UNIVERSITY OF SKÖVDE

Project Report

Building a Short-term Forecasting model to Predict the Call Volume of an Answering Service Company

Project in Data Science 15 credits
Spring term Year

Students: Cường Nguyễn, Georgios Kokolakis
Supervisor: Jonas Karlsson
Examiner: Addi Ait-Mlouk

Abstract

This project focuses on time series forecasting for call volume prediction in a call answering company. Four different models, namely XGBoost, RNN, SARIMAX, and SVR, were employed to assess their predictive capabilities and performance. The study utilised a dataset consisting of 2,783,712 incoming call records from January 1st to December 31st, 2022, encompassing 14 call groups. Evaluation metrics such as MSE, RMSE, and MAE were utilised to compare the models.

The results demonstrated the effectiveness of the models in accurately forecasting call volumes. XGBoost achieved an MSE of 0.002, RMSE of 0.04, and MAE of 0.03, indicating a high level of accuracy and a close alignment with the observed data. The RNN model performed similarly well, with a MSE of 0.00005, RMSE of 0.007, and MAE of 0.005, showcasing its ability to capture underlying patterns in the time series data effectively in the residual analysis; as well as its capabilities of outperforming all other models given the necessary resources. On the other hand, the SARIMAX model exhibited a RMSE of 0.039, MSE of 0.0015, and MAE of 0.025, indicating a lower level of accuracy compared to the RNN model but slightly better than XGBoost. The SVR model achieved the best results with a RMSE of 0.006, MSE of 0.00004, and MAE of 0.005, highlighting its strong predictive capabilities.

Limitations, including data availability and computational resources, were acknowledged, suggesting the need for diverse datasets and computational considerations. Future research could explore ensemble and hybrid models, incorporate additional features, and enhance interpretability for improved forecasting accuracy and decision-making. The practical implications of these findings are substantial for call answering companies, enabling efficient resource allocation and enhanced operational efficiency in handling call volumes.

Table of Contents

1. Introduction

Companies are continuously looking for innovative methods to boost customer happiness and optimise their operations in today's fiercely competitive business environment. Businesses can achieve considerable improvements in a number of crucial areas, including call service operations. Forecasting client demand and call traffic is essential for call service organisations to maintain service levels, utilise workers to the fullest extent possible, and cut expenses. Accurate forecasting can assist businesses in finding areas where processes can be automated or call routing can be improved. Such results are supported by the work of (Danese & Kalchschmidt, 2011).

A call service company's forecasting model is created by examining previous call volume data, spotting trends and patterns, and applying statistical methods to forecast call volume in the future. Advanced analytics and machine learning algorithms have made it possible for businesses to use strong tools to create extremely accurate forecasting models that account for a variety of characteristics, such as seasonality, day of the week, time of day, and holidays (Leszko, 2020).

As described by (Laurinavicius, 2023) there are some key benefits from the use of call answering services. To begin with, time efficiency; while it is not practical to answer every call, an unanswered call for a growing business would mean losing a potential customer. As a result, answering services would ensure that all calls are handled quickly and competently. In most cases, using a service is less expensive than hiring and training a full-time employee. Another advantage of using call answering services is cost effectiveness. The price for answering services is reasonable, especially since you just pay for the minutes you use; thus, it costs more to hire full-time staff to answer calls and live chat messages around-the-clock. Costs associated with the hiring process, onboarding, compensation, benefits, employment taxes, etc., must be taken into consideration. Last but not least, an answering service would offer improvement in customer service. An answering service offers round-the-clock coverage, making it possible for clients to get in touch with businesses at any time. Reputable answering services have qualified representatives who can answer questions from clients, handle orders, schedule appointments, or qualify warm leads.

Call centres are the front end of the organisation and are responsible for managing the relationship with customers. They can employ thousands of agents and are tasked with providing an optimal experience to the customer. Workload forecasting is a crucial part of many different systems and software tools are used to facilitate staff management and optimise the working schedule. Workforce Management (WFM) tools could include simple spreadsheets like MS Excel or special workforce management system solutions like Protime/Tamigo or ATOSS (Kellermayr-Scheucher et al., 2023). The main variable is the average time per call and the number of incoming calls. Scheduling is further subdivided into long, medium and short-term. In some cases, it is convenient to outsource their work to external companies named workforce management companies (Baldon, 2019).

Time series forecasting is a statistical technique that has been widely used to predict future values of a variable based on its past behaviour. Time series models are commonly used in various fields, including finance, economics, engineering, and healthcare. The call answering company used in this study is a mid-sized organisation that receives calls related to technical support and customer service. In this project we aim to build a short-term forecasting model for an answering service company called Samres (*Tjänster Inom Kundservice & Ärendehantering - Samres.*). The company offers round-the-clock customer support on behalf of individual professionals, entrepreneurs, and businesses. Their services include, among others, support through live chat, a virtual receptionist and appointment scheduling. Samres delivers business to business solutions for errand handling, support, intermediating services, and administration. They specialise in delivering customer contact services to organisations primarily in the traffic sector. They have also developed expertise in communication solutions for people with special needs and handle around 6 million errands annually in both the private and public sectors. The company uses a platform for workforce management named simTree, developed by deepNumbers (*DeepNUMBERS*). The platform gives the possibility to the user to upload data and makes predictions for agent occupancy during the week. The current method for the forecast is based on the average of historic data for the specific day, which also accounts for special days.

The data used in this study includes historical call volume data for the year 2022, with timestamp observations. This data will be used to train and evaluate different time series models to determine which one provides the most accurate forecasts.

In this paper, we will first discuss the importance of call volume forecasting in call centres and the challenges associated with it. We will then provide an overview of different time series models commonly used in forecasting, including the statistical model ARIMA (Autoregressive Integrated Moving Average), and machine learning-based models such as XGBoost. We will compare the performance of these models based on various evaluation metrics, including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and residual analysis (Cerqueira et al., 2020) (Belloto & Sokolovski, 1985).

The remainder of the paper is organised as follows. The Background section provides a literature review of call volume forecasting in call centres and the different time series models used in forecasting. The Methods section contains the Data Description which explains the data used in this study and the features involved. It also presents the pre-processing steps required to explore and fit the data into the models as well as the application of different time series models. The Results section provides a discussion of the findings and the performance of the models. Finally, the Conclusion section concludes the paper and offers recommendations for future research.

2. Background

Time series is a data format consisting of a time unit and values associated with it, which can be stored in two ways. Namely, a recording of time series as discrete points, such as financial-economical time series or as regular intervals such as hourly, daily, monthly, yearly, or even irregular intervals (Leszko, 2020). Time series modelling is a dynamic research area that focuses on collecting and studying past observations to develop an appropriate model to make forecasts. It is used to generate future values for the series and is essential for practical fields such as business, economics, finance in stock market prediction, science and engineering in weather forecasting. The study of time series modelling is ongoing. In one sentence, time series forecasting is the practice of making future predictions based on historical data using seasonal, trend, cyclic, and random components to generate a time series forecasting model (Naim et al., 2018). Research has been done to develop efficient models to improve forecasting accuracy, leading to the development of various models (Adhikari & Agrawal, 2013).

2.1 Time Series Models

2.1.1 ARIMA

The Autoregressive Integrated Moving Average (ARIMA) (Box and Jenkins 1990) model is one of the most well-liked and commonly applied stochastic time series models. This model's implementation starts with the fundamental presumption that the time series under consideration is linear and conforms to a specific, well-known statistical distribution, such as the normal distribution. The Autoregressive (AR), Moving Average (MA), and Autoregressive Moving Average (ARMA) models are subclasses of the ARIMA model. The AR model uses historical values with coefficients to predict future values. Meanwhile, the MA model makes assumptions of the current error term based on the past error terms. Specially, the ARMA is a combination of both AR and MA models which takes both past values and past error to forecast the future values. The ARIMA model is broadly suitable for series that do not exhibit the same behaviour over time, and historical data alone is not enough to predict the future behaviours. Box and Jenkins have developed the Seasonal ARIMA (SARIMA), a very successful version of the ARIMA model for forecasting seasonal time series (Box et al., 2015). The ARIMA model's success is mostly attributable to its ability to simply represent a variety of time series types and the related Box-Jenkins approach for the best model building process (Adhikari & Agrawal, 2013).

A practical method for creating ARIMA models was created by Box and Jenkins based on prior work, and it has a significant influence on time series analysis and forecasting applications. Three iterative steps—model identification, parameter estimate, and diagnostic testing—make up the Box-Jenkins approach (Box et al., 2015). The fundamental premise of model identification is that a time series should theoretically possess some autocorrelation qualities if it is produced by an ARIMA process. One or more potential models for the given time series can frequently be found by comparing the actual autocorrelation patterns with the theoretical ones. Box and Jenkins suggested using the sample data's autocorrelation and partial autocorrelation functions as the fundamental tools for determining the ARIMA model's order (G. P. Zhang, 2003).

Data transformation is frequently required during the identification stage in order to make the time series stationary. Building an ARIMA model that is effective for forecasting requires stationarity. The statistical properties of a stationary time series, such as the mean and the autocorrelation structure, remain constant over time. Before an ARIMA model can be fitted, differencing and power transformation are frequently used on the data to eliminate the trend and stabilise the variance when the observed time series exhibits trend and heteroscedasticity.

Estimating the model parameters is simple once a preliminary model has been defined. The parameters are estimated in a way that minimises the total amount of mistakes. An approach for nonlinear optimization can be used to accomplish this. The diagnostic evaluation of the model's suitability is the final stage of model construction. In essence, this is being done to see if the model's assumptions on the errors are true. The goodness of fit of the model that is being considered provisionally to the historical data can be evaluated using a number of diagnostic statistics and plots of the residuals. The procedures of parameter estimation and model verification should be repeated after choosing a new tentative model if the first one is insufficient. Alternative model(s) may be suggested with the aid of diagnostic data. Once a good model has been chosen, this three-step process is often repeated multiple times. The chosen model can then be applied to make predictions (G. P. Zhang, 2003).

SARIMAX

SARIMAX (Hyndman and Athanasopoulos 2018) is another time series model that is known to covariate with the data at hand can be integrated into other time series models to improve the prediction performance of future values. In the time series modelling process, using an exogenous variable refers to the insertion of an external input to a model. The application of SARIMAX can be found in the research of (M. Xie et al., 2013) where it was developed to describe relevant pricing trends and forecast the next-day price based on past data of Elspot prices and relative variables in Sweden. The model performed well as its Mean Absolute Percentage Error (MAPE) is considered very good. On the other hand, The SARIMAX model is not without restrictions. One apparent example is that it only considers one-day effects of exogenous factors without taking further time-lag effects into account. However, long-term time-lag effects are unavoidable since it takes time for a market to react to external developments. To deal with delayed effects, a vector autoregressive model might be used. If long-term equilibriums exist among level stationary variables, a vector error correction model based on vector autoregressive might be created. (Pradhan & Bhat, 2009).

2.1.2 Artificial Neural Networks

In the field of time series forecasting, artificial neural networks (ANNs) have recently gotten more and more attention. Although primarily inspired by biology, ANNs have now been successfully used in a variety of contexts, particularly for predicting and classification applications. When used to solve time series forecasting problems, ANNs shine because of their innate capacity to perform non-linear modelling without making any assumptions about the statistical distribution that the observations would follow. Based on the provided data, the relevant model is adaptively created. ANNs are data-driven and self-adaptive by nature as a result of this (Adhikari & Agrawal, 2013). Even if the sample data contains noisy information, ANNs may frequently correctly guess the unseen portion of a population after learning the data supplied to them. Additionally, ANNs are universal functional approximators. Artificial neural networks are model free dynamics, which signifies that the model structure is not explicitly defined by mathematical equations or assumptions. Instead, the network learns the underlying patterns and relationships directly from the data. This characteristic makes ANNs suitable for modelling complex systems where the underlying dynamics are not fully understood or difficult to describe using explicit mathematical models (Beckmann et al., 2023). They are among the computational intelligence systems that have been frequently employed for forecasting and approximation purposes. The fact that ANN models are universal approximators and can accurately approximate a broad range of functions is one of their main benefits over other classes of nonlinear models (Khashei & Bijari, 2010).

Given their benefits, it is not unexpected that artificial neural networks have drawn a lot of interest in time series forecasting. As (Y. Chen et al., 2005) and (Giordano et al., 2007) suggested, artificial neural networks are a strong alternative to several conventional time series models. There may be numerous approaches to define an ANN structure that corresponds to the problem because an ANN structure is not always the same for a particular problem. Depending on the issue, more than one hidden layer, feedforward or feedback connections, or in some situations, direct connections between input and output layer may be acceptable. Numerous attempts have been made to automatically create neural network architectures. Early techniques include algorithms for construction and pruning. EPNet and the NeuroEvolution of Augmenting Topologies (NEAT) are two ways to optimise ANN design and weights. The work of (B.-T. Zhang et al., 1997), who presented a method of evolutionary induction of the sparse neural trees, served as inspiration for using a tree to depict a NN-like model. The architecture and weights of higher order sigma-pi neural networks were evolved using genetic programming and breeder genetic algorithms, respectively, based on the representation of neural trees. This paper will deal with one category of ANNs presented below.

Recurrent Neural Networks

Although in the areas of robustness, efficiency and automation, Recurrent Neural Networks (RNNs) have a long way to go compared to exponential smoothing or autoregressive integrated moving average (ARIMA). RNNs have become a competitive forecasting method as most notably shown in the winning method of the M4 competition (Makridakis Competition) which was a large-scale competition held in 2018 for time series forecasting (Makridakis et al., 2020). They have become especially well-liked in the field of natural language processing. RNNs are universal approximators, just like ANNs. Contrary to ANNs, recurrent cells' feedback loops naturally address the temporal dependencies and temporal order of the sequences (Schäfer & Zimmermann, 2006).

Every RNN is made up of several RNN units. The Elman RNN (ERNN) cell, the long short-term memory (LSTM) cell, and the gated recurrent unit (GRU) cell are the three most often used RNN units for tasks involving sequence modelling. In addition to these, new variations such as depth-gated LSTM, clockwork RNNs, stochastic recurrent networks, and bidirectional RNNs have been developed (Hewamalage et al., 2021)

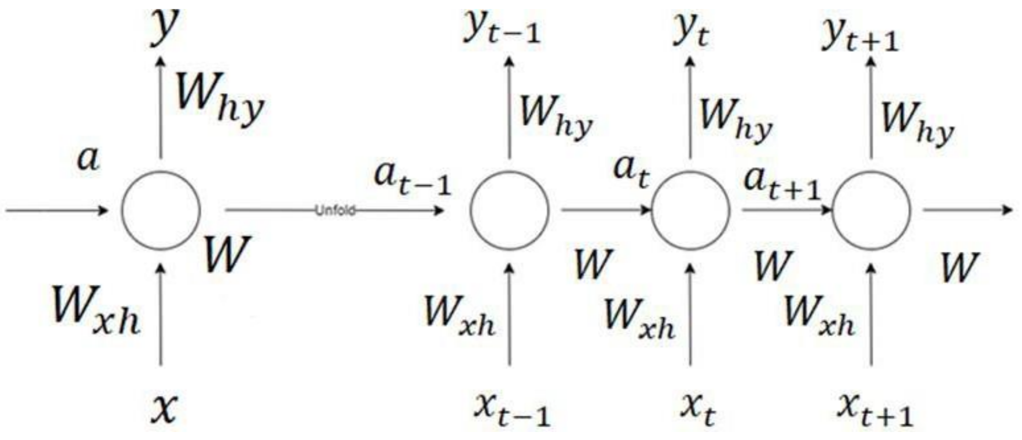


Figure 1. Architecture of an RNN

According to (Hiriyannaiah et al., 2020), the unrolled diagram on the right represents a visual depiction of the recurrent neural network (RNN) shown on Figure 1. The RNN is a type of neural network that is capable of processing sequential data by scanning it from left to right. In the diagram, there are three sets of weights: W_{xh}, W, and W_{hy}. The W_{xh} weights connect the input layer to the hidden layer, W represents the weights connecting the hidden layer to itself, and W_{hy} denotes the weights connecting the hidden layer to the output layer. These weights determine the strength of the connections between the layers. The activation parameter, denoted as a, represents the activation of the hidden layer. It carries information and is passed from one hidden layer to the next as the RNN progresses through the sequential data. Importantly, the RNN shares the same set of parameters (W_{xh}, W, and W_{hy}) across all time steps. This means that the weights used in the connections remain consistent throughout the scanning process. During each time step, when making a prediction at time t, the RNN considers not only the input x_t at time t but also the information from the previous time step (t-1). This is achieved through the activation parameter a and the weights W, which pass the information from the previously hidden layer to the current hidden layer.

2.1.3 XGBoost

A family of potent machine-learning methods known as gradient boosting machines has achieved notable success in a variety of real-world applications. They can be learned with regards to various loss functions, for example, and are extremely adaptable to the specific needs of the application (Natekin & Knoll, 2013). As part of that family, eXtreme Gradient Boosting was proposed by (Chen et al., 2015) and is also a boosting algorithm, meaning that it applies simple classifiers iteratively and aggregate the results to get better prediction results (Mayr et al., 2014). It is demonstrated in the literature that the XGBoost model has the characteristics of low computing complexity, quick running speed and good accuracy. The goal of the Boosting algorithm is to merge numerous weak classifiers into a powerful classifier (Wang & Guo, 2020). As a lifting tree model, XGBoost combines various tree models to create a powerful classifier (Figure 2).

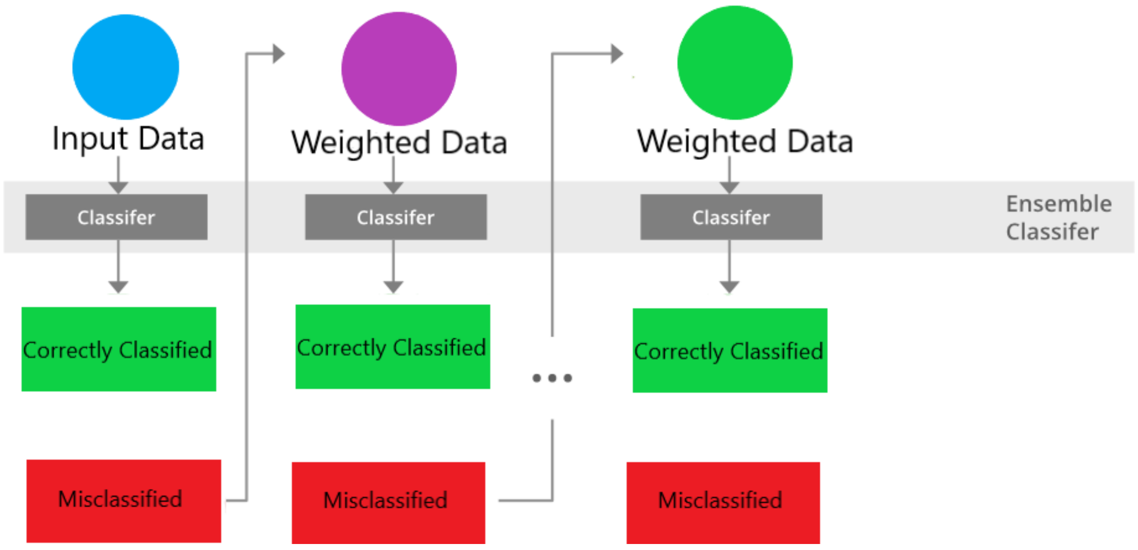


Figure 2. XGBoost model

The success of XGBoost is primarily due to its scalability in all situations. On a single machine, the system operates rather fast (Chen and Guestrin 2016), and in distributed or memory-constrained environments, it is scalable to billions of samples. XGBoost's scalability is made possible by several significant systems and algorithmic improvements on the existing gradient boosting machines. Among these innovations are a novel tree learning approach is used to handle sparse data, and handling instance weights in approximation tree learning is made possible via a theoretically supported weighted quantile sketch procedure. Learning is accelerated by parallel and distributed computing, which speeds up model exploration. What's more, XGBoost uses out-of-core processing to let data scientists process billions of instances on a desktop, without the help of distributed computing frameworks such as Tensorflow or Pytorch. Finally, combining these methods to create an end-to-end system that scales to even larger data sets with the least amount of cluster resources is even more appealing (Chen and Guestrin 2016).

2.1.4 SVM

According to (Adhikari & Agrawal, 2013), many support vector machine (SVM) forecasting models have been created by researchers. With the creation of Vapnik's SVM idea (Boser et al., 1992), significant progress in the field of time series forecasting took place. SVMs were initially developed to address pattern classification issues, but they have since found extensive use in a variety of different domains, including signal processing, function estimation, regression, and time series prediction issues. Utilising the structural risk minimization (SRM) concept, the goal of SVM is to identify a decision rule with a high generalizability potential (Jaramillo et al., 2017). Due to this, the SVM methodology has become a widely used method, particularly for issues involving time series forecasting (Adhikari & Agrawal, 2013). Only a portion of the training data points, known as the support vectors, are necessary for SVM to solve a given problem. Another crucial aspect of SVM is that in this case, training is equal to resolving a quadratic optimization problem with linear constraints. Therefore, unlike other conventional stochastic or neural network methods, the solution obtained by applying the SVM method is always distinct and globally optimal. The ability to independently regulate the quality and complexity of the solution, regardless of the dimension of the input space, is perhaps the most astounding feature of SVM (Jaramillo et al., 2017). With the aid of some specialised functions called support vector kernels, the input points are typically transferred to a high-dimensional feature space in SVM applications, which frequently produces good generalisation even in high dimensions. Support vector machines are generally acknowledged to be more accurate than traditional time series forecasting techniques and other forms of neural networks (Jaramillo et al., 2017). The basic concept of SVM implements nonlinear class boundaries using a linear model and some nonlinear mapping of the input vectors into the high-dimensional feature space. A linear model constructed in the new space is able to represent a non-linear decision boundary in the original space. In the new space, an Optimal Hyperplane is created. The Maximised margin is the maximum separation between the decision classes. The training examples that are closet to the maximum margin hyperplane are called Support vectors (Kim 2003).

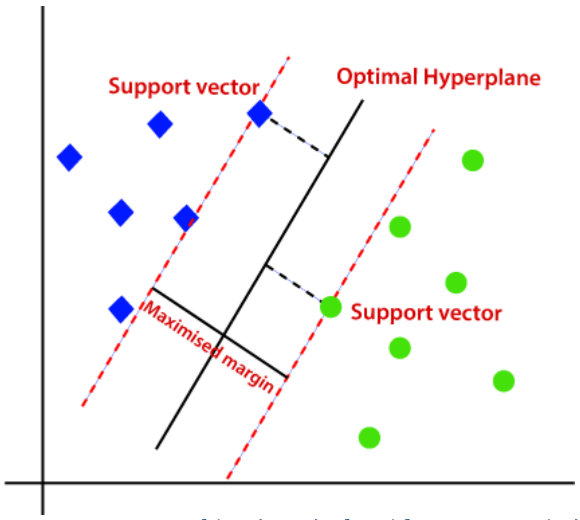


Figure 3. SVM algorithm (Support Vector Machine (SVM) Algorithm - Javatpoint)

Support Vector Regressor (SVR)

The SVM's (Boser et al., 1992) sparse solution and strong generalisation make it amenable to adjustment to regression issues. As presented by (Awad and Khanna 2015), by inserting the "tube", an ∑-insensitive zone surrounding the function, SVM generalisation to SVR is made possible. To discover the tube that best approximates the continuous-valued function while balancing model complexity and prediction error, this tube reformulates the optimization problem. SVR is more precisely described as an optimization problem by first constructing a convex-insensitive loss function to be reduced and locating the flattest tube that contains most of the training cases. As a result, using the loss function and the geometrical characteristics of the tube, a multi-objective function is created. The convex optimization is then done using the proper numerical optimization procedures, which has a unique solution. Support vectors, which are training samples that lie outside the tube's perimeter, are used to represent the hyperplane (Figure 3). In a supervised learning setting, the training and test data are assumed to be independent and identically distributed (iid), drawn from the same fixed but unknowable probability distribution function. As in SVM, the support vectors in SVR are the most significant instances that affect the shape of the tube.

Support vector regression is expected to perform well for time series analysis because of their strong generalisation ability and guarantee for global minima for given training data. As being examined in travel-time prediction, Support Vector Regressor (SVR) has shown effectiveness in time-series analysis (Chun-Hsin Wu et al., 2004). It performed admirably in the experiments since it can cover quickly and avoid local minimums during the training process. Additionally, the SVR model provides a better level of prediction accuracy and stability in short-term load forecasting (Y. Chen et al., 2017).

2.2 Evaluation Metrics

In this section, we discuss the evaluation metrics commonly used for regression models. These metrics provide valuable insights into the performance and accuracy of the model predictions. We will focus on Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Residual Analysis.

Mean Squared Error (MSE):

MSE (Equation 1) is a widely used evaluation metric for regression models. It measures the average squared difference between the predicted and actual values. By squaring the errors, MSE gives higher weight to larger errors. A lower MSE indicates better model performance, with zero being the best possible value. However, MSE can be sensitive to outliers (Kalyan Das et al., 2004).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2$$

$$n = number\ of\ \ data\ points$$

$$Y_i = observed\ values$$

$$\widehat{Y}_i = predicted\ values$$

Equation 1. MSE

Root Mean Squared Error (RMSE):

RMSE (Equation 2) is derived from MSE by taking the square root of the average squared difference between the predicted and actual values. It provides a measure of the average magnitude of the errors in the same units as the target variable. RMSE is particularly useful for interpreting the errors and comparing different models. Similar to MSE, a lower RMSE indicates better model performance (Cerqueira et al., 2020) (Chai & Draxler, 2014).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2}{n}}$$

Equation 2. RMSE

Mean Absolute Error (MAE):

MAE (Equation 3) measures the average absolute difference between the predicted and actual values. Unlike MSE and RMSE, MAE does not square the errors, making it less sensitive to outliers. MAE provides a straightforward interpretation of the error magnitude and is particularly useful when outliers are present in the data (Chai & Draxler, 2014).

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |Y_i - \widehat{Y}_i|$$

Equation 3. MAE

Residual Analysis:

Residual analysis involves examining the residuals, which are the differences between the predicted and actual values. It is a crucial step in evaluating regression models. By analysing the residuals, we can assess the model's performance in terms of bias, heteroscedasticity, and the presence of outliers. Various techniques such as scatter plots, histograms, and Q-Q plots can be used to visualise the residuals and identify any patterns or deviations from the model assumptions (Belloto & Sokolovski, 1985). Residual analysis helps us determine if the model captures the underlying patterns and if there are any systematic errors or unexplained variations. It provides insights into model adequacy and can guide further model improvements or refinements (Belloto & Sokolovski, 1985).

It is important to use multiple evaluation metrics in combination to obtain a comprehensive understanding of the model's performance. MSE, RMSE, and MAE provide quantitative measures of the prediction accuracy, while residual analysis offers a visual and diagnostic assessment of the model's fit to the data. By considering these evaluation metrics, researchers and practitioners can make informed decisions about the quality and effectiveness of their regression models (Cerqueira et al., 2020).

2.3 Related Work

Several studies have been conducted on time series forecasting, specifically focusing on call volume data in call service companies. The existing body of research provides valuable insights into various modelling techniques, evaluation metrics, and approaches for improving the accuracy of call volume predictions. This part of the report provides an overview of the key studies and their findings in the field of time series forecasting for call volume data.

In the work of (Leszko, 2020), the goal of the researcher is to produce time series forecasts of how many incoming calls relating to press-published ads are anticipated to arrive at a call centre over the course of 40 days. The results indicate that statistical models such as ARIMA and SARIMA have better metrics compared to the RNN, although the author points out that data used to perform these predictions might not have been sufficient. Finally, according to the findings, the SARIMAX algorithm is the one that produces the most promising results.

The thesis of (Baldon, 2019) investigates the use of machine learning approaches for call load forecasting. The author uses SARIMA as a benchmark in order to compare the model to Seasonal ANN and LSTM. The results of the thesis indicate that SARIMA performs better when it comes to predicting call volume on a daily basis compared to the Machine Learning models. On the other hand, both Seasonal ANN and LSTM outperform SARIMA when it comes to hourly predictions.

In their work, (I. Paliari et al., 2021) examine the application of modern Machine Learning methods, i.e., LSTM and XGBoost, on forecasting financial time series. Using the ARIMA model metrics as a benchmark to evaluate the performance of the Machine Learning Models, the authors reach the outcome that both contemporary ML models outperform the benchmark across the board. The findings of their research indicate that contemporary ML techniques are not only safe to use in stock market prediction, but also frequently outperform more conventional statistical methods.

Building on the above, this project aims to combine their findings and explore some of their limitations. Namely, this project aims to do a larger comparison of models, including the SARIMAX, XGBoost, SVM and RNN using LSTM. All models will be examined on our dataset and tested on their ability to predict the call volume. Moreover, given the results of the above thesis on an hourly basis, we will be applying the models on 15-minute intervals to both reaffirm the findings concerning short term forecasting and to further investigate the results over a much shorter period.

3. Methods

This section provides a comprehensive overview of the methodologies employed in the project, covering essential steps such as data description, pre-processing, resampling, and feature engineering. Additionally, it explores the application of prominent models discussed above, XGBoost, RNN-LSTM, SARIMAX and SVR for accurate and reliable time series forecasting. Furthermore, this section discusses crucial considerations like data visualisation, normalisation, denormalisation, and the crucial step of splitting the dataset for training and testing purposes.

3.1 Data Description

Following is a general description of the dataset used for the experiments. For the purposes of this project, we were provided with a dataset of records of incoming calls in the company's call centre, taking place from January 1st to December 31st, 2022. The dataset has 2,783,712 incoming call records from 14 call groups in total and some features. The records included six features, i.e., ServiceID, Time Stamp, Answered, Waiting Answered, Talking and Waiting Abandoned. The time stamp comes in the format of (YYYY-MM-DD HH:MM:SS) to mark each unique record starting from January 1st to December 31st, 2022. The integer ServiceID column describes different customer's occupations (call groups), thus call centre's agents with the same specialty could handle the call accordingly. Relating to the capacity limitation of the workforce, the binary Answered feature determines if the call is answered or not. For answered calls, the client's waiting time in the queue is represented in seconds by the Waiting Answered feature. The Talking column clarifies the duration of the call in seconds. Finally, the Waiting Abandoned feature is specialised for the waiting time in seconds in the queue until the call is abandoned by the customer before receiving an answer from one of the call centre's operators.

3.2 Data Pre-processing

In the data pre-processing stage, the dataset obtained from the call service company is prepared for further analysis and modelling. The initial dataset is loaded, and its shape is examined to gain an understanding of the data size. The first few rows of the dataset are inspected to identify the available columns and their corresponding information.

To ensure data quality and consistency, unnecessary columns are dropped from the dataset. Additionally, column names are modified to provide more meaningful descriptions. For example, the column 'ServiceID' is renamed as 'callgroupid', 'Time_stamp' as 'timestamp', 'Answered' as 'answered', 'WaitingAnswered' as 'gap', 'Talking' as 'duration', and

'WaitingAbandoned' as 'drop'. This renaming improves the clarity and interpretability of the dataset. To handle categorical data, a replace_dict is created to map specific call group IDs to numeric values. This transformation allows the models to effectively process the categorical information. The 'timestamp' column is converted to a datetime format enabling efficient time-based operations and analysis. The dataset is checked for missing values. The absence of missing values in the dataset ensures the reliability of subsequent analyses and modelling. Finally, the 'timestamp' column is set as the index of the dataset to facilitate time series operations and resampling.

3.3 Resampling and Feature Engineering

Resampling the dataset at regular time intervals is crucial for capturing the underlying patterns and fluctuations in call volume data. In this study, the dataset is resampled at 15-minute intervals. This resampling step aggregates the data within each 15-minute interval and applies aggregate functions to compute relevant features. The specified interval is suitable for the purposes of short-term forecasting while at same time it allows us to maintain enough data size to train our models. The resampled data frame is constructed by aggregating the call volume, total duration, and total gap within each 15-minute interval. The 'call volume' column represents the count of calls in each interval, 'total_duration' represents the sum of call durations, and 'gap' represents the sum of waiting times between calls. These aggregated features provide valuable insights into the overall call activity and waiting times within specific time intervals.

To gain further granularity and analyse the call volume by call group, a pivot table is created. This pivot table groups the data by the resampled time intervals and call group IDs, allowing for a comprehensive understanding of call volumes specific to each call group within the defined time intervals. This information is crucial for evaluating the performance of different call groups and identifying any variations in call volumes across time.

Additional features are incorporated to enhance the forecasting models. The 'unix_timestamp' column is generated by converting the resampled timestamp to Unix timestamps, representing the number of seconds. This numeric representation of time can be beneficial for capturing time-based patterns and trends. Moreover, the 'Day of Week' column is created to categorise each resampled interval based on the corresponding day of the week. This feature enables the analysis of weekly patterns and variations in call volumes.

3.4 Data visualisation

To observe a higher level of granularity, we aggregated data points not only on weekly but also daily and hourly scales. At the weekly scale, each data point was illustrated by 672 logs at interval scale. For the daily scale, we represented an interval of 96 records as one data point. Meanwhile, only 4 records at interval scales were used to obtain one datapoint at hourly scale. There is a sharp drop in call volume between July and September in Figure 4 (weekly). Moreover, daily and hourly scales in Figure 5 and Figure 6 show visible multiple seasonalities, and some peaks are observed, except 2 outliers in May and December. Given the volume of data, we opted to disregard these outliers as it does not critically influence the overall outcome. Additionally, in Figure 7 we plot the number of calls during a week period on different days of the month with each data observation representing call volume at one interval scale. From Figure 7, the same trends were examined: the number of calls inclined during the middle of the day and decreased at the end of the day. Last but not least, the graph illustrates that the number of calls on the weekend were much lower compared to weekdays.

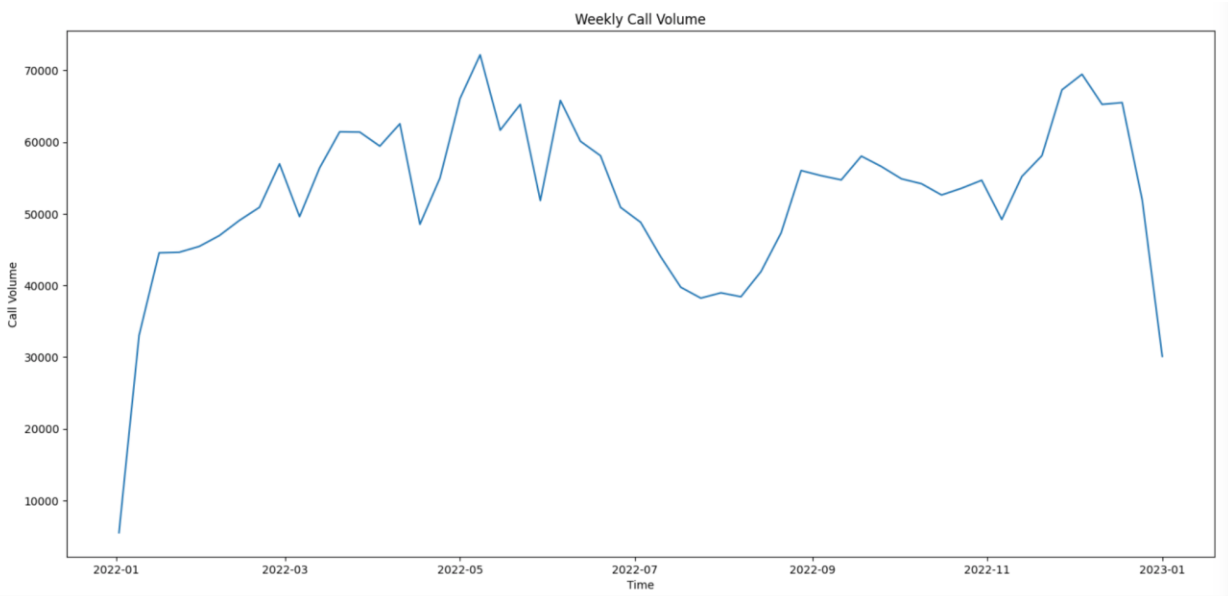


Figure 4. Weekly patterns

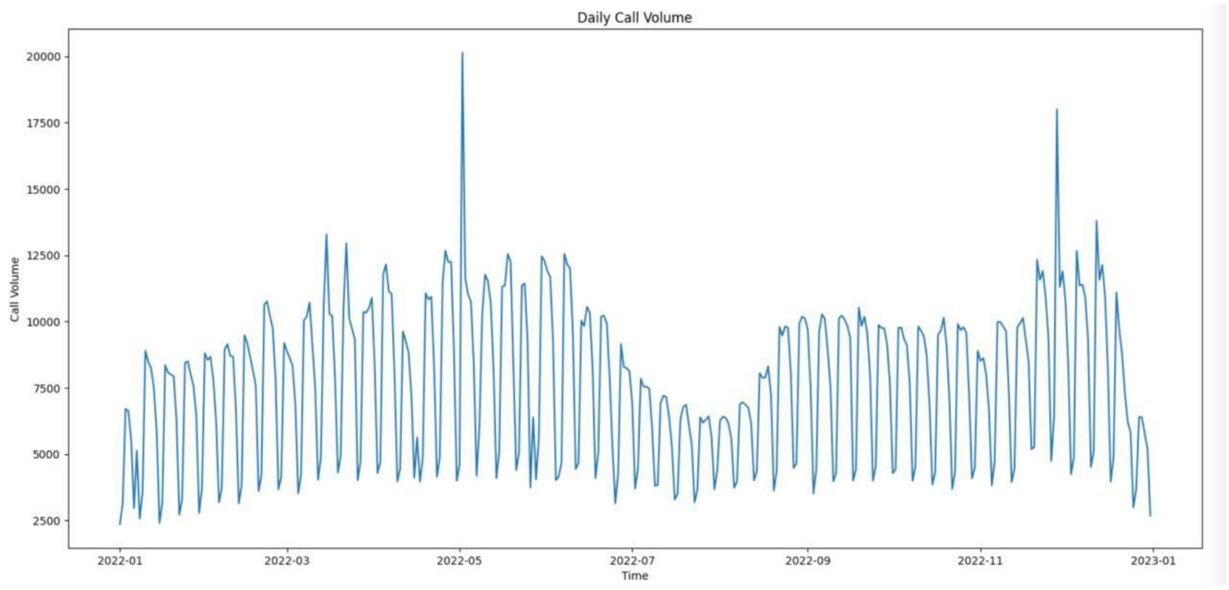


Figure 5. Daily patterns

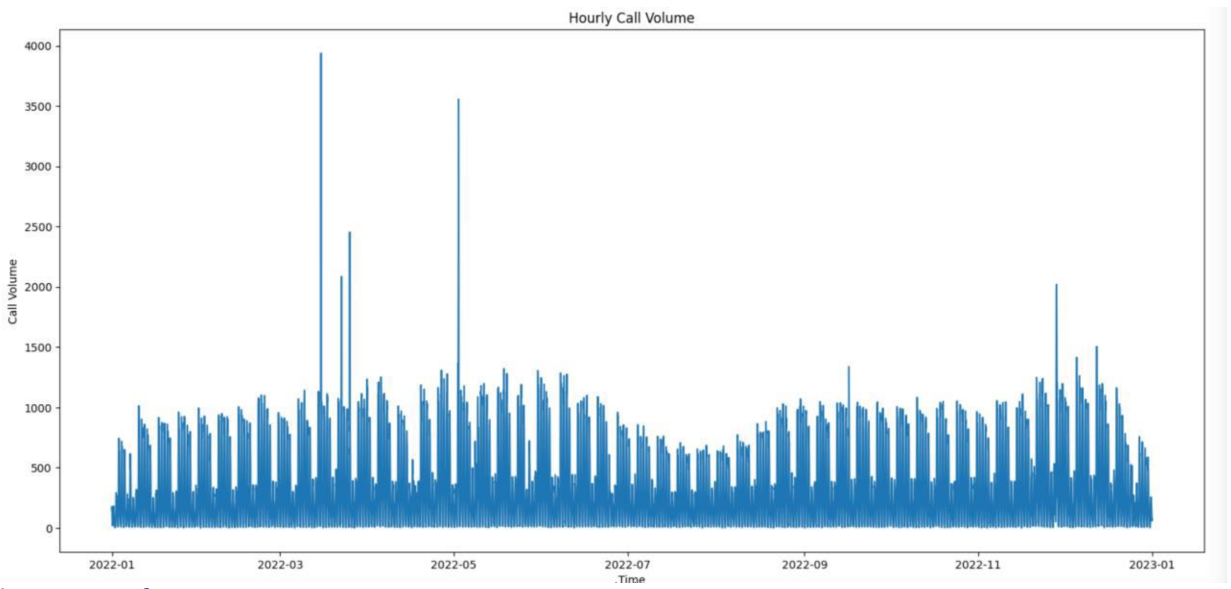


Figure 6. Hourly patterns

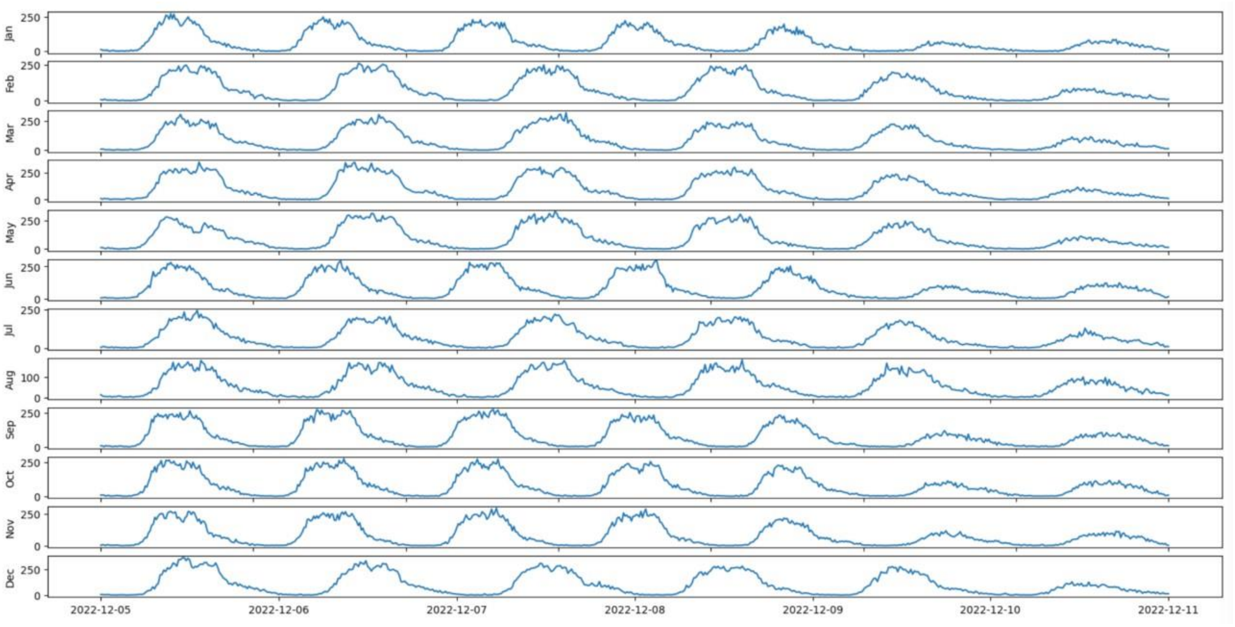


Figure 7. Arbitrary daily pattern in each month

3.5 Splitting the Dataset

To ensure the effectiveness of machine learning models, it is crucial to evaluate their performance on unseen data (Tan et al., 2021). In the context of time series forecasting for call volume data, an important step is to split the dataset into training and testing sets. This separation allows us to train the model on a subset of the data and assess its performance on unseen instances.

Since the observations in the dataset were acquired sequentially, with each observation being recorded at a precise point in time. The time series' values might be influenced by previous values and may display patterns, trends, or seasonality. As a result, while projecting future values, we wanted to verify that the training data comprises of observations that happened temporally before the testing data by separating the dataset using indexing. This enables the model to properly anticipate future time points by learning from prior patterns and dynamics in the data. The dataset was split into training and testing data using indexing.

The first instances of data points are assigned to the training data, while the remaining instances are assigned to the testing data. The margins of the data remain the same with 80% of the data used for training and 20% for testing.

By splitting the data into training and testing sets, we can simulate real-world scenarios where the model encounters new, unseen instances during evaluation. This process helps to assess the model's generalisation capability and its ability to make accurate predictions on unseen data. Additionally, it allows us to detect potential issues such as overfitting, where the model memorises the training data but fails to generalise well to new data (Tan et al., 2021).

3.6 Normalising and Denormalising the Data

The normalisation and denormalization of data play a crucial role in various data analysis and modelling tasks, including time series forecasting and it is applied to the Unix timestamp feature of our Dataset. One commonly used technique for data normalisation is the MinMaxScaler, which scales the data to a specific range, typically between 0 and 1. The MinMaxScaler has several benefits and is widely employed in practice (Shaheen et al., 2020). Figure 8 and Equation 4 make the theoretical foundation of normalisation clear to the reader (Muhammad Ali & Faraj, 2014). If casting the data to the range of 0 to 1 is necessary, then:

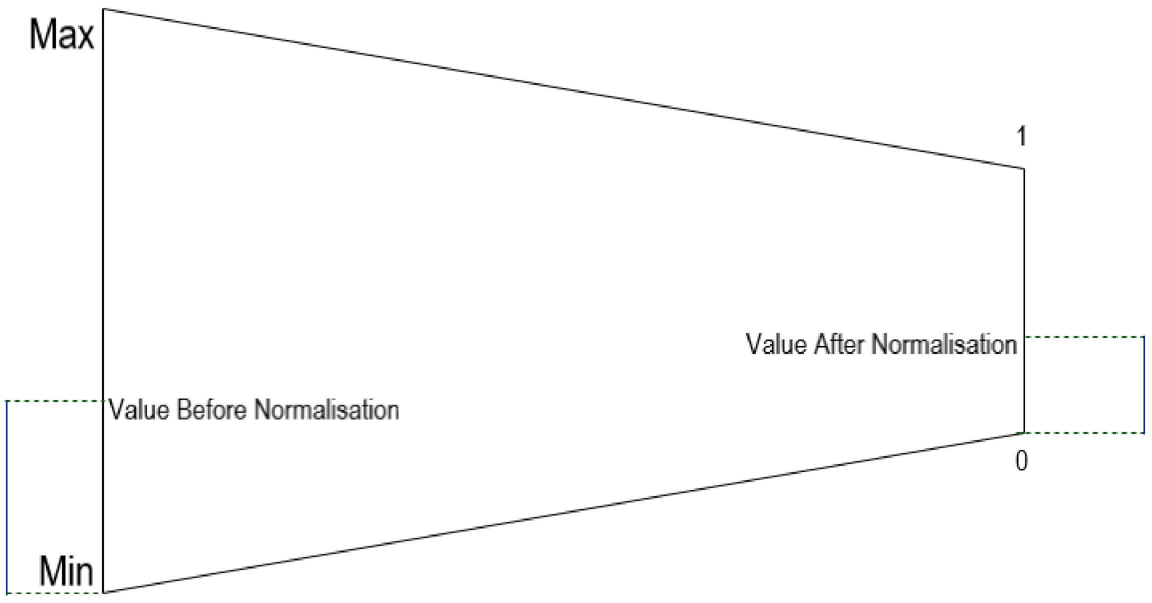


Figure 8. Normalisation

$Let\ x\ be\ the\ Value\ Before\ Normalisation\ and\ y\ the\ Value\ After\ Normalisation, then\ from\ trigonometry$:

$$\frac{y - 0}{1 - 0} = \frac{x - Min}{Max - Min} \Leftrightarrow y = \frac{x - Min}{Max - Min}$$

Equation 4. Normalisation

Normalisation is important for several reasons. Firstly, it brings all features or variables to a similar scale, which is particularly useful when dealing with data that have different units or measurement scales. By scaling the data to a standardised range, we ensure that no single feature dominates the modelling process solely due to its larger numerical values. This prevents bias in the model's learning process and allows for fair comparisons and interpretations of the importance of different features (Shaheen et al., 2020).

In the context of time series forecasting, normalisation is particularly valuable. Time series data often exhibit dynamic and varying ranges, making it difficult for models to capture the patterns and relationships effectively. By normalising the data, we bring the time series to a consistent scale, allowing the models to discern and learn meaningful patterns without being overwhelmed by large or small values (Bhanja & Das, 2018).

After training the model and obtaining predictions on the normalised data, denormalization becomes necessary to interpret and evaluate the results properly. Denormalization is the process of transforming the predictions back to their original scale, aligning them with the original data distribution (Muhammad Ali & Faraj, 2014). The MinMaxScaler provides a convenient method for Denormalising the predictions by reversing the scaling transformation. Denormalization is essential for understanding the practical implications of the predictions. It allows us to interpret the forecasted values in their original units, such as call volumes or durations, facilitating decision-making processes based on real-world context. Additionally, denormalization is crucial when evaluating the performance of the model using metrics such as MSE or MAE, as these metrics are calculated in the original scale of the data (Muhammad Ali & Faraj, 2014).

3.7 XGBoost Model

Now the data is split and normalised, the application of the XGBoost algorithm for time series forecasting can take place. First an instance of the XGBRegressor class is created, which is a specific implementation of the XGBoost algorithm for regression problems. This model is chosen due to its capability to capture complex patterns and relationships in the data, making it suitable for predicting the call volume in a time series problem, as demonstrated in the background section (Mayr et al., 2014) (Chen & Guestrin, 2016) (Chen et al., 2015)

The model is trained using the fit() method, which optimises the model's parameters to minimise the prediction errors. GridsearchCV is used to fine-tune the model, i.e., the Number of boosting rounds is set to 100, the Learning rate of the model to 0.1, the Maximum depth of each tree is set to 6, although it should be noted that even when set to 7 or 8 the results were similar. Last but not least, both the Fraction of features and fraction of samples used for each tree was set to 1. During the training process, the model learns the underlying patterns and dependencies in the training data, enabling it to make accurate predictions. Once the model is trained, it is ready to make predictions on the test data. The predict() method is used to generate predictions for the call volume based on the input features in X_test. The predicted values are stored in the y_pred variable.

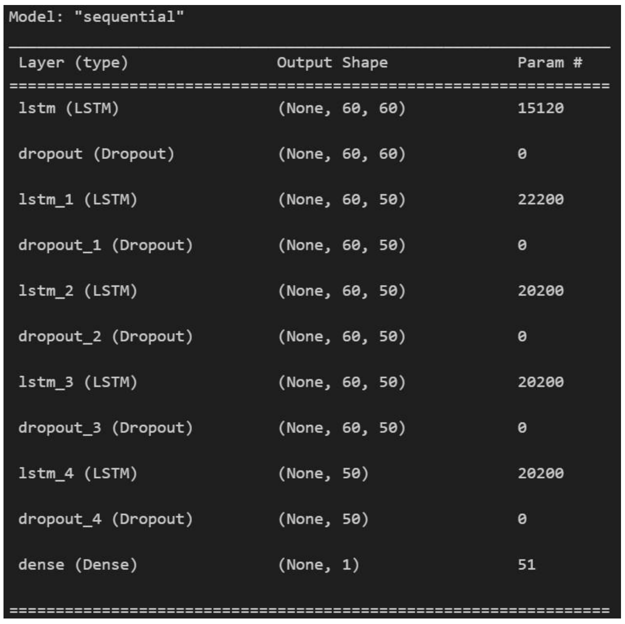
To evaluate the performance of the XGBoost model, the mean squared error (MSE) is calculated by comparing the predicted call volume (y_pred_denormalized) with the actual call volume (y_test). The mean_squared_error() function from scikit-learn is utilised for this purpose. The MSE provides a measure of the average squared difference between the predicted and actual values, giving insight into the accuracy of the model's predictions.

3.8 RNN-LSTM model

Having split the data, we first need to reshape it appropriately to be able to feed the data to the RNN. LSTM models require the input data to be in a specific format, namely a three-dimensional array with dimensions [samples, timesteps, features]. The data is reshaped for both the training and testing datasets. For the training data, a loop is used to iterate over the scaled training data. In each iteration, a sequence of previous data points is extracted, along with the corresponding call volume values. These sequences are then appended to the X_train array. Additionally, the corresponding target call volume value at index i is appended to the y_train array. The same process is repeated for the testing data. Sequences of previous data points and their corresponding call volume values are extracted, and they are appended to the X_test array. The target call volume values for the testing data are appended to the y_test array. By reshaping the data in this manner, the LSTM model can effectively learn the temporal dependencies and patterns present in the time series data (I. Paliari et al., 2021). The sequences of previous data points serve as input features, and the corresponding call volume values act as the target variable for the model to learn from.

The model begins by creating an instance of the Sequential class, which allows us to build the model layer by layer. The Sequential model is a linear stack of layers, where each layer is added one after another. The parameters of the RNN are tuned according to the medium size of the dataset, through thorough experimentation. The first layer added to the model is an LSTM layer. LSTMs are defined with a specific number of units, which determines the dimensionality of the output space. In this case, the LSTM layer has 60 units as an appropriate input for the amount of the data available. The return_sequences parameter is set to True, indicating that the LSTM layer should return the sequence of outputs rather than a single output. The input_shape parameter specifies the shape of the input data. A Dropout layer is added after each LSTM layer. Dropout is a regularisation technique that randomly drops a fraction of the input units during training, which helps prevent overfitting. In this code, a dropout rate of 0.2 is used, meaning that 20% of the input units are randomly set to 0 during each training step.

The model continues with additional LSTM and Dropout layers, each following the same pattern. Multiple LSTM layers allow the model to learn hierarchical representations and capture complex temporal patterns in the data. The accuracy of the model did not improve significantly after adding 4 LSTM and 4 Dropout layers, thus given the computational limitations of this project; using different instances and experimenting with the layers, it was determined that the number of layers that yielded the best results were 5 LSTM and 5 Dropout layers. After the last LSTM layer, a Dense layer is added. This layer consists of a single unit and uses a linear activation function as we are dealing with a regression task (Figure 9). It serves as the output layer of the model, responsible for predicting the next call volume based on the previous input sequences.


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 60)            15120

 dropout (Dropout)           (None, 60, 60)            0

 lstm_1 (LSTM)               (None, 60, 50)            22200

 dropout_1 (Dropout)         (None, 60, 50)            0

 lstm_2 (LSTM)               (None, 60, 50)            20200

 dropout_2 (Dropout)         (None, 60, 50)            0

 lstm_3 (LSTM)               (None, 60, 50)            20200

 dropout_3 (Dropout)         (None, 60, 50)            0

 lstm_4 (LSTM)               (None, 50)                20200

 dropout_4 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
```
Figure 9. Neural Network Summary

Once the model is constructed, it needs to be compiled before training. The compile() method is used to specify the optimizer, loss function, and metrics to be used during the training process. As mentioned, the task concerns regression in this case, thus the Adam optimizer is chosen, which is an adaptive learning rate optimization algorithm. The mean squared error (MSE) is selected as the loss function, as it measures the average squared difference between the predicted and actual values. Additionally, the mean absolute error (MAE) is included as a metric to track during training.

The model is then trained using the fit() method. The X_train and y_train datasets are used as input, with a specified number of epochs and batch size. The number of epochs determines the number of times the model will iterate over the entire training dataset, while the batch size indicates the number of samples used in each training update. During training, the model learns to minimise the mean squared error loss and optimise its weights and biases. Due to the medium size of the dataset, the model was tested in instances of epochs ranging from 10 to 20 with the batch size set to 32. In those instances, it was observed that the accuracy of the model did not increase significantly as the epochs increased, thus the final input of the epochs is set to 15.

The validation_data parameter is set to (X_test, y_test), allowing the model's performance to be evaluated on the test dataset during training. This provides insight into how well the model generalises to unseen data. Once training is complete, the model's performance is evaluated using the evaluate() method. The X_test and y_test datasets are used to calculate the loss and MAE on the test set.

3.9 SARIMAX Model

An instance of the SARIMAX class can be created by providing the training data and a host of model configuration parameters. A SARIMAX model comprises Auto Regressive component (AR), Differencing (I), and Moving Average (MA) for the seasonal component of the series, as well as additional seasonality period parameters, and exogenous variable. These seasonal supplemental parameters are Seasonal Auto-Regressive order, Seasonal Difference order, Seasonal Moving Average order, and the number of time steps for a single seasonal period.

According to (Liu et al. 2016), time series is defined as a series of quantitative measurements made at different times. Time is assumed as a discrete variable, X_t denotes the observation at time t, and ϵ_t denotes the zero-mean random noise term at time t. The MA(q) model considers the process: $X_t = \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \epsilon_t$ where β_i is the coefficient. Similar to MA(q) models, Autoregression model, denoted by AR(p), satisfies $X_t = \sum^{p}{}_{i=1} \alpha_i X_{t-i} + \epsilon_t$. In other words, each X_t is assumed to be a noisy linear mixture of the preceding p observations. However, time series data are not always realisations of a stationary process. Some of them may include deterministic patterns. Consider using the differential approach to deal with such significant serial correlations. For instance, the first order differences of X_t can be computed by $\nabla X_t = X_t + X_{t-1}$, and the second order differences of X_t by $\nabla^2 X_t = \nabla X_t - \nabla X_{t-1}$ and so on.

To examine the data stationary, we decomposed the data into 3 components: Trend, Seasonality and Noise

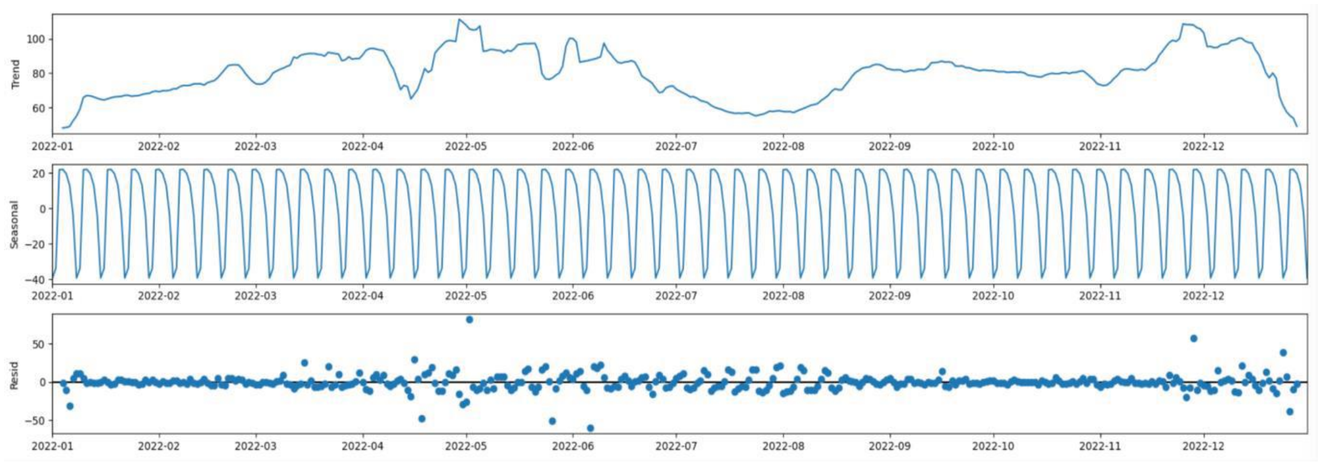


Figure 10. Series plots

From Figure 10, although there is no trend and little noise spotted, the data is showing daily seasonality which equal to 96 observations (s=96). Additionally, we applied Augmented Dickey-Fuller test (ADF) on the time series and got the p-value very close to zero, and the test statistic value is -32.012 which is highly negative, hence we can reject the null hypothesis of non-stationary. Since the time series is stationary, the difference is set to 0. The choice of exogenous variable was based on the day of the week.

After identifying the Differencing, Seasonality values and exogenous variable, we continued to discover optimal values for non-seasonal and non-seasonal AR order and MA order. In this case, Grid Search with various combinations of values was applied for the SARIMAX model, and the choices were made based on the lowest Akaike Information Criterion (AIC) value. Interpreting from the result, the best set for p,q,P,Q parameters are 1,1,1,1 respectively.

As the model's parameters was constructed, we built and then trained the model SARIMAX using fit() based on the train set. After using the model for predicting the test data, RMSE, MSE and MAE were used to evaluate the accuracy of the model.

3.10 SVR Model

For the SVR Model, we first transform the train, test data to the form [batch, timesteps]. As we want to use the first 4 timesteps as input features (1 hour), and the output would be the data for the 5th timestep, hence the timesteps was set to 5. Since the SVR requires a 2D input matrix or tensor, nested list comprehension was applied to the train and test sets.

To retrieve the appropriate parameters, we perform GridSearchCV and Cross-Validation on a grid set of parameters, and the best set is evaluated based on the mean square errors metric. According to the evaluation result, we specified the kernel to linear as it allows for non-linear relationships between the input variables and the target variables. The gamma parameter was set to 0.01 illustrated that only nearby points would have significant influence on the decision boundary. The value 0.1 of C parameter allowed the model to have more errors on training data. Finally, the best value of epsilon parameter was set to 0.01 made the model more sensitive to error.

After constructing the model, it was trained using x_train, y_train. Then, the SVR model was used to make predictions on test sets. Subsequently, the data was scaled back to original form and its plot would be used together with the predicted plot to evaluate the result. Furthermore, RMSE, MSE and MAE are used to estimate the model accuracy.

4. Results and Discussion

The aim of this study was to predict the call volume of a call answering company on a 15-minute interval basis using different models: XGBoost, Recurrent Neural Network (RNN), SARIMAX and SVR. In the process, the aptness of those models to make dependable predictions was tested in order to compare them. The performance of the models was evaluated based on several evaluation metrics including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and a residual analysis.

4.1 XGBoost and RNN

The XGBoost model demonstrated excellent predictive capabilities, yielding an MSE of 0.002, RMSE of 0.04, and MAE of 0.03 (Table 1). These results indicate a high level of accuracy in predicting the call volume, with minimal error between the predicted and actual values. The low MSE and RMSE values suggest that the model's predictions closely align with the observed data, indicating a good fit. The MAE value further supports the model's accuracy, indicating that, on average, the predicted call volumes deviate by only 0.03 from the actual values.

Similarly, the RNN model also performed remarkably well in predicting the call volume. It achieved a MAE of 0.00005, RMSE of 0.007, and MAE of 0.005 (Table 1), indicating an even better average deviation between the predicted and actual values compared to the XGBoost model. The lower MAE value suggests that the RNN model captured the underlying patterns in the time series data effectively, resulting in highly accurate predictions.

Metrics / Models	MSE	RMSE	MAE
XGBoost	0. 002	0.04	0.03
RNN	0.00005	0.007	0.005

Table 1. Results of XGBoost & RNN

These results highlight the efficacy of both the XGBoost and RNN models in forecasting call volume for the call answering company. The small prediction errors indicate the models' ability to capture the dynamics and trends in the time series data, allowing for reliable and precise predictions. Below are the Residual graphs of both models, Figure 11 and Figure 12 depict the residuals of XGBoost and RNN respectively. It is important to keep in mind that the input was transformed into a 2D tensor using layered list comprehension and time steps in order to fit the XGBoost model, which affected the number of records in both sets and accounts for the difference in the scale of the x axis below.

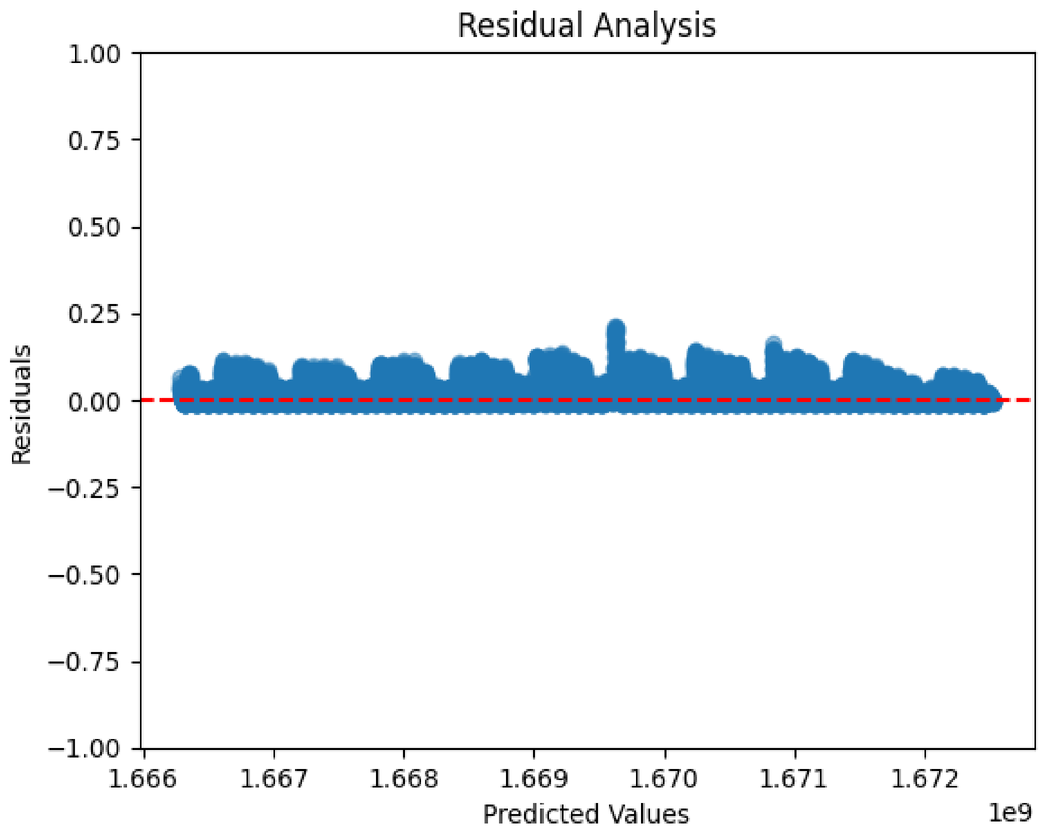


Figure 11. XGBoost Residual Plot

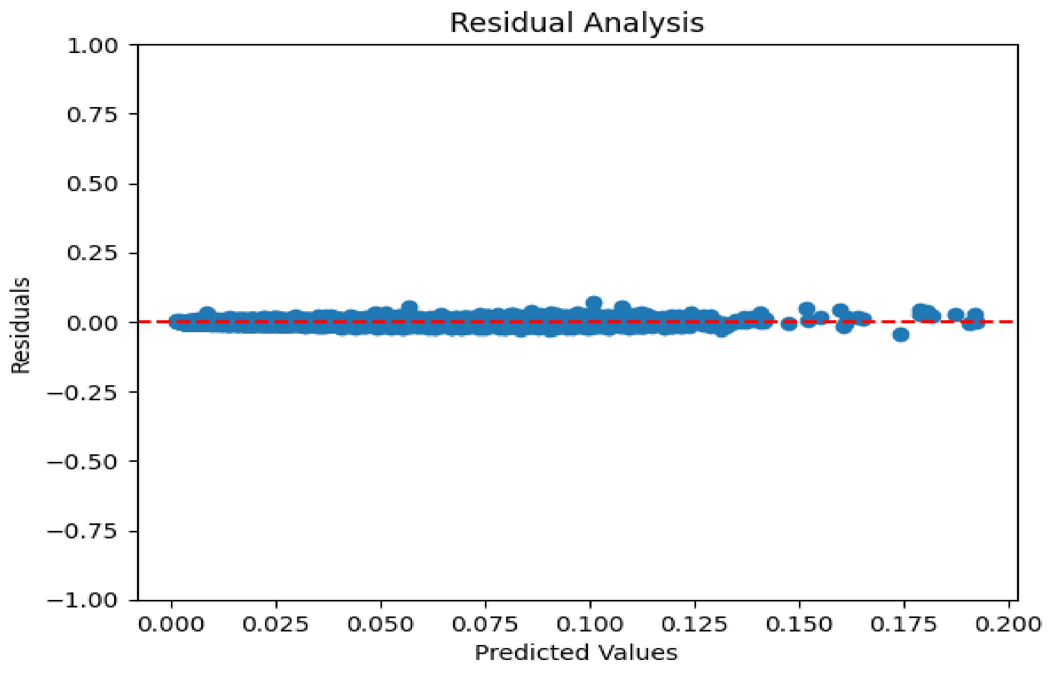


Figure 12. RNN Residual Plot

According to (L. Chen et al., 2020), a residual graph where most values are around 0 indicates that the model's predictions are close to the actual values on average. In other words, the model is performing well in capturing the underlying patterns and trends in the data. The residuals, which are the differences between the predicted and actual values, being centred around 0 suggests that the model is making relatively accurate predictions. A residual value of 0 indicates that the predicted value matches the actual value exactly. Positive residuals indicate that the predicted values are higher than the actual values, while negative residuals indicate that the predicted values are lower than the actual values. Having residuals close to 0 means that the model is neither consistently overestimating nor underestimating the target variable. The above presented graphs clearly demonstrate the proximity of our predictions to the actual values in both models. The RNN-LSTM model outperforms the XGBoost in terms of metrics however the most important difference is the residual analysis plots. While the XGBoost residual plot demonstrates divergence across the x-values, implying potential instability, the RNN exhibits consistently low residuals for both high and low predicted values. Consequently, the RNN proves to be a stable choice across a wider range of predicted values, enhancing its overall performance and reliability.

Comparing the performance of the two models, it is important to consider their strengths and limitations. Due to its exceptional accuracy, efficacy, and adaptability, XGBoost has become the most used technique for creating predictive models. When compared to other algorithms, XGBoost provides several benefits, including the capacity to manage missing data, highly parallelizable code, and huge and complicated datasets (Tarwidi et al., 2023). It also performs well in scenarios with relatively smaller datasets and provides fast and efficient predictions (Chen & Guestrin, 2016). On the other hand, the RNN model leverages its ability to capture temporal dependencies and is particularly effective in modelling complex sequential patterns, with the prospect of performing even better with the use of larger datasets (Schäfer & Zimmermann, 2006). However, it may require more computational resources and time for training compared to XGBoost.

These results have practical implications for call answering companies. Accurate call volume predictions can assist in optimising resource allocation, managing staffing levels, and improving overall operational efficiency (Laurinavicius, 2023). By leveraging these predictive models, companies can better anticipate call volume fluctuations and adjust their workforce accordingly, ensuring that customer needs are met efficiently.

The superior performance of both models indicates their potential application in real-world scenarios for call volume forecasting. The XGBoost model provides a computationally efficient and interpretable solution, while the RNN model leverages its ability to capture temporal dependencies in the data. The choice between the two models depends on specific requirements, computational constraints, and interpretability needs (Barrie Wetherill et al., 1986). In our case, the choice between those models, depends on the needs and computational limitations of the company. The XGBoost would offer a quick with low computational requirements solution, while the RNN outperforms the XGBoost at the cost of greater resources.

4.2 SARIMAX and SVR

The comparison of the SVR and SARIMAX models in the context of short-term forecasting yielded interesting findings. The SARIMAX model, with a RMSE of 0.05, MSE of 0.002 and MAE 0.03 outperformed the SVR model, which had an RMSE of 0.006, MSE of 0.00004, MAE of 0.005 (Table 2), in forecasting the short-term behaviour of the time series.

Metrics / Models	MSE	RMSE	MAE
SARIMAX	0.0015	0.039	0.025
SVR	0.00004	0.006	0.005

Table 2. Results of SARIMAX & SVR

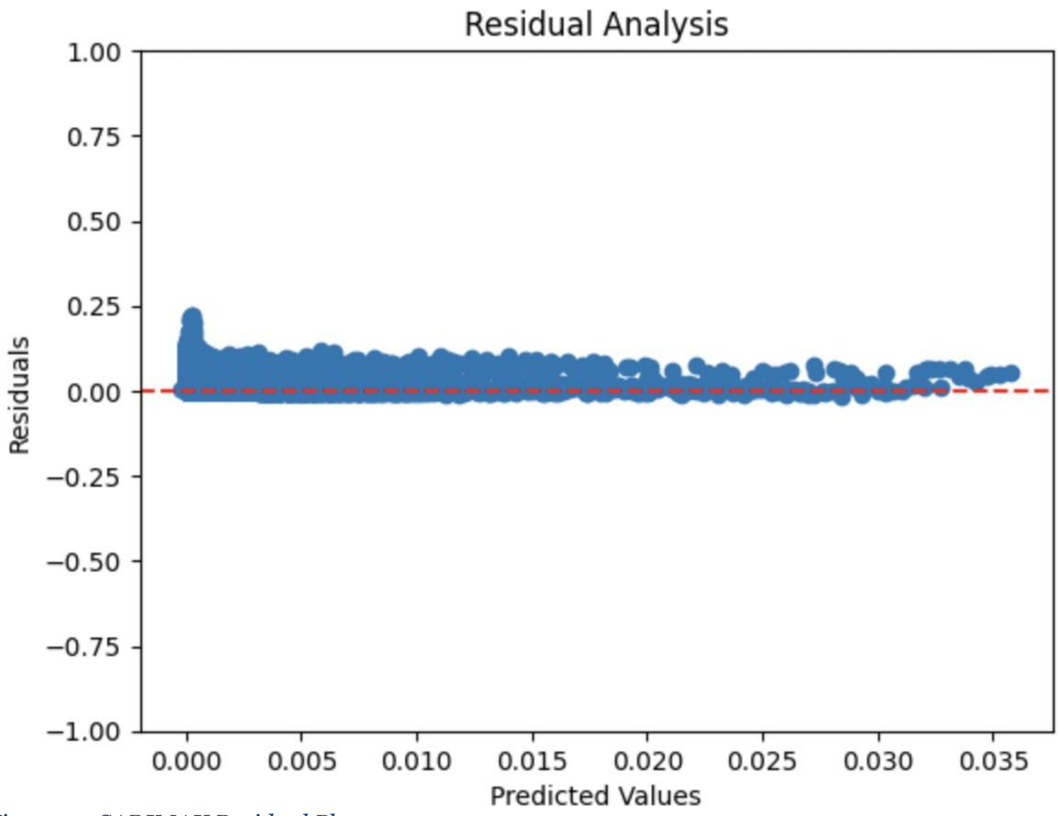


Figure 13. SARIMAX Residual Plot

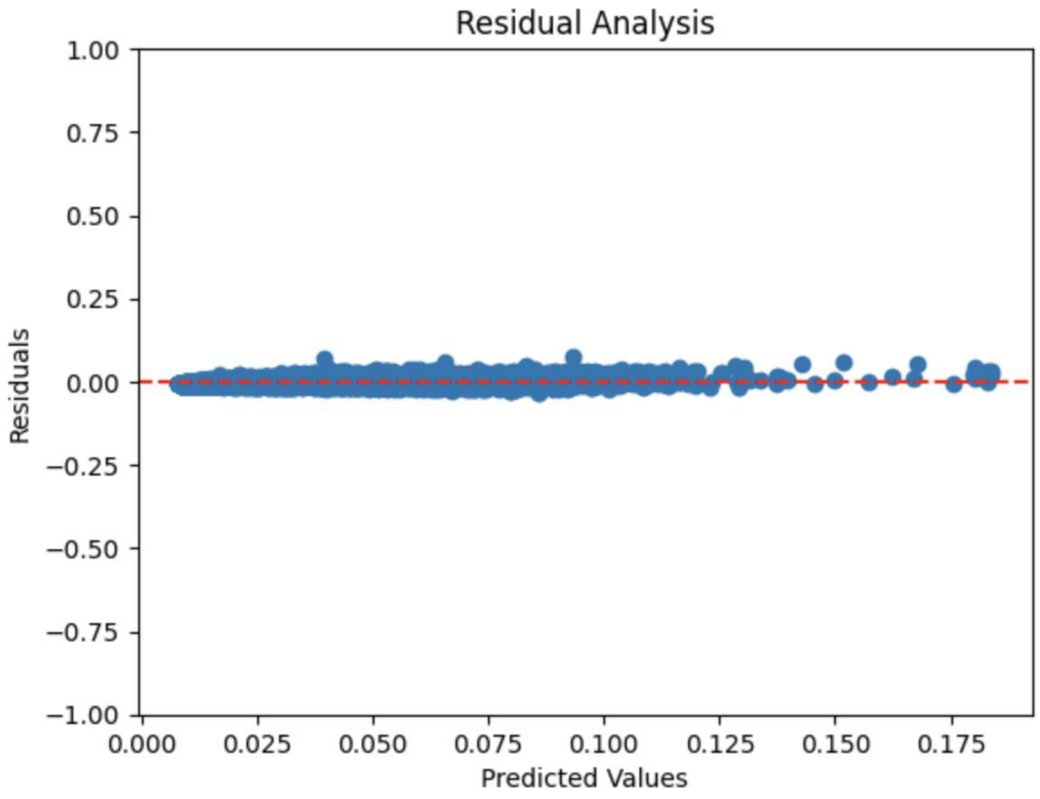


Figure 14. SVR Residual Plot

Figure 13 and Figure 14 thrive in situations when the underlying patterns in the data are not effectively represented by the SARIMAX model compared to the SVR model.

In the context of this project, not only the time complexity was found to be better but also the SARIMAX model result evaluations were outperformed by the SVR model. In instances where the time series data has a more linear connection and seasonality and other complicated patterns are best represented by a linear regression technique, the SVR model may be advantageous.

Additionally, although the SVR model with a radial basis function (RBF) kernel has been used to predict short-term time series data, achieving good performance in comparison with trivial predictors such as the seasonal mean (SM) or a random walk (RW)(Lippi et al., 2013), this project exhibits a promising result of applying SVR model using linear function.

These findings show that the SVR model gives more accurate short-term predictions by using support vectors and optimizing the margin around the projected values. This finding emphasizes the significance of using non-linear models, such as SVR, when dealing with short-term forecasting problems regarding the provided data.

4.3 Discussion

The above results provide a comparison between models in time series forecasting, with the intent to find the one(s) that better approaches our problem in terms of model accuracy. The models with the most promising results found in literature are explored. The one(s) that score the highest in the aforementioned metrics, will be deployed and monitored. Overall, this study aims to provide valuable insights into the application of time series models in call volume forecasting for call centres. The results of this study can help call centre managers make informed decisions regarding resource allocation and service level agreements, leading to improved customer satisfaction and operational efficiency.

The proposed models demonstrated promising results as presented in Table 3. The RNN-LSTM and SVR were able to capture the underlying patterns of the data as the residual analysis reveals; with the expectation (both from literature and through experimentation) that the neural network will yield even better results with bigger data size for training. The company can now review the models and decide which one fits their needs and capabilities better as all predict remarkably well the call volume, but each has a drawback. Namely in the case of SVR and RNN-LSTM the high accuracy comes at the cost of resources as both require computational power and RNN requires large data size to be trained. XGBoost might offer a lower accuracy however it requires a much shorter execution time. SARIMAX was outperformed in every way and the resources it requires are significant making it an unsuitable choice for our purposes.

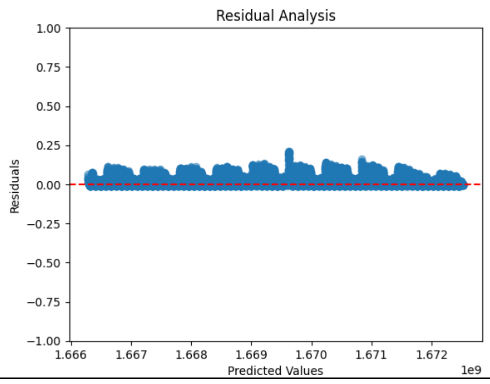
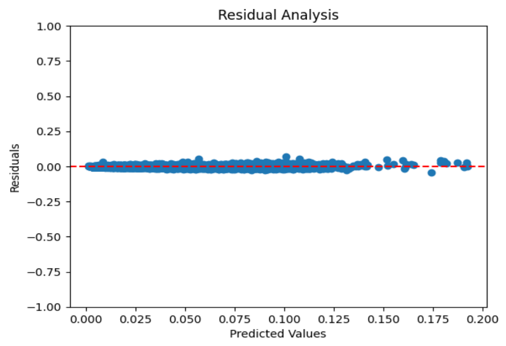
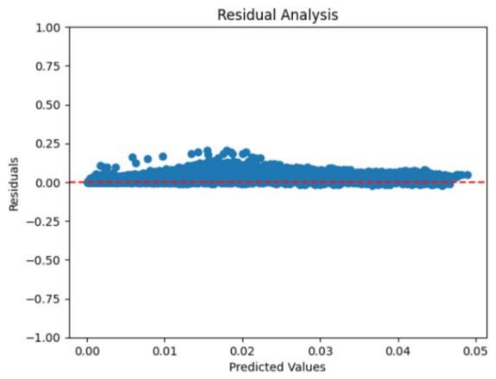
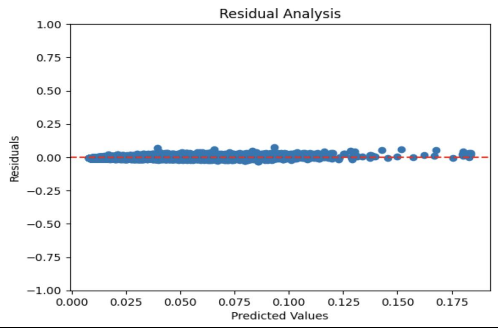
Metrics / Models	MSE	RMSE	MAE	Residual Analysis
XGBoost	0. 002	0.04	0.03	
RNN	0.00005	0.007	0.005	
SARIMAX	0.0015	0.039	0.025	
SVR	0.00004	0.006	0.005	

Table 3. Model comparison

The practical implications of these findings are significant for call answering companies. Accurate call volume predictions enable efficient resource allocation, improved staffing levels, and enhanced operational efficiency. By leveraging these predictive models, companies can proactively anticipate call volume fluctuations and ensure prompt and effective customer service.

5. Conclusion

This study focused on predicting call volume for a call answering company using different models, namely XGBoost, RNN, SARIMAX, and SVR. The results demonstrated the effectiveness of SVR and RNN in accurately forecasting call volumes. The SVR model exhibited excellent predictive capabilities, with minimal error between predicted and actual values, while the RNN model captured the underlying patterns in the time series data effectively, resulting in even more accurate predictions. Although we expected a promising result from the SARIMAX model, it yielded similar results with the XGBoost and required a greater execution time. Given the scope of this project and the resources available there were several limitations that could be addressed for improved results.

First and foremost, the data availability; the study utilised a dataset of incoming call records from a single call answering company for the year 2022. The findings may be specific to this dataset and may not generalise to other call centres or different time periods. The inclusion of data from multiple call centres or a longer time span could provide a broader perspective on call volume patterns. At the same time, we would be able to monitor the performance of the models on unseen data and reach more definite conclusions when comparing them. The performance of the models may vary across different call centres with distinct characteristics and call volume patterns. The findings of this study should be validated in diverse call centre settings to ensure their generalizability and reliability.

Computational Resources are another major limitation of this study. The RNN model, due to its nature of capturing temporal dependencies, may require significant computational resources and longer training times compared to XGBoost. This limitation could impact its practical applicability in scenarios with limited computational capabilities or time constraints which applies to our case.

While this study provided valuable insights into call volume prediction using XGBoost, RNN, SARIMAX and SVM models, there are several avenues for future research and improvement. Our research was limited into four models with promising results. While the desired outcome was reached in terms of accuracy, some literature suggests that ensemble, hybrid models could achieve even more in that department (Wang & Guo, 2020) (Bhati et al., 2021). Combining the predictions from different models might mitigate the limitations of individual models and lead to improved forecasting performance.

Moreover, investigating additional features that could influence call volume, such as holidays, promotional events, or external factors, may improve the accuracy of the predictions. Including such factors as input variables in the models can enhance their forecasting capabilities.

Last but not least, exploring methods to enhance the interpretability of the models' predictions can provide valuable insights for decision-making. Techniques such as feature importance analysis or model visualisation can aid in understanding the underlying factors driving call volume fluctuations.

Division of Work

The division of work is conducted within the guidelines of the course project. Namely, the Introduction and Background sections are written by both participants, as the literature, and the goal are common. In the Methods section the Data Description, Resampling and Feature Engineering, Data Visualisation, Splitting the Dataset, Normalising and Dernomalising subsections are a product of both individuals, however, each participant develops separate models in our effort to answer the research question. Cường Nguyễn applies the Seasonal Autoregressive Integrated Moving Average with Exogenous Variables (SARIMAX) statistical model as well as regression using the Support Vector Regressor (SVR). Georgios Kokolakis handles the application of the gradient boosting algorithm eXtreme Gradient Boosting (XGBoost) and the development of a recurrent neural network (RNN) using long short-term memory (LSTM). The participants then separately explain their findings in the Results and Discussion section. Finally, the Conclusion section is worked jointly for closing remarks.

References

Adhikari, R., & Agrawal, R. K. (2013). An introductory study on time series modeling and forecasting. ArXiv Preprint ArXiv:1302.6613.

Awad, M., & Khanna, R. (2015). Support Vector Regression. In M. Awad & R. Khanna (Eds.), Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers (pp. 67–80). Apress. https://doi.org/10.1007/978-1-4302-5990-9_4

Baldon, N. (2019). Time series forecast of call volume in call centre using statistical and machine learning methods.

Barrie Wetherill, G., Duncombe, P., Kenward, M., Köllerström, J., Paul, S. R., & Vowden, B. J. (1986). Choosing a regression model. In G. B. Wetherill, P. Duncombe, M. Kenward, J. Köllerström, S. R. Paul, & B. J. Vowden (Eds.), Regression Analysis with Applications (pp. 230–248). Springer Netherlands. https://doi.org/10.1007/978-94-009-4105-2_11

Beckmann, P., Köstner, G., & Hipólito, I. (2023). Rejecting Cognitivism: Computational Phenomenology for Deep Learning.

Belloto, J., & Sokolovski, T. (1985). Residual analysis in regression. American Journal of Pharmaceutical Education, 49(3), 295–303.

Bhanja, S., & Das, A. (2018). Impact of data normalization on deep neural network for time series forecasting. ArXiv Preprint ArXiv:1812.05519.

Bhati, B. S., Chugh, G., Al-Turjman, F., & Bhati, N. S. (2021). An improved ensemble based intrusion detection technique using XGBoost. Transactions on Emerging Telecommunications Technologies, 32(6), e4076. https://doi.org/10.1002/ett.4076

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. 144–152.

Box, G. E. P., & Jenkins, G. (1990). Time Series Analysis, Forecasting and Control. Holden-Day, Inc.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time Series Analysis: Forecasting and Control. Wiley. https://books.google.se/books?id=rNt5CgAAQBAJ

Cerqueira, V., Torgo, L., & Mozetič, I. (2020). Evaluating time series forecasting models: An empirical study on performance estimation methods. Machine Learning, 109(11), 1997–2028. https://doi.org/10.1007/s10994-020-05910-7

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE). Geoscientific Model Development Discussions, 7(1), 1525–1534.

Chen, L., Wu, L., Hong, R., Zhang, K., & Wang, M. (2020). Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. Proceedings of the AAAI Conference on Artificial Intelligence, 34(01), 27–34. https://doi.org/10.1609/aaai.v34i01.5330

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., & Zhou, T. (2015). Xgboost: Extreme gradient boosting. R Package Version 0.4-2, 1(4), 1–4.

Chen, Y., Xu, P., Chu, Y., Li, W., Wu, Y., Ni, L., Bao, Y., & Wang, K. (2017). Short-term electrical load forecasting using the Support Vector Regression (SVR) model to calculate the demand response baseline for office buildings. Applied Energy, 195, 659–670. https://doi.org/10.1016/j.apenergy.2017.03.034

Chen, Y., Yang, B., Dong, J., & Abraham, A. (2005). Time-series forecasting using flexible neural tree model. Information Sciences, 174(3–4), 219–235.

Chun-Hsin Wu, Jan-Ming Ho, & D. T. Lee. (2004). Travel-time prediction with support vector regression. IEEE Transactions on Intelligent Transportation Systems, 5(4), 276–281. https://doi.org/10.1109/TITS.2004.837813

Danese, P., & Kalchschmidt, M. (2011). The impact of forecasting on companies' performance: Analysis in a multivariate setting. Leading Edge of Inventory Research, 133(1), 458–469.

https://doi.org/10.1016/j.ijpe.2010.04.016

DeepNUMBERS. (n.d.). Retrieved May 3, 2023, from https://deepnumbers.se/

Giordano, F., La Rocca, M., & Perna, C. (2007). Forecasting nonlinear time series with neural network sieve bootstrap. Computational Statistics & Data Analysis, 51(8), 3871–3884.

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. International Journal of Forecasting, 37(1), 388–427. https://doi.org/10.1016/j.ijforecast.2020.06.008

Hiriyannaiah, S., Srinivas, A. M. D., Shetty, G. K., G.M., S., & Srinivasa, K. G. (2020). Chapter 4—A computationally intelligent agent for detecting fake news using generative adversarial networks. In S. Bhattacharyya, V. Snášel, D. Gupta, & A. Khanna (Eds.), Hybrid Computational Intelligence (pp. 69–96). Academic Press. https://doi.org/10.1016/B978-0-12-818699-2.00004-4

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and practice. OTexts.

I. Paliari, A. Karanikola, & S. Kotsiantis. (2021). A comparison of the optimized LSTM, XGBOOST and ARIMA in Time Series forecasting. 2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA), 1–7. https://doi.org/10.1109/IISA52424.2021.9555520

J. Jaramillo, J. D. Velasquez, & C. J. Franco. (2017). Research in Financial Time Series Forecasting with SVM: Contributions from Literature. IEEE Latin America Transactions, 15(1), 145–153. https://doi.org/10.1109/TLA.2017.7827918

Kalyan Das, Jiming Jiang, & J. N. K. Rao. (2004). Mean squared error of empirical predictor. The Annals of Statistics, 32(2), 818–840. https://doi.org/10.1214/009053604000000201

Kellermayr-Scheucher, M., Niedermeier, M., & Brandtner, P. (2023). Applications and Perceptions of Workforce Management Systems for Warehouse Operation—Results and Findings from Expert Interviews. CENTERIS – International Conference on ENTERprise Information Systems / ProjMAN – International Conference on Project MANagement / HCist – International Conference on Health and Social Care Information Systems and Technologies 2022, 219, 255–262. https://doi.org/10.1016/j.procs.2023.01.288

Khashei, M., & Bijari, M. (2010). An artificial neural network (p,d,q) model for timeseries forecasting. Expert Systems with Applications, 37(1), 479–489. https://doi.org/10.1016/j.eswa.2009.05.044

Kim, K. (2003). Financial time series forecasting using support vector machines. Support Vector Machines, 55(1), 307–319. https://doi.org/10.1016/S0925-2312(03)00372-2

Laurinavicius, T. (2023, April 12). Best Answering Services (2023) – Forbes Advisor. https://www.forbes.com/advisor/business/software/best-answering-services/

Leszko, D. (2020). Time series forecasting for a call center in a Warsaw holding company. http://hdl.handle.net/10362/102939

Liu, C., Hoi, S. C., Zhao, P., & Sun, J. (2016). Online arima algorithms for time series prediction. 30(1).

M. Lippi, M. Bertini, & P. Frasconi. (2013). Short-Term Traffic Flow Forecasting: An Experimental Comparison of Time-Series Analysis and Supervised Learning. IEEE Transactions on Intelligent Transportation Systems, 14(2), 871–882. https://doi.org/10.1109/TITS.2013.2247040

M. Xie, C. Sandels, K. Zhu, & L. Nordström. (2013). A seasonal ARIMA model with exogenous variables for elspot electricity prices in Sweden. 2013 10th International Conference on the European Energy Market (EEM), 1–4. https://doi.org/10.1109/EEM.2013.6607293

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. M4 Competition, 36(1), 54–74. https://doi.org/10.1016/j.ijforecast.2019.04.014

Mayr, A., Binder, H., Gefeller, O., & Schmid, M. (2014). The evolution of boosting algorithms. Methods of Information in Medicine, 53(06), 419–427.

Muhammad Ali, P., & Faraj, R. (2014). Data Normalization and Standardization: A Technical Report. https://doi.org/10.13140/RG.2.2.28948.04489

Naim, I., Mahara, T., & Idrisi, A. R. (2018). Effective Short-Term Forecasting for Daily Time Series with Complex Seasonal Patterns. International Conference on Computational Intelligence and Data Science, 132, 1832–1841. https://doi.org/10.1016/j.procs.2018.05.136

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. Frontiers in Neurorobotics, 7. https://www.frontiersin.org/articles/10.3389/fnbot.2013.00021

Pradhan, K. C., & Bhat, K. S. (2009). An empirical analysis of price discovery, causality and forecasting in the nifty futures markets. International Research Journal of Finance and Economics, 1(26), 83–92. Scopus.

Schäfer, A. M., & Zimmermann, H. G. (2006). Recurrent Neural Networks Are Universal Approximators. In S. D. Kollias, A. Stafylopatis, W. Duch, & E. Oja (Eds.), Artificial Neural Networks – ICANN 2006 (pp. 632–640). Springer Berlin Heidelberg.

Shaheen, H., Agarwal, S., & Ranjan, P. (2020). MinMaxScaler Binary PSO for Feature Selection. In A. K. Luhach, J. A. Kosa, R. C. Poonia, X.-Z. Gao, & D. Singh (Eds.), First International Conference on Sustainable Technologies for Computational Intelligence (pp. 705–716). Springer Singapore.

Tan, J., Yang, J., Wu, S., Chen, G., & Zhao, J. (2021). A critical look at the current train/test split in machine learning. ArXiv Preprint ArXiv:2106.04525.

Tarwidi, D., Pudjaprasetya, S. R., Adytia, D., & Apri, M. (2023). An optimized XGBoost-based machine learning method for predicting wave run-up on a sloping beach. MethodsX, 10, 102119. https://doi.org/10.1016/j.mex.2023.102119

Tjänster inom kundservice & ärendehantering—Samres. (n.d.). Retrieved May 3, 2023, from https://www.samres.se/en/

Wang, Y., & Guo, Y. (2020). Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost. China Communications, 17(3), 205–221. https://doi.org/10.23919/JCC.2020.03.017

Zhang, B.-T., Ohm, P., & Mühlenbein, H. (1997). Evolutionary Induction of Sparse Neural Trees. Evolutionary Computation, 5(2), 213–236. https://doi.org/10.1162/evco.1997.5.2.213

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159–175. https://doi.org/10.1016/S0925-2312(01)00702-0