

**The University of South Bohemia in České Budějovice**

Faculty of Science

**Comparative Analysis of Machine Learning and Deep  
Learning Models for Time Series Forecasting**

Master Thesis

**Shivangi Shailesh Patel**

Supervisor: Prof. Dr. Michael Scholz

České Budějovice 2026

## **BIBLIOGRAPHIC DETAILS AND DECLARATION**

### **Bibliographic details**

Patel, S. S., 2026: Srovnávací analýza modelů strojového a hlubokého učení pro prognózování časových řad (Comparative Analysis of Machine Learning and Deep Learning Models for Time Series Forecasting). Mgr. Thesis, in English. - 66 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

### **Annotation**

This thesis examines the comparative suitability of statistical, machine learning, deep learning, and simple benchmark approaches for daily time series forecasting. It uses an anonymised canteen guest count data set and investigates how forecasting behaviour is affected by the amount of training data, noisy target values, missing and imputed predictors, a later temporal holdout, and computational requirements. The methods considered include Linear Regression, ARIMAX, Random Forest, XGBoost, LSTM, and GRU, together with seasonal naive and moving average benchmarks. The aim is to provide a transparent basis for selecting forecasting methods for small, structured operational data sets.

### **Materials not available electronically**

The thesis does not include any materials that cannot be published electronically.

### **Student's declaration**

I declare that I am the author of this qualification thesis and that in writing it I have used the sources and literature displayed in the list of used sources only.

Name: Shivangi Shailesh Patel

Matriculation Number: 12400599

Student Number: B23557

Place and date: Willmering, 30.07.2026

Student's signature:



Technische Hochschule Deggendorf

Faculty of Computer Science

Degree Programme: Master Artificial Intelligence and Data Science

# **Vergleichende Analyse von Machine Learning und Deep Learning Modellen für die Zeitreihenprognose**

## **Comparative Analysis of Machine Learning and Deep Learning Models for Time Series Forecasting**

Master Thesis

Submitted in partial fulfilment of the requirements for the degree of

**Master of Science (M.Sc.)**

---

Submitted by

**Shivangi Shailesh Patel**

Matriculation Number: 12400599

---

**Submission Date** 20 July 2026

**First Examiner** Prof. Dr. Michael Scholz, Deggendorf Institute of Technology

**Second Examiner** Prof. Dr. Florian Wahl, Deggendorf Institute of Technology

## Erklärung

Name des Studierenden: Shivangi Shailesh Patel

Name des Betreuenden: Prof. Dr. Michael Scholz

Thema der Abschlussarbeit:

Vergleichende Analyse von Machine Learning und Deep Learning Modellen für  
die Zeitreihenprognose

1. Ich erkläre hiermit, dass ich die Abschlussarbeit gemäß § 35 Abs. 7 RaPO (Rahmenprüfungsordnung für die Fachhochschulen in Bayern, BayRS 2210-4-1-4-1-WFK) selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Deggendorf, 19.07.2026  
Datum



Unterschrift des Studierenden

2. Ich bin damit einverstanden, dass die von mir angefertigte Abschlussarbeit über die Bibliothek der Hochschule einer breiteren Öffentlichkeit zugänglich gemacht wird:

- ☒ Nein  
☐ Ja, nach Abschluss des Prüfungsverfahrens  
☐ Ja, nach Ablauf einer Sperrfrist von ...Jahren.

Deggendorf, 19.07.2026  
Datum



Unterschrift des Studierenden

Bei Einverständnis des Verfassenden vom Betreuenden auszufüllen:

Eine Aufnahme eines Exemplars der Abschlussarbeit in den Bestand der Bibliothek und die Ausleihe des Exemplars wird:

- ☐ Befürwortet  
☐ Nicht befürwortet

Deggendorf, Datum

Unterschrift des Betreuenden

## Abstract

This thesis compares statistical, machine learning, deep learning, and simple benchmark methods for daily time series forecasting under practical data conditions. Linear Regression, ARIMAX, Random Forest, XGBoost, LSTM, and GRU are evaluated on an anonymised guest count series through five phases: baseline forecasting, training data size, noisy and incomplete training data, a later temporal holdout, and computational efficiency. Seasonal naive, 7 day and 28 day moving average, and four week seasonal moving average forecasts are included as rule based benchmarks. Performance is assessed using root mean squared error and mean absolute error. On the primary 80/20 split, Seasonal Naive achieves the lowest overall MAE at 136.13, Random Forest achieves the lowest RMSE among all evaluated methods at 261.54, and XGBoost achieves the lowest MAE among the six fitted models at 157.92. The seasonal moving average improves markedly on the ordinary moving averages but does not outperform Seasonal Naive on MAE. On the later 70/30 holdout, XGBoost achieves the lowest RMSE, while Seasonal Naive again achieves the lowest overall MAE. LSTM and GRU improve with more observations but remain less accurate than the leading fitted alternatives. The findings show that deep learning is not automatically advantageous for a small, structured business series with strong weekly recurrence and lag predictors. The contribution is a transparent multi condition comparison and practical model selection guidance. Limitations concerning target leakage, single seed evaluation, temporal holdout interpretation, and model size measurement are explicitly documented.

**Keywords:** time series forecasting; machine learning; deep learning; Random Forest; XGBoost; LSTM; GRU; robustness; model selection

# Contents

|                                                                   |            |
|-------------------------------------------------------------------|------------|
| <b>Abstract</b>                                                   | <b>i</b>   |
| <b>List of Abbreviations</b>                                      | <b>vii</b> |
| <b>1 Introduction</b>                                             | <b>1</b>   |
| 1.1 Background and Motivation . . . . .                           | 1          |
| 1.2 Problem Statement . . . . .                                   | 2          |
| 1.3 Research Gap . . . . .                                        | 3          |
| 1.4 Research Questions . . . . .                                  | 3          |
| 1.5 Research Objectives . . . . .                                 | 4          |
| 1.6 Scope and Delimitations . . . . .                             | 4          |
| 1.7 Contributions . . . . .                                       | 5          |
| 1.8 Structure of the Thesis . . . . .                             | 5          |
| <b>2 Foundations of Time Series Forecasting</b>                   | <b>6</b>   |
| 2.1 Time Series and Forecasting Tasks . . . . .                   | 6          |
| 2.2 Temporal Components . . . . .                                 | 6          |
| 2.3 Transforming a Time Series into Supervised Learning . . . . . | 7          |
| 2.4 Chronological Training and Testing . . . . .                  | 8          |
| 2.5 Statistical Reference Models . . . . .                        | 8          |
| 2.5.1 Linear Regression . . . . .                                 | 8          |
| 2.5.2 ARIMAX . . . . .                                            | 8          |
| 2.6 Tree Based Machine Learning . . . . .                         | 9          |
| 2.6.1 Random Forest . . . . .                                     | 9          |
| 2.6.2 XGBoost . . . . .                                           | 9          |
| 2.7 Recurrent Deep Learning . . . . .                             | 10         |
| 2.7.1 Recurrent Neural Networks . . . . .                         | 10         |
| 2.7.2 LSTM . . . . .                                              | 10         |
| 2.7.3 GRU . . . . .                                               | 10         |
| 2.8 Scaling and Optimisation . . . . .                            | 11         |
| 2.9 Error Measures . . . . .                                      | 11         |
| 2.10 Real World Data Challenges . . . . .                         | 12         |
| 2.10.1 Limited Data . . . . .                                     | 12         |
| 2.10.2 Noise . . . . .                                            | 12         |
| 2.10.3 Missing Values . . . . .                                   | 13         |
| 2.10.4 Distribution Shift and Concept Drift . . . . .             | 13         |
| 2.11 Reproducibility and Random Seeds . . . . .                   | 13         |
| 2.12 Efficiency and Practical Model Quality . . . . .             | 14         |
| <b>3 Related Work</b>                                             | <b>15</b>  |
| 3.1 Overview of the Literature . . . . .                          | 15         |
| 3.2 Classical Models versus Deep Learning . . . . .               | 15         |
| 3.3 Deep Learning Architectures for Forecasting . . . . .         | 16         |
| 3.4 Data Size and Scaling Behaviour . . . . .                     | 16         |
| 3.5 Noise Robustness . . . . .                                    | 17         |
| 3.6 Missing Data and Imputation . . . . .                         | 18         |
| 3.7 Distribution Shift and Concept Drift . . . . .                | 18         |

|          |                                                                       |           |
|----------|-----------------------------------------------------------------------|-----------|
| 3.8      | Efficiency, Reproducibility, and Baseline Quality . . . . .           | 19        |
| 3.9      | Synthesis and Derived Research Need . . . . .                         | 19        |
| <b>4</b> | <b>Data Understanding and Preprocessing</b>                           | <b>21</b> |
| 4.1      | Data Source and Forecasting Target . . . . .                          | 21        |
| 4.2      | Original Variables . . . . .                                          | 21        |
| 4.3      | Preprocessing Pipeline . . . . .                                      | 22        |
| 4.4      | Selected Predictors . . . . .                                         | 23        |
| 4.5      | Descriptive Statistics . . . . .                                      | 23        |
| 4.6      | Weekly Structure . . . . .                                            | 25        |
| 4.7      | Primary Train Test Split . . . . .                                    | 26        |
| 4.8      | Training Data Subsets . . . . .                                       | 27        |
| 4.9      | Noise Scenario . . . . .                                              | 27        |
| 4.10     | Missing Value Scenario . . . . .                                      | 28        |
| 4.11     | Scaling and Sequence Construction . . . . .                           | 28        |
| 4.12     | Data Quality and Validity Considerations . . . . .                    | 29        |
| <b>5</b> | <b>Methodology</b>                                                    | <b>30</b> |
| 5.1      | Research Design . . . . .                                             | 30        |
| 5.2      | Why These Models Were Selected . . . . .                              | 31        |
| 5.3      | Common Predictor Specification . . . . .                              | 32        |
| 5.4      | Model Configurations . . . . .                                        | 32        |
| 5.5      | Phase 1: Baseline Performance . . . . .                               | 33        |
| 5.6      | Phase 2: Training Data Size . . . . .                                 | 34        |
| 5.7      | Phase 3: Noise and Missing Values . . . . .                           | 34        |
| 5.8      | Phase 4: Later Temporal Holdout . . . . .                             | 35        |
| 5.9      | Phase 5: Efficiency . . . . .                                         | 35        |
| 5.10     | Evaluation Metrics . . . . .                                          | 36        |
| 5.11     | Random Seed and Reproducibility . . . . .                             | 36        |
| 5.12     | Implementation Environment . . . . .                                  | 36        |
| 5.13     | Threats to Experimental Validity . . . . .                            | 37        |
| <b>6</b> | <b>Results</b>                                                        | <b>38</b> |
| 6.1      | Presentation of Results . . . . .                                     | 38        |
| 6.2      | Phase 1: Baseline Performance . . . . .                               | 38        |
| 6.2.1    | Simple Weekly and Moving Average Benchmarks . . . . .                 | 39        |
| 6.3      | Phase 2: Effect of Training Data Size . . . . .                       | 39        |
| 6.4      | Phase 3: Noise and Missing Training Data . . . . .                    | 41        |
| 6.5      | Phase 4: Performance on the Later 30% Holdout . . . . .               | 42        |
| 6.6      | Phase 5: Repeated Training Time and Model Size . . . . .              | 43        |
| 6.7      | Accuracy Efficiency Relationship . . . . .                            | 44        |
| 6.8      | Open Day Robustness Check . . . . .                                   | 45        |
| 6.9      | Consolidated Phase Wise Ranking . . . . .                             | 46        |
| <b>7</b> | <b>Discussion</b>                                                     | <b>48</b> |
| 7.1      | Interpretation Strategy . . . . .                                     | 48        |
| 7.2      | Baseline Comparison . . . . .                                         | 48        |
| 7.3      | Research Question 1: Effect of Data Size . . . . .                    | 49        |
| 7.4      | Research Question 2: Robustness to Noise and Missing Values . . . . . | 50        |
| 7.5      | Research Question 3: Generalisation to a Later Period . . . . .       | 51        |
| 7.6      | Research Question 4: Computational Requirements . . . . .             | 51        |
| 7.7      | Why the Deep Models Underperformed . . . . .                          | 52        |

|          |                                                       |           |
|----------|-------------------------------------------------------|-----------|
| 7.8      | Critical Assessment of Feature Leakage . . . . .      | 53        |
| 7.9      | Seed Use and Stochastic Reliability . . . . .         | 54        |
| 7.10     | Practical Model Selection Guidelines . . . . .        | 54        |
| 7.11     | Implications for Stakeholders . . . . .               | 55        |
| 7.12     | Research Contribution Revisited . . . . .             | 56        |
| 7.13     | Limitations . . . . .                                 | 56        |
| <b>8</b> | <b>Conclusion and Future Work</b>                     | <b>58</b> |
| 8.1      | Summary of the Study . . . . .                        | 58        |
| 8.2      | Answers to the Research Questions . . . . .           | 59        |
| 8.2.1    | Effect of Training Data Size . . . . .                | 59        |
| 8.2.2    | Robustness to Noise and Missing Values . . . . .      | 59        |
| 8.2.3    | Generalisation to a Later Period . . . . .            | 59        |
| 8.2.4    | Computational Requirements . . . . .                  | 60        |
| 8.2.5    | Practical Model Selection Guidance . . . . .          | 60        |
| 8.3      | Scientific and Practical Contribution . . . . .       | 61        |
| 8.4      | Required Corrections Before Operational Use . . . . . | 61        |
| 8.5      | Future Work . . . . .                                 | 62        |
| 8.6      | Final Conclusion . . . . .                            | 62        |
|          | <b>Bibliography</b>                                   | <b>64</b> |

**List of Figures**

4.1 Daily guest count and the primary chronological 80/20 split . . . . . 24

4.2 Distribution of the processed daily guest target . . . . . 25

4.3 Mean daily guest count by weekday . . . . . 26

5.1 Overall experimental workflow . . . . . 30

6.1 Baseline RMSE versus mean training time across repeated runs . . . . . 45

## List of Tables

|     |                                                                                |    |
|-----|--------------------------------------------------------------------------------|----|
| 4.1 | Variables available in the original daily dataset . . . . .                    | 22 |
| 4.2 | Predictors used in the forecasting models . . . . .                            | 23 |
| 4.3 | Summary of the processed target and chronological split . . . . .              | 24 |
| 4.4 | Descriptive statistics of the model predictors after preprocessing . . . . .   | 24 |
| 5.1 | Model configurations used in the experiments . . . . .                         | 33 |
| 6.1 | Phase one baseline forecasting performance . . . . .                           | 38 |
| 6.2 | Simple benchmark performance on the primary 80/20 test split . . . . .         | 39 |
| 6.3 | Phase two performance across training data sizes . . . . .                     | 39 |
| 6.4 | Phase three performance under training data corruption . . . . .               | 41 |
| 6.5 | Phase four results for the 70/30 chronological split . . . . .                 | 43 |
| 6.6 | Repeated training time and framework appropriate fitted model size . . . . .   | 44 |
| 6.7 | Forecasting accuracy on all test days and open canteen days . . . . .          | 45 |
| 6.8 | Best observed forecasting method by accuracy condition and evaluation metric . | 46 |
| 7.1 | Practical model selection guidance derived from the experiment . . . . .       | 55 |

## List of Abbreviations

| Abbreviation | Meaning                                                           |
|--------------|-------------------------------------------------------------------|
| AI           | Artificial Intelligence                                           |
| ARIMA        | Autoregressive Integrated Moving Average                          |
| ARIMAX       | Autoregressive Integrated Moving Average with exogenous variables |
| DL           | Deep Learning                                                     |
| GBRT         | Gradient Boosting Regression Tree                                 |
| GRU          | Gated Recurrent Unit                                              |
| LSTM         | Long Short Term Memory                                            |
| MAE          | Mean Absolute Error                                               |
| ML           | Machine Learning                                                  |
| MSE          | Mean Squared Error                                                |
| RAM          | Random Access Memory                                              |
| RF           | Random Forest                                                     |
| RMSE         | Root Mean Squared Error                                           |
| RNN          | Recurrent Neural Network                                          |
| XGBoost      | Extreme Gradient Boosting                                         |

# 1 Introduction

## 1.1 Background and Motivation

Forecasting uses historical data to estimate future outcomes. In operational settings, even a modest enhancement in forecasting accuracy can facilitate optimal decision making in domains such as workforce planning, purchasing, production scheduling, inventory management, energy utilization, and service delivery. Daily guest count forecasting is particularly useful in organisational contexts where staffing, food preparation, purchasing, and capacity are planned from day to day. For instance, estimating the expected number of customers or visitors can facilitate the determination of staffing requirements, food preparation quantities, operating capacity, and material needs. Nevertheless, the usefulness of a forecasting model should not be evaluated solely based on minor discrepancies in numerical error. A reliable model should also maintain its effectiveness in scenarios where historical data is limited, measurements are subject to errors, some observations are missing, or guest count patterns undergo changes over time.

Time series forecasting has historically been approached through the utilization of statistical methodologies that encompass the modeling of trends, seasonality, autocorrelation, and external influences. Autoregressive integrated moving average models are among the most prominent representatives of this family [1, 2]. In recent years, machine learning methods have become increasingly prevalent due to their capacity to discern nonlinear relationships from engineered predictors. Tree ensembles, including Random Forest and Gradient Boosting, are particularly well suited for the analysis of structured business data. These ensembles have the capacity to represent interactions without the necessity of a manually specified functional form, as shown by Breiman and by Chen and Guestrin [3, 4]. Deep learning models extend this idea by learning distributed representations and temporal dependencies directly from sequences. LSTM and GRU networks are frequently used because their gating mechanisms are intended to preserve useful information across time [5, 6].

The growing popularity of deep learning has created an implicit expectation that a more complex model should also provide a better forecast. That expectation is not generally valid. Forecasting competitions and comparative studies repeatedly show that model quality depends on the data, the evaluation design, and the strength of the baseline [7, 8]. A deep neural network has more representational capacity than a linear model, but this capacity is useful only when the training data contain enough information to estimate the additional parameters. When the data set is small, strongly seasonal, or dominated by a few engineered predictors, a tree ensemble or classical time series model may provide better generalisation with much lower computational cost. Conversely, recurrent networks may be advantageous when long dependencies, nonlinear temporal interactions, and a large number of related sequences are available [9, 10].

A second problem is that model comparisons are often performed under ideal conditions. The data are cleaned in advance, the training set has a fixed size, and the evaluation is based on a single test split. Such an experiment answers a narrow question: which model performed best on one prepared data set under one configuration? It does not answer whether the same ranking remains valid when less history is available, when the training values contain noise, when predictors are incomplete, or when later observations follow a different distribution. These conditions are normal rather than exceptional in practice. Therefore, an accuracy only comparison on clean data is insufficient for operational model selection.

This thesis addresses that limitation through a five phase experimental comparison. The analysis is based on an anonymised daily guest count data set containing the target count, weather information, holiday indicators, and calendar attributes. Linear Regression and ARIMAX provide statistical reference points. Random Forest and XGBoost represent two different ensemble learning principles: bagging and boosting. LSTM and GRU represent gated recurrent neural networks. Seasonal Naive, ordinary moving averages, and a seasonal moving average provide rule based references that test whether model complexity adds value beyond weekly recurrence and simple smoothing. The methods are examined under a clean baseline, three training set sizes, noise and missing value scenarios, a later temporal test window, and an efficiency measurement. Root mean squared error and mean absolute error are used because they reveal different aspects of forecast error.

The purpose is not to prove that one model class is universally superior. The more defensible objective is to identify which models are suitable under the observed conditions and to explain why the ranking changes. This distinction is important. A single winner can be reported for a particular test set, but a practical recommendation requires a broader understanding of data volume, robustness, computational effort, and experimental uncertainty.

### 1.2 Problem Statement

A wide range of statistical, machine learning, and deep learning models is available for time series forecasting. This variety creates a model selection problem. Organisations may choose a complex method because it is fashionable, choose a simple method because it is convenient, or rely on a result obtained from one clean data split. None of these approaches guarantees that the selected model will remain effective in realistic conditions.

The central problem examined in this thesis is the absence of a clear, evidence based basis for selecting between representative machine learning and deep learning models for a relatively small real world daily forecasting task. The available data may be limited, noisy, incomplete, and temporally non stationary. A model demonstrating accuracy on the original data may become unstable when these conditions change. A model with a marginally reduced error may also be impractical if it necessitates a significantly greater amount of training time or computational resources.

The problem is therefore multi dimensional. It includes predictive accuracy, sensitivity to data quantity, sensitivity to data quality, performance on later observations, and efficiency. A meaningful comparison must keep the forecasting target and evaluation metrics consistent while systematically changing these conditions.

### 1.3 Research Gap

Preliminary studies have yielded significant insights regarding distinct components of forecasting. A body of research has demonstrated that meticulously designed gradient boosted trees can exhibit superior performance in several forecasting benchmarks when compared to more intricate neural architectures [8]. A review of the literature on recurrent and other deep learning architectures is provided, with an emphasis on the strengths and limitations of deep forecasting models. A variety of research streams have been dedicated to the examination of noisy labels [12], missing value imputation [13, 14], generalisation [15], and concept drift [16].

These studies do not negate the necessity for an application specific comparison. A critical examination of the available data sets reveals significant variations in forecasting horizons, feature structures, and model configurations. Of particular concern is the tendency to examine these factors in isolation. One paper may investigate architecture choice, another missing values, and another scaling behavior. A practitioner who must select a model for a particular business time series still requires a unified experiment that employs the same models, predictors, test logic, and metrics across multiple operating conditions.

The issue addressed in this thesis is not the development of a novel forecasting algorithm. The absence of a transparent, multi condition evaluation for the selected real world data set is a salient issue. The study integrates five factors within a unified framework, subsequently translating the findings into a set of practical model selection guidelines. The efficacy of this comparative approach is contingent upon the explicit articulation of experimental controls and limitations. Comparative research in other supervised learning settings has demonstrated that algorithm performance is contingent upon data complexity and sample size [17].

### 1.4 Research Questions

The study is guided by the following research questions:

- RQ1:** How does the amount of available training data affect the forecasting accuracy of the selected statistical, machine learning, and deep learning models?
- RQ2:** How robust are the selected models when the historical training target is noisy or selected training predictors are missing and imputed?
- RQ3:** How well do the selected models generalise from an earlier training period to a later holdout period?
- RQ4:** What differences in training time and model memory requirements are observed among the

selected forecasting models?

**RQ5:** What practical model selection guidance follows from the empirical results for a small, structured daily guest count forecasting task?

These questions are limited to the data, predictors, forecast horizon, and implementations examined in this thesis. They do not imply a universal ranking of forecasting algorithms.

## 1.5 Research Objectives

The overall objective is to establish an evidence based basis for selecting a forecasting model for a small, structured daily canteen guest count series. The study therefore aims to:

- O1:** determine which model families provide reliable point forecasts for the available guest count data;
- O2:** identify how strongly the model ranking depends on data quantity, data quality, and the position of the test period;
- O3:** assess whether the model ranking remains stable when evaluation is restricted to days on which the canteen is open;
- O4:** document computational requirements through repeated timing measurements and framework appropriate model size measures; and
- O5:** translate the results into practical guidance for data scientists and canteen decision makers.

These objectives describe the intended knowledge and practical outcomes. The five experimental phases in Chapter 5 are the methodological steps used to achieve them.

## 1.6 Scope and Delimitations

The study focuses on point forecasting of one daily target series. The test observations are predicted using external predictors and lagged target values. Probabilistic intervals are not estimated. The model set is intentionally limited to six methods. This keeps the experiment interpretable and allows the selected models to represent distinct modelling assumptions without turning the thesis into an exhaustive benchmark.

The noise and missing value experiments modify the training data only. They therefore measure the effect of imperfect historical learning data, not the effect of corrupt predictors arriving at prediction time. The distribution shift phase uses a later chronological holdout but does not implement a formal drift detection algorithm. The efficiency analysis uses one warm up run followed by at least ten measured training runs per model. Mean training time and standard deviation are reported. Ordinary R models are measured with their fitted object size, while XGBoost and TensorFlow/Keras models are measured through framework specific serialised model files. These boundaries are important when interpreting the conclusions.

## 1.7 Contributions

The thesis makes four contributions.

First, it provides a unified experimental design that evaluates accuracy, data size sensitivity, robustness, temporal generalisation, and efficiency with the same daily forecasting problem. Second, it compares three fitted model families rather than presenting only a machine learning versus deep learning contrast. The inclusion of Linear Regression and ARIMAX shows whether either modern model family improves on transparent statistical references, while Seasonal Naive and moving average benchmarks show whether fitted models improve on simple weekly recurrence and smoothing rules. Third, it derives practical recommendations from the phase wise behaviour instead of relying on one overall accuracy table. Fourth, it documents methodological limitations that materially affect the strength of the conclusions, including potential feature leakage, the interpretation of the temporal split, stochastic variation, and the distinction between serialised model size and peak RAM usage.

The contribution is empirical and decision oriented. No new learning algorithm is claimed. The value lies in showing how model choice changes when the operating condition changes and in identifying where the current evidence is strong or weak.

## 1.8 Structure of the Thesis

Chapter 2 introduces the forecasting concepts, model families, error measures, and reproducibility principles needed for the analysis. Chapter 3 reviews comparative forecasting research, data size effects, noise, missing data, distribution shift, and efficiency. Chapter 4 describes the guest count data set, preprocessing, feature engineering, and chronological splits. Chapter 5 explains the six models, the five experimental phases, the random seed, and the implementation choices. Chapter 6 presents the numerical results without extending them beyond the observed experiments. Chapter 7 interprets the findings, answers the research questions, tests alternative explanations, and develops model selection guidelines. Chapter 8 summarises the main conclusions, limitations, and future work. The full source code, machine readable result tables, generated figures, and supplementary notes are provided with the electronic thesis project. Extensive program listings are not included in the printed thesis.

## 2 Foundations of Time Series Forecasting

### 2.1 Time Series and Forecasting Tasks

A time series is defined as an ordered sequence of observations indexed by time. The distinction between time series learning and an ordinary tabular regression problem is rooted in the ordering of the elements. In an independently sampled regression data set, the order of rows can often be altered without altering the interpretation of the data. In a time series, the process of shuffling has the potential to disrupt the established temporal structure. This disruption can allow future information to exert influence on previously recorded data, thereby altering the interpretation of past events. Trend, seasonality, calendar effects, delayed responses, and serial correlation depend on the chronological order of the observations [1, 18]. The number of guests on day  $t$  is denoted by  $y_t$ , and the vector of predictors available for that day is denoted by  $x_t$ . A one step point forecast can be expressed as follows:

$$\hat{y}_t = f(y_{t1}, y_{t2}, \dots, x_t; \theta), \quad (2.1)$$

where  $f(\cdot)$  is a forecasting model and  $\theta$  represents its estimated parameters. Lagged target values can represent short term persistence and recurring seasonal behaviour. External predictors, such as weather and holiday information, can provide additional contextual information. The specific predictors used in this study are described in Chapter 5.

The experiments produce point forecasts rather than probability distributions. A point forecast estimates one expected value for each day. A probabilistic forecast would additionally describe uncertainty through quantiles, prediction intervals, or a full conditional distribution. Methods such as DeepAR were developed specifically for probabilistic forecasting across many related series [9]. The present data contain one principal target series, and the selected evaluation metrics are point error measures; probabilistic modelling is therefore outside the main scope.

### 2.2 Temporal Components

Several recurring components are commonly used to describe a time series. A trend represents gradual long term movement. Seasonality is a systematic pattern that repeats at a known calendar frequency, such as a weekly or annual cycle. Cyclic behaviour also repeats but not necessarily with a fixed period. Irregular variation contains short lived or unpredictable changes. Real business series frequently combine all four components.

Daily time series often contain weekly seasonal patterns because operational activity differs across weekdays. Lagged observations can represent this recurring structure, while weather and

calendar variables may provide additional contextual information. Weak marginal correlations do not necessarily imply that a predictor is irrelevant, because its effect may be nonlinear or conditional on other variables. Tree based methods can represent such interactions more naturally than a purely additive linear specification. The empirical temporal patterns observed in this study are presented in Chapter 4.

A further component is regime behaviour. The daily guest count can be close to zero on closed days and substantially higher on open days. This creates a mixture like target distribution rather than one smooth Gaussian process. The distinction matters because a single model must learn both whether the daily guest count is near zero and, when the operation is active, how high the guest attendance will be. A variable that directly indicates opening status can therefore be highly predictive. However, it must be known independently at forecast time. If the indicator is derived from the target itself, it creates target leakage. This issue is revisited in Chapters 4 and 7.

### 2.3 Transforming a Time Series into Supervised Learning

Tree ensembles and linear regression operate on rows of predictors. A time series can be transformed into this form by creating lagged variables and calendar features. For day  $t$ , a supervised feature vector can be expressed generally as

$$\mathbf{x}_t = [\mathbf{z}_t, y_{t1}, y_{t2}, \dots, y_{tp}]^\top, \quad (2.2)$$

where  $\mathbf{z}_t$  contains external or calendar related predictors and the remaining terms represent lagged target values. The exact feature specification used in the empirical analysis is presented in Chapter 5.

For recurrent networks, rows are further grouped into sequences. With a sequence length  $L$ , the input for one prediction can be represented as a tensor

$$\mathbf{X}_t = [\mathbf{x}_{tL}, \mathbf{x}_{tL+1}, \dots, \mathbf{x}_{t1}], \quad (2.3)$$

and the corresponding target is  $y_t$ . This representation allows the network to process a sequence of multivariate historical observations. The sequence length and number of predictors used in this study are specified in Chapter 5.

The supervised transformation has an important consequence: the model does not automatically know all calendar structure. It learns only from the supplied variables and the selected sequence length. A sequence that is too short may omit relevant temporal dependencies, while an unnecessarily long sequence may increase model complexity without adding useful information. The sequence length is therefore a modelling assumption rather than a neutral preprocessing choice. The value used in this study is reported in Chapter 5.

## 2.4 Chronological Training and Testing

Forecasting evaluation must respect time. A random split can place future observations in the training set and earlier observations in the test set. Even when the target itself is not directly leaked, the model is then trained on patterns that would not have been available at the historical prediction date. Chronological splitting is the simplest way to avoid this problem.

A chronological split assigns earlier observations to model training and later observations to model evaluation. This preserves temporal order and ensures that the model is tested on observations occurring after the training period. The precise split proportions and date ranges used in this study are presented in Chapters 4 and 5.

A single holdout remains less informative than rolling origin evaluation. In rolling origin evaluation, the training window advances repeatedly and forecasts are generated for several origins. This reduces dependence on one boundary and reveals whether performance changes across time. Nevertheless, a fixed chronological holdout is transparent and computationally manageable for a thesis that repeats six models across several scenarios. The limitation is explicitly retained rather than hidden.

## 2.5 Statistical Reference Models

### 2.5.1 Linear Regression

Multiple Linear Regression models the conditional expectation of the target as an additive function of predictors:

$$y_t = \beta_0 + \sum_{j=1}^p \beta_j x_{t,j} + \varepsilon_t, \quad (2.4)$$

where  $\varepsilon_t$  is an error term and  $p$  is the number of predictors. The model is easy to estimate, fast to train, and straightforward to interpret. Its main limitation is structural: nonlinear relationships and interactions must be specified explicitly. Linear Regression is included because a complex model should be compared with a simple, transparent baseline. Without this reference, an apparently strong result may only demonstrate that the target is predictable, not that the advanced model is necessary.

### 2.5.2 ARIMAX

An ARIMA model represents a time series through autoregressive terms, differencing, and moving average errors. In compact notation, an ARIMA( $p, d, q$ ) process can be expressed as

$$\phi(B)(1B)^d y_t = c + \theta(B)\varepsilon_t, \quad (2.5)$$

where  $B$  is the backshift operator,  $\phi(B)$  is the autoregressive polynomial,  $\theta(B)$  is the moving

average polynomial, and  $d$  is the differencing order. ARIMAX extends this structure with exogenous regressors:

$$\phi(B)(1B)^d y_t = c + \boldsymbol{\gamma}^\top \mathbf{x}_t + \theta(B)\varepsilon_t. \quad (2.6)$$

ARIMAX can be estimated using manually specified or automatically selected autoregressive, differencing, and moving average orders. The concrete order selection procedure, predictor specification, and forecasting strategy used in this study are described in Chapter 5.

## 2.6 Tree Based Machine Learning

### 2.6.1 Random Forest

A regression tree partitions the predictor space into regions and assigns a constant prediction to each terminal region. A single tree can model nonlinear effects but is unstable: small changes in the data may change the chosen splits. Random Forest reduces this instability by training many trees on bootstrap samples and by considering a random subset of predictors at each split [3]. The regression prediction is the average

$$\hat{y}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B T_b(\mathbf{x}), \quad (2.7)$$

where  $T_b$  is the prediction of tree  $b$  and  $B$  is the number of trees.

The main strength of Random Forest is variance reduction. It can represent nonlinear relationships, thresholds, and interactions without scaling the predictors. It is often competitive on modest tabular data sets. Its limitations include reduced extrapolation beyond the range of observed targets, larger model objects than a linear model, and randomness from bootstrap sampling and feature selection. The number of trees used in the empirical implementation is specified in Chapter 5.

### 2.6.2 XGBoost

Gradient boosting builds an additive ensemble sequentially. Each new tree is trained to correct residual errors made by the current ensemble. XGBoost provides a computationally efficient and regularised implementation of gradient boosted trees [4]. The objective can be written as

$$\mathcal{L}^{(m)} = \sum_{i=1}^n \ell \left( y_i, \hat{y}_i^{(m1)} + f_m(\mathbf{x}_i) \right) + \Omega(f_m), \quad (2.8)$$

where  $f_m$  is the new tree,  $\ell$  is the loss function, and  $\Omega$  penalises tree complexity. The specific loss function, number of boosting rounds, and remaining implementation settings used in this study are described in Chapter 5.

Compared with Random Forest, boosting can reduce bias by focusing successive trees on difficult observations. It can also overfit when the number and depth of trees are not controlled. The performance of gradient boosting depends on choices such as learning rate, tree depth, regularisation, subsampling, and the number of boosting rounds. The configuration used in this study is documented in Chapter 5.

## 2.7 Recurrent Deep Learning

### 2.7.1 Recurrent Neural Networks

A recurrent neural network updates a hidden state as a sequence is processed:

$$\mathbf{h}_t = g(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b}). \quad (2.9)$$

The hidden state acts as a learned summary of previous inputs. Standard recurrent networks can struggle to learn long dependencies because gradients may vanish or explode during back-propagation through time. Gated architectures were introduced to regulate how information is stored, updated, and forgotten.

### 2.7.2 LSTM

The LSTM architecture uses an internal cell state and several gates [5]. A common formulation is

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i), \quad (2.10)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f), \quad (2.11)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o), \quad (2.12)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c), \quad (2.13)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \quad (2.14)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \quad (2.15)$$

The input, forget, and output gates control information flow. The concrete LSTM architecture and training settings are documented in Chapter 5.

### 2.7.3 GRU

The GRU simplifies gated recurrence by combining parts of the LSTM mechanism [6]. A typical formulation uses reset and update gates:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t1}), \quad (2.16)$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t1}), \quad (2.17)$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h (\mathbf{r}_t \odot \mathbf{h}_{t1})), \quad (2.18)$$

$$\mathbf{h}_t = (\mathbf{1z}_t) \odot \mathbf{h}_{t1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t. \quad (2.19)$$

A GRU usually has fewer parameters than an LSTM with the same number of units. This can reduce training cost and may help on smaller data sets, although empirical performance depends on the task. The concrete architecture and training configuration used in this study are presented in Chapter 5.

## 2.8 Scaling and Optimisation

Neural networks are sensitive to predictor scale. If one variable ranges from zero to one and another from zero to several thousand, gradient based optimisation can become inefficient. A common preprocessing method for neural networks is min max scaling:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}. \quad (2.20)$$

To avoid information leakage, scaling parameters should be estimated only from the training data and then applied unchanged to later observations. When the target is scaled during model training, predictions must be transformed back to the original unit before evaluation. Tree based models generally do not require scaling because their split decisions depend on the ordering of values rather than on gradient based optimisation. The concrete scaling procedure used in this study is described in Chapter 5.

Recurrent forecasting models are commonly trained with differentiable regression losses and gradient based optimisers such as Adam [19]. Training loss and evaluation metrics are related but not identical. Mean squared error is the square of RMSE before the square root is applied. Optimising squared error gives large residuals greater influence, which is consistent with RMSE based evaluation.

## 2.9 Error Measures

Let  $e_t = y_t - \hat{y}_t$  denote the forecast error for observation  $t$  and let  $n$  be the number of test observations. RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{t=1}^n e_t^2}. \quad (2.21)$$

RMSE penalises large errors strongly because residuals are squared. It is expressed in the original target unit after taking the square root. A model can therefore have a good average absolute error but a worse RMSE if a small number of days are predicted very poorly.

MAE is defined as

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^n |e_t|. \quad (2.22)$$

MAE weights deviations linearly and is easier to interpret as the average absolute number of guests by which a forecast is wrong. Neither metric is universally sufficient. Reporting both helps distinguish typical error from sensitivity to large misses. Percentage errors are not used because the target contains zero count days, for which percentage calculations are undefined or unstable.

## 2.10 Real World Data Challenges

### 2.10.1 Limited Data

Deep learning is often associated with scaling behaviour: performance can improve predictably as data and model capacity increase [20, 21]. This does not imply that every deep model improves enough to become preferable on every data set. A recurrent network with thousands of parameters trained on only several hundred sequences may have high estimation variance. Tree ensembles can be more data efficient because their inductive bias fits structured predictors and threshold effects.

Data size also interacts with recency. Using more history can reduce variance, but old observations may represent a different operating regime. Consequently, a model trained on a shorter but more recent historical window may occasionally outperform the same method trained on a longer history when the recent period resembles the test period more closely. Experiments based on trailing windows therefore change both sample size and temporal coverage. The concrete training fractions used in this study are described in Chapter 5.

### 2.10.2 Noise

Noise can enter a forecasting pipeline through measurement error, manual data entry, delayed reporting, or changes in the definition of the target. Gaussian noise can be represented as

$$y_t^{\text{noise}} = y_t + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \sigma_\eta^2), \quad (2.23)$$

where  $\sigma_\eta$  controls the magnitude of the perturbation. This represents label noise rather than predictor noise. The concrete noise level and experimental procedure used in this study are specified in Chapter 5. Research shows that neural networks can sometimes fit despite substantial

label corruption, particularly with large data sets [12]. On a small regression series, however, robustness cannot be assumed. Each model’s inductive bias determines whether noise is averaged out, fitted, or propagated into future predictions.

### 2.10.3 Missing Values

Missingness can be handled by deleting observations, simple imputation, model based imputation, or architectures that learn missingness patterns. Advanced methods such as BRITS and graph recurrent imputation explicitly model incomplete multivariate sequences [13, 14]. Simple mean imputation is a basic reference strategy in which missing values are replaced by the mean of the corresponding variable.

Mean imputation is computationally inexpensive and preserves the number of observations. However, it reduces variability and may weaken temporal structure because the same central value replaces different missing contexts. It should therefore be treated as a simple benchmark rather than an optimal missing data method. The missingness rate, affected predictors, and imputation procedure used in this study are specified in Chapter 5.

### 2.10.4 Distribution Shift and Concept Drift

A forecasting model assumes that patterns learned from the past remain sufficiently informative for the future. Distribution shift occurs when the statistical properties of predictors, targets, or their relationship change. Concept drift is often defined more specifically as a change in the mapping from predictors to target [16]. Drift may be abrupt, gradual, recurring, or incremental.

A later chronological holdout can provide a practical stress test because a model is trained on an earlier period and evaluated on later observations. Such a comparison can reveal whether performance remains stable when the target level, predictor distribution, or calendar composition changes.

A single chronological split does not identify the cause of a performance change. Differences may result from a smaller training sample, a different seasonal composition, a longer test interval, or an actual change in the relationship between predictors and target. The exact temporal holdout used in this study is therefore presented in Chapter 5 and is interpreted as a test of later period generalisation rather than definitive evidence of concept drift adaptation.

## 2.11 Reproducibility and Random Seeds

Machine learning experiments contain several random processes. Random Forest draws bootstrap samples and random feature subsets. Neural networks randomly initialise weights and may use stochastic mini batch operations. The noise and missing value scenarios also use random sampling. If these random numbers are uncontrolled, two executions can produce different results.

A pseudo random number generator produces a deterministic sequence from an initial seed.

Documenting and controlling the seed supports reproducibility by providing the same starting state for stochastic procedures. The concrete seed values and framework settings used in the experiments are specified in Chapter 5.

Three qualifications are necessary. First, setting a seed once does not reset the generator before every model, because earlier random operations advance its state. Second, some neural network operations can remain nondeterministic depending on hardware and backend configuration. Third, one perturbation realisation does not measure stochastic uncertainty. Repeated configurations and summary statistics are therefore preferable when stochastic variation is relevant.

### **2.12 Efficiency and Practical Model Quality**

Forecast accuracy is only one component of model quality. Training time affects experimentation, retraining frequency, and deployment cost. Model size influences storage and memory requirements. Interpretability, maintenance effort, and sensitivity to hyperparameters also matter.

Wall clock training time can be affected by background processes, software initialisation, caching, and hardware. Repeated measurements are therefore more informative than a single observation. Model size also requires a framework appropriate definition, because an interface object may not include the underlying model. The concrete timing and object size procedures are specified in Chapter 5.

A practical conclusion should consequently consider at least four questions: Is the error low enough for the decision? Is the result stable under realistic data changes? Is the computational cost acceptable? Is the measurement itself valid? The later discussion applies these questions to the observed model rankings.

## 3 Related Work

### 3.1 Overview of the Literature

Research on time series forecasting has expanded from statistical models to machine learning, recurrent neural networks, and convolutional architectures. This development has increased the number of available methods but has not produced one model that is consistently best. The literature instead points to interactions among architecture, sample size, horizon, feature engineering, data quality, and evaluation design.

The papers reviewed for this thesis can be grouped into six themes: general comparisons between classical and deep models, deep learning architecture reviews, data size and scaling behaviour, robustness to noise, missing data handling, and distribution shift. A seventh theme concerns experimental reproducibility and the strength of baselines. These streams motivate the five phase evaluation used in the empirical part.

### 3.2 Classical Models versus Deep Learning

Elsayed et al. directly question whether deep learning models are necessary for time series forecasting [8]. Their study transforms forecasting into a window based regression problem and compares eight deep models with a feature engineered gradient boosting baseline across nine data sets. The gradient boosted model outperforms the evaluated deep architectures in many cases. The importance of this result lies less in declaring boosting universally superior and more in demonstrating that a strong classical baseline can invalidate conclusions drawn from weak comparisons. Feature engineering, input windows, and output construction can compensate for the absence of an end to end neural representation.

The present thesis follows the same caution but uses a narrower application. XGBoost and Random Forest receive lag and contextual predictors, while LSTM and GRU receive seven day sequences of the same feature family. The comparison is not perfectly architecture neutral because the representations differ, but it avoids comparing a fully engineered machine learning model against an uninformative neural input.

The M4 forecasting competition provides broader evidence that complexity alone is not a guarantee of accuracy [7]. Hybrid and combination methods performed strongly, and several sophisticated learning approaches did not dominate established statistical techniques. Forecasting competitions differ from the current work because they include many series and horizons, but they reinforce the need for multiple baselines and transparent evaluation.

Comparative work in adjacent supervised learning domains reaches a similar conclusion. Scholz and Wimmer show that classifier performance depends on sample size, dimensionality, and data

characteristics, and that no method dominates every complexity scenario [17]. Although their task is classification rather than forecasting, the methodological lesson is relevant: recommendations should be conditioned on the data regime.

Domingos summarises several practical principles that also apply here, including the importance of representation, generalisation, and the mismatch between training data and deployment conditions [22]. A model is only as useful as the assumptions connecting its training distribution to future observations.

### **3.3 Deep Learning Architectures for Forecasting**

Lara Benítez et al. compare recurrent, convolutional, and hybrid deep learning architectures for time series forecasting [11]. Their experiments show that architecture selection affects both accuracy and efficiency. LSTM can be highly accurate in many settings, while convolutional alternatives can be more stable and computationally efficient. The study illustrates that even within deep learning, there is no single architecture that optimises all objectives.

Gamboa reviews the use of deep learning for time series analysis and explains why learned temporal representations can reduce reliance on manual feature engineering [23]. This is a central attraction of recurrent networks: they can learn a hidden state from ordered observations rather than requiring the analyst to define every useful lag interaction. The benefit depends on sufficient data and a sequence design that exposes the relevant temporal scale.

Kong et al. provide a recent survey of deep time series forecasting and organise the field according to model architecture, feature extraction, data sets, and forecasting paradigms [10]. Their synthesis shows a rapid expansion of recurrent, convolutional, and graph based models. It also highlights unresolved challenges related to model complexity, heterogeneous data, and generalisation. The survey supports the choice to treat the present experiment as one controlled comparison rather than an attempt to represent the complete state of the art.

DeepAR demonstrates a setting in which recurrent neural networks are particularly well motivated: probabilistic forecasting across a large collection of related time series [9]. A global model can share information across items and learn scale dependent guest count patterns. The current data set contains one target series and fewer than one thousand daily observations. It therefore lacks one of DeepAR’s main sources of strength: many related sequences. This difference helps explain why positive results from global deep forecasting cannot be transferred automatically to the present case.

### **3.4 Data Size and Scaling Behaviour**

Deep learning scaling research reports systematic relationships among model capacity, data quantity, compute, and loss [20, 21]. These results are often interpreted as support for larger models and larger data sets. Such scaling laws are empirical regularities within specific domains

and regimes. They do not imply that adding old or mismatched observations always improves a forecast.

For time series forecasting, sample size has several meanings. It can refer to the number of time points in one series, the number of related series, the number of independent windows created from the series, or the diversity of observed regimes. A daily series with 666 training rows creates fewer than 666 independent learning examples because adjacent windows overlap. The effective information content is therefore lower than the row count suggests.

Elsayed et al. show that feature engineered boosting can remain competitive even when deep architectures are available [8]. DeepAR, by contrast, benefits from sharing patterns across many series [9]. Together, these studies suggest a conditional hypothesis: deep models become more compelling as the number and diversity of sequences increase, while tree based methods may be more data efficient for one modest structured series.

The data size phase in this thesis uses trailing 20%, 50%, and 100% subsets of the original training period. This design retains temporal proximity to the test set but changes more than sample size. The 20% subset is the most recent and may be distributionally closer to the test period. Therefore, a smaller data result can be better either because the model is less affected by obsolete history or because of stochastic variation. The literature on scaling does not justify ignoring this recency effect.

### 3.5 Noise Robustness

Noise robustness has been studied extensively in deep classification. Rolnick et al. show that neural networks can learn useful decision functions even under very high levels of label noise when sufficient clean signal and data are available [12]. Zhang et al. demonstrate that over parameterised networks can also memorise random labels, raising fundamental questions about generalisation [15]. These findings are not contradictory. A network can fit noise and still generalise under some conditions; the outcome depends on data volume, optimisation, implicit regularisation, and noise structure.

The current task is regression with temporally ordered data, not image classification. Gaussian perturbation of the training target changes every noisy observation continuously rather than flipping a discrete class. The recurrent networks also receive lag features derived before the target is perturbed. Consequently, the experiment contains a particular inconsistency: the noisy target used for learning is not fully reflected in the precomputed lag predictors. This can partially preserve clean temporal information. The results should therefore be interpreted as robustness to noisy training labels in the presence of otherwise clean features, not as a complete noisy series simulation.

Tree ensembles respond differently to noise. Random Forest averages many trees and can reduce variance, but individual trees may still partition around noisy targets. Gradient boosting sequen-

tially corrects residuals and can focus strongly on difficult or noisy observations if regularisation is limited. The expected ranking is therefore not obvious. A measured comparison is necessary.

### 3.6 Missing Data and Imputation

Missing values are common in sensor and operational data. Causes may include sensor failure, communication loss, manual omissions, system downtime and delayed integration. The mechanism behind missing data may be completely random, related to observed variables, or related to the missing value itself. These mechanisms affect whether a simple imputation is unbiased.

BRITS treats missing values as variables within a bidirectional recurrent system and updates them during backpropagation [13]. The method uses temporal dynamics and consistency between forward and backward directions. Cini et al. use graph neural networks to model spatial and temporal dependencies for multivariate imputation [14]. Both methods are substantially more expressive than mean imputation.

The purpose of the missing data phase in this thesis is different. It does not benchmark imputation algorithms. It asks whether the forecasting models remain useful after a simple, widely accessible repair. Ten per cent of the values in temperature, precipitation, lag one, and lag seven are independently removed in the training set and replaced by column means. The procedure reduces local temporal variation, particularly in lag features. A model that depends heavily on precise lag values may therefore deteriorate.

Because the test predictors remain complete, the experiment measures damage to model estimation, not resilience during live inference. A production system would also need a rule for missing future weather or lag values. Advanced imputation methods from the literature provide possible extensions.

### 3.7 Distribution Shift and Concept Drift

Gama et al. define and review concept drift adaptation for learning systems exposed to changing data streams [16]. Drift can affect the prior distribution of the target, the predictor distribution, or the conditional relationship between predictors and target. Adaptation strategies include retraining, sliding windows, ensembles, weighting recent observations, and explicit change detectors.

The current study does not implement an adaptive learner. Instead, it uses a chronological split in which the final 30% of the series forms the test set. This design tests whether a model trained on older observations can be applied to a later period. While it is a useful practical test, it should not be confused with a formal concept drift experiment. Although the later period has a higher average guest count, the cause of this shift is not isolated. Calendar mix, business growth, operational change, and seasonal composition are all possible explanations.

A stronger drift study would compare distributions statistically, use multiple rolling windows,

detect change points, and evaluate retraining policies. The current phase provides initial evidence about temporal generalisation and motivates such future work.

### **3.8 Efficiency, Reproducibility, and Baseline Quality**

Model evaluation is vulnerable to weak baselines, selective reporting, and incomplete implementation details. Ferrari Dacrema et al. demonstrate this problem in neural recommendation research, where several claimed advances become less convincing under careful reproduction and tuned baselines [24]. The domain differs, but the warning is applicable: novelty and complexity should not substitute for controlled comparison.

Forecast evaluation has additional pitfalls, including inappropriate random splits, inconsistent horizons, leakage from preprocessing, and metrics that do not match the decision context. Hewamalage et al. discuss common evaluation errors and recommend transparent, task appropriate procedures [25]. Bergmeir et al. examine conditions under which cross validation is valid for autoregressive prediction and reinforce the need to account for dependence [26].

Reproducibility requires more than publishing code. Package versions, hardware, seeds, preprocessing order, split dates, and stochastic repetition can all affect results. Seeds are set for the statistical environment, the numerical backend, and TensorFlow. This is better than leaving all randomness uncontrolled. However, models are evaluated once, and seeds are not reset before every fit. Differences between nominally identical full data configurations across experimental runs illustrate why repeated runs are preferable.

Efficiency measurement also requires care. Training time depends on the execution environment, so the benchmark uses a warm up and repeated runs and reports mean and standard deviation. Framework specific serialised model size is used for external pointer models. The result remains environment specific rather than hardware independent.

### **3.9 Synthesis and Derived Research Need**

The literature supports five conclusions that shape the experiment.

First, strong classical baselines can match or outperform deep models, particularly when feature engineering is effective and data are limited [8]. Second, recurrent architectures can model complex temporal dependencies, but their benefit depends on scale, horizon, and task structure [10, 11]. Third, noise and missing data are distinct problems and should not be reduced to one generic notion of data quality [12–14]. Fourth, later period evaluation is essential because model rankings obtained from one historical regime may not persist [16]. Fifth, conclusions are only as credible as their baselines, splits, measurements, and reproducibility controls [24, 25].

The present thesis combines these points in one application specific framework. The study does not claim to close every gap. It provides a structured empirical answer for one real world daily series and identifies the experimental improvements required before broader generalisation is

justified.

## 4 Data Understanding and Preprocessing

### 4.1 Data Source and Forecasting Target

The empirical analysis uses an anonymised data set from one canteen located in Hamburg, Germany. Each row represents one calendar day and records the number of people who visited that canteen on that day. The data therefore describe a daily guest count at a single location. They do not measure the number of meals requested by each guest and they do not combine several canteen sites.

The original R object contains 840 daily observations and 13 variables, covering approximately January 2023 to April 2025. After creating one day and seven day lag predictors, the first seven observations lack a complete lag history and are removed. The processed modelling data contain 833 ordered observations. The target variable, `n_guest`, is the observed daily number of canteen guests. The forecasting task is a rolling one day ahead prediction based on weather information for Hamburg, holiday indicators, an operational status variable, and recently observed guest counts.

Hamburg weather variables are used because the canteen itself is located in Hamburg. The geographical match is therefore intentional rather than an arbitrary choice of weather station. The identity of the organisation is anonymised, but the city and the fact that the observations refer to one canteen are methodologically relevant and are stated explicitly.

### 4.2 Original Variables

Table 4.1 summarises the variables stored in the original data object. Some variables are used directly, some are renamed, and some are retained only for descriptive analysis.

| Variable         | Type        | Description                                         |
|------------------|-------------|-----------------------------------------------------|
| date             | Date        | Calendar date of the observation                    |
| n_guest          | Numeric     | Daily guest count and forecasting target            |
| year, month, day | Numeric     | Calendar components                                 |
| weekday          | Categorical | Name of the weekday                                 |
| hh_sh            | Logical     | Indicator of a school holiday in Hamburg            |
| name_sh          | Categorical | Name of the school holiday period, where applicable |
| hh_ph            | Logical     | Indicator of a public holiday in Hamburg            |
| name_ph          | Categorical | Name of the public holiday, where applicable        |
| temp_hamburg     | Numeric     | Daily temperature measurement for Hamburg           |
| precip_hamburg   | Numeric     | Daily precipitation measurement for Hamburg         |
| weather_type     | Categorical | Derived weather classification                      |

**Table 4.1:** Variables available in the original daily dataset

The model scripts use temperature and precipitation rather than the categorical weather type. School and public holidays are used as binary variables. Calendar year, month, day, and weekday are not supplied directly to the models. Weekly behaviour is represented mainly through the seventh lag. This is a compact design, although adding explicit weekday indicators could allow the models to distinguish weekly structure without relying entirely on observed target history.

### 4.3 Preprocessing Pipeline

Before preprocessing, one observation with a recorded value of 10 guests on a Saturday was corrected to zero. The date was identified as a closed day, and the recorded value was inconsistent with the operating schedule and the surrounding weekly pattern. This was a manual data quality correction rather than a model generated adjustment. The original value, corrected value, date, and justification are documented in the supplementary data quality record to preserve reproducibility.

The observations were first ordered chronologically. Two lagged target variables were then constructed. The first lag represents the guest count from the immediately preceding day, while the seventh lag represents the guest count from the corresponding weekday one week earlier. The school and public holiday indicators were renamed consistently for use throughout the analysis. Observations without a complete lag history were excluded, which removed the first seven rows of the ordered series.

An operational status indicator was also created to distinguish zero count days from positive count days. This indicator is highly informative because it separates closed or inactive days from normal operating days. However, an operational status variable is suitable for forecasting only

when the opening schedule is known independently before the forecast is issued. If the variable is derived directly from the guest count on the same day, it introduces target leakage.

The reported experiments retain the operational status variable because it is part of the supplied experimental design. Its potential leakage effect is therefore treated explicitly as a limitation. For operational deployment, the variable should be obtained from an independently maintained opening calendar or excluded before the models are retrained.

#### 4.4 Selected Predictors

Seven predictors are supplied to each model:

| Predictor      | Form    | Rationale                                                                                              |
|----------------|---------|--------------------------------------------------------------------------------------------------------|
| temp_hamburg   | Numeric | Weather conditions may influence daily visitor behaviour.                                              |
| precip_hamburg | Numeric | Rainfall may affect travel decisions and attendance levels.                                            |
| school_holiday | Binary  | School holidays may change work schedules and family routines.                                         |
| public_holiday | Binary  | Public holidays may influence opening conditions and the daily number of guests.                       |
| canteen_open   | Binary  | Represents the operational status of the canteen and must be known in advance to prevent data leakage. |
| lag_1          | Numeric | Represents short term persistence based on the previous day's guest count.                             |
| lag_7          | Numeric | Represents recurring weekly patterns in the daily guest count.                                         |

**Table 4.2:** Predictors used in the forecasting models

The features were selected from domain knowledge rather than an automated selection algorithm. This choice keeps the experiment interpretable and ensures that every model receives the same core information. It also means that the results reflect the quality of this particular representation. A different feature set, including weekday indicators, rolling means, month, trend terms, or interactions, could change the ranking.

#### 4.5 Descriptive Statistics

The processed series contains 833 observations. The mean daily guest count is 873.16, the standard deviation is 638.02, and the median is 1,088. Together with 159 zero count days, the difference between the mean and median indicates a non symmetric distribution with distinct closed day and open day regimes. Observed values range from 0 to 1,892 guests.

| Subset                  | Rows | Mean   | Standard deviation | Median  |
|-------------------------|------|--------|--------------------|---------|
| Complete processed data | 833  | 873.16 | 638.02             | 1088.00 |
| Training set, 80%       | 666  | 841.49 | 608.15             | 1064.00 |
| Test set, 20%           | 167  | 999.46 | 733.86             | 1350.00 |

**Table 4.3:** Summary of the processed target and chronological split

Table 4.4 reports descriptive statistics for every model input after preprocessing. For binary variables, the mean also represents the proportion of observations coded as one.

| Variable       | Mean   | Standard deviation | Minimum | Maximum |
|----------------|--------|--------------------|---------|---------|
| temp_hamburg   | 10.18  | 6.49               | 5.70    | 24.30   |
| precip_hamburg | 2.42   | 4.87               | 0.00    | 43.10   |
| school_holiday | 0.25   | 0.43               | 0       | 1       |
| public_holiday | 0.03   | 0.16               | 0       | 1       |
| canteen_open   | 0.81   | 0.39               | 0       | 1       |
| lag_1          | 871.67 | 637.87             | 0.00    | 1892.00 |
| lag_7          | 868.89 | 638.47             | 0.00    | 1892.00 |

**Table 4.4:** Descriptive statistics of the model predictors after preprocessing

The average for the test period is higher than for the training period. The standard deviation for the test period is also higher. This indicates that the final 20% of the observations differ in level and variability from the earlier training period. This makes the test more realistic but more challenging.

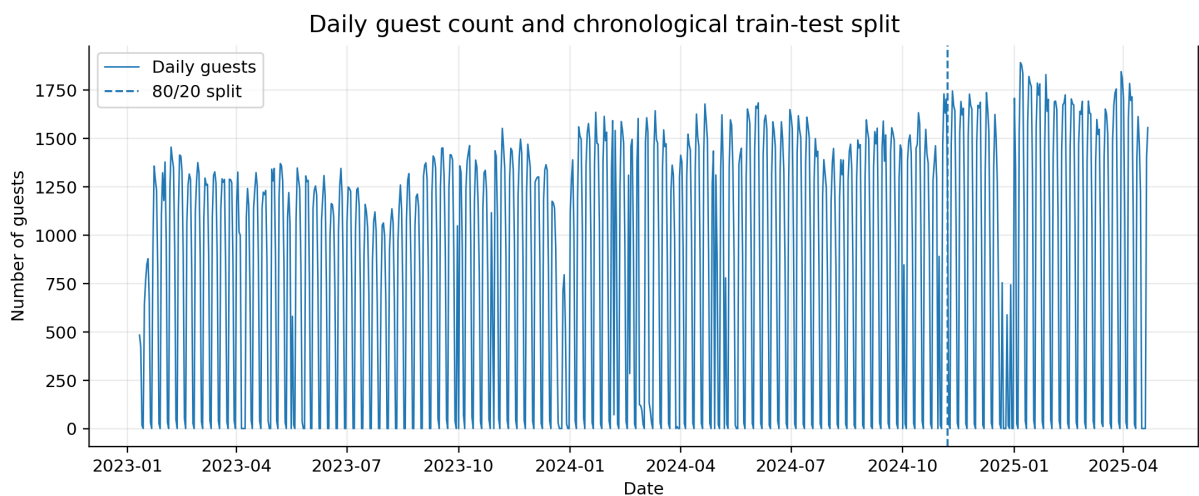
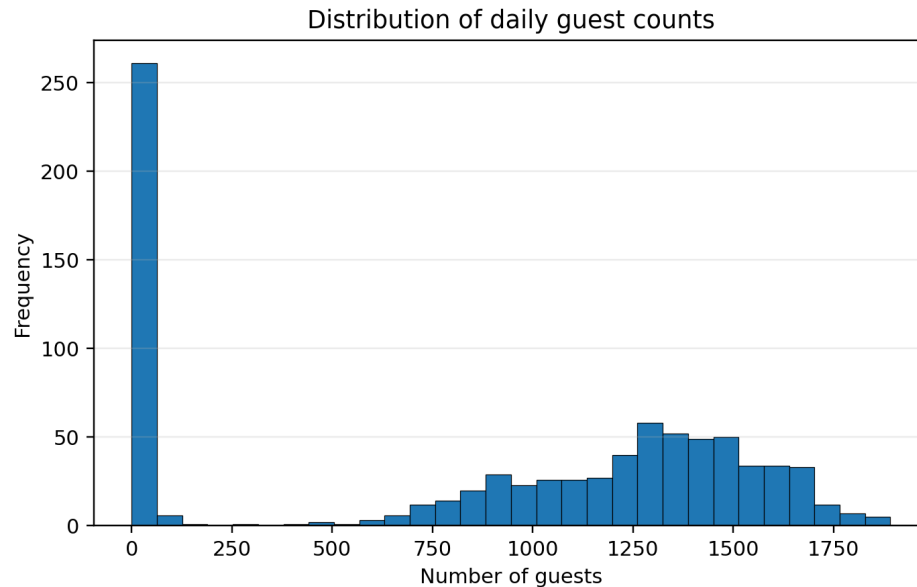
**Figure 4.1:** Daily guest count and the primary chronological 80/20 split

Figure 4.1 shows pronounced weekly oscillation and changes in level over the observation period. The repeated near zero values are consistent with closed or minimally active days. The later period includes several high observations, which is consistent with the higher mean in the test set.

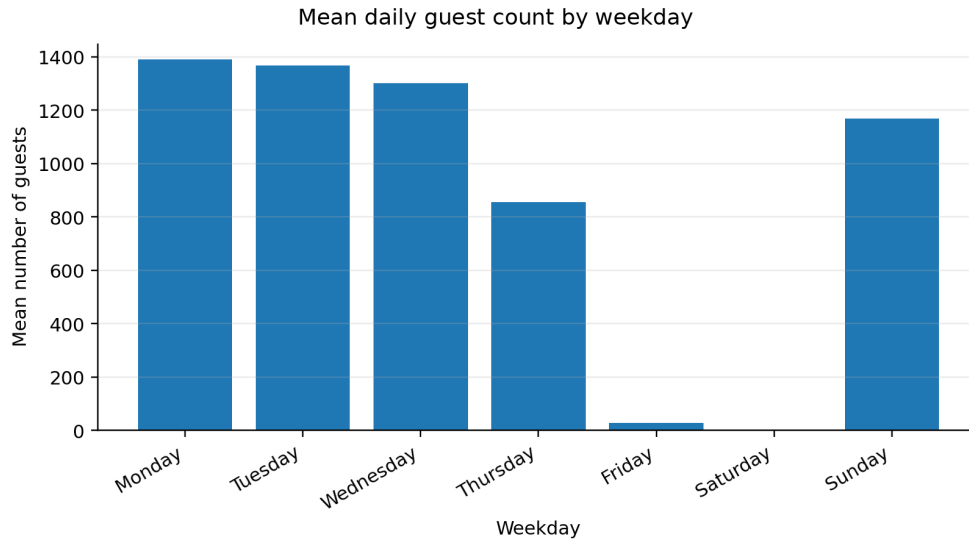


**Figure 4.2:** Distribution of the processed daily guest target

The histogram in Figure 4.2 confirms a mixed distribution with a concentration near zero and a broad positive count range. A single linear relationship is unlikely to capture every aspect of this structure. At the same time, the strong operational pattern can make the task easier if opening status is known.

## 4.6 Weekly Structure

The mean target differs strongly across weekdays, as illustrated in Figure 4.3. Several days have high average attendance, while Friday and Saturday are near zero in the supplied labels. Weekly structure is therefore a dominant characteristic of the series.



**Figure 4.3:** Mean daily guest count by weekday

The linear correlation between `lag_7` and the target is approximately 0.85, while the correlation for `lag_1` is approximately 0.48. Temperature and precipitation have weak marginal correlations with the target. These values support the inclusion of weekly lag information. They do not prove causality or indicate how the variables interact nonlinearly.

School holiday days have a lower average target than non school holiday days in the processed data. Public holidays are rare, with only 22 processed observations, so their estimated effect is uncertain. A model can easily fit an unstable holiday relationship when one category has few examples.

## 4.7 Primary Train Test Split

The observations were ordered chronologically before the training and test sets were created. A chronological split was selected instead of a random split because the objective is to forecast future daily guest counts from historical observations. Randomly mixing earlier and later dates could allow future patterns to influence model training and would therefore provide an unrealistic estimate of forecasting performance.

After preprocessing and construction of the one day and seven day lag variables, 833 complete observations remained. The first 80% of these observations were assigned to the training set, while the final 20% were reserved for testing. This produced 666 training observations and 167 test observations.

The processed data cover the period from 8 January 2023 to 19 April 2025. The primary training period extends from 8 January 2023 to 3 November 2024, and the test period extends from 4 November 2024 to 19 April 2025. Consequently, every test observation occurs after all observations used for model training.

The same test period is retained in phases one to three. This allows the baseline, training data size, target noise, and missing predictor results to be compared on the same observations.

Phase four uses an alternative 70/30 chronological split. In that experiment, the first 583 observations are used for training and the remaining 250 observations are used for testing. Because phase four uses a different test period, its absolute errors are interpreted within that phase rather than compared directly with the errors from phases one to three.

Chronological ordering is preserved throughout all experiments. Each model is therefore trained only with information that would have been available before the corresponding forecast period.

### 4.8 Training Data Subsets

The data size experiment uses the most recent 20%, 50%, and 100% of the 666 row primary training set. The approximate subset sizes are 133, 333, and 666 observations. The subsets are taken from the end of the training period so that each reduced data experiment remains chronologically adjacent to the fixed test period.

This design is reasonable for practical forecasting because recent data may be most relevant. It also introduces a confounding factor: data size and historical coverage change together. The 20% subset represents a recent period with a higher average target than the complete training history. A result that favours 20% may therefore reflect recency rather than a true advantage of fewer observations.

An alternative design would use expanding prefixes or multiple randomly selected temporal blocks. Expanding prefixes would answer how performance evolved as more historical days became available at a fixed origin, while trailing windows answer whether older history helps the current forecast. The implemented design addresses the second question.

### 4.9 Noise Scenario

Gaussian noise was added only to the training target. The standard deviation of the perturbation was defined as 10% of the standard deviation of the training target. Given a training target standard deviation of approximately 608 guests, the resulting perturbation had a standard deviation of approximately 61 guests. The test target remained unchanged.

The lag predictors had been created before the target values were perturbed. The models therefore received a noisy training target together with lagged predictors calculated from the original series. This design primarily evaluates robustness to imperfect historical labels rather than robustness to a fully corrupted time series. A stricter alternative would regenerate the lag variables after introducing noise.

## 4.10 Missing Value Scenario

For the missing value experiment, approximately 10% of the training observations were selected independently for each of four predictors: temperature, precipitation, the one day lag, and the seven day lag. The selected entries were treated as missing.

Each missing value was replaced with the arithmetic mean of the corresponding predictor calculated from the remaining non missing training observations. The imputation parameters were therefore estimated only from the training data. No information from the test set was used during imputation.

Because the missing observations were selected independently for each predictor, more than 10% of the training rows could contain at least one missing entry. Under independent selection, the expected proportion of rows containing at least one missing value is approximately

$$10.9^4 = 34.4\%. \quad (4.1)$$

The exact proportion depends on the overlap among the selected observations. The test predictors remained complete, meaning that this experiment evaluates the effect of incomplete historical training data rather than missing inputs during live prediction.

## 4.11 Scaling and Sequence Construction

Only the inputs and targets used by the recurrent neural networks were scaled. For each predictor, the minimum and maximum values were estimated from the current training subset. The same training based parameters were then applied to the corresponding test observations. This prevented information from the test period from influencing the transformation.

The target variable was scaled in the same manner during neural network training. After prediction, the estimated values were transformed back to the original guest count scale before RMSE and MAE were calculated.

The scaled observations were arranged into sequences of seven consecutive days. Each sequence contained seven predictors across seven time steps, and the guest count of the following day served as the prediction target. The resulting input structure therefore represented one complete week of multivariate historical information.

To construct the first test sequence, the final seven observations from the training period were used as historical context. This ensured that the first test day prediction had the same seven day input history as all subsequent predictions.

The test target values were used only as evaluation outcomes and were not included in the input sequences. This sequence construction is consistent with rolling one day ahead forecasting.

## 4.12 Data Quality and Validity Considerations

Three data issues affect interpretation.

First, the operational status feature is derived from the same day target and may create leakage. Second, lagged target values assume that realised previous attendance is available when the next forecast is issued. This is appropriate for daily rolling forecasts but not for a long multi step plan generated without intermediate observations. Third, only one time series is available. The effective number of independent patterns is lower than the row count because adjacent days and overlapping sequences are correlated.

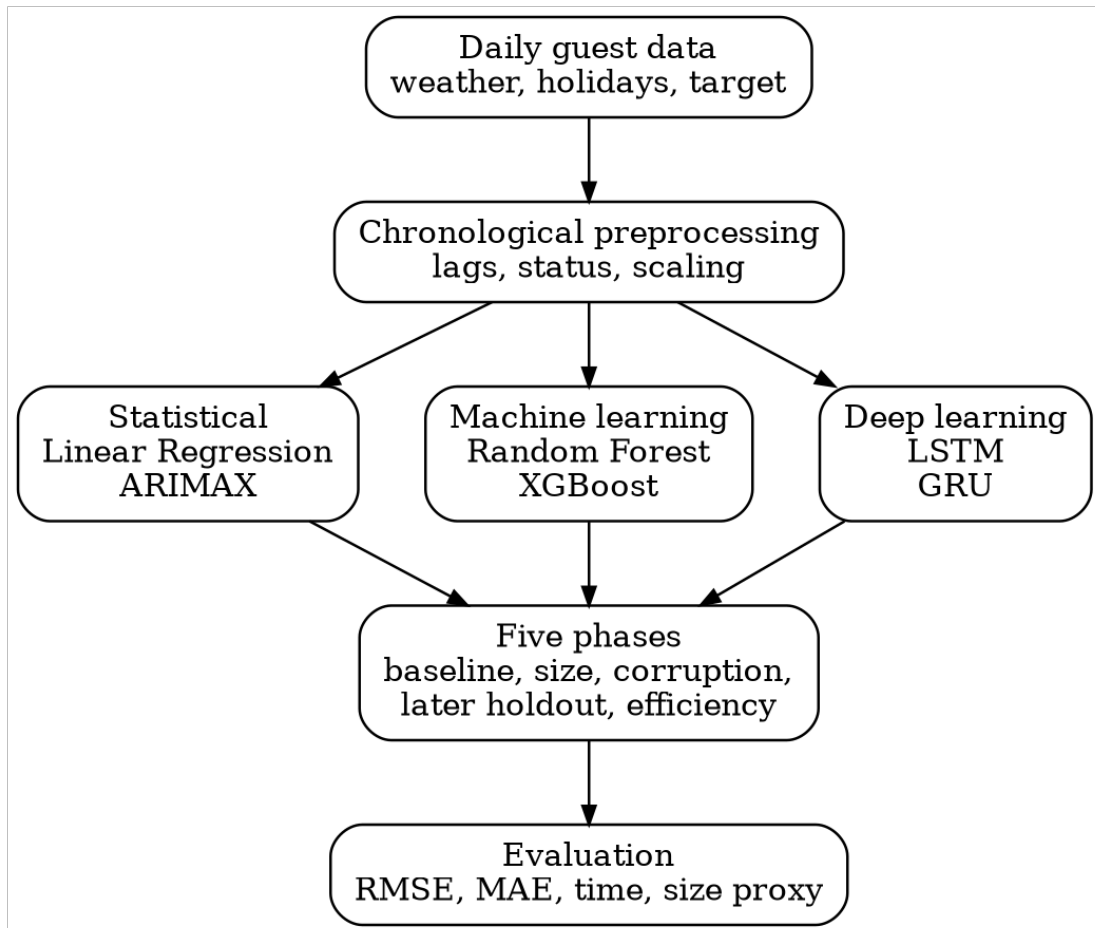
These issues do not make the experiment unusable. They define what the results mean. The conclusions concern a rolling one day forecast with observed recent history and the provided feature construction. Broader claims require revised features, repeated temporal evaluation, and additional series.

## 5 Methodology

### 5.1 Research Design

The study uses a comparative experimental design. Six forecasting models are trained on the same target and core predictors, and their behaviour is observed under five controlled phases. The design is simulation based because data quality and training conditions are deliberately modified, but the underlying time series is real operational data.

The independent experimental factors are model type, training data fraction, training data condition, and temporal split. The dependent measures are RMSE, MAE, elapsed training time, and the reported R object size. The primary unit of evaluation is a daily forecast in the chronological test period.



**Figure 5.1:** Overall experimental workflow

## 5.2 Why These Models Were Selected

The model set is designed to cover different modelling principles with a manageable number of implementations. Six fitted models are complemented by four rule based benchmarks: Seasonal Naive, 7 day Moving Average, 28 day Moving Average, and a four week Seasonal Moving Average. The objective is not to include every forecasting algorithm. An exhaustive list would make tuning inconsistent and interpretation shallow.

**Linear Regression as a Transparent Baseline.** Linear Regression establishes the performance of a simple additive model. It trains quickly, has low implementation complexity, and makes the contribution of advanced methods easier to judge. If an ensemble or neural network cannot improve materially on this baseline, its additional complexity is difficult to justify.

**ARIMAX as a Classical Time Series Baseline.** ARIMAX represents traditional time series modelling and explicitly accounts for autocorrelated residual structure while including external predictors. It complements Linear Regression because two models with the same regressors can behave differently when serial dependence is modelled. The rolling one step update also reflects a common forecasting workflow.

**Random Forest as a Bagging Ensemble.** Random Forest is selected because it is robust, widely used, and effective on nonlinear tabular regression. It represents bagging: many high variance trees are averaged to stabilise predictions. It also requires little preprocessing and can model interactions between weather, holiday, lag, and status features.

**XGBoost as a Boosting Ensemble.** XGBoost represents sequential gradient boosting. It is included because boosted trees are strong forecasting baselines when time series are converted into supervised learning windows [8]. Comparing Random Forest and XGBoost separates the effect of model family from the effect of the ensemble strategy.

**LSTM and GRU as Gated Recurrent Networks.** LSTM and GRU are selected because they are established sequence models with different gating complexity. LSTM uses separate input, forget, and output mechanisms; GRU uses a more compact update structure. Both can learn temporal dependencies without the analyst specifying every interaction. Their use tests the common assumption that sequence aware deep learning should outperform models based on engineered lags.

**Seasonal and Moving Average Benchmarks.** Seasonal Naive repeats the observed guest count from seven days earlier and directly represents the strong weekly recurrence in the data. The 7 day and 28 day moving averages test whether generic smoothing of recent attendance is sufficient, while the four week seasonal moving average combines smoothing with weekday

comparability. These methods require no parameter fitting and provide essential references for judging whether the fitted models add practical predictive value.

**Reasons for Excluding Additional Models.** Prophet, support vector regression, convolutional networks, and other additional model families are excluded to preserve a focused comparison. Their absence limits breadth but improves consistency.

### 5.3 Common Predictor Specification

The statistical and tree based models use the formula

$$y_t = f(T_t, P_t, H_t^{\text{school}}, H_t^{\text{public}}, O_t, y_{t1}, y_{t7}). \quad (5.1)$$

The recurrent networks receive seven consecutive rows of these same seven predictors. This arrangement gives deep models access to a learned weekly sequence while classical models receive explicit lag values for the forecast day.

The feature set is deliberately fixed across phases. Changing the predictors together with the scenario would make it unclear whether the observed performance difference came from the model or representation.

### 5.4 Model Configurations

Table 5.1 lists the principal model settings and provides a concise implementation overview.

| Model             | Main setting                                      | Implementation note                                                                       |
|-------------------|---------------------------------------------------|-------------------------------------------------------------------------------------------|
| Linear Regression | Default ordinary least squares                    | R linear model implementation; additive predictor specification                           |
| ARIMAX            | Automatic ARIMA order with exogenous regressors   | Automatic ARIMA order selection; rolling one step forecasts with observed history updates |
| Random Forest     | 100 trees                                         | Random forest implementation in R; remaining parameters kept at their default values      |
| XGBoost           | 100 boosting rounds; squared error objective      | Gradient boosting implementation in R; remaining parameters kept at their default values  |
| LSTM              | 50 units; one recurrent layer; dense output layer | 20 epochs, batch size 32, Adam optimizer, and MSE loss                                    |
| GRU               | 50 units; one recurrent layer; dense output layer | 20 epochs, batch size 32, Adam optimizer, and MSE loss                                    |

**Table 5.1:** Model configurations used in the experiments

These settings are compact rather than fully optimised. The absence of extensive hyperparameter search avoids tuning one model much more heavily than another. It also means the results compare representative baseline implementations, not the maximum possible performance of each algorithm.

## 5.5 Phase 1: Baseline Performance

In the first phase, all six fitted models and the four rule based benchmarks were evaluated using the primary 80% training period and the final 20% test period. No artificial noise or missing values were introduced. This phase established the reference forecasting performance against which the later experimental conditions were compared.

Linear Regression, Random Forest, and XGBoost produced forecasts using the fitted relationship between the selected predictors and the target. ARIMAX was evaluated using a rolling one step ahead procedure in which the observed guest count became part of the available history after each forecast.

For LSTM and GRU, the predictors and target were scaled using parameters estimated from the training data. The observations were transformed into seven day sequences, and the resulting predictions were returned to the original guest count scale before the error measures were calculated.

This phase identifies the strongest fitted model and the strongest overall forecasting method under the original data condition. It does not by itself evaluate robustness, temporal transfer, or

computational efficiency.

## 5.6 Phase 2: Training Data Size

The second phase examined the effect of training data quantity by repeating the complete comparison of fitted models and rule based benchmarks with 20%, 50%, and 100% of the primary training period. For a training fraction  $q$ , the number of observations was defined as

$$n_q = \lfloor qn_{\text{train}} \rfloor. \quad (5.2)$$

The reduced subsets were taken from the end of the original training period. Consequently, every subset ended immediately before the fixed test period. This design represents a trailing window forecasting situation in which the most recent history is retained.

Data transformations were recalculated separately for every training fraction. The statistical model structure, tree ensembles, scaling parameters, and recurrent sequences were therefore estimated using only the observations available in the corresponding subset.

This phase evaluates two competing effects. A larger sample can improve parameter estimation and expose the models to a wider range of patterns. However, older observations may be less representative of the later test period. The experiment therefore measures the combined effect of data quantity and historical coverage rather than assuming that more observations must always improve forecasting accuracy.

## 5.7 Phase 3: Noise and Missing Values

The third phase evaluated the fitted models and rule based benchmarks under three training conditions: the original clean data, a noisy training target, and incomplete training predictors. The same chronological test set was retained in all three conditions and was not artificially modified. This design allowed the effect of the training data condition to be isolated while preserving a common evaluation period.

**Target Noise Condition.** An independent Gaussian perturbation was added to every training target observation. The perturbation had a mean of zero and a standard deviation equal to 10% of the standard deviation of the original training target. The lag predictors remained based on the unmodified series. The condition therefore measures robustness to noisy historical labels while the remaining predictors retain their original information. It does not represent a fully corrupted time series in which the future lag values would also contain the same measurement errors.

**Missing Predictor Condition.** Approximately 10% of the training values were removed independently from temperature, precipitation, the one day lag, and the seven day lag. The holiday

indicators and operational status variable were not altered. Missing entries were replaced with the mean of the corresponding predictor calculated from the non missing training observations. The test data were not used to estimate the imputation values. This condition therefore measures how the forecasting models respond when the historical predictors are incomplete and repaired using a simple training based imputation procedure.

## **5.8 Phase 4: Later Temporal Holdout**

Phase four changes the chronological split from 80/20 to 70/30. The first 583 processed rows are used for training and the final 250 rows for testing. The fitted models, predictors, and rule based benchmarks otherwise remain unchanged.

The phase was originally described as distribution shift analysis. It is more precisely a later holdout stress test. The later period has a higher mean target, which is compatible with a distribution change. Nevertheless, the experiment does not separate target shift from reduced training size, changed season composition, or longer test duration.

The research question is therefore answered conservatively: the phase indicates which model transferred best to this later period, not which model has a general drift adaptation capability.

## **5.9 Phase 5: Efficiency**

The fifth phase compared the computational requirements of the six models. Training time was measured as elapsed wall clock time. Data loading, test set prediction, and error calculation were excluded from the timing interval so that the measurement focused on model construction and training.

Because elapsed time can be influenced by software initialisation, caching, and other processes running on the computer, one preliminary warm up fit was excluded. Each model was then trained ten times under the same configuration. The arithmetic mean of the ten elapsed times was used as the principal training time measure, while the standard deviation, minimum, and maximum were retained to show run to run variation.

Model size was measured using a framework appropriate procedure. For Linear Regression, ARIMAX, and Random Forest, the fitted R object was measured. XGBoost was measured through its serialised model representation. For LSTM and GRU, the object size was obtained in the Python TensorFlow/Keras environment so that the underlying neural network object was represented rather than only the small R interface object.

These measurements describe elapsed training time and fitted model object size. They do not represent process level peak memory consumption, because additional memory can be allocated for input batches, intermediate activations, gradients, and backend operations.

## 5.10 Evaluation Metrics

RMSE and MAE are calculated on the original guest count scale. Lower values indicate better predictions. The same test observations are used within phases one to three, allowing direct error comparison. Phase four has a different test set, and its absolute errors are interpreted within that phase.

No statistical significance test is applied. Forecast errors are temporally dependent, and each stochastic model is run once. A simple paired test on daily errors would not fully address model training uncertainty. Repeated seed and rolling origin evaluation is recommended for future work.

## 5.11 Random Seed and Reproducibility

A fixed random seed of 123 was used for the stochastic procedures. The same seed value was applied to the random number generators used by the statistical computing environment, the numerical backend, and the neural network framework.

The seed controls processes such as bootstrap sampling, neural network weight initialisation, noise generation, and missing value selection. Its purpose is not to assign scientific meaning to the number 123, but to provide a documented and reproducible starting state.

A fixed initial seed does not guarantee that every model receives an identical random state. Random number generators advance whenever a random operation is performed. A model trained later in an experimental sequence can therefore receive a different state from a nominally identical model trained in another phase. Hardware specific neural network operations can also remain nondeterministic.

For the efficiency experiment, the relevant random states were reset before each measured fit and the training procedure was repeated ten times. Forecasting accuracy was obtained from one fitted run per experimental configuration, whereas training time was summarised across ten repeated fits.

## 5.12 Implementation Environment

The experiments were conducted in an R based statistical computing environment. The recurrent neural networks were trained using a TensorFlow/Keras backend accessed from the main analysis environment.

For reproducibility, the final experiment record includes the operating system, R version, Python version, relevant package versions, processor, installed memory, and the availability of a hardware accelerator. These details must correspond to the same computer on which the final experiments were executed.

The efficiency comparison excludes one warm up fit and uses ten measured training runs for

every model. Mean elapsed time and standard deviation are reported. Model size is represented by the fitted object for ordinary R models and by the underlying framework object for models managed by external computational frameworks.

These measurements provide information about execution time and fitted model size. They should not be interpreted as hardware independent performance or as direct measurements of peak RAM consumption.

### 5.13 Threats to Experimental Validity

Four validity categories are considered.

**Construct validity** concerns whether the experiment measures the intended concepts. The noise phase measures noisy training targets, not all forms of noisy time series. The missing phase measures mean imputed training predictors, not missing live inputs. The 70/30 phase measures later period transfer, not formal adaptation.

**Internal validity** concerns whether performance differences are caused by the model rather than uncontrolled factors. Single stochastic runs, default hyperparameters, and different ARIMAX update logic can affect the ranking. The same core feature set and chronological test logic reduce but do not eliminate these concerns.

**External validity** concerns generalisation beyond the data set. One daily guest series is insufficient to claim a universal ML versus DL result. The findings are most relevant to small, structured, strongly weekly business guest count series.

**Conclusion validity** concerns whether the evidence supports the stated claims. The thesis reports exact observed errors but avoids claims of statistical superiority. It distinguishes clear results, such as the large training time gap, from uncertain results, such as memory consumption and the cause of phase four differences.

## 6 Results

### 6.1 Presentation of Results

This chapter presents the results of the five experimental phases. Interpretation in relation to the research questions is reserved for Chapter 7. Error values are reported in guests. Lower RMSE and MAE indicate better point forecasts.

The forecasting error tables reproduce the recorded experiment outputs. Stochastic fitting and perturbation scenarios use documented seeds. The efficiency experiment is summarised from ten measured runs after a warm up, using the mean and standard deviation rather than one runtime.

### 6.2 Phase 1: Baseline Performance

Table 6.1 reports the baseline results obtained with the primary 80/20 chronological split.

| Model class      | Model             | RMSE          | MAE           |
|------------------|-------------------|---------------|---------------|
| Statistical      | Linear Regression | 314.75        | 211.27        |
| Statistical      | ARIMAX            | 281.99        | 198.50        |
| Machine learning | Random Forest     | <b>261.54</b> | 199.82        |
| Machine learning | XGBoost           | 273.82        | <b>157.92</b> |
| Deep learning    | LSTM              | 389.00        | 287.90        |
| Deep learning    | GRU               | 358.42        | 252.66        |

**Table 6.1:** Phase one baseline forecasting performance

Among the six fitted statistical, machine learning, and deep learning models, Random Forest produces the lowest baseline RMSE at 261.54, while XGBoost produces the lowest baseline MAE at 157.92. These statements refer only to the six fitted models. When the simple benchmarks reported in Table 6.2 are included, the seasonal naive forecast achieves the lowest overall MAE at 136.13. The distinction between “best fitted model” and “best overall forecasting method” is therefore maintained throughout the discussion and conclusion.

ARIMAX is the strongest fitted statistical model, with RMSE 281.99 and MAE 198.50. Linear Regression has RMSE 314.75 and MAE 211.27. Both recurrent networks have larger errors than the four non neural fitted models. GRU performs better than LSTM on both metrics in this phase. The benchmark comparison also shows why model accuracy must be judged against simple seasonal rules rather than only against other trained models.

### 6.2.1 Simple Weekly and Moving Average Benchmarks

Because lag seven is strongly associated with the target, a seasonal naive forecast is an essential reference. It predicts each test day guest count with the observed count from seven days earlier. Two expanding history moving average forecasts and one seasonal moving average forecast are also included. The seasonal moving average uses the mean guest count from the same weekday over the previous four weeks. Their results are reported in Table 6.2.

| <b>Benchmark</b>                          | <b>RMSE</b>   | <b>MAE</b>    |
|-------------------------------------------|---------------|---------------|
| Seasonal naive, lag 7                     | 366.53        | <b>136.13</b> |
| Moving average, 7 days                    | 714.74        | 625.10        |
| Moving average, 28 days                   | 738.96        | 660.80        |
| Seasonal moving average, previous 4 weeks | <b>388.56</b> | 194.00        |

**Table 6.2:** Simple benchmark performance on the primary 80/20 test split

The seasonal naive benchmark achieves the lowest overall MAE, 136.13, outperforming every fitted model on this metric, including XGBoost at 157.92. Its RMSE of 366.53 is, however, substantially higher than the leading fitted models. Repeating the guest count from the same weekday one week earlier therefore gives a low typical absolute error but occasionally produces large misses. Random Forest, XGBoost, and ARIMAX reduce these large errors sufficiently to obtain lower RMSE values.

The 7 day and 28 day moving average benchmarks perform poorly on both metrics because they smooth across days from different operating regimes. The seasonal moving average performs considerably better, with RMSE 388.56 and MAE 194.00, because it averages comparable weekdays rather than adjacent calendar days. Nevertheless, it does not improve on the seasonal naive MAE or the strongest fitted model RMSE. Taken together, the benchmark results show that weekly recurrence is highly informative, whereas unstructured smoothing of recent attendance is not an adequate forecasting rule for this series.

## 6.3 Phase 2: Effect of Training Data Size

The data size experiment evaluates the models with the most recent 20%, 50%, and 100% of the primary training period. The complete results are shown in Table 6.3.

**Table 6.3:** Phase two performance across training data sizes

| <b>Training fraction</b> | <b>Model</b>         | <b>RMSE</b> | <b>MAE</b> |
|--------------------------|----------------------|-------------|------------|
| 20%                      | Seasonal Naive       | 366.53      | 136.13     |
| 20%                      | 7 day Moving Average | 714.74      | 625.10     |

| Training fraction | Model                   | RMSE          | MAE           |
|-------------------|-------------------------|---------------|---------------|
| 20%               | 28 day Moving Average   | 738.96        | 660.80        |
| 20%               | Seasonal Moving Average | 388.56        | 194.00        |
| 20%               | Linear Regression       | 321.65        | 178.12        |
| 20%               | ARIMAX                  | 264.06        | 160.78        |
| 20%               | Random Forest           | 298.05        | 230.87        |
| 20%               | XGBoost                 | <b>262.29</b> | <b>113.14</b> |
| 20%               | LSTM                    | 588.13        | 529.68        |
| 20%               | GRU                     | 444.89        | 329.23        |
| 50%               | Seasonal Naive          | 366.53        | <b>136.13</b> |
| 50%               | 7 day Moving Average    | 714.74        | 625.10        |
| 50%               | 28 day Moving Average   | 738.96        | 660.80        |
| 50%               | Seasonal Moving Average | 388.56        | 194.00        |
| 50%               | Linear Regression       | 312.35        | 211.61        |
| 50%               | ARIMAX                  | 281.60        | 198.50        |
| 50%               | Random Forest           | <b>261.97</b> | 205.56        |
| 50%               | XGBoost                 | 273.27        | 162.64        |
| 50%               | LSTM                    | 438.63        | 335.96        |
| 50%               | GRU                     | 480.32        | 373.46        |
| 100%              | Seasonal Naive          | 366.53        | <b>136.13</b> |
| 100%              | 7 day Moving Average    | 714.74        | 625.10        |
| 100%              | 28 day Moving Average   | 738.96        | 660.80        |
| 100%              | Seasonal Moving Average | 388.56        | 194.00        |
| 100%              | Linear Regression       | 314.75        | 211.27        |
| 100%              | ARIMAX                  | 281.99        | 198.50        |
| 100%              | Random Forest           | <b>271.57</b> | 210.54        |
| 100%              | XGBoost                 | 273.82        | 157.92        |
| 100%              | LSTM                    | 373.83        | 269.64        |
| 100%              | GRU                     | 363.21        | 272.29        |

At 20%, XGBoost has the lowest overall RMSE and MAE. Its RMSE of 262.29 is slightly lower than ARIMAX at 264.06, and its MAE of 113.14 is also lower than the seasonal naive benchmark. At 50% and 100%, Random Forest has the lowest RMSE among all evaluated methods, while the seasonal naive benchmark has the lowest overall MAE at 136.13. XGBoost remains the fitted model with the lowest MAE at both training fractions. The benchmark values are nearly unchanged across the three fractions because these rule based methods require only recent historical observations rather than fitting parameters to the complete training sample.

LSTM improves strongly as the training fraction increases. Its RMSE decreases from 588.13 at 20% to 438.63 at 50% and 373.83 at 100%. This is a 36.4% reduction from the smallest to the largest subset. GRU is not monotonic: its RMSE increases from 444.89 at 20% to 480.32 at 50%, then decreases to 363.21 at 100%.

The non neural models do not improve monotonically. ARIMAX and XGBoost achieve their lowest phase two RMSE with the recent 20% subset. Linear Regression changes only moderately. Random Forest improves substantially from 20% to 50%, but its 100% RMSE in this experimental run is slightly worse than its 50% result.

## 6.4 Phase 3: Noise and Missing Training Data

Table 6.4 reports the clean, noisy target, and missing predictor results. The clean rows reproduce the baseline configuration used in the third phase.

**Table 6.4:** Phase three performance under training data corruption

| Scenario           | Model                   | RMSE          | MAE           |
|--------------------|-------------------------|---------------|---------------|
| Clean              | Seasonal Naive          | 366.53        | <b>136.13</b> |
| Clean              | 7 day Moving Average    | 714.74        | 625.10        |
| Clean              | 28 day Moving Average   | 738.96        | 660.80        |
| Clean              | Seasonal Moving Average | 388.56        | 194.00        |
| Clean              | Linear Regression       | 314.75        | 211.27        |
| Clean              | ARIMAX                  | 281.99        | 198.50        |
| Clean              | Random Forest           | <b>261.54</b> | 199.82        |
| Clean              | XGBoost                 | 273.82        | 157.92        |
| Clean              | LSTM                    | 389.00        | 287.90        |
| Clean              | GRU                     | 358.42        | 252.66        |
| Target noise       | Seasonal Naive          | 366.53        | <b>137.14</b> |
| Target noise       | 7 day Moving Average    | 714.70        | 625.15        |
| Target noise       | 28 day Moving Average   | 738.94        | 661.03        |
| Target noise       | Seasonal Moving Average | 388.88        | 195.54        |
| Target noise       | Linear Regression       | 314.88        | 211.43        |
| Target noise       | ARIMAX                  | 284.82        | 197.61        |
| Target noise       | Random Forest           | <b>280.59</b> | 215.16        |
| Target noise       | XGBoost                 | 293.88        | 189.75        |
| Target noise       | LSTM                    | 386.54        | 279.69        |
| Target noise       | GRU                     | 401.11        | 292.57        |
| Missing predictors | Seasonal Naive          | 366.53        | <b>136.13</b> |
| Missing predictors | 7 day Moving Average    | 714.74        | 625.10        |

| Scenario           | Model                   | RMSE          | MAE    |
|--------------------|-------------------------|---------------|--------|
| Missing predictors | 28 day Moving Average   | 738.96        | 660.80 |
| Missing predictors | Seasonal Moving Average | 388.56        | 194.00 |
| Missing predictors | Linear Regression       | 313.89        | 220.02 |
| Missing predictors | ARIMAX                  | <b>238.76</b> | 149.69 |
| Missing predictors | Random Forest           | 283.18        | 221.17 |
| Missing predictors | XGBoost                 | 268.81        | 175.97 |
| Missing predictors | LSTM                    | 386.15        | 281.77 |
| Missing predictors | GRU                     | 369.47        | 283.44 |

The seasonal naive benchmark has the lowest overall MAE in all three scenarios: 136.13 on clean data, 137.14 under target noise, and 136.13 under missing predictors. Its RMSE remains higher than the leading fitted models. The moving average benchmarks are also almost unchanged, particularly under missing predictors, because they use only the historical target series and not the corrupted exogenous predictors. Linear Regression changes very little in RMSE across the three conditions. Its MAE rises from 211.27 to 220.02 in the missing predictor scenario. Random Forest and XGBoost deteriorate under target noise: their RMSE values increase by approximately 7.3% relative to the clean phase three run. GRU shows the largest RMSE increase under target noise, rising by approximately 11.9%.

LSTM's RMSE is almost unchanged under target noise and missing predictors. Its absolute accuracy remains weaker than the tree based and statistical models. Stability of a high error is therefore not equivalent to good forecasting performance.

ARIMAX improves unexpectedly under the missing predictor scenario, reaching RMSE 238.76 and MAE 149.69. XGBoost also improves slightly in RMSE relative to its clean run, although its MAE becomes larger. These outcomes are reported as observed; the experiment does not establish that missing values improve forecasting.

## 6.5 Phase 4: Performance on the Later 30% Holdout

Phase four uses the 70/30 chronological split. Table 6.5 gives the resulting errors.

| <b>Model</b>            | <b>RMSE</b>   | <b>MAE</b>    |
|-------------------------|---------------|---------------|
| Seasonal Naive          | 358.52        | <b>134.53</b> |
| 7 day Moving Average    | 697.60        | 614.25        |
| 28 day Moving Average   | 712.70        | 637.91        |
| Seasonal Moving Average | 346.15        | 164.13        |
| Linear Regression       | 309.49        | 209.04        |
| ARIMAX                  | 285.88        | 201.85        |
| Random Forest           | 250.02        | 181.92        |
| XGBoost                 | <b>248.67</b> | 139.24        |
| LSTM                    | 367.27        | 268.01        |
| GRU                     | 339.56        | 239.18        |

**Table 6.5:** Phase four results for the 70/30 chronological split

XGBoost has the lowest overall RMSE at 248.67 and the lowest MAE among the fitted models at 139.24. The seasonal naive benchmark has the lowest overall MAE at 134.53. Random Forest is close to XGBoost in RMSE, with a difference of 1.35 guests, but has a substantially higher MAE. The seasonal moving average performs better than the two ordinary moving averages but remains weaker than XGBoost and Random Forest in RMSE. GRU again performs better than LSTM, but both recurrent models remain less accurate than the tree based models and ARIMAX.

The phase four test set differs from the phase one test set. The numerical reduction in RMSE for several models relative to phase one must therefore not be interpreted as an improvement caused solely by distribution shift. The relevant result is the within phase ranking.

## 6.6 Phase 5: Repeated Training Time and Model Size

Training time can be affected by background processes, software initialisation, caching, and operating system activity. One preliminary warm up fit was therefore excluded, after which every model was trained ten times. Table 6.6 reports the mean elapsed training time, its standard deviation, the observed range, and the framework appropriate fitted model size.

| Model             | Runs | Mean time (s) | Std. dev. (s) | Range (s) | Model size (KB) |
|-------------------|------|---------------|---------------|-----------|-----------------|
| Linear Regression | 10   | 0.006         | 0.008         | 0.00 0.02 | 250.31          |
| ARIMAX            | 10   | 0.330         | 0.014         | 0.31 0.36 | 110.90          |
| Random Forest     | 10   | 0.091         | 0.038         | 0.07 0.20 | 1160.53         |
| XGBoost           | 10   | 0.409         | 0.043         | 0.37 0.50 | 348.65          |
| LSTM              | 10   | 3.982         | 0.272         | 3.60 4.33 | 160.13          |
| GRU               | 10   | 4.525         | 0.420         | 4.06 5.46 | 127.89          |

**Table 6.6:** Repeated training time and framework appropriate fitted model size

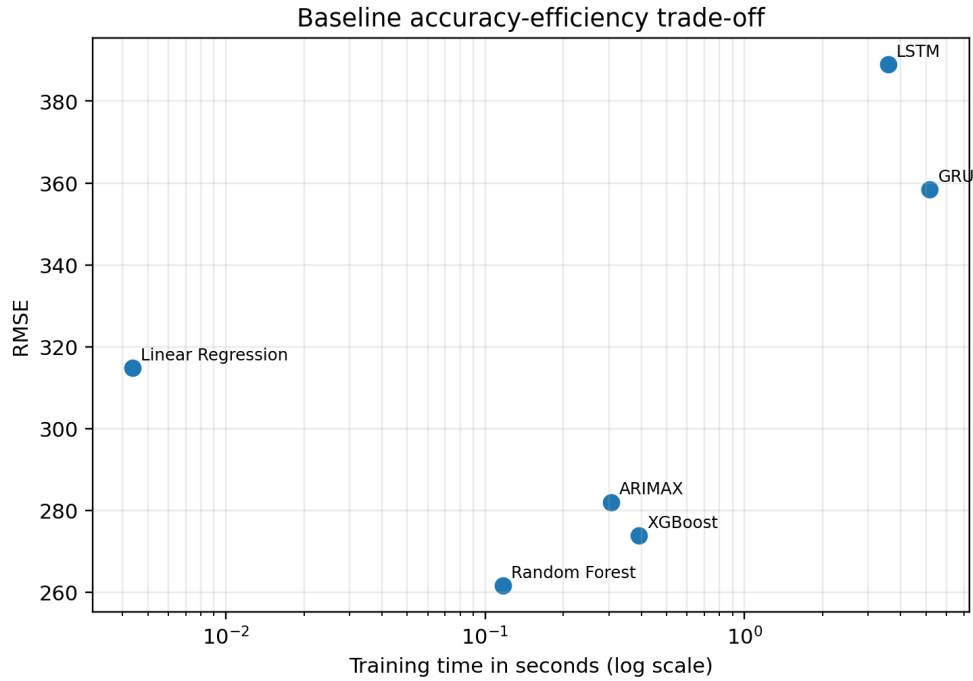
The table confirms that Linear Regression is the fastest model. Random Forest also trains quickly despite producing the lowest baseline RMSE. XGBoost requires more time than Random Forest but remains below half a second on average. The recurrent networks require several seconds per fit and show greater run to run variation.

The model size values are not all based on the same low level representation. The ordinary statistical and forest models use fitted R objects, XGBoost uses its framework representation, and the LSTM and GRU sizes are obtained from the Python TensorFlow/Keras model objects. The comparison therefore avoids reporting only the small R interface pointer for the neural models. These values describe fitted model object size rather than peak memory consumed during training.

## 6.7 Accuracy Efficiency Relationship

A Pareto front, also called a skyline, contains the models for which no alternative is simultaneously more accurate and less computationally demanding. A model is dominated when another model has both a lower forecast error and a lower training time. The concept does not combine unlike quantities into one arbitrary score; instead, it identifies efficient choices under multiple criteria. Figure 6.1 is interpreted using this dominance relation.

Figure 6.1 combines phase one RMSE with the mean training time from the ten measured efficiency runs. Models toward the lower left provide lower error and shorter training time.



**Figure 6.1:** Baseline RMSE versus mean training time across repeated runs

The Pareto interpretation must use the mean repeated runtime rather than one execution. A model belongs to the front only when no alternative has both a lower RMSE and a lower mean training time. Runtime standard deviations should also be considered when two means are similar.

## 6.8 Open Day Robustness Check

Closed days form a distinct zero count regime and can make forecasting appear easier than planning on operating days. The six models were therefore evaluated both on the complete chronological test set and on the subset of test dates for which the canteen was open. For every model, RMSE and MAE were recalculated after filtering out closed days while retaining the same fitted model and predictions.

| Model             | All days RMSE | Open days RMSE | All days MAE | Open days MAE |
|-------------------|---------------|----------------|--------------|---------------|
| Linear Regression | 314.75        | 310.00         | 211.27       | 209.73        |
| ARIMAX            | 281.99        | 272.93         | 198.50       | 195.49        |
| Random Forest     | 261.54        | 286.40         | 199.82       | 232.96        |
| XGBoost           | 273.82        | 301.15         | 157.92       | 187.09        |
| LSTM              | 389.00        | 341.27         | 287.90       | 283.86        |
| GRU               | 358.42        | 302.87         | 252.66       | 240.89        |

**Table 6.7:** Forecasting accuracy on all test days and open canteen days

After closed days were excluded, ARIMAX achieved the lowest open day RMSE of 272.93, while XGBoost achieved the lowest open day MAE of 187.09. The ranking therefore changed compared with the all day evaluation, where Random Forest had the lowest RMSE and XGBoost had the lowest MAE.

Random Forest and XGBoost showed higher errors on open days than on the complete test set. This indicates that their all day performance was partly supported by the comparatively easier prediction of closed or zero count days. By contrast, the open day RMSE values of Linear Regression, ARIMAX, LSTM, and GRU were lower than their corresponding all day RMSE values.

The results show that model robustness depends on whether closed days are included in the evaluation. XGBoost remained the strongest model according to MAE, while ARIMAX became the strongest model according to RMSE on open canteen days. For operational planning, the open day results are therefore more relevant than the complete test results because staffing and food preparation decisions are made primarily for days on which the canteen is operating.

The opening status indicator used in this retrospective evaluation was taken from the recorded data. In an operational forecasting system, this status must be available independently from an opening schedule before the forecast is generated.

## 6.9 Consolidated Phase Wise Ranking

Table 6.8 summarises the best forecasting method by metric within each directly reported accuracy condition.

| Accuracy condition                  | Lowest RMSE overall | Lowest MAE overall |
|-------------------------------------|---------------------|--------------------|
| Baseline 80/20 split                | Random Forest       | Seasonal Naive     |
| 20% training subset                 | XGBoost             | XGBoost            |
| 50% training subset                 | Random Forest       | Seasonal Naive     |
| 100% training subset                | Random Forest       | Seasonal Naive     |
| Noisy training target               | Random Forest       | Seasonal Naive     |
| Missing and mean imputed predictors | ARIMAX              | Seasonal Naive     |
| Later 30% holdout                   | XGBoost             | Seasonal Naive     |
| Open day subset, fitted models only | ARIMAX              | XGBoost            |

**Table 6.8:** Best observed forecasting method by accuracy condition and evaluation metric

Table 6.8 summarises accuracy rankings only. Training speed is reported separately in Table 6.6 because “fastest training” is an efficiency criterion rather than an RMSE or MAE condition.

Linear Regression is the fastest fitted model, while the four rule based benchmarks require no parameter fitting and therefore have effectively zero training time. No deep learning model is the best performer in any reported accuracy condition. Seasonal Naive accounts for most overall MAE wins, while Random Forest, XGBoost, and ARIMAX account for the fitted model and RMSE wins. This result is limited to the selected data, architecture, tuning, and evaluation design.

## 7 Discussion

### 7.1 Interpretation Strategy

The results show a clear empirical pattern: Seasonal Naive provides the lowest overall MAE in the main baseline and later holdout evaluations, the selected tree based methods are the strongest fitted models, the statistical models remain competitive, and the recurrent networks do not justify their additional complexity for the available data. That statement is supported by the observed tables, but it requires qualification. The experiment is based on one small series, compact neural architectures, limited tuning, a potentially leaky operational status feature, and stochastic variation. The purpose of this discussion is therefore to separate robust conclusions from plausible but unproven explanations.

### 7.2 Baseline Comparison

On the primary baseline split, Random Forest achieves the lowest RMSE among all evaluated methods, Seasonal Naive achieves the lowest overall MAE, and XGBoost achieves the lowest MAE among the six fitted models. These different winners reveal a meaningful difference in error distribution. XGBoost is more accurate on the average day, but Random Forest appears to avoid some large misses that receive greater weight in RMSE. In an operational setting, the preferred metric should depend on the cost of forecast errors. If a few severe under forecasts cause major staffing or supply problems, RMSE is relevant. If each additional absolute guest error has approximately constant cost, MAE is easier to interpret.

The strong performance of tree ensembles is consistent with the structure of the data. The predictor set is small, largely tabular, and contains powerful lag and status information. Random Forest and XGBoost can learn thresholds and interactions without estimating a large sequence model. The result also aligns with the argument of Elsayed et al. that feature engineered boosting can outperform more complex deep architectures on forecasting tasks [8].

ARIMAX performs reasonably well and improves on Linear Regression in baseline RMSE. This suggests that modelling serial structure adds value beyond the additive exogenous relationship. However, ARIMAX receives a methodological advantage in the test loop: the actual target is appended after every one step forecast. The tree and neural models also receive precomputed lags that contain realised past guest counts, so the comparison is broadly aligned with a rolling daily use case. It would not be aligned for a multi day forecast issued all at once.

LSTM and GRU underperform. The simplest explanation is insufficient data relative to model capacity. With 666 training rows and overlapping seven day windows, the recurrent networks see only a modest number of distinct patterns. Another explanation is that weekly guest count

is already summarised efficiently by lag seven. A recurrent network has little advantage when the strongest temporal relationship is explicitly supplied to every model. The result does not establish that LSTM and GRU are poor forecasting methods. It shows that their theoretical ability to model sequences did not translate into better test error under this particular representation and training regime.

### 7.3 Research Question 1: Effect of Data Size

**RQ1 asks how the amount of training data affects statistical, machine learning, and deep learning performance.**

The most direct finding is that LSTM benefits substantially from additional data. Its RMSE falls from 588.13 with 20% of the training set to 373.83 with 100%. This supports the expectation that a recurrent neural network needs more observations to estimate its parameters. GRU also performs best with the full data, although its 50% result is worse than its 20% result. The non monotonic GRU path is likely influenced by stochastic initialisation, optimisation variance, and changes in the historical period.

The tree based and statistical models do not follow a simple more data is better pattern. XGBoost and ARIMAX have their lowest phase two RMSE at 20%. Random Forest improves from 20% to 50%, then becomes slightly worse at 100% in that script. Linear Regression changes little.

A superficial interpretation would be that less data is better for XGBoost and ARIMAX. That conclusion is too strong. The phase two subsets are trailing windows. The recent 20% subset is temporally closer to the test period and has a higher target level than the full history. Its advantage can therefore come from distributional similarity rather than reduced sample size. The experiment combines two factors: number of observations and recency. A clean data volume study would hold the data generating regime constant, which is difficult with one evolving time series.

The Random Forest result also exposes stochastic variation. The 100% phase two RMSE is 271.57, while the nominally similar baseline Random Forest result is 261.54. The model settings and data size are the same, but the random number state differs because Random Forest is trained after other stochastic operations in the experimental sequence. This ten point difference is too large to ignore when interpreting small gaps between models. Repeated seeds are necessary before claiming that the 50% Random Forest is truly better than the 100% version.

The defensible answer to RQ1 is therefore conditional:

- Additional data clearly helps LSTM in the observed runs and eventually helps GRU.
- Tree based and statistical models reach useful performance with much smaller samples.
- More historical observations do not guarantee lower error when older data differ from the test regime.
- The deep learning models remain less accurate than the best classical models even at 100%

of the available training data.

The result challenges a common assumption in both directions. It is incorrect to assume that deep learning should be best merely because it can learn sequences. It is also incorrect to state that data size had no effect on deep learning; LSTM improved strongly. The actual conclusion is that the available increase was not enough to close the accuracy gap.

## 7.4 Research Question 2: Robustness to Noise and Missing Values

**RQ2 asks which models remain stable when the training target is noisy or selected predictors are incomplete and imputed.**

Under target noise, Linear Regression changes very little. ARIMAX also changes only slightly. Random Forest and XGBoost deteriorate by approximately 7.3% in RMSE, while GRU deteriorates by approximately 11.9%. LSTM's RMSE is marginally lower than in the clean run. These results suggest that Linear Regression, ARIMAX, and LSTM are numerically stable under the specific perturbation, whereas GRU and the tree ensembles are more strongly affected.

Stability should not be confused with superiority. LSTM remains less accurate than the leading models. A model with an RMSE close to 387 in both conditions is stable, but it is still operationally weaker than an ARIMAX model with an RMSE close to 285. Robustness must therefore be interpreted together with baseline accuracy.

The target noise design also limits the conclusion. Noise is added to the training target after the lag variables have been calculated from the clean target. The models therefore retain clean temporal information in `lag_1` and `lag_7`. A fully corrupted time series would require these lag variables to be recalculated from the noisy target. The current design therefore evaluates resilience to noisy labels supported by clean predictors. It does not measure robustness to a data source that corrupts both the target and its later use as a lagged predictor.

The missing predictor results are more surprising. ARIMAX improves substantially after 10% missingness and mean imputation, while XGBoost's RMSE also improves slightly. These results must not be interpreted as evidence that missing data improve forecasting accuracy. Several alternative explanations are more plausible:

1. Mean imputation reduces predictor variability and can act as accidental regularisation.
2. The random missingness pattern may have removed unusual or influential predictor values that otherwise contributed to overfitting in this particular realisation.
3. Automatic ARIMA order selection may produce a different model after imputation, meaning that the missing predictor condition can create a different ARIMAX specification from the clean baseline.
4. Only one missingness pattern was evaluated, so the result may reflect a favourable random perturbation rather than a systematic robustness advantage.
5. The rolling one step evaluation updates the model history with the observed test target after

each forecast, which can reduce the influence of imperfect training predictors.

6. The operational status feature and the complete test predictors remain highly informative, which may further reduce the effect of missing continuous predictors.

The improved ARIMAX result may therefore indicate specification sensitivity rather than a genuine benefit from missing data. The correct answer to RQ2 is that Linear Regression and ARIMAX are the most stable under the noisy target condition, while ARIMAX achieves the best result in the single missing predictor experiment. The improvement observed after imputation is not a causal robustness finding. It should be treated as an experimental anomaly that requires repeated missingness masks, inspection of the selected ARIMA orders, and sensitivity analysis with alternative imputation methods.

### 7.5 Research Question 3: Generalisation to a Later Period

**RQ3 asks whether deep learning models generalise better than statistical and tree based models under a later temporal holdout.**

The answer from phase four is no. XGBoost produces the lowest RMSE and MAE, followed closely by Random Forest in RMSE. LSTM and GRU remain less accurate than all tree based models and ARIMAX. The recurrent models do not show an advantage under the later 30% test period.

This result is consistent with the baseline and suggests that the leading tree ensembles are not merely fitting a short final test window. They remain strong when the test period is lengthened and the training period is reduced to 70%.

The phase should not be over labelled as concept drift adaptation. Both the primary and phase four experiments are chronological. The second simply moves the split boundary earlier. The later test period has a higher target mean, which indicates distributional change, but the design does not identify whether the conditional mapping has changed. XGBoost's good performance may arise because lag seven and operational status continue to carry most of the predictive signal. A formal concept drift study would compare rolling error over time, detect change points, and evaluate adaptation policies.

The conservative conclusion is that the deep models did not generalise better in the observed later holdout scenario. No evidence supports choosing LSTM or GRU for temporal robustness in this data set.

### 7.6 Research Question 4: Computational Requirements

**RQ4 asks what differences in repeated training time and framework appropriate model size are observed among the selected models.**

In the observed configuration, the deep learning models do not create this trade off because they are both slower and less accurate. Across ten measured runs, LSTM requires an average of

3.982 seconds and GRU 4.525 seconds, compared with 0.091 seconds for Random Forest and 0.409 seconds for XGBoost. The fastest model, Linear Regression, is less accurate than the tree ensembles but still far faster than every other method.

Random Forest provides the strongest baseline RMSE and trains quickly. XGBoost provides the strongest baseline MAE and the best phase four performance, with a mean training time of approximately 0.41 seconds. For this data set, the tree ensembles occupy the favourable accuracy efficiency region.

The absolute times are small, so practical importance depends on deployment. Training for five seconds is not problematic if a model is retrained once per month. It matters more during extensive hyperparameter tuning, repeated cross validation, or deployment on restricted hardware. The relative gap is therefore informative, while the business cost is context dependent.

The serialised model size results describe fitted artifact storage, not peak RAM consumption. A direct RAM comparison would require a separate process level peak memory benchmark.

## 7.7 Why the Deep Models Underperformed

Several factors can explain the recurrent network results.

**Small Effective Sample Size.** The series has 666 rows in the primary training set, and seven step windows overlap heavily. Adjacent samples share six of seven time steps. The network therefore receives fewer independent patterns than a tabular row count suggests. DeepAR and other successful deep forecasting systems often train on many related series, which is not the case here [9].

**Strong Engineered Lags.** Lag seven has a high association with the target and directly represents weekly seasonality. Tree models can use it immediately. The recurrent network must learn how to combine the sequence while also receiving lag variables inside each step. This can be redundant rather than beneficial.

**Limited Tuning.** Both recurrent models use one layer with 50 units, 20 epochs, batch size 32, and default Adam settings. No validation based early stopping, dropout, learning rate schedule, unit search, sequence length search, or repeated training is used. The compact design is fair as a baseline but does not establish the best possible LSTM or GRU performance.

**Target Distribution.** The target mixes 159 zero count observations with a broad range of positive guest counts. The all days and open days comparison in Table 6.7 shows that this mixture is not only a descriptive characteristic; it materially affects measured forecasting accuracy and model ranking. On the complete test set, Random Forest has the lowest RMSE among the six fitted models and XGBoost has the lowest fitted model MAE. After closed days are removed,

ARIMAX obtains the lowest open day RMSE at 272.93, while XGBoost retains the lowest open day MAE at 187.09. Random Forest and XGBoost both record higher errors on open days than on all days, whereas the RMSE of Linear Regression, ARIMAX, LSTM, and GRU decreases.

These results indicate that the zero count closure regime changes the error distribution and can support models that identify closed days effectively. The open day subset represents the operationally narrower but practically important task of forecasting attendance when the canteen is active. A single squared error output must otherwise represent both the closure regime and the broad positive count regime. A two stage model could first use an independently known opening schedule or a leakage free classifier and then forecast positive guest counts. Alternatively, a count distribution, zero inflated model, or weighted loss could be considered.

**Feature Leakage Masks the Need for Sequence Learning.** The same day `canteen_open` variable is derived from whether the target is positive. This gives all models a near perfect indication of the zero regime. Once that regime is revealed, the remaining task is more tabular. The tree models can exploit the binary split efficiently. A leakage free rerun could change absolute errors and potentially change the relative advantage of sequence models.

## 7.8 Critical Assessment of Feature Leakage

The most important methodological weakness is the construction

```
canteen_open = as.logical(n_guest > 0).
```

A predictor for day  $t$  is calculated from the target for day  $t$ . Unless opening status is available independently from a schedule, the model receives future information. This violates the normal forecasting information set.

The leakage does not automatically make every comparison meaningless. All six models receive the same variable, so the relative ranking still describes how they use the provided features. However, it makes the absolute errors optimistic and can favour models that split efficiently on the binary status. It also weakens the claim that the model forecasts whether the canteen will be open.

The proper correction is to replace the derived variable with an external calendar containing planned opening status. If such a calendar exists, the feature is valid because the information is known in advance. If it does not exist, the variable should be removed and the models retrained. The final thesis submission should state clearly which interpretation is true. A supervisor or reviewer is likely to identify this issue, and concealing it would be more damaging than reporting it transparently.

## 7.9 Seed Use and Stochastic Reliability

The use of seed 123 has a valid purpose: it makes pseudo random operations start from a documented state. It affects Random Forest bootstrap samples, noise draws, missing value masks, and neural initialisation. Without a seed, debugging and replication would be difficult.

The seed does not guarantee that all outputs are identical across scripts. A random number generator advances every time a random value is requested. In phase two, models are fitted repeatedly inside a loop. The 100% Random Forest therefore does not start from the same random state as the phase one Random Forest, even though both experiments begin from the same documented seed. This explains why the two reported errors differ.

A better pattern is:

1. clear the Keras backend session before each neural fit;
2. set R, NumPy, and TensorFlow seeds immediately before model construction;
3. use a different documented seed for each repeated run;
4. report the mean, standard deviation, and best/worst result across runs.

This would convert stochastic variation from an uncontrolled nuisance into a measured uncertainty. Until such repetitions are performed, small differences should not be described as definitive superiority.

## 7.10 Practical Model Selection Guidelines

The empirical results support the decision matrix in Table 7.1.

| Situation                                            | Preferred model                                       | starting | Reason                                                                          |
|------------------------------------------------------|-------------------------------------------------------|----------|---------------------------------------------------------------------------------|
| Need lowest typical absolute error                   | XGBoost                                               |          | Lowest baseline MAE and best later hold-out performance                         |
| Need protection against large errors                 | Random Forest                                         |          | Lowest baseline RMSE and short training time                                    |
| Need transparent statistical benchmark               | ARIMAX and Linear Regression                          |          | Establishes whether nonlinear or deep learning complexity is necessary          |
| Very small recent training window                    | XGBoost or ARIMAX                                     |          | Strongest phase two results using 20% of the training data                      |
| Training labels contain moderate noise               | Linear Regression or ARIMAX as robust references      |          | Minimal RMSE change in the observed noisy target experiment                     |
| Predictors contain missing values                    | Impute, then compare ARIMAX and XGBoost; repeat masks |          | ARIMAX was strongest in one run, but the result requires replication            |
| Large collection of related series becomes available | Reconsider LSTM, GRU, or DeepAR                       |          | Deep learning models may benefit from larger and more diverse sequence datasets |
| Strict compute or maintenance constraints            | Linear Regression, Random Forest, or XGBoost          |          | Sub second training in the measured experimental configuration                  |

**Table 7.1:** Practical model selection guidance derived from the experiment

The guidelines are starting points, not universal rules. Model selection should still use leakage free features, repeated temporal validation, and metrics linked to operational cost.

## 7.11 Implications for Stakeholders

For data scientists, the results emphasise the importance of strong simple baselines, chronological validation, repeated stochastic runs, and leakage checks before architecture complexity is increased. The seasonal naive forecast is particularly important because the seventh lag is strongly associated with the target. A complex model should be preferred only when it improves on this weekly repeat benchmark under the same information set. The unexpected ARIMAX improvement after mean imputation should be treated as a diagnostic signal rather than as evidence that missing data are beneficial; repeated missingness masks and inspection of the selected ARIMA orders are required.

For canteen operators, the model output can support staffing and food preparation planning,

but only if the opening schedule and input variables are available before the forecast is issued. Forecast errors should be translated into operational consequences such as excess preparation, food waste, understaffing, and service delays. The present results favour tree ensembles as practical candidates, but deployment should follow a leakage free rerun and a comparison restricted to genuinely open days. Closed days can otherwise dominate accuracy metrics and make a model appear more useful than it is for planning guest volumes on operating days.

## 7.12 Research Contribution Revisited

The contribution of the thesis is not a new forecasting architecture. Its value is the multi dimensional empirical framework and the resulting practical evidence.

The first contribution is breadth within one controlled application. Accuracy, data size, noisy labels, missing predictors, later period transfer, and efficiency are examined with the same six models. The second contribution is the inclusion of statistical references, which prevents the ML versus DL comparison from becoming a false two option choice. The third contribution is the evidence that a small structured business series can favour tree ensembles even when recurrent networks are designed for sequence learning. The fourth contribution is critical transparency: the study identifies where the code supports a conclusion and where the design is too weak.

This last point is scientifically important. A thesis is stronger when it states that one memory measure is invalid or one feature may leak target information than when it presents every output as equally trustworthy. The purpose of evaluation is not to make the chosen method look successful; it is to determine what the evidence actually supports.

## 7.13 Limitations

The principal limitations are summarised below.

1. Only one daily target series is analysed, limiting external validity.
2. The same day operational status variable is derived from the target unless an external schedule exists.
3. Model hyperparameters are largely fixed or default and are not tuned with a validation set.
4. Stochastic models are run once; seed related uncertainty is not quantified.
5. The data size design changes both sample size and temporal coverage.
6. Noise is added only to the training target, while lags remain based on the clean series.
7. Missingness is applied only to training predictors and is repaired with simple mean imputation.
8. The 70/30 split is a temporal stress test, not a formal concept drift adaptation experiment.
9. Elapsed time remains hardware dependent, and serialised model size does not equal peak process memory.
10. No prediction intervals or business cost analysis are included.

These limitations narrow the conclusions but also define a concrete future research agenda.

## 8 Conclusion and Future Work

### 8.1 Summary of the Study

This thesis examined whether more complex forecasting models provide a practical advantage for a relatively small, structured daily guest count series. Six fitted methods were evaluated: Linear Regression and ARIMAX as statistical references, Random Forest and XGBoost as tree based machine learning models, and LSTM and GRU as recurrent deep learning models. A seasonal naive forecast, 7 day and 28 day moving average forecasts, and a four week seasonal moving average forecast were additionally used as simple benchmarks. The fitted model comparison was organised into five phases covering baseline accuracy, training data size, noisy and incomplete training data, a later temporal holdout, and computational efficiency. Root mean squared error and mean absolute error were used to represent complementary views of forecast quality.

The central result is that model complexity did not translate into consistently superior forecasting performance. On the primary 80/20 chronological split, the seasonal naive benchmark achieved the lowest overall MAE at 136.13. Among the six fitted models, Random Forest achieved the lowest RMSE at 261.54 and XGBoost achieved the lowest MAE at 157.92. The ordinary moving average benchmarks were substantially weaker, while the seasonal moving average improved markedly by averaging comparable weekdays. This shows that the useful simple structure was specifically weekly recurrence rather than generic smoothing of recent attendance. On the later 30% holdout, XGBoost produced the lowest overall RMSE and the lowest fitted model MAE, while Seasonal Naive retained the lowest overall MAE. LSTM and GRU benefited from the complete training history, but neither recurrent model reached the accuracy of the strongest tree ensemble.

The all days and open days analysis further qualified the baseline ranking. After closed days were removed, ARIMAX achieved the lowest open day RMSE at 272.93 and XGBoost retained the lowest open day MAE at 187.09 among the six fitted models. This change confirms that the zero count closure regime affects both the target distribution and the apparent relative performance of the models.

The study therefore does not support the general assumption that deep learning should be selected merely because the data form a time series. For the analysed problem, the training sample contained only 666 observations in the primary split and the strong seventh lag already represented much of the weekly temporal structure. Under these conditions, the seasonal naive benchmark was difficult to beat on MAE, while tree ensembles used the engineered information effectively and at low computational cost.

## 8.2 Answers to the Research Questions

### 8.2.1 Effect of Training Data Size

The first research question asked how the amount of training data affects the three model families. The effect depended on the model. LSTM improved substantially as the training fraction increased from 20% to 100%, and GRU obtained its best result with the full training period. The deep learning models therefore showed the expected dependence on data quantity. However, the additional observations did not close the performance gap to Random Forest and XGBoost.

The statistical and tree based models did not show a simple monotonic relationship between sample size and error. XGBoost and ARIMAX performed strongly with the recent 20% window, while Random Forest achieved its best phase two RMSE at 50%. This pattern indicates that increasing the number of observations also adds older operating conditions. More data can reduce estimation uncertainty, but it can also reduce temporal relevance when the process changes. Consequently, data quantity and data recency should be treated as separate design choices.

### 8.2.2 Robustness to Noise and Missing Values

The second research question concerned stability under imperfect historical data. Linear Regression showed the smallest RMSE change after Gaussian noise was added to the training target. ARIMAX also remained comparatively stable. Random Forest, XGBoost, and GRU deteriorated more clearly in this condition. LSTM changed little, but its absolute error remained high; stability alone therefore did not imply usefulness.

In the missing predictor run, ARIMAX produced an unexpectedly low error after mean imputation. This was an observed result from one perturbation realisation rather than evidence that missingness improves forecasting. The missing masks were generated from one perturbation realisation, the affected predictors included strong lag variables, and the ARIMAX order could change after imputation. Repeated masks and stronger imputation baselines are required before a general robustness ranking can be stated. The defensible conclusion is that model responses to missingness were configuration dependent and that one imputation method should not be accepted without sensitivity analysis.

### 8.2.3 Generalisation to a Later Period

The third research question asked whether the recurrent networks generalise better when models are trained on an earlier period and tested on a later period. The results did not support this expectation. XGBoost achieved RMSE 248.67 and MAE 139.24 on the 70/30 split, followed closely by Random Forest in RMSE. GRU and LSTM remained less accurate than the statistical and tree based alternatives.

This conclusion is restricted to the implemented holdout design. The experiment changed the

split boundary but did not quantify drift, detect change points, or adapt the models online. It demonstrates later period performance, not a universal ability to handle concept drift. A formal drift study would require several rolling origins, statistical distribution tests, and repeated retraining policies.

### 8.2.4 Computational Requirements

The fourth research question addressed whether additional computational cost is justified. In the measured configuration, it was not. Random Forest and XGBoost trained in less than half a second and produced the strongest accuracy. LSTM and GRU required several seconds and had larger errors. Linear Regression was fastest, although its accuracy was lower than the tree ensembles.

The reported model file sizes are reproducible storage measures but cannot be interpreted as complete memory consumption. A process level peak memory benchmark would be required for direct RAM comparison. The Seasonal Naive, 7 day Moving Average, 28 day Moving Average, and Seasonal Moving Average methods require no parameter fitting and therefore have effectively zero training time. They also do not require a serialised fitted model object, although the relevant historical observations must be retained for prediction. Their computational cost is therefore lower than that of every fitted model, but their storage requirement is not directly comparable with model file size.

### 8.2.5 Practical Model Selection Guidance

The fifth research question asked which practical guidance can be derived. The seasonal naive forecast should be the first benchmark for this series because it achieved the lowest overall baseline MAE and directly reflects the strong weekly pattern. A fitted model should only be preferred for typical absolute error when it improves on this benchmark under the same evaluation design. Among the six fitted models, XGBoost is the strongest candidate when MAE is prioritised, while Random Forest is a strong all days candidate when large deviations represented by RMSE are prioritised. For open day RMSE, however, ARIMAX is the strongest fitted model. The moving average benchmarks should not be used as primary forecasting methods here because both the 7 day and 28 day versions produced large errors. The seasonal moving average was more competitive but did not displace Seasonal Naive on MAE or the strongest fitted models on RMSE. LSTM and GRU should be reconsidered when substantially more data, multiple related series, longer temporal contexts, or richer tuning resources become available.

The broader guideline is to select a model by evidence from the intended operating condition rather than by model reputation. A compact, leakage free tree model can be more appropriate than a neural network for small tabular time series problems. Conversely, the present results do not imply that recurrent models are generally inferior; they show that their advantages did not emerge in this particular data and evaluation design.

### 8.3 Scientific and Practical Contribution

The scientific contribution is a controlled multi condition comparison in which the same target, predictors, evaluation metrics, and model families are carried through five experimental phases, complemented by seasonal naive, ordinary moving average, and seasonal moving average benchmarks. The work links model accuracy with data quantity, data quality, later period performance, target regime composition, and computational effort instead of drawing conclusions from one clean data model comparison. The seasonal naive result demonstrates why a strong task specific benchmark is necessary: without it, XGBoost would incorrectly appear to have the lowest overall baseline MAE.

The practical contribution is a model selection framework for similar daily operational forecasting tasks. The results suggest beginning with a seasonal naive rule, ordinary moving average references, and a seasonal moving average reference, then using Linear Regression and ARI-MAX as fitted statistical baselines before evaluating Random Forest and XGBoost as nonlinear candidates. Deep learning models should be added when the data volume and sequence structure justify their tuning and maintenance cost. This staged procedure reduces the risk of choosing an unnecessarily complex model and makes the added value of each modelling step explicit.

A further contribution is the explicit separation between reproduced numerical results and claims that the implementation cannot support. The potential leakage of the same day operational status variable, the limitations of a single seed, the narrow interpretation of the noise and missingness tests, and the invalidity of R object size as total memory are documented rather than concealed. This critical treatment improves the reproducibility and credibility of the work.

### 8.4 Required Corrections Before Operational Use

The current experiment should not be moved directly into a production forecasting process. The operational status variable must first be verified. If the opening state is known in advance from an independent schedule, the schedule should be joined by date and its provenance documented. If it is not independently available, the feature must be removed and every phase rerun. This correction may increase all errors and can alter the relative model ranking.

Stochastic models should then be evaluated over several documented seeds. The random seed should be reset immediately before each Random Forest and neural network fit, and TensorFlow determinism settings should be recorded. Mean error, standard deviation, and confidence intervals should replace values from one perturbation realisation. A rolling origin evaluation should be added so that each model is tested at several forecast origins rather than one split boundary.

Finally, efficiency should be measured with repeated timings after an initial warm up and with process level peak memory. Hardware, operating system, R version, package versions, CPU or GPU use, and thread settings should be stated. These changes are necessary if the efficiency

results are intended to guide deployment rather than provide an exploratory comparison.

## 8.5 Future Work

Future research should first repeat the study with leakage free predictors and explicit calendar variables such as weekday, month, planned closure, and event indicators. Separate models for open/closed status and positive guest count could represent the zero heavy target more naturally. Prediction intervals or probabilistic forecasts would also provide decision makers with uncertainty rather than one point estimate.

The robustness experiments can be extended through multiple noise levels, block missingness, consecutive sensor outages, and missing values in the test period. Mean imputation should be compared with seasonal interpolation, Kalman smoothing, model based imputation, BRITS, and graph based approaches where the data structure supports them. A proper drift experiment should evaluate rolling windows, expanding windows, change point detection, and periodic or trigger based retraining.

The model set can be expanded after the evaluation protocol is strengthened. Additional candidates include regularised regression, support vector regression, LightGBM, temporal convolutional networks, and DeepAR for collections of related series. Their inclusion should be accompanied by comparable tuning budgets and validation rules so that architecture complexity is not confused with greater optimisation effort.

A final extension is to connect statistical error with operational consequences. Over forecasting and under forecasting may have different costs in staffing, purchasing, food waste, or service quality. A cost weighted loss function and a simulation of downstream decisions could reveal that the model with the lowest RMSE or MAE is not necessarily the model with the greatest business value.

## 8.6 Final Conclusion

For the available daily guest count series, the seasonal naive benchmark achieved the lowest overall MAE on the primary 80/20 split and on the later 70/30 holdout, demonstrating the strength of the recurring weekly attendance pattern. Among the six fitted models, XGBoost achieved the lowest baseline MAE and Random Forest the lowest baseline RMSE. The open day analysis qualified this result: XGBoost retained the lowest MAE among the fitted models, whereas ARIMAX achieved the lowest RMSE after closed days were excluded. The 7 day and 28 day moving average benchmarks performed poorly because simple smoothing did not preserve the weekly operating structure. The seasonal moving average performed substantially better than the ordinary moving averages but still did not exceed Seasonal Naive on MAE or the leading fitted models on RMSE.

The recurrent neural networks improved with more training data but did not outperform the

strongest statistical or tree based alternatives in the evaluated configurations. The results therefore reinforce a central forecasting principle: complex models must be compared with strong, problem specific simple benchmarks, and conclusions must be tied to the target distribution and intended operating condition. For this study, weekly seasonality, the treatment of closed days, the available sample size, and the evaluation metric were more decisive than the assumed hierarchy of model complexity.

## Bibliography

- [1] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*, 5th ed. Wiley, 2015.
- [2] Rob J. Hyndman and Yeasmin Khandakar. “Automatic Time Series Forecasting: The forecast Package for R.” *Journal of Statistical Software*, vol. 27, no. 3, pp. 1–22, 2008. DOI: 10.18637/jss.v027.i03.
- [3] Leo Breiman. “Random Forests.” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. DOI: 10.1023/A:1010933404324.
- [4] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System.” In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016. DOI: 10.1145/2939672.2939785.
- [5] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short Term Memory.” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.
- [6] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. “Learning Phrase Representations using RNN Encoder Decoder for Statistical Machine Translation.” In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014. DOI: 10.3115/v1/D14.1179.
- [7] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. “The M4 Competition: Results, Findings, Conclusion and Way Forward.” *International Journal of Forecasting*, vol. 34, no. 4, pp. 802–808, 2018. DOI: 10.1016/j.ijforecast.2018.06.001.
- [8] Shereen Elsayed, Daniela Thyssens, Ahmed Rashed, Hadi Samer Jomaa, and Lars Schmidt Thieme. “Do We Really Need Deep Learning Models for Time Series Forecasting?” *arXiv preprint arXiv:2101.02118*, 2021. DOI: 10.48550/arXiv.2101.02118.
- [9] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. “DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks.” *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020. DOI: 10.1016/j.ijforecast.2019.07.001.
- [10] Xiangjie Kong, Zhenghao Chen, Weiyao Liu, Kaili Ning, Lechao Zhang, Syauqie Muhammad Marier, Yichen Liu, Yuhao Chen, and Feng Xia. “Deep Learning for Time Series Forecasting: A Survey.” *International Journal of Machine Learning and Cybernetics*, vol. 16, pp. 5079–5112, 2025. DOI: 10.1007/s13042-025-02560-w.
- [11] Pedro Lara Benítez, Manuel Carranza García, and José C. Riquelme. “An Experimental Review on Deep Learning Architectures for Time Series Forecasting.” *International Journal*

- of Neural Systems*, vol. 31, no. 3, pp. 2130001, 2021. DOI: 10.1142/S0129065721300011.
- [12] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. “Deep Learning is Robust to Massive Label Noise.” *arXiv preprint arXiv:1705.10694*, 2017. DOI: 10.48550/arXiv.1705.10694.
- [13] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. “BRITS: Bidirectional Recurrent Imputation for Time Series.” In *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [14] Andrea Cini, Ivan Marisca, and Cesare Alippi. “Filling the Gaps: Multivariate Time Series Imputation by Graph Neural Networks.” In *International Conference on Learning Representations*, 2022.
- [15] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. “Understanding Deep Learning Requires Rethinking Generalization.” In *International Conference on Learning Representations*, 2017.
- [16] João Gama, Indrè Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. “A Survey on Concept Drift Adaptation.” *ACM Computing Surveys*, vol. 46, no. 4, pp. 44:1–44:37, 2014. DOI: 10.1145/2523813.
- [17] Michael Scholz and Tristan Wimmer. “A Comparison of Classification Methods Across Different Data Complexity Scenarios and Datasets.” *Expert Systems with Applications*, vol. 168, pp. 114217, 2021. DOI: 10.1016/j.eswa.2020.114217.
- [18] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*, 3rd ed. Melbourne, Australia: OTexts, 2021.
- [19] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” In *International Conference on Learning Representations*, 2015.
- [20] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. “Deep Learning Scaling is Predictable, Empirically.” *arXiv preprint arXiv:1712.00409*, 2017. DOI: 10.48550/arXiv.1712.00409.
- [21] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. “Scaling Laws for Neural Language Models.” *arXiv preprint arXiv:2001.08361*, 2020. DOI: 10.48550/arXiv.2001.08361.
- [22] Pedro Domingos. “A Few Useful Things to Know About Machine Learning.” *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012. DOI: 10.1145/2347736.2347755.
- [23] John C. B. Gamboa. “Deep Learning for Time Series Analysis.” *arXiv preprint arXiv:1701.01887*, 2017. DOI: 10.48550/arXiv.1701.01887.

- [24] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. “Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches.” In *Proceedings of the 13th ACM Conference on Recommender Systems*, pp. 101–109, 2019. DOI: 10.1145/3298689.3347058.
- [25] Hansika Hewamalage, Klaus Ackermann, and Christoph Bergmeir. “Forecast Evaluation for Data Scientists: Common Pitfalls and Best Practices.” *Data Mining and Knowledge Discovery*, vol. 37, pp. 788–832, 2023. DOI: 10.1007/s10618-022-00894-5.
- [26] Christoph Bergmeir, Rob J. Hyndman, and Bonsoo Koo. “A Note on the Validity of Cross Validation for Evaluating Autoregressive Time Series Prediction.” *Computational Statistics and Data Analysis*, vol. 120, pp. 70–83, 2018. DOI: 10.1016/j.csda.2017.11.003.
- [27] Deggendorf Institute of Technology. “Thesis Guideline.” Version 1.0, updated 27 June 2023. Deggendorf Institute of Technology, 2023. Available at: [https://www.th-deg.de/files/0/Studium/Abschlussarbeiten/leitfaden\\_abschlussarbeiten\\_en.pdf](https://www.th-deg.de/files/0/Studium/Abschlussarbeiten/leitfaden_abschlussarbeiten_en.pdf). Accessed: 18 July 2026.
- [28] University of South Bohemia, Faculty of Science. “Provision of the Vice Dean for Study P5: On the Submission of Bachelor’s, Master’s, and Advanced Master’s Theses at the Faculty of Science, USB.” Effective 1 November 2024. Available at: <https://www.prf.jcu.cz/en/faculty/documents/study-regulations-related-topics/p5>. Accessed: 18 July 2026.