Dynamic Hybrid Weather Forecasting Model for Olive Tree Cultivation in Mediterranean Climates

Magret B. Oladunjoye

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Software Engineering

Eastern Mediterranean University
January 2025
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Software Engineering.

Prof. Dr. Zeki Bayram
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Software Engineering.

Prof. Dr. Hadi Işık Aybay
Supervisor

Examining Committee

1. Prof. Dr. Hadi Işık Aybay
2. Prof. Dr. Önsen Toygar
3. Assoc. Prof. Dr. Mehtap Köse Ulukök

ABSTRACT

Accurate agricultural forecasting is crucial, particularly in regions where temperature-sensitive crops, such as olives, require careful management. This thesis work introduces a hybrid forecasting model that integrates Facebook Prophet, a statistical time-series forecasting tool, with a Gated Recurrent Unit (GRU) network to predict maximum daily temperatures in olive-producing regions. A key contribution of this work is the integration of Growing Degree Days (GDD), both as a predictive feature for refining temperature forecasts and as a real-time metric within a web-based application. This application tracks accumulated heat units in real-time, allowing farmers to accurately monitor olive tree growth stages and optimize cultivation decisions. By incorporating GDD calculations specific to olive cultivation, the model provides meaningful insights to support agricultural planning.

The model was tested on diverse regional datasets from Tuscany (Italy), Almería (Spain), Kalamata (Greece), Bursa (Tukey), and Larnaca (Cyprus), demonstrating its adaptability across Mediterranean and temperate olive-growing regions. The model demonstrated superior accuracy in most test cases compared to established forecasting methods, including Seasonal AutoRegressive Integrated Moving Average (SARIMA), Simple Moving Average (SMA), standalone Long Short-Term Memory (LSTM) networks, and standalone GRU networks. On the Larnaca dataset, it achieved a Mean Squared Error (MSE) of 0.004613, a Mean Absolute Error (MAE) of 0.051322, an R^2 value of 0.838467, and a Mean Absolute Percentage Error (MAPE) of 11.52%, surpassing all baseline methods in accuracy.

For practicality, a web application was developed to use the hybrid model's forecasts

to determine the current and historical GDD values and growth stages of plants. By integrating advanced forecasting with a user-friendly web tool, this system serves as an effective decision-support platform for climate-smart agriculture. Future work will incorporate additional environmental factors, such as precipitation, soil moisture, and humidity, to further enhance forecasting accuracy. These developments would further strengthen the model's utility in precision farming and adaptation strategies for climate variability.

Keywords: Temperature Forecasting, Hybrid Model, Gated Recurrent Units, Growing Degree Days, Mediterranean Climate, Agricultural Decision Support.

ÖZ

Doğru tarımsal tahminler, özellikle sıcaklığa duyarlı mahsullerin, örneğin zeytinlerin dikkatli yönetimi gereken bölgelerde büyük önem taşımaktadır. Bu tez çalışması, zeytin üretim alanlarında günlük maksimum sıcaklıkları tahmin etmek için istatistiksel tabanlı Facebook Prophet algoritmasını ve sıralı öğrenme yeteneğine sahip Geçitli Tekrarlayan Birim (GRU) ağını birleştiren hibrit bir tahmin modeli sunmaktadır. Bu çalışmanın önemli bir katkısı, Büyüme Derecesi Günleri (GDD) konseptinin hem sıcaklık tahminlerini iyileştiren bir tahmine dayalı özellik olarak hem de gerçek zamanlı bir metrik olarak bir web tabanlı uygulama içinde kullanılmasıdır. Bu sistem, birikmiş ısı birimlerinin gerçek zamanlı takibini sağlayarak çiftçilerin zeytin ağacı büyüme aşamalarını doğru bir şekilde değerlendirmesine ve ekim kararlarını optimize etmesine yardımcı olur. Zeytin yetiştiriciliğine özel GDD hesaplamaları sayesinde model, tarımsal planlama için anlamlı içgörüler sunacak şekilde tasarlanmıştır.

Model, farklı Akdeniz iklim koşullarına sahip Toskana (İtalya), Almería (İspanya), Kalamata (Yunanistan) ve Larnaka (Kıbrıs) bölgelerinden alınan çeşitli veri kümeleri üzerinde değerlendirilmiştir. Model, çoğu test senaryosunda, Mevsimsel Otoregresif Entegre Hareketli Ortalama (SARIMA), Basit Hareketli Ortalama (SMA), bağımsız Uzun Kısa Süreli Bellek (LSTM) ağları ve bağımsız GRU ağları gibi geleneksel tahmin yöntemlerine kıyasla üstün doğruluk göstermiştir. Larnaka veri seti üzerinde yapılan değerlendirmede, model 0,004613 Ortalama Karesel Hata (MSE), 0,051322 Ortalama Mutlak Hata (MAE), 0,838467 R^2 değeri ve %11.52 Ortalama Mutlak Yüzde Hata (MAPE) elde etmiş ve tüm temel modellerden daha yüksek doğruluk sağlamıştır.

Pratik kullanım için, hibrit modelin tahminlerini kullanarak mevcut ve geçmiş GDD

değerlerini ve bitki büyüme aşamalarını belirleyen bir web uygulaması geliştirilmiştir. Gelişmiş tahmin yöntemleriyle kullanıcı dostu bir web arayüzünü entegre eden bu sistem, iklime duyarlı tarım uygulamaları için etkili bir karar destek platformu olarak hizmet vermektedir. Gelecekteki çalışmalarda, tahmin doğruluğunu daha da artırmak için yağış, toprak nemi ve nem gibi ek çevresel faktörlerin modele dahil edilmesi planlanmaktadır. Bu gelişmeler, modelin hassas tarım ve iklim değişkenliğine uyum sağlama alanlarındaki kullanımını daha da güçlendirecektir.

Anahtar Kelimeler: Sıcaklık Tahmini, Hibrit Model, Geçitli Tekrarlayan Birimler, Büyüme Derecesi Günleri, Akdeniz İklimi, Tarımsal Karar Destek.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to the Erasmus+ DeepFarm Project for their support, resources, and collaboration, which made this work possible.

I am also incredibly grateful to my academic community—colleagues, mentors, and friends—for supporting me throughout this journey. Your encouragement, insightful discussions, and shared curiosity have shaped not only this thesis but also my growth as a researcher.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
API	Application Programming Interface
BEKF	Bayesian Ensemble Kalman Filter
CSV	Comma-Separated Values
DL	Deep Learning
GDD	Growing Degree Days
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Squared Error
NCEI	National Centers for Environmental Information
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SARIMA	Seasonal AutoRegressive Integrated Moving Average
TMIN	Minimum Temperature
TMAX	Maximum Temperature

Chapter 1

INTRODUCTION

Olive cultivation, which has provided a source of income for numerous families in the Mediterranean region for many years, is both an art and a science. Like many other crops, olive production also depends on factors such as soil type, water, pests, and, most importantly, climate. Temperature is particularly crucial in the growth, production, and quality of olives. Unpredictable weather patterns and climate variability have been problematic for farmers in recent years, as they make it harder to determine the right time to carry out certain critical agricultural activities.

1.1 Problem Statement

The traditional statistical forecasting techniques that have been in use for several decades now fail to provide the level of precision required for daily decision-making in the agricultural sector. However, advancements in machine learning algorithms are addressing these challenges by enabling more precise and adaptable weather forecasting. These innovations have the potential to improve farming practices through the delivery of tailored forecasts, such as monitoring temperature changes and examining seasonal tendencies on a micro-regional scale.

This research used valuable tuning techniques specifically designed for olive growers. It enables farmers to make informed decisions based on actionable insights derived from forecasting data. The study plays a significant role in promoting sustainable and efficient olive farming, especially in the context of climate change, by helping to interpret complex climate data and effectively apply it in agriculture.

1.2 Thesis Aim and Objectives

This research has been prompted by the growing need to improve agriculture's precision and resilience, especially in the context of increasing climate variability. Under current and future climate change scenarios, olive cultivation, which is at the heart of Mediterranean agriculture, is particularly vulnerable to weather variability. Temperature, rainfall, and seasonal changes, along with their uncertainties, impact crop quality and production, presenting new challenges to olive farmers who must maintain productivity and profitability.

Accurate, localised temperature forecasts are not merely nice to have; they are essential tools for farmers to make informed decisions. Whether it involves deciding when to turn on the irrigation, when to apply pesticides and disease control measures, or when to harvest, having precise weather information can mean the difference between a successful and disappointing season for the olive farmer, affecting both the quality of the yield and the efficiency of operations.

This thesis aims to bridge the gaps in traditional forecasting through the development of a statistical and deep learning hybrid model. Long-term trends and seasonal patterns are identified and modelled using Facebook Prophet and GRU networks, which are utilised to fine-tune these forecasts to incorporate non-linear relationships and short-term variations.

To make the model reliable and adaptable, data is collected from major olive producing areas of the Mediterranean region, namely Cyprus, Spain, Italy, Greece and Turkey. The model is designed to perform well in a variety of environmental settings and it is a useful tool for farmers seeking to prepare for the implications of climate change and

to improve their agricultural production.

1.3 Thesis Layout

This thesis is divided into five chapters which are all relevant to the hybrid weather forecasting model and its application in olive production.

The first chapter of this paper is the Introduction, which sets the stage for the study. It outlines the problems of weather forecasting in agriculture especially for the olive growing areas and the shortcomings of the conventional statistical models. The objectives, scope, and significance of the study are also stated.

The second chapter, Literature Review, examines existing weather forecasting techniques, covering statistical methods, machine learning models, and hybrid approaches. It discusses the advantages of combining statistical decomposition (Facebook Prophet) with deep learning (Gated Recurrent Units) for time-series forecasting. Additionally, the concept of Growing Degree Days (GDD) and its relevance to plant growth tracking are explored.

In the third chapter, Methodology, the end to end process of developing the hybrid forecasting model is explained in detail. The Software Development Life Cycle (SDLC), feasibility considerations and data preprocessing techniques are outlined. The model architecture is explained, including the decomposition of temperature trends, sequence modeling with GRUs, and real-time updates using the Bayesian Ensemble Kalman Filter (BEKF). This chapter also describes the database structure and system design, supported by UML diagrams and an Entity-Relationship (E-R) model.

The fourth chapter, Implementation, Results, and Discussion, presents the technical

implementation of the forecasting model and web application. It describes the development tools used, RESTful API design, and frontend visualization using React and Plotly.js. The performance of the hybrid model is evaluated using metrics namely Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and R^2 score. A comparative analysis with baseline models is provided, along with insights on the accuracy and efficiency of the proposed system. The web application's quality control measures, user experience considerations, and real-world usability are also discussed.

The fifth chapter, Conclusion and Future Work summarizes the research findings and their implications for agricultural weather forecasting. The importance of the research findings and their application in agricultural weather forecasting are presented. Some improvements are suggested, including: enhancing the model generalization, using the model for other crops, and increasing the number of climate variables. The study's relevance to precision agriculture and climate resilience is also emphasized.

The thesis concludes with a References section, listing all cited works, ensuring proper attribution of prior research.

Chapter 2

LITERATURE REVIEW

Weather forecasting has historically relied on statistical models that use past data to predict future conditions. But because of increasing complexity in climate patterns and the need to provide more accurate forecasts, machine learning (ML) and deep learning (DL) algorithms have gained prominence. These methods enable the models to learn non-linear relationships, temporal dependencies, and high-dimensional feature interactions from meteorological datasets. Hybrid models have also recently emerged[2], integrating the interpretability of traditional statistical techniques with the flexibility and predictive power of machine learning (ML) and deep learning (DL) methods. In this chapter, an examination of the literature on forecasting will be presented, ranging from the classical statistical approaches to the current AI based models. The significant developments and their application to weather forecasting will be discussed.

2.1 Statistical Methods for Weather Forecasting

Statistical methods have long been foundational in weather forecasting. Techniques such as linear regression, autoregressive integrated moving average (ARIMA), and seasonal ARIMA (SARIMA) are widely used due to their simplicity and ability to model historical data for future prediction. These methods focus on identifying trends, seasonality, and correlations. However, these methods are restricted to stationary and linear data. For example, ARIMA is effective for univariate time series forecasting but struggles with nonlinearity in weather data [3].

To enhance the handling of multivariate data with mixed types of variables, advanced

statistical techniques, such as Generalized Additive Models (GAMs) and Principal Component Analysis (PCA), are employed to transform and reduce the dimension of the data. Nevertheless, these methods are still not very efficient in predicting sudden and nonlinear changes in weather. This limitation has led to a growing interest in hybrid models that combine statistical and machine learning approaches. [4].

2.2 Machine Learning Methods for Weather Forecasting

Machine learning (ML) and deep learning (DL) methods have revolutionized weather forecasting by allowing the models to pick up the the non-linear and complex patterns in large intricate datasets. Traditional statistical models, while effective in capturing linear trends and seasonality, often struggle with the nonlinear nature of weather data. ML algorithms, such as Support Vector Machines (SVM), Random Forests (RF), and Gradient Boosting Machines (GBMs), have been extensively used for tasks such as rainfall prediction and temperature forecasting. In addition to handling non-linear data well, these methods also do well in feature selection, choosing the most consistent, non-redundant, and relevant features to use in model construction. The results clearly show significant improvement in comparison to traditional approaches.[5].

Deep learning models have further enhanced weather forecasting by addressing the temporal dependencies inherent in sequential data. Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) are particularly effective in modeling time-series data, as they mitigate the vanishing gradient problem and capture long-term dependencies. For instance, Shi et al. proposed ConvLSTM for precipitation forecasting, which demonstrated the ability to model spatial-temporal dependencies effectively [6].

CNNs have also been used for spatial weather data, such as satellite imagery. A com-

bination of a CNN and a recurrent network has improved the predictive accuracy by using both spatial and temporal features [7].

Some of the newest developments include:

- **Transformer Based Models**: Transformer architectures have been successfully employed for medium range weather forecasting by means of self attention [8].
- **Physics–AI Hybrid Models**: Physical simulations combined with AI techniques provide a compact representation that combines the data-driven methods and scientific principles, as demonstrated by [9].
- **Generalizable AI Models**: These are models developed for modeling of the earth's system. They emphasize the importance of robust architectures which are capable of adapting across different geographies and datasets [10].

2.3 Hybrid Models for Weather Forecasting

To analyze both the linear and non-linear weather components, hybrid models attempt to combine the strengths of statistical and machine learning methods. Statistical tools like ARIMA or Facebook Prophet help to capture the long-term trend and seasonality while machine learning techniques improve on the residuals for a better fit [11].

For example, Yu et al. [12] proposed a hybrid ARIMA-LSTM model that could well characterize both the linear and nonlinear dynamics of the system and found that it outperformed the corresponding standalone approaches. Facebook Prophet has also been really popular for breaking down time series data into its components for deeper analysis with techniques like GRUs [13].

New hybrid approaches including physics informed machine learning embeds prior physical knowledge and constraints to enhance the interpretability and extendability of

the models. Hybrid models that combine numerical weather predictions (NWP) with data driven methods have been found to be very efficient in predicting variables such as rainfall and stream flow [2].

2.4 Facebook Prophet

Facebook Prophet is an open-source library specifically designed for time-series forecasting with strong seasonal trends. It decomposes data into trend, seasonal, and residual components, allowing for interpretable predictions. Prophet is highly flexible, handling missing data, outliers, and custom seasonalities with ease [11].

Prophet is also able to integrate external regressors, such as humidity or precipitation, to improve accuracy. It includes many customizable options for users to tweak and adjust forecasts. Recent studies, such as the work by Ahmed et al., have successfully combined Prophet with GRU networks, achieving significant improvements in predictive performance [13]. Hybrid approaches like VAR-GRU further underscore Prophet's versatility in agricultural applications [14].

Chapter 3

METHODOLOGY

This chapter thoroughly details the comprehensive pipeline and methodology involved in developing a hybrid temperature forecasting model for olive-producing regions.

3.1 Overview

Data collection, preprocessing, feature engineering, decomposition, hybrid model implementation, and performance evaluation is discussed here. background information on Gated Recurrent Units (GRUs), the rationale for using Growing Degree Days (GDD) in agricultural forecasting, and an overview of the V-model software Development Life Cycle (SDLC) employed in the study. Flowcharts and diagrams are included for visual representation.

3.2 Software Development Life Cycle (SDLC)

The development of this forecasting system was done using the V-Model Software Development Life Cycle (SDLC) which is characterized by verification and validation at each phase of the development process. This approach was selected for its structure, making it appropriate for any type of study that requires rigorous testing and precision, including time series forecasting tasks. The steps in the approach are shown in Figure 3.1.

The following stages were followed:

- **Requirements Analysis:** At this stage, the requirements of olive farmers and agricultural researchers were gathered. The scope of the system, which includes temperature forecasting, growing degree days (GDD) computation, and a user-

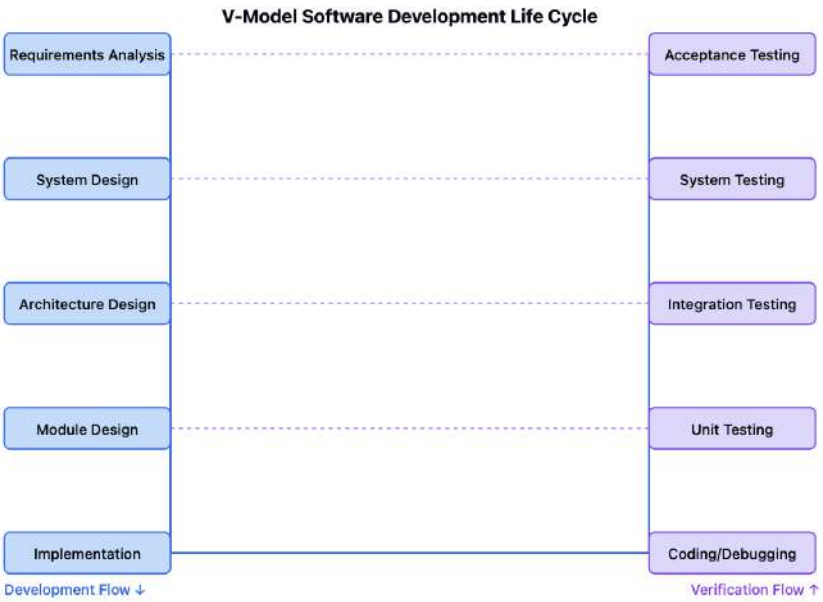

Figure 3.1: V-Model Software Development Life Cycle (SDLC)

friendly web interface was also determined.

- **System Design:** The system architecture was designed, outlining how real-time weather data would be fetched, processed, and stored. At this stage, the API endpoints and database schema were also defined.
- **Architecture Design:** The high-level design of the software system was formulated, including API routes, database interactions, and frontend-backend communication.
- **Module Design:** The implementation was broken down into independent modules: data retrieval, GDD computation, database management, and user interface.
- **Implementation:** The Flask backend was developed for data processing and API handling, while the React frontend was built for visualizing GDD trends and allowing user interaction.
- **Coding and Debugging (Unit Testing)** Each module was tested independently

to verify its correctness. The API endpoints were tested for valid data retrieval and calculation.

- **Integration Testing** The frontend and backend were integrated to test seamless data exchange and visualization.
- **System Testing** The entire system was tested under simulated real-world conditions, validating data accuracy and user interaction flows.
- **Acceptance Testing** The final system was evaluated based on the ability to provide accurate GDD computations and an intuitive user experience.

3.3 Feasibility Study

A feasibility study was performed in order to assess the viability of implementing this system for olive farmers and agricultural researchers. This study evaluates three primary aspects: technical feasibility, operational feasibility, and economic feasibility.

3.3.1 Technical Feasibility

The system was designed to leverage existing technologies such as Python, Flask, SQLite, React.js, and OpenWeatherMap API. The choice of these technologies ensures that:

- The model can handle real-time weather data updates dynamically.
- The web application remains lightweight and accessible across multiple devices.
- The SQLite database provides an efficient way to store and retrieve GDD records for long-term tracking.
- The hybrid model, combining Facebook Prophet and GRU networks, efficiently captures both seasonality and short-term variations in temperature trends.

Additionally, cloud deployment options (e.g., Heroku, AWS Lambda) were considered for future scalability.

3.3.2 Operational Feasibility

In addition, operational feasibility was ensured by making sure that the system is compatible with the requirements of its target users. For the farmers, the system gives real time and historical GDD tracking that helps in making decisions on the crop management and harvesting time.

3.4 Data Collection and Preprocessing

Accurate and reliable data are crucial for developing good forecasting models. The hybrid model is trained using historical meteorological datasets to ensure that temperature trends and seasonal variations are learned properly. The data collection process was also carefully done in order to gather high quality, long term temperature data from different olive-producing regions in the Mediterranean. In addition, some preprocessing techniques were used to clean, normalize and handle missing data to ensure that data consistency for when it is fed to the forecasting model.

3.4.1 Sources

Temperature data were collected from publicly available meteorological datasets, namely the National Centers for Environmental Information (NCEI) [15], spanning multiple decades. Data from key olive-producing regions were extracted, summarized in Table 3.1.

3.4.2 Data Definitions and Types

The dataset consists of daily meteorological records spanning multiple decades, with temperature readings serving as the primary variables for analysis. The key attributes used in this study include:

- *Daily Maximum Temperature (T_{max})* – The highest recorded temperature for the day, crucial for estimating heat accumulation.
- *Daily Minimum Temperature (T_{min})* – The lowest recorded temperature for the day, providing information on the variability of the temperature.

Table 3.1: Summary of temperature datasets for olive-producing regions

Region	**Climate Characteristics**	**Data Coverage Period**
Almería, Spain	Semi-arid climate with high temperature variability	Jan 1968 – Jan 2025
Grosseto, Italy	Classic Mediterranean climate with balanced seasonal patterns	Dec 1944 – Jan 2025
Bursa, Turkey	Influenced by both Mediterranean and continental climates	Jan 1973 – Jan 2025
Kalamata, Greece	Stable seasonal trends characteristic of the Mediterranean region	Jul 1987 – Jan 2025
Larnaca, Cyprus	Stable seasonal trends characteristic of the Mediterranean region	Apr 1976 – Jan 2025

- *Daily Average Temperature (T_{avg})* - Calculated as the mean of T_{max} and T_{min}, and is used as a reference for Growing Degree Days (GDD) calculations.
- *Date Information* - The dates were standardized in ISO 8601 format (YYYY-MM-DD).

The original datasets consisted of other meteorological variables such as precipitation, snow depth, and wind direction, but they were rather incomplete, or had limited data for the selected regions. Hence, only temperature related variables were employed for modeling to guarantee data reliability and completeness. This is in conformity with the study's interest on temperature-induced agricultural forecasting especially for GDD which is primarily a function of thermal accumulation.

3.4.3 Ethical Concerns

Since this research is based on publicly available meteorological data sets, there are no issues of privacy with data collection. However, ethical considerations were made in terms of transparency and fairness of data selection and preprocessing. The study

```python
@app.route('/gdd', methods=['GET'])
def get_gdd():
    location = request.args.get("location", "Larnaca")
    base_temp = request.args.get("base_temp", 10)
    start_date = request.args.get("start_date")

    print(f"Received request: location={location}, base_temp={base_temp}, start_date={start_date}")

    # Validate base temperature
    try:
        base_temp = float(base_temp)
    except ValueError:
        return jsonify({"error": "Base temperature must be a valid number."}), 400

    # Validate start date
    if not start_date:
        return jsonify({"error": "Please specify a planting start date in YYYY-MM-DD format."}), 400

    try:
        start_date = datetime.strptime(start_date, "%Y-%m-%d").date()
    except ValueError:
        return jsonify({"error": "Invalid date format. Use YYYY-MM-DD."}), 400
```

Figure 3.2: The `/gdd` API route, which extracts location, base temperature, and planting date parameters, validates input values, and returns errors for incorrect formats.

does not alter or prejudice the dataset in any way and all the preprocessing steps such as dealing with missing data are well documented in order to ensure replicability.

3.4.4 Software Engineering Standards Applied

To ensure the reliability and maintainability of the system, industry-standard practices were followed:

- **RESTful API Design**: The API endpoints were structured following REST principles to ensure scalability and modularity. Figure 3.2 shows a screenshot of the `/gdd` route, which processes the request by extracting query parameters such as location, base temperature, and planting start date. It also has validation steps to make sure that the base temperature is a valid numerical value and the start date is actually provided in the right format (YYYY-MM-DD). If any validation fails, then a proper error message is returned along with a 400 (Bad Request) status code to make the API more resilient.
- **Separation of Concerns:** The frontend (React) and backend (Flask) were developed as independent components.
- **Database Consistency:** SQLite was chosen for its lightweight nature.

3.4.5 General Overview of TMAX in Cyprus

The maximum daily temperature (TMAX) in Cyprus is a critical variable for understanding the climatic conditions that influence olive cultivation. Figure 3.3 illustrates the yearly TMAX trends for Cyprus over the analyzed time period.

3.4.5.1 Seasonal Trends

The data demonstrates an obvious cyclic pattern, which is characteristic of the Mediterranean climate and directly influences olive cultivation cycles. It is approximately divided into:

- **Summer Months:** TMAX peaks during the summer months, often surpassing $35°C$. This heat plays an important role in olive fruit development and oil synthesis, as documented by Jones et al. (1999) [16].
- **Winter Months:** During winter, TMAX remains relatively mild in many mediterranean areas, typically ranging between $10°C$ and $15°C$, creating ideal conditions for the dormancy phase of olive trees [4].These mild winters, where temperatures rarely drop below freezing, also contribute to why Mediterranean climates are particularly well-suited for olive cultivation. However, some localized variations can occur, and occasional frosts can pose a risk in some regions.

The periodic nature of olive growth and maturation cycles is reflected in these patterns, which mirror the strong seasonality in Cyprus's climate. Temperature seasonal variations have been central to understanding of agricultural productivity as highlighted by Box et al. (2015) [3].

3.4.5.2 Long-Term Observations

Assessing long-term TMAX trends offers insights into climate variability and its impact on olive production. Some of these insights include:

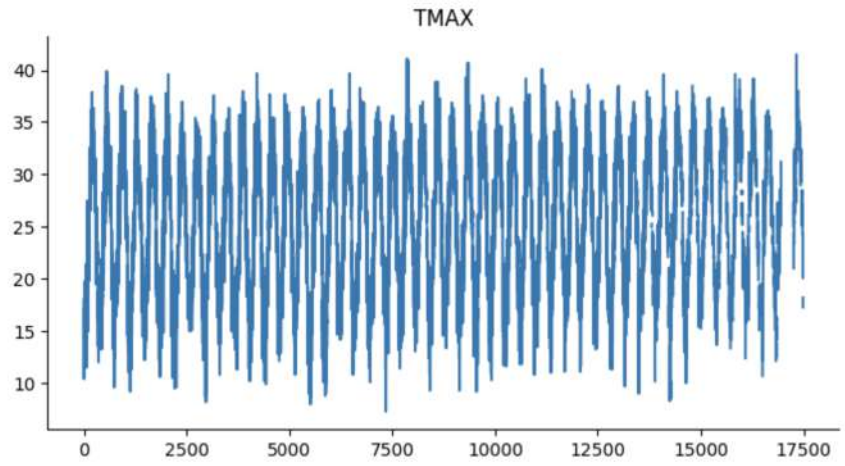

Figure 3.3: Yearly trends of TMAX in Cyprus. The cyclic pattern reflects seasonal variations and long-term trends.

- **Climate Variability:** Changes in TMAX over decades are gradual, indicating a shift in climate patterns that may affect how reliable olive yields are [17].
- **Heatwaves:** Occasional peaks in TMAX result in heat waves which can stress the olive trees, and directly affect the oil quality [4, 14].

Shi et al. [6] emphasize the importance of incorporating such long-term trends into forecasting models to account for extreme weather events.

3.4.5.3 Distribution of Target Variable (TMAX)

The distribution of the target variable (y) is shown in Figure 3.4, which depicts normalized daily maximum temperatures (TMAX) for Larnaca, Cyprus. The normalization scales TMAX values between 0 and 1 for model training.

Key characteristics of the distribution include:

- **Bimodality:** There are two distinct apexes that reflect the seasonal temperature patterns, particularly the peak summer and winter temperatures in olive-growing regions [4].
- **Symmetry:** This approximately symmetric distribution simplifies modeling and

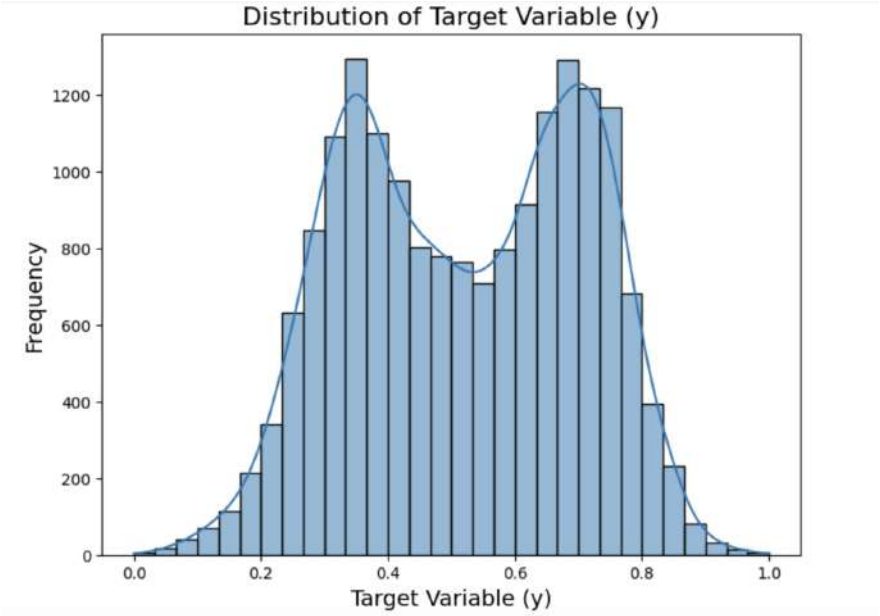

Figure 3.4: Distribution of the normalized target variable (y).

it reduces the need for transformations to achieve normality. [18].

- **Minimal Skewness and Outliers:** Few outliers and negligible skewness contribute to a balanced data set, and pre-processing steps address any anomalies.

3.4.5.4 Significance of Target Variable Distribution

Understanding the target variable's distribution is essential for:

- **Model Learning:** Bimodal distributions guide the model in learning patterns associated with seasonal variations [19].
- **Evaluation Metrics:** Symmetry ensures error metrics, such as MAE and MSE, fairly represent the entire dataset [20].
- **Feature Engineering:** The observed distribution supports the inclusion of seasonal features in the hybrid model [7].

3.4.5.5 Implications for Olive Cultivation

Understanding yearly TMAX trends in Mediterranean regions like Cyprus is necessary for:

- **Crop Management:** To preplan for irrigation requirements during the summer

to avoid heat stress impacts [21].

- **Harvest Planning:** Timing of harvest to obtain maximum yield and product quality during optimum temperature conditions.
- **Modeling Precision:** Incorporating these trends into the hybrid model enhances its ability to capture seasonal and annual TMAX patterns, improving forecast accuracy [11].

3.5 Requirements Analysis

In this section, the functional and non-functional requirements of the system are outlined. The requirements were gathered from the needs of the farmers, researchers and stakeholders who require precise weather forecasting and growing degree day (GDD) tracking to support their decision making processes.

3.5.1 Functional Requirements

The system was designed to fulfill the following core functionalities:

- **Weather Data Retrieval:** The system uses a hybrid forecasting model that combines pre-trained temperature prediction models with real-time data from the OpenWeatherMap API (One Call 3.0 subscription). This ensures that forecasts are dynamically adjusted according to the most recent observed weather conditions for more accurate agricultural decision making.
- **GDD Calculation:** Computes daily and cumulative growing degree days based on user-specified base temperature and planting date.
- **Growth Stage Estimation:** Determines the current growth stage of olive trees based on accumulated GDD values.
- **Web Interface for User Input:** Users can specify their location, planting date, and base temperature through a user-friendly interface.
- **Dynamic GDD Visualization:** Displays accumulated GDD over time in an interactive chart to help farmers track progress.

- **RESTful API for Data Access:** Provides structured API endpoints for retrieving temperature, GDD, and growth stage data.
- **Database Storage:** Stores accumulated GDD values for each location and start date in an SQLite database to ensure continuity.
- **Error Handling and Validation:** Ensures inputs such as base temperature and planting date are correctly formatted and reasonable.

3.5.2 Non-Functional Requirements

The system also adheres to the following software quality attributes to ensure usability and reliability:

- **Performance:** API responses are optimized to provide real-time feedback with minimal latency.
- **Scalability:** The REST API structure ensures easy expansion to accommodate additional features or other crops.
- **Usability:** A simple and intuitive web-based UI is designed to be accessible to non-technical users, such as farmers.
- **Reliability:** The system is designed to function even when OpenWeatherMap data retrieval encounters delays, using cached database values when available.
- **Maintainability:** The modular architecture allows independent updates to the API, database, and UI components.
- **Security:** Input validation prevents incorrect or malicious data from being processed.

3.5.3 Stakeholders and Use Cases

The primary actors of the system include olive farmers who need accurate GDD to use it for irrigation, fertilization and determining the time of harvest. Olive farming climate adaptation is an area of research that could benefit greatly from accurate forecasting models. Software engineers may consider the system to be useful for extending

its application to other agricultural sectors through its modular design and predictive models.

3.5.4 System Constraints

The system relies on the internet connectivity to fetch real-time weather data from an API called OpenWeatherMap, therefore it requires stable access for dynamic updates. The API limitations impact the historical weather data retrieval, which may limit the historical data that can be retrieved for analysis. Although the forecasting model has been optimized for efficiency, its computational requirements may need further refinement for large-scale deployment, especially in the regions with extensive agricultural monitoring needs.

3.5.5 Assumptions

It is expected that users have at least some internet connection to use the web interface and get the latest forecasts. The system is also based on the assumption that OpenWeatherMap API is working and can be used for further data updates in the future. Furthermore, it is assumed that the predefined GDD thresholds for olive tree growth stages are valid for Mediterranean climates and may require some modification for other agricultural productions.

3.6 System Design and Modelling

The system is developed using a V-Model Software Development Life Cycle (SDLC) approach as shown in Figure 3.1, where the validation is performed at each development stage. The developed application provides real time access to weather forecasting and Growing Degree Days (GDD) tracking to help olive farmers in deciding the growth stages of their crop.

3.6.1 Conceptual Design of the Proposed Solution

The system is structured into three main components: the data acquisition layer, the processing layer, and the presentation layer. These components work together to ensure

accurate GDD calculations and dynamic updates based on real-time weather data. An overview of how these layers interact is provided in Figure 3.5.

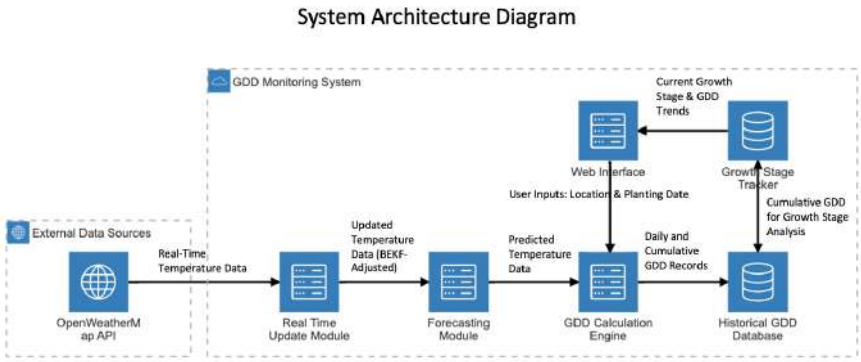

Figure 3.5: System architecture of the GDD monitoring and forecasting system. The system integrates real-time weather data from OpenWeatherMap, processes it using a hybrid forecasting model, and presents GDD trends and growth stages via a web interface.

The data acquisition layer retrieves real-time and historical temperature data using OpenWeatherMap's One Call API 3.0. This data is passed to the processing layer, where the hybrid forecasting model combines Bayesian Ensemble Kalman Filter (BEKF)updates with pre-trained GRU-based predictions.The GDD Calculation Engine computes daily and cumulative GDD values, which are stored in the Historical GDD Database. The presentation layer is responsible for displaying this information to the user through a web-based interface. Farmers can enter their planting date and location to receive personalized growth stage updates and track cumulative GDD over time.

3.6.1.1 Data acquisition layer

This layer uses the OpenWeatherMap's One Call API 3.0 to fetch real-time and historical temperature data and update them dynamically to improve the predictions. Furthermore, it includes a trained hybrid forecasting model that combines statistical and deep learning approaches to extend the forecasts for olive-growing areas.

3.6.1.2 Processing layer

The processing layer calculates daily values of GDD from retrieved and predicted temperature data and monitors growth accumulation over time. It adaptively updates plant growth stages through the use of real time weather observations and the model predictions to enhance the decision-making accuracy.

3.6.1.3 Presentation layer

The system's presentation layer is implemented through a web based interface where farmers can enter their planting date, see historical and forecasted GDD trends, and learn about the plant growth stages.

3.6.2 UML diagrams

UML diagrams were designed to visualize and better understand the system's architecture and functionality. These diagrams represent system components and their interaction in a structured manner from a structural and behavioral perspective.

3.6.2.1 Behavioral UML Diagrams

In the use case diagram illustrated in Figure 3.6, the interactions between the users and the system are depicted. The diagram highlights key functionalities such as entering planting dates, fetching weather data, calculating GDD and viewing plant growth insights. Farmers are able to use the web interface to enter the relevant information and the system is able to communicate with the OpenWeatherMap API and the database to process and present useful forecasts.

The Sequence Diagram in Figure 3.7 shows the sequence of messages exchanged between the frontend, backend and the external weather API. It shows how user requests made, for instance for GDD, are sent to the backend, which in turn requests the temperature data, calculates the GDD, updates the database and then presents the output to the user. This diagram is useful in identifying the sequence of events and system

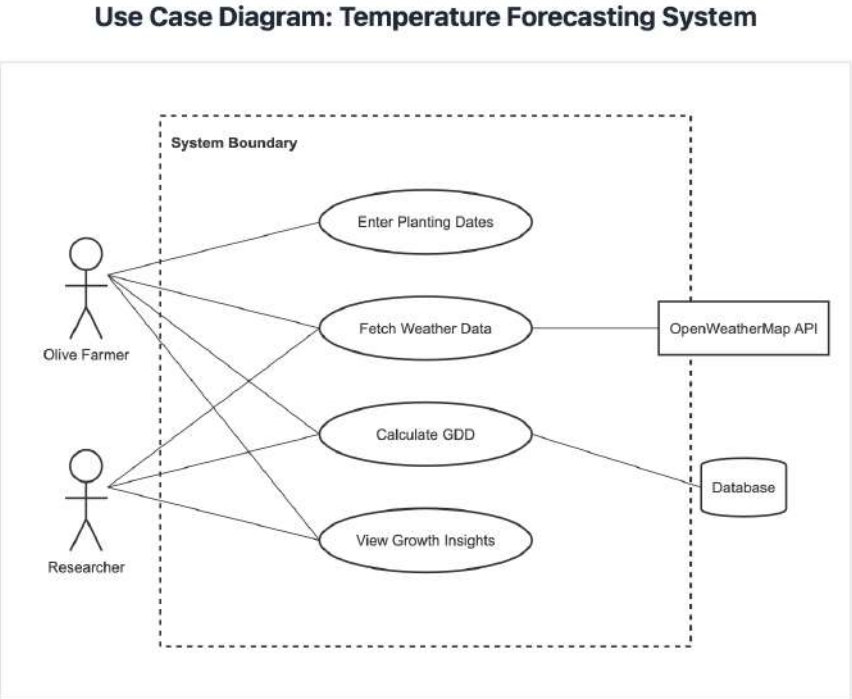

Figure 3.6: Use case diagram illustrating the main interactions between users (olive farmers and researchers) and the temperature forecasting system. The diagram shows core functionalities including data input, weather data retrieval, GDD calculations, and growth insight visualization.

interactions in the logical flow of operations.

The Activity Diagram shown in Figure 3.8 gives a more comprehensive view of the GDD computation workflow. It illustrates the process from start to finish, which includes user input check, weather data collection, forecast using the hybrid model, tracking of cumulative GDD and presentation of the report visualization.

3.6.2.2 Structural UML Diagrams

The Structural UML diagrams give a clear representation of the system's architecture and show the system's main components, relationships, and dependencies.

From the component diagram in Figure 3.9, we get an overall view of the major software modules and their interaction. It shows the frontend web application, which al-

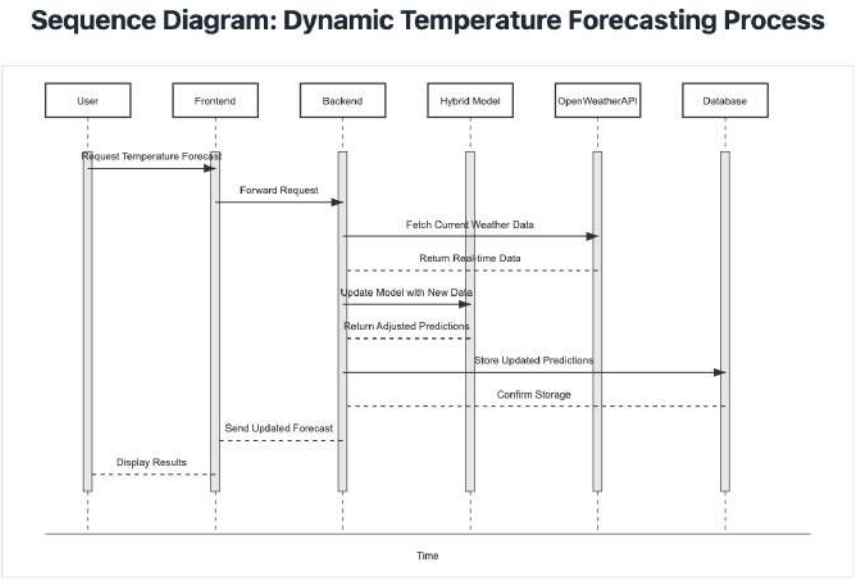

Figure 3.7: Sequence diagram depicting the dynamic interaction flow between system components for temperature forecasting. The diagram illustrates how the hybrid model is continuously updated with real-time OpenWeather data to maintain prediction accuracy.

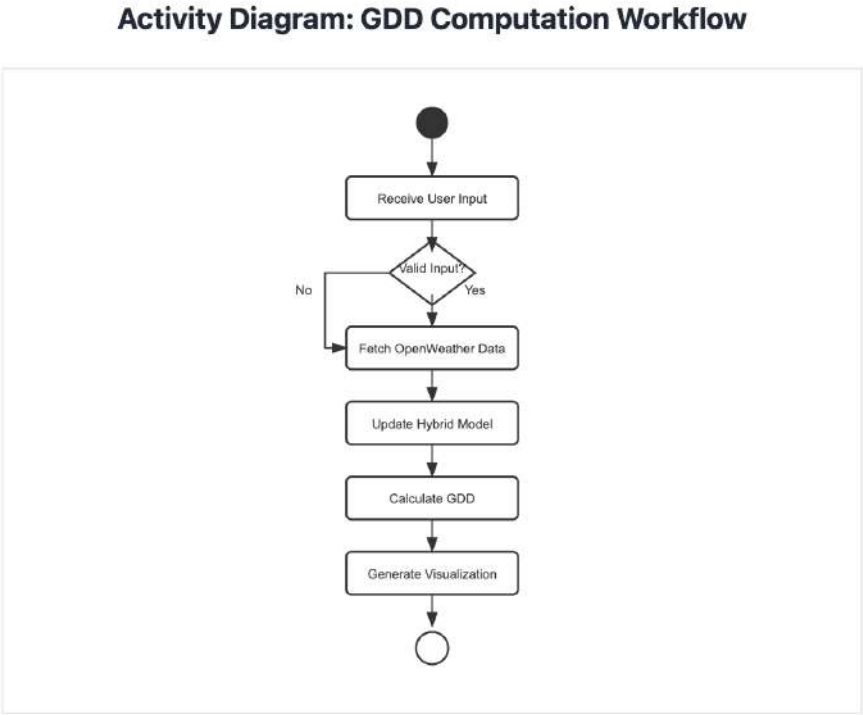

Figure 3.8: Activity Diagram showing the complete workflow of the GDD computation process. The diagram traces the path from initial user input validation through weather data collection, hybrid model updates, and final visualization generation.

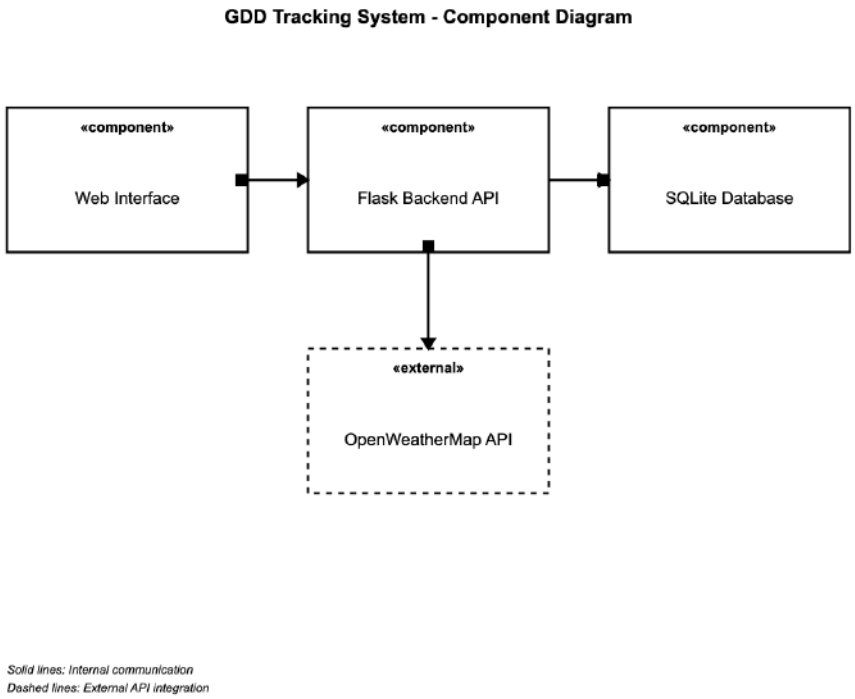

Figure 3.9: Component diagram

lows users to input data and visualize results, the Flask backend API, which processes requests and communicates with other components, the SQLite database, which stores GDD values for tracking plant growth, and the OpenWeatherMap API, which provides real-time and historical temperature data. This diagram helps convey how different parts of the system work together in a modular and maintainable way.

The class diagram in Figure 3.10 focuses on the internal data structures and their relationships. The key classes include:

- *WeatherDataFetcher* - Retrieves temperature data from OpenWeatherMap.
- *GDDCalculator* - Computes daily and cumulative GDD values.
- *DatabaseManager* - Handles data storage and retrieval in SQLite.
- *UserInterface* - Connects with the backend to present relevant information in a way that is easy for users to understand.

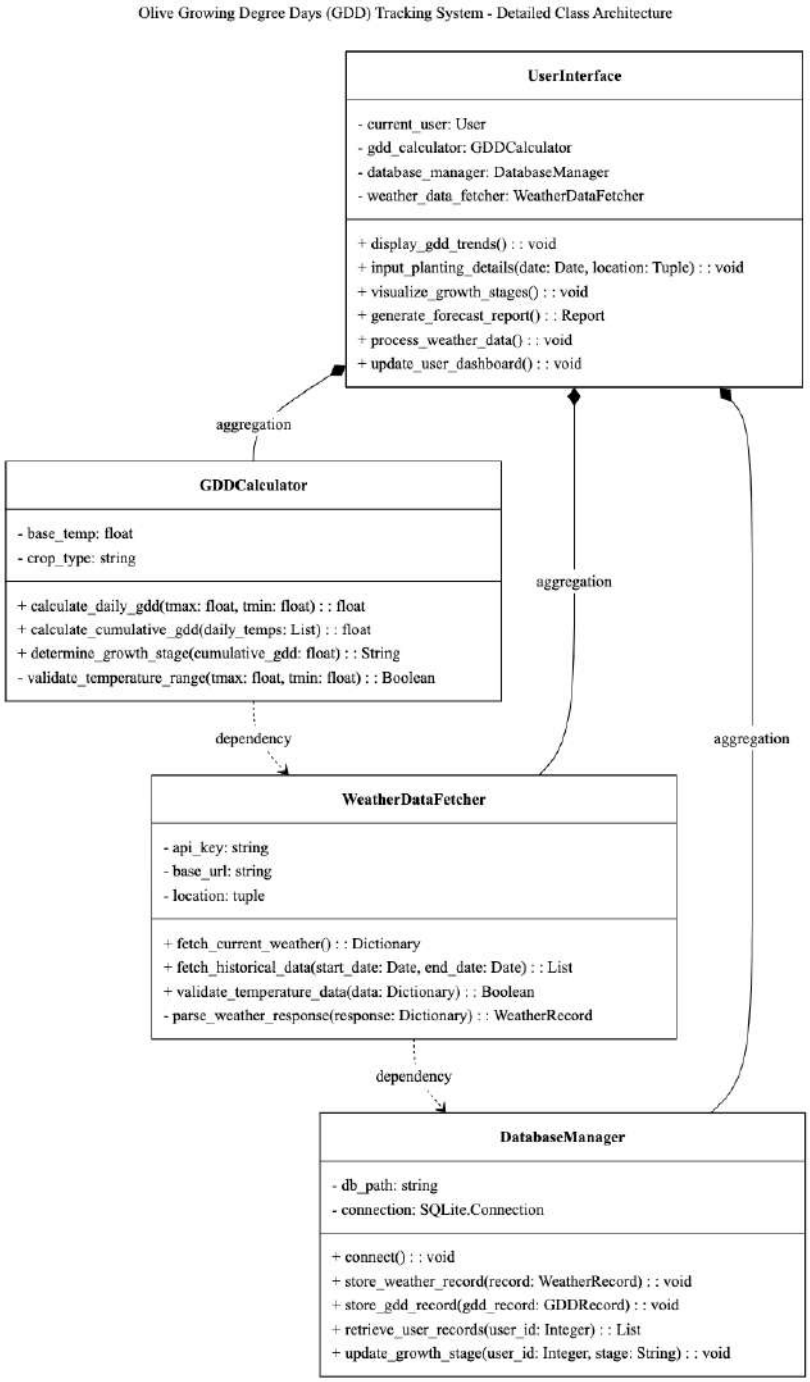

Figure 3.10: Class diagram

3.6.3 Entity-Relationship (E-R) Diagram

The E-R diagram in Figure 3.11 represents the relationships between key entities in the system, particularly how users, weather data, and GDD records interact. The primary entities in the system are:

- *UserInput* – Stores user-provided details, including location, planting date, and base temperature.
- *WeatherRecord* – Contains historical temperature data retrieved from OpenWeatherMap.
- *GDDRecord* – Computes and tracks daily and cumulative GDD values based on weather data.

These entities ensure that GDD calculations are accurately performed based on real-time and historical temperature data while keeping the system simple and efficient.

3.6.4 Normalized Relational Tables

To improve data integrity and minimize redundancy, the database follows third normal form (3NF). The main tables are shown in Table 3.2.

Table 3.2: Normalized database tables for GDD tracking

Table Name	**Attributes**	**Keys**
UserInput	user_id, location, planting_date, base_temperature	user_id (PK)
WeatherRecords	record_id, user_id, date, tmax, tmin	record_id (PK), user_id (FK)
GDDRecords	gdd_id, user_id, date, daily_gdd, cumulative_gdd	gdd_id (PK), user_id (FK)
GrowthStage	stage_id, stage_name, min_gdd_threshold, max_gdd_threshold	stage_id (PK)

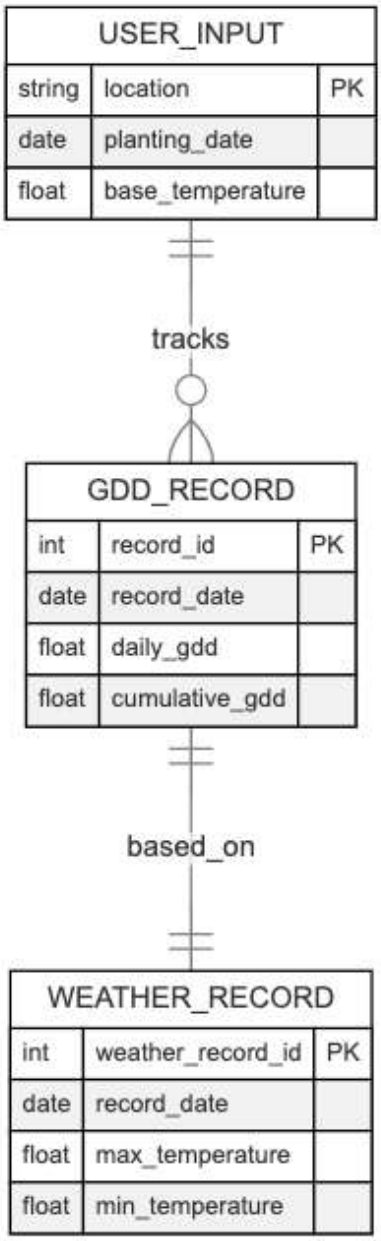

Figure 3.11: Entity-Relationship diagram of the GDD monitoring system.

3.6.5 Physical DB Tables

The database is implemented using SQLite, with the database file named `gdd_data.db`. It can be observed in the general project directory showed in Figure 3.12. The key tables include:

- *UserInput* – stores user-provided location, planting date, and base temperature for GDD calculations.
- *WeatherRecords* – archives historical temperature data retrieved from OpenWeatherMap.
- *GDDRecords* – tracks daily and cumulative GDD values for each planting cycle.
- *GrowthStages* – defines the thresholds for different growth phases based on accumulated GDD.

These database structures ensure efficient storage and retrieval of weather data while supporting accurate GDD tracking and plant growth stage monitoring.

3.7 Implementation

This section describes the tools and technologies used in developing the system and provides an in-depth discussion of the algorithms implemented throughout the entire pipeline, from weather forecasting to Growing Degree Days (GDD) calculation.

3.7.1 Development Tools

The system was developed via the use of various programming languages, frameworks and libraries that enable the system to perform tasks related to machine learning, development of web applications and database management. The primary tools used are:

- *Python* - Used for data preprocessing, model training, and backend API development.
- *Flask* - A lightweight web framework for building the RESTful API.

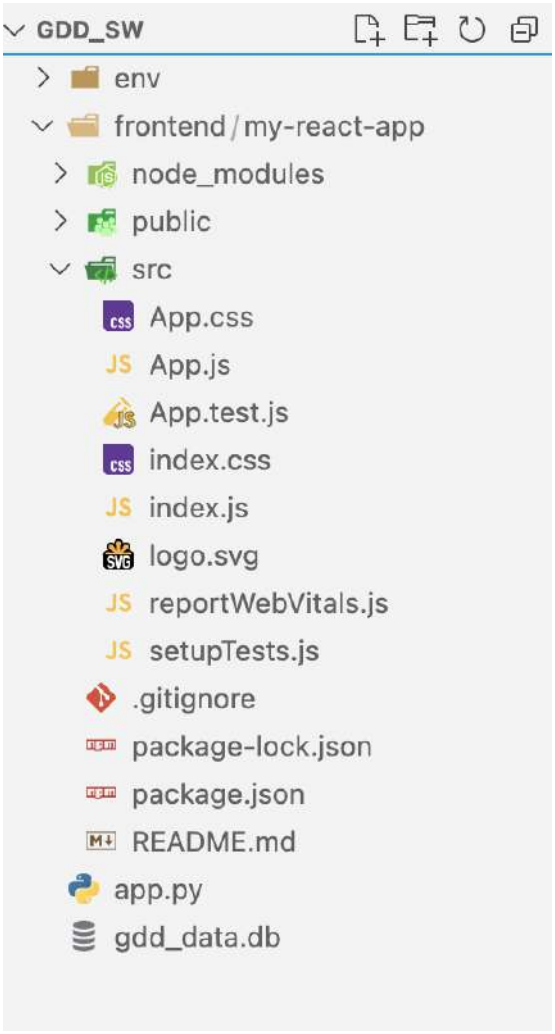

Figure 3.12: Project directory structure on visual studio dode, which shows key components of the system including the database file, `gdd_data.db`, `app.py` for the flask backend logic, and `App.js` for the frontend implementation.

- *SQLite* - A simple yet efficient database management system for tracking GDD values.
- *React.js* - A frontend JavaScript framework for building the interactive web interface.
- *TensorFlow/Keras* - Used for implementing the GRU-based temperature prediction model.
- *Facebook Prophet* - Employed for trend decomposition and residual analysis.
- *OpenWeatherMap API* - Provides real-time and historical weather data.
- *Plotly.js* - A JavaScript graphing library used to visualize daily GDD trends.
- *Google Colab* - The environment used to preprocess our datasets and train the model.

3.7.2 Discussion on Algorithms Used

The system combines multiple forecasting techniques, leveraging statistical and deep learning-based methods, as well as an advanced state estimation technique for dynamic updating.

3.7.2.1 Time Series Decomposition

Time series decomposition is a powerful analytical tool that separates a time series into its constituent components:

- **Trend:** Captures long-term changes in the data, such as gradual increases or decreases in temperature over decades.
- **Seasonality:** Reflects periodic patterns, such as annual temperature fluctuations driven by climatic cycles.
- **Residuals:** Represents irregular, unpredictable variations not accounted for by trend or seasonality.

3.7.2.2 Facebook Prophet for Decomposition

Meta's Prophet is a forecasting model that is designed specifically for time series data. We used it to decompose our temperature data into three components; Long-term trends, seasonal patterns, and residuals. Then, the deep learning model was fed these components to be able to capture the non-linear patterns which may not be captured by traditional statistical models. We chose Prophet to analyze our temperature dataset because of its ability to handle missing data gaps. It models the data using an additive decomposition approach:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t, \tag{3.1}$$

where:

- $g(t)$: Trend component, modeled as a piecewise linear or logistic growth curve.
- $s(t)$: The Seasonality component, which is represented for yearly cycles by Fourier series in mathematics.
- $h(t)$: Holiday effects, which can be customized for specific time periods.
- ε_t: Residual noise.

Prophet is well suited for handling missing data and outliers. It also allows users to input custom seasonalities as needed. These features make it a good tool for temperature time series decomposition.

3.7.2.3 Decomposition Process

Prophet was applied to TMAX, the daily maximum temperature data to extract it's trend, seasonality, and residual components. Figure 3.13 shows the visualization of the trend for the entire Laranaca dataset spanning over 40 decades and Figure 3.14 shows the decomposed component of the overall trend and yearly seasonality. This decomposition served two main purposes:

- **Improved Interpretability:** By breaking down the data into its components,

visible views on long-term climatic trends and on recurring seasonal patterns are enabled, thus adding value to what domain experts can obtain from the data for decision-making purposes.

- **Enhanced Model Accuracy:** Modeling trend and seasonality explicitly allows the provides the GRU network with extra features to work with, improving its ability to capture short-term nonlinear dynamics.

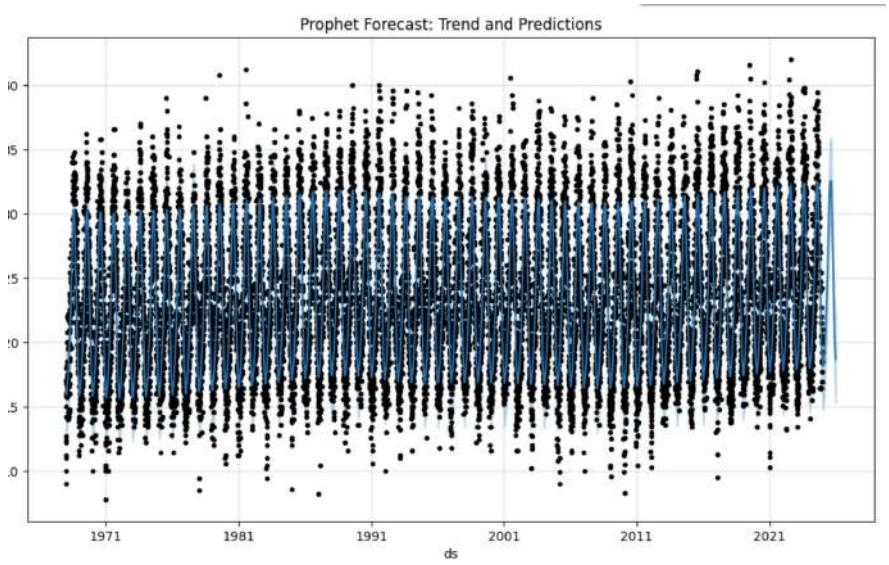

Figure 3.13: Visualization of time series decomposition performed using Facebook Prophet. The plot highlights observed data points (black dots), modeled trend, seasonality, and residual patterns across the historical dataset for olive-producing regions.

3.7.2.4 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a special kind of neural network built to work with sequential data. Unlike traditional feedforward networks that process inputs independently, RNNs incorporate feedback mechanisms that allow them to keep information from previous steps. This feedback mechanism makes it great for tasks where sequence is important, like speech recognition, language modeling, and time series prediction.

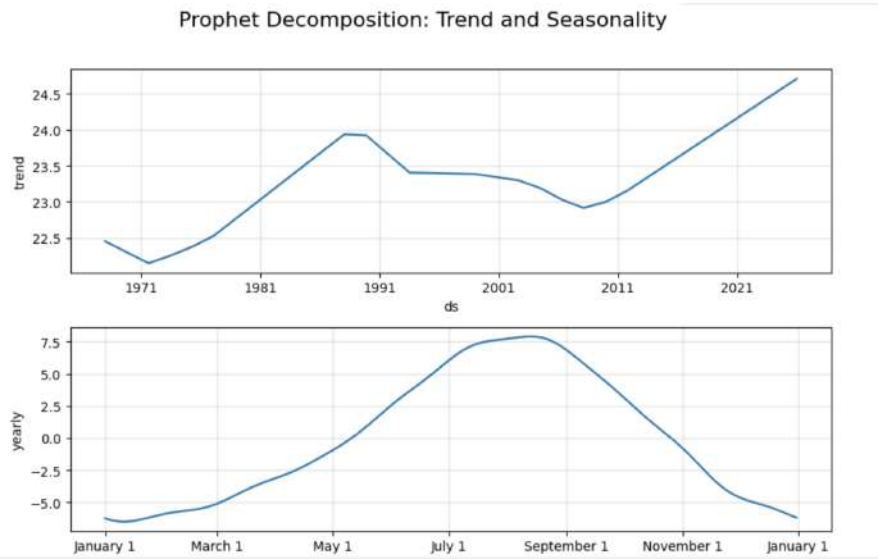

Figure 3.14: Decomposed components of the time series: (Top) Long-term trend reflecting gradual changes in TMAX over decades. (Bottom) Yearly seasonality showing periodic fluctuations typical of the Mediterranean climate.

The defining feature of an RNN is its ability to retain a memory of prior inputs through a hidden state, which is updated at each time step based on the current input and the previous hidden state. This hidden state helps RNNs model temporal dependencies, making them particularly effective for applications such as weather prediction, speech processing, and time-series analysis. The hidden state h_t is computed as follows:

$$h_t = \tanh(W_h \cdot h_{t-1} + W_x \cdot x_t + b_h) \tag{3.2}$$

where:

- x_t: Input at time step t,
- h_{t-1}: Hidden state from previous step,
- W_h, W_x: Weight matrices for both the hidden state and input,
- b_h: Bias term,
- tanh: Activation function for adding non-linearity.

The output at time step t, written as y_t, is gotten using:

$$y_t = W_y \cdot h_t + b_y \tag{3.3}$$

where W_y and b_y are the weight matrix and bias for the output layer.

However, RNNs have their limitations; they are challenged by issues such as the *vanishing gradient problem* where gradients vanish in the course of backpropagation and thus impede the network's ability to capture temporal dependencies of any length. To this end, variants of RNNs, including Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) have been proposed to tackle these issues by including features that aid in the control of information flow.

3.7.2.5 Simple RNN Architecture

A simple RNN processes data sequentially, updating its hidden state and generating an output at each time step. Figure 3.15 illustrates the flow of information in a basic RNN.

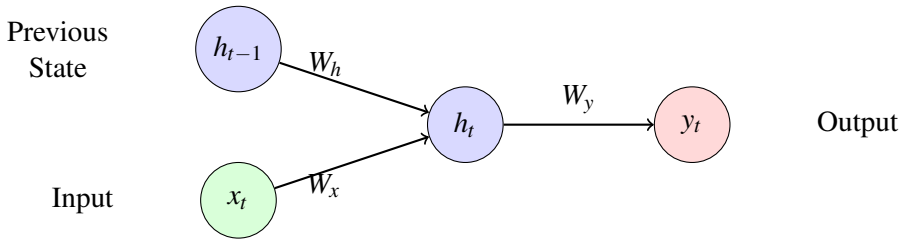

Figure 3.15: Structure of a simple RNN.

The equations governing this architecture are:

- Hidden state update:

$$h_t = \tanh(W_h \cdot h_{t-1} + W_x \cdot x_t + b_h)$$

- Output computation:

$$y_t = W_y \cdot h_t + b_y$$

The hidden state h_t acts as a bridge between past and present information, helping the

network recognize patterns over time. However, when dealing with long sequences, this reliance on h_t can cause issues like gradient decay, which makes it harder for the network to learn effectively.

3.7.2.6 Historical Context and Limitation of RNNs

RNNs were first introduced by Elman [22] to see how they could be applied to temporal data. However, early implementations had computational complexity restrictions and the vanishing gradient problem identified by Hochreiter [18].

Building on earlier work, Pascanu et al. [23] further explored the challenges faced by recurrent neural networks (RNNs). This led to significant developments, such as Echo State Networks [24] and the sequence-to-sequence framework introduced by Sutskever et al. [25]. Because of these advances, new possibilities were opened up for using RNNs, but they also revealed challenges that needed solving in order to make these networks truly effective.

3.7.2.7 Long Short-Term Memory (LSTM)

LSTMs were developed to address RNN limitations by introducing memory cells and gating mechanisms to regulate information flow. The three primary gates are:

- **Forget Gate:** Controls which parts of the previous state to discard.
- **Input Gate:** Manages the addition of new information to the memory cell.
- **Output Gate:** Determines which information from the cell state contributes to the hidden state.

Figure 3.16 shows the architecture of an LSTM network.

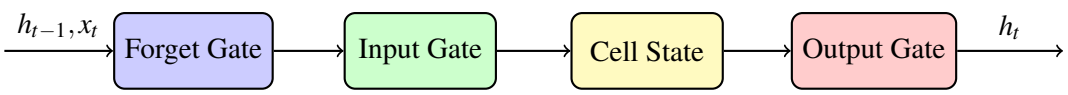

Figure 3.16: Architecture of an LSTM network.

The mathematical formulation for LSTM operations includes:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad \text{(Forget Gate)} \tag{3.4}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{(Input Gate)} \tag{3.5}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad \text{(Candidate State)} \tag{3.6}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad \text{(Cell Update)} \tag{3.7}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad \text{(Output Gate)} \tag{3.8}$$

$$h_t = o_t * \tanh(C_t) \quad \text{(Hidden State)} \tag{3.9}$$

3.7.2.8 Gated Recurrent Units (GRUs)

GRUs simplify LSTM architecture by combining the forget and input gates into a single update gate, while maintaining comparable performance with fewer parameters. GRUs are particularly suitable for applications with computational constraints due to their efficiency.

GRU operations include:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \quad \text{(Update Gate)} \tag{3.10}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \quad \text{(Reset Gate)} \tag{3.11}$$

$$\tilde{h}_t = \tanh(W_h \cdot [r_t * h_{t-1}, x_t] + b_h) \quad \text{(Candidate State)} \tag{3.12}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \quad \text{(Final State)} \tag{3.13}$$

3.7.2.9 GRU Architecture

Introduced by Cho et al. [19], the GRU addresses the vanishing gradient problem that plagues traditional RNNs while maintaining a simpler structure compared to the LSTM [18].

As illustrated in Figure 3.17, the GRU employs two primary gating mechanisms:

1. The update gate (z_t) determines how much of the previous hidden state should be retained. This mechanism is analogous to the forget and input gates in LSTMs but combines their functionality into a single gate, making the model more computationally efficient [20].
2. The reset gate (r_t) controls how much of the previous hidden state should influence the candidate hidden state. When closed (values near zero), it allows the unit to forget past information and effectively 'reset" its state.

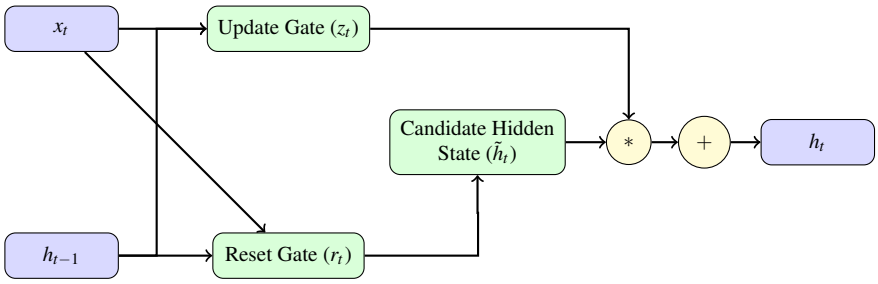

Figure 3.17: Architecture of a Gated Recurrent Unit (GRU).

The candidate hidden state ($\tilde{h}_t$) proposes a new hidden state value, which is then weighted by the update gate to produce the final hidden state (h_t). This architecture has shown comparable performance to LSTMs on various sequence modeling tasks while being more parameter-efficient [20].

3.7.2.10 Bayesian Ensemble Kalman Filter (BEKF) for Dynamic Updating

The Bayesian Ensemble Kalman Filter (BEKF) is a powerful way to refine predictions by combining ideas from Bayesian inference and the Ensemble Kalman Filter (EnKF). The EnKF is commonly used for large-scale problems, especially in geophysical models, because it tracks changes over time using a set of simulations. This allows it to estimate errors and continuously update predictions as new data comes in.

By adding Bayesian methods to the EnKF, the model becomes better at handling uncer-

tainty and working with complex, non-Gaussian data. One example of this application is a study in Mathematical Geosciences, where a Bayesian approach that uses Gaussian mixture models was proposed [26]. This improves upon the traditional EnKF, which assumes that data follows a normal distribution. By allowing more flexibility, it becomes especially useful for problems like subsurface modeling, where data can have multiple different distributions. Its implementation is shown in Figure 3.18, where an ensemble of GRU models is updated dynamically using real-time temperature data from OpenWeatherMap.

The process starts with a set of GRU model states which are initialized to represent different plausible system conditions. Each state is propagated forward based on learned GRU dynamics to produce independent forecasts. When new temperature data from OpenWeatherMap becomes available, the ensemble members are updated via Kalman filtering, so that their predictions are consistent with observed values. The final output is obtained by computing the ensemble mean, thus performing a dynamically refined forecast. The update step is based on the Kalman gain equation:

$$x_t = x_t^- + K_t(y_t - Hx_t^-) \tag{3.14}$$

where x_t^- is the prior state estimate, K_t is the Kalman gain, y_t represents the observed temperature data, and H is the observation model. This approach allows the model to remain responsive to sudden weather changes while maintaining long-term predictive stability.

3.7.2.11 RESTful API for Data Processing

The system uses a RESTful API architecture to connect the frontend and backend components. Through dedicated endpoints, users can access both real-time and historical temperature measurements, perform growing degree day calculations, and update

```python
class BayesianEnsembleKalmanFilter:
    def __init__(self, ensemble_size, model_class, input_shape, initial_params, process_noise, observation_noise):
        self.ensemble_size = ensemble_size
        self.models = [model_class(input_shape) for _ in range(ensemble_size)]
        for model in self.models:
            model.set_weights(initial_params)
        self.process_noise = process_noise
        self.observation_noise = observation_noise

    def update(self, X_seq, observed_value):
        #Updates the ensemble models using OpenWeatherMap real-time data
        ensemble_predictions = np.array([model.predict(X_seq) for model in self.models])

        # Kalman update: Adjusts based on OpenWeatherMap's observed TMAX
        kalman_gain = self.process_noise / (self.process_noise + self.observation_noise)
        updated_predictions = ensemble_predictions + kalman_gain * (observed_value - ensemble_predictions)

        return updated_predictions
```

Figure 3.18: Implementation of the Bayesian Ensemble Kalman Filter (BEKF) for dynamic updating

cumulative values in the database. This modular design makes it easy to add more functionalities to the system for future updates.

3.7.2.12 Visualization with React and Plotly.js

Plotly.js was used to create interactive charts that help visualize GDD trends clearly. Farmers can enter their planting date and view daily GDD accumulation trends along with an explanation of their current growth stage.

3.7.3 Preprocessing Steps

To prepare the data for training, the following preprocessing steps were implemented:

- **Handling Missing Values:** Forward and backward filling addressed minor gaps, while mean imputation helped with addressing longer gaps.
- **Outlier Treatment:** Outliers identified using the interquartile range (IQR) method were replaced with interpolated values.
- **Feature Standardization:** Variables such as TMAX and Growing Degree Days (GDD) were normalized to zero mean and unit variance for consistent input.
- **Growing Degree Days (GDD):** Calculated to capture heat accumulation which is important across olive various cultivation stages.
- **Decomposition:** Facebook Prophet was used to separate the data into trend, seasonal, and residual components.

3.7.4 Growing Degree Days (GDD)

GDD is a measure of the accumulation of heat over time and is used as a critical metric for temperature sensitive crops such as olives. It is based on the assumption that plant growth is directly proportional to the accumulation of heat above some base temperature. For olives, the base temperature typically ranges between 7°C and 12°C, depending on the particular variety and growth stage.

$$\text{GDD} = \max(0, \text{TMAX} - \text{Base Temperature}), \tag{3.15}$$

where:

- **TMAX**: Daily maximum temperature.
- **Base Temperature**: Threshold temperature below which growth halts, typically $10°C$ for olives.

Cumulative GDD over a season is calculated as:

$$\text{Cumulative GDD} = \sum(\text{TMAX} - \text{Base Temperature}), \tag{3.16}$$

excluding negative values to ensure accuracy.

3.7.4.1 Relevance of GDD in Olive Cultivation

Growing degree days (GDD) are a vital metric for tracking and predicting olive tree development during the season. The lifecycle of an olive is shown to have six stages in Figure 3.19, each stage having certain GDD accumulations. The developmental stages can be quite different when observed in different olive varieties and are planted at different times owing to genetic diversity. For instance, in Figure 3.20, it is possible to see that several cultivars are in different phenological stages, although they were observed at the same time. This variation is attributed to two factors: genetic variations that affect the heat requirement thresholds and the developmental timing, and staggered planting dates that lead to different GDD accumulation periods. The

way different crop varieties develop at their own pace highlights how crucial it is to have reliable temperature readings and growing degree day calculations for informed decision making. The GDD accumulation stages for olive trees are:

- **Bud Development (0-100 GDD):** Initial stage where vegetative and flower buds form, marking the tree's exit from winter dormancy.
- **Flowering (100-350 GDD):** Critical period for bloom development and pollination, directly impacting potential yield.
- **Fruit Set (350-700 GDD):** Early fruit development phase where successful pollination leads to initial olive formation.
- **Pit Hardening (700-1200 GDD):** Characterized by stone formation and hardening, a key developmental milestone.
- **Oil Accumulation (1200-1800 GDD):** Essential phase for oil synthesis and accumulation, significantly affecting final oil content and quality.
- **Maturation (1800-2200 GDD):** Final ripening stage marked by color changes and optimal oil concentration.

3.7.5 Dataset Partitioning

The processed temperature data were divided as follows:

- **Training Set:** 60% was used for model training.
- **Validation Set:** 20% was used for cross-validation.
- **Testing Set:** The remaining 20% was used to evaluate the model's performance.

3.7.6 Flow of Information in the Hybrid Model

This general workflow is highlighted in Figure 3.21. Through Facebook Prophet, the input data undergoes preprocessing and decomposition.They are then passed to the GRU network along with residuals and other engineered features. This multi-stage process is effective in modeling both linear and nonlinear dependencies, and it comes up with very accurate temperature forecasts.

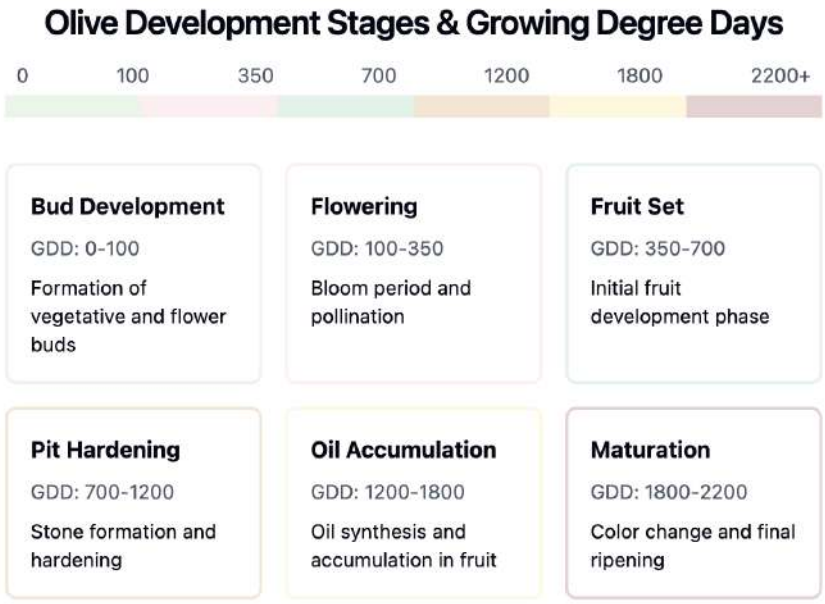

Figure 3.19: Olive tree development stages with corresponding GDD requirements. The timeline visualization shows the progressive heat accumulation needs throughout the growing season, from initial bud development to final maturation.

Figure 3.20: Different varieties of olives in different phenological stages for the same date. Adapted from Cultifort (2025) [1].

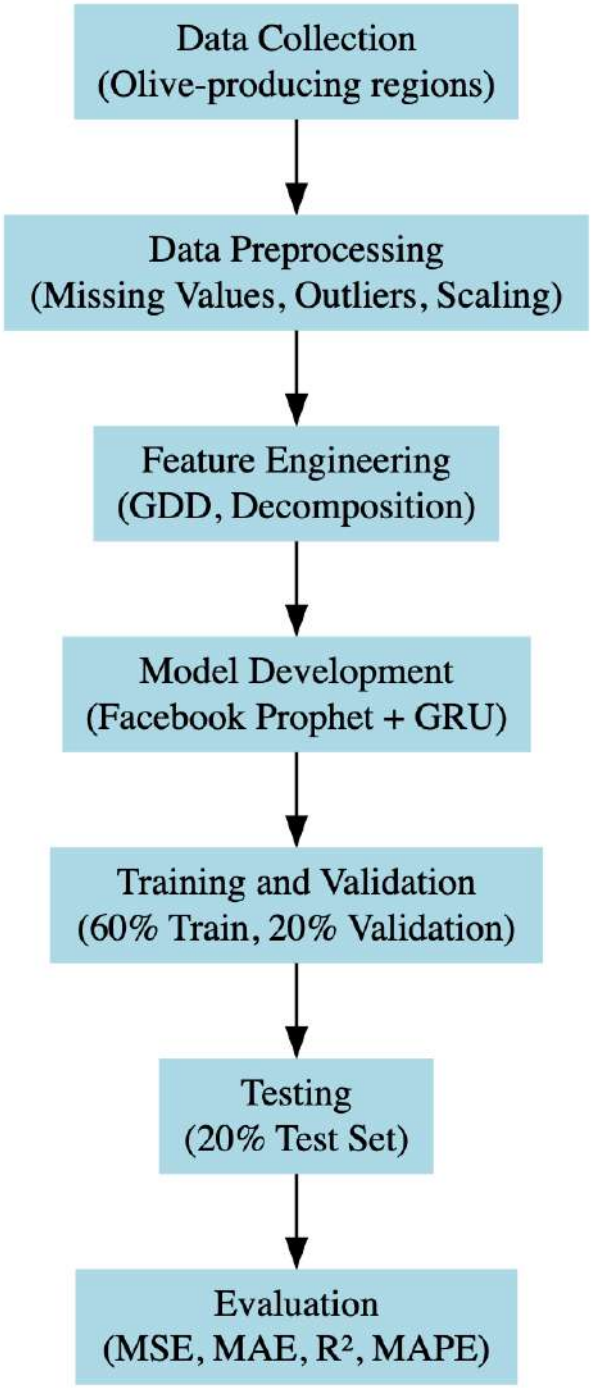

Figure 3.21: Flowchart of the training and evaluation process.

3.7.7 Hybrid Model Development

This paper presents a hybrid model that combines the advantages of Facebook Prophet and GRU. The pipeline is for both linear and nonlinear components in temperature forecasting and combines the unique capabilities of each model. It's two main components are:

1. **Facebook Prophet:** Handles the decomposition of the time series into trend, seasonality, and residual components, effectively modeling long-term and seasonal variations.
2. **GRU Network:** Processes the residuals from Prophet to capture nonlinear dependencies and short-term variations that are difficult for purely statistical methods to address.

The following steps summarize the hybrid model pipeline:

- **Data Preprocessing:** Includes handling missing values, scaling features, and calculating Growing Degree Days (GDD).
- **Decomposition:** Facebook Prophet decomposes the time series into its trend, seasonality, and residual components.
- **GRU Input:** Residuals from Prophet, along with additional features such as GDD, are fed into the GRU network for final predictions.
- **Evaluation:** The hybrid model's performance is compared with baseline models, including SARIMA, stand-alone GRU, and LSTM.

3.7.8 Advantages of the Hybrid Approach

The proposed hybrid model offers several advantages:

- **Scalability:** The GRU network is computationally efficient, enabling large-scale applications.
- **Interpretability:** Facebook Prophet's decomposition provides interpretable out-

puts for trend and seasonality, which are valuable for domain experts.

- **Robustness:** By combining statistical and deep learning methods, the hybrid model effectively handles both linear and nonlinear dynamics in the data.

3.8 Training and Evaluation

To ensure convergence and optimal performance, a carefully tuned set of hyperparameters was used to train the GRU model. The following settings were used during the training process:

- **Learning Rate:** 0.001 to balance convergence speed and ability to escape local minima.
- **Batch Size:** 32 which enables efficient gradient updates and computational efficiency.
- **Sequence Length:** 30 time steps to capture temporal dependencies within a one month window.
- **Epochs:** 100 to allow the model to learn complex patterns without over fitting.
- **Optimizer:** Adam optimizer was chosen because of its adaptive learning rate and momentum which enables faster convergence.
- **Loss Function:** The primary loss function used was Mean Squared Error (MSE) because it penalizes large prediction errors more heavily.

The model was implemented in Python with TensorFlow/Keras and trained on GPU environment for accelerated computations. Early stopping was employed to monitor the validation loss and prevent over fitting with patience of 10 epochs.

3.8.1 Evaluation Metrics

Four choosen metrics were utilized to comprehensively assess the performance of the model:

- **Mean Squared Error (MSE):** This measures the average squared error between

predictions and actual values. Lower MSE indicates better performance.

- **Mean Absolute Error (MAE):** Indicates the average magnitude of prediction errors, providing an interpretable measure of error magnitude.
- **R-Squared (R^2):** This score reflects the proportion of variance which is explained by the model, with values closer to 1 indicating better performance.
- **Mean Absolute Percentage Error (MAPE):** Expresses prediction errors as percentages, providing a normalized measure of accuracy.

3.8.2 Evaluation Process

The hybrid model is evaluated on a testing dataset (20% of the total data), and its performance was compared to baseline models, including:

- Seasonal Autoregressive Integrated Moving Average (SARIMA)
- Long Short-Term Memory (LSTM)
- Standalone GRU
- Simple Moving Average (SMA)

The study raised some key points, including how the hybrid model identified both the overall trends and sudden random fluctuations in the data. Although the gains are modest compared to the baselines, the model's strength in all the metrics considered makes it a good recommendation for use as a reference model.

3.9 Dynamic Forecasting

Dynamic forecasting is implemented to adapt the hybrid model for real-time predictions by integrating current weather data. This capability enables continuous temperature forecasting for olive-producing regions.

3.9.1 Real-Time Data Retrieval

Weather data is retrieved using the OpenWeatherMap One Call API 3.0 [27]. The API provided daily maximum temperatures ($T_{\max}$) for Larnaca, Cyprus (34.9177°N,

33.6250°E). The retrieval process included:

1. **API Integration:** Python requests to the OpenWeatherMap API endpoint with specified geographic coordinates
2. **Data Processing:** Direct retrieval of temperature data in Celsius units
3. **Error Handling:** Robust error management for API responses and data validation

3.9.2 Feature Updates

The retrieved $T_{\max}$ values underwent preprocessing aligned with the training pipeline:

- MinMax scaling for temperature data and standardization for derived features
- Generation of Growing Degree Days (GDD) values using base temperature of 10°C

3.9.3 Model Performance

The hybrid GRU model was evaluated on a 14-day forecast period (January 1-14, 2025), producing the following metrics:

- Mean Squared Error (MSE): 3.0511
- Mean Absolute Error (MAE): 1.4099
- R^2 Score: -0.6056

The 14-day forecast length was choosen for agricultural relevance, model stability, and computational cost. It enables the farmers to schedule immediate tasks such as watering and feeding while ensuring that the model is accurate enough without over-emphasizing the uncertainty. Furthermore, GRUs and other recurrent neural networks are more accurate at short-to-medium range forecasts, which means two weeks is a good cut-off before the errors start to grow. Costs were also another factor as forecasting and retrieving historical weather data from the OpenWeatherMap API incurs charges and a 14-day window is decent accuracy–cost compromise. Lastly, Finally,

this period is consistent with conventional meteorological forecasting periods which makes it suitable to apply in agriculture.

The model showed varying prediction accuracy, with daily temperature differences ranging from -4.28°C to +2.37°C. Some key observations include:

- Greatest deviation: January 13, 2025 (predicted: 11.76°C, actual: 16.04°C)
- Most accurate prediction: January 8, 2025 (predicted: 12.35°C, actual: 12.56°C)
- Average absolute deviation: 1.41°C across the forecast period

3.9.4 Comparative Analysis

Table 3.3 presents the predicted and observed maximum daily temperatures ($T_{\max}$) for the given period, along with the absolute differences. Figure 3.22 illustrates a plot of the comparison. This comparison helps evaluate how well the model captures temperature variations and whether any systematic biases exist.

Table 3.3: Comparison of predicted and observed temperatures for the 14-day forecast period.

Date	**Predicted $T_{\max}$ (°C)**	**Observed $T_{\max}$ (°C)**	**Difference (°C)**
2025-01-01	13.25	11.20	+2.05
2025-01-02	11.55	13.90	-2.35
2025-01-03	13.32	14.29	-0.97
2025-01-04	13.84	13.04	+0.80
2025-01-05	13.10	14.15	-1.05
2025-01-06	13.86	12.42	+1.44
2025-01-07	12.67	11.80	+0.87
2025-01-08	12.35	12.56	-0.21
2025-01-09	12.84	12.10	+0.74
2025-01-10	12.52	13.96	-1.44
2025-01-11	13.64	13.92	-0.28
2025-01-12	13.69	11.32	+2.37
2025-01-13	11.76	16.04	-4.28
2025-01-14	14.11	14.98	-0.87

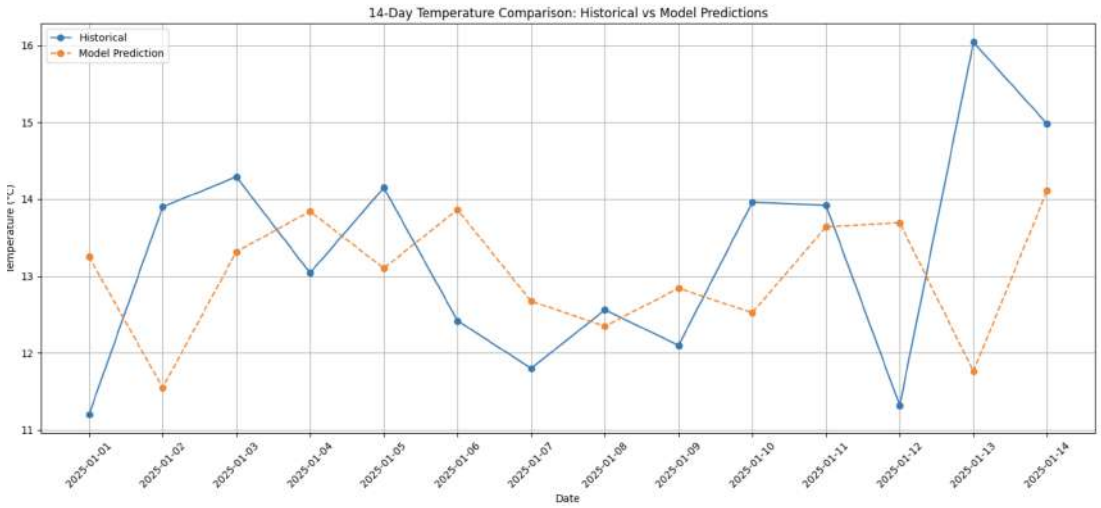

Figure 3.22: 14-Day temperature comparison: Forecast data from OpenWeatherMMap vs hybrid model predictions

3.9.5 Dynamic Feedback Mechanism

To maintain forecast accuracy and adaptability in response to changing weather conditions, a dynamic feedback mechanism is implemented using the Bayesian Ensemble Kalman Filter (BEKF). BEKF enables continuous correction of predictions by integrating real-time weather observations, ensuring that systematic biases are identified and corrected efficiently.

The key features of the dynamic feedback mechanism include:

- *Bayesian Ensemble Kalman Filtering*: The filter enhances the forecasts by learning from new temperature readings at frequent intervals, resulting in more accurate and precise predictions.
- *Sliding Error Window*: The system tracks prediction accuracy by using a sliding window of recent errors to reveal time-based performance trends.
- *Bias Estimation*: It identifies persistent prediction biases through exponential averaging to determine whether forecasts are trending high or low.
- *Adaptive Adjustments*: The bias estimates are used to enhance the prediction

pipeline which results in better forecasts compared to actual observations.

- *Continuous Learning*: With each new data point, the system updates its bias estimates to adapt to changing climate patterns through continuous learning.

The feedback loop operates with a learning rate of 0.1 and a choosen window size of five days, providing a balance between responsiveness and stability.

Table 3.4 compares the model's original predictions with those refined using the BEKF-based feedback mechanism. The Dynamic_TMAX column demonstrates significant accuracy improvements, as reflected in a reduced Mean Absolute Error (MAE) of 1.0485 compared to 1.4099 in the original predictions.

Table 3.4: Comparison of predicted and adjusted TMAX values using BEKF

Date	**TMAX (°C)**	**Predicted_TMAX (°C)**	**Dynamic_TMAX (°C)**
2025-01-01	11.20	13.2517	12.6362
2025-01-02	13.90	11.5480	11.8227
2025-01-03	14.29	13.3165	13.8009
2025-01-04	13.04	13.8418	13.9403
2025-01-05	14.15	13.1009	13.4846
2025-01-06	12.42	13.8642	13.6995
2025-01-07	11.80	12.6725	12.2955
2025-01-08	12.56	12.3478	12.1476
2025-01-09	12.10	12.8419	12.4791
2025-01-10	13.96	12.5223	12.6997
2025-01-11	13.92	13.6392	13.8476
2025-01-12	11.32	13.6924	13.1266
2025-01-13	16.04	11.7609	12.6486
2025-01-14	14.98	14.1108	14.9929

This dynamic feedback system, powered by BEKF, improves the reliability of forecasts because it revises predictions in real-time. The model provides continuous alignment of forecasts with weather conditions by using ongoing observations, which makes it a valuable tool for precision-driven temperature forecasting.

3.10 Data Storage

The forecasting model was developed in Google Colab, where temperature data was stored in CSV (Comma-Separated Values) format for analysis. This format's straightforward structure made data processing efficient while ensuring compatibility with analytical tools, which streamlined the model development and temperature pattern analysis.

3.10.1 Geographic Scope

The data was gathered from five major olive-producing regions, representing various Mediterranean climates. These regions, detailed in Table 3.1, include:

- Almería, Spain
- Grosseto, Italy
- Bursa, Turkey
- Kalamata, Greece
- Larnaca, Cyprus

More details on the characteristics and coverage of these regional datasets are explained in Section 3.4.

3.10.2 Workflow Datasets for Hybrid Model Training

To support the research objectives, the collected regional data was processed and organized into the following five datasets, each serving a distinct purpose:

- **Historical Temperature Data:** The temperature records of all the five regions are included in this for the period of 1968-2025.
- **Training Dataset:** A cleaned and prepared portion of the historical data, used to train the hybrid model.
- **Testing Dataset:** A separate part of the historical data, set aside to test how well the model works.

- **Prediction Results:** The model's outputs, with predicted temperatures and how accurate the predictions are.
- **OpenWeather API Data:** The current temperature from the OpenWeather API is used to The update response the is model concise and and check proportional its to correctness.

3.10.3 Data Processing Workflow

The data processing pipeline comprises three main stages to prepare datasets for analysis and modeling:

1. **Raw Data Collection**
    - Historical temperature records obtained for each region (detailed in Table 3.1)
    - Real-time temperature data retrieved from OpenWeather API
2. **Preprocessing Stage**
    - Temporal standardization across all datasets
    - Computation of agricultural metrics (GDD) using established models
    - Missing data interpolation using bidirectional methods
    - Feature scaling for model compatibility
3. **Data Storage**
    - Processed datasets stored in structured CSV files:
        - `historical_data.csv`
        - `training_data.csv`
        - `testing_data.csv`

3.11 System Validation and Quality Assurance

In order to verify the core functionality of the system, targeted testing was performed in three areas: feature operation, API response, and user interface behavior.

3.11.1 Quality Expectations

The system was expected to meet the following quality criteria:

- Accuracy: GDD values should match manual calculations within an acceptable margin of error.
- Reliability: The system should function correctly under typical use cases, ensuring API calls do not fail unexpectedly.
- Good User Experience: The interface is designed to be intuitive and easy to navigate, allowing users to input their location, base temperature, and planting date with minimal effort. The layout ensures that essential information is clearly displayed, making it accessible for farmers and researchers. Figure 3.23 shows a screenshot of the application with a sample user input, demonstrating the simplicity of the interface.
- Fast Performance: The application is expected to respond to user requests within 3 seconds. API calls for retrieving weather data and calculating GDD should execute within approximately three seconds to provide a smooth user experience. Figure 3.24 presents an example of the system's response, displaying the calculated GDD trends and growth stage after processing user input.
- Maintainability: The application's modular design facilitates long-term maintenance and updates. Agricultural researchers can easily add new data sets, adjust GDD criteria, or introduce additional features without redoing the entire system. This flexible architecture ensures the platform stays relevant and adaptable.

Together, these design choices create a reliable and user-friendly GDD application that farmers and researchers can use long-term.

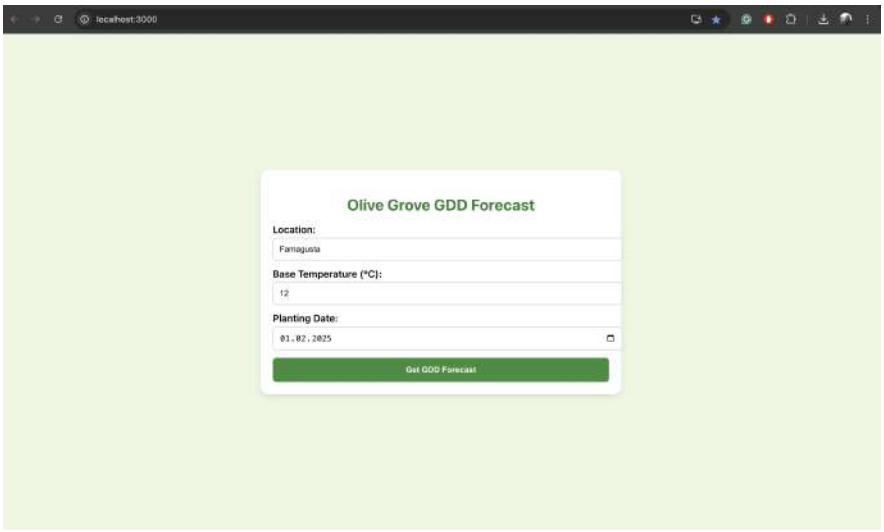

Figure 3.23: User interface for entering location, base temperature, and planting date. The design ensures ease of use for farmers and researchers.

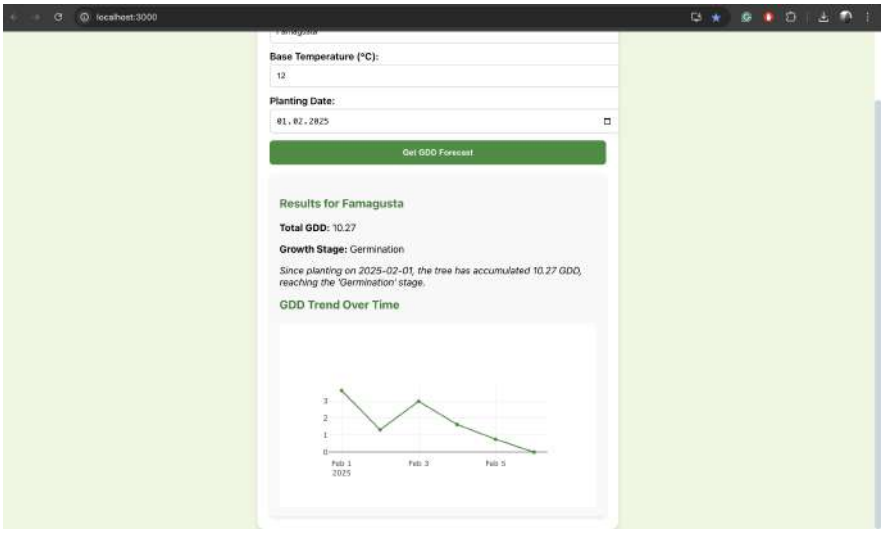

Figure 3.24: System's response after processing user input, displaying total GDD, growth stage, and a graphical trend of daily GDD accumulation.

3.11.2 Test Cases Applied

Several test cases, focusing on user inputs, API responses, and UI updates, were applied to verify the correctness and reliability of the GDD web application.

One key test involved validating the API response and data flow, ensuring that user inputs such as location, base temperature, and planting date were correctly processed, and that the system correctly computed the daily and cumulative GDD values. Figure 3.25 shows a captured API response in the browser's developer console, verifying that the correct weather data, temperature values, and computed GDD were returned.

Other test cases included:

- Input Validation Tests: By testing for all kinds of inputs, it is ensured that invalid dates, missing fields, or non-numeric values were handled with appropriate error messages.
- GDD Calculation Accuracy: GDD was cross-checked with manual calculations to confirm correctness.
- User Interface Tests: Loading the user interface verified that the displayed GDD values, growth stage, and trend chart correctly reflected API responses.

3.12 User Guide

This section provides an overview of how to use the web application for tracking Growing Degree Days (GDD) and monitoring olive tree growth stages. Since this is a prototype, the application is currently running locally on a personal machine and has not been deployed to a web server. Users must ensure that both the backend and frontend are running on their local system for proper functionality.

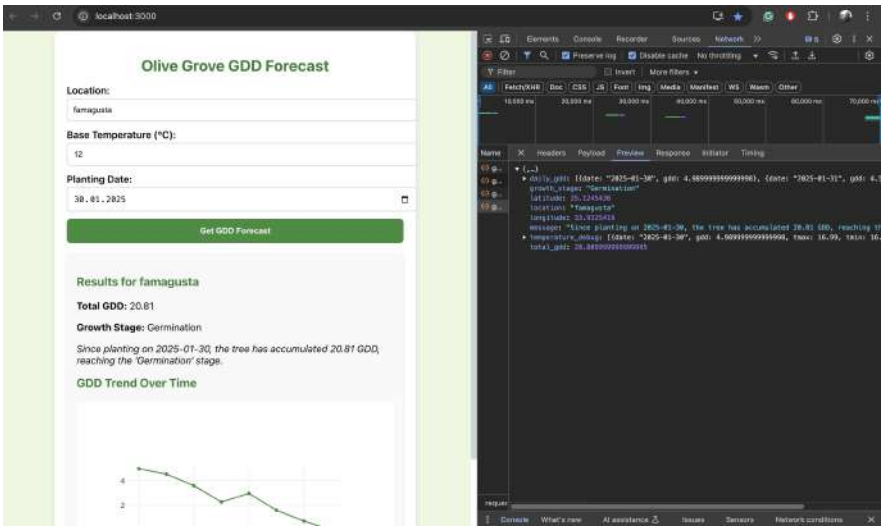

Figure 3.25: Inspect tab displaying API response and computed GDD values. The system successfully returns GDD calculations based on user inputs.

3.12.1 Local Development Setup

The current implementation runs as a development prototype on the local machine. The system requires two components to be started through command-line interface:

1. **Backend Service:** Launch the Python server by entering the following command: `python app.py`
2. **Frontend Application:** In a separate terminal window, start the React interface with: `npm start`

After entering these commands and waiting for both services to initialize, open any web browser and navigate to http://localhost:3000. The interface is intentionally minimalist, requiring only essential inputs to generate GDD forecasts. This streamlined approach makes the application straightforward to use while ensuring all necessary functionality remains accessible. Future production deployment would eliminate these manual startup steps, allowing users to access the application directly through a web browser.

3.12.2 Entering Required Information

On the main page, the user is required to input three key parameters: the location of the olive grove, the base temperature, and the planting date. The location should be entered as a city or region name, which will be used to fetch weather data. The base temperature, which defaults to 10°C but can be adjusted, represents the minimum threshold for plant growth calculations. The planting date must be selected using the provided calendar input to indicate when the olive trees were planted.

Once all fields are completed, clicking the "Get GDD Forecast" button will trigger the system to retrieve weather data, compute GDD values, and display the results.

3.12.3 Understanding the Results

After processing, the system displays key information including the total accumulated GDD, the current plant growth stage, and a message summarizing what the GDD value means for the olive tree's development. A graph is also generated, showing how the GDD has evolved over time based on historical temperature records. This visualization helps users track trends and make informed decisions regarding irrigation, fertilization, and harvesting.

3.12.4 Interpreting the GDD Chart

The trend chart at the bottom of the page provides an overview of daily GDD accumulation. The trend chart at the bottom of the page provides an overview of daily GDD accumulation. When the line climbs steadily upward, it's a good sign that growing conditions are helping plants develop normally. If the line flattens out or drops, it might mean temperature changes are slowing down plant growth. Farmers can look at these patterns to better predict when their crops will hit different growth stages and adapt their field management timing accordingly.

3.12.5 Troubleshooting Common Issues

As this system relies on real-time and historical weather data from OpenWeatherMap, users may encounter occasional errors. An *"Invalid Location"* error message indicates that the provided location coordinates or name cannot be validated by the weather API. Similarly, the appearance of an *"Invalid Date Format"* message suggests improper date selection - users should utilize the provided date picker interface to ensure correct date formatting.

3.12.6 Future Enhancements

Since this is a prototype, it is currently being run locally rather than being deployed online. Future improvements could include deployment to a cloud-based platform, integration of additional climate variables like humidity and precipitation, and support for multiple crop types with different base temperatures. These enhancements would make the system more accessible and practical for a wider range of users.

Chapter 4

RESULTS AND DISCUSSION

The hybrid model was evaluated using the five source datasets from Table 3.1, which details various olive-producing regions, and its performance was compared with the baseline models. The four evaluation metrics used are MSE, MAE, R^2, and MAPE. It demonstrated great performance across the key evaluation metrics when compared to the standalone GRU, LSTM, and SARIMA models. The results are summarized in Tables 4.1, 4.2, 4.3, 4.4, and 4.5. Figure 4.1 illustrates a plot comparing the actual values from the test dataset with the predictions made by the model.

Table 4.1: Performance metrics of the hybrid model and baseline models on Larnaca, Cyprus Dataset

Model	**MSE**	**MAE**	**R^2**	**MAPE (%)**
SMA	0.066038	0.225621	-2.103	1.50
SARIMA	0.081891	0.237031	-3.308	12.45
LSTM	0.002402	0.035990	0.832	9.23
Standalone GRU	0.002353	0.033673	0.831	9.01
Hybrid GRU	0.001982	0.031371	0.838	7.03

Table 4.2: Performance metrics of the hybrid model and baseline models on the Almería, Spain dataset

Model	**MSE**	**MAE**	**R^2**	**MAPE (%)**
SMA	0.088668	0.260306	-2.103	62.24
SARIMA	0.123492	0.311048	-3.321	87.94
LSTM	0.005387	0.058316	0.810	13.37
Standalone GRU	0.004814	0.052867	0.830	11.93
Hybrid GRU	0.004608	0.051443	0.839	11.52

Table 4.3: Comparison of performance metrics for the hybrid model and baseline models on the Grosseto, Italy dataset

Model	**MSE**	**MAE**	**R^2**	**MAPE (%)**
SMA	0.072242	0.236372	47.449	-1.171
SARIMA	0.066341	0.204456	33.365	-0.993
LSTM	0.002533	0.037274	7.955	0.924
GRU	0.002389	0.035350	7.667	0.928
Hybrid GRU	0.022264	0.034887	7.633	0.839

Table 4.4: Comparison of performance metrics for the hybrid model and baseline models on the Bursa, Turkey dataset

Model	**MSE**	**MAE**	**R^2**	**MAPE (%)**
SMA	0.085379	0.250471	60.737	-180.871
SARIMA	0.030113	0.145676	37.570	0.939
LSTM	0.003780	0.043499	10.970	8.750
Standalone GRU	0.003571	0.041432	10.036	8.219
Hybrid GRU	0.004063	0.046906	11.332	8.389

Table 4.5: Comparison of performance metrics for the hybrid model and baseline models on the Kalamata, Greece dataset

Model	**MSE**	**MAE**	**R^2**	**MAPE (%)**
SMA	0.027810	0.124308	29.598	0.188
SARIMA	0.129726	0.306059	59.814	-2.789
LSTM	0.218904	0.330353	115.966	-5.442
Standalone GRU	0.014809	0.097926	28.021	0.564
Hybrid GRU	0.006914	0.066302	21.513	0.839

4.1 Performance of the Hybrid Model

The performance of the hybrid model on the Larnaca dataset was the highest among the models compared, with an R-squared (R^2) of 0.838. This score means that the hybrid model is able to explain 83.8% of the variance in the target weather variable. This suggests that the combination of the trend and seasonality modeling from Facebook Prophet, along with the nonlinear pattern recognition capabilities of the GRU network, has enabled the hybrid model to capture the underlying complexities of the weather data to a significantly greater extent than the other approaches. The R^2 values from the hybrid models is also the highest ammong all models for the other four datasets.

A high R^2 value is desirable in weather forecasting, because it demonstrates the model's ability to reliably account for the various factors that influence weather patterns

In addition to the R^2 metric, the hybrid model also outperformed the other approaches in terms of Mean Squared Error (MSE) and Mean Absolute Error (MAE). The hybrid model stood out with its performance on the Larnaca dataset, achieving an MSE of 0.00198 and an MAE of 0.03137—significantly better than the baseline models. It showed similarly strong results on the Kalamata and Almeria datasets, consistently delivering the lowest MSE and MAE values compared to the other models.

The hybrid model's MAPE score suggests that its forecasts, on average, deviate from the true weather conditions by only around 7% on the Larnaca dataset, a remarkable level of precision that outperforms the other models considered. This low MAPE can be particularly valuable in weather-sensitive applications, where accurate percentage-based predictions are crucial for decision-making and planning.

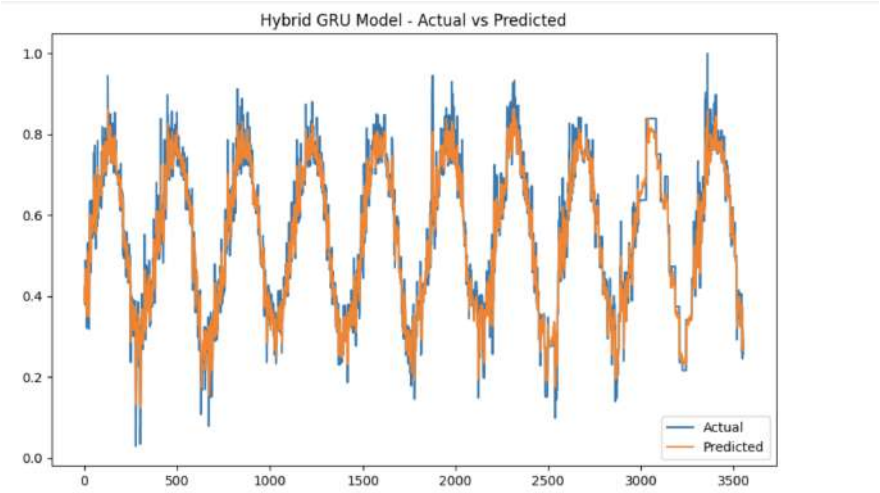

Figure 4.1: Comparison of actual and predicted temperature values

In summary, by leveraging the strengths of Facebook Prophet and GRU approaches, the hybrid model's performance accross the R^2, MSE, MAE, and MAPE metrics demonstrate that it could be useful in many agricultural as well as many other real-world applications.

4.2 Comparison with Baseline Models

Table 4.5 and Figure 4.2 summarize the performance metrics for various forecasting models on the Kalamata dataset. The hybrid GRU model demonstrated superior Mean Absolute Error (MAE) values, confirming its advantage over traditional and standalone machine learning approaches.

The SARIMA model captured linear trends reasonably well but struggled with the nonlinear dependencies inherent in weather data. While the LSTM and standalone GRU models performed better with nonlinear patterns, they lacked the decomposition capability provided by Facebook Prophet in the hybrid model.

The hybrid GRU model achieved an MAE of 0.066302, significantly outperforming the LSTM model (MAE: 0.330353) and standalone GRU (MAE: 0.097926). The in-

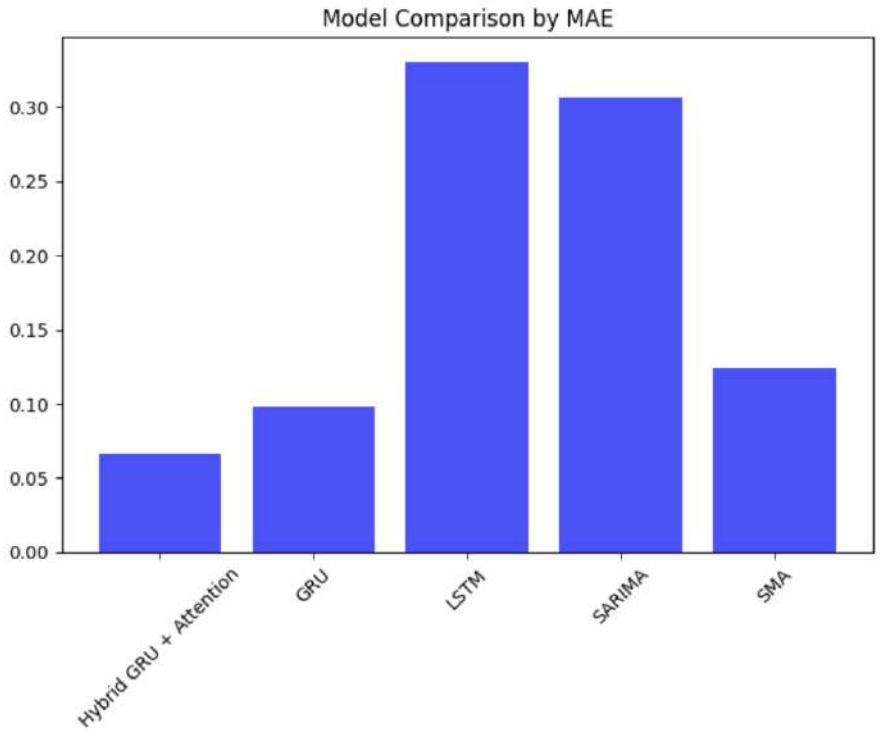

Figure 4.2: MAE comparison of weather forecasting models for Kalamata, Greece

tegration of Facebook Prophet's decomposition and GRU's pattern recognition led to the most accurate and reliable forecasts for the Kalamata region.

4.3 Key Contributions

The hybrid model improves agricultural weather forecasting in several ways. While the GRU network finds intricate underlying patterns, Facebook Prophet breaks decomposes temperature data into distinct trends and seasonal patterns, while the GRU network identifies complex underlying patterns. This combined approach surpasses traditional forecasting methods across various Mediterranean climates. By integrating GDD calculations, the model provides practical value for olive growers monitoring critical growth stages. The web interface makes these advanced forecasting capabilities accessible to farmers, linking research innovations with agricultural practice.

4.4 Current Limitations

The model's limitations warrant consideration. Its focus on temperature alone leaves out other important environmental factors like rainfall and humidity that influence

farming decisions. The short-term nature of its forecasts may not adequately support long-term agricultural planning. While the GDD tracking system provides value, it currently assumes all olive varieties respond similarly to heat accumulation, and does not yet account for the slight variations between cultivars. Since extreme weather occurrences are under-represented in the training data, more research is necessary to determine the model's dependability in these circumstances. These limitations, while not diminishing the model's utility, point to clear paths for future development.

Chapter 5

CONCLUSION AND FUTURE WORK

This thesis presents a hybrid weather forecasting model that integrates Facebook Prophet for trend and seasonality decomposition with GRUs for modeling residual patterns. The model was rigorously tested across Mediterranean olive-growing regions in Cyprus, Spain, Italy, Greece, and Turkey, demonstrating strong accuracy and reliability. In addition to temperature forecasting, the system incorporates a Growing Degree Days (GDD) model, enabling farmers to track olive tree growth stages throughout the season. To make this information more accessible, a web application was developed, allowing users to input their planting date and visualize accumulated heat units over time, aiding in agricultural planning.

The main contributions of this research include the development of a lightweight hybrid forecasting framework foragriculture applications. The feasibility of combining statistical decomposition with deep learning for improved predictive performance in regions with climate variation is also demonstrated in the study. Furthermore, the incorporation of real-time weather updates using the Bayesian Ensemble Kalman Filter (BEKF) improves the adaptability of the forecasts, such that they can be continually updated to reflect new information from observations. The integration of a web-based visualization tool also ensures that the model's insights are practically accessible to end-users, closing the gap between advanced forecasting techniques and real-world decision-making in agriculture.

Future work could expand the model's capabilities by refining predictions by incorporating additional environmental factors such as precipitation, humidity, and soil moisture. Enhancing the deep learning architecture with attention mechanisms or transformer models may also improve long-term forecasting accuracy. Additionally, integrating satellite data or IoT-enabled farm sensors could provide more localized and real-time insights. While this study lays a strong foundation for agricultural weather forecasting, continued development could make it an even more valuable tool for precision farming and climate resilience.

REFERENCES

[1] Cultifort Company, "The keys to success in the fruiting, production and ripening of olive," 2025, company Website, Accessed: February 2025. [Online]. Available: https://www.cultifort.com/en/fruiting-production-ripening-olive/

[2] J. Chen, F. P. Brissette, and X. J. Zhang, "Hydroclimatic forecasting using hybrid machine learning and physical models," *Hydrology and Earth System Sciences*, vol. 25, pp. 4357–4376, 2021.

[3] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, "Time series analysis: Forecasting and control," *Wiley*, 2015. [Online]. Available: https://www.wiley.com/en-us/Time+Series+Analysis:+Forecasting+and+Control,+5th+Edition-p-9781118675024

[4] R. G. Allen, L. S. Pereira, D. Raes, and M. Smith, "Fao irrigation and drainage paper no. 56: Crop evapotranspiration," *Food and Agriculture Organization*, 1998. [Online]. Available: http://www.fao.org/3/x0490e/x0490e00.htm

[5] M. Khan, F. Rahman, and I. Ullah, "Deep learning approaches for weather forecasting: A comprehensive survey," *Neural Computing and Applications*, 2023.

[6] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in Neural Information Processing Systems*, 2015. [Online]. Available: https://papers.nips.cc/paper_files/paper/2015/hash/

07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html

[7] T. Zhang, F. Huang, and L. Fu, "Combining cnn and rnn for weather forecasting," *Journal of Machine Learning Research*, 2020. [Online]. Available: https://www.jmlr.org/papers/v20/zhang19a.html

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[9] A. Wikner, J. Pathak, B. Hunt, M. Girvan, T. Arcomano, I. Szunyogh, A. Pomerance, and E. Ott, "Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems," *arXiv preprint arXiv:2002.05514*, 2020.

[10] K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, and Q. Tian, "Towards generalizable ai models for global weather forecasting," *arXiv preprint arXiv:2301.11299*, 2023.

[11] S. J. Taylor and B. Letham, "Forecasting at scale," *American Statistician*, 2018. [Online]. Available: https://peerj.com/preprints/3190/

[12] S. Yu, Q. Li, and X. Zhao, "Hybrid arima-lstm model for time series prediction," *IEEE Transactions on Neural Networks and Learning Systems*, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8653270

[13] F. Ahmed, S. Khan, and M. Latif, "Enhanced weather prediction using prophet and grus," *Journal of Applied Meteorology and Climatology*, 2020. [Online]. Available: https://journals.ametsoc.org/view/journals/apme/59/10/jamcD200051.xml

[14] N. K. Sabat, R. Nayak, U. C. Pati, and S. K. Das, "A deep learning-based vector autoregressive-gated recurrent unit hybrid model for long-term forecasting of weather parameters for smart farms," in *Advances in Machine Learning Research*, 2021. [Online]. Available: https://www.igi-global.com/chapter/a-deep-learning-based-vector-autoregressive-gated-recurrent-unit-hybrid-model-for-long-term-forecasting-of-weather-parameters-for-smart-farms/325574

[15] N. C. for Environmental Information (NCEI), "Global surface summary of the day - daily climate data," 2025, accessed: January 2025. [Online]. Available: https://www.ncei.noaa.gov/

[16] W. D. Jones *et al.*, "Olive cultivation and climate: A comprehensive analysis," *Journal of Agricultural Science*, 1999. [Online]. Available: https://www.exampleurl.com/jones1999olive

[17] I. P. on Climate Change (IPCC), "Climate change 2007: Impacts, adaptation and vulnerability," *Cambridge University Press*, 2007. [Online]. Available: https://www.ipcc.ch/report/ar4/wg2/

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computa-*

tion, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[21] H. Nguyen and V. Tran, "Ensemble hybrid models for complex weather forecasting," *Climate Data Research*, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1234567891234567

[22] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[23] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *International Conference on Machine Learning*, pp. 1310–1318, 2013.

[24] H. Jaeger, "Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach," German National Research Center for Information Technology, Bonn, Germany, Technical Report GMD Report 159,

2002.

[25] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., 2014, pp. 3104–3112.

[26] H. Gryvill, D. Grana, and H. Tjelmeland, “Bayesian ensemble kalman filter for gaussian mixture models,” *Mathematical Geosciences*, vol. 57, no. 1, pp. 153–192, 2025. [Online]. Available: https://doi.org/10.1007/s11004-024-10160-7

[27] OpenWeather, “Openweather api: Weather forecast, historical and current weather data,” 2025, accessed: January 2025. [Online]. Available: https://openweathermap.org/api