): There is no penalty for a certain amount of deviation (epsilon) from the forecasted value in this loss function. It aids in managing data and anomalies.

The goal of SVR is to identify a hyperplane that maximizes the margin inside an epsilon insensitive tube while not perfectly fitting the data. The goal is to reduce loss as much as possible while permitting some error. Optimizing the weight vector is required for this.

2.2.5 Light GBM (Light Gradient Boosting Machine)

A model called Light Gradient Boosting Machine, or Light GBM, is intended to effectively train big datasets. It is designed especially for gradient boosting and is a member of the learning algorithm family. Light GBM differs from other tree-growing methods in that it follows a leaf-based approach rather than a depth-first one. This method lowers the model's complexity.

Light GBM converts data into features for time series forecasting, such as lag values and external influences. The next step involves building a series of decision trees iteratively, with each one intended to correct errors made by the group. This methodology enables Light GBM to effectively capture patterns and handle non-linear relationships in time series data. The popular parameters in this model are following;

num_leaves: This parameter controls the maximum number of leaves in one tree. Increasing num_leaves tend to make the model more complex, allowing it to capture more intricate patterns in the data. However, a higher value also increases the risk of overfitting.

learning_rate: Learning rate determines the step size at each iteration while moving toward a minimum of the loss function. A smaller learning rate requires more iterations but can lead to better convergence. It is a crucial hyperparameter that balances the trade-off between model training time and accuracy.

n_estimators: This parameter sets the number of boosting rounds or the number of trees to be built. Increasing n_estimators generally improve model performance, but there is a point beyond which further trees may not significantly contribute to better results. However, more trees also mean longer training times.

reg_alpha: Regularization term on weights (L1 regularization). It adds a penalty term for the complexity of the model. Higher values of reg_alpha increase the regularization strength, helping to prevent overfitting by discouraging the model from assigning excessive importance to any single feature.

Light GBM's ability to handle large datasets is one of its key advantages, which makes it especially useful in situations involving sizable amounts of time series data. Its flexibility in handling missing values and support for distributed computing further improve its scalability and speed.

Light GBM performance is greatly influenced by hyperparameter tuning, with parameters such as learning rate, maximum tree depth, and number of leaves having a significant impact on the model's accuracy. (Saksonita et al., 2022) cites Light GBM as a reliable and efficient option for semiconductor cleanroom forecasting. The study emphasizes Light GBM's effectiveness in predicting outlier particles in this specific environment, proving its suitability as a reliable forecasting tool.

2.2.6 Gradient Boosting Regression (GBR)

Another popular ensemble learning method for time series forecasting is gradient boosting regression (GBR). Usually, the process starts with establishing a prediction, which is the target variable's average. Then, by training them using the negative gradient of the loss functions predictions for the current model, the algorithm creates learners frequently using shallow decision trees. To account for the differences between the predicted values, residuals are computed, and the weak learners prediction is modified and incorporated into the current model. This process is repeated, with each weak learner concentrating on fixing mistakes in the previously constructed ensemble.

In time series forecasting tasks, GBR is utilized as a prediction technique for observations. It entails segmenting the data into target values, which stand in for upcoming observations, and features like lagged values, or external factors. When it comes to identifying non-linear relationships in data and effectively managing missing information, GBR is especially useful in preparing data for forecasting. Nevertheless, optimizing GBR performance necessitates meticulous adjustment of hyperparameters, particularly when handling extensive datasets that might require substantial processing power.

Table 1 Comparative summary highlighting the key differences among Light GBM, Gradient Boosting Regressor, and XGBoost.

Feature	Light GBM	Gradient Boosting Regressor	XGBoost
Algorithm Type	Gradient Boosting	Gradient Boosting	Gradient Boosting
Tree Building Approach	Leaf-wise (Best-First)	Level-wise (Depth-First)	Depth-wise (Depth-First)
Handling Categorical Data	Native Support	Requires Encoding	Requires Encoding
Parallel Training	Yes, supports parallel and distributed training	No	Yes, supports parallel training
Memory Usage	Lower memory consumption	Higher memory consumption	Moderate memory consumption
Regularization	Regularization is available (lambda and gamma)	Regularization options (alpha and lambda)	Regularization options (alpha and lambda)
Handling Missing Values	Native support for missing values	Requires pre-processing for missing values	Native support for missing values
Speed	Generally faster due to leaf-wise approach	Slower due to level-wise approach	Comparable speed, optimized for efficiency
Scalability	Highly scalable, efficient on large datasets	Less scalable compared to Light GBM	Scalable, efficient for large datasets
Use Cases	Efficient for large datasets, handles categorical features well	Broad use cases, suitable for regression problems	Broad use cases, suitable for classification and regression problems

2.2.7 XGBoost (eXtreme Gradient Boosting)

When it comes to tasks like regression, classification, and even time series forecasting, XGBoost is a highly effective and popular machine learning algorithm. In order to produce a final prediction that is both dependable and accurate, XGBoost functions as a learning technique that integrates the predictions from several weak models, most often decision trees.

The secret to the algorithms' performance is their capacity to manage relationships in the data while retaining a high degree of predictive accuracy. Boost converts data into features that include lag values and pertinent external factors in the context of time series forecasting. After that, a series of decision trees are built, each tree repairing the mistakes made by the previous ensemble.

XGBoost includes practical features that add to its efficacy. One feature is regularization, which discourages the use of excessively complicated models and so prevents overfitting. The algorithm also provides a loss function so that users can customize the model to fit their own goals. XGBoost is known for its ability to handle missing values effectively and identify patterns in time series data. Optimizing Boosts performance requires parameter tuning, where factors like learning rate, tree depth, and boosting rounds significantly impact the models' accuracy. Due to its versatility and reliability Boost is a good choice for time series forecasting tasks as it delivers precise predictions across various scenarios.

The comparison among Light GBM, Gradient Boosting Regressor, and XGBoost in Table 1 reveals distinctive features that make each algorithm suitable for specific scenarios. Light GBM, utilizing a leaf-wise tree-building approach, stands out for its lower memory consumption and efficient handling of large datasets, making it an excellent choice for scenarios with extensive data. Its native support for categorical data and parallel/distributed training further enhances its scalability. On the other hand, Gradient Boosting Regressor, with its level-wise tree-building approach, may be comparatively slower and less memory-efficient but is versatile across regression problems. XGBoost, striking a balance between the two, combines efficient memory usage with native support for missing values, making it suitable for various use cases, including both classification and regression tasks. The choice among these algorithms ultimately depends on the specific characteristics of the dataset and the nature of the prediction problem at hand.

2.2.8 Recurrent Neural Network (RNN):

Artificial neural networks that process sequential data by preserving an internal state or memory are known as recurrent neural networks, or RNNs. In contrast to feedforward neural networks, which process input data in a single pass, RNNs are capable of handling sequences of arbitrary length and maintaining information over time.

Two representations of a recurrent neural network (RNN) are shown in **Figure 3**; the left representation is an unrolled version, while the right representation is a loop. Within the network, the closed-loop representation indicates the recurrence of connections. The unrolled version, on the other hand, shows how data moves sequentially through various time steps. Each step's input is denoted by "x," the network's internal state by "S," and the weights of the connections are defined by the matrices "V," "W," and "U". The final output of the network is represented by the letter "o." Understanding how information is logically and sequentially processed over a series of time steps is made easier with the help of this unfolding visualization.

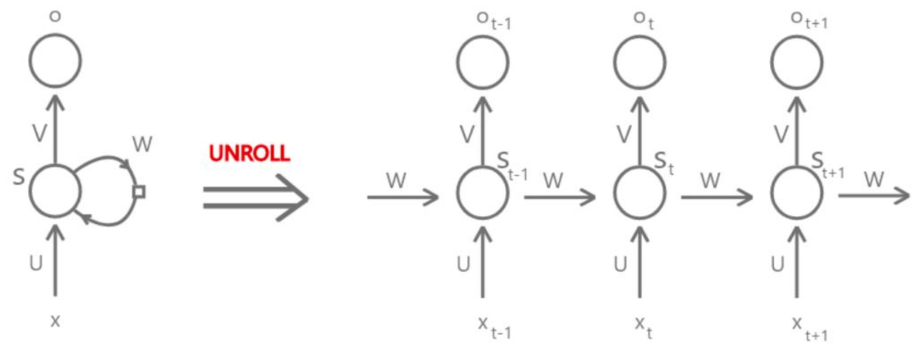


Figure 3 A recurrent neural network's (RNN) two versions: the unrolled version on the right and the closed-loop version on the left (Baldon, 2019)

According to (Werbos, 1988), Paul Werbos is regarded as the father of backpropagation through time (BPTT). His research is regarded as the most thorough source of information on recurrent neural networks (RNNs). An additional noteworthy study by (Graves & Schmidhuber, 2005) makes a case for the efficacy of RNNs in speech recognition. Since then, RNNs have been used in many different fields, including natural language processing, picture captioning, and time series forecasting.

RNNs handle time series forecasting as though it were a series of times-corresponding data points. Every step of the sequence is processed separately by the model, which updates its state based on the information gathered from previous steps and the current input.

Nevertheless, the vanishing gradient problem presents a difficulty for conventional RNNs. This issue limits their capacity to accurately identify long-term dependencies. Advanced RNNs have been developed, such as the Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM), to get around this restriction. These versions overcome the limitations of RNNs and improve their ability to identify and leverage long-term dependencies.

2.2.9 GRU (Gated Recurrent Unit)

The GRU, a type of network (RNN) is widely used for tasks like predicting call arrival volume in sequence modeling and forecasting. It's a variant of RNN that addresses the vanishing gradient problem through gating techniques. In **Figure 4** (on the right) you can see the model architecture. An overview of the GRU model's elements is provided below:

- **Hidden State**: The GRU stores data from previous time steps in a hidden state that corresponds to the model's memory.
- **Update Gate**: The update gate controls how much old data should be. How much new information should be incorporated into the hidden state.
- **Reset Gate**: The reset gate determines how much of the state to ignore when calculating the current hidden state. It allows the reset or deletion of data.
- **Candidate Activation**: At each time step the candidate activation calculates a hidden state by combining information from the hidden state and the current input.

- **Final Hidden State:** is determined by using the update gate to blend the hidden state with the candidate activation. This represents updated memory or information, for that time step.

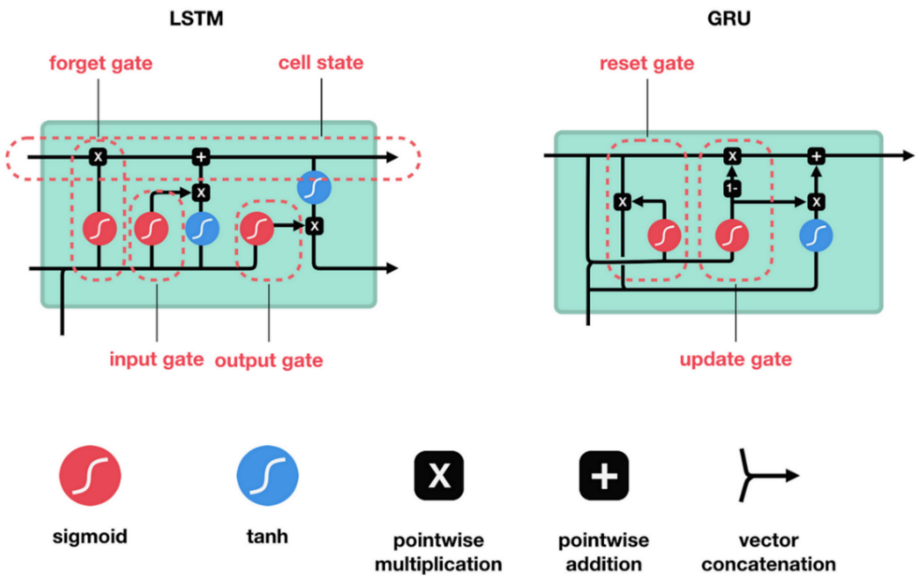


Figure 4 LSTM and GRU model architecture (Phi, 2018)

GRU models have shown performance in tasks like predicting call arrival volume in time series forecasting. One practical application of GRU models is optimizing call center staffing based on accurate call arrival predictions. By ensuring the number of agents during peak times wait times can be reduced. Customer satisfaction can be increased.

2.2.10 LSTM (Long Short-Term Memory)

Long-term dependencies in sequential data, like time series data, can be captured using long-term support graph networks, or LSTMs. They are appropriate for modeling patterns because they have the capacity to selectively remember or forget information over extended periods of time. In this research paper a new type of neural network (RNN) architecture was proposed by (Hochreiter & Schmidhuber, 1997). This innovative architecture incorporated memory cells and gating mechanisms, which addressed the problem of vanishing gradients commonly encountered in RNNs. These elements were introduced first, and they made LSTM (long-short-term memory) RNNs possible. The LSTM architecture consists of four elements as depicted in Figure 4 on the left:

- **Forget gate**: The forget gate controls the flow of information from the previous hidden state to the current hidden state. It determines which past information is still relevant and should be retained while filtering out data.
- **Input gate**: The input gate controls how much new information is incorporated into the cell state from the current input. It decides which aspects of the current input are relevant and should be incorporated into the cell state.
- **Cell state update**: To update the cell state, the cell state update combines information from the forget gate, input gate, and current input. The cell state is the LSTM's core memory component, and it maintains long-term dependencies in sequential data.
- **Output gate**: At the current time step, the output gate determines which information from the cell state is used to produce the output. It determines how much information about the cell state is exposed to the network's subsequent layers.

A deep learning model is proposed by (Kanthanathan et al., 2020) to forecast the workload in contact centers. A recurrent neural network (RNN) trained on call data serves as the foundation for the model. The RNN can learn pattern of time in call data and then use this information to predict future workload. The suggested model was tested using call data from a large contact center. The model achieved MAE of 1.2 calls per hour when compared to other forecasting models. According to the researchers, the proposed approach is an invaluable resource for contact centers looking to improve their prediction abilities. The model is accurate, helpful, and adaptable to various data sets.

2.3 Time Series Cross-Validation

To overcome the particular difficulties that time-ordered datasets provide in machine learning evaluation, TimeSeriesSplit is a kind of time series cross-validation.

Time series cross-validation is an important way to handle the challenges of time-ordered data in machine learning. It uses methods like TimeSeriesSplit to make sure we thoroughly evaluate the model. Unlike regular cross-validation that randomly splits data, time series cross-validation, especially with TimeSeriesSplit, carefully divides the dataset into parts one after the other as shown in **Figure 5**. In this method, each part is used for both training and testing in different rounds. The training part includes past data, and the testing part has data from later times. For example, if we have 100 pieces of data and set n_splits to 5, the model goes through training and evaluation five times. In each round, it moves step by step through the time sequence, keeping the order intact.

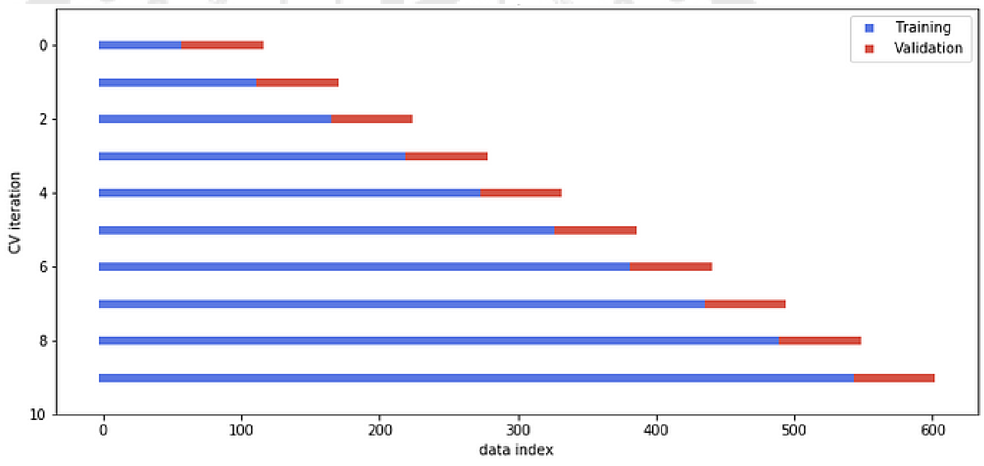


Figure 5 Time Series Cross Validation

This method is useful in time series analysis because it keeps the model from learning about upcoming data while it is being trained. This makes it possible to evaluate its predictive performance in a realistic manner. Sci Kit Learn's TimeSeriesSplit library implementation allows users to define the number of splits (folds) according to the unique properties of their time series data. Time series data must be cross-validated in order to accurately assess a model's ability to generalize over time to hitherto unidentified patterns and identify potential issues such as overfitting to specific trends in the data.

2.4 Error Metrics

In order to assess the accuracy of forecasting models, error metrics are necessary. These metrics quantify the variation between the expected and actual values in a time series. In time series analysis, a model's accuracy is determined by how well it can uncover hidden patterns and trends within the data. Several error metrics are used to assess a forecasting model's performance. Among the most often used error metrics are mean absolute error (MAPE), mean squared error (MSE), root mean square error (RMSE), and mean absolute percentage error (MAPE).

N represents the total amount of value in the time series,

A_i represents the actual value at time i,

F_i represents the forecasted value at time i,

Σ is represents the total of the squared discrepancies between the actual and anticipated values, and | $i = 1$ to N | denotes the summation over all values of i from 1 to N.

2.4.1 Mean Absolute Error (MAE)

One popular error metric used in time series forecasting is mean absolute error (MAE). the time series' average absolute difference between the values of the forecast (F_i) and actual (A_i) values. Divide the total number of samples (N) by the absolute differences for each time point to get the MAE. The average size of the model's prediction error is measured by the MAE, which is expressed in the same units as the time series data. In general, a model with a lower MAE value is more dependable, whereas a larger MAE number indicates less accuracy.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} | A_i - F_i |$$

2.4.2 Mean Squared Error (MSE)

Another statistic that's frequently used in time series forecasting is mean squared error (MSE). The formula for each value in a time series represents the average of the squared differences between anticipated (F_i) and actual (A_i) values. To calculate MSE, square the difference between each time point, add the squared differences, and then divide the total number of samples (N).

Large errors or data outliers are more likely to occur in the MSE than in the MAE since it squares the errors. Expressed in time series data squared units, the MSE calculates the average scale of the squared errors in the model's forecasts. An MSE of less than or equal to a model's accuracy indicates that it is more accurate.The formula for Mean Squared Error (MSE) is as follows:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (A_i - F_i)^2$$

2.4.3 Root Mean Squared Error (RMSE)

In time series forecasting, another commonly used error metric is the Root Mean Squared Error (RMSE). This equation represents a square root of the average of the squared differences between the anticipated (F_i) and actual (A_i) values. To compute RMSE, you square the difference for each time point, sum up these squared differences, take the average, and then calculate the square root of that average. Larger errors are penalized more than smaller errors by the RMSE since the differences are squared. The Root Mean Squared Error (RMSE) can be computed using the formula below:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (A_i - F_i)^2}$$

2.4.4 Mean Absolute Percentage Error (MAPE)

In time series forecasting, mean absolute percentage error (MAPE) is another popular error metric. The average percentage difference in a time series between anticipated (F_i) and actual (A_i) values. To compute MAPE, you calculate the absolute percentage difference for each time point, sum up these absolute percentage differences, and then divide by the total number of samples (N). The final value is expressed as a percentage by multiplying it by 100. A useful metric for comparing forecasting model performance across different time series, particularly when the actual values' magnitudes differ significantly, is the mean absolute percentage error (MAPE) of the model. This formula can be used to determine MAPE:

$$MAPE = \frac{100}{N} \sum_{i=1}^{N} \left| \frac{A_i - F_i}{A_i} \right|$$

Note that the MAPE is less appropriate for time series with large seasonal changes or outliers since it tends to overstate large errors and can produce endless or undefined results when the actual values are zero.

2.4.5 R-squared

The coefficient of determination, also known as R-squared (R2), expresses how closely the forecasts made by the model match the actual values. The following formula is used to calculate it:

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (A_i - F_i)^2}{\sum_{i=1}^{N} (A_i - \overline{A})^2}$$

Where $\overline{A}$ denotes the average of the actual values. R^2 is a number between 0 and 1, with 0 indicating that the model explains no data variability and 1 indicating that it fits perfectly. It indicates how well the independent variable predicts the dependent variable's variance, with greater values suggesting greater model efficiency.

2.5 Related Research

2.5.1 Review of ML and AutoML Solutions to Forecast Time-Series Data (Alsharef et al., 2022)

Automated machine learning (AutoML) and machine learning (ML) techniques are commonly used to forecast time-series data. This paper looks closely at these methods. Both conventional machine learning models and AutoML systems—which help with model selection and hyperparameter tuning—are examined in this study.

In summary, the importance of analyzing time-series data has grown significantly in the last ten years due to the increase in large datasets in various fields. The review discusses different methods for time-series modeling, focusing on linear modeling, deep learning (DL), and Automated Machine Learning (AutoML). Linear models are simple but may not be very accurate in predicting outcomes. DL requires expertise and patience in designing models and optimizing parameters. AutoML, which can automatically identify and optimize machine learning models, shows promise but is still evolving for time-series applications. The article introduces various methods, including AutoML frameworks, for predicting time-series data, offering a comparison. The study acknowledges limitations, such as the lack of an empirical study on the effectiveness of different methods. The next step involves conducting experiments to compare different machine learning and AutoML techniques to address the challenges in modeling time-series data effectively.

2.5.2 Study on the Continuous Quality Improvement of Telecommunication Call Centers Based on Data Mining (Shu-guang et al., 2007)

The study focuses on using data mining techniques to continuously improve the caliber of contact centers for telecom services. The study offers a model that identifies the crucial elements affecting call center quality through feature selection, data pretreatment, and model construction. Data from a telecom provider is used to validate the model.

The paper introduces service quality metrics, explores continuous improvement models using data mining, and focuses on specific areas like Interactive Voice Response (IVR) efficiency and agent performance metrics. It also presents a model for analyzing and controlling Average Speed of Answer (ASA) using data mining and Statistical Process Control (SPC). The results demonstrate that the suggested framework is capable of identifying and improving the significant factors influencing call center quality. Telecommunications companies can use data mining techniques to continuously improve their call center operations, according to the study.

2.5.3 Time Series Forecast of Call Volume in Call Centre using Statistical and Machine Learning Methods (Baldon, 2019)

This study explores the field of time series analysis with a particular emphasis on call load time series modeling and forecasting in contact centers. A time series is a set of data points collected sequentially, examined for temporal correlations, and modeled to identify trends and seasonality. In real-world settings, predicting future values is essential, particularly in contact centers where staff scheduling optimization depends on managing call load. For forecasting, contact centers have typically used basic statistical models or handwritten forecasts. The thesis presents machine learning as an alternative to the widely used SARIMA model in call centers. Specifically, it provides LSTM and Seasonal ANN models. The Normalized Mean Squared Error (NMSE) and the Symmetric Mean Absolute Percentage Error (SMAPE), which evaluates the models' level of accuracy. Three datasets from Teleopti are used in the studies that make up this study. The findings show that SARIMA outperforms Seasonal ANN and LSTM with less data points for daily-scale forecasting on all three datasets. On the other hand, Seasonal ANN and LSTM perform better on an hourly basis and demonstrate resilience across a 160-point forecasting horizon. However, SARIMA shows no relationship at all between the quantity of data points and model quality, whereas Seasonal ANN and LSTM both show improvement as sample size increases. This investigation highlights how machine learning can improve call load forecasting, especially when it comes to handling the complex temporal patterns found in call center data.

2.5.4 Forecasting Call and Chat Volumes at Online Helplines for Mental Health (de Boer et al., 2023)

The aim of this study is to examine how online mental health helplines can predict the total number of calls and chats they will receive by using statistical forecasting models. Over the course of a year, the researchers gathered data from a comprehensive online helpline, and then they used a variety of forecasting models to predict the volume of calls and chats that will be received in the next month. In this study, forecasting methods including exponential smoothing, seasonal ARIMA, ARIMA, and simple linear regression were applied. Forecasting model performance is assessed using the MAPE. The gap between expected and actual values is represented by the percentage value known as MAPE. The seasonal ARIMA model is superior to all others in predicting. The model achieved a MAPE of less than 10%, significantly lower than the LSTM's MAPE, which exceeded 15%. As indicated by the study, statistical forecasting models can effectively predict the volume of calls and online chats for mental health helplines. This predictive capability enables improved planning for staffing levels and resource allocation in helpline operations.

CHAPTER 3
METHODOLOGY

This chapter describes the methodology and data analysis tools used in this study on the following topics:

1. Data understanding
2. Exploratory data analysis and visualization
3. ACF and PACF interpretation
4. Data preparation and feature engineering
5. Data pre-processing
6. Statistical and Machine Learning System
7. Feature Importance

3.1 Data Understanding

(Services, 2023) contains information on Citizen Service Request (CSR) call center calls in Cincinnati, Ohio. The dataset includes information on the caller's type of service request, the time and date of the call, the location of the service request for help, and the request's status.

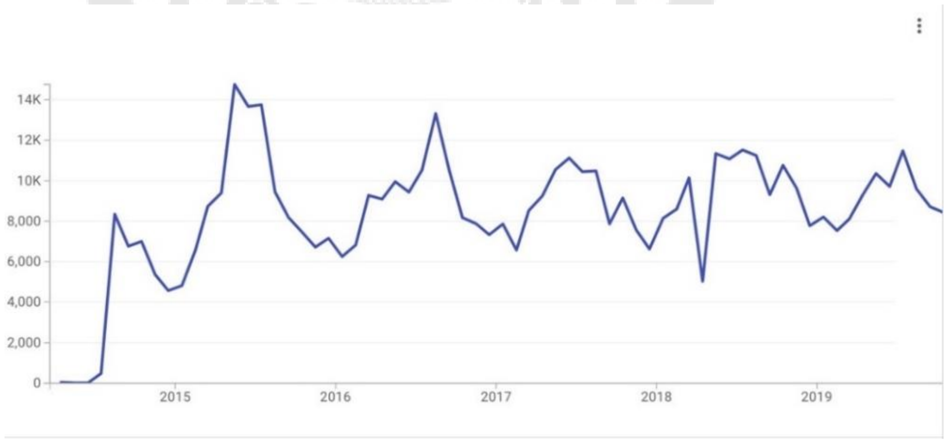


Figure 6 Monthly call trend of the Citizen Service Request (CSR) dataset from 2014 – 2021

Table 2 shows 856,405 rows representing 23 data points from 2015 to 2022. The data is updated daily and is able to filtered by date, location, status, and service request type. Figure 6 depicts the monthly trend to demonstrate the fluctuation. The dataset presents useful insights into the types of issues that Cincinnati citizens face, as well as the level of service offered by the CSR call center.

Table 2 Description of the Citizen Service Request (CSR)

Column Name	Description	Filters
AGENTDISPID	The disposition of the call and how it was completed when an agent has completed their call	-
CALLSTARTDT	Date and time when the incoming call occurred	-
CALLACTIONID	A code for internal actions that occurred as the call completed	-
CALLACTIONREASONID	A code providing additional details for the CallActionId for the internal actions that occurred	-
CALLID	Unique to a leg of the call. This along with SeqNum together form a primary Key.	-
CALLTYPEID	The code for if a call is incoming or outgoing. 1 = and incoming regular call and 2 = outbound virtual call	
CONNCLEARDT	The date and time when the call was completed	-
DNIS	The number that the caller dialed when it was picked up at the DPS call center	-
QUEUEENDDT	The date and time when the caller entered the queue (i.e. the call started)	-
SEQNUM	The unique ID for each incoming call.	-

Table 2 (Continued)

Column Name	Description	Filters
SERVICE_ID	The service id code identifies the recipient of the call. 106 is an incoming DPS call with Call Type = Regular Call. 107 is an incoming DPS call with Call Type = Virtual Call.	-
STATION	The physical station that the representative answered the call.	-
WORKGROUP_ID	Internal designations for what group the taker of a call belongs to. All DPS calls will be code 19.	-
WRAPENDDT	The date and time stamp when the representative has completed their after-call work.	-
QUEUESTARTDT	The date and time when the caller entered the queue (i.e. the caller is on hold)	-
PREVIEWENDDT	The date and time stamp of the preview getting accepted by the call representative.	-
PREVIEWSTARTDT	The date and time stamp of the preview appearing on the call representative's computer.	-
ANSWERDT	If Call Type = Virtual Call, then this field is the date/time stamp of when the customer answered the return call from DPS. Otherwise this field will be null.	-
ANSWER_SPEED_SECS	The total time in seconds it took for the call representative to answer the phone. This field is only applicable for regular calls.	-
TALK_TIME_SECS	The total amount of time in seconds the call lasted.	IF Call Type = Regular Call THEN QueueEndDt - QueueStartDt, ELSEIF Call Type = Virtual Call THEN ConnClearDt - Answerdt
WRAP_TIME_SECS	Wrap time is the time spent by a representative doing after call work once a call has been completed. This field is the total time in seconds that a representative did after call work.	WrapEndDt - ConnClearDt

Table 2 (Continued)

Column Name	Description	Filters
SERVICE_LEVEL	This Y/N field identifies if the call representative answered the phone call in 93 seconds or less. The 3 seconds accounts for any recorded messages that are given before the call enters the queue. DPS has a self-chosen key performance indicator that 90% of calls should be answered within 90 seconds.	DPS has a self-chosen key performance indicator that 90% of calls should be answered within 90 seconds.
ABANDONED	An abandoned call is a call that is ended before any conversation has occurred. This field is a Y/N attribute on if an incoming call to the DPS call center was abandoned.	This is found using the calculation: IF USER_ID is NULL AND CallActionId = 5 AND AnswerSpeed <> 0 THEN 'Y' ELSE 'N'.
CALL_TYPE	There are two possible incoming call types -regular call and virtual call. When an individual call into the DPS call center, they are given an option to wait on hold for a representative or they can choose to have a call center representative call them back on their turn in the queue. A regular call is classified as a call where the individual waited on hold while a virtual call is classified as a call where the individual chose to have the call center call them back.	While a virtual call is technically an outbound call, the data is reflecting information from an original incoming call that was returned and therefore counts as an incoming call.
UNIQUE_ID	SeqNum + Callid	
ANSWERED	This attribute shows whether the call was answered. If the call = "Not Answered" that means there was a technical error and the call was dropped before a representative could answer the phone.	

3.2 Exploratory data analysis and visualization

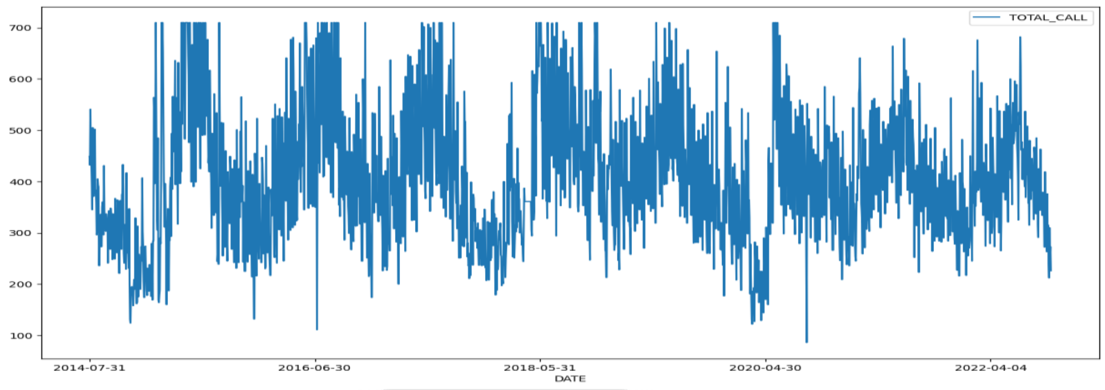


Figure 7 Total call per day visualization

From 2014 to 2022, the department's call arrivals follow a yearly pattern. Every July or Summer, the volume was higher than in other months. The number of calls ranged from 86 to 710, with an overall average of 403. Missing values are linearly interpolated in 2018. In early 2020, there was a drop trend, which could have been caused by a factor affecting call arrivals at the time.

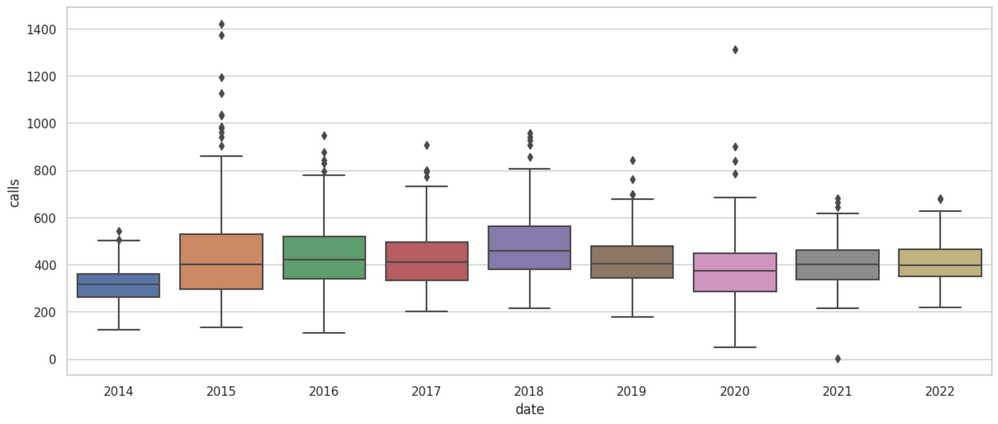


Figure 8 Boxplot of total call per year before outliers handling

The total calls in summer 2015, the second year of service, are highly fluctuated from the original values of over 1,400 calls. Figure 8 shows significant outlier calls from 2015 to 2020, which can be handled by setting minimum and maximum bounds, as well as zero values, which are repeated every weekend and holiday. **Figure 9** depicts the results of the histogram of total daily calls.

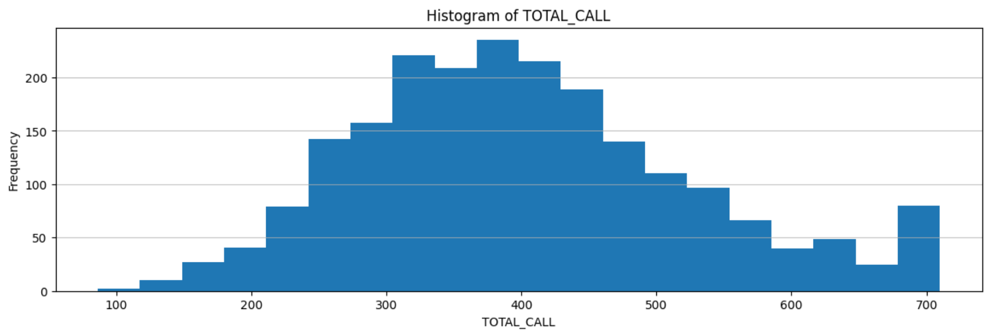


Figure 9 Total daily call histogram after outliers handling

3.3 Data Preparation and Feature Engineering

Accurate and efficient time series model development involves stages such as data preparation and feature engineering. Because they set the stage for the models' success, these actions are crucial.

The process of preparing data for analysis entails cleaning and preprocessing it. Duplicate and missing value removal, handling outliers, and smoothing or altering the data as needed are all included in this. Because utilizing data during training can produce erroneous findings, preparation of the data is crucial.

The process of choosing and creating features that will be used to train the model is known as feature engineering. Finding patterns and trends in the data, as well as choosing relevant time series components including trend, seasonality, and cyclical patterns, are all part of this process. It also entails extracting features from sources that could have an impact on the target variable. Feature engineering plays a critical role in developing a model that enables precise future value prediction based on historical data.

- **Data Cleaning**: Model accuracy can be harmed by missing values, outliers, and other anomalies in time series data. As a result, cleaning the data before feeding it to the model is critical. Data smoothing, outlier identification and removal, and imputation of missing values are a few popular methods for data cleaning.

- **Data Normalization**: To compare and combine time series effectively it is often necessary to normalize the data. This involves converting the data into a scale such, as min-max scaling or z score normalization.
- **Time Feature**: retrieving data on the day of the week, month, year, etc. Timestamps on data can be used to find trends and seasonality-related patterns in the data.
- **Lag Features**: are model features that are derived from prior time series observations. For example, we can utilize the values at times t-1, t-2, t-3, and so on as features if we wish to forecast the time series value at time t.
- **Rolling Statistics**: Rolling statistics are all about computing summary statistics, such mean or standard deviation, over a sliding window of observations. This can help to capture data seasonality and trends.
- **Fourier Transform**: By decomposing a time series into its frequency components the Fourier Transform enables us to recognize cyclic patterns in data and eliminate any unwanted noise.
- **Wavelet Transform**: An additional technique for dissecting a time series into its basic frequency components is the wavelet transform. It works well when examining non-stationary time series that show variations in variance or mean over time.
- **Recurrent Neural Networks (RNNs):** specifically built to handle sequential data, such as time series, are called RNNs.. Their ability to grasp relationships between past and future values makes them valuable, for both data preparation and feature engineering purposes.

3.4 Data Preprocessing

Data preprocessing is a stage in getting a time series data set ready for modeling. In time series modeling, the objective of data preprocessing is to produce a valuable, well-organized data collection that can be utilized to improve forecasting models that are exact and accurate. The following are a few typical techniques for preparing data used in time series prediction:

- Resampling: The time series data needs to be converted to a certain time period, like weekly, monthly, annual, or daily. Data analysis can be made easier and noise in the data can be reduced via resampling.
- Encoding: This involves translating category variables into numerical values that the model can use. One-hot encoding and label encoding are two popular encoding methods.
- Dimensionality Reduction: This entails maintaining the most important features in the dataset while minimizing the number of features. Two methods for lowering dimensionality are Singular Value Decomposition (SVD) and Principal Component Analysis (PCA).
- Missing Values: This covers handling missing data. Imputation and deletion are two common methods.

The ARIMA and SARIMA models for time series forecasting require extra data processing processes in addition to cleaning, preprocessing, and time series decomposition. The method includes checking for stationarity to make sure the data is stationary, differencing the data if needed, evaluating the ACF and PACF diagrams for the ARIMA model's order, fitting the SARIMA or ARIMA model to the data, and utilizing diagnosis to test the model's performance.

Unlike other forecasting methods, deep learning models require specific data preparation. This involves formatting the data for the model, splitting it into training and testing sets, and then designing and training a deep learning model with appropriate parameters. Finally, the model's performance is evaluated on the testing data. Notably, the specific data processing steps and techniques used will vary depending on the individual forecasting problem and the chosen model.

3.5 ACF and PACF

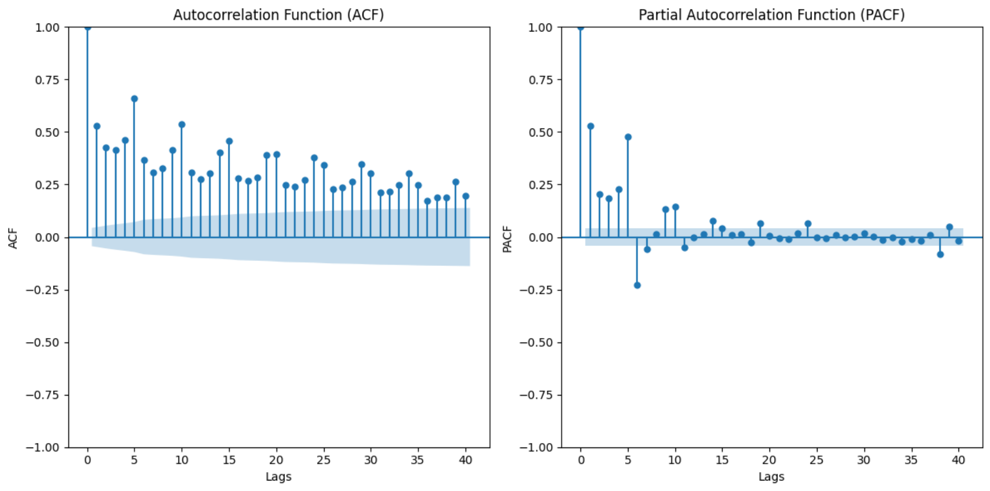


Figure 10 Example of ACF and PACF plot

Identifying the ideal ARIMA model for time series forecasting involves analyzing two key graphs: ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) in **Figure 10**. ACF reveals the correlation between a time series and its past values, while PACF shows the correlation after considering past correlations. Significant patterns in these graphs help identify the order of the ARIMA model:

- Sharp initial spike followed by gradual decline in ACF: Indicates the need for differencing of order k.
- PACF shows a spike at lag k and then no more spikes, indicating an AR model of order k.
- Gradual decline in both ACF and PACF: Points towards an ARMA model.
- Spikes at the same lag in both graphs: Suggests a seasonal model like SARIMA.

Interpreting ACF and PACF requires expertise and involves considering factors beyond just the graphs, such as domain knowledge, data availability, and model fit, to ultimately choose the best ARIMA model order.

3.6 Feature Importance

The unveiling the most impactful features in a time series is crucial to understanding its underlying forces and making informed decisions. This process, known as feature importance, helps identify and rank the variables that contribute most significantly to the target variable's variation over time. Here are some key methods:

3.6.1 Correlation Analysis: Throughout the time series, investigate the relationships between other features and the target feature. More influence over the result is indicated by a stronger correlation.

3.6.2 Lag Analysis: Analyze how each feature and the target variable relate to one another over various time periods. This can show important temporal trends and unearth delayed effects.

3.6.3 Decomposition: Break down the time series into its constituent parts, including seasonality, trend, and residual noise. Examine how each feature contributes to these elements to learn about its relationship to particular patterns.

3.6.4 Feature Importance Algorithms: Use algorithms for machine learning such as gradient boosting models, random forests, and decision trees. With regard to estimating the target variable, these algorithms automatically evaluate each feature's value, offering insightful information on how important it is.

A measure known as "Gain," reflecting the contribution of each feature to the model's performance, is employed by XGBoost to evaluate feature significance in the context of time series forecasting. The Gain represents the increase in model performance resulting from splits based on a specific feature and is computed for each feature during the construction of decision trees within the ensemble. The accumulated gains across all trees provide an overall assessment of each feature's influence. Features with higher aggregated gains are considered more influential in determining the model's performance. XGBoost enables users to examine these scores, facilitating the identification of crucial features in the forecasting model. This feature importance analysis proves valuable for tasks like feature selection, model interpretation, and comprehending the factors impacting the model's predictions. Users can customize parameters to tailor feature importance estimates to specific evaluation metrics or importance types.

3.6.5 Recursive Feature Elimination (RFE): RFE is an effective strategy that gradually eliminates the least important elements from a model until the target number is retained. A more concentrated and effective study is made possible by this approach, which aids in identifying the most significant aspects in a time series.

3.6.6 Domain Knowledge: Integrating domain knowledge plays a key role in interpreting the significance of specific features. Experts familiar with the specific domain can offer valuable insights into the real-world significance of various variables, enriching the understanding of the time series data.

3.7 Machine Learning

The data flow in this study is depicted in **Figure 12** and describes as follows;

- Pre-processing involved transforming raw data from 856,430 calls into a daily format. The series was refined by trimming both the initial and concluding segments to eliminate outliers. The resulting dataset spans from July 31, 2014, to October 14, 2022, covering a total of 2,135 days, with holidays excluded as depicted in Figure 12 of the dataframe.
- Daily summaries of call data are generated by calculating the total number of calls, abandoned calls, and answered calls, as well as the average service level. Additionally, average values for relevant metrics like answer speed, talk time, and wrap time are computed. Finally, abandonment and answer rates are derived from the respective call counts.

- Data for weekdays (Monday to Friday) is used for this analysis, as days with no calls (holidays and days off) are excluded.

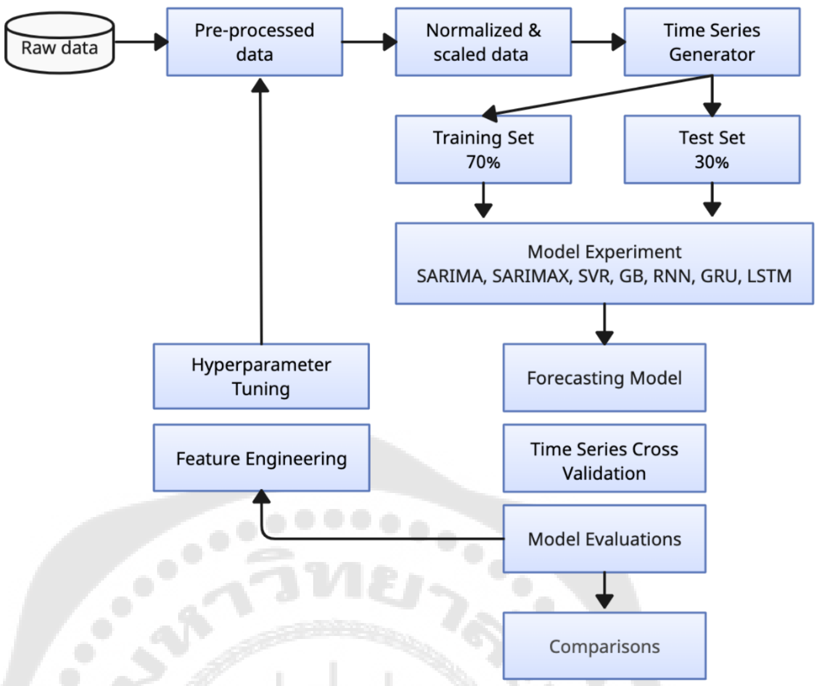
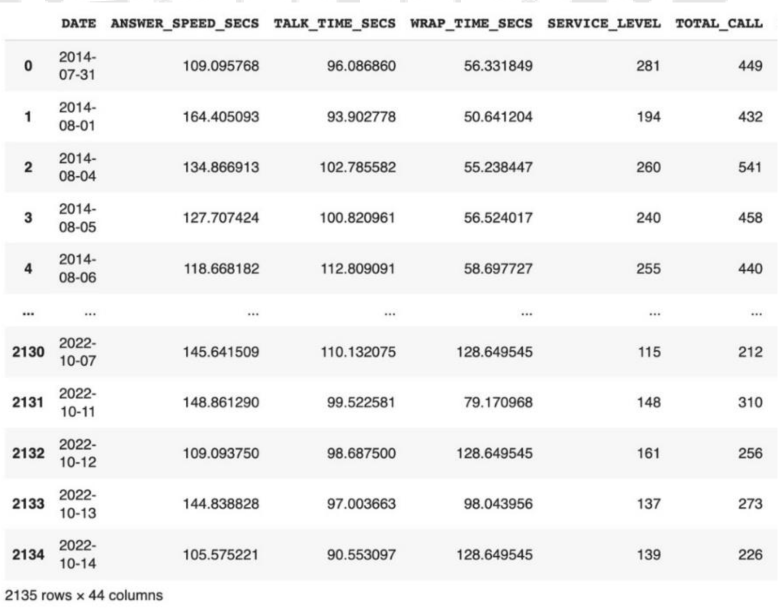


Figure 11 The data flow diagram of the experiment.



	DATE	ANSWER_SPEED_SECS	TALK_TIME_SECS	WRAP_TIME_SECS	SERVICE_LEVEL	TOTAL_CALL
0	2014-07-31	109.095768	96.086860	56.331849	281	449
1	2014-08-01	164.405093	93.902778	50.641204	194	432
2	2014-08-04	134.866913	102.785582	55.238447	260	541
3	2014-08-05	127.707424	100.820961	56.524017	240	458
4	2014-08-06	118.668182	112.809091	58.697727	255	440
...	...	...	...	...	...	...
2130	2022-10-07	145.641509	110.132075	128.649545	115	212
2131	2022-10-11	148.861290	99.522581	79.170968	148	310
2132	2022-10-12	109.093750	98.687500	128.649545	161	256
2133	2022-10-13	144.838828	97.003663	98.043956	137	273
2134	2022-10-14	105.575221	90.553097	128.649545	139	226

2135 rows × 44 columns

Figure 12 Preprocessed dataframe.

- Normalize or Standardize: Ensure all data shares the same scale for consistent analysis.
- Leverage Time Series Generator: Utilize a dedicated time series generator tool to efficiently split the data into training (70%) and testing (30%) sets.
- Employ the 30% testing set to train and modify the data for further analysis across different algorithms.
- Time Series Cross-Validation: Implement time series cross-validation with three folds (n_splits = 3) to assess model performance on various data splits.
- Performance Evaluation: Utilize diverse metrics such as MAE, RMSE, MAPE, R^2 to thoroughly evaluate the performance of different models.
- Continuous Improvement: Iterate through feature engineering and hyperparameter tuning to optimize model performance and uncover better options.

3.7.1 Training set and Test set

The dataset is split into two sets: training (70%) and testing (30%). The training set, encompassing 1,708 days, is used to build a prediction model, while the test set, consisting of 427 days, serves to evaluate the model's performance.

3.7.2 Data decomposition

Data decomposition, a cornerstone of data analysis, delves into the heart of a dataset by meticulously separating it into its fundamental components: trend, seasonality, and residual noise. This process unveils hidden patterns and grants us a deeper understanding of the data's temporal variations. Analysts can gain valuable insights by isolating distinct elements, leading to more accurate modeling and predictions. The original data is represented by the top graph in **Figure 13**. In the second graph, we can see a downward trend. The pattern did not clearly show seasonality.

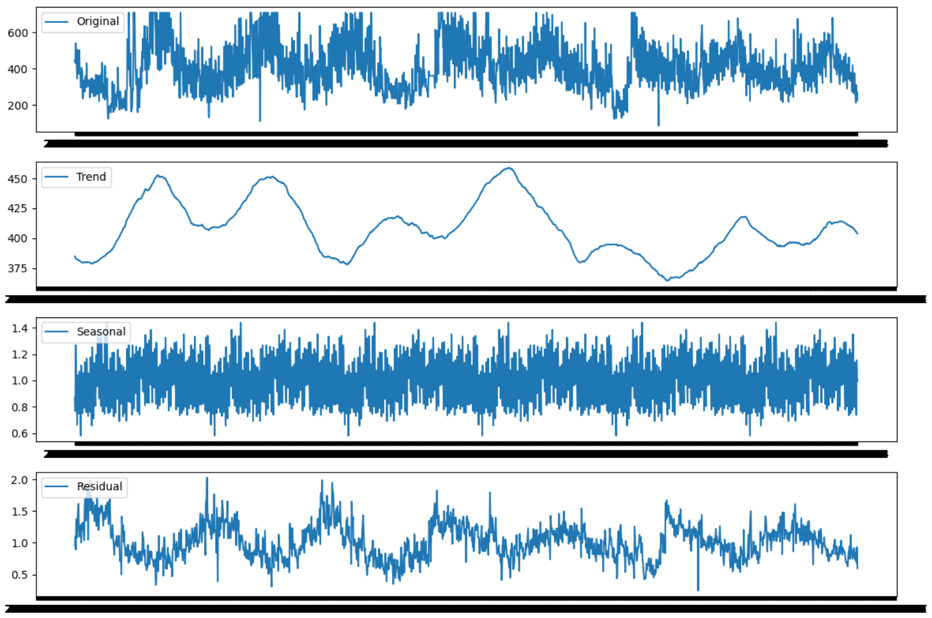


Figure 13 Yearly (365 days) decomposition

3.8 Feature Engineering

This research leverages time and lag features to grant the model the power to discern the intricate relationships between TOTAL_CALL and other features, ultimately revealing the key factors that contribute to accurate prediction. These features capture the essential temporal patterns and dependencies, significantly enhancing the model's grasp of the data and its predictive capabilities.

3.8.1 Time Features

To improve the model's forecasting capabilities, particularly for time series data like call volume or service performance, various time features are incorporated into the dataset. These features capture specific temporal aspects of each data entry, enabling the model to learn and leverage temporal patterns for improved prediction.

- **Temporal Features**: DAY_OF_WEEK, MONTH_DAY, YEAR_DAY, WEEK_YEAR : These features directly extracted from the 'DATE' column provide contextual information about the day of the week, month, year, and week of the year for each entry.

- **Seasonal Features:** Binary indicators represent the seasons (FALL, SPRING, SUMMER, WINTER). Each feature is assigned 1 if the entry falls within the corresponding season and 0 otherwise.
- **Monthly Features**: Similarly, binary features represent each month of the year (JAN, FEB, ..., DEC). An entry receives 1 for its respective month and 0 for other months.
- **Yearly Features:** YEAR_2014, YEAR_2015, ..., YEAR_2022: features differentiate the specific year of each entry (YEAR_2014, YEAR_2015, ..., YEAR_2022). The feature corresponding to the entry's year is set to 1 and others to 0.
- **Day of the Week Features:** Each day of the week is represented by a binary feature (MONDAY, TUESDAY, ..., SUNDAY). The feature relevant to the entry's day is assigned 1 and the rest 0.

3.8.2 Lag Features

By identifying temporal connections and past patterns in the data, lag characteristics are essential to time series research, such as anticipating total call volumes. The goal variable ('TOTAL_CALL') at different time intervals is represented by these properties, which enable the model to detect trends, seasonality, and other temporal patterns that are essential for precise forecasting.

- Lag1 to Lag5: These features capture 5 days of seasonality (Monday to Friday) and aim to learn any potential weekly patterns affecting call volume.
- Lag10 to Lag30: This range of lag features (2 to 6 weeks) investigates the influence of longer-term historical data on future call volumes.
- TOTAL_CALL_LAG_1 to TOTAL_CALL_LAG_30: These features represent the specific lag observations created for 'TOTAL_CALL', providing the model with historical information at different timeframes.

Figure 14 visually depicts all features used in the analysis, offering a comprehensive overview of the data utilized for model training. These lag features, combined with other temporal features, create a complete dataset tailored to each model, ultimately enhancing its ability to learn complex temporal relationships and generate accurate predictions for future call volumes.

3.9 Feature Selection

Finding the most informative features that most accurately represent the target variable is the goal of feature selection. Through the removal of superfluous and noisy information, the model is improved and more accurate predictions are produced. Effective feature selection is especially important for lowering total data complexity and avoiding models that require a lot of processing power. Statistical time series models lack the ability to manage feature selection on their own, but other algorithms, such Artificial Neural Networks, do. As part of the preprocessing stage, a rudimentary feature selection mechanism has already been put in place. Curse of Dimensionality: When datasets have too many sparse features, a phenomenon known as the Curse of Dimensionality is mitigated by dropping one of the created dummy features after one-hot encoding Category variables.

3.9.1 Correlation Analysis

The link between each external feature and the target variable is examined using correlation analysis in order to choose the feature. More significant features can be those having a greater correlation. Fifteen qualities are linked to TOTAL_CALL. For 15 features, correlated features with a value greater than 0.2 are selected in **Figure 15**.

3.9.2 Tree-Based Models

Gini Importance, also known as Mean Gini Decrease, quantifies the overall impact of a feature on the purity of all nodes across an ensemble of decision trees. This provides valuable insights into the relative importance of each feature for classification.

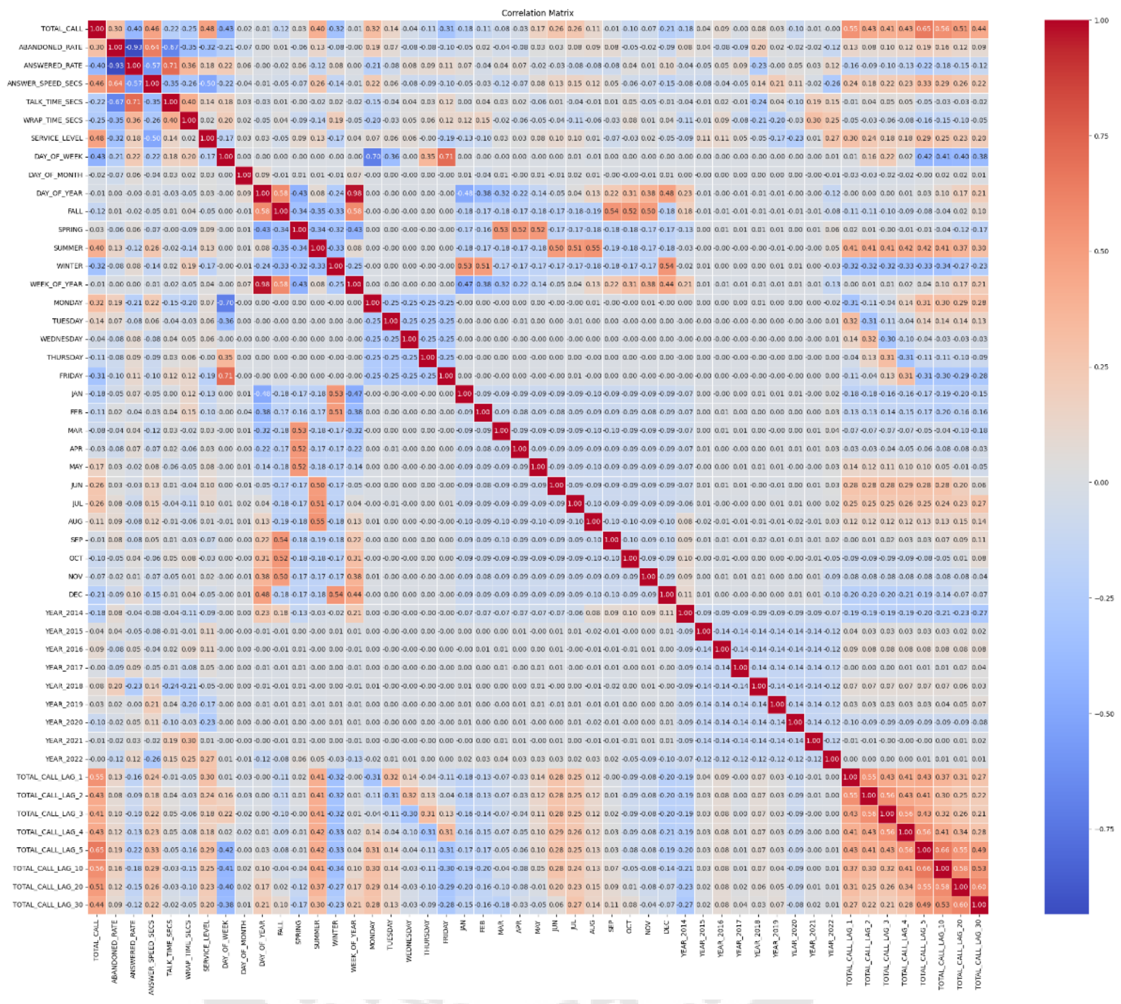
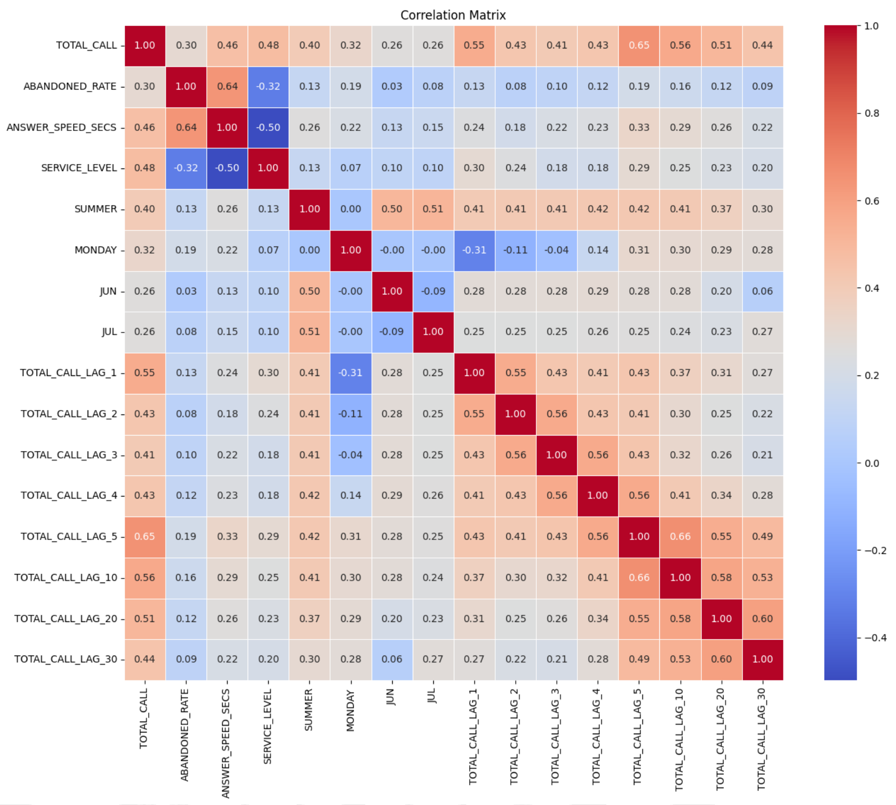


Figure 14 Correlation of total 48 features



	TOTAL_CALL	ABANDONED_RATE	ANSWER_SPEED_SECS	SERVICE_LEVEL	SUMMER	MONDAY	JUN	JUL	TOTAL_CALL_LAG_1	TOTAL_CALL_LAG_2	TOTAL_CALL_LAG_3	TOTAL_CALL_LAG_4	TOTAL_CALL_LAG_5	TOTAL_CALL_LAG_10	TOTAL_CALL_LAG_20	TOTAL_CALL_LAG_30
TOTAL_CALL	1.00	0.30	0.46	0.48	0.40	0.32	0.26	0.26	0.55	0.43	0.41	0.43	0.65	0.56	0.51	0.44
ABANDONED_RATE	0.30	1.00	0.64	-0.32	0.13	0.19	0.03	0.08	0.13	0.08	0.10	0.12	0.19	0.16	0.12	0.09
ANSWER_SPEED_SECS	0.46	0.64	1.00	-0.50	0.26	0.22	0.13	0.15	0.24	0.18	0.22	0.23	0.33	0.29	0.26	0.22
SERVICE_LEVEL	0.48	-0.32	-0.50	1.00	0.13	0.07	0.10	0.10	0.30	0.24	0.18	0.18	0.29	0.25	0.23	0.20
SUMMER	0.40	0.13	0.26	0.13	1.00	0.00	0.50	0.51	0.41	0.41	0.41	0.42	0.42	0.41	0.37	0.30
MONDAY	0.32	0.19	0.22	0.07	0.00	1.00	-0.00	-0.00	-0.31	-0.11	-0.04	0.14	0.31	0.30	0.29	0.28
JUN	0.26	0.03	0.13	0.10	0.50	-0.00	1.00	-0.09	0.28	0.28	0.28	0.29	0.28	0.28	0.20	0.06
JUL	0.26	0.08	0.15	0.10	0.51	-0.00	-0.09	1.00	0.25	0.25	0.25	0.26	0.25	0.24	0.23	0.27
TOTAL_CALL_LAG_1	0.55	0.13	0.24	0.30	0.41	-0.31	0.28	0.25	1.00	0.55	0.43	0.41	0.43	0.37	0.31	0.27
TOTAL_CALL_LAG_2	0.43	0.08	0.18	0.24	0.41	-0.11	0.28	0.25	0.55	1.00	0.56	0.43	0.41	0.30	0.25	0.22
TOTAL_CALL_LAG_3	0.41	0.10	0.22	0.18	0.41	-0.04	0.28	0.25	0.43	0.56	1.00	0.56	0.43	0.32	0.26	0.21
TOTAL_CALL_LAG_4	0.43	0.12	0.23	0.18	0.42	0.14	0.29	0.26	0.41	0.43	0.56	1.00	0.56	0.41	0.34	0.28
TOTAL_CALL_LAG_5	0.65	0.19	0.33	0.29	0.42	0.31	0.28	0.25	0.43	0.41	0.43	0.56	1.00	0.66	0.55	0.49
TOTAL_CALL_LAG_10	0.56	0.16	0.29	0.25	0.41	0.30	0.28	0.24	0.37	0.30	0.32	0.41	0.66	1.00	0.58	0.53
TOTAL_CALL_LAG_20	0.51	0.12	0.26	0.23	0.37	0.29	0.20	0.23	0.31	0.25	0.26	0.34	0.55	0.58	1.00	0.60
TOTAL_CALL_LAG_30	0.44	0.09	0.22	0.20	0.30	0.28	0.06	0.27	0.27	0.22	0.21	0.28	0.49	0.53	0.60	1.00

Figure 15 Correlated features selection

CHAPTER 4
EXPERIMENTAL RESULTS

We will present the experimental results of the time series forecasting models discussed in Chapter 3 in this chapter. We will compare the performance of the various models on the dataset and discuss each model's studies.

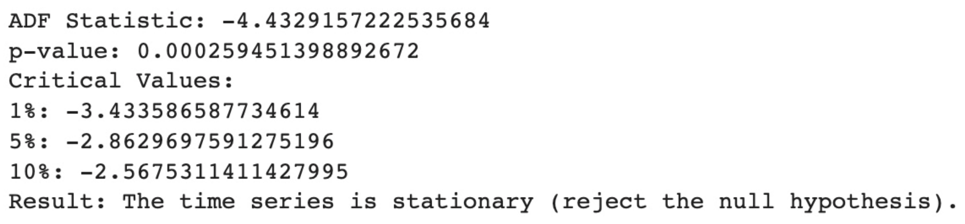
4.1 Statistical Test

In time series analysis, uncovering a dataset's inherent stationarity is crucial. Two key tools, the Augmented Dickey-Fuller (ADF) test and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test, play a vital role in this process.

The Augmented Dickey-Fuller (ADF) : This test investigates at the presence of a "unit root," a sign of non-stationarity, in a time series. Potential stationarity is suggested by a rejected null hypothesis of non-stationarity, highlighting the significance of differencing in achieving stationarity.

KPSS Test (Kwiatkowski-Phillips-Schmidt-Shin Test) : investigates the null hypothesis of stationarity around a deterministic trend, building on the findings of the ADF test. This reveals possible long-term trends or structural fractures within the data by assisting in the determination of whether a series is stationary around a trend.

The ADF and KPSS tests complement one another. The ADF test is especially useful for determining the need for differencing, whereas the KPSS test is more concerned with detecting stationarity around a trend. Analysts can make informed decisions about appropriate transformations and modes thanks to their combined application.

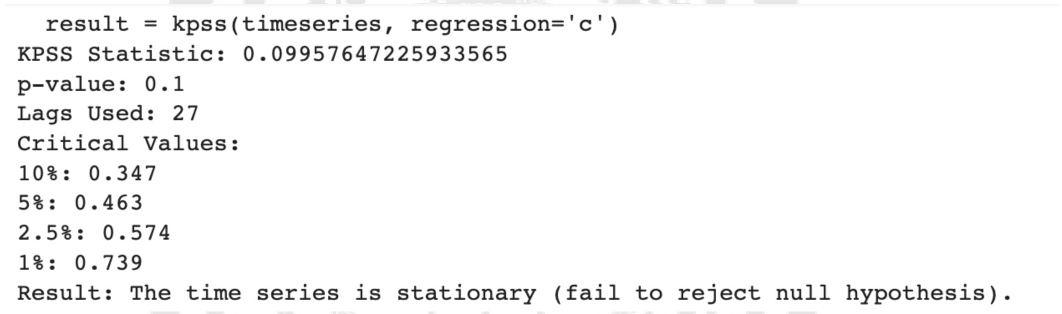


```
ADF Statistic: -4.4329157222535684
p-value: 0.000259451398892672
Critical Values:
1%: -3.433586587734614
5%: -2.8629697591275196
10%: -2.5675311411427995
Result: The time series is stationary (reject the null hypothesis).
```

Figure 16 Augmented Dickey-Fuller test results.

There is substantial evidence for stationarity in **Figure 16**, with an ADF Statistic of -4.43 and a p-value of 0.00026. Critical values of -3.43 (1%), -2.86 (5%), and -2.57 (10%) at various significance levels further corroborate this. This strongly rejects the null hypothesis of non-stationarity because the p-value is extremely low and the ADF Statistic is substantially smaller than these critical values.

As a result, the test's definitive result verifies that the time series is stationary. This suggests that stationarity was probably achieved without the need for differencing, which is important information for further study and modeling. By using this knowledge, models that are more precise and efficient can be created, improving time series data predictions and understanding.



```
  result = kpss(timeseries, regression='c')
KPSS Statistic: 0.09957647225933565
p-value: 0.1
Lags Used: 27
Critical Values:
10%: 0.347
5%: 0.463
2.5%: 0.574
1%: 0.739
Result: The time series is stationary (fail to reject null hypothesis).
```

Figure 17 KPSS test results

The KPSS test employed here yielded a statistic of approximately 0.10, a p-value of 0.1, and 27 lags. Notably, the critical values for different significance levels indicate a range of acceptable values: 0.347 (10%), 0.463 (5%), 0.574 (2.5%), and 0.739 (1%).

Given that the p-value is quite large (0.1) and the KPSS Statistic is below all of these key values, the null hypothesis of stationarity around a deterministic trend is not rejected. As a result, the test result supports the first hypothesis by confirming the stationarity of the time series. This result suggests that the time series shows signs of stationarity and may possibly point to an underlying trend. This important realization facilitates further research and model development by revealing details about the structural stability of the data and opening up a deeper knowledge of its long-term behavior.

4.2 SARIMAX

4.2.1 ACF and PACF results

Through the examination of the ACF and PACF plots, analysts can obtain significant knowledge for determining the proper AR and MA orders (incorporating seasonal and non-seasonal elements) as well as pertinent exogenous variables that enhance the SARIMAX model's prediction ability.

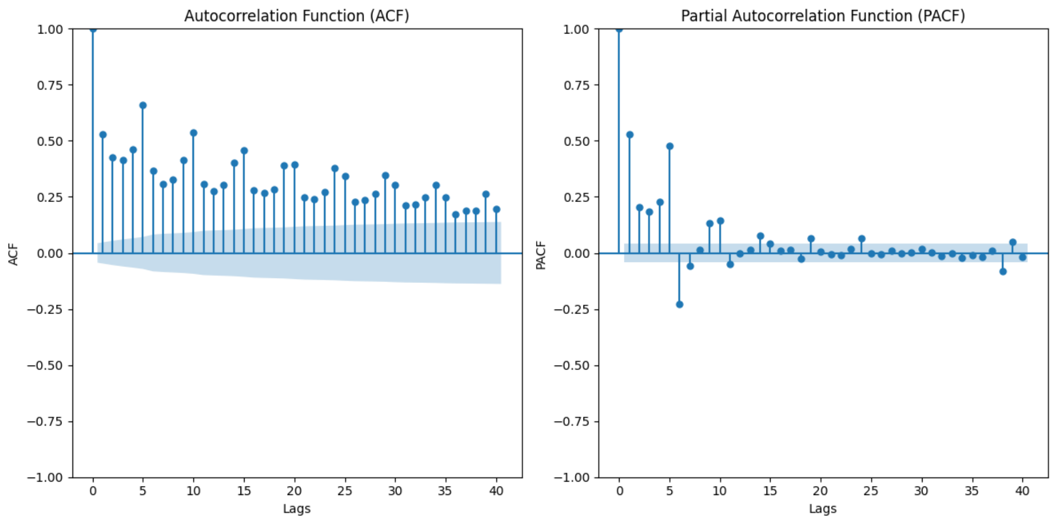


Figure 18 ACF and PACF of original data

Figure 18 reveals crucial information about the underlying patterns and trends within our time series data. Notably, the ACF and PACF plots exhibit a "long memory" effect, meaning the autocorrelation values at successive lags decay slower than expected for a purely random process. This suggests that first-order differencing is necessary to remove the trend-related structures and achieve stationarity. Furthermore, the presence of prominent spikes at every fifth lag in both the ACF and PACF plots signifies a clear seasonal pattern with an order of 5. Identifying this periodicity is crucial for selecting suitable parameters in time series models, particularly SARIMA. This valuable insight sheds light on the data's temporal characteristics, paving the way for future modeling and analysis efforts to be tailored accordingly.

Following the application of first-order differencing and a seasonal difference of order 5, the ACF and PACF plots in **Figure 19** offer further insights into potential model parameters. The ACF plot specifically reveals autocorrelation potentially present at the first or second lag. This suggests that AR(1, 2, or 4) and MA(1, 2, or 4) models might be appropriate choices for further investigation.

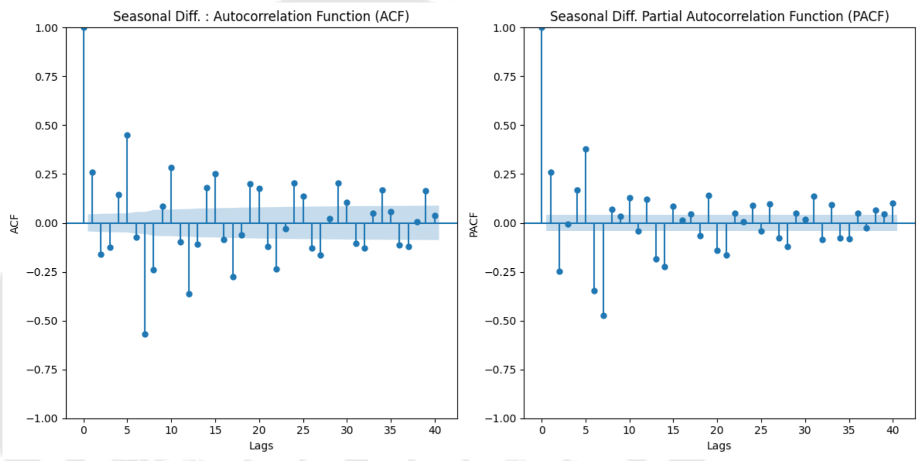


Figure 19 ACF and PACF of 1st differenced data

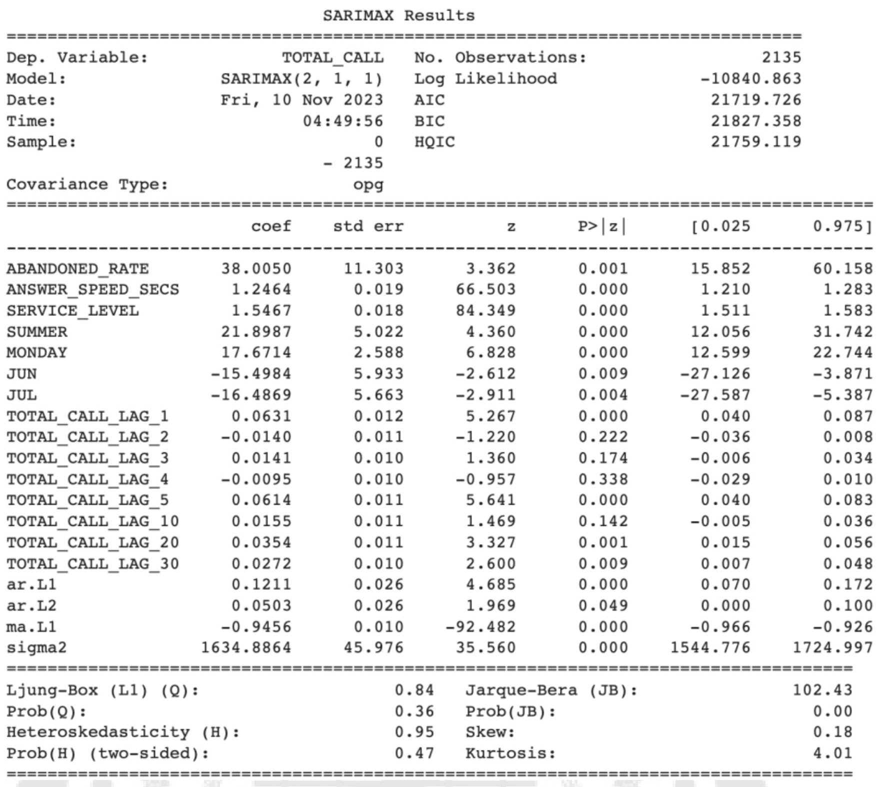
4.2.2 Model Identification and Diagnostics

The Akaike Information Criterion (AIC) is a statistical measure used for model selection, comparing the goodness of fit of different models. In the context of SARIMAX modeling, the AIC is particularly valuable for choosing the model parameters.

To optimize the forecasting model's parameters, a targeted randomized approach will be implemented. This strategy involves systematically evaluating 50 different combinations of AR and MA order settings within a limited range. This efficient approach aims to expedite the experimentation process while still comprehensively exploring a relevant parameter space. By focusing on the combinations with the lowest AIC (Akaike Information Criterion), the optimal configuration for accurate time series forecasting will be identified. This adaptable strategy balances efficiency with effectiveness, ultimately aiming to maximize the model's predictive performance.

SARIMAX(2, 1, 1) x (0, 0, 0, 5), The study employed a SARIMAX(2, 1, 1) x (0, 0, 0, 5) model to predict the TOTAL_CALL variable, incorporating both an exogenous variable and lagged values. Analyzing the results sheds light on the model's overall effectiveness and the individual impact of each feature. Several metrics like the log likelihood (-10840.863) and information criteria (including AIC) suggest a reasonable fit between the model and the observed data. Additionally, the coefficients associated with specific variables like ABANDONED_RATE, ANSWER_SPEED_SECS, and SERVICE_LEVEL provide valuable information about their predictive power for call volume. It's crucial to consider the statistical significance of these coefficients, indicated by their p-values. This assessment helps determine the reliability of each feature in explaining the variability observed in the TOTAL_CALL variable. By examining these various aspects, the study paints a comprehensive picture of the model's performance and offers valuable insights into the key factors influencing call volume.

Figure 20 delves into the world of the model's residuals, examining their characteristics through various diagnostic measures. The Ljung-Box test for serial correlation reveals no significant dependence between the residuals, as evidenced by its low p-values. However, a closer look at the skewness and kurtosis values indicates a slight rightward skew and heavier tails than a normal distribution. While these deviations are not alarming, they warrant further consideration and potential refinement to enhance the model's overall robustness.



```
                               SARIMAX Results
==============================================================================
Dep. Variable:              TOTAL_CALL   No. Observations:                 2135
Model:               SARIMAX(2, 1, 1)   Log Likelihood              -10840.863
Date:                Fri, 10 Nov 2023   AIC                          21719.726
Time:                        04:49:56   BIC                          21827.358
Sample:                             0   HQIC                         21759.119
                               - 2135
Covariance Type:                  opg
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
ABANDONED_RATE       38.0050     11.303      3.362      0.001      15.852      60.158
ANSWER_SPEED_SECS     1.2464      0.019     66.503      0.000       1.210       1.283
SERVICE_LEVEL         1.5467      0.018     84.349      0.000       1.511       1.583
SUMMER               21.8987      5.022      4.360      0.000      12.056      31.742
MONDAY               17.6714      2.588      6.828      0.000      12.599      22.744
JUN                 -15.4984      5.933     -2.612      0.009     -27.126      -3.871
JUL                 -16.4869      5.663     -2.911      0.004     -27.587      -5.387
TOTAL_CALL_LAG_1      0.0631      0.012      5.267      0.000       0.040       0.087
TOTAL_CALL_LAG_2     -0.0140      0.011     -1.220      0.222      -0.036       0.008
TOTAL_CALL_LAG_3      0.0141      0.010      1.360      0.174      -0.006       0.034
TOTAL_CALL_LAG_4     -0.0095      0.010     -0.957      0.338      -0.029       0.010
TOTAL_CALL_LAG_5      0.0614      0.011      5.641      0.000       0.040       0.083
TOTAL_CALL_LAG_10     0.0155      0.011      1.469      0.142      -0.005       0.036
TOTAL_CALL_LAG_20     0.0354      0.011      3.327      0.001       0.015       0.056
TOTAL_CALL_LAG_30     0.0272      0.010      2.600      0.009       0.007       0.048
ar.L1                 0.1211      0.026      4.685      0.000       0.070       0.172
ar.L2                 0.0503      0.026      1.969      0.049       0.000       0.100
ma.L1                -0.9456      0.010    -92.482      0.000      -0.966      -0.926
sigma2             1634.8864     45.976     35.560      0.000    1544.776    1724.997
===================================================================================
Ljung-Box (L1) (Q):                   0.84   Jarque-Bera (JB):               102.43
Prob(Q):                              0.36   Prob(JB):                         0.00
Heteroskedasticity (H):               0.95   Skew:                             0.18
Prob(H) (two-sided):                  0.47   Kurtosis:                         4.01
===================================================================================
```

Figure 20 Exogenous correlated features and model summary of SARIMAX (2,1,1) (0,0,0,5) results

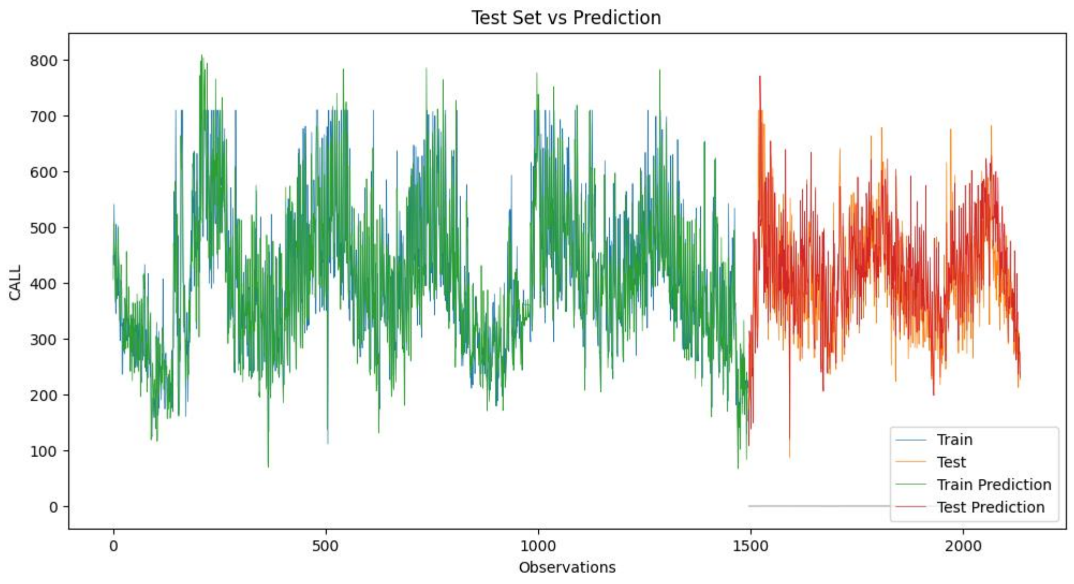


Figure 21 Correlated Features : Train and Test Actual vs Prediction plot of SARIMAX (2,1,1) (0,0,0,5) results

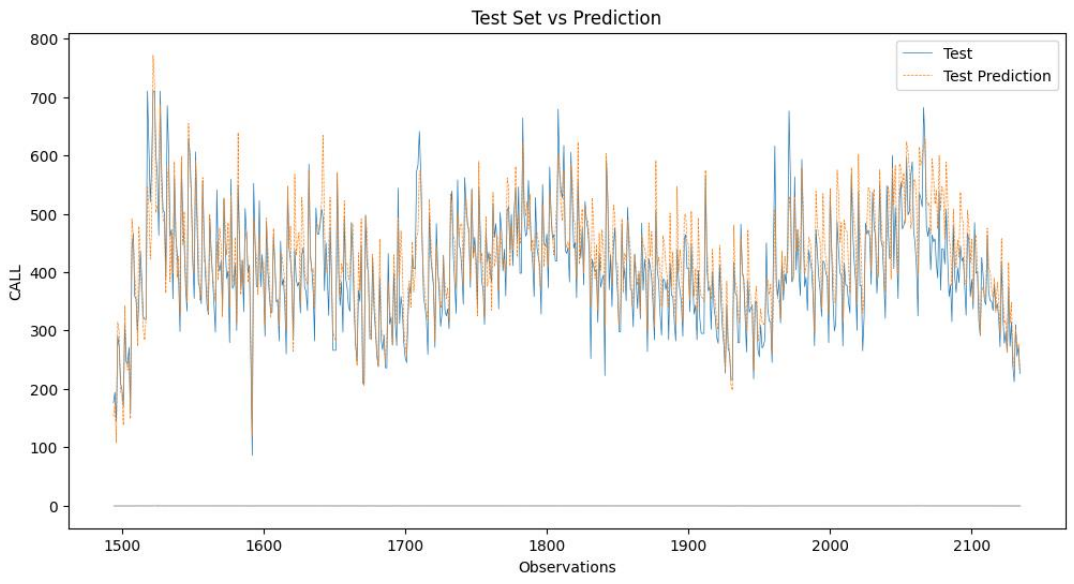


Figure 22 Correlated Features : Test Actual vs Prediction plot of SARIMAX (2,1,1) (0,0,0,5)

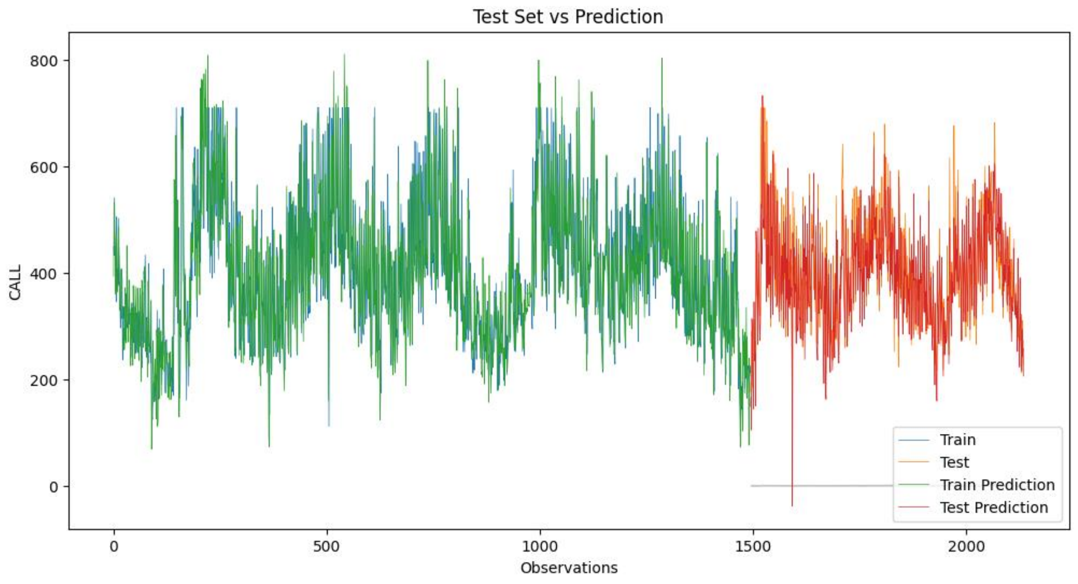


Figure 23 Full Features : Train and Test Actual vs Prediction plot of SARIMAX (2,1,1) (0,0,0,5) results

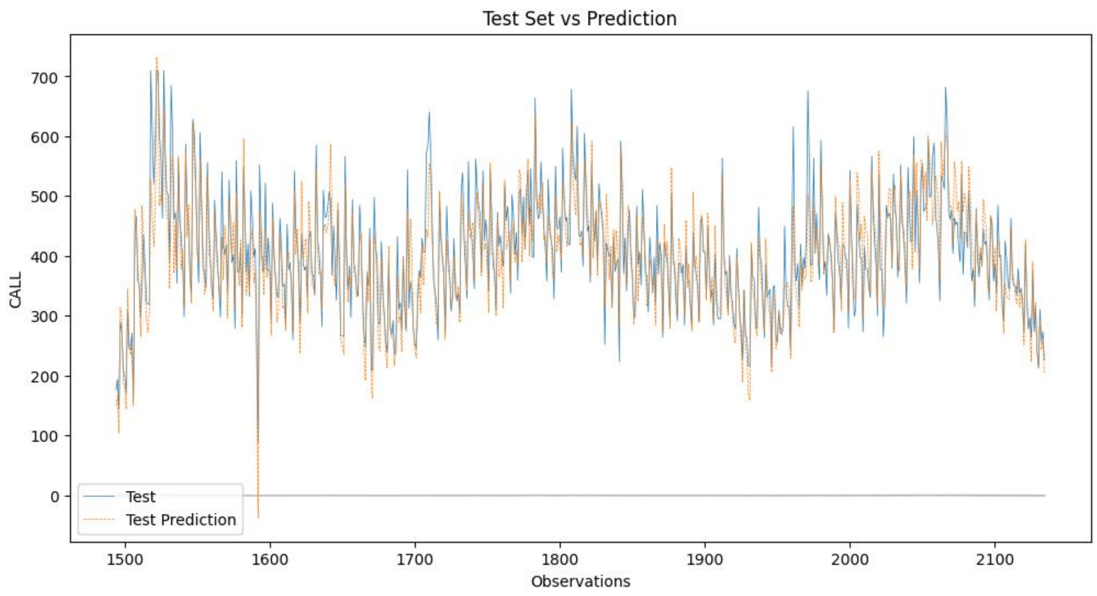


Figure 24 Full Features : Test Actual vs Prediction plot of SARIMAX (2,1,1)(0,0,0,5)

4.2.3 Feature Importance of SARIMAX

The analysis of correlated features identified several significant contributors to call volume. Higher abandonment rates (ABANDONED_RATE, x1), faster answer speeds (ANSWER_SPEED_SECS, x4), and improved service levels (SERVICE_LEVEL) all exhibited strong positive correlations with increased call volume, as reflected by their high coefficients and low p-values (**Figure 20**). Additionally, seasonal and temporal factors, such as summer seasonality (SUMMER), Mondays (MONDAY), and lagged call counts (TOTAL_CALL_LAG_1, TOTAL_CALL_LAG_5, TOTAL_CALL_LAG_20), also demonstrated high correlations and statistical significance, indicating their influence on call volume patterns.

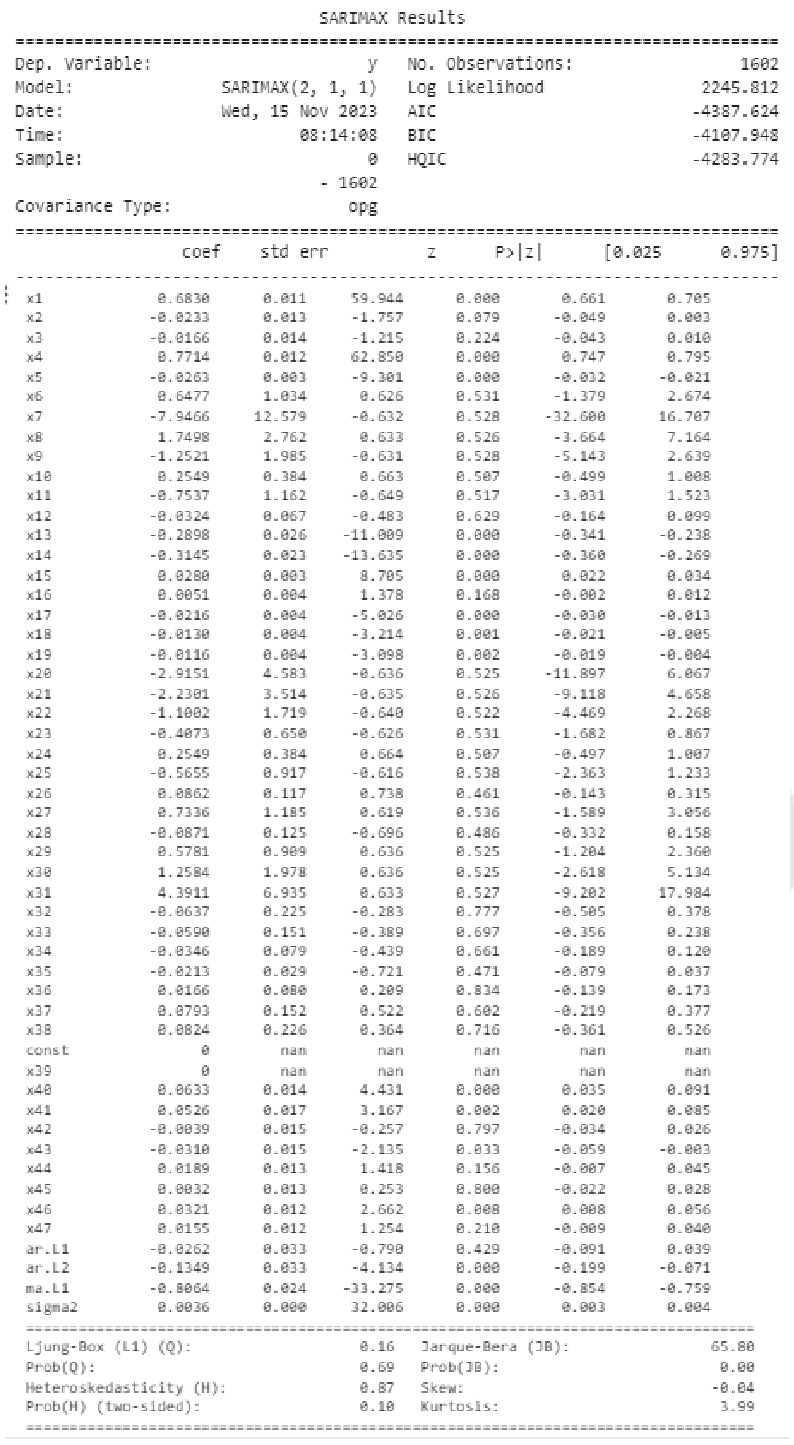
In the full dataset analysis, however, a slightly different set of features emerged as significant contributors. Among the features with high coefficients and statistical significance in **Figure 25** were ABANDONED_RATE (x1), ANSWER_SPEED_SECS (x4), TALK_TIME_SECS (x5), WINTER (x13), WEEK_OF_YEAR (x14), MONDAY (x15), and TOTAL_CALL_LAG_1 (x40). This suggests that certain characteristics, particularly those associated with call metrics, seasonality, and day of the week, consistently and significantly influence call volume across various analyses. These insights are crucial for interpreting the SARIMAX model's predictions and refining the model further to achieve optimal performance.

These specific features, namely 'ABANDONED_RATE', 'ANSWER_SPEED_SECS', 'SEVICE_LEVEL', 'MONDAY', and 'TOTAL_CALL_LAG_1', were chosen on purpose for further experimentation. The inclusion of 'SERVICE_LEVEL' in the selected features is crucial. Excluding it resulted in a notable decrease in overall model performance, with the R-Squared metric dropping by 22%. Examining the model's behavior with respect to this collection of five features—more especially, how well they correlate with the objective variable—is the aim. The purpose of this focused selection is to assess the predictive power of the SARIMAX model and ascertain whether a smaller set of characteristics chosen by correlation can produce predictions that are both accurate and successful for the provided time series data.

Key metrics for SARIMAX models broken down into three feature categories are provided in Table 3. The "Correlated Feature" (graphs in **Figures 21–22**) and "Selected Feature" (graphs in **Figures 23–24**) models exhibit lower MAE, MSE, MAPE, and RMSE, as well as higher R-Squared, in comparison to the "Full Dataset" model (graphs in **Figures 26–27**), suggesting that feature selection may have an impact on model performance. This implies that the predictive power of the SARIMAX model may be significantly impacted by feature selection.

Table 3 Summary metrics on SARIMAX model with different features.

ML Algorithms	MAE	MSE	MAPE	RMSE	R-Squared
SARIMAX (Correlated Feature)	**32.11**	**1764.97**	**8.31%**	**41.91**	**84.57%**
SARIMAX (Full dataset)	57.73	6136.18	15.69%	68.80	54.81%
SARIMAX (Selected Feature)	33.21	1860.02	8.73%	43.03	84.01%



```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1602
Model:               SARIMAX(2, 1, 1)   Log Likelihood                2245.812
Date:                Wed, 15 Nov 2023   AIC                          -4387.624
Time:                        08:14:08   BIC                          -4107.948
Sample:                             0   HQIC                         -4283.774
                               - 1602
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
 x1            0.6830      0.011     59.944      0.000       0.661       0.705
 x2           -0.0233      0.013     -1.757      0.079      -0.049       0.003
 x3           -0.0166      0.014     -1.215      0.224      -0.043       0.010
 x4            0.7714      0.012     62.850      0.000       0.747       0.795
 x5           -0.0263      0.003     -9.301      0.000      -0.032      -0.021
 x6            0.6477      1.034      0.626      0.531      -1.379       2.674
 x7           -7.9466     12.579     -0.632      0.528     -32.600      16.707
 x8            1.7498      2.762      0.633      0.526      -3.664       7.164
 x9           -1.2521      1.985     -0.631      0.528      -5.143       2.639
 x10           0.2549      0.384      0.663      0.507      -0.499       1.008
 x11          -0.7537      1.162     -0.649      0.517      -3.031       1.523
 x12          -0.0324      0.067     -0.483      0.629      -0.164       0.099
 x13          -0.2898      0.026    -11.009      0.000      -0.341      -0.238
 x14          -0.3145      0.023    -13.635      0.000      -0.360      -0.269
 x15           0.0280      0.003      8.705      0.000       0.022       0.034
 x16           0.0051      0.004      1.378      0.168      -0.002       0.012
 x17          -0.0216      0.004     -5.026      0.000      -0.030      -0.013
 x18          -0.0130      0.004     -3.214      0.001      -0.021      -0.005
 x19          -0.0116      0.004     -3.098      0.002      -0.019      -0.004
 x20          -2.9151      4.583     -0.636      0.525     -11.897       6.067
 x21          -2.2301      3.514     -0.635      0.526      -9.118       4.658
 x22          -1.1002      1.719     -0.640      0.522      -4.469       2.268
 x23          -0.4073      0.650     -0.626      0.531      -1.682       0.867
 x24           0.2549      0.384      0.664      0.507      -0.497       1.007
 x25          -0.5655      0.917     -0.616      0.538      -2.363       1.233
 x26           0.0862      0.117      0.738      0.461      -0.143       0.315
 x27           0.7336      1.185      0.619      0.536      -1.589       3.056
 x28          -0.0871      0.125     -0.696      0.486      -0.332       0.158
 x29           0.5781      0.909      0.636      0.525      -1.204       2.360
 x30           1.2584      1.978      0.636      0.525      -2.618       5.134
 x31           4.3911      6.935      0.633      0.527      -9.202      17.984
 x32          -0.0637      0.225     -0.283      0.777      -0.505       0.378
 x33          -0.0590      0.151     -0.389      0.697      -0.356       0.238
 x34          -0.0346      0.079     -0.439      0.661      -0.189       0.120
 x35          -0.0213      0.029     -0.721      0.471      -0.079       0.037
 x36           0.0166      0.080      0.209      0.834      -0.139       0.173
 x37           0.0793      0.152      0.522      0.602      -0.219       0.377
 x38           0.0824      0.226      0.364      0.716      -0.361       0.526
 const              0        nan        nan        nan         nan         nan
 x39                0        nan        nan        nan         nan         nan
 x40           0.0633      0.014      4.431      0.000       0.035       0.091
 x41           0.0526      0.017      3.167      0.002       0.020       0.085
 x42          -0.0039      0.015     -0.257      0.797      -0.034       0.026
 x43          -0.0310      0.015     -2.135      0.033      -0.059      -0.003
 x44           0.0189      0.013      1.418      0.156      -0.007       0.045
 x45           0.0032      0.013      0.253      0.800      -0.022       0.028
 x46           0.0321      0.012      2.662      0.008       0.008       0.056
 x47           0.0155      0.012      1.254      0.210      -0.009       0.040
 ar.L1        -0.0262      0.033     -0.790      0.429      -0.091       0.039
 ar.L2        -0.1349      0.033     -4.134      0.000      -0.199      -0.071
 ma.L1        -0.8064      0.024    -33.275      0.000      -0.854      -0.759
 sigma2        0.0036      0.000     32.006      0.000       0.003       0.004
 ===================================================================================
 Ljung-Box (L1) (Q):                   0.16   Jarque-Bera (JB):                65.80
 Prob(Q):                              0.69   Prob(JB):                         0.00
 Heteroskedasticity (H):               0.87   Skew:                            -0.04
 Prob(H) (two-sided):                  0.10   Kurtosis:                         3.99
 ===================================================================================
```

Figure 25 Exogenous full features and model summary of SARIMAX (2,1,1) (0,0,0,5) results

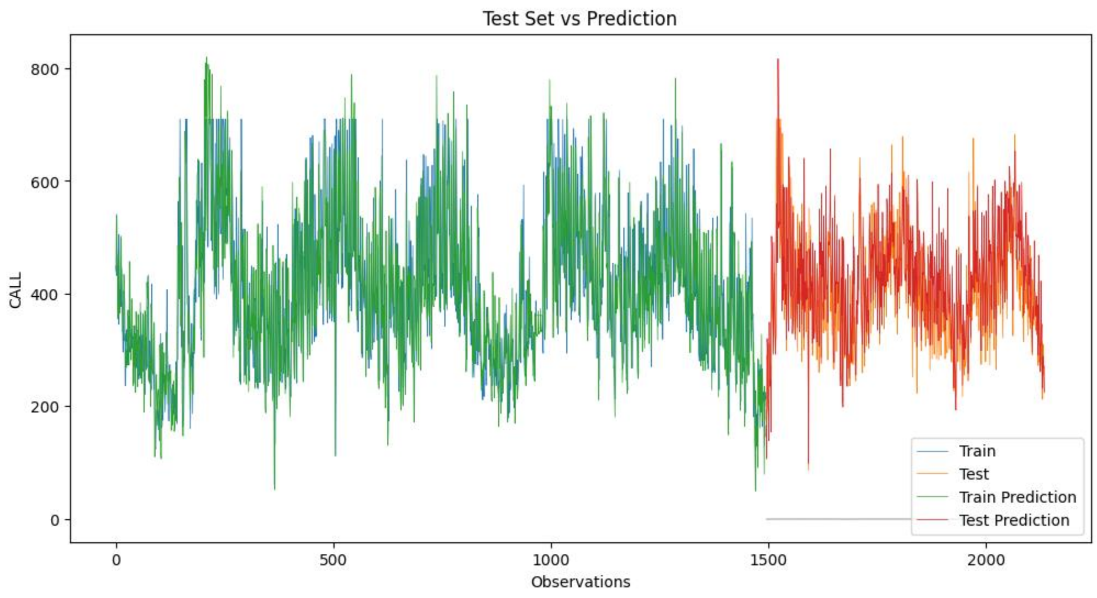


Figure 26 Selected Features : Train and Test Actual vs Prediction plot of SARIMAX (2,1,1) (0,0,0,5) results

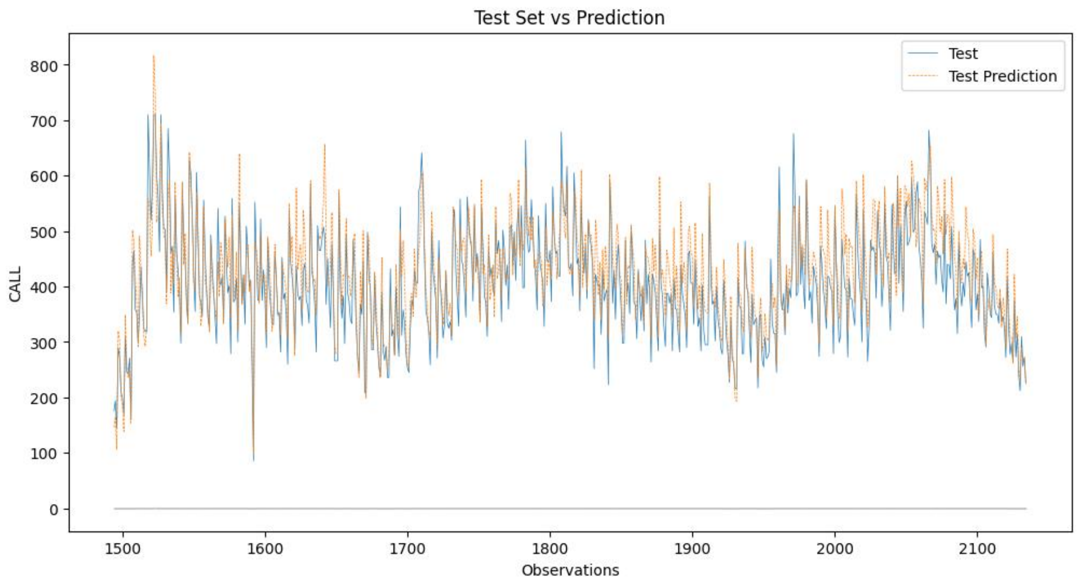


Figure 27 Selected Features : Test Actual vs Prediction plot of SARIMAX (2,1,1) (0,0,0,5)

4.3 Support Vector Regression (SVR)

Support Vector Regression (SVR) emerges as a robust and versatile method for time series forecasting. Leveraging the core principles of support vector machines, SVR significantly enhances traditional regression models by effectively capturing non-linear relationships and hidden patterns within time-dependent data. This capability allows SVR to deliver accurate and reliable predictions across diverse forecasting scenarios. With careful parameter tuning, meticulous feature selection, and a well-designed experimental approach, SVR unlocks valuable insights and empowers informed decision-making in various domains, including finance, healthcare, and beyond. A comprehensive list of hyperparameters is provided in Table 4 for further exploration.

Table 4 SVR Grid Search for Hyperparameter Optimization

Parameter name	Parameter value
C	1, 5, 10, 20, **50**,100
epsilon	0.1, 0.01, **0.001**, 0.0001, 0.00001
gamma	scale', **'auto'**, 0.1, 0.01, 0.001

Kernel : This parameter defines the type of hyperplane used to separate data points in a high-dimensional space. It essentially determines the shape of the decision boundary. RBF is used in the experiment.

Regularization (C): The regularization parameter, denoted as C, plays a crucial role in controlling the balance between minimizing training error and ensuring good generalization to unseen data. A smaller C value increases the emphasis on regularization, preventing the model from fitting the training data too closely, thus promoting better performance on new, unseen data. Regularization is primarily applied during the training phase, influencing the model's learning process. In our experiment, the hyperparameter tuning process involved a grid search to identify the optimal C value for the Support Vector Regression (SVR) model. The best C value, obtained through this process, reflects the chosen level of regularization that contributes to the model's ability to generalize effectively to testing data

Epsilon (epsilon): Epsilon is the margin of tolerance where errors are not penalized.

Kernel Coefficient (gamma): Gamma defines how far a single training example influences a non-linear kernel (RBF, Polynomial).

The best parameters from grid search for SVR is (kernel='rbf', C=50, epsilon=0.001, gamma='auto'). Table 5 summarizes the performance metrics of SVR in a time series forecasting task. SVR models are evaluated based on MAE, MSE, MAPE, RMSE, and R^2. The selected features are adopted from the top 6 most importance of XGBoost in Figure 32. The SVR model with "Selected Feature" set outperforms others with the lowest MAE (25.13), MSE (1208.86), MAPE (6.15%), and RMSE (34.66), along with the highest R^2 (90.56%). Feature selection significantly enhances predictive accuracy. The "Correlated Feature" SVR model performs well, demonstrating competitive metrics, while the "Full dataset" SVR model exhibits slightly higher errors and lower R^2, indicating potential noise. Overall, these results emphasize the importance of thoughtful feature selection in optimizing SVR for time series forecasting. The forecast results are visualized in Figure 28 – 29.

Table 5 Summary metrics on SVR model with different features.

ML Algorithms	MAE	MSE	MAPE	RMSE	R-Squared
SVR (Correlated Feature)	26.20	1344.36	6.34%	36.60	89.40%
SVR (Full dataset)	39.87	2841.64	10.22%	52.09	78.85%
SVR (Selected Feature)	**25.13**	**1208.86**	**6.15%**	**34.66**	**90.56%**

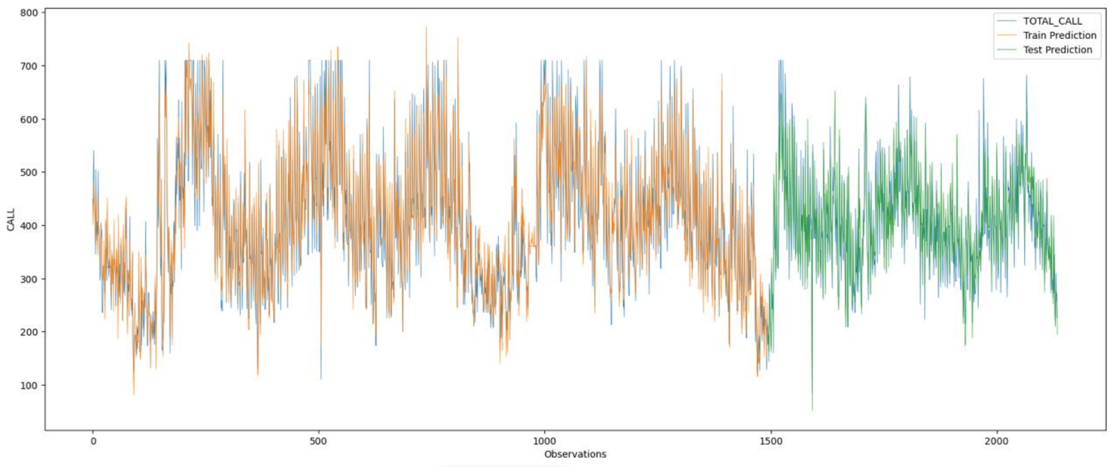


Figure 28 Selected Feature: Train and Test Actual vs Prediction plot of SVR

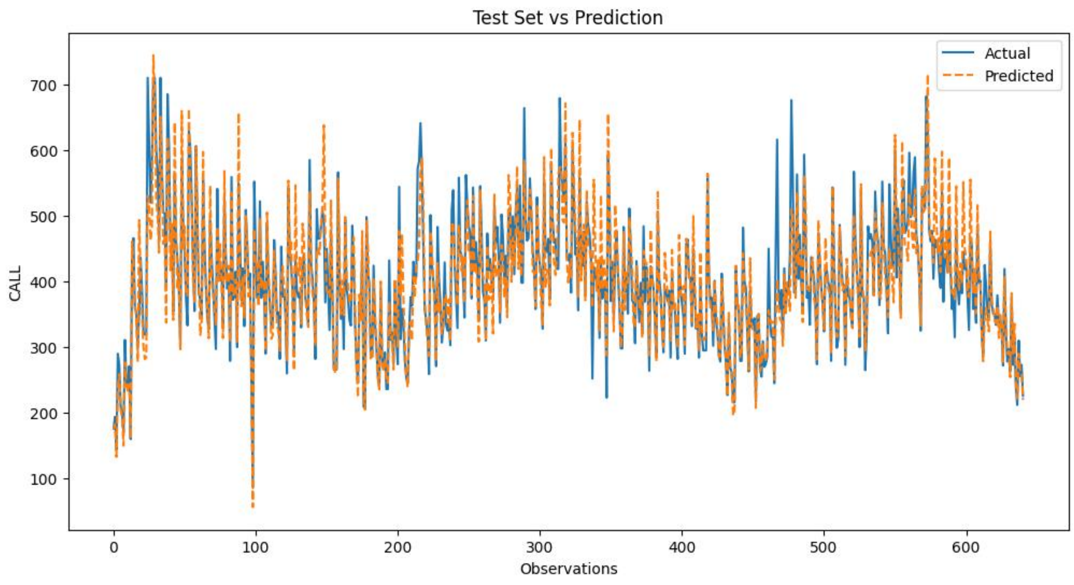


Figure 29 Selected Feature: Test Actual vs Prediction plot of SVR

4.5 Gradient Boosting Models

In this experimental study, the initial phase involved the application of hyperparameter tuning to LightGBM, Gradient Boosting Regressor, and XGBoost models using a dataset comprising correlated features. The primary objective was to ascertain the optimal hyperparameter configuration through an exhaustive Grid Search. The best parameters identified in this phase were subsequently employed as fixed hyperparameters in subsequent analyses. The subsequent stages of the experiment sought to investigate the model's performance under different feature subsets. Three distinct feature groups were considered: the full dataset, a subset of correlated features, and a manually selected feature set. To rigorously evaluate model performance and robustness, Time Series Cross-Validation was employed, ensuring that temporal dependencies in the data were appropriately considered during the analysis. The experimental design thus encompassed both hyperparameter tuning and feature subset exploration, providing a comprehensive examination of the models' behavior under various configurations, with a particular emphasis on its performance in the context of time series data. The ensuing sections present the findings of these analyses and shed light on the interplay between hyperparameters, feature subsets, and model performance.

4.5.1 Light GBM

In the parameter tuning process for LightGBM, a predefined hyperparameter grid is established, encompassing variations for key model parameters such as num_leaves, learning_rate, n_estimators, reg_alpha, and reg_lambda explained in Chapter 2. This grid serves as the search space for identifying the optimal hyperparameter combination. Subsequently, an instance of the LightGBM model is created, either with default hyperparameters or an initial set. The parameter tuning itself is carried out using a search technique, such as grid search or randomized search, where the model is trained and evaluated on different combinations of hyperparameters.

Table 6 Light GBM Grid Search for Hyperparameter Optimization

Hyperparameter name	Parameter value
num_leaves	[**20**, 30, 40]
learning_rate	[0.05, **0.1**, 0.2]
n_estimators	[100, **400**, 1000]
reg_alpha	[0.1, **0.5**, 1.0]
reg_lambda	[**0.1**, 0.5, 1.0]

reg_lambda: Regularization term on weights (L2 regularization). Similar to reg_alpha, reg_lambda adds a penalty term for the complexity of the model. Higher values of reg_lambda increase the regularization strength, promoting a more generalized model by penalizing large coefficient.

In Table 6 we have identified the hyperparameters through a grid search. This particular combination, consisting of a learning rate of 0.1, 400 estimators, 20 leaves a regularization alpha of 0.5 and a regularization lambda of 1.0 has proven to yield the highest model performance, for the assigned task.

The feature selection process involved the identification of a subset that represents the intersection of two key criteria: highly correlated features (Figure 15) and the top 10 features with the highest gain (**Figure 30**) in the LightGBM model. Specifically, the features 'ABANDONED_RATE', 'ANSWER_SPEED_SECS', 'SERVICE_LEVEL', 'TOTAL_CALL_LAG_1', 'TOTAL_CALL_LAG_5', and 'TOTAL_CALL_LAG_10' were chosen. This strategic selection aimed to capture the shared characteristics of features exhibiting strong correlation while simultaneously incorporating those deemed most influential by the LightGBM model based on their gain values. By integrating these criteria, the chosen feature set represents a nuanced combination of correlated attributes and top-performing features, providing a comprehensive basis for subsequent analyses.

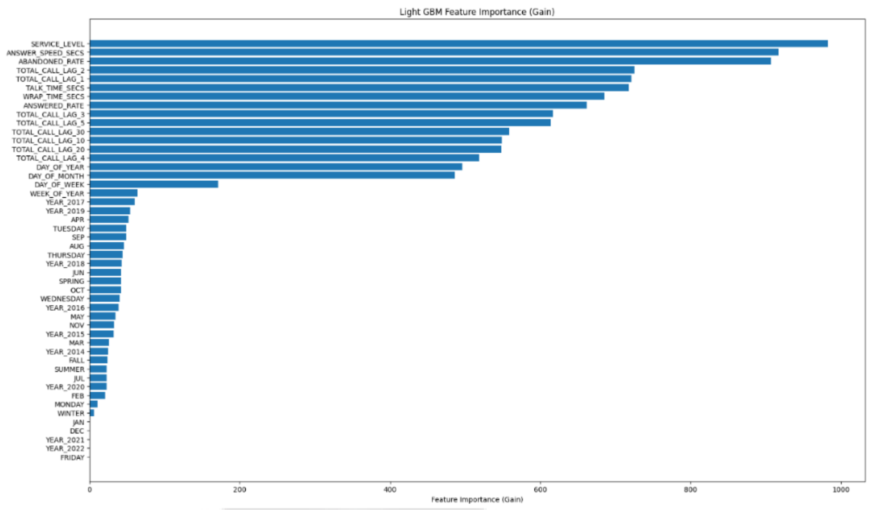


Figure 30 Feature Importance Light GBM in Full dataset

The feature selection process involved the identification of a subset that represents the intersection of two key criteria: highly correlated features (Figure 15) and the top 10 features with the highest gain (Figure 30) in the LightGBM model. Specifically, the features 'ABANDONED_RATE', 'ANSWER_SPEED_SECS', 'SERVICE_LEVEL', 'TOTAL_CALL_LAG_1', 'TOTAL_CALL_LAG_5', and 'TOTAL_CALL_LAG_10' were chosen. This strategic selection aimed to capture the shared characteristics of features exhibiting strong correlation while simultaneously incorporating those deemed most influential by the LightGBM model based on their gain values. By integrating these criteria, the chosen feature set represents a nuanced combination of correlated attributes and top-performing features, providing a comprehensive basis for subsequent analyses.

These results **in** Table 7 provide a comprehensive evaluation of the LightGBM model's performance under different feature configurations. The metrics include measures of accuracy (MAE, MSE), relative performance (MAPE), and goodness of fit (RMSE, R-Squared). The comparison between the three configurations, including one with correlated features, one with the full dataset, and one with selected features, offers insights into how feature selection impacts model performance. Overall, the model appears to perform well, with the selected feature set achieving slightly better results in terms of MAE, MSE, MAPE, and RMSE, as well as a marginally higher R-squared value compared to the other configurations.

Table 7 Summary metrics on Light GBM model with different features.

ML Algorithms	MAE	MSE	MAPE	RMSE	R-Squared
LightGBM (Correlated Feature)	**30.73**	1820.33	**7.73%**	42.52	84.92%
LightGBM (Full dataset)	31.07	1835.32	7.91%	42.66	84.77%
LightGBM (Selected Feature)	30.95	**1799.3**	7.81%	**42.25**	**84.95%**

4.5.2 Gradient Boosting Regressor

experiment's chosen model for predicting call center performance is a Gradient Boosting Regressor. A thorough hyperparameter tuning process is carried out in table 8 to optimize the model's performance. Key hyperparameters such as 'n_estimators,' 'learning_rate,''max_depth,' and'subsample' are systematically investigated to determine the best combination. The grid search yields the optimal model hyperparameters, which are 'learning_rate': 0.1,'max_depth': 3, 'n_estimators': 300, and'subsample': 0.8. These hyperparameters are essential for fine-tuning the model for improved predictive accuracy.

The feature importance analysis gives useful information about the importance of many aspects influencing call center performance. The entire dataset feature importance reveals in **Figure 31** that certain attributes such as 'YEAR_2022,' 'YEAR_2021,' and 'JAN' have low value, implying a limited impact on call center metrics. On the contrary, characteristics such as 'TOTAL_CALL_LAG_5,' 'SERVICE_LEVEL,' and 'ANSWER_SPEED_SECS' show significant significance, underscoring their critical role in forecasting call center outcomes.

Table 8 Summary metrics on Light GBM model with different features.

Hyperparameter name	Parameter value
n_estimators	[100, 200, **300**]
eta	[0.05, **0.1**, 0.2]
max_depth	[**3**, 5, 7]
subsample	[**0.8**, 1.0]

Further investigation into correlated features identifies a set of attributes that exhibit a strong interdependence. Particularly, 'TOTAL_CALL_LAG_5', 'ANSWER_SPEED_SECS’, and 'SERVICE_LEVEL' emerge as correlated features with high importance, emphasizing their collective impact on call center performance. Understanding these correlations allows for a more nuanced interpretation of the model's decision-making process.

Additionally, manual selection of features reveals a subset of attributes that researchers may consider for a focused analysis. 'TOTAL_CALL_LAG_10,' 'ANSWERED_RATE,' 'ABANDONED_RATE,' and 'TOTAL_CALL_LAG_1' demonstrate notable importance, signifying their potential as key indicators of call center performance.

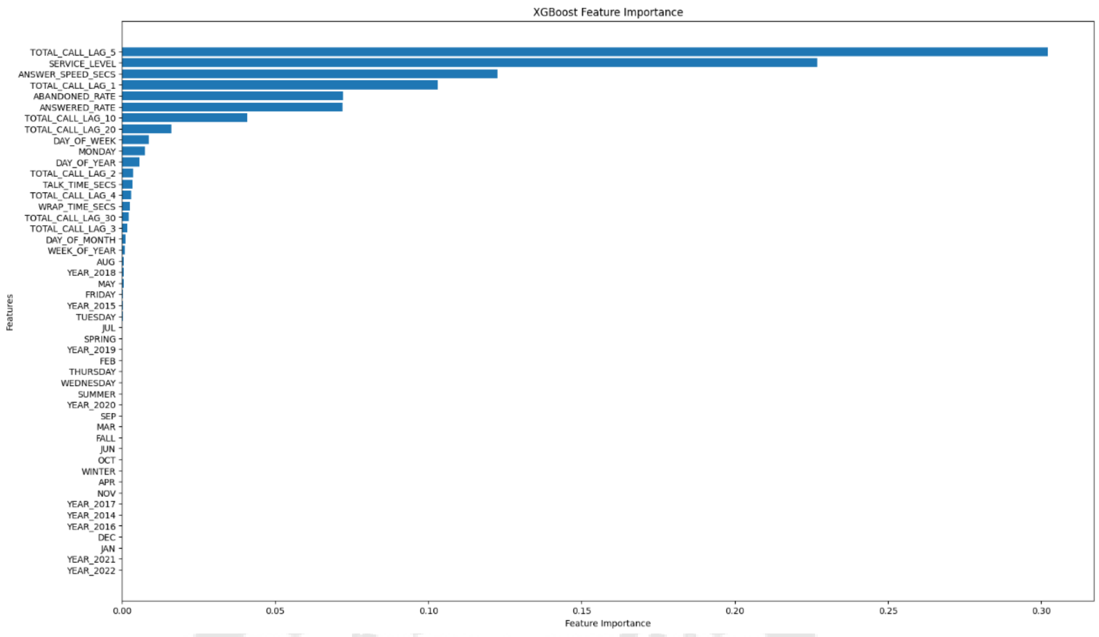


Figure 31 Feature Importance Gradient Boosting Regressor in Full dataset

Table 9 Summary metrics on Gradient Boosting Regressor model with different features.

ML Algorithms	MAE	MSE	MAPE	RMSE	R-Squared
GBR(Correlated Feature)	29.19	1596.85	7.32%	39.89	86.62%
GBR (Full dataset)	29.35	1606.37	7.45%	39.93	86.65%
GBR (Selected Feature)	28.77	**1545.42**	**7.18%**	**39.24**	**87.04%**

In the table 9, the GBR model with the manually selected features performs slightly better than the models using the full dataset or correlated features. However, all three models demonstrate reasonable predictive performance, as evidenced by the high R-Squared values and relatively low error metrics. The choice of features has a discernible impact on model performance, emphasizing the importance of feature selection in building effective machine learning models.

4.5.3 XGBoost

In this experiment, we aim to optimize the performance of an XGBoost regression model by tuning key hyperparameters through a systematic grid search approach. The hyperparameters under consideration include n_estimators, eta, gamma, max_delta_step, max_depth, max_leaves, and min_child_weight. The n_estimators parameter determines the number of boosting rounds, while eta controls the learning rate to prevent overfitting. The gamma parameter adds a regularization term to minimize overcomplicated models, and max_delta_step adjusts the step size during optimization. max_depth and max_leaves influence the depth and number of leaves in each tree, respectively, impacting the model's complexity. Finally, min_child_weight sets the minimum sum of instance weight required in a child node. These hyperparameters collectively govern the trade-off between model complexity and generalization.

Table 10 shows the optimal hyperparameter values that contribute to high-performance results in the XGBoost regression model. These parameters include objective='reg:squarederror', booster='gbtree', n_estimators=200, eta=0.1, gamma=0.001, max_delta_step=1, max_depth=5, max_leaves=25, and min_child_weight=1. These carefully chosen settings are associated with superior model performance, as evidenced by the graph.

The process of selecting features entailed the identification of a subset that captures the intersection of two pivotal criteria: highly correlated features (as depicted in Figure 15) and the top 10 features with the highest gain (illustrated in **Figure 32**) according to the XGBoost model. Specifically, the features 'ABANDONED_RATE', 'ANSWER_SPEED_SECS','SERVICE_LEVEL','TOTAL_CALL_LAG_1','TOTAL_CALL_LAG_5', 'TOTAL_CALL_LAG_10', and 'SUMMER' were chosen. The fifth lag, denoted by "TOTAL_CALL_LAG_5," consistently retains a dominant influence within the full dataset, a trend that remains evident when examining correlated features (**Figure 35**). Significantly, "TOTAL_CALL_LAG_5" continues to stand out as the foremost contributor, surpassing the impact of other features in the dataset.

Table 10 Light GBM Grid Search for Hyperparameter Optimization

Hyperparameter name	Parameter value
n_estimators	[100, **200**, 300]
eta	[0.05, **0.1**, 0.2]
gamma	[**0.0001**, 0.001, 0.01]
max_delta_step	[0, **1**, 2]
max_depth	[3, **5**, 7]
max_leaves	[20, **25**, 30]
min_child_weight	[**1**, 2, 3]

The XGBoost models were evaluated across three distinct feature subsets, each showcasing competitive performance (Table 11). Notably, the "Selected Feature" group, comprised of a mere seven features, outperformed the comprehensive "Full Dataset" configuration in terms of both MAE and MSE. This compelling result underscores the significance of the specifically chosen features which demonstrate a pronounced impact on the XGBoost model's forecasting capabilities. The efficiency of this reduced yet influential feature set highlights its critical role in enhancing the model's predictive accuracy compared to a broader array of features. The forecasting visualization is depicted in **Figures 30** and **31**.

Table 11 Summary metrics on XGBoost model with different features.

ML Algorithms	MAE	MSE	MAPE	RMSE	R-Squared
XGBoost (Correlated Feature)	30.53	**1751.30**	7.79%	**41.72**	**85.26%**
XGBoost (Full dataset)	31.16	1857.66	7.98%	42.86	84.49%
XGBoost (Selected Feature)	**30.51**	1751.47	**7.73%**	41.72	85.12%

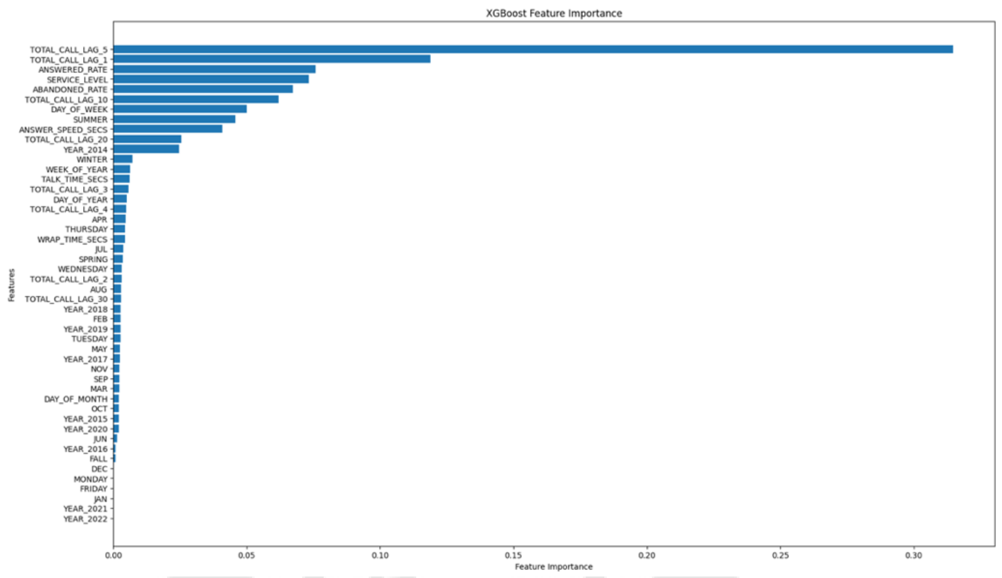


Figure 32 XGBoost Feature Importance (Full dataset)

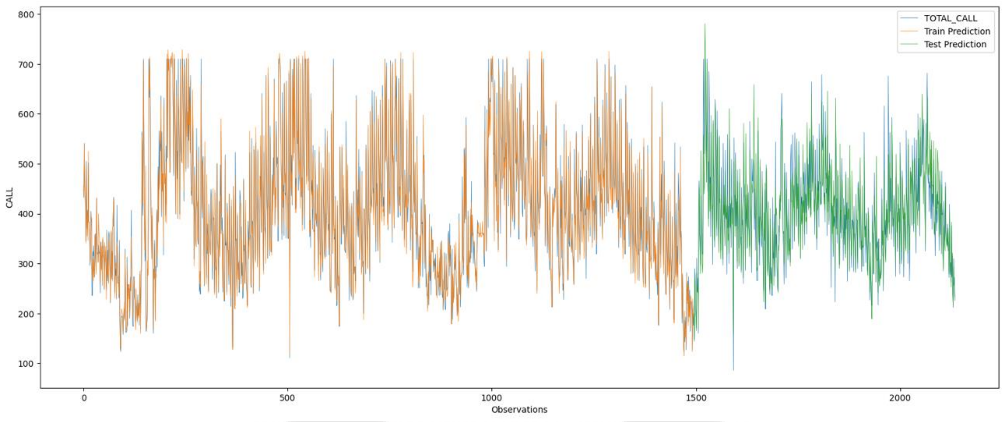


Figure 33 Correlated Feature: Train and Test Actual vs Prediction plot of XGBoost

4.6 Deep Learning

Deep learning model like RNN, GRU, and LSTM, in call center forecasting offers significant advantages. In this experiment, the dataset is split into training and testing sets for evaluating the performance of different RNN architectures on time series forecasting. The training dataset constitutes 70% of the entire dataset, and the remaining 30% is designated as the testing dataset. For each randomly selected hyperparameter combination, the training process involves using the TimeseriesGenerator to create sequences of data for both the input features (X) and the target variable (y). The training sequences are generated with a specified lookback period, allowing the model to learn patterns in the time series data.

The training phase spans multiple epochs, where the model is iteratively updated based on the training data. Early stopping is implemented with a patience of 10 epochs, monitoring the validation loss to prevent overfitting. The average validation loss is calculated over the training epochs, serving as an indicator of the model's performance on unseen data.

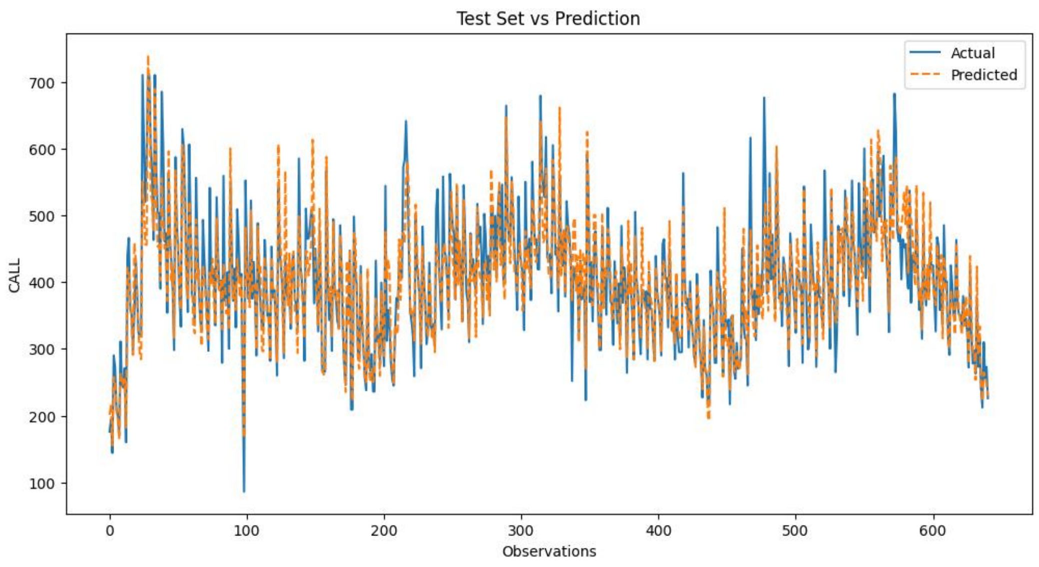


Figure 34 Correlated Feature: Train and Test Actual vs Prediction plot of XGBoost

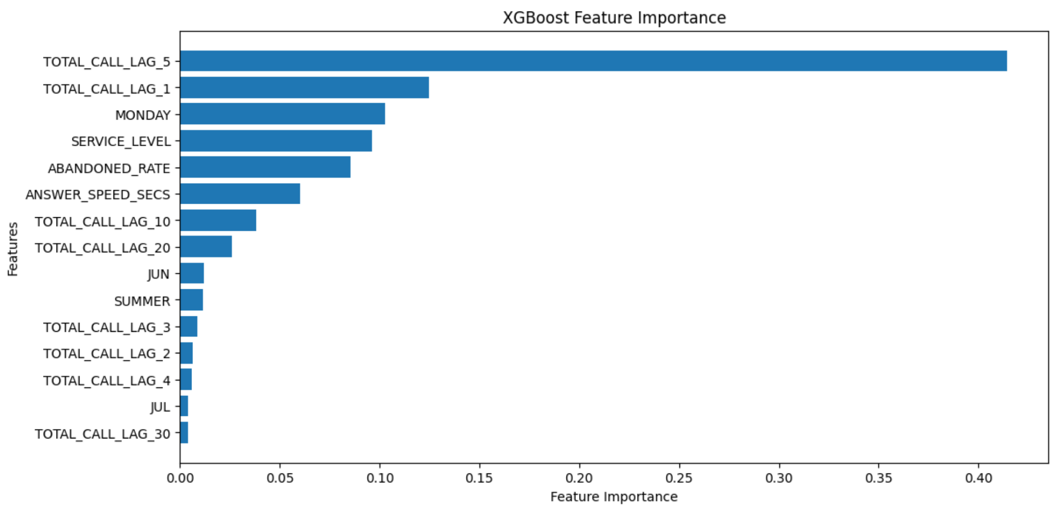


Figure 35 XGBoost Feature Importance (Correlated Features)

After training, the model is utilized to make predictions on both the training and testing datasets. The predictions are inverse transformed to the original scale using Min-Max scaling. Evaluation metrics are then computed for both the training and testing predictions.

Table 12 Experimental setup configurations of deep learning model

Model	Nodes	Batch size	Layer Added	Sequence Length
Simple RNN	32, 64, 128,256, 512	16, 32, 64	32, 64, 128	5, 10, 20
GRU				
LSTM				

By assessing the model's performance on both datasets, this experiment aims to gauge the generalization capabilities of the trained RNN architectures. It helps in understanding how well the models have learned the underlying patterns in the training data and how effectively they can make accurate predictions on previously unseen testing data. This comprehensive evaluation strategy ensures a robust assessment of the models' forecasting capabilities across different hyperparameter configurations.

A number of experimental setups with varying batch sizes, lookback times, network node counts, and layer architectures are listed in Table 12. A metric for evaluating the model's performance in every configuration is provided: the average validation loss. One important metric that indicates how well the model generalizes to new data during training is the validation loss. The validation loss decreases with increasing model prediction accuracy. The parameters of the model are used;

- EarlyStopping parameter (verbose=1, patience=10, monitor='val_loss') When the validation loss stops improving, the training process is terminated early. The model will cease training if, after a predetermined number of epochs (patience), the validation loss does not improve. It is indicated that the validation loss is being tracked by the monitor='val_loss'.
- ReduceLROnPlateau(monitor='val_loss,' factor=0.5, patience=5, min_lr=1e-7, verbose=1) description: The learning rate is adjusted by the ReduceLROnPlateau callback when the validation loss reaches a plateau. After a predetermined number of epochs with no improvement in validation loss (defined by patience), the learning rate is lowered by a facto. Min_lr establishes the bottom bound for the learning rate.
- Compilation: loss='mean_squared_error': The loss function is Mean Squared Error (MSE).
- Optimizer='adam': For optimization, the Adam optimizer is used.
- metrics=['mae']: MAE is another metric to keep track of during training.

Following the identification of ideal hyperparameters for each model, the chosen configurations are used in a thorough evaluation procedure that includes the use of time series cross-validation. This methodology is achieved by utilizing the scikit-learn library's TimeSeriesSplit class, which systematically partitions the dataset into three distinct folds (n_splits=3), distinguishing discrete temporal segments to confirm metrics results.

4.6.1 SimpleRNN

Table 13 presents the results of experimenting with different configurations of a SimpleRNN model for time series prediction. The highlighted configuration, SimpleRNN with 32 nodes, a lookback of 5, batch size 64, and an additional Dense layer with 64 units, stands out as the most promising based on its low average validation loss of 0.021087. This indicates that the model, when trained with these specific hyperparameters, achieved superior performance in predicting the target variable on the validation dataset. While other configurations exhibit slightly higher validation losses, this particular setup demonstrates the potential for effective time series forecasting. Consequently, it is a strong candidate for further investigation and implementation in real-world scenarios where accurate predictions are crucial.

In this experiment, three distinct feature groups were employed for the Simple RNN model, with feature selection based on the most impactful attributes identified by a Gradient Boosting model to be “Selected Feature” group such as 'ABANDONED_RATE’,'ANSWER_SPEED_SECS,'SERVICE_LEVEL’,'TOTAL_CALL_LAG_1 ,'TOTAL_CALL_LAG_5,' 'TOTAL_CALL_LAG_10,' and 'ANSWERED_RATE’. The "Correlated Feature" group produced promising results in Table 14. The model demonstrated accuracy with a MAE of 58.92, emphasizing precise predictions. The MSE at 6256.60 and RMSE at 78.72 highlighted the model's precision. An R-Squared value of 50.70% suggests that the model has demonstrated a moderate proficiency in capturing underlying data patterns. However, when the model was applied to the "Full Dataset," it faced challenges, resulting in higher MAE, MSE, and RMSE values, and a diminished R-Squared of 13.02%. Notably, despite the overall performance being relatively modest, the "Selected Feature" group showcased results comparable to the "Correlated Feature." With a MAE of 59.30 and an R-Squared of 49.25%, this finding underscores the significance of feature selection in enhancing the Simple RNN's predictive capabilities.

While the model's overall performance may be considered suboptimal, the improved performance with limited features highlights the potential benefits of a more focused feature set up.

Table 13 Experimental Setup Configurations and Results of RNN

Type	Nodes	Lookback	Batch Size	Layer Added	Average Validation Loss
SimpleRNN	32	5	64	64	0.021087
SimpleRNN	32	20	64	32	0.026774
SimpleRNN	64	10	16	128	0.024709
SimpleRNN	64	20	64	128	0.02632
SimpleRNN	64	20	32	32	0.026546
SimpleRNN	128	5	64	128	0.023994
SimpleRNN	128	20	16	32	0.027361
SimpleRNN	128	20	64	32	0.027887
SimpleRNN	128	20	16	64	0.028016
SimpleRNN	256	5	16	128	0.025049
SimpleRNN	512	20	32	64	0.026175
SimpleRNN	512	20	32	32	0.031727
SimpleRNN	512	10	64	32	0.033783

Table 14 Summary metrics on Simple RNN model with different features.

ML Algorithms	MAE	MSE	MAPE	RMSE	R-Squared
Simple RNN (Correlated Feature)	**58.92**	**6256.6**	**15.75%**	**78.72**	**50.70%**
Simple RNN (Full dataset)	83.22	11262.74	21.40%	104.98	13.02%
Simple RNN (Selected Feature)	59.3	6465.89	15.79%	80	49.25%

4.6.2 GRU

The results of various configurations evaluated on a GRU model for time series prediction are summarized in Table 15. The setup with a GRU with 256 nodes, a lookback duration of 5, a batch size of 64, and an extra Dense layer with 32 units outperforms the others, with an excellent the validation loss of 0.021568.

Table 15 Experimental Setup Configurations and Results of GRU

Type	Nodes	Lookback	Batch Size	Layer Added	Average Validation Loss
GRU	32	20	64	32	0.025781
GRU	32	10	16	64	0.026253
GRU	32	20	16	128	0.026334
GRU	32	20	16	64	0.034352
GRU	32	10	64	128	0.043032
GRU	64	10	16	32	0.024401
GRU	64	5	32	64	0.025012
GRU	64	5	16	32	0.025103
GRU	64	20	32	32	0.030067
GRU	64	20	64	128	0.037075
GRU	128	10	32	64	0.02495
GRU	128	10	64	64	0.026905
GRU	**256**	**5**	**64**	**32**	**0.021568**
GRU	256	5	64	64	0.023658
GRU	256	10	64	32	0.02521
GRU	256	20	16	128	0.026368
GRU	256	20	16	32	0.026714
GRU	512	20	64	32	0.028279

The results from the GRU model across different feature groups in Table 15 provide valuable insights into its predictive capabilities. In the case of the "Correlated Feature" group, the GRU model achieved a MAE of 64.34. The associated MSE at 7322.52 and RMSE at 84.89 reflect the model's precision. The R² value of 43.44% suggests that the model captures a substantial portion of the variance in the data. Interestingly, when applied to the "Full Dataset," the GRU model exhibited slightly lower MAE and MSE values, along with a higher R² (48.17%). Notably, the "Selected Feature" group demonstrated the best performance, with a lower MAE of 58.88, MSE of 6307.37, and an elevated R² of 50.52%. This underscores the importance of feature selection, as the model benefited from focusing on specific relevant features, namely 'ABANDONED_RATE,''ANSWER_SPEED_SECS,''SERVICE_LEVEL,''TOTAL_CALL_LAG_1,''TOTAL_CALL_LAG_5,''TOTAL_CALL_LAG_10,' and 'ANSWERED_RATE.' These findings emphasize the potential for enhancing GRU model performance by judiciously selecting input features based on their significance in the context of the prediction task.

Table 16 Summary metrics on GRU model with different features.

ML Algorithms	MAE	MSE	MAPE	RMSE	R-Squared
GRU (Correlated Feature)	64.34	7322.52	16.84%	84.89	43.44%
GRU (Full dataset)	60.49	6787.43	15.66%	81.53	48.17%
GRU (Selected Feature)	**58.88**	**6307.37**	**15.83%**	**79.00**	**50.52%**

4.6.3 LSTM

The experimentation with different configurations of LSTM models shown in Table 17 for time series prediction revealed that the model with 128 nodes, a lookback of 5, batch size 16, and an additional Dense layer with 32 units gained the best hyperparameters. This specific configuration achieved the lowest average validation loss of 0.021457, indicating superior performance in predicting the target variable on the validation dataset. Among the various setups tested, this LSTM model configuration demonstrated the most promising results. It outperformed other configurations, such as those with 32, 64, or 256 nodes, across different lookback periods, batch sizes, and layer additions. These findings highlight the significance of tuning hyperparameters for LSTM models to achieve optimal predictive capabilities, with a focus on the number of nodes, lookback period, batch size, and layer architecture.

Table 17 Experimental Setup Configurations and Results of LSTM

Type	Nodes	Lookback	Batch Size	Layer Added	Average Validation Loss
LSTM	32	10	64	32	0.024289
LSTM	32	5	64	128	0.02632
LSTM	32	10	16	128	0.026478
LSTM	32	20	64	64	0.034471
LSTM	64	5	32	128	0.026081
LSTM	64	20	16	64	0.026931
LSTM	64	20	64	64	0.030435
LSTM	**128**	**5**	**16**	**32**	**0.021457**
LSTM	128	10	32	32	0.023483
LSTM	128	5	64	32	0.024624
LSTM	128	5	64	64	0.030634
LSTM	256	10	16	32	0.023492
LSTM	256	10	32	32	0.024401
LSTM	256	5	16	64	0.026767
LSTM	256	20	16	64	0.026995
LSTM	256	10	64	64	0.029533
LSTM	256	20	64	128	0.035616
LSTM	512	5	64	64	0.023475
LSTM	512	10	64	32	0.025123

The LSTM model, trained on the "Selected Feature" group, which includes 'ABANDONED_RATE','ANSWER_SPEED_SECS','SERVICE_LEVEL','TOTAL_CALL_LAG_1', 'TOTAL_CALL_LAG_5', 'TOTAL_CALL_LAG_10', and 'ANSWERED_RATE,' exhibited superior predictive performance (Table 18) compared to other feature sets. This configuration resulted in a significantly lower MSE, indicating reduced overall prediction errors. Similarly, the MAE for the "Selected Feature" group was notably lower, emphasizing improved accuracy in predicting the target variable. The model also achieved a lower RMSE, underlining enhanced precision. These results collectively highlight the effectiveness of feature selection in optimizing the LSTM model for time series prediction, with a notable increase in predictive power as indicated by the improved R-squared value.

Table 18 Summary metrics on LSTM model with different features

ML Algorithms	MAE	MSE	MAPE	RMSE	R-Squared
LSTM (Correlated Feature)	61.41	8316.18	18.18%	90.58	57.76%
LSTM (Full dataset)	62.99	7717.16	17.57%	87.69	60.35%
LSTM (Selected Feature)	**54.53**	**6212.75**	**15.20%**	**78.48**	**68.31%**

4.7 Data Transformation

The use of differencing transformations in time series analysis provides several benefits, including improved model performance and prediction accuracy. Differentiating helps stabilize variance and eliminate seasonality by computing the difference between consecutive observations, allowing for a more robust analysis of temporal patterns (see **Figure** 33). The total call is more volatile, and the metrics are poor.

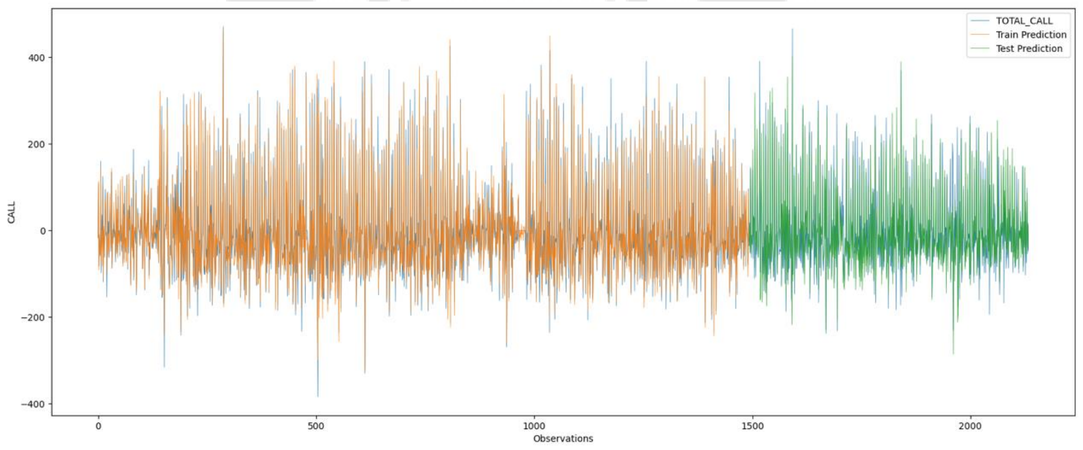


Figure 36 TOTAL_CALL after differenced transformation.

The results presented in table 19 showcase the performance metrics of various machine learning algorithms under two different data transformation scenarios: "No Day off series" and "Differenced Transform. In the "No Day off series" transformation, several algorithms, including SVR, Gradient Boosting Regressor (GBR), XGBoost, and LightGBM, exhibit relatively consistent and competitive performance across all metrics. SVR, in particular, stands out with the lowest MAE, MAPE, and RMSE, indicating accurate predictions and minimal error. Simple RNN, GRU, and LSTM, on the other hand, show higher errors, with LSTM performing the best among the three.

Table 19 Comparison model performance metrics by using non-transformed observations and differenced transformed observations.

Transformation	ML Algorithms	MAE	MAPE	RMSE	R^2
No Day off series	SARIMAX	32.11	8.31%	41.91	84.57%
	SVR	26.64	6.34%	36.60	89.40%
	LightGBM	30.73	7.73%	42.52	84.92%
	Gradient Boosting Regressor	29.19	7.32%	39.89	86.62%
	XGBoost	30.53	7.79%	41.72	85.26%
	Simple RNN	58.92	15.75%	78.72	50.70%
	GRU	64.34	16.84%	84.89	43.44%
	LSTM	61.41	18.18%	90.58	57.76%
Differenced Transform	SARIMAX	31.97	inf%	41.55	86.02%
	SVR	27.44	3.71E-02	38.53	86.49%
	LightGBM	33.47	6.16+00%	46.97	77.46%
	Gradient Boosting Regressor	33.44	inf%	47.01	77.42%
	XGBoost	33.56	5.10E-02	47.48	79.48%
	Simple RNN	139.63	7.14E+17	77.04	46.27%
	GRU	54.01	7.14E-02	77.04	46.27%
	LSTM	54.25	6.90%	76.15	47.51%

In the "Differenced Transform" scenario, the results are notably different. The SARIMAX model achieves improved performance in terms of MAE and R-Squared, indicating better predictive accuracy after differencing the data. However, for several algorithms, such as Simple RNN, GRU, and LSTM, the differencing transformation results in extremely high values for certain metrics, particularly in MAPE, suggesting challenges or instability in modeling the differenced series.

The exceptionally high MAPE values observed in SARIMAX and GBR models in the "Differenced Transform" scenario can be attributed to the introduction of differencing, potentially leading to division by very small or zero values during MAPE calculation. When differencing results in near-zero denominators due to subtracting consecutive observations, the MAPE can yield an extremely large or infinite percentage error (Inf%). This sensitivity of SARIMAX and GBR models to small denominators highlights a potential compromise in their predictive accuracy when differencing generates such challenging conditions.

4.8 Result Analysis

In Table 20, a comprehensive summary of performance metrics is presented, showcasing the evaluation results for various machine learning algorithms across three feature groups: "Correlated Feature," "Full Dataset," and "Selected Feature." The metrics include MAE, MSE, MAPE, RMSE, and R^2. This table provides a detailed insight into the comparative performance of each algorithm within different feature contexts, offering valuable information for model selection and feature engineering strategies.

4.8.1 MAE Analysis

Figure 37 displays the MAE graph illustrating the performance of various machine learning algorithms across three distinct feature groups. In the Correlated Feature group, SVR stands out with the lowest MAE of 26.20, indicating superior predictive accuracy compared to other algorithms in capturing patterns in the correlated features. In contrast, GRU and LSTM models exhibit higher MAE values, with GRU recording 64.34 in the Correlated Feature group, suggesting challenges in predicting the target variable within this feature set.

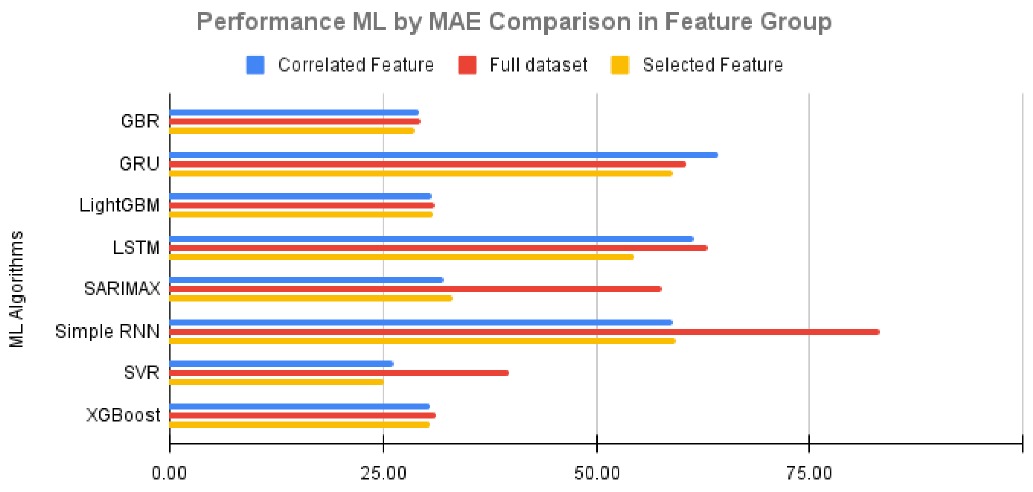


Figure 37 MAE model performance comparison

Moving to the Full dataset group, GBR showcases consistent performance with the lowest MAE of 29.35, reinforcing its robustness across different feature compositions. While GRU and LSTM maintain their relatively higher MAE values, the gap between GBR and the recurrent neural network models narrows, highlighting their improved performance on the broader dataset. In the Selected Feature group, SVR stands out as the top-performing algorithm with the lowest MAE of 25.13, showcasing its effectiveness in capturing patterns within this particular feature subset.

In the Selected Feature group, SVR stands out as the top performer with the lowest MAE of 25.13, emphasizing its adaptability to specific feature subsets and its ability to deliver precise predictions. GBR, LightGBM, and XGBoost showcase commendable performance, maintaining competitive MAE values ranging from 28.77 to 30.95. LSTM also demonstrates efficacy within this more focused feature subset, achieving a notable MAE of 54.53. This collective evaluation provides a comprehensive understanding of how these algorithms perform relative to each other in capturing patterns within the selected feature group.

4.8.2 MAPE Analysis

Within the Correlated Feature group, SVR emerges as the frontrunner, exhibiting an impressive MAPE of 6.34% (**Figure** 38). This remarkable performance highlights SVR's strength in accurately predicting values for datasets characterized by correlated features. GBR, LightGBM, and XGBoost also demonstrate commendable performance, achieving MAPE values ranging from 7.32% to 7.79%.

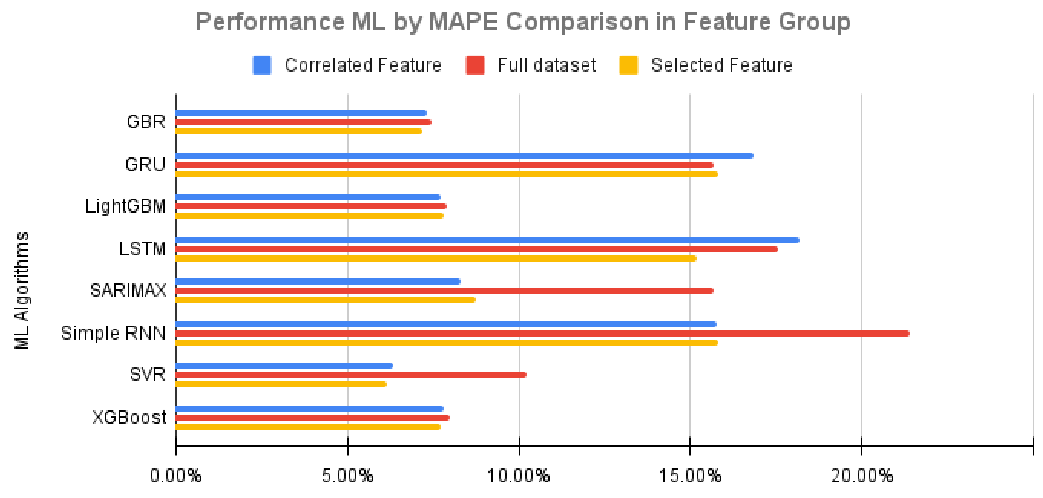


Figure 38 MAPE model performance comparison

SVR has a slightly higher MAPE of 10.22% in the Full Dataset group. Nonetheless, it displays versatility in making correct predictions across a wide range of datasets. GBR, LightGBM, and XGBoost continue to show high accuracy, with MAPE values ranging from 7.45% to 7.98%, confirming their consistency in capturing underlying patterns in large datasets. SARIMAX and deep learning models (Simple RNN, GRU, LSTM) are included for evaluation in the Full Dataset group, providing insights into their relative performance. SARIMAX has a MAPE of 15.69%, indicating competitive accuracy when compared to other models in this dataset. Deep learning models, notably Simple RNN, GRU, and LSTM, perform differently. Simple RNN has a MAPE of 21.40%, showing larger prediction errors, whereas GRU and LSTM outperform with MAPE values of 15.66% and 17.57%, respectively. It's worth noting that these results emphasize the various performance aspects of the Full Dataset group's models.

SVR is the highest performer in the Selected Feature group, with the lowest MAPE of 6.15%, demonstrating its adaptability to specific feature subsets and precise prediction capabilities. GBR, LightGBM, and XGBoost also perform well, with MAPE values ranging from 7.18% to 7.81%, confirming their dependability in catching patterns in this more limited feature subset. SARIMAX performs competitively with a MAPE of 8.73%, showcasing its effectiveness in capturing temporal patterns. Deep learning models, including Simple RNN, GRU, and LSTM, contribute to the analysis within this feature subset. Despite Simple RNN displaying a higher MAPE of 15.79%, GRU and LSTM achieve lower MAPE values of 15.83% and 15.20%, respectively.

4.8.3 RMSE Analysis

Figure 39 illustrates the RMSE across different feature groups. In the Correlated Feature group, SVR stands out with the lowest RMSE of 36.60, emphasizing its strong predictive capabilities when dealing with correlated features. GBR, LightGBM, and XGBoost also exhibit competitive performance, with RMSE values ranging from 39.24 to 42.52, showcasing their reliability in accurate predictions.

Transitioning to the Full Dataset group, GBR stands out with the lowest RMSE of 39.93, demonstrating its superior predictive performance within this feature group. While SVR also performs well with an RMSE of 52.09, it shows a slightly higher error compared to GBR, highlighting the nuances in their predictive capabilities across diverse datasets. LightGBM and XGBoost maintain strong accuracy, with RMSE values ranging from 42.52 to 42.66, showcasing their reliability in capturing underlying patterns within extensive datasets.

In the Selected Feature group, SVR once again emerges as the top performer, achieving the lowest RMSE of 34.66. This underscores SVR's adaptability to specific feature subsets, resulting in precise predictions. GBR, LightGBM, and XGBoost showcase noteworthy performance, with RMSE values ranging from 41.72 to 42.25, affirming their effectiveness in capturing patterns within this more focused feature subset.

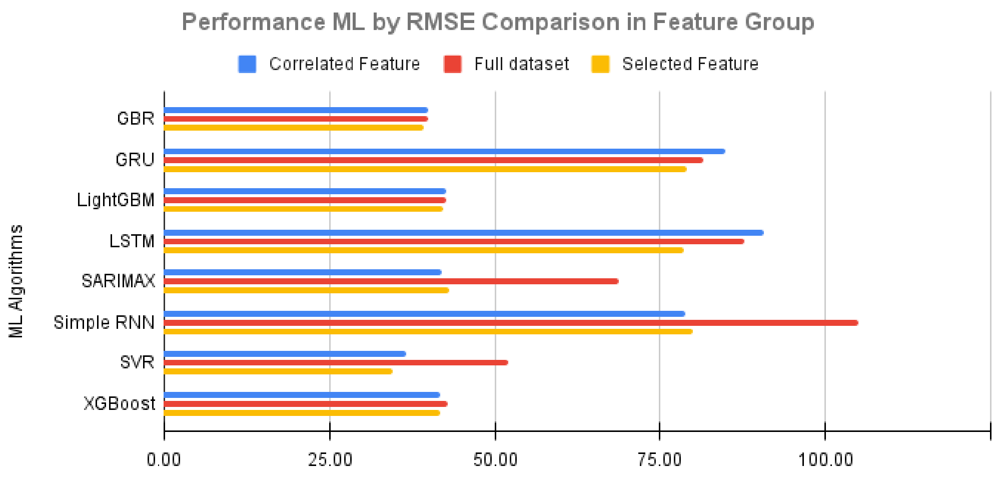


Figure 39 RMSE model performance comparison

4.8.4 R-Squared Analysis

Within the Correlated Feature group, SVR emerges as the standout performer, showcasing impressive R^2 value of 89.40% as Figure 40. This underscores SVR's robust capability to elucidate the variance in the data, particularly in the context of correlated features. Furthermore, GBR, LightGBM, and XGBoost demonstrate consistent and strong performance, maintaining R^2 values in the range of 86.62% to 85.26%. SARIMAX exhibits strong explanatory power with an R^2 value of 84.57%, indicating its effectiveness in explaining the variance in the data. The deep learning models display varying performance, with Simple RNN showing an R^2 value of 50.70%, GRU at 43.44%, and LSTM leading with an R^2 value of 57.76%.

For the Full Dataset group, R^2 of SVR drops 78.85%. GBR, LightGBM, and XGBoost maintain high accuracy, with R^2 values ranging from 86.62% to 85.26%, confirming their consistency in capturing underlying patterns in large datasets. However, in this larger dataset, the performance of RNN model is comparably worse, with R^2 values suggesting less effective explanatory power.

Table 20 Summary metrics by model in 3 feature group

Group Feature	ML Algorithms	MAE	MAPE	RMSE	R^2
Correlated Feature	SARIMAX	32.11	8.31%	41.91	84.57%
	SVR	**26.2**	**6.34%**	**36.6**	**89.40%**
	LightGBM	30.73	7.73%	42.52	84.92%
	GBR	29.19	7.32%	39.89	86.62%
	XGBoost	30.53	7.79%	41.72	85.26%
	Simple RNN	58.92	15.75%	78.72	50.70%
	GRU	64.34	16.84%	84.89	43.44%
	LSTM	61.41	18.18%	90.58	57.76%
Full dataset	SARIMAX	57.73	15.69%	68.8	54.81%
	SVR	39.87	10.22%	52.09	78.85%
	LightGBM	31.07	7.91%	42.66	84.77%
	GBR	**29.35**	**7.45%**	**39.93**	**86.65%**
	XGBoost	31.16	7.98%	42.86	84.49%
	Simple RNN	83.22	21.40%	104.98	13.02%
	GRU	60.49	15.66%	81.53	48.17%
	LSTM	62.99	17.57%	87.69	60.35%
Selected Feature	SARIMAX	33.21	8.73%	43.03	84.01%
	SVR	**25.13**	**6.15%**	**34.66**	**90.56%**
	LightGBM	30.95	7.81%	42.25	84.95%
	GBR	28.77	7.18%	39.24	87.04%
	XGBoost	30.51	7.73%	41.72	85.12%
	Simple RNN	59.3	15.79%	80	49.25%
	GRU	58.88	15.83%	79	50.52%
	LSTM	54.53	15.20%	78.48	68.31%

SVR is the top performance in the Selected Feature group, with R^2 value of 90.56%, highlighting its adaptability to various feature subsets and ability to give exact predictions. GBR, LightGBM, and XGBoost all work admirably, with R^2 values ranging from 84.92% to 85.12%. This validates their ability in capturing patterns within this more focused feature subset. Overall, the R^2 analysis provides a full knowledge of how these methods perform in explaining variance in distinct feature groups.

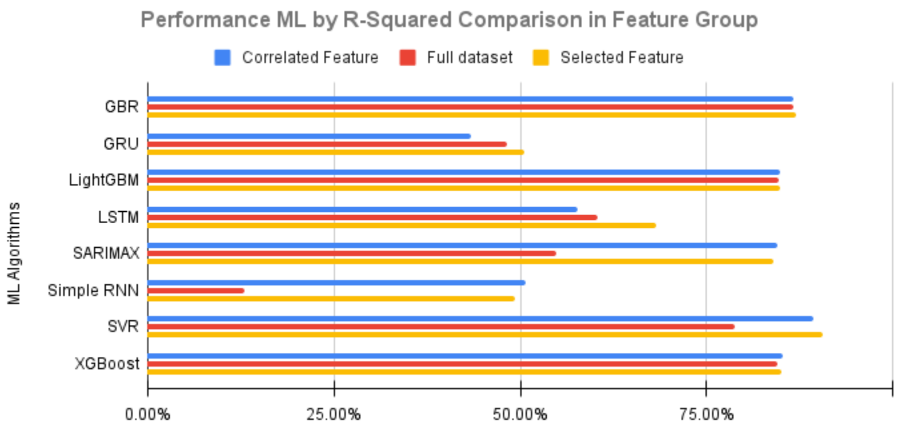


Figure 40 R-Squared model performance comparison

CHAPTER 5
CONCLUSION AND DISCUSSION

5.1 Conclusion

This research delved into the crucial role of Workforce Management (WFM) in enhancing call center effectiveness. Optimized WFM practices, encompassing call volume forecasting and agent scheduling, are key to achieving operational efficiency and cost reduction. Advanced time series models, such as SARIMAX, SVR, Gradient Boosting, RNN, GRU, and LSTM, proved instrumental in accurately predicting contact volume, a cornerstone of successful WFM implementation.

The experiment produced insightful findings and tested several machine learning algorithms for time series forecasting in a call center environment. The models with the lowest MAE, RMSE, MAPE, and highest R2 values were SVR and GBR. This illustrates their capacity to forecast contact volume with accuracy, enabling the strategic distribution of labor over various time intervals.

For efficient training and testing, the system model incorporated significant data pretreatment, metrics translation, normalization, and the usage of time series generators. Even greater model performance was achieved through feature engineering and hyperparameter optimization. The experiment extends beyond the utilization of call center metrics alone; additional time and lag features are engineered to afford the model a more comprehensive understanding of available options. Subsequently, these features are categorized into three distinct groups: the Full Dataset, the Correlated Feature group, which comprises features highly correlated with a correlation coefficient exceeding 0.2, and the Selected Feature group, which includes manually chosen features based on their frequent occurrence and their significance in each model, determined by low p-values in SARIMAX and higher feature importance in the Gradient Boosting model. Key features such as service level call, abandon rate, answer rate, 1st lag of call especially 5th Lag of call with 0.29 feature importance value in GBR and XGBoost models have been identified as pivotal contributors to this study, underscoring their significance in the predictive modeling process.

The experiment's findings demonstrate how crucial sophisticated forecasting methods are for maximizing resource allocation, achieving customer service targets, and eventually raising total customer happiness. Such data-driven strategies will be more and more crucial as call centers develop to remain competitive in the ever-changing field of customer support operations.

5.2 DISCUSSION

5.2.1 Data pre-processing and Differencing Transformations

It was discovered how data preprocessing techniques affected the performance of the model. Higher errors were first observed when the complete dataset containing day-off and holiday information was used, suggesting that these attributes may contain noise. Overall metrics improved after day off entries were removed, suggesting that they had a detrimental effect. The performance of machine learning models can be greatly impacted by the use of differencing procedures in time series analysis. But the measurements suffered from deconstructing and diffusing the data, especially the MAPE metrics. These results highlight how difficult preprocessing choices can be and how crucial it is to take a balanced approach to information retention and noise reduction. Considering multiple indicators guarantees a comprehensive comprehension of the model's efficacy.

5.2.2 Interpretability

In machine learning, the trade-off between interpretability and complexity is critical. While complex models, like deep neural networks, often excel in capturing intricate patterns, their "black box" nature raises concerns about interpretability, bias, fairness, and robustness. In this context, the notable performance of interpretable models like SVR and Gradient Boosting is highlighted. These models strike a balance by providing strong predictive capabilities alongside interpretability. However, it's essential to acknowledge that neural networks, despite their complexity, did not perform well in this study. The choice between a complex, black-box model and a simpler, interpretable model depends on the specific context and requirements of the application.

5.2.3 Dataset and model complexity

Machine learning algorithms' performance is essentially dependent on the features of the dataset and the nature of the task at hand. While SVR and GBR performed well in this study, it is important to note that the efficacy of deep learning models varies depending on the dataset or task. Poor deep learning performance can be due to a variety of issues, one of which being model complexity. Deep neural networks, due to their complexity, may experience difficulties when the dataset is small or lacks the diversity essential for full learning. The risk of overfitting increases in such instances, as these models may capture noise in the data rather than generalizable patterns. This problem is exacerbated further by a lack of training data.

5.2.4 Flexibility and Non-Linearity

SVR is a robust choice for time series forecasting tasks, particularly when dealing with non-linear relationships between variables. Its effectiveness stems from its ability to capture complex patterns that other models may struggle with. SVR is also less sensitive to outliers and requires fewer parameters than other non-linear models, making it less prone to overfitting and better suited for smaller datasets. Additionally, SVR offers a higher degree of interpretability compared to deep learning models, providing insights into the relationships between variables and aiding in understanding the model's behavior. Empirical evidence supports the superiority of SVR in various time series forecasting tasks, particularly in scenarios with non-linear patterns.

5.2.5 Feature Importance

The Full Dataset, Correlated Feature group, and Selected Feature group are the three separate feature groups that the study uses to carefully classify its features. The purpose of this strategic grouping is to provide insight into the various effects that distinct feature subsets have on the effectiveness of predictive models. Specifically, the SARIMAX model performs competitively by utilizing just 5 features: Monday, the first lag, abandoned rate, answer speed, and service level call. This result emphasizes the superiority of a targeted feature selection over a more comprehensive strategy.

SVR is the model that performs best out of all the models that were examined in this study. Its improved performance with selected features that come from the top important features found in the XGBoost model is remarkable. This emphasizes how important feature engineering and selection are to optimizing the performance of sophisticated in SVR.

Within the realm of Gradient Boosting models, specific features consistently emerge as top contributors. Notably, the 5th lag, service level call, and answer speed stand out as the top 3 most important features across GBR and XGBoost. However, in the case of LightGBM, the 5th lag is less important feature, service level call, answer speed, and abandoned rate are the top 3. The abandoned rate takes precedence as the most influential feature for Light GBM in the Selected feature set. This variation in feature importance underscores the nuances of model-specific preferences and highlights the importance of considering individual model behaviors.

RNN produce poor outcomes when used on the entire dataset, GRU and LSTM networks function similarly on all feature sets. This shows that in order to avoid overfitting and improve generalization, careful feature selection is essential, as demonstrated by the sensitivity of RNNs to feature dimensions.

The 5th lag feature proves to be crucial in forecasting, especially for working days. Its significance lies in capturing patterns related to the same business day over a 5-day period. The 1st lag, representing the closest call or the call from the previous day, is also important. Monday emerges as a pivotal day for call reception, aligning with the observed pattern of increased call volumes on that day.

Abandoned rate, answer speed, and service level call are not merely metrics that accompany the call process—they are invaluable indicators that offer insights into customer behavior, system performance, and operational efficiency. By recognizing the importance of these features, organizations can better prepare for and adapt to fluctuations in call volumes, ultimately improving overall service quality and customer satisfaction.

5.3 Limitation

This work offers valuable insights into call volume forecasting. However, there are several areas for further exploration and refinement. Analyzing hourly data and incorporating additional features like top contact reasons, locations, and contact channels could offer a more granular understanding of temporal patterns and the factors influencing the time series.

Moreover, utilizing domain expertise within the company may greatly enhance the interpretability and contextual relevance of the model, resulting in better feature engineering and model selection. Extending the study's reach by examining a wider array of models and datasets would yield a more comprehensive understanding of the model's applicability and generalizability in other contexts.

Experimenting with a wider range of models may also reveal different strategies that, in some situations, perform better than or supplement current models. Resolving these issues would open the door for additional study and development with the goal of enhancing the call volume forecasting model's accuracy, interpretability, and generalizability over time.

REFERENCES

Alsharef, A., Aggarwal, K., Sonia, Kumar, M., & Mishra, A. (2022). Review of ML and AutoML Solutions to Forecast Time-Series Data. *Archives of Computational Methods in Engineering*, *29*(7), 5297-5311. https://doi.org/10.1007/s11831-022-09765-0

Baldon, N. (2019). *Time series Forecast of Call volume in Call Centre using Statistical and Machine Learning Methods* (Publication Number 2019:666) [Student thesis, DiVA. http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-265002

Box, G. E. P., & Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Holden-Day. https://books.google.co.th/books?id=1WVHAAAAMAAJ

de Boer, T. R., Mérelle, S., Bhulai, S., Gilissen, R., & van der Mei, R. (2023). Forecasting call and chat volumes at online helplines for mental health. *BMC Public Health*, *23*(1), 984. https://doi.org/10.1186/s12889-023-15887-2

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks : the official journal of the International Neural Network Society*, *18*, 602-610. https://doi.org/10.1016/j.neunet.2005.06.042

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-term Memory. *Neural computation*, *9*, 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Kanthanathan, C., Carty, G., Raja, M. A., & Ryan, C. (2020, 3-5 Dec. 2020). Recurrent Neural Network based Automated Workload Forecasting in a Contact Center. 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS),

Koole, G. M., & Li, S. (2023). A Practice-Oriented Overview of Call Center Workforce Planning. *Stochastic Systems*. https://doi.org/10.1287/stsy.2021.0008

Phi, M. (2018). *Illustrated Guide to LSTM's and GRU's: A step by step explanation*.

Saksonita, K., Jae sung, K., & Wan sup, C. (2022). A Comparison of Time Series Forecast Models for Predicting the Outliers Particles in Semiconductor Cleanroom [A Comparison of Time Series Forecast Models for Predicting the Outliers Particles in

Semiconductor Cleanroom]. *The Journal of Korean Institute of Information Technology*, *20*(11), 137-146. https://doi.org/10.14801/jkiit.2022.20.11.137

Services, C. o. C.-D. o. P. (2023, December 6). *Citizen Service Request (CSR) Call Center Calls*. City of Cincinnati - Department of Public Services. https://data.cincinnati-oh.gov/Efficient-Service-Delivery/Citizen-Service-Request-CSR-Call-Center-Calls/k2qr-ck2v

Shu-guang, H., Li, L., & Er-shi, Q. (2007, 9-11 June 2007). Study on the Continuous Quality Improvement of Telecommunication Call Centers Based on Data Mining. 2007 International Conference on Service Systems and Service Management,

Singh, A., Kotiyal, V., Sharma, S., Nagar, J., & Lee, C.-C. (2020). A Machine Learning Approach to Predict the Average Localization Error With Applications to Wireless Sensor Networks. *IEEE Access*, *8*, 208253-208263. https://doi.org/10.1109/ACCESS.2020.3038645

Takwi, F. (2021). *SERVICE OPERATIONS MANAGEMENT IMPROVING SERVICE DELIVERY*.

Tunnicliffe Wilson, G. (2016). Time Series Analysis: Forecasting and Control,5th Edition, by George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel and Greta M. Ljung, 2015. Published by John Wiley and Sons Inc., Hoboken, New Jersey, pp. 712. ISBN: 978-1-118-67502-1. *Journal of Time Series Analysis*, *37*, n/a-n/a. https://doi.org/10.1111/jtsa.12194

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, *1*(4), 339-356. https://doi.org/https://doi.org/10.1016/0893-6080(88)90007-X

VITA