Evaluating Statistical and Tree-Based Forecasting Methods in Python and ScalaTion

by

Zainab Agboola

(Under the Direction of John Miller)

Abstract

The COVID-19 pandemic has highlighted the critical need for accurate forecasting methods to predict mortality rates and inform public health interventions. This research evaluates the comparative performance of tree-based forecasting models, specifically Gradient Boosting, Random Forest, and Linear Model Trees, against traditional statistical forecasting approaches like ARMA and Random Walk models. The study is conducted using two distinct programming environments, Python and ScalaTion, to examine their influence on forecasting accuracy.

The analysis employs Symmetric Mean Absolute Percentage Error (sMAPE) as the primary metric to assess the forecasting models. Findings indicate that tree-based methods outperform traditional models in predictive accuracy across both platforms, demonstrating their robustness in handling complex data patterns. Moreover, the research confirms high consistency between Python and ScalaTion implementations, with minor variations attributed to platform-specific numerical and optimization differences.

Additionally, this thesis explores the impact of rolling validation combined with train-test splits on model performance, revealing that regression tree models maintain superior accuracy across multiple forecasting horizons. This study contributes to the literature by providing a comprehensive evaluation of traditional and tree-based forecasting methods in varying programming environments, offering insights into their suitability for pandemic-related forecasting applications.

Index words: Forecasting, Tree-Based Models, Statistical Methods, Python, ScalaTion, COVID-19

Evaluating Statistical and Tree-Based Forecasting Methods in Python and ScalaTion

by

Zainab Agboola

B.Sc., Lagos State University, Nigeria, 2013

A Thesis Submitted to the Graduate Faculty of the
University of Georgia in Partial Fulfillment of the Requirements for the Degree.

Master of Science

Athens, Georgia

2024

Evaluating Statistical and Tree-Based Forecasting Methods in Python and ScalaTion

by

Zainab Agboola

Acknowledgments

I am deeply grateful to God for making this a success. I extend my heartfelt thanks to my advisor, Dr. John Miller, and to the committee members Dr. Ismailcem B. Arpinar and Dr. Frederick Maier, whose guidance, expertise, and unwavering patience have been instrumental in shaping this research. Your mentorship has been invaluable, and I am privileged to have had the opportunity to learn from you.

Special thanks go to my family for their love, understanding, and the countless moments they sacrificed while I pursued this milestone. To my parents and younger brother, thank you for always believing in me and providing the encouragement I needed to persevere. I am also indebted to my colleagues and peers, especially Adetayo Okunoye, whose thought-provoking discussions and collaboration have enriched my understanding and broadened my perspectives.

Lastly, I want to acknowledge the University of Georgia for providing me with the resources and environment to undertake this research. This journey has been both challenging and rewarding, and I am immensely thankful for everyone who contributed to its completion.

Contents

Acknowledgments **iv**

List of Figures **vi**

List of Tables **vii**

1 Introduction **1**

1.1 Motivation for Study 2

1.2 Research Objective 2

1.3 Additional Issues 3

2 Literature Review **4**

2.1 Related Study: COVID-19 Forecasting 4

2.2 Related Study: Traditional and Tree-Based Forecasting Methods 5

2.3 Related Study: Tree-Based Forecasting and Deep Learning Forecasting Methods 5

3 Python and ScalaTion Software Environments for Time Series and Regression Trees **7**

3.1 Time Series Analysis and Regression Trees in Python 7

3.2 Python Libraries for Tree-Based Forecasting 7

3.3 Implementation of Tree-Based Forecasting in ScalaTion 8

4 Time Series Predictive Analytics in ScalaTion – Traditional Statistical Models and Regression Trees Models **10**

4.1 Predictive Modeling in ScalaTion 10

4.2 Analytical Databases in ScalaTion 11

4.3 Removing Identifiers 12

5 Methodology and Data Set **14**

5.1 Baseline Models 14

5.2 Multi-Horizon Forecasting 14

5.3 Rolling Validation, Train-and-Test Method and In-Sample Training 15

5.4 Direct and Recursive Methods of Multi-Horizon Forecasting 15

5.5 Traditional Forecasting Models 16
5.6 Tree-Based Forecasting Models 19
5.7 How Regression Trees are Adapted to Time Series 21
5.8 COVID-19 Data Set 21
5.9 Cross-Correlation Analysis 23

6 Performance Evaluation and Discussions **25**
6.1 Symmetric Mean Absolute Percentage Error (sMAPE) Metric 25
6.2 Test Setup 26
6.3 Forecasting Performance: Traditional Forecasting Models 27
6.4 Rolling Validation: Traditional Forecasting Models 27
6.5 Forecasting Performance of Regression Tree Models 29
6.6 Rolling Validation: Tree-Based Forecasting Models 30
6.7 Conclusion 31
6.8 Future Studies 31
6.9 Bibliography 33

List of Figures

1 **Regression Tree Diagram** 19

2 **Plot of Daily and Weekly Deaths from COVID-19** 22

3 ACF Plots of the Daily and Weekly Deaths 22

4 Plot of Weekly Deaths from COVID-19 26

List of Tables

1 **Types of Time Series Forecasting Models** 18
2 Variable Descriptions 24

Table 4: sMAPEs TnT with Rolling Validation, skip=0 29

Chapter 1

Introduction

Accurate forecasting is crucial in fields like finance, economics, and biomedical sciences. Due to extensive data collection during the COVID-19 pandemic, there is a unique opportunity to study pandemics forecasting. Forecasting the number of deaths and infections of COVID-19 is crucial for policymakers and healthcare systems to allocate resources, control outbreaks, and implement preventive measures to protect public health. Thus, utilizing predictive modeling is important for accurate forecasting. This thesis extends the current literature on COVID-19 pandemics forecasting by comparing the performance of tree-based forecasting methods against the traditional forecasting approaches within two distinct programming environments.

Traditional statistical forecasting methods like Random Walk, Autoregressive Integrated Moving Average (ARIMA) model, Auto-regressive (AR) model, Moving Average (MA) model, Exponential Smoothing, etc. often fail to model complex, non-linear data patterns effectively (Rady, Fawzy, and Abdel Fattah, 2021). To overcome the shortcomings of these forecasting methods, the field of data science has advanced machine learning-based approaches. These include tree-based and deep learning-based forecasting techniques, which are designed to effectively enhance prediction accuracy. While traditional and deep learning-based models are prevalent in forecasting literature, tree-based methods like Linear Model Trees (LMT), Random Forests (RF), and Gradient Boosting (GB) have proven superior in competitions such as the M5, GEFCom, and various Kaggle forecasting competitions (Januschowski et al., 2022; Bojer and Melgaard, 2020). Therefore, tree-based methods represent a promising avenue in forecasting techniques.

In data science, choosing the right software environments and libraries is as crucial as selecting the best forecasting methods because these decisions impact the efficiency, effectiveness, and viability of projects, especially in fields that demand accurate predictions, such as epidemiological forecasting. Python and ScalaTion stand out as two prominent programming environments for forecasting. Consequently, forecasting accuracy hinges on the synergy between choosing the correct software environment and effective forecasting techniques.

1.1 Motivation for Study

Making useful predictions is often clouded in uncertainty. So, accurate forecasting is an indispensable tool across various disciplines. Predictive models, both traditional and tree-based, tend to be highly parameterized and are inherently data-dependent. Thus, these models have the potential to perform very well for short and near-term forecasting.

Since the COVID-19 pandemic, government agencies and researchers in academia have forecasted COVID-19 deaths, infections, etc. using one or a combination of traditional and deep-learning-based forecasting techniques (see Crammer et al., 2020; Barmparis and Tsironis, 2020; Picchiotti et al., 2020; Nishimoto and Inoue, 2020; Shamil, et al., 2021; Fazeli et al., 2020; Javeri et al., 2021). These studies compare traditional and deep learning techniques using metrics like Mean Absolute Error (MAE) and symmetric Mean Absolute Percentage Error (sMAPE).

As statistical software, including black-boxes and object-oriented systems that utilize time series forecasting, continue to grow across various disciplines, there has been a corresponding rise in the development of time series forecasting libraries for these statistical platforms. Prominent examples are Statsmodels and Sktime in Python, Time Series Analytical Packages in R (see R project), Spark-Ts and Flint in Spark, etc. However, it is important to note that several packages are not part of the core platform, rather they are often built and published in these platforms by third-party additions. In contrast, ScalaTion utilizes its underlying libraries, analytical packages, and forecasting models that are natively incorporated within the framework.

Considering the differences in programming environments between Python and ScalaTion, as well as the limitations of traditional forecasting methods relative to tree-based models, an evaluation of both traditional and tree-based methods in Python and ScalaTion can help determine their suitability for forecasting models. Furthermore, tree-based methods benefit from features like hyperparameter tuning, feature engineering, and ensembling, but they require robust software to effectively balance efficiency and accuracy. Januschowski et al. (2022) emphasize the need for powerful software to minimize loss functions. Evaluating tools like Python and ScalaTion can help determine their suitability for tree-based forecasting in machine learning projects. Ultimately, while tree-based methods are increasingly favored for time-series analysis, their success depends on the strength of the underlying statistical software.

1.2 Research Objective

The main goal of this research is to assess the efficiency and practicality of tree-based models for predicting COVID-19 mortality rates. This involves evaluating predictive models against baseline models—Random Walk (RW), Random Walk with slope adjustment (RWS), Auto Regressive (AR), and the Auto Regressive Moving Average(ARMA)—as identified by Miller (2024). These baselines serve as benchmarks due to their simplicity. Because time series data are time-ordered and serially correlated, the quality of forecasts is likely to degrade as the forecasting horizon increases.

Although the application of rolling validation combined with train and test split (TnT) as opposed to in-sample testing (In-ST) to time series models addresses some of these anomalies, the extent to which these corrections are effective for regression tree models remains unclear in the literature. Specifically, the objectives of this research include:

1. **Comparative Analysis:** How do regression trees and traditional statistical models perform relative to the baseline benchmarks in terms of forecasting accuracy. Examining if the programming environment influences the result of these forecasting performances.

2. **Rolling Validation:** Does the use of rolling validation with train and test splits, as opposed to in-sample testing, affect the relative performance of regression trees compared to traditional statistical models?

3. **Overall Performance Evaluation:** Conduct a detailed overall performance evaluation of regression trees and traditional statistical models developed in Python and ScalaTion. Which programming environment, offers the superior performance for forecasting COVID-19 mortality rates?

1.3 Additional Issues

The rest of the thesis is organized as follows: Chapter 2 contains the literature review pertaining to relevant topics, and Chapter 3 examines software environments and libraries that were used and presents some background information about the different approaches. Chapter 4 deals with time series analytics in ScalaTion. Chapter 5 focuses on the methodology and dataset of this thesis. Finally, in Chapter 6, the conclusion is presented, as well as what the future works could entail.

This thesis evaluates the performance of tree-based forecasting models (Gradient Boosting, Random Forest, and Linear Model Trees) compared to traditional statistical models (RW, ARMA, and AR) for predicting COVID-19 mortality rates. The results demonstrate that tree-based models consistently outperform traditional statistical models in terms of forecasting accuracy. For instance, in Python, Gradient Boosting achieved an average sMAPE of 18.423% under in-sample validation, compared to 39.814% for the AR(1) model. Similarly, in ScalaTion, Gradient Boosting yielded an average sMAPE of 48.240%, while AR(1) achieved 39.573%. These discrepancies highlight the challenges in optimizing tree-based models within ScalaTion.

The results also reveal that regression tree models maintain superior accuracy across multiple forecasting horizons when evaluated under rolling validation. For example, in Python, Gradient Boosting achieved an average sMAPE of 25.567% across six horizons under rolling validation, compared to 48.043% for the ARMA(1,0) model. Similarly, Random Forest achieved an average sMAPE of 23.404%, further underscoring the robustness of tree-based models.

Although Python consistently exhibits better alignment with expectations, ScalaTion demonstrates comparable performance in traditional statistical models like AR and ARMA, where sMAPE deviations between platforms are within 1.5%. These findings emphasize the need for further debugging and optimization of tree-based model implementations in ScalaTion to ensure parity with Python.

Chapter 2

Literature Review

In the last few years, academic research groups, government bodies, industrial teams, and individuals have produced forecasts related to COVID-19. As mortality and morbidity in the U.S. rose between 2020 and 2021, a great amount of research was conducted to facilitate measures for predicting the COVID-19 pandemic course. High mortality rate from COVID-19 pandemic made it clear to public health policymakers and academic research groups that effective measures to counter future pandemics are needed.

2.1 Related Study: COVID-19 Forecasting

Studies have been conducted since the beginning of the COVID-19 pandemic. Most of these studies dwell on forecasting the number of people affected across hospitals and intensive care units (ICU). Barmparis and Tsironis (2020) use the Susceptible, Infected and Recovered (SIR) model to forecast COVID-19 infection rate in eight countries and find that the infection in China follows a Gaussian distribution, while the United Kingdom and the U.S. have discrepancies in their peak dynamics. Fazeli, Moatamed, and Sarrafzadeh (2020) use statistical methodologies and machine learning to model the underlying patterns of COVID-19 occurrences and find that these methods exhibit clear training convergence and efficient prediction results. Javeri et al. (2021) utilized Automated Machine Learning techniques such as Neural Architecture, a deep learning approach, and find that this method improves forecasting accuracy. Picchiotti et al. (2020) used the Susceptible, Exposed, Infectious, and Recovered (SEIR) model to adjust for mobility restrictions due to regional lockdowns and find that the SEIR model improves the forecasting power of COVID-19 prediction. Nishimoto and Inoue (2020) used a simple curve-fitting model that can be implemented in Microsoft Excel using a log-normal function. Shamil et al. (2021) investigate how agent-based models are used in COVID-19 infection rate prediction.

2.2 Related Study: Traditional and Tree-Based Forecasting Methods

Traditional forecasting methods like Random Walk and ARIMA models are popular for forecasting stochastic time series data. Khan (2013) uses the ARIMA model to forecast the price of gold between 2003 and 2012 and shows that the ARIMA (0, 1, 1) model is the best model for this purpose. While traditional forecasting methods have long been favored in economics and stock market studies, their application in epidemiological research has gained prominence following the COVID-19 outbreak (See Yang et al., 2020; Ismail et al., 2020). However, these traditional approaches typically presume that the time series data are linear and adhere to a specific distribution, such as the normal distribution, as noted by Taib (2014) and Rady, Fawzy, and Abdel Fattah (2021).

Krauss, Do, and Huck (2017) examine stock market movements using logistic regression. The authors compared gradient boosting and random forest to traditional regression methods, finding that tree-based approaches outperformed regression due to their superior ability to capture non-linear dependencies in the S&P 500 data. In predicting climate extremes, Ham and Kug (2015) also show that tree-based methods, especially gradient boosting, are superior to traditional regression models.

2.3 Related Study: Tree-Based Forecasting and Deep Learning Forecasting Methods

> "The prominence of Gradient Boosted Decision Trees (GBDTs) on Kaggle is the most glaring difference between what is used on Kaggle and what is fashionable in academia" – Anthony Goldbloom, the CEO of Kaggle.

This section of the literature review explains why tree-based models frequently surpass traditional and deep learning forecasting methods by examining both theoretical frameworks and empirical studies. It highlights the benefits of tree-based approaches and reviews research that supports these advantages across different applications, including epidemiological forecasting.

Forecasting literature extensively explores the search for efficient time series forecasting techniques, motivated by the dynamic nature of the problem and the desire for enhanced results. Navin (2016) created a forecasting model to predict gold prices using decision trees and support vector regression (SVR), utilizing historical gold price data. The results demonstrated that the decision tree was faster in processing data and yielded a lower mean square error than the SVR.

Recent discussions in machine learning literature often highlight the prevalence of deep learning-based forecasting models (refer to Zhou et al., 2021; Rasul, Steward, Schuster, and Vollgraf, 2021; Lim, Arik, Loeff, and Pfister, 2019). Januschowski et al. (2022) credit the widespread adoption of deep learning-based forecasting methods to the availability of open-source, specialized packages such as GluonTS (Alexandrov et al., 2020), PytorchTS (Rasul, 2021), and PyTorch Forecasting that implement these techniques.

However, in M5, Global Energy Forecasting Competitions (GEFCom), and various Kaggle competitions, tree-based methods like Decision Trees (DT), Random Forests (RF), and Gradient Boosted Decision Trees (GBDT) have shown superior performance over deep learning models (Bojer and Melgaard, 2020; Januschowski et al., 2022). Notably, Januschowski et al. (2022) observed that tree-based models excel over deep learning in handling data that is not related to images, text, or videos. Tree-based forecasting methods consistently rank highly, securing top positions in these competitions and achieving top 10 in many others, as outlined by Bojer and Meldgaard (2020). Deep learning forecasting methods like Neural Networks, using methods like DeepAR and Neural Basis Expansion Analysis for Time Series (NBEATS), secured second and third places in the M5 accuracy competition.

Chapter 3

Python and ScalaTion Software Environments for Time Series and Regression Trees

This chapter examines the tools of predictive statistical analysis in Python and ScalaTion focusing on tree-based forecasting. Python and ScalaTion stand out as two prominent programming environments for complex statistical and machine learning analysis. Python is well known for its robust forecasting data science libraries: Pandas, Numpy, scikit-learn, Statsmodels, Tensor Flow, Keras, GluonTs, PytorchTS, etc. Although less widely used, ScalaTion excels in simulation and mathematical modeling by leveraging on its functional programming and efficient large-scale data processing capabilities.

Python is highly regarded for its intuitive usability and extensive suite of data science libraries, including Pandas, scikit-learn, and Matplotlib, rendering it highly suitable for a broad spectrum of data analysis and predictive modeling endeavors. Conversely, ScalaTion, though less frequently utilized, excels in areas of simulation and mathematical modeling, capitalizing on ScalaTion's functional programming attributes like Java Virtual Machine (JVM)/Java Development Kit (JDK) interoperability and its proficiency in managing large-scale data processing with remarkable efficiency.

3.1 Time Series Analysis and Regression Trees in Python

An open-source programming package, Python is known for its simple and readable syntax, flexibility, and robust community and support. Python supports time series analysis and regression trees in the following libraries:

3.2 Python Libraries for Tree-Based Forecasting

1. **Scikit-learn (Sklearn)**
   One of the most popular libraries in Python, Sklearn is mostly utilized in predictive analytics, data

mining, classification regression, etc. Sklearn also supports limited time series algorithms like shaping time series data into future matrices. Specifically, Sklearn can be utilized for random forest regression and random forecast analysis in the areas of parameter tuning, number of trees, and max depth. Specifically, Sklearn library can be found at https://scikit-learn.org/stable.

2. **Sktime**
   Built on Sklearn, Sktime is a time series version of Sklearn that facilitates and transforms time series data into supervised learning problems. Sktime also performs time series classifications like classifications, regression trees, etc. The Sktime library that pertains to the time series analysis can be found at https://www.sktime.net/en/stable.

3. **Statsmodels**
   Statsmodels is the main Python library for modeling traditional time series techniques like Auto-Regressive Integrated Moving Average Model (ARIMA), Random Walk, etc. This library integrates with Pandas for handling time series forecasting. This library can be found at https://www.statsmodels.org/stable/index.html.

4. **PyTorch-Forecasting and PyTorchTS**
   As the name implies, the PyTorch Forecasting library handles time series forecasting with deep learning models. These libraries can implement advanced models like Temporal Fusion Transformers, which can handle multivariate time series with multi-horizons. PyTorch can also handle Neural Networks models like Recurrent Neural Networks (RNNs). This library can be found at https://github.com/jdb78/pytorch-forecasting?tab=readme-ov-file.

5. **XGBoost**
   The XGBoost library is utilized for classification, but it can also be used for gradient boosting. This library supports GPU acceleration, thus speeding up computations. It also combats overfitting by utilizing advanced regularization features. This library can be found at https://xgboost.readthedocs.io/en/latest/.

6. **LightGBM**
   The LightGBM library supports high-performance gradient boosting frameworks like Gradient Boosting Decision Trees (GBDT) and Gradient Boosting Regression Trees (GBRT). It also supports fast training and reducing memory usage during the implementation of classification and regression trees algorithms. This library can be found at https://lightgbm.readthedocs.io/en/latest/.

3.3 Implementation of Tree-Based Forecasting in ScalaTion

ScalaTion programming language is an offshoot of Scala, another programming language developed by Martin Odersky in 1993 (Odersky, Spoon, and Venners, 2016). Scala simplifies simulation modeling by reducing the complexities of using Java. ScalaTion programming language is developed as an academic research project by John A. Miller of the University of Georgia. ScalaTion functions as an embedded Domain-Specific Language (DSL) designed for modeling and simulation (M&S) and provides support for multi-paradigm modeling, enabling its use in simulation, optimization, and analytics (Cotterell, Miller, and Horton, 2011). According to Cotterell, Miller, and Horton (2011), the earlier version of ScalaTion was restricted to random variation generation, output, and comparative analysis. Improving on the earlier version of ScalaTion, Professor John A. Miller of the University of Georgia has enhanced ScalaTion's programming capability to cover tools for analytical packages, thereby addressing the needs of large datasets that require parallel and/or distributed computing for statistical analytics and machine learning.

The key attributes of ScalaTion include a functional object model, Java Virtual Machine (JVM) Compatibility, extensive libraries and tools, extensive enhancing features (such as comprehension, tuples, generic arrays, etc.), and performance optimization. The latest release of ScalaTion, ScalaTion 2.0, has ten top-level packages:

- **Animation:** The animation package supports the animation of models.
- **Calculus:** The calculus package supports numerical differentiation and integration.
- **Database:** The database package supports relational and graph databases.
- **Dynamics:** The dynamics package supports the development of ODE models.
- **Mathstat:** The mathstat package supports basic math and statistics.
- **Modeling:** The modeling package supports the development of several types of data science models.
- **Optimization:** The optimization package supports linear and nonlinear optimization.
- **Random:** The random package supports random variate generation.
- **Scala2d:** The scala2d package supports simple 2D graphics in Scala, based upon `java.swing`, `java.awt`, and `java_awt_geom`.
- **Simulation:** The simulation package supports the development of simulation models.

Chapter 4

Time Series Predictive Analytics in ScalaTion – Traditional Statistical Models and Regression Trees Models

This chapter covers data management and data analytics in ScalaTion. According to Miller (2020), ScalaTion facilitates multi-paradigm modeling that can be used for simulation, optimization, and analytics. To ensure that high-quality data are used in forecasting, data preprocessing should be implemented prior to applying analytics techniques. ScalaTion provides a variety of preprocessing time series and data structures. ScalaTion makes provision for tools that can handle, store, scale, preprocess, and perform analytics on large datasets beyond those found in traditional On-Line Analytic Preprocessing (OLAP). These analytical components in ScalaTion consist of six tools: predictors, classifiers, forecasters, clusterers, recommenders, and reducers.

4.1 Predictive Modeling in ScalaTion

Like other statistical software, modeling types in ScalaTion depend on the nature of the response variable. Forecasting models such as ARIMA, Regression Trees, and Gradient Boosting models are used if the goal is to forecast continuous response variables into the future. Beyond the prediction function of simple regression models, time series are set up to use prior values of the response variables or past errors (or innovations) to improve forecasts.

Traditional Statistical Models

Besides the response variable or the shocks, other variables (also called exogenous variables) may be used to improve forecasting. Thus, a time-dependent model can be formulated as:

$$y = f(x, t; b) + \epsilon$$

where:

- y is a response variable,
- x is an input vector,
- ϵ is the residual term,
- t is time, and
- b is the vector of parameters of the function that can be scaled so that the predictive model matches available data.

This model can be generalized by turning y into a vector, the parameters into a matrix, and allowing feedback into the function f.

Estimating the parameters for traditional statistical techniques like AR, the model predicts the next value y_t from the last p values, each weighted by its own coefficient, ϕ_j. Here, the residual term is represented by ϵ_t, in:

$$y_t = \mu + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \epsilon_t$$

Estimating the parameters involves forming Yule-Walker equations and using one of the Method of Moments (MOM) or Least Squares Estimation (LSE). After these coefficients are estimated, the AR(p) model can be used for forecasting.

The 'Forecaster' trait within the 'scalation.modeling.forecasting' package in ScalaTion provides a common framework for several forecasters for estimating the parameters. The 'Stats4TS' class in the 'scalation.mathstat' package defines the Auto-Correlation Function (ACF). It holds the mean, variance, the auto-covariance vector, and the auto-correlation vector (Miller, 2024). In addition to the 'Forecaster' trait, several Quality of Fit (QoF) measures are available, including R-squared, R-Squared adjusted, MAE, MSE, RMSE, and AIC in the 'Fit' trait package.

4.2 Analytical Databases in ScalaTion

Prior to the application of time series analytics in ScalaTion, data preprocessing should be applied to the data to ensure that the data is of high quality. ScalaTion database has Python-like dataframes and SQL-like APIs. In ScalaTion, the data management framework is provided by ScalaTion's Time Series Database (TSDB). ScalaTion's TSDB has built-in capabilities for handling time series data. It is also able to store non-time series data as well. It provides two Application Programming Interfaces (APIs) for convenient access to the data:

1. Columnar Relational Algebra API

2. SQL-Like API

In ScalaTion's database module, the ScalaTion TSDB is located within the `timeseries_db` package, a subpackage of the `columnar_db` package. Also, several adjustments to Coordinated Universal Time (UTC) can be made on TSDB. For example, the `TimeNum` class, a class wrapped around Java's `Instant`, facilitates time-zone information storage and adjustment time, including micro- and nanoseconds and Daylight Savings Time.

4.3 Removing Identifiers

Since arbitrary values serve no purpose in data analytics beyond their use as unique identifiers, they should be removed once they have fulfilled this function. For example, ScalaTion may utilize a composite primary key, 'Seq(0, 1)', which combines both time and sensorID for unique identification. These unique identifications must be removed during data preprocessing.

Object Coverter

Convert a VectorS into a VectorI by mapping each distinct value in VectorS into a distinct numeric integer value, returning the new vector and the bidirectional mapping. Use the 'from' method in BiMap to recover the original string. e.g., VectorS ("A", "B", "C", "A", "D") will be mapped to VectorI (0, 1, 2, 0, 3)

Identifying and Handling Missing Values

Missing values are common in datasets. Special characters such as dots ('.'), question marks ('¿), large integers ('99999'), etc., are often used to indicate missing values in datasets. In data management, zeros or negative numbers may also indicate missing values when such values are invalid or unsuitable for the application. In ScalaTion, a 'MissingVal' parameter in the 'replaceMissingValues' API can be set to zero or any value to be considered as missing and will need to be imputed. Missing values are sometimes handled by imputation.

Imputation implementations are based on the 'Imputation' trait in the 'scalation.modeling' package (See Miller, 2020). ScalaTion offers flexible APIs for imputing missing values such as Linear Interpolation, Moving Average, etc., within the class 'Imputation.replaceMissingValues'. Another alternative could be to use a modeling technique like SARIMA for imputation.

Detecting Outliers

Strategies for managing outliers include addressing them separately or re-coding them as missing values, thus enabling a unified approach to handling both outliers and missing data. To manage outliers, ScalaTion has several methods, including:

- ‘DistanceOutlier‘: Identifies outliers as data points that are a specified number of standard deviation units from the mean.
- ‘QuantileOutlier‘: Uses the smallest and largest percent values.
- ‘QuartileXOutlier‘: Applies an expansion multiplier beyond the middle two quartiles.

An example in Miller (2020) for reassigning an outlier to ‘noDouble‘ in ScalaTion’s indicator of a missing value of type ‘Double‘ is:

```
DistanceOutlier.rmOutlier(traffic.column("speed"))
```

Chapter 5

Methodology and Data Set

This chapter outlines the methodology employed in this thesis and provides information about the dataset used. In this chapter, benchmark forecasting models for evaluating the performance of all other models are identified. Because forecasting is more challenging than one-step ahead predictions (that is $h > 1$), a benchmark is established for the forecasting horizon. This chapter also examines, In-Sample Testing (In-ST), and Train-and-Test (TnT) with Rolling Vallidation.

5.1 Baseline Models

Traditional statistical forecasting methods and decision tree-based machine learning techniques function as complementary models. Studies like Miller et al. (2024) adopt some of the traditional statistical forecasting techniques like Random Walk (RW), Random Walk with slope adjustment (RWS), and Null Model as reliable baseline models for evaluating the performance of more quality models, including those based on deep learning models. Like Miller et al. (2024), this thesis uses three models as baselines to assess the forecasting performance of tree-based models in Python and ScalaTion programming environments.

5.2 Multi-Horizon Forecasting

The quality of forecast is likely to degrade as the forecasting horizon increases because time series data are time ordered and serially correlated. To correct this anomaly, a multi-horizon forecasting technique is employed across the time series models (see Miller, 2024; Hyndman and Athanasopoulos, 2018). The horizon refers to the number of time units (e.g., days or weeks) into the future for which forecast values are to be generated. While there is no established hard and fast rule on the length of the forecasting horizon, Hyndman and Athanasopoulos (2018) recommend that the number of lags h be 10 for non-seasonal processes, that is $1 \leq h \leq 10$. Miller (2024) recommends the length of the forecasting horizon to be 6 for COVID-19, that is $1 \leq h \leq 6$. Unless stated otherwise, Miller (2024) approach will be used in this thesis.

5.3 Rolling Validation, Train-and-Test Method and In-Sample Training

With respect to traditional statistical techniques and regression trees forecasting analytics, the concept of rolling validation is closely related to multi-horizon forecasting. Cross-validation methods are not suitable for time-series data due to the existence of serial dependence. A rolling validation method is used to correct this anomaly.

Rolling validation is employed with Train-and-Test (TnT) analytic method. Using the $1 \leq h \leq 6$ horizon, the rolling window is moved forward by one step. This step appends the first value from the test set to the training set and removes the first value from the training set. Then, rolling windows move to the end of the test set, and 6-steps forecasts are obtained on each step.

Alternatively, in-sample training (In-ST) can be employed without rolling validation. Here, the In-ST training technique does not require rolling validation. Instead, In-ST is assessed and computed over the full dataset. However, for TnT with rolling validation, each retraining may produce a slightly different values for the mean and parameters.

5.4 Direct and Recursive Methods of Multi-Horizon Forecasting

Still on Multi-Horizon Forecasting, Hyndman et al. (2012) identify three main techniques available for multi-horizon forecasting:

Recursive Method of Multi-Horizon Forecasting

The recursive forecasting method generates forecasts sequentially, storing them in a matrix referred to as y_f, starting with actual data and progressively incorporating earlier forecasts (Miller, 2024). Over time, depending on the model's reliance on past values, these forecasts transition to being based entirely on previously forecasted data.

Direct Method of Multi-Horizon Forecasting

The direct method applies a trained forecasting function f to recent data to obtain an h-step ahead forecast. In this case, the training minimizes the h-step ahead forecast error. For example, in a single output model like ARIMA, a separate model could be built for each forecasting horizon 1 through h. Thus, in the Direct Method of rolling validation for time series, each test set is composed of a single future observation, and the training set includes only prior data.

Hybrid Method of Forecasting

This method which combines both recursive and direct methods ensuring both short-term precision and computational tractability for extended horizons enhancing overall forecast accuracy.

5.5 Traditional Forecasting Models

To predict the future values of a response variable, say y, it is important to identify variables that could influence y. The most obvious predictor is the previous (or lagged) values of y itself. Traditional forecasting models are mostly based on this simple but fundamental concept.

Random Walk (RW) Model

A time series is said to follow an RW model if the first differences are random. That is, the series is non-stationary. The RW model is one of the simplest time series models, and it is applicable to time series data when the difference have Gaussian distribution and are independent of each other. An RW model for a variable y_t can be expressed as:

$$y_t = y_{t-1} + \epsilon_t$$

where ϵ_t follows a Gaussian distribution.

Random Walk with Slope Adjustment (RWS) Model

RWS is an extension of the baseline RW model. It adds a 'slope' to the RW model to account for trends in the data. This model assumes that future values will continue in the direction of the past trend, adjusted by a slope derived from historical data. Like the RW model, the white noise ϵ_t in the model also follows a Gaussian distribution. The general equation for the RWS model is:

$$y_t = \gamma + y_{t-1} + \epsilon_t$$

where the term γ is the slope.

Null Model

Also known as the mean model, the null model is a basic baseline model that predicts future values as the historical mean. Although the null model is generally less accurate than the RW model, it is a good alternative to the RW model if the average is stable over time.

$$y_t = \omega_y + \epsilon_t$$

where μ_y is the mean of the response variable.

Trend Model

Unlike the null model, which assumes that the average is stable over time, the trend model adjusts for trend in the data by adding a slope parameter to the null model. That is, the null model is just a generalization of trend and similar models. Following Miller (2024), the trend model is also considered a basic baseline model in this study. The trend model is denoted by:

$$y_t = \alpha_0 + \alpha_1 t + \epsilon_t$$

where α_0 and α_1 are the intercept and slope of the trend model, respectively.

Simple Exponential Smoothing (SES) Model

In simple terms, the SES model is a forecasting technique that adjusts the value of past observation values to compute the next predicted value by exponentially decreasing the weights given to the older values. The SES model is most useful when there is no trend or seasonal patterns in the data. This model is denoted by:

$$\hat{y}_t = \alpha \hat{y}_{t-1} + (1 - \alpha)\hat{y}_{t-1}$$

where μ is the weight. The SES Model updates the forecast by applying the smoothing constant parameter α to the most recent observation and $(1 - \alpha)$ to the past smoothed values.

Moving Average (MA) Model

The Moving Average (MA) model forecasts future values of a time series by smoothing short-term fluctuations (or shocks) and minimizing noise, thereby revealing the underlying trend in the data. The model equation for an $MA(q)$ includes the past q values of ϵ_t as shown below:

$$y_t = \mu + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t$$

Auto-Regressive (AR) Model

The AR model is ideally suited when the future values of a variable linearly depend on its lagged values. A p^{th}-order Auto-Regressive $AR(p)$ model predicts the next value y_t from the sum of the last p values each weighted by its own coefficient/parameter, say ϕ_j. Yule-Walker equations are recursively applied to solve for the coefficients/parameters of autoregressive (AR) models (see Miller, 2024). This model is denoted by:

$$y_t = \alpha + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t$$

Auto-Regressive Moving Average (ARMA) Model

An $ARMA(p, q)$ time series model of order p and q integrates $AR(p)$ and $MA(q)$ models. This model is often used for describing stationary time series. It combines the principles of autoregression and moving average: the autoregressive component $AR(p)$ represents the current value of the series as a linear function of its previous values, and the moving average component $MA(q)$ depicts it as a linear function of previous error terms. By combining both $AR(p)$ and $MA(q)$ models, the $ARMA$ model captures the dynamics and noise within the series. The model equation can be formulated as follows:

$$y_t = \alpha + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \theta_p y_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t$$

Auto-Regressive Integrated Moving Average (ARIMA) Model

When time series data is not stationary over time, the $ARIMA$ forecasting model is effective in predicting future values. To account for non-stationarity in this case, a non-seasonal differencing hyper-parameter is added to the $ARMA$ model, resulting in a new model $ARIMA(p, q, d)$. The differencing operation involves subtracting a value from its previous value. The formula for taking the difference is below:

$$\Delta y_t = y_t - y_{t-1}$$

Table 1: **Types of Time Series Forecasting Models**

Model Type	**Short Description**	**Type**
Baseline Random Walk	Random Walk Baseline - Guess the previous value	Statistical Forecasting Model
RWS	Random Walk with Slope	Statistical Forecasting Model
SES	Simple Exponential Smoothing	Statistical Forecasting Model
SMA	Simple Moving Average	Statistical Forecasting Model
WMA	Weighted Moving Average	Statistical Forecasting Model
AR	Auto-Regressive (AR) Model	Statistical Forecasting Model
ARMA	Auto-Regressive Moving Average Model	Statistical Forecasting Model
ARIMA	Auto-Regressive Integrated Moving Average Model	Statistical Forecasting Model
RFRTM	Random Forest Regression Tree Model	Regression Tree
GBRTM	Gradient Boosting Regression Tree Model	Regression Tree
LMRTM	Linear Model Regression Tree Model	Regression Tree

5.6 Tree-Based Forecasting Models

This section focuses on regression tree models: Random Forest Regression (RFR), Gradient Boosting Regression (GBR), and Linear Model Trees (LMT). Analogous to Decision Trees and other classification or regression algorithms, a Regression Tree is a machine learning approach that utilizes classification and regression methods by dividing data into branches to make predictions. These models are a part of predictive tools that can effectively handle complex, non-linear patterns observed in time series data.

Additionally, regression trees estimate their parameters by incorporating adjustments for depth, leave nodes, pruning, and splitting to enhance their predictive accuracy and prevent overfitting.

In contrast to traditional statistical modeling techniques of the previous sections, which minimize errors (predict method), regression tree models are related to decision tree models in the sense that both are types of predictive models that divide data into branches to make predictions. However, the two methods are used for different types of predictive tasks. Decision trees are utilized for classification problems, aiming to categorize each input into one of several discrete classes, where the response variable y is defined on small domains $y \in B$ or $y \in \{0, 1, \ldots, k-1\}$. On the other hand, regression trees predict continuous outcomes.

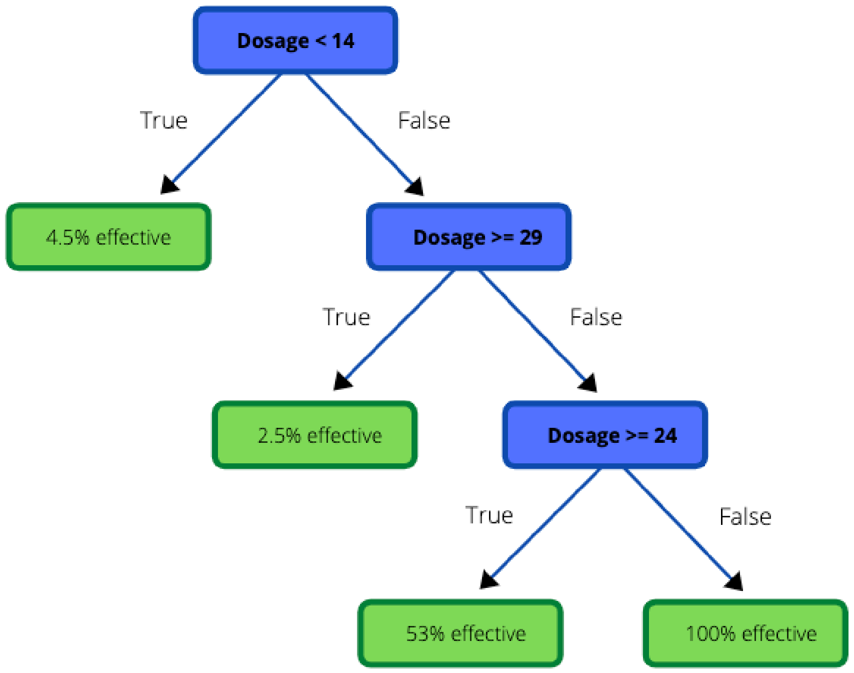

Figure 1: **Regression Tree Diagram**

Linear Model Trees (LMT)

In the modeling of LMT, the feature split identifies suitable thresholds to partition the variables using piecewise linear prediction functions. Like all regression tree models, building an LMT essentially requires finding thresholds for splitting variables/features. Unlike other regression trees, for LMT the region for leaf node is covered by hyperplanes. LMT employs multiple linear regression across these hyperplanes instead of using the average of all the points to estimate the parameters.

As the degrees of freedom in the leaves becomes smaller, the stepwise refinement reduces the number of parameters. Using the example provided in Miller (2024), consider a binary split, at a threshold θ_k that partitions the dataset into two groups. Applying this splitting variable X_j, we may split the rows in the X matrix into left and right groups:

$$\text{left}_k(X) = \{X_i \mid X_{ij} \leq \theta_k\}$$

$$\text{right}_k(X) = \{X_i \mid X_{ij} > \theta_k\}$$

For splitting variable X_j, the threshold θ_k should be chosen to minimize the weighted sum of the Mean Squared Error (MSE) of the left and right sides. A node should only be split and pruned if its multiple regression model is significantly worse than the combination of the two regression models of its children.

Gradient Boosting Regression (GBR)

This is a machine learning technique used to predict continuous values by combining multiple weak models (usually decision trees) in a sequential manner. It starts with a simple model, calculates residuals (errors), and fits new models to these residuals to improve predictions iteratively. The new model corrects the mistakes of the previous ones, and the predictions are combined to form a strong model. It is effective for capturing complex patterns in data but requires careful tuning to avoid overfitting. Additionally, Training multiple sequential models can be time-consuming, especially for large datasets or a high number of iterations. It can be sensitive to outliers, as residuals might disproportionately focus on extreme values. By focusing on minimizing residual errors, GBR often achieves high predictive accuracy.

$$\hat{F}(x) = \arg\min_F \sum_{i=1}^{N} L(y_i, F(x_i))$$

- $\hat{F}(x)$: The optimized predictive model.
- y_i: The actual target value for the i-th data point.
- $F(x_i)$: The prediction of the model for input x_i.
- $L(y_i, F(x_i))$: The loss function.

5.6.1 Random Forest Regression (RFR)

Random Forest Regression is a machine learning algorithm that uses an ensemble of decision trees to predict continuous values. It builds multiple decision trees during training and combines their predictions to improve accuracy and robustness. At each split in a decision tree, a random subset of features is considered for splitting, reducing correlation between trees and improving generalization. Trees grow to their maximum depth without pruning, ensuring that they can capture as much information as possible from the sampled data, the predictions from all individual trees in the forest are averaged to produce the final prediction. The final prediction $\hat{y}$ of the RFR of a given input vector x is computed by averaging the predictions of all numbers of trees in the forest.

$$\hat{y}(x) = \frac{1}{N}\sum_{i=1}^{N} t_i(x)$$

where:

- $\hat{y}$: Predicted output.
- $t_i(x)$: Prediction from the i-th tree for input x.
- N: Total number of trees.

5.7 How Regression Trees are Adapted to Time Series

Adaptation of time series requires modifying how data is structured and used in the model. It transforms time series data into supervised learning format. A regression tree typically predicts a target variable (y) using features (X), and the features are created using lagged values of time series created by itself. Since time series data is sequential, it requires careful transformation to capture temporal dependencies. Features are created using lagged values of the time series itself, this transformation ensures that the model has access to past information which is critical for accurate predictions. In addition to lagged values of the response variable, external variables (exogenous variables) that may influence the target can also be included, which can be integrated into the feature set.Time-based features can be encoded as categorical or numerical variables which helps to capture periodic effects or irregularities.

5.8 COVID-19 Data Set

The main source of the dataset used in this study is Our World in Data (OWID). This dataset contains COVID-19 cases, deaths, hospitalizations, ICU patients, tests, vaccinations, etc. The OWID dataset is aggregated from various sources, including the official Center for Disease Control (CDC) and the Center for Systems Science and Engineering at Johns Hopkins University (CSSE-JHU). Although the OWID data contains COVID-19 information from all over the world, only U.S.-related cases are used in this thesis.

While the dataset on COVID-19 is of daily and weekly frequency, weekly patterns in the 'new death' variable are more distinct than daily frequency, as illustrated in Figure 2.

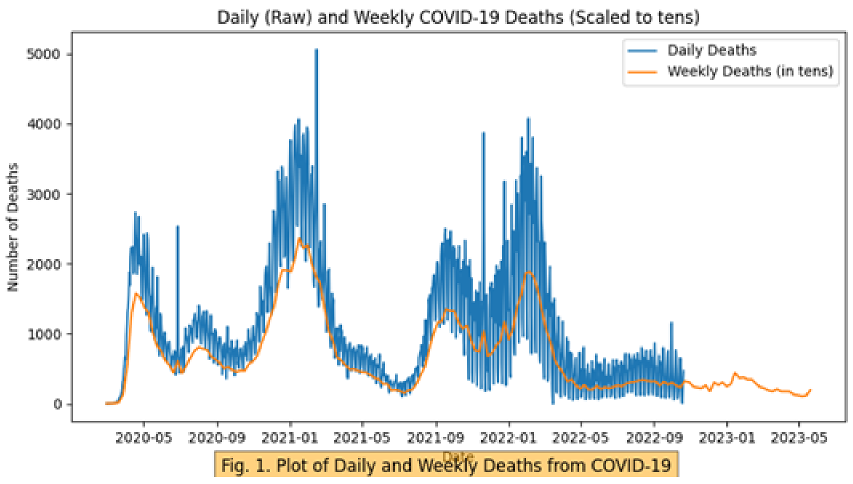

Figure 2: **Plot of Daily and Weekly Deaths from COVID-19**

The autocorrelation function (ACF), also depicted in 2, further corroborates this weekly pattern. The weekly ACF is smoother with less oscillation, implying that for strategic planning and understanding broader trends without the daily noise, weekly data could be more useful.

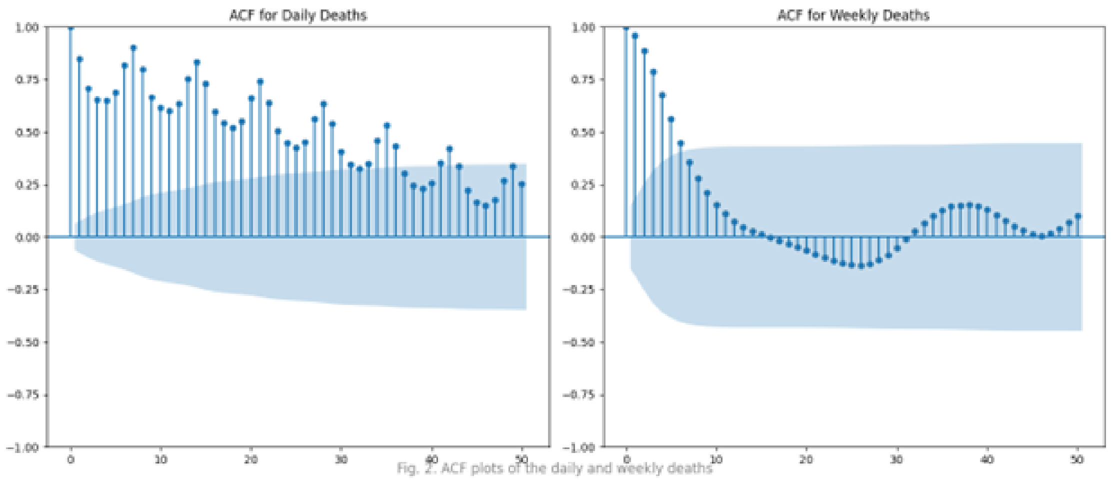

Figure 3: ACF Plots of the Daily and Weekly Deaths

Similar patterns in COVID-19 death rates have been observed in other studies like Miller et al. (2024), indicating that weekly data frequencies provide a more robust basis for analysis. This study, therefore, employs a weekly data frequency for assessing COVID-19 death trends.

The data covers the period February 29th, 2020, to May 20th, 2024, February 29th, 2020, being when the first COVID-19 death in the US was recorded. So, the data covers 169 weeks of data. However, 116 weeks of data is utilized in the study because COVID-19 reported cases reduced, and the trend flattened by May 7, 2022. The variable descriptions are listed on table 2:

5.9 Cross-Correlation Analysis

The Cross-Correlation examines the relationship between two time series to determine how one series is related to or affects the other at various time lags. It is used to find patterns, lead-lag relationships, or dependencies between time-dependent data. It also normalizes or scales the data if needed and ensures time series are stationary.

Table 2: Variable Descriptions

Variable	**Description**
new_cases	New confirmed cases of COVID-19
new_deaths	New deaths attributed to COVID-19
reproduction_rate	Real-time estimate of the effective reproduction rate (R) of COVID-19. See here
icu_patients	Number of COVID-19 patients in intensive care units (ICUs) each week
hosp_patients	Number of COVID-19 patients in hospital each week
new_tests	New tests for COVID-19 each week
positive_rate	The share of COVID-19 tests that are positive, given as a rolling 7-day average
tests_per_case	Tests conducted per new confirmed case of COVID-19, given as a rolling 7-day average
people_vaccinated	Total number of people who received at least one vaccine dose
people_fully_vaccinated	Total number of people who received all doses prescribed by the vaccination protocol
total_boosters	Total number of COVID-19 vaccination booster doses administered
new_vaccinations	New COVID-19 vaccination doses administered
excess_mortality_Abs	Cumulative difference between the reported number of deaths since 1 January 2020 and the projected number of deaths for the same period based on previous years
excess_mortality_cum	Percentage difference between the cumulative number of deaths since 1 January 2020 and the cumulative projected deaths for the same period based on previous years
excess_mortality	Percentage difference between the reported number of weekly deaths in 2020–2021 and the projected number of deaths for the same period based on previous years
excess_mortality_Mill	Cumulative difference between the reported number of deaths since 1 January 2020 and the projected number of deaths for the same period based on previous years, per million people

Chapter 6

Performance Evaluation and Discussions

The result and discussion section of this thesis aims to comprehensively evaluate the performance of regression tree forecasting against traditional forecasting approaches within Python and ScalaTion programming environments. First, this section is organized to evaluate how the choice of statistical software and forecasting models influence forecast performance. Prior to that, the choice of statistical software should not affect the forecasting evaluated in this section. Next, the analysis in this section examines the performance of regression trees and traditional statistical models relative to their baseline models. sMAPE and MAE are the main metrics utilized for this purpose. Furthermore, this section delves into the impacts of validation techniques, including train-test splits with rolling validation (TnT) and in-sample testing (In-ST), on the performance of regression trees and traditional statistical models relative to their baseline models and within Python and ScalaTion. Lastly, the programming environment, Python or ScalaTion, that offers superior overall performance for forecasting is presented.

6.1 Symmetric Mean Absolute Percentage Error (sMAPE) Metric

The forecast accuracy is measured by the symmetric Mean Absolute Percentage Error (sMAPE), where smaller values indicate better performance. The sMAPE is one of the many metrics used to measure the accuracy of a forecasting model. Although sMAPE is like the traditional Mean Absolute Percentage Error (MAPE), it has an in-built symmetric measure which treats over- and under-forecasting equally. The sMAPE is calculated using the formula:

$$\text{sMAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \frac{|y_t - \hat{y}_t|}{(|y_t| + |\hat{y}_t|)/2}$$

where n is the number of observations, y_t is the actual value, and $\hat{y}_t$ is the predicted value. The apparent advantage of sMAPE over MAPE is that it typically can avoid divide-by-zero errors.

6.2 Test Setup

The response variable, which is COVID-19 deaths, is described as the new deaths attributed to COVID-19. Although the OWID dataset covers 169 weeks from February 29th, 2020, to May 20th, 2024, only the first 116 weeks of data are covered in this study because COVID-19-related deaths flattened by May 7, 2022. only the first 116 weeks of data is covered in this study because covid-19 related deaths flattened by May 7, 2022 as shown on the red colored portion of the trend on Figure 4 below

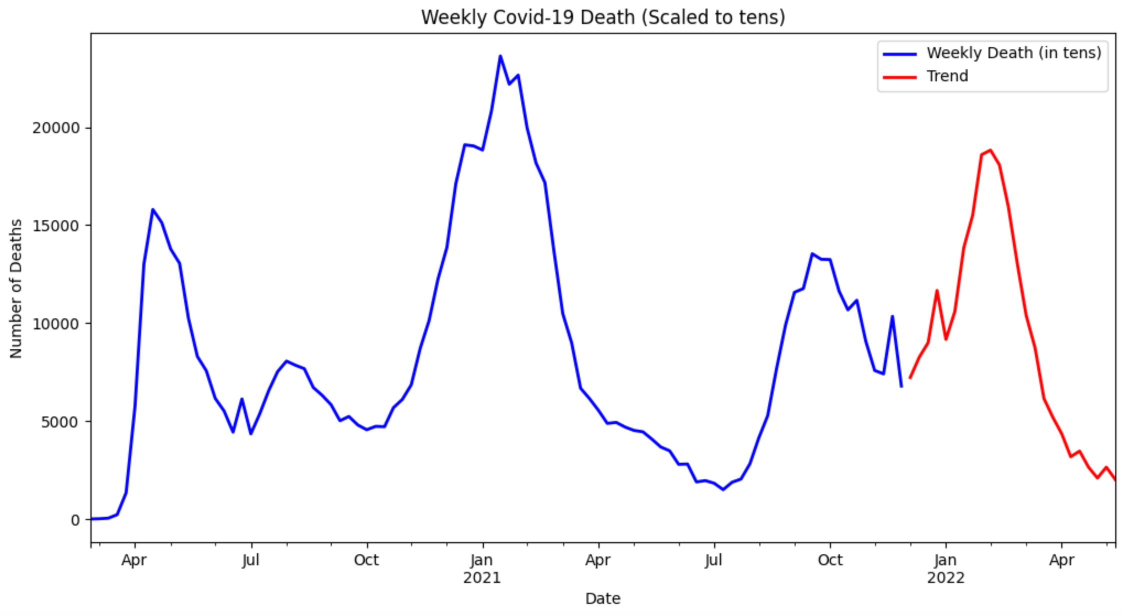

Figure 4: Plot of Weekly Deaths from COVID-19

Since classical multi-fold cross-validation is not applicable to time series data (see Miller, 2024) due to the presence of serial correlation, in-sample validation and rolling validation with test and train split are used in this study. Specifically, I train and test on the same dataset to implement in-sample validation on both traditional and regression tree models. Since in-sample validation trains and tests on the same dataset, it is considered the basic and starting point of forecasting performance analysis in this study.

For rolling validation, the train-test split is set at 80/20. That is, 80% of the data is used for training, and the remaining 20% of the dataset is used for testing the rolling validation to make a new COVID-19 death forecast. A forecast horizon of $1 \leq h \leq 6$ is used in training and testing the forecasting models. That is, an input from a rolling window which contains the information from $t-5$ to time t, and an out-of-sample forecast for time $t+h$. Then, the rolling window moves forward, and the process is repeated. On the other hand, I also consider the ability of this model to predict the value one step ahead in the in-sample validation (also known as skip $= 1$ in ScalaTion and Python).

To evaluate the performance of both traditional statistical forecasting methods and the Random Walk model, the Random Walk model is considered the baseline method against which the performance of other forecasting models is measured. Three traditional statistical models are considered in this study: Random Walk (RW), Auto-Regressive AR (1), and Auto-Regressive Moving Average ARMA (1, 0) models. Three regression tree models are evaluated against the benchmark and traditional forecasting models: Gradient Boosting Regression Tree (GBRT) Model, Random Forest Regression Tree (RFRT) Model, and Linear Model Regression Tree (LMRT) Model.

6.3 Forecasting Performance: Traditional Forecasting Models

I compare the forecasting performance of statistical models RW, AR (1), ARMA (1,0). The statistical forecasting model is shown in Table 3 below. For each model, a 6-week forecasting horizon specified sMAPE of COVID-19 'new deaths' is computed for each horizon (or week) under the in-sample validation.

I compare the performance of all the statistical models with the baseline under ScalaTion and Python. RW model exhibited perfect alignment between ScalaTion and Python across all horizons, with identical sMAPE values. This consistency underscores the correctness and robustness of the implementation in both platforms. The average sMAPE for the RW model was 42.086%. For the AR (1) model, Python produced slightly higher sMAPE values than ScalaTion, with an average sMAPE of 39.814% in Python compared to 39.573% in ScalaTion. The absolute difference of 0.241 corresponds to a 0.609% higher value in Python, indicating near-equivalent results. Python showed marginally higher sMAPE values than ScalaTion for the ARMA (1,0) model. The average sMAPE was 40.549% in Python compared to 39.588% in ScalaTion, with an absolute difference of 0.961. This represents a 2.428% higher value in Python, indicating strong agreement with minor variance.

Summing up, the results demonstrate excellent consistency between ScalaTion and Python for all three models, with minimal discrepancies. However, the Random Walk model exhibited perfect parity, while differences for the AR (1) and ARMA (1,0) models were within a 2.4% margin. These small deviations could stem from platform-specific implementation differences, such as parameter initialization or numerical precision.

The close alignment across platforms supports the reliability of both ScalaTion and Python for time series forecasting, with the observed differences.

6.4 Rolling Validation: Traditional Forecasting Models

This section presents the sMAPEs of traditional statistical models in ScalaTion and Python under Train and Test split with Rolling Validation with skip of 0. Like the in-sample validation, skip of 1 method in Tables 3, AR and ARMA forecasting models outperform the benchmark in both ScalaTion and Python environments, thus demonstrating equivalent forecasting efficiency between Python and ScalaTion.

Notably, the AR and ARMA forecasting models have the least sMAPEs in both Python and ScalaTion environments, thus showing greater consistency. In Table 4, both ScalaTion and Python produced

Table 3: sMAPEs, In-sample, Skip = 1

Horizon	RW	AR (1)	ARMA (1,0)
ScalaTion			
1	19.037	18.880	20.219
2	29.580	28.591	29.911
3	39.074	37.096	38.153
4	47.464	45.503	45.586
5	55.179	51.446	52.292
6	62.182	55.924	57.367
Average	42.086	39.573	39.588
Python			
1	19.037	18.755	19.740
2	29.580	28.567	29.537
3	39.074	37.240	38.067
4	47.464	45.801	45.294
5	55.179	51.896	52.718
6	62.182	56.624	57.940
Average	42.086	39.814	40.549
Abs Difference	0.000	0.241	0.961
As a % of ScalaTion	0.000%	0.609%	2.428%

identical average sMAPE values of 46.514%, indicating perfect alignment. This highlights the consistency and robustness of the RW implementation in both environments.

The AR (1) model showed marginal differences between the platforms, with Python yielding an average sMAPE of 48.137%, compared to 48.367% in ScalaTion. The absolute difference is 0.231, reflecting a 0.477% lower sMAPE in Python. ARMA (1,0) Python and ScalaTion results were nearly identical, with average sMAPEs of 48.043% in Python and 47.990% in ScalaTion. The absolute difference is 0.053, corresponding to a 0.110% increase in Python.

The results demonstrate high consistency between ScalaTion and Python implementations across all models, with differences well within 3% for any model. The AR (1) and ARMA (1,0) exhibited near-equivalent performance.

These findings reinforce the reliability of both platforms for rolling validation. Again, the minor discrepancies likely arise from platform-specific optimization factors, and they are unlikely to affect the overall conclusions of this study. Further analysis could explore whether these small variations impact long-term forecasting accuracy.

Table : Table 4: sMAPEs TnT with Rolling Validation, skip=0

Horizon	RW	AR(1)	ARMA(1,0)
ScalaTion			
1	18.671	19.159	19.001
2	27.572	31.198	30.394
3	40.939	44.485	43.801
4	52.350	55.312	54.825
5	64.248	65.554	65.374
6	75.302	74.497	74.546
Average	46.514	48.367	47.990
Python			
1	18.671	18.894	18.211
2	27.572	31.331	30.197
3	40.939	44.545	43.374
4	52.350	55.939	55.320
5	64.248	64.653	65.832
6	75.302	73.458	75.325
Average	46.514	48.137	48.043
Abs Difference	0.000	0.231	0.053
As a % of ScalaTion	0.000%	0.477%	0.110%

6.5 Forecasting Performance of Regression Tree Models

In this section, I compare the forecasting performance of regression tree models in ScalaTion and Python environments. For each model, a 6-week forecasting horizon is specified, and the sMAPE of 'COVID-19' new-deaths are computed for each horizon (or week) under the in-sample validation as shown in Table 5.

Gradient Boosting ScalaTion average sMAPE of 48.240% while Python average sMAPE is 18.423%, with an absolute difference of 19.919, corresponding to a 61.809% reduction in Python. The large discrepancy suggests possible implementation issues in ScalaTion for this model.

Random Forest ScalaTion has an average sMAPE of 48.798% and Python exhibits an average sMAPE of 28.879%, with an absolute difference of 19.919, equating to a 40.820% reduction in Python. This model also shows a substantial deviation, indicating potential problems in the ScalaTion results.

For the Linear Model Tree, ScalaTion exhibits an average sMAPE of 49.262%, while Python shows an average sMAPE of 19.299%, with an absolute difference of 29.963, representing a 60.824% reduction in Python. While smaller than the other models, the difference is still significant and suggests inconsistencies.

The results indicate that Python consistently produces significantly lower sMAPEs across all models compared to ScalaTion. The discrepancies range from 60% to nearly 70%, suggesting that the ScalaTion

Table 5: sMAPEs, In-sample, Regression Tree Models, Skip = 0

Horizon	Gradient Boosting	Random Forest	Linear Model
ScalaTion			
1	21.5134	18.3946	20.4691
2	25.9738	24.5295	30.2268
3	41.1092	42.6127	40.3940
4	56.5362	58.8470	56.7983
5	68.8753	69.8738	70.2365
6	75.4296	78.5313	77.4489
Average	48.240	48.798	49.262
Python			
1	13.4349	20.9072	11.8157
2	15.9854	24.7410	14.2732
3	17.4999	28.4262	18.8723
4	19.6251	31.6257	21.1263
5	21.3466	33.1615	26.8958
6	22.6463	34.4114	22.8107
Average	18.423	28.879	19.299
Abs Difference	29.817	19.919	29.963
As a % of ScalaTion	61.809%	40.820%	60.824%

results for Gradient Boosting, Random Forest, and Linear Model Tree may be inflated due to potential bugs or inaccuracies in the implementation.

The Python results align better with expectations for in-sample validation, suggesting they are likely more reliable. Further investigation is needed to debug the ScalaTion implementation, particularly for Gradient Boosting and Random Forest models, to ensure accurate comparisons. For the purposes of this study, Python results may serve as a more reliable benchmark.

6.6 Rolling Validation: Tree-Based Forecasting Models

In this section, Table 6 reports the symmetric Mean Absolute Percentage Errors (sMAPEs) for three forecasting models: Gradient Boosting, Random Forest, and a Linear Model Tree, evaluated using rolling validation over six forecast horizons. All models were implemented in Python.

Table 6: sMAPEs, Rolling Validation, Python

Horizon	Gradient Boosting	Random Forest	Linear Model
1	19.9130	19.86320	11.8157
2	20.0057	19.84750	14.2732
3	21.5491	20.88040	18.8723
4	26.1935	23.71190	21.1263
5	30.7703	26.55390	26.8958
6	34.9731	29.56960	22.8107
Average	25.567	23.404	19.299

6.7 Conclusion

This thesis examines the predictive powers of Regression tree and Traditional statistical models in two distinct programming environments – Python and ScalaTion, with focusing on the performance of both models in predicting COVID-19 deaths. For this comparison, this thesis uses sMAPE. Importantly, the findings in this thesis demonstrate that the choice of programming environment does not significantly affect the predictive performance of the models.

Additionally, the findings in this thesis show that Python results align better with expectations for In-Sample and TnT with rolling validation, suggesting they are likely more reliable. Furthermore, investigation is needed to debug the ScalaTion implementation, despite the inherent advantages of ScalaTion in handling time-series data through its embedded libraries and optimized packages, the observed deviations emphasize the need for rigorous debugging and validation. Ensuring consistent hyperparameter settings, numerical precision, and parameter initialization in ScalaTion is essential to achieve results comparable to Python.

Within statistical forecasting models, advanced statistical models like AR and ARMA outperform their baseline models in both Python and ScalaTion, suggesting their enhanced ability to handle complex forecasting tasks compared to simpler models. Among traditional models, AR and ARMA models performed exceptionally well. While Python emerges as the more reliable platform in this study, the results underscore the necessity of addressing potential bugs in ScalaTion's implementation. This will not only ensure fair comparisons but also leverage ScalaTion's capabilities for more accurate and scalable forecasting in future studies.

6.8 Future Studies

This research has several opportunities for future investigation in the field of forecasting, particularly in the context of Tree-based models and pandemic response:

1. **Integration of Advanced Machine Learning Techniques:** Future studies could explore the incorporation of advanced machine learning methods, such as deep learning architectures, into Tree-based forecasting models. This integration may yield improvements in predictive accuracy and model interpretability, particularly for complex non-linear relationships in the data.

2. **Expansion of Exogenous Variables:** Future research should investigate the impact of additional exogenous variables, such as socioeconomic factors, mobility data, and vaccination rates, on the accuracy of forecasting models. By broadening the feature set, researchers may uncover new insights and improve model robustness.

3. **Long-Term Forecasting and Seasonal Adjustments:** Investigating the long-term forecasting capabilities of Tree-based models, particularly in the presence of seasonal trends, could enhance understanding of pandemic dynamics. Future work could explore methods for effectively incorporating seasonal adjustments within the framework of tree-based models.

By pursuing these directions, future research can further advance the understanding and application of Tree-based forecasting methods, ultimately contributing to improved pandemic preparedness and response.

6.9 Bibliography

- Barmparis, G., and Tsironis, G. (2020). Estimating the infection horizon of COVID-19 in eight countries with a data-driven approach. *Chaos, Solitons & Fractals*, 135, 109842. https://doi.org/10.1016/j.chaos.2020.109842

- Bojer, C. S., & Meldgaard, J. P. (2020). Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*. https://arxiv.org/pdf/2009.07701

- Cotterell, M. E., Miller, J. A., and Horton, T. (2011). Unicode in Domain-Specific Programming Languages for Modeling & Simulation: ScalaTion as a Case Study. *University of Georgia*. https://doi.org/10.48550/arXiv.1112.1751

- Cutler, R., Edwards, T.C., Beard, K.H., Cutler, A., Hess, K.T., Gibson, J., Lawler, J.J. (2007). Random forests for classification in ecology. *Ecology Society of America*, 2783-2792. https://doi.org/10.1890/07-0539.1

- Cramer, E. Y., Lopez, V. K., Niemi, J., et al. (2021). Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the US. *medRxiv*. https://doi.org/10.1073/pnas.2113561119

- Fazeli, S., Moatamed, B., and Sarrafzadeh, M. (2020). Statistical analytics and regional representation learning for COVID-19 pandemic understanding. https://doi.org/10.48550/arXiv.2008.07342

- Ham, Y., and Kug, J. (2015). Improvement of ENSO Simulation Based on Intermodel Diversity. *Journal of Climate*, 28, 998-1015. https://doi.org/10.1175/JCLI-D-14-00376.1

- Hyndman, R. J., and Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*. OTexts.

- Hyndman, R. J., and Taieb, S. B. (2012). Recursive and direct multi-step forecasting: the best of both worlds. *Citeseer*, 19.

- Ismail, L., Materwala, H., Znati, T., Turaev, S., Khan (2020). Tailoring time series models for forecasting coronavirus spread: Case studies of 187 countries. *Computational and Structural Biotechnology Journal*, 2972-3206. https://doi.org/10.1016/j.csbj.2020.09.015

- Januschowski, T., Wang, Y., Torkkola, K., et al. (2022). Forecasting with trees. *International Journal of Forecasting*, 38, 1473-1481. https://doi.org/10.1016/j.ijforecast

- Javeri, I. Y., Toutiaee, M., Arpinar, I. B., et al. (2021). Improving Neural Networks for Time Series Forecasting using Data Augmentation and AutoML. https://arxiv.org/abs/2103.01992

- Jeon, Y., and Seong, S. (2021). Robust recurrent network model for intermittent time-series forecasting. *International Journal of Forecasting*. http://dx.doi.org/10.1016/j.ijforecast.2021.07.004

- Khan, M. M. A. (2013). Forecasting gold prices (Box Jenkins approach). *International Journal of Emerging Technology and Advanced Engineering*, 3(3), 662-670.
- Krauss, C., Do, X. A., Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the SP 500. *FAU Discussion paper in Economics (Working Paper)*. https://www.econstor.eu/bitstream/10419/130166/1/856307327.pdf
- Lim, B., Arik, S. O., Loeff, N., Pfister, T. (2019). Temporal fusion transformers for interpretable multi-horizon time series forecasting. https://arxiv.org/abs/1912.09363
- Miller, J. A. (2020). *Introduction to Data Science using ScalaTion*. Release 2. Department of Computer Science, University of Georgia. https://cobweb.cs.uga.edu/~jam/scalation_guide/data_science.pdf
- Miller, J. A. (2024). *Introduction to Computational Data Science using ScalaTion*. Department of Computer Science, University of Georgia. https://cobweb.cs.uga.edu/~jam/scalation_guide/comp_data_science.pdf
- Navin, G. V. (2015). Big Data Analytics for Gold Price Forecasting Based on Decision Tree Algorithm and Support Vector Regression (SVR). *International Journal of Science and Research (IJSR)*, 4(3), 2026-2030.
- Nishimoto, Y., and Inoue, K. (2020). Curve-fitting approach for COVID-19 data and its physical background. https://www.medrxiv.org/content/10.1101/2020.07.02.20144899v2.full
- Odersky, M., Spoon, L., Venners, B. (2016). *Programming in Scala*. Artima.
- Picchiotti, N., Salvioli, M., Zanardini, E., et al. (2020). COVID-19 pandemic: a mobility-dependent SEIR model with undetected cases in Italy, Europe, and the US. https://arxiv.org/pdf/2005.08882
- Rady, H. A., Fawzy, H., and Abdel Fattah, A. M. (2021). Time Series Forecasting Using Tree-Based Methods. *Journal of Statistics Applications Probability*, 1, 229-244.
- Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. (2021). Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting. http://arxiv.org/abs/2101.12072
- Shamil, M. S., Farheen, F., Ibtehaz, N., Khan, I. M., and Rahman, M. S. (2021). An agent-based modeling of COVID-19: Validation, analysis, and recommendations. *Cognitive Computation*. https://www.medrxiv.org/content/10.1101/2020.07.05.20146977v1.full.pdf
- Tweedie, M. (1947). Functions of a statistical variate with given means, with special reference to Laplacian distributions. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43, 41–49. Cambridge University Press.
- Zhou, H., Zhang, S., Peng, J., et al. (2021). Informer: Beyond Efficient Transformer for Long Sequence, Time Series Forecasting. In *The thirty-fifth AAAI conference on artificial intelligence*.