## SOIL HEAT FLUX DYNAMICS MODELLING USING TEMPORAL DEEP LEARNING FOR DETERMINING PLANT ROOT ZONE TEMPERATURE

Bonafide record of work done by

| **MANJUSHA S** | (21Z317) |
| --- | --- |
| **MYDHILI NAYAKI R** | (21Z324) |
| **SHRINIDHI S** | (21Z352) |
| **SWETHA M** | (21Z362) |
| **VEDAVARSHINI A** | (21Z368) |
|  |  |

## 19Z720 – PROJECT WORK 1

**GUIDE: Dr. N. Gopikarani**

Dissertation submitted in the partial fulfillment of the requirements for the degree of

## BACHELOR OF ENGINEERING BRANCH:COMPUTER SCIENCE AND ENGINEERING



NOVEMBER 2024

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING PSG COLLEGE OF TECHNOLOGY

**(Autonomous Institution)**

## COIMBATORE – 641 004

**PSG COLLEGE OF TECHNOLOGY**

(Autonomous Institution)

COIMBATORE – 641 004

**COMPUTER SCIENCE & ENGINEERING**

**SOIL HEAT FLUX DYNAMICS MODELLING USING TEMPORAL DEEP LEARNING FOR DETERMINING PLANT ROOT ZONE TEMPERATURE**

Bonafide record of work done by

**Manjusha S (21Z317)**

**Mydhili nayaki R (21Z324)**

**Shrinidhi S(21Z352)**

**Swetha M (21Z362)**

**Vedavarshini A (21Z368)**

Dissertation submitted in partial fulfillment of the requirements for the degree of **BACHELOR OF ENGINEERING**

BRANCH: COMPUTER SCIENCE AND ENGINEERING

of Anna University

NOVEMBER 2024

**Dr. N Gopikarani Dr. G. Sudha Sadasivam** Faculty Guide Head of the Department Assistant Professor (SG) Department of Computer Science

Department of Computer Science & Engineering

& Engineering

Certified that the candidate was examined in the viva voce examination held on\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner External Examiner

**CERTIFICATE**

Certified that this report titled **“SOIL HEAT FLUX DYNAMICS MODELLING USING TEMPORAL DEEP LEARNING FOR DETERMINING PLANT ROOT ZONE TEMPERATURE”** for the Project Work I (19Z720) is a bonafide work of **“Manjusha S(21Z317), Mydhili nayaki R(21Z324), Shrinidhi S(21Z352), Swetha M(21Z362), Vedavarshini A(21Z368)”** who have carried out the work for the partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion.

**Place :** Coimbatore

**Date :**

**Faculty Guide**

Dr. N Gopikarani

Department of Computer Science and Engineering

PSG College of Technology,

Coimbatore - 641004

COUNTERSIGNED

**Dr. Sudha Sadasivam G**

**HEAD**

**Department of Computer Science and Engineering**

**PSG College of Technology, Coimbatore - 641004**

**TABLE OF CONTENTS PAGE**

[**ACKNOWLEDGEMENT**](#_3znysh7) **1**

[**SYNOPSIS**](#_2et92p0) **2**

**LIST OF FIGURES 3**

**LIST OF TABLES 4**

1. **INTRODUCTION 5**

1.1.[PROBLEM STATEMENT](#_tyjcwt) 6

1. **LITERATURE SURVEY 7**

2.1. LITERATURE SURVEY AND INFERENCES 7

1. **SYSTEM REQUIREMENTS 14**

3.1 HARDWARE REQUIREMENTS 14

3.2 SOFTWARE REQUIREMENTS 14

1. **SYSTEM DESIGN 15**

4.1 DESIGN 15

4.2 TOOLS AND DATASET USED 17

4.3 DATASET DESCRIPTION 19

4.4 EXISTING MODELS 20

4.5 SYSTEM WORKFLOW 21

4.6 PROPOSED SOLUTION 26

1. **IMPLEMENTATION 33**

5.1 DATA UNDERSTANDING AND PREPROCESSING 33

5.2 ALGORITHM 41

5.3 RESULT ANALYSIS 47

**6. CONCLUSION 51**

**7. BIBLIOGRAPHY 52**

**8. APPENDIX 53**

## ACKNOWLEDGEMENT

We would like to extend our sincere thanks to our Principal, **Dr. K. Prakasan**, PSG College of Technology, Coimbatore, for providing us with this opportunity to develop and complete our project in our field of study.

We express our sincere thanks to our Professor and the Head of the Department, **Dr. Sudha Sadasivam G**, Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, for encouraging and allowing us to present the project at our department premises for the partial fulfillment of the requirements leading to the award of B.E. degree.

We take immense pleasure in conveying our thanks and a deep sense of gratitude to our Professor and the Programme Coordinator, **Dr. Shanthi V**, Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, for his relentless support and motivation to constantly up skill ourself throughout this project.

We would like to express our gratitude to our faculty guide **Dr. N Gopikarani**, Assistant Professor (SG), Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, for her valuable suggestions, encouragement, whole-hearted cooperation, constructive criticism, and guidance throughout this project work.

Additionally, we would like to extend our sincere gratitude to **Dr. S. Arul Jothi**, Assistant Professor (Sr. Gr.), Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, whose continuous support and guidance have been instrumental in the successful completion of our project.

## 

## 

## 

## SYNOPSIS

The problem statement highlights the significant challenge of accurately predicting soil heat flux dynamics, which are crucial for regulating plant root zone temperature. Inadequate control of root zone temperature can lead to poor plant health, reduced crop yields, and increased vulnerability to pests and diseases, especially in the context of Indian agriculture, where productivity is often hampered by suboptimal soil temperature regulation.

The objective of this project is to develop a robust ensemble model using Temporal Convolutional Networks (TCNs) and Artificial Neural Networks (ANNs) to dynamically forecast soil heat flux. The model will leverage sequential environmental data and soil temperature records to predict heat flux, optimize root zone temperature, and ultimately enhance agricultural productivity.

The project involves two main tasks:

1. **Data Collection and Preparation**: Compiling a dataset of soil temperature, other environmental factors and soil heat flux data.
2. **Model Development and Training**: Designing and training a Temporal Convolutional Network (TCN) capable of capturing complex temporal patterns in sequential data and Artificial Neural Networks (ANN) to predict soil heat flux with high accuracy.

By accomplishing these tasks, the project aims to improve agricultural productivity and sustainability by providing farmers with actionable insights into soil heat flux dynamics. This advancement aligns with India’s goals for food security and economic growth by reducing crop losses and optimizing farming practices through better soil temperature management.

**LIST OF FIGURES**

| **FIGURE** | **FIGURE TITLE** | **PAGE** |
| --- | --- | --- |
| 4.5.1 | Ensemble Model Workflow | 21 |
| 4.5.2 | Stacking in Ensembling | 23 |
| 4.5.3 | Workflow of Optimization Module | 24 |
| 4.5.4 | Correlation Heatmap | 25 |
| 4.6.1 | TCN Architecture | 26 |
| 4.6.2 | ANN Architecture | 28 |
| 5.1.1 | Box plot of soil heat flux | 33 |
| 5.1.2 | Hexbin plot | 34 |
| 5.1.3 | Correlation Matrix heatmap | 34 |
| 5.1.4 | Diurnal Variation and Overall Temporal trends | 35 |
| 5.1.5 | Autocorrelation Function (ACF) | 36 |
| 5.1.6 | Partial Autocorrelation Function (PACF) | 36 |
| 5.1.7 | Feature Selection Results | 37 |
| 5.1.8 | Root Zone Temperature Trend | 38 |
| 5.1.9 | Soil Temperature Trend | 39 |
| 5.2.1 | Predictions for Validation set | 48 |
| 5.2.2 | Predictions for Testing set | 49 |
| 5.2.3 | Future Predictions- Forecasting | 49 |
| 5.2.4 | Optimized Temperature in various Timeframes | 50 |

**LIST OF TABLES**

| **TABLE** | **TABLE TITLE** | **PAGE** |
| --- | --- | --- |
| 2.1 | Literature Survey | 8 |
| 3.1.1 | Hardware Requirements | 14 |
| 3.2.1 | Software Requirements | 14 |
| 4.2.1 | Tools and Dataset Used | 17 |
| 5.3.1 | Performance Metrics - MSE | 47 |
| 5.3.2 | Performance Metrics - MAE | 47 |
| 5.3.3 | Performance Metrics - R-squared | 48 |

**CHAPTER 1**

**INTRODUCTION**

In the evolving landscape of modern agriculture, ensuring the optimal health and productivity of crops is a critical challenge, particularly in regions like India where agriculture forms the backbone of the economy. One of the key factors influencing crop health and yields is the regulation of soil temperature, particularly in the root zone. Fluctuations in soil heat flux, which govern the transfer of energy within the soil, directly impact the root zone temperature, influencing plant growth, resilience, and overall agricultural productivity.

Traditionally, managing soil heat flux dynamics has been a manual and labor-intensive process, often relying on observational methods and historical data to guide farming decisions. However, with increasing environmental variability and the growing demand for higher yields, traditional methods are becoming insufficient. Suboptimal root zone temperature management leads to reduced crop yields, poor plant health, and increased vulnerability to pests and diseases.

To address this challenge, this project seeks to leverage the power of Artificial Neural Network (ANN) and Temporal Convolutional Networks (TCNs) to develop a predictive model capable of dynamically forecasting soil heat flux. By capturing complex temporal patterns in environmental data, the model aims to optimize root zone temperature, improve crop yields, and ultimately enhance the sustainability of Indian agriculture. This approach not only represents a breakthrough in soil heat flux prediction but also aligns with the broader goals of boosting agricultural productivity and ensuring food security.

## 1.1. PROBLEM STATEMENT:

* Productivity of Indian agriculture is often hampered by suboptimal soil temperature regulation, particularly in the root zone of plants.
* Inadequate root zone temperature control can lead to poor plant health, reduced crop yields and increased vulnerability to pests and diseases.
* Soil heat flux governs the transfer of energy within the soil.
* Need for handling sequential data and capturing complex temporal patterns.
* By leveraging TCN and ANN , this project aims to develop a robust predictive model that can dynamically forecast soil heat flux and optimize root zone temperature.
* This advancement is expected to significantly improve agricultural productivity and sustainability in India, aligning with the nation's goals for food security and economic growth.

# CHAPTER 2

# LITERATURE SURVEY

**INTRODUCTION:**

Recent advancements in soil science have led to innovative approaches for predicting soil properties and dynamics, utilizing machine learning, remote sensing, and statistical techniques. These studies enhance the accuracy of soil moisture, heat flux, and other critical parameters, facilitating improved agricultural management and environmental conservation.

**LITERATURE SURVEY:**

1. Bruno César Comini de Andrade et al. address inefficiencies in traditional soil heat flux models by employing an Artificial Neural Network (ANN) to enhance predictions across diverse land covers in South America. Utilizing data from 23 flux towers, their model achieves up to a 43% reduction in mean absolute error. This approach highlights the importance of land cover information in improving predictive accuracy and understanding soil-plant-atmosphere interactions. ***[1]***
2. Leugim Corteze Romio et al. presents a numerical model aimed at estimating soil thermal conductivity using field experimental data. The model addresses the need for accurate predictions of thermal conductivity, which is crucial for various applications in agriculture and environmental science. By utilizing empirical data collected from field experiments, the authors aim to enhance the reliability of soil thermal conductivity estimates across different soil types. ***[2]***
3. James F. Cross and Darren T. Drewry explore the application of ensemble machine learning techniques to improve the estimation of soil heat flux. By integrating multiple machine learning models, the authors aim to enhance predictive accuracy while ensuring interpretability of the results. The research addresses the challenges associated with traditional methods, which often lack transparency and reliability in estimating soil heat flux across varying environmental conditions. ***[3]***
4. Yujie Liu et al. investigate the effectiveness of Temporal Convolutional Networks (TCNs) for time series prediction in their study. Recognizing the limitations of traditional forecasting methods, the authors aim to leverage the capabilities of TCNs to capture temporal dependencies in sequential data. By employing this advanced deep learning architecture, the research seeks to enhance prediction accuracy and address challenges associated with modeling complex time series data. ***[4]***
5. Rameshwar Garg et al. provide a comprehensive survey of machine learning algorithms for time series analysis and forecasting. The study emphasizes the significance of accurate forecasting in various fields and examines models such as ARIMA, Prophet, and Long Short-Term Memory (LSTM) networks. By exploring both traditional and hybrid approaches, the authors offer valuable insights into the latest advancements in time series prediction methodologies. ***[5]***
6. Ruixue Li et al. addresses the limitations of existing soil prediction models, which often struggle with inefficiency and inaccuracy. By leveraging near-infrared spectroscopy data, the Multi\_CNN model aims to simultaneously predict multiple soil properties, thereby improving the overall predictive performance. ***[6]***
7. Anze Liang et al. investigates the variations of soil heat flux (G) in urban riparian woodlands and its correlation with net radiation (Rn). Using data from automated sensors, the study finds that G is influenced by leaf area index (LAI) and soil water content (SWC), aiming to enhance energy balance models for these ecosystems. ***[7]***
8. Toby A. Adjuik and Sarah C. Davis investigated machine learning models for predicting soil CO2 emissions using the GRACEnet database. It finds that random forest and gradient boosting algorithms outperform others, demonstrating ML's effectiveness as a cost-efficient alternative to traditional measurement methods. ***[8]***
9. [Zoren P. Mabunga](https://ieeexplore.ieee.org/author/37088382646) and Jennifer C. Dela Cruzpresents an optimized model utilizing Gaussian Process Regression (GPR) to predict soil moisture levels in smart agriculture. By leveraging GPR’s ability to provide probabilistic predictions and quantify uncertainty, the study aims to enhance soil moisture estimation, which is crucial for effective irrigation management and crop yield optimization. ***[9]***
10. Anze Liang et al. investigates the effects of raising root zone temperature (RZT) on the growth and metabolite production of hydroponically grown red leaf lettuce. Conducted under controlled air temperatures, the research aims to understand how a slight increase in RZT can enhance plant productivity and nutrient uptake, ultimately contributing to better crop yields in plant factory systems. ***[10]***

| **S.NO** | **PAPER TITLE** | **AUTHORS AND PUBLISHER** | **SUMMARY** |
| --- | --- | --- | --- |
| 1 | Artificial Neural Network Model of Soil Heat Flux over Multiple Land Covers in South America | Bruno César Comini de Andrade, Olavo Correa Pedrollo, Anderson Ruhoff, Adriana Aparecida Moreira  **(MDPI)** | **Advantages:**   * Improved accuracy with reduced mean absolute error. * Incorporates diverse land cover data for better predictions.   **Disadvantages:**   * Computational complexity requiring significant resources. * Potential for overestimation in specific land covers.   **Algorithm Used:**   * Artificial Neural Network (ANN) |
|  | A Numerical Model to Estimate the Soil Thermal Conductivity Using Field Experimental Data | Leugim Corteze Romio, Débora Regina Roberti, Lidiane Buligon, Tamires Zimmer, and Gervásio Annes Degrazia **(MDPI)** | **Advantages:**   * Enhanced accuracy in estimating soil thermal conductivity using real field data. * Broad applicability across various soil types and conditions..   **Disadvantages:**   * Accuracy heavily reliant on the quality and quantity of field data. * Calibration may require extensive data collection and processing.   **Algorithm Used:**   * Numerical modeling approach integrating empirical data |
| 3 | Ensemble Machine Learning for Interpretable Soil Heat Flux Estimation | James F. Cross, Darren T. Drewry  **(Elsevier)** | **Advantages:**   * Enhanced predictive accuracy compared to single-model techniques. * Improved interpretability of factors influencing soil heat flux.   **Disadvantages:**   * More complex to implement and tune compared to simpler models. * Requires high-quality datasets for effective training.   **Algorithm Used:**   * Ensemble Machine Learning Techniques |
| 4 | Time Series Prediction Based on Temporal Convolutional Network | Yujie Liu, Hongbin Dong, Xingmei Wang, Shuang Han  (**IEEE)** | **Advantages:**   * High accuracy in capturing long-range dependencies in data. * Allows for parallel processing, reducing training time.   **Disadvantages:**   * Implementation can be complex and requires careful tuning. * Needs large datasets to avoid overfitting and ensure effective training.   **Algorithm Used:**   * Temporal Convolutional Networks (TCNs) |
| 5 | Machine Learning Algorithms for Time Series Analysis and Forecasting | Rameshwar Garg, Shriya Barpanda, Girish Rao Salanke N S, Ramya Shivamadegowda  (**Cornell Archives)** | **Advantages:**   * Comprehensive overview of various forecasting methods available today. * Exploration of hybrid models enhances potential forecasting accuracy.   **Disadvantages:**   * Advanced models can be complex to implement and require expertise. * Performance is heavily reliant on the quality and quantity of data available.   **Algorithm Used:**   * ARIMA, Prophet, LSTM, and hybrid models. |
| 6 | Simultaneous Prediction of Soil Properties Using Multi\_CNN Model | Ruixue Li, Bo Yin, Yanping Cong and Zehua Du **(MPDI)** | **Advantages:**   * Predicts multiple soil properties simultaneously. * Combines one-dimensional and two-dimensional convolutions for better accuracy. * Effective across diverse datasets and soil types. * Faster processing with dual-stream architecture.  Disadvantages:  * Requires more computational resources and expertise. * Needs high-quality near-infrared spectroscopy data. * Higher complexity can lead to overfitting.  Algorithm Used: Dual-Stream CNN (Multi\_CNN):   * One-Dimensional Convolutions for spectral data. * Two-Dimensional Convolutions for spatial analysis |
| 7 | Daily Dynamics of Soil Heat Flux and Its Relationship with Net Radiation in Different Urban Riparian Woodlands | Anze Liang, Changkun Xie, Jing Wang and Shengquan Che **(MPDI)** | **Advantages:**   * Examines the interplay between soil heat flux and net radiation across various urban riparian woodlands. * Aims to enhance energy balance models, contributing to better simulations for riparian ecosystems. * Employs SPSS for data analysis, utilizing linear and polynomial regression models for robust results.  Disadvantages:  * The relationship between soil properties and environmental factors can be intricate, requiring extensive data collection and analysis. * Results may be specific to the studied locations, limiting generalizability to other regions or woodland types.  Algorithm Used:  * Data Analysis Techniques: Linear and polynomial regression models |
| 8 | Machine Learning Approach to Simulate Soil CO2 Fluxes under Cropping Systems | Toby A. Adjuik and Sarah C. Davis  **(MDPI)** | **Advantages:**   * Demonstrates that ML can accurately predict soil CO2 fluxes using readily available field data. * Offers a more efficient alternative to traditional measurement methods for estimating soil CO2 emissions. * Leverages the extensive GRACEnet database, enhancing model training and validation.   **Disadvantages:**   * Some ML algorithms may require substantial computational resources and expertise for implementation. * The study identifies that certain variables, while influential, may not be highly predictive in correlation analysis.   **Algorithm Used:**   * KNN: K-nearest neighbor regression. * SVR: Support vector regression. * RF: Random forest regression. * GB: Gradient boosted regression. |
| 9 | An Optimized Soil Moisture Prediction Model for Smart Agriculture Using Gaussian Process Regression | [Zoren P. Mabunga](https://ieeexplore.ieee.org/author/37088382646); Jennifer C. Dela Cruz **(IEEE)** | **Advantages:**   * GPR offers not only point estimates but also uncertainty quantification, allowing for better decision-making. * The model can adapt to complex relationships in the data, making it suitable for various agricultural conditions. * GPR does not require a fixed number of parameters, allowing it to adjust based on the data complexity.   **Disadvantages:**   * GPR can be computationally expensive, particularly with large datasets, as it requires the inversion of a covariance matrix. * The performance of GPR heavily depends on the selection of kernel functions and their hyperparameters, which can significantly impact results. * As the number of data points increases, the computational cost can grow cubically, making it less efficient for very large datasets.   **Algorithm Used:**  Gaussian Process Regression (GPR) |
| 10 | Raising Root Zone Temperature Improves Plant Productivity and Metabolites in Hydroponic Lettuce Production | Anze Liang, Changkun Xie, Jing Wang, Shengquan Che **(Frontiers in Plant Science)** | **Advantages:**   * Raising RZT by 3°C improved plant growth and increased key metabolites such as chlorophyll and carotenoids. * The study demonstrated that higher RZT enhances the uptake of essential nutrients, benefiting overall plant health. * Conducted in a hydroponic system, allowing precise control over environmental variables for optimal growth conditions.   **Disadvantages:**   * The effectiveness of RZT increases may vary with different plant species or environmental conditions. * Heating the root zone requires energy, which could increase operational costs in large-scale production. * The study focuses on one type of lettuce, which may limit the generalizability of the findings to other crops.   **Algorithm Used:**   * Principal Component Analysis (PCA) * Weighted Correlation Network Analysis (WGCNA) |

Table 2.1 Literature Survey

**CONCLUSION:**

The findings highlight the effectiveness of advanced modeling techniques in understanding soil dynamics. Innovations such as Gaussian Process Regression for soil moisture and ensemble machine learning for heat flux estimation are crucial for smart agriculture. These approaches not only improve prediction accuracy but also contribute to sustainable agricultural practices and a better understanding of environmental processes.

**CHAPTER 3**

**SYSTEM REQUIREMENTS**

**3.1 HARDWARE REQUIREMENTS**

| **Component** | **Specification** |
| --- | --- |
| CPU | Multi-core processor (e.g., Intel i7, AMD Ryzen 7) |
| GPU | High-performance GPU (e.g., NVIDIA RTX 3080, Tesla V100) |
| RAM | At least 16 GB (32 GB or more preferred) |
| Storage | High-speed SSD (at least 1 TB), additional backup |
| Networking | Reliable high-speed internet connection |

Table 3.1.1 Hardware Requirements

**3.2 SOFTWARE REQUIREMENTS**

| **Component** | **Specification** |
| --- | --- |
| Operating System | Linux (Ubuntu) or Windows 10/11 |
| Development Environment | Python, Jupyter Notebook, PyCharm, Visual Studio Code or Kaggle Notebook |
| Deep Learning Frameworks | TensorFlow, PyTorch |
| Libraries | NumPy, pandas, SciPy, scikit-learn, Keras |
| Visualization Tools | Matplotlib, Seaborn, TensorBoard |

Table 3.2.1 Software Requirements

**CHAPTER 4**

**IMPLEMENTATION AND SYSTEM DESIGN**

**4.1 DESIGN**

The system workflow is discussed in two phases: **temporal forecasting and analysis of soil heat flux values** and the **determination and optimization of plant root zone temperature**.

**4.1.1 Temporal forecasting and analysis of soil heat flux values**

**System Architecture:**

The system architecture for predicting soil heat flux dynamics using a Temporal Convolutional Network (TCN) and Artificial Neural Network (ANN) ensemble is composed of several interconnected components:

* **Data Collection:** The first step involves gathering historical soil heat flux data, weather parameters, soil properties, and other relevant environmental factors from

**PANGAEA - Data Publisher for Earth & Environmental Science**

* **Data Preprocessing:** Collected data undergoes preprocessing to remove noise, handle missing values using Mean Imputation, and ensure consistency. Plots like boxplot, hexbin plot indicated the presence of outliers and were handled using IQR techniques.The sequential data is cleaned and normalized for time-series modeling. Techniques such as Min-Max scaling and feature selection techniques are applied to the time-series data. Temporal sequences are then split into training, validation, and test sets.
* **Model Training:** The core of the architecture involves training an ensemble model that combines a Temporal Convolutional Network (TCN) and an Artificial Neural Network (ANN). The TCN captures long-range temporal dependencies and local patterns in the time-series data, while the ANN models complex non-linear relationships between features. The two models are trained separately for multiple epochs ranging from 10 to 100, and their outputs are stacked using a stacking method-ensembling approach to generate final predictions.

TCN: This model captures sequential dependencies over time with convolutions, eliminating the need for recurrent connections.

ANN: This model captures non-linear feature interactions and further processes the output from TCN for refined predictions.

* **Model Evaluation:** The ensemble model is evaluated using metrics such as Mean Absolute Error (MAE), and R² score to assess how well the model predicts soil heat flux . Performance comparisons between the TCN, ANN, and ensemble model are made to determine the best approach.

**4.1.2 Determination and optimization of plant root zone temperature**

The determination and optimization of root zone temperature is performed using the temporal forecasting results obtained from the TCN + ANN model.

Soil temperatures at depths of 10 cm, 20 cm, 40 cm, and 80 cm were weighted using an exponential decay formula. This was based on the assumption that shallower depths have a larger influence on RZT. The weighted sum of soil temperatures was computed, yielding the RZT for each timestamp.

* **Time-of-Day Segmentation**: Data was divided into four intervals based on time (Night, Morning, Afternoon, and Evening) for better analysis of temperature patterns across different parts of the day.
* **Optimization Setup**: The RZT values were optimized using **Bayesian optimization** with **Gaussian Process minimization**, targeting an ideal temperature of 25°C. The optimization process aimed to minimize deviations from this ideal temperature, accounting for penalties if the RZT exceeded or fell below acceptable limits.
* **Evaluation:** To evaluate the effectiveness of the optimization, key performance metrics were computed:
  + RMSE (Root Mean Squared Error) to measure the average squared difference between the optimized RZT and the ideal temperature.
  + MAE (Mean Absolute Error) to indicate the average absolute error between optimized and ideal temperatures.
  + MAPE (Mean Absolute Percentage Error) and SMAPE (Symmetric Mean Absolute Percentage Error) for evaluating the relative percentage-based errors, with SMAPE giving a symmetric measure of error between actual and ideal temperatures.
  + Max Error to find the largest deviation between optimized RZT values and the ideal temperature.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

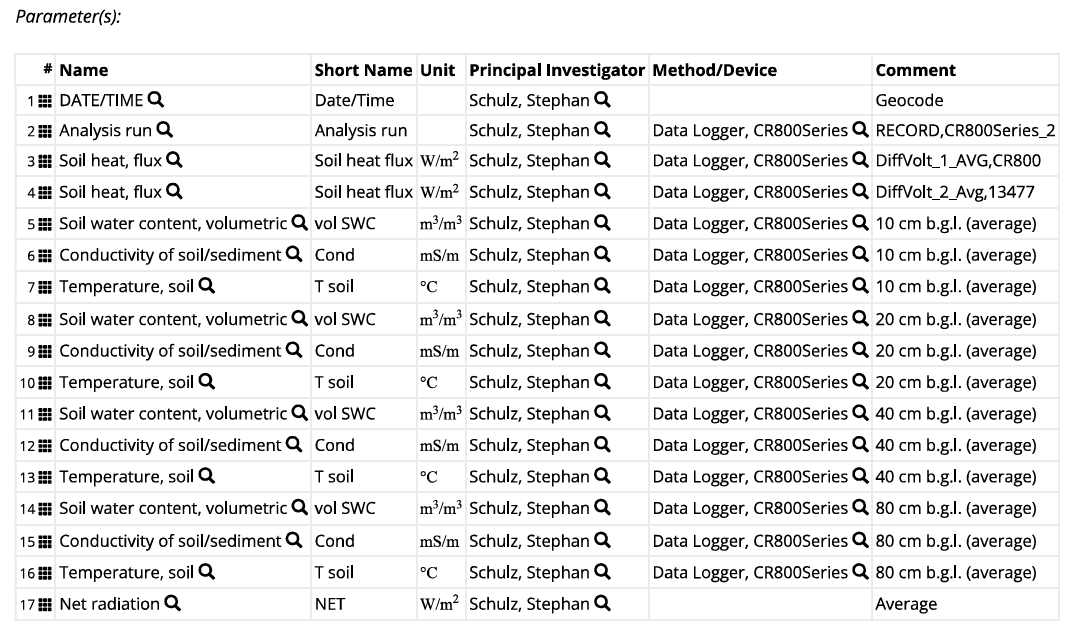
## 

## 4.2 TOOLS AND DATASET USED

| **Criteria** | **Tools to be used** | **Description** |
| --- | --- | --- |
| Data Collection & Preprocessing | Pandas & NumPy (Python Libraries) | For cleaning, preprocessing, and handling large sequential datasets efficiently. |
| Deep Learning Framework | TensorFlow / PyTorch, Keras  Darts | Implement and train the Artificial Neural Networks and Temporal Convolutional Networks (TCNs) ensemble model for modelling soil heat flux dynamics  A high-level API built on top of TensorFlow, suitable for designing and tuning the ANN, TCN architecture |
| Temporal Sequence Modeling | Temporal Convolutional Networks (TCNs) | Specialized for handling sequential data. This architecture will be used to capture complex temporal patterns in soil heat flux data. |
| Optimization | Adam / RMSprop Optimizers  Bayesian Optimizers | For tuning the weights of the deep learning model and optimizing the predictive performance of soil heat flux dynamics  Bayesian optimization is applied to optimize the Root Zone Temperature (RZT) toward an ideal value. |
| Visualization | Matplotlib / Seaborn | For visualizing the temporal patterns in the data, model performance and predictions of soil heat flux over time.  To analyze the optimization steps of root zone temperature visually. |

Table 4.2.1 Tools and Dataset Used

**Dataset used:**



Phase 2 of the system employs the forecasting results obtained from the ensemble model to train the Bayesian optimization model.

**4.3 DATASET DESCRIPTION**

The dataset contains soil and environmental measurements, including soil heat flux, water content, conductivity, temperature, and net radiation, collected at various depths and recorded using data loggers.

**Description of the attributes present in the dataset:**

* Date/Time:

Records the exact timestamp for each data entry, allowing for temporal analysis of soil and environmental variables.

* Analysis run:

Specifies each unique session or iteration of analysis, which may involve different processing or calibration steps.

* Soil heat flux:

Represents the rate of heat transfer per unit area in the soil (measured in W/m²), indicating energy movement within the soil, which can affect temperature and biological processes.

* Soil water content (volumetric):

Measures the volume of water in the soil as a fraction of the total soil volume (m³/m³), collected at various depths (10 cm, 20 cm, 40 cm, etc.), providing insights into soil moisture levels and water availability for plants.

* Conductivity of soil/sediment:

Indicates the soil’s ability to conduct electricity (measured in mS/m), which varies based on factors like soil texture, salinity, and moisture content; a useful parameter for assessing soil health and salinity levels.

* Temperature, soil:

Records the soil temperature at different depths (in °C), offering valuable data on thermal gradients, which influence root growth, microbial activity, and soil chemical processes.

* Net radiation:

The balance between incoming solar radiation and outgoing terrestrial radiation (measured in W/m²); it represents the net energy available at the surface, impacting evaporation, plant growth, and soil temperature dynamics.

## 

## 

## 

## 

## 

## 4.4 EXISTING MODELS

**1. Artificial Neural Networks**

Bruno César Comini de Andrade et al. tackle inefficiencies in traditional soil heat flux models by utilizing an Artificial Neural Network (ANN) to improve prediction accuracy across diverse land covers in South America. The ANN model, trained on data from 23 flux towers, reduces the mean absolute error by up to 43%. This highlights the ANN’s capability to capture complex, non-linear relationships between variables, underscoring the role of land cover information in enhancing predictive performance and deepening insights into soil-plant-atmosphere interactions. ***[1]***

**2.Temporal Convolutional Networks**

Yujie Liu et al. explore the potential of Temporal Convolutional Networks (TCNs) for time series prediction in their study. Acknowledging the shortcomings of traditional forecasting approaches, the authors focus on utilizing TCNs to effectively capture temporal dependencies in sequential data. Through this advanced deep learning architecture, the research aims to improve prediction accuracy and overcome challenges in modeling complex time series data. ***[4]***

**3.ARIMA, Prophet, LSTM, and hybrid models**

Rameshwar Garg et al. present an extensive survey of machine learning algorithms for time series analysis and forecasting. Highlighting the importance of accurate predictions across diverse domains, the study reviews models like ARIMA, Prophet, and Long Short-Term Memory (LSTM) networks. By examining both conventional and hybrid approaches, the authors provide key insights into recent developments in time series prediction techniques. ***[5]***

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 4.5 SYSTEM WORKFLOW

The system workflow is designed to integrate both Temporal Convolutional Networks (TCN) and Artificial Neural Networks (ANN) for forecasting. The process begins with data acquisition and preprocessing, after which the model development phase is initiated. The TCN model processes the input through convolution layers, dilation, and residual block connections before applying an activation function to produce an output tensor. Concurrently, the ANN model processes the input through hidden layers with neurons activated by ReLU. The outputs of both models are combined using stacking, leading to the development of a meta model, which ultimately generates the final predictions for forecasting.

**4.5.1 Temporal forecasting and analysis of soil heat flux values**

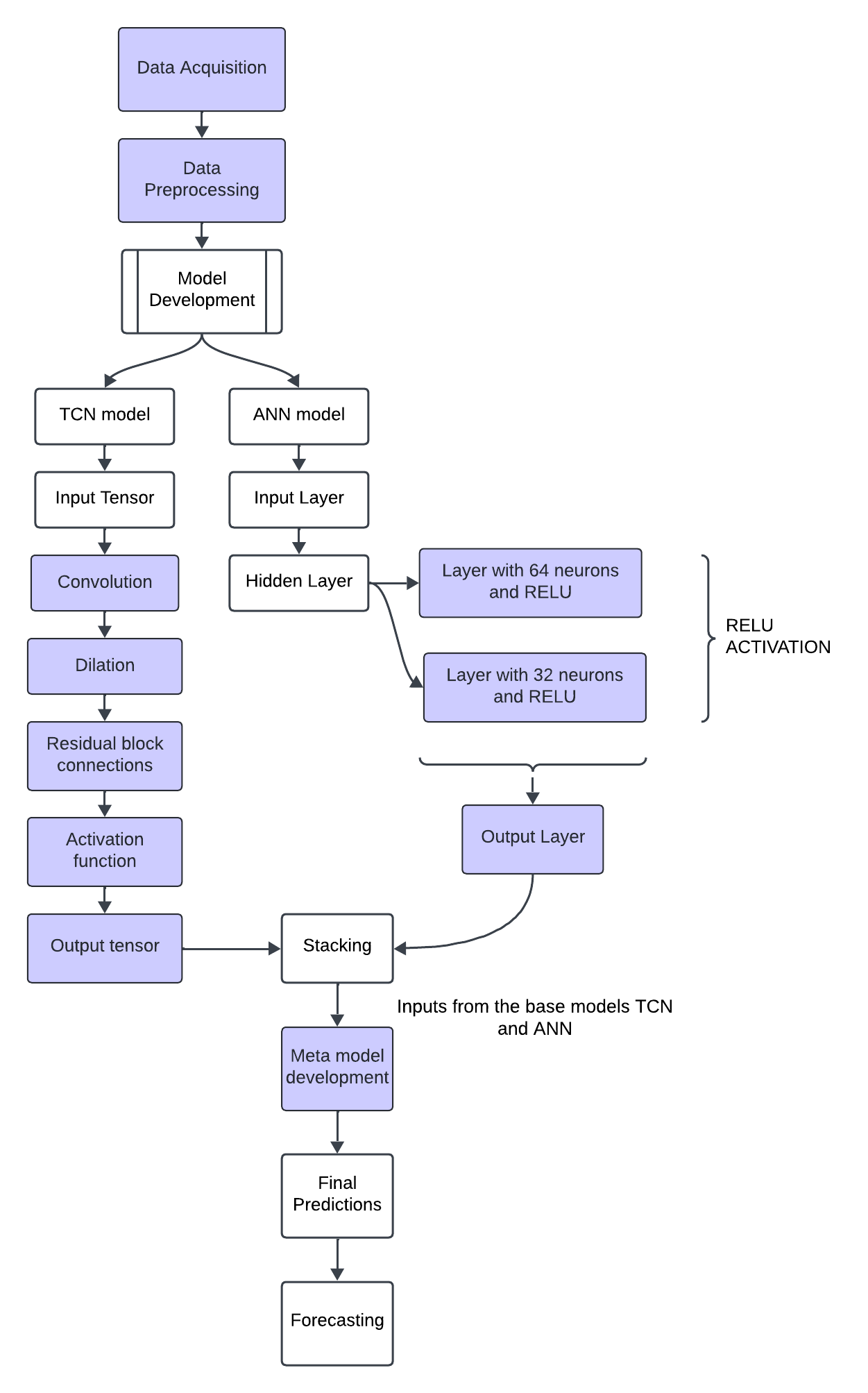


Figure 4.5.1 Flow diagram representing the workflow of the model

**Steps:**

1. **Data Collection:**

Gather soil and environmental data over time to create a dataset that captures variations in soil heat flux and related parameters.

1. **Data Preprocessing:**

Clean the dataset by handling missing values, removing unnecessary columns, scaling features, and selecting relevant features to improve model accuracy.

1. **Model Development:**

Develop separate Temporal Convolutional Network (TCN) and Artificial Neural Network (ANN) models to capture different types of patterns. The TCN is designed for temporal sequences, while the ANN focuses on feature relationships.

1. **TCN Model Architecture:**

The TCN model uses dilated convolutions and residual connections, allowing it to capture long-range dependencies in the data while maintaining efficiency.

1. **ANN Model Architecture:**

The ANN consists of an input layer and two hidden layers with 64 and 32 neurons, using the ReLU activation function to introduce non-linearity and model complex feature interactions.

1. **Model Stacking:**

Combine the outputs of the TCN and ANN models through stacking, creating a meta-model that leverages the strengths of both models for improved predictive performance. It is further explained in Fig. 4.4.1.2

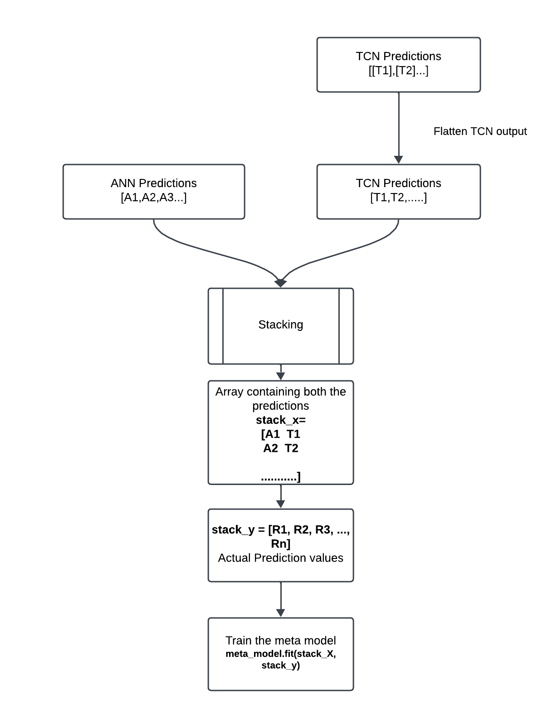


Fig.4.5.2 Flow diagram representing the stacking process to develop an ensemble model

1. **Final Forecasting:**

The meta-model generates final predictions of soil heat flux, enabling accurate forecasting for future data points based on historical patterns.

**4.5.2 Determination and optimization of plant root zone temperature**

Bayesian optimization utilizes a surrogate model, a Gaussian Process (GP), to approximate the objective function, which is the deviation of the Root Zone Temperature (RZT) from an ideal target of 25°C. The workflow is demonstrated in Fig. 4.4.2.1.

The objective function measures how far the RZT deviates from 25°C, adding penalties when RZT is too high or too low, and favoring values close to this ideal temperature. The goal of Bayesian optimization is to minimize the error in RZT over multiple iterations. Instead of evaluating the objective function (computationally expensive) directly each time, Bayesian optimization relies on this surrogate model to balance exploration of new regions and exploitation of known promising areas.

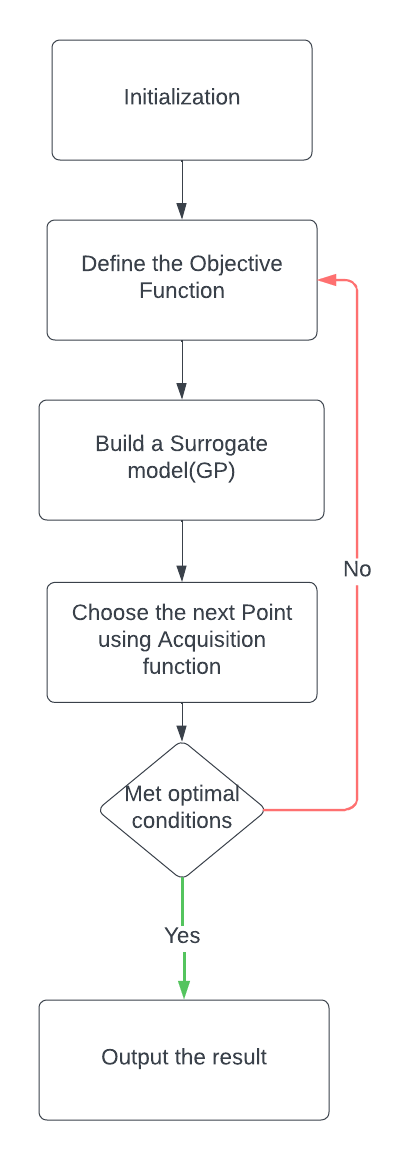


Fig. 4.5.3 Flow diagram representing the working of Bayesian Optimization

Based on data understanding, we realize that temperatures vary over different depths. Thus we employ exponential decay to determine weights associated with temperatures available at depths 10, 20, 40 and 80 cms. We also determine features that are highly correlated with temperature using the correlation heat map shown in Fig 4.4.2.2

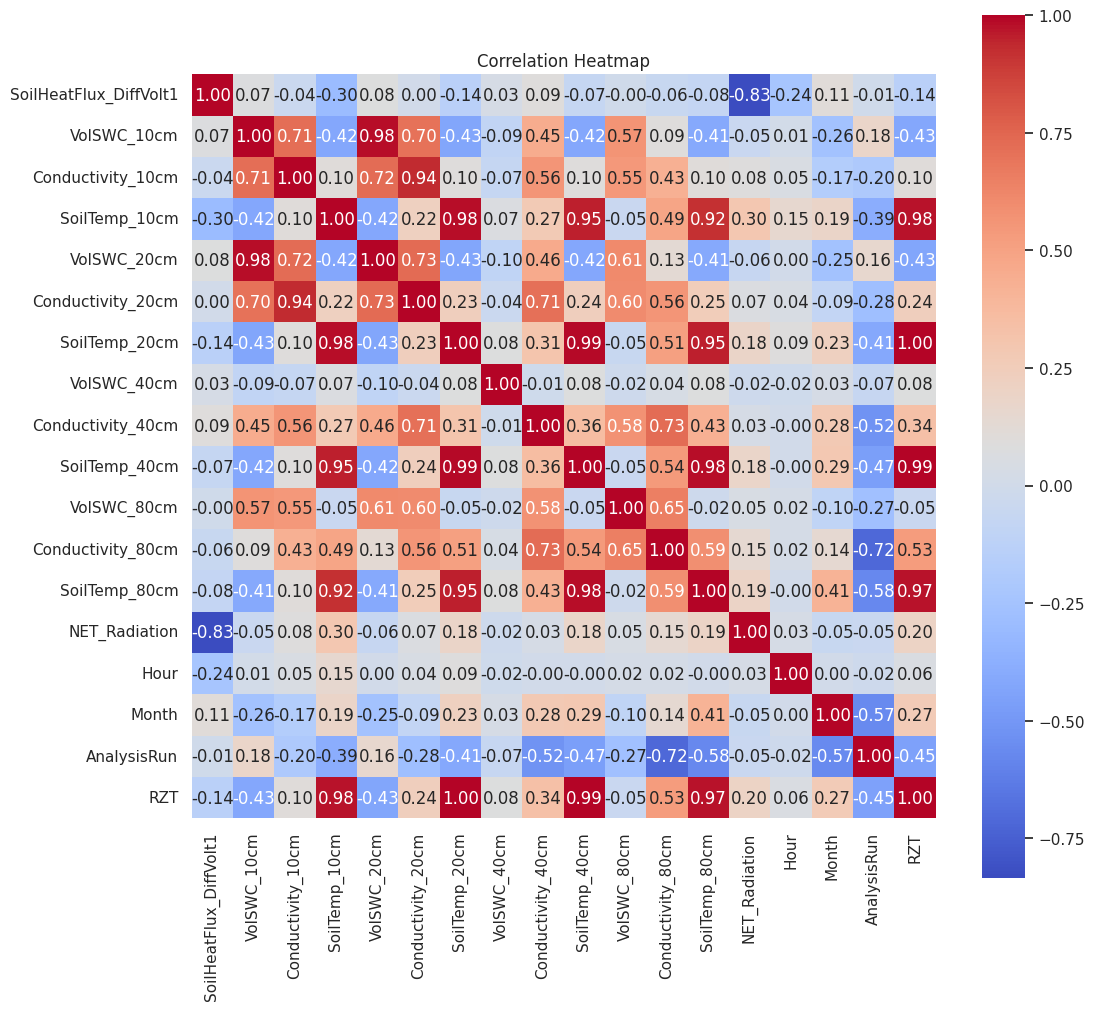


Fig. 4.5.4 Correlation heatmap illustrating the relationships between various soil parameters

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 4.6 PROPOSED SOLUTION

**4.6.1 Temporal forecasting and analysis of soil heat flux values**

**1. Temporal Convolution Network:**

* TCNs use 1D convolutional layers with causal convolutions to ensure that the output at time step t only depends on the inputs from time steps t and earlier, preserving the temporal order.
* TCNs incorporate dilated convolutions to expand the receptive field of the model exponentially, allowing it to capture long-range dependencies efficiently.
* These networks use residual connections to facilitate the training of deep networks, preventing the vanishing gradient problem.
* TCNs can handle variable-length input sequences and are particularly effective in tasks that involve temporal sequences, such as forecasting.
* By adjusting the dilation factor and the number of layers, TCNs can be tuned to capture both short-term and long-term patterns in the data.

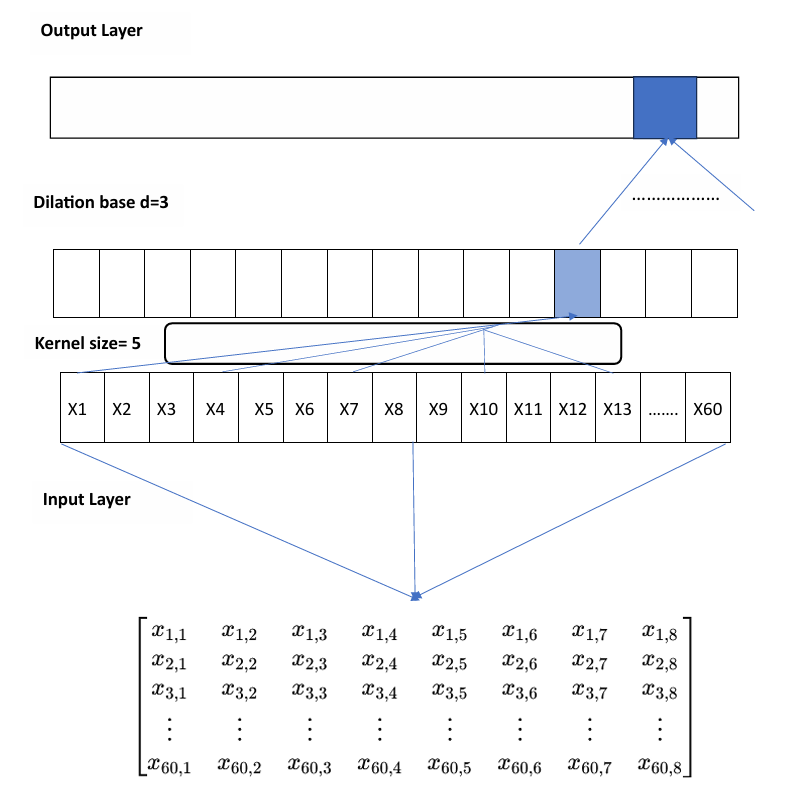
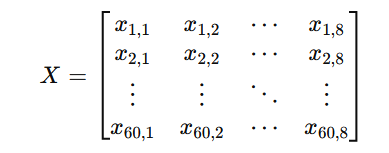


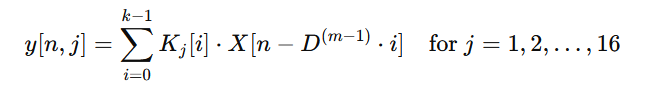
Figure 4.6.1 Architecture of Temporal Convolutional Network (TCN)

**Mathematical representation of TCN:**

* **Input Representation**: The matrix X shows input data with dimensions 60x8, representing time series data for 60 time steps and 8 features (e.g., volume, conductivity, net radiation, etc.).

 (1)

* **Convolution Operation**: The equation represents the convolution operation where y[n,j]y[n,j]y[n,j] is calculated by summing the product of a filter K over the input sequence X. The result is a feature map with 16 filters (represented by j).

 (2)

* **Dropout Application**: A dropout layer with a probability of p=0.05 is applied to prevent overfitting by randomly setting a fraction of input units y′ to zero during training.

 (3)

* **Output Layer Calculation**: The output layer calculation involves flattening the dropout result and applying a linear transformation with weights W and bias b to produce the final output prediction.

 (4)

**2. Artificial Neural Network:**

* ANNs are composed of layers of interconnected neurons, typically including input, hidden, and output layers, which allow for processing complex data.
* They learn by adjusting connection weights through backpropagation, minimizing the error between predicted and actual outputs.
* Non-linearity is introduced using activation functions such as sigmoid, ReLU, and tanh, enabling the network to capture complex patterns in data.

The model consists of an **input layer**, **two hidden layers**, and an **output layer**.

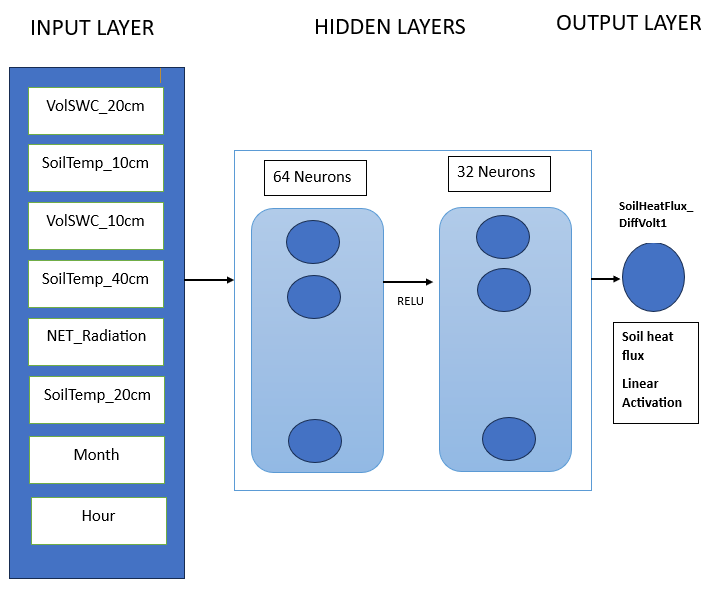


Figure 4.6.2 Architecture of Artificial Neural Network

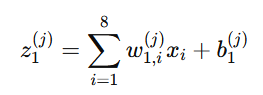
**Mathematical representation of ANN:**

**Input Vector**:

 (1)  
Represents the initial input features for the neural network.

**First Hidden Layer**:

* **Weighted Sum**

: (2)

Computes the weighted sum of inputs for each neuron in the first hidden layer.

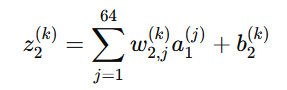
* **Activation**:

 (3)

Applies the ReLU activation to introduce non-linearity.

**Second Hidden Layer**:

* **Weighted Sum**:

 (4)

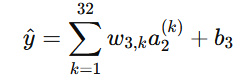
​ Calculates the weighted sum of activations from the first hidden layer.

* **Activation**:

 (5)

Uses ReLU to activate each neuron in the second hidden layer.

**Output Layer**:

**** (6)

**Stacking in Ensembling**

Stacking, also known as stacked generalization, is an ensemble learning technique that combines multiple models to improve predictive performance. In the context of the TCN-ANN stacked model, stacking involves using the outputs from both the Temporal Convolutional Network (TCN) and the Artificial Neural Network (ANN) as inputs to a meta-model, which learns how to best combine these predictions for final forecasting.

The main advantage of stacking is that it allows the model to leverage the strengths of different algorithms. The TCN captures temporal dependencies in time-series data, while the ANN learns complex relationships between features. By combining their outputs, the meta-model can provide more accurate predictions than any individual model alone.

**Algorithm: Stacking Ensemble Learning with TCN and ANN**

1. Initialize Base Models

- Define base models:

BaseModels = [TCN, ANN]

2. Train Base Models

For each model in BaseModels:

a. Initialize model parameters

b. For each sample in Train:

i. Perform forward propagation to generate predictions.

ii. Calculate loss using a suitable loss function.

iii. Perform backpropagation to update model parameters

c. Store predictions on Val set:

ValPredictions[model] = model.predict(Val)

3. Prepare Stacking Dataset

a. Create an empty dataset for meta-model:

StackingData = []

b. For each sample in Val:

i. Combine predictions from TCN and ANN:

CombinedPredictions = [ValPredictions[TCN][i], ValPredictions[ANN][i]]

ii. Append combined predictions and actual target value to StackingData:

StackingData.append(CombinedPredictions + [ActualTargetValue[i]])

5. Train Meta-Model

- Initialize meta-model (e.g., Linear Regression).

- Fit meta-model using StackingData:

MetaModel.fit(StackingData)

6. Final Predictions

For each new sample in Test:

a. Generate predictions from TCN:

TCN\_Prediction = TCN.predict(NewSample)

b. Generate predictions from ANN:

ANN\_Prediction = ANN.predict(NewSample)

c. Combine these predictions:

CombinedInput = [TCN\_Prediction, ANN\_Prediction]

d. Get final prediction from meta-model:

FinalPrediction = MetaModel.predict(CombinedInput)

7. End Algorithm

**4.6.2 Determination and optimization of plant root zone temperature**

**Algorithm for Bayesian Optimization using Gaussian Processes**

**1. Initialization**

Define the objective function f(x) that needs to be minimized. In your case, it evaluates the penalties based on deviations of RZT from the ideal temperature.

Set initial parameters for the optimization, including:

* Bounds for the RZT values based on the dataset.
* Initial sample points (RZT values) for training the Gaussian Process (GP) model.

**2.** **Define Objective Function**

Implement the objective function that quantifies the performance based on:

* Penalties for deviations from the ideal temperature (25°C).
* Strong penalties for values below the lower bound or above the upper bound.
* Encourage RZT to reach the ideal temperature by incorporating penalties for both under and overestimations.

**3. Gaussian Process Setup**

Initialize the Gaussian Process model:

* Choose a suitable kernel function (e.g., Radial Basis Function (RBF), Matern, etc.) for capturing the relationships in the data.
* Fit the GP model using the initial sample points and their corresponding objective function values.

**4. Acquisition Function**

Define the acquisition function (e.g., Expected Improvement (EI)) to decide where to sample next. The acquisition function helps balance exploration (trying new areas) and exploitation (refining known good areas).

The acquisition function should compute the expected improvement over the current best result.

**5.** **Optimization Loop**

Repeat the following steps until a stopping criterion is met (e.g., maximum iterations for convergence):

* **Sample New Point**: Use the acquisition function to select a new RZT value based on the GP model.
* **Evaluate Objective Function**: Compute the objective function value at the newly sampled RZT value.
* **Update GP Model**: Update the GP model with the new sample point and its objective function value.
* **Update Best Solution**: Keep track of the best RZT value encountered and its corresponding objective function value.

**6. Final Output**

Once the optimization loop is complete, output the best-found RZT value that minimizes the objective function, along with its associated performance metrics.

## 

## 

## 

## 

## 

## CHAPTER 5

**IMPLEMENTATION**

## 5.1 DATA UNDERSTANDING AND PREPROCESSING

**5.1.1 Temporal forecasting and analysis of soil heat flux values**

**Data Understanding**

The dataset is numeric, containing attributes primarily of type int and float, and consists of 425,262 entries. Several observations were made regarding data quality, distribution, and trends:

* Missing Values:

There are missing values in 10 columns, with around 10 missing entries overall. These were handled during preprocessing.

* Outliers and Data Cleaning:

The SoilHeatFlux\_DiffVolt1 column exhibited extreme values such as -9999, which needed to be addressed through cleaning.

SoilHeatFlux\_DiffVolt2 contained all values as zero, so this column was removed due to its lack of informational value.

Figure 4.3.1


Figure 5.1.1 Box plot of Soil Heat Flux by Net Radiation Ranges

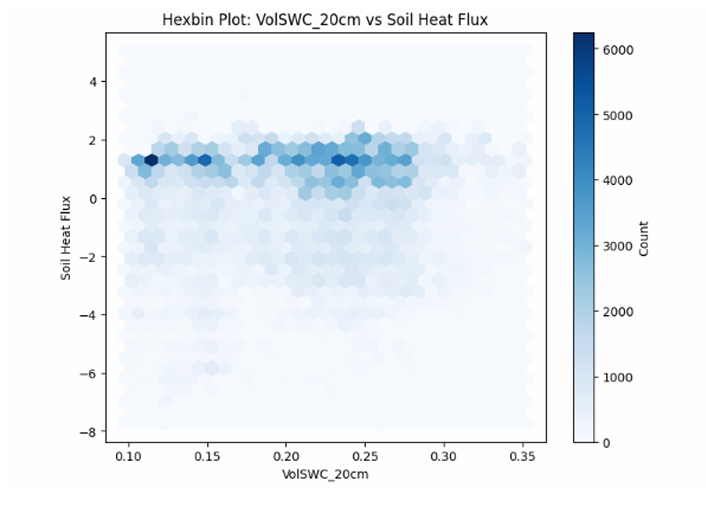


Figure 5.1.2 Hexbin plot showing the relationship between soil volume at a depth of 20 cm and soil heat flux

* Correlation Analysis:

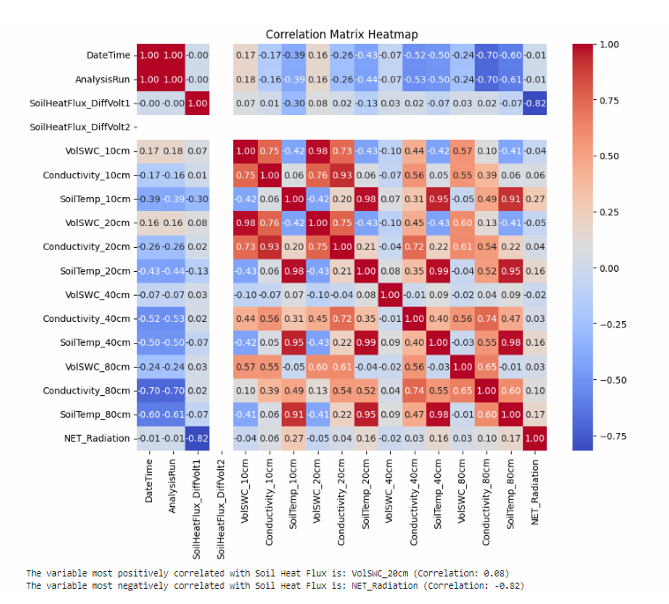
Using a correlation heat matrix, key columns were identified that significantly contribute to the model’s predictive capabilities.

Figure 5.1.3 Correlation heatmap illustrating the relationships between various soil parameters

* Distribution Insights:

The Soil Heat Flux distribution, visualized through histograms, showed a skewed pattern, indicating the presence of outliers. Box plots and hexbin plots further confirmed the presence of outliers and highlighted correlations between variables.

* Temporal and Diurnal Variations:

Diurnal Patterns: Soil heat flux tends to be most negative during the day due to surface heating by solar radiation, indicating downward heat conduction. At night, the values shift slightly upward as heat conducts from the soil to the surface.

Overall Temporal Trends: Soil heat flux generally stays negative, with occasional sharp increases to positive values, indicating brief heat gain periods.

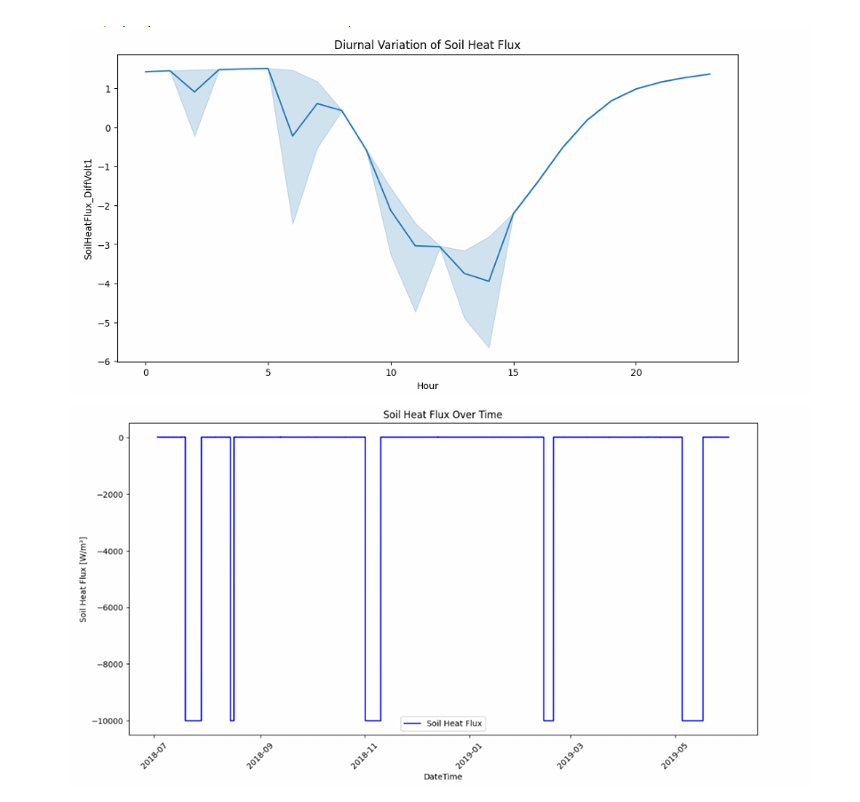


Figure 5.1.4 Diurnal variation of soil heat flux and Overall Temporal Trends

* Trend and Variability:

Rolling mean and standard deviation calculations showed periodic fluctuations in soil heat flux, with an overall downward trend indicating net heat loss.

Autocorrelation (ACF):

Analysis of the autocorrelation function (ACF) revealed flat correlations, suggesting a white noise process where values are largely random and uncorrelated.

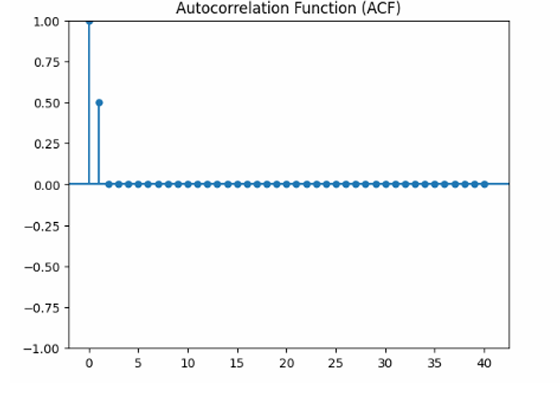


Figure 5.1.5 Autocorrelation function (ACF) plot showing correlation between the past and current values

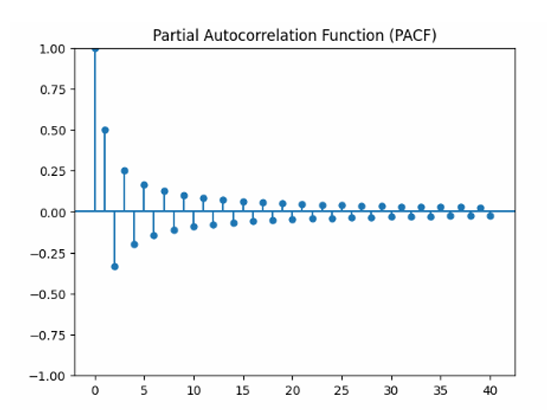


Figure 5.1.6 Partial autocorrelation function (PACF) plot illustrating the correlation while accounting for the effects of intervening observations

* Annual Analysis:

Seasonal variations in soil heat flux and related parameters were observed through annual plots, providing insights into how these features vary throughout the year.

**Data Preprocessing**

Several preprocessing steps were employed to prepare the data for modeling:

* Column Dropping and Renaming:

The SoilHeatFlux\_DiffVolt2 column was dropped due to constant zero values.

Columns were renamed for improved readability and understanding.

* Handling Missing Values:

Mean imputation was used to fill missing values, ensuring data consistency without introducing bias.

* Scaling:

A MinMax scaler was applied to normalize data across features, aiding model convergence.

* Outlier Removal:

Interquartile Range (IQR) filtering was used to detect and remove outliers, resulting in a cleaner dataset.

* Feature Selection:
* Univariate Analysis: Statistical methods such as Chi-squared and ANOVA tests were employed to identify the most significant features.
* Recursive Feature Elimination (RFE): An iterative approach was used to eliminate features with minimal impact on model performance.
* Random Forest Importance: This method assessed the significance of each feature by measuring the effect on model accuracy when features were randomly permuted.

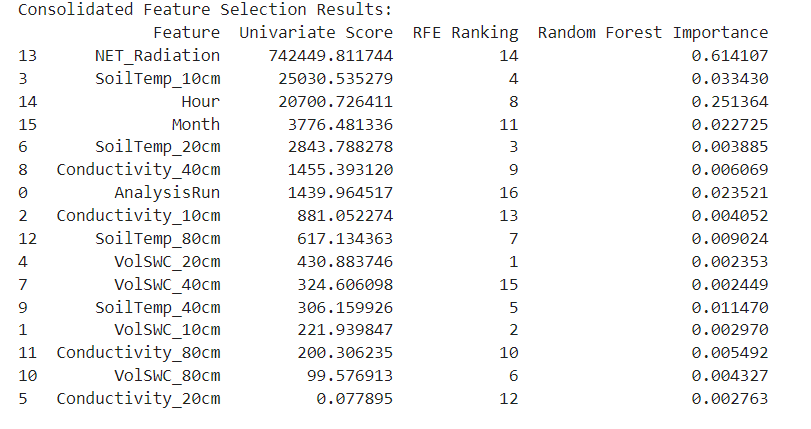


Figure 5.1.7 Results for the three Feature Selection Techniques

* Recommended Features:

Volume at 20 cm, Volume at 10 cm, Net Radiation, Conductivity, Soil Temperature at 20 cm, Soil Temperature at 10 cm, Soil Temperature at 80 cm, and DateTime.

* Resampling of time series data:

Resampling the Time Series dataset in an interval of 10 minutes to augment the number of data points in the training, testing, and validation sets.

**5.1.2 Determination and optimization of plant root zone temperature**

**Data Understanding**

We visualize how soil temperatures vary with depth over a specified time period. It helps in analyzing temperature trends, comparing depths and understanding how soil temperature might change with seasonal variations or other environmental factors.

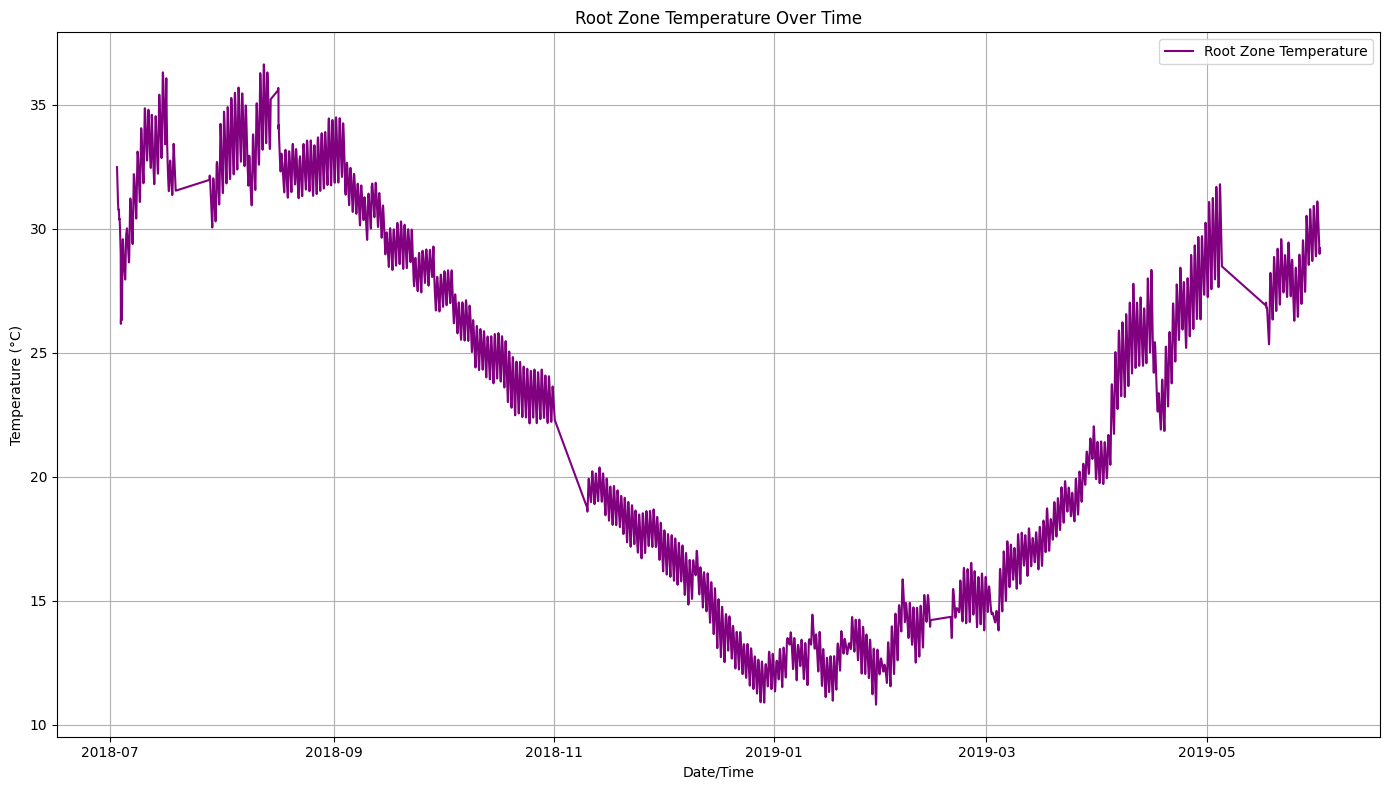
****

Figure 5.1.8 Plot showing the trend of Root Zone Temperature over Time (2018-2019)

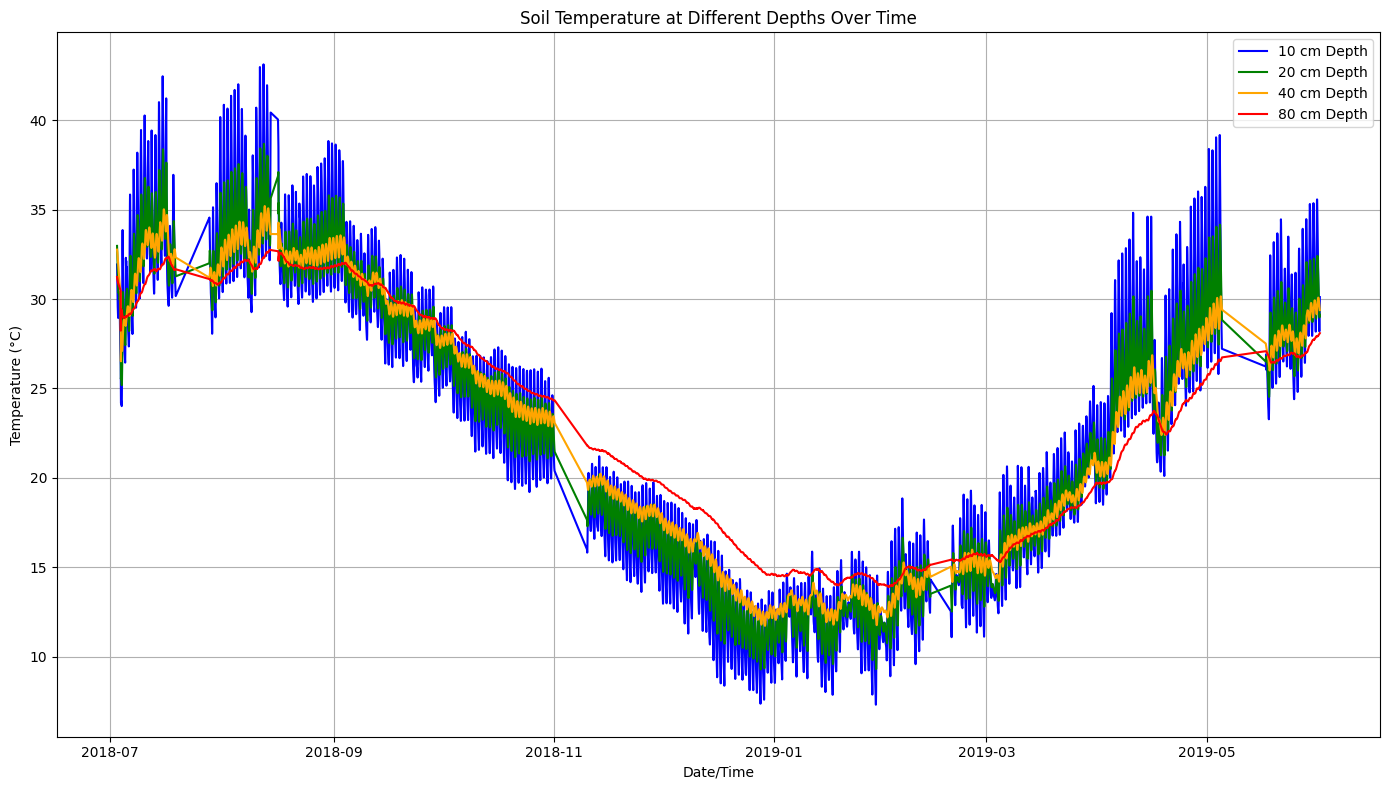


Figure 5.1.9 Plot showing the trend of Soil Temperatures at specific depths over Time (2018-2019)

**Analysis of Soil Temperature Patterns**

**1. Seasonal Variation**

Soil temperature exhibits a clear seasonal pattern across all depths, with temperatures reaching their peak during summer (mid-year) and declining in winter (end of the year). This seasonal trend reflects the influence of external climatic conditions on soil thermal dynamics.

**2. Temperature Decrease with Depth**

Observations indicate that shallower depths (10 cm) experience greater temperature fluctuations. This responsiveness is attributed to the direct influence of air temperature changes. Conversely, deeper layers (e.g., 40 cm and 80 cm) exhibit more stable temperature readings, suggesting that they are less susceptible to immediate environmental variations.

**3. Time Lag with Depth**

A notable time lag is observed in temperature changes at deeper layers (e.g., 80 cm) compared to shallower layers. This delay highlights the gradual penetration of heat through the soil profile, which affects how quickly deeper soil temperatures respond to surface changes.

**4. Moderation Effect**

Deeper soil layers (40 cm and 80 cm) display fewer temperature spikes, contributing to a more stable environment for plant roots. This stability is crucial, as it helps mitigate the impact of surface temperature fluctuations, thereby fostering better growth conditions for root systems.

**5. Overall Trends**

The analysis reveals that deeper soil layers undergo gradual seasonal changes, while shallower layers closely reflect immediate variations in air temperature. This disparity emphasizes the different roles that various soil depths play in the overall thermal environment.

**Weighted Average for Root Zone Temperature (RZT)**

The approach to calculating the Root Zone Temperature (RZT) involves assigning weights to soil temperatures at various depths to reflect their influence on plant roots:

• **Heavier Weight on Intermediate Depths (20 cm and 40 cm)**: These layers strike a balance between responsiveness to surface conditions and stability, making them critical for optimal root growth.

• **Lower Weight on Surface Depth (10 cm)**: While surface temperatures fluctuate significantly, their immediate effects diminish with depth.

• **Moderate Weight on Deep Depth (80 cm)**: This depth provides stability but is less responsive to rapid environmental changes.

**Data Preprocessing**

* **NaN Handling**: Any rows with NaN values are filtered out to avoid issues in metrics calculations and optimization. This ensures only valid data points are used in further processing.
* **Exponential Decay Weighting**: Soil temperatures at four depths (10 cm, 20 cm, 40 cm, and 80 cm) are assigned weights using exponential decay, assuming the influence of shallower depths on RZT is more significant. The decay rate (alpha = 0.3) controls how quickly influence decreases with depth. These weights are normalized to ensure they sum to 1, allowing a proper balance in the contribution of each depth.
* **Time-of-Day Segmentation**: Data is segmented into four intervals (Night, Morning, Afternoon, and Evening) based on the hour of the day, allowing for analysis of temperature patterns across different times.
* **RZT Calculation**: Using the weighted sum of soil temperatures, the RZT is computed for each timestamp, yielding a consolidated temperature representing the root zone.
* **Dynamic Bounds Setup**: Dynamic bounds for RZT are calculated, using the existing RZT values to establish realistic limits for the optimization process.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 5.2 ALGORITHM

**Algorithm for ANN:**

1. Initialize ANN Model

- Set input size to 7 (VolSWC\_20cm, SoilTemp\_10cm, VolSWC\_10cm, SoilTemp\_40cm, NET\_Radiation, SoilTemp\_20cm, Month, Hour)

2. Define ANN Architecture

- Input Layer: 7 features

- Hidden Layer 1:

- Neurons: 64

- Activation Function: ReLU

- Hidden Layer 2:

- Neurons: 32

- Activation Function: ReLU

- Output Layer:

- Neurons: 1 (Soil Heat Flux)

- Activation Function: Linear

3. For each epoch in range(1, num\_epochs):

a. For each training sample in the dataset:

i. Input features into the Input Layer:

X = [VolSWC\_20cm, SoilTemp\_10cm, VolSWC\_10cm, SoilTemp\_40cm, NET\_Radiation, SoilTemp\_20cm, Month, Hour]

ii. Forward Propagation:

Hidden\_Layer1\_Output = ReLU(W1 \* X + b1)

Hidden\_Layer2\_Output = ReLU(W2 \* Hidden\_Layer1\_Output + b2)

Output = Linear(W3 \* Hidden\_Layer2\_Output + b3)

iii. Calculate Loss:

Loss = (Predicted\_Soil\_Heat\_Flux - Actual\_Soil\_Heat\_Flux)^2

b. Backward Propagation:

i. Compute gradients of the loss with respect to weights and biases (W1, b1, W2, b2, W3, b3)

ii. Update weights using an optimization algorithm (e.g., Gradient Descent or Adam)

4. End For

5. After training is complete, Use trained model for prediction on new input data:

Soil\_Heat\_Flux\_Prediction = ANN\_Predict([New\_Features])

6. End Algorithm

**Algorithm for TCN:**

1. Initialize Parameters

- Define input matrix X of size (sequence\_length, features), e.g., X ∈ R^(60×8)

- Define convolutional filters K\_j for each output channel j

- Set dilation factor D(m-1) for each layer m

2. Input Representation

- Organize the input data X as a matrix with time steps as rows and features as columns.

3. For each epoch in range(1, num\_epochs):

a. For each training sample in the dataset:

i. For each output channel j in the layer:

A. For each time step n:

1. Compute output y[n, j] as follows:

y[n, j] = Σ (from i=0 to k-1) K\_j[i] \* X[n - D(m-1) \* i]

where k is the filter size.

b. Apply Dropout:

i. y' = Dropout(y, p=0.05)

c. Flatten and Output Layer Calculation:

i. Output = f(W \* Flatten(y') + b)

where W is the weight matrix and b is the bias.

d. Calculate Loss:

i. Loss = (Predicted\_Output - Actual\_Output)^2

e. Backward Propagation:

i. Compute gradients of the loss with respect to weights and biases (W, b)

ii. Update weights using an optimization algorithm (e.g., Gradient Descent or Adam)

4. End For

5. After training is complete:

a. Use trained model for prediction on new input data:

Soil\_Heat\_Flux\_Prediction = TCN\_Predict([New\_Features])

6. End Algorithm

**Algorithm: TCN-ANN Stacked Model for Forecasting**

1. Data Acquisition

- Collect and organize the dataset required for forecasting.

2. Data Preprocessing

a. Clean and preprocess the data as needed.

b. Perform scaling, normalization, or feature engineering.

c. Split the dataset into training, validation, and test sets.

3. Model Development

- Develop separate models: TCN model and ANN model.

4. TCN Model Development

a. Input Representation:

- Represent input data as a tensor suitable for TCN (multi-dimensional time series).

b. Convolution Operation:

- For each layer in the TCN:

i. Apply convolution operations with varying dilation factors to capture temporal dependencies.

c. Residual Block Connections:

- Implement residual blocks to ensure stable learning.

d. Activation Function:

- Apply activation functions (e.g., ReLU) after each convolution layer.

e. Output Tensor:

- Obtain the output tensor from the TCN model representing learned features.

5. ANN Model Development

a. Input Layer:

- Initialize the ANN model to accept specified input features.

b. Hidden Layers:

- Layer 1: 64 neurons with ReLU activation.

- Layer 2: 32 neurons with ReLU activation.

c. Output Layer:

- Apply linear activation function to predict the target variable (e.g., Soil Heat Flux).

6. Stacking

a. Combine output tensors from TCN and ANN models.

b. Feed combined outputs as inputs into a meta-model.

7. Meta Model Development

a. Develop a meta-model that takes inputs from both TCN and ANN outputs.

b. Train the meta-model to make final predictions based on combined learned features.

8. Final Predictions

- Use the trained meta-model to generate final predictions for the target variable.

9. Forecasting

a. Apply the entire stacked model pipeline to new data for future forecasting.

b. Output forecasted values as final results.

10. End Algorithm

**Algorithm - RZT Optimization using Bayesian Optimization**

**Import Required Libraries**:

Import the necessary libraries: pandas, numpy, scipy, skopt, matplotlib, and tqdm.

**Load the Dataset**:

Define the function load\_data(file\_path) to load the dataset:

Use pd.read\_excel() to read the data.

Convert the DateTime column to datetime format.

Set DateTime as the index.

Call the function with the specified file path to load the data.

**Define Constants**:

Specify the depths of soil temperatures (10 cm, 20 cm, 40 cm, and 80 cm).

Set the decay constant alpha = 0.3 for the exponential decay weights.

**Calculate Exponential Decay Weights**:

For each depth, compute raw weights using the formula:

Normalize the weights so that their sum equals 1.

**Map Normalized Weights to Soil Temperature Columns**:

Create a dictionary weights that associates each soil temperature column with its corresponding normalized weight.

**Calculate Root Zone Temperature (RZT)**:

Define the function calculate\_rzt(data, weights):

Compute the RZT as a weighted sum of soil temperatures:

Call this function to add the RZT column to the dataset.

**Segment Data by Time of Day**:

Define time intervals: Night (before 6 AM), Morning (6 AM to 12 PM), Afternoon (12 PM to 5 PM), Evening (after 5 PM)

Create a new column TimeOfDay in the dataset based on these intervals.

**Calculate Minimum and Maximum RZT Values**:

Group the dataset by TimeOfDay and calculate the minimum and maximum RZT values for each segment.

Print the minimum and maximum RZT values before optimization.

**Define Dynamic Bounds for Optimization**:

Set lower and upper bounds for RZT:

Lower bound: maximum of 10 or minimum RZT value.

Upper bound: minimum of 35 or maximum RZT value + 5.

Store the bounds in a list: bounds = [(lower\_bound, upper\_bound)].

**Define Objective Function**:

Implement the function comprehensive\_objective(rzt):

Initialize an empty list penalties.

Extract the RZT value from the input list rzt.

For each row in the dataset:

Compare the RZT value against the ideal temperature range (18°C to 25°C) and bounds.

Append penalties to the penalties list based on the conditions specified:

Strong penalties for values below the lower bound or above the upper bound.

Moderate penalties for values outside the ideal temperature range.

Handle potential NaN values in penalties.

Return the mean of the penalties.

**Define Optimization Function**:

Implement the function optimize\_row(row):

Use gp\_minimize from the skopt library:

Pass the comprehensive\_objective as the objective function.

Specify the bounds, number of calls (n\_calls=20), acquisition function (acq\_func='EI'), random state, and initial value from the current row’s RZT.

Return the optimized RZT value.

**Run Bayesian Optimization**:

Initialize an empty list optimized\_rzts to store optimized RZT values.

Iterate over each row in the dataset using tqdm for a progress bar:

Call optimize\_row(row) for each row to get the optimized RZT.

Append the result to optimized\_rzts.

**Add Optimized Values to DataFrame**:

Create a new column Optimized\_RZT in the dataset to store the optimized values.

**Display Results After Optimization**:

Print the optimized minimum and maximum RZT values for each time of day segment after optimization.

## 

## 5.3 RESULT ANALYSIS

**Performance metrics for the Ensemble model:**

### MSE (Mean Squared Error):

* MSE measures the average squared difference between the predicted values and the actual values. It penalizes larger errors more than smaller ones, making it useful when large errors are particularly undesirable.
* **Formula**

MSE**=**  (1)

| **Epochs** | **ANN** | **TCN** | **TCN + ANN** |
| --- | --- | --- | --- |
| 100 | 0.2885 | 3.9944 | 0.2901 |
| 75 | 0.3060 | 9.1605 | 0.2866 |
| 50 | 0.4318 | 3.4131 | 0.1559 |
| 25 | 0.3305 | 3.7179 | 0.1563 |
| 10 | 0.4456 | 3.7188 | 0.2834 |

Table 5.3.1 Performance Metrics - MSE

### MAE (Mean Absolute Error):

* MAE calculates the average of the absolute differences between predicted and actual values. It provides a linear score without the exaggeration caused by squaring the errors.
* **Formula**

MAE **= |** (2)

| **Epochs** | **ANN** | **TCN** | **TCN+ANN** |
| --- | --- | --- | --- |
| 100 | 0.4124 | 1.4298 | 0.1713 |
| 75 | 0.4151 | 2.4254 | 0.1729 |
| 50 | 0.4226 | 1.3366 | 0.2935 |
| 25 | 0.4664 | 1.3659 | 0.2948 |
| 10 | 0.4778 | 1.3498 | 0.1468 |

Table 5.3.2 Performance Metrics - MAE

### R-squared (Coefficient of Determination):

* R squared is a statistical measure that shows the proportion of variance in the dependent variable that is predictable from the independent variable(s). It gives an idea of how well the model captures the relationship between the predictors and the target.
* **Formula**

**=**  (3)

**->** Residual sum of squares

**->** Total sum of squares

| **Epochs** | **ANN (%)** | **TCN** | **TCN+ANN (%)** |
| --- | --- | --- | --- |
| 100 | 93.56 | -0.3619 | 94.16 |
| 75 | 85.26 | -3.1234 | 94.10 |
| 50 | 90.97 | -0.1637 | 94.69 |
| 25 | 92.70 | -0.2676 | 94.67 |
| 10 | 91.09 | -0.2679 | 94.99 |

Table 5.3.3 Performance Metrics - R-squared

**Results**

**Predictions for Validation set**

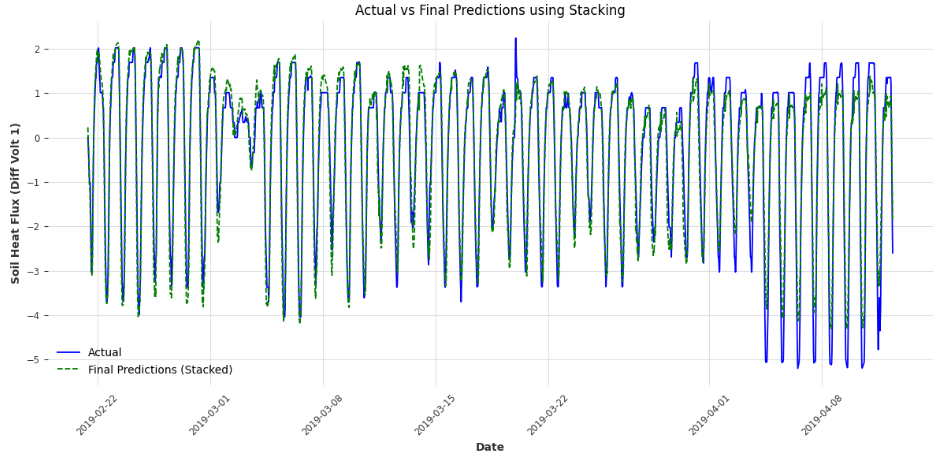
****

Figure 5.2.1 Plot showing the Actual vs Final Predictions of the ensemble model for the validation set

**Predictions for Testing set**

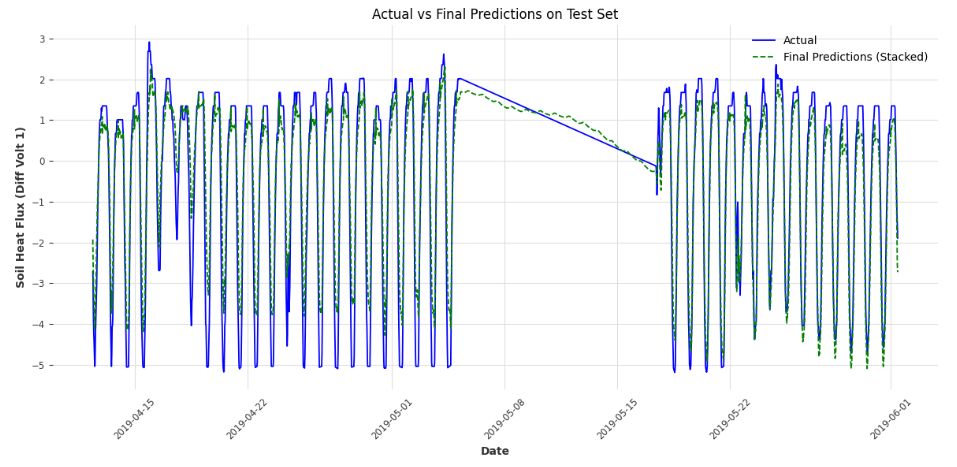
****

Figure 5.2.2 Plot showing the Actual vs Final Predictions of the ensemble model for the test set

**Future predictions - Forecasting**

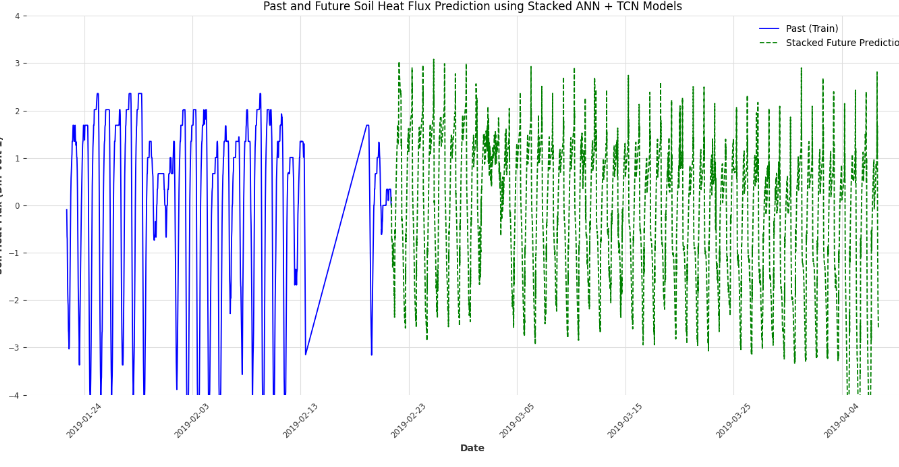
****

Figure 5.2.3 Plot showing the past and forecasted soil heat flux values predicted using ensemble model

**‘**

**Results for RZT Optimization Module:**

Before Optimization:

Afternoon - Min RZT: 17.25°C, Max RZT: 32.62°C

Evening - Min RZT: 17.26°C, Max RZT: 32.45°C

Morning - Min RZT: 17.24°C, Max RZT: 35.01°C

Night - Min RZT: 17.25°C, Max RZT: 33.31°C

After Optimization:

Morning - Optimized Min RZT: 18.00°C, Optimized Max RZT: 24.93°C

Afternoon - Optimized Min RZT: 18.00°C, Optimized Max RZT: 24.59°C

Evening - Optimized Min RZT: 18.01°C, Optimized Max RZT: 24.99°C

Night - Optimized Min RZT: 18.01°C, Optimized Max RZT: 24.08°C

Metrics and Results:

RMSE: 1.7582070426846619

MAE: 1.5220519743817684

MAPE: 0.07079311508752412

SMAPE: 0.07395064807160293

Max Error: 3.4999488812673683

Percentage of values within the optimal range (18-25°C): 100.00%

Highest Accuracy is **99.44%** and is achieved at threshold **±3.450°C**

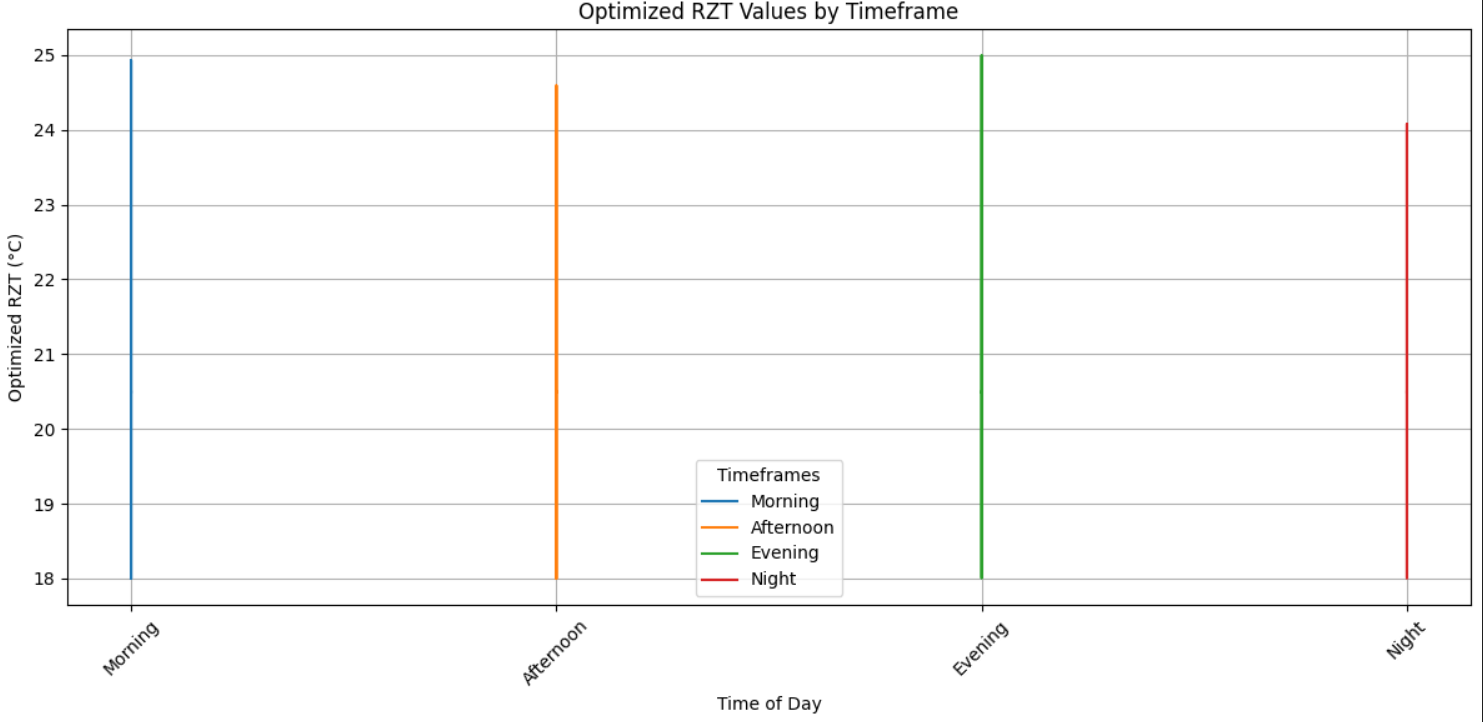
****

Figure 5.2.4 Plot showing the optimized values of the temperatures in four timeframes

**CHAPTER 6**

**CONCLUSION**

In conclusion, the project successfully developed an ensemble model for forecasting soil heat flux dynamics, which is crucial for optimizing plant root zone temperature and enhancing agricultural productivity. The primary models considered were Temporal Convolutional Networks (TCNs) and Artificial Neural Networks (ANNs). While the TCN model effectively captured long-range temporal patterns through dilated convolutions, its performance varied across different epochs. The ANN model, which excelled in capturing non-linear feature interactions, achieved an accuracy of 93.56%, demonstrating its capability to process data but not fully addressing the temporal aspects as effectively as the TCN.

To further improve predictive performance, an ensemble model combining TCN and ANN was developed, leveraging the temporal strengths of TCN alongside the general feature-learning capacity of ANN. This ensemble model achieved an accuracy of 94.69%, showing notable improvements in prediction accuracy compared to individual models. Evaluation metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared values revealed that the ensemble approach provided more accurate predictions for soil heat flux dynamics.

Additionally, Bayesian optimization was introduced to refine the optimization of plant root zone temperature. The initial RZT was determined using an exponential decay method, providing a robust starting point for the optimization process. By integrating Bayesian optimization with the ensemble model, we could effectively explore and exploit the parameter space, leading to enhanced accuracy in predicting soil temperature variations.

Ultimately, while both TCN and ANN models contributed valuable insights, the TCN+ANN ensemble model combined with Bayesian optimization emerged as the best-performing solution for predicting soil heat flux. This approach not only offered superior accuracy and reliability in forecasting soil temperature variations but also demonstrated significant potential for advancing agricultural practices through optimized management of soil resources.

## 

## CHAPTER 7

## BIBLIOGRAPHY

**[1]** *B. C. C. de Andrade, O. C. Pedrollo, A. Ruhoff, and A. A. Moreira, "Artificial Neural Network Model of Soil Heat Flux over Multiple Land Covers in South America”*

**[2]** *L. C. Romio, D. R. Roberti, L. Buligon, T. Zimmer, and G. A. Degrazia, "A Numerical Model to Estimate the Soil Thermal Conductivity Using Field Experimental Data"*

**[3]** *J. F. Cross and D. T. Drewry, "Ensemble Machine Learning for Interpretable Soil Heat Flux Estimation"*

**[4]** *Y. Liu, H. Dong, X. Wang, and S. Han, "Time Series Prediction Based on Temporal Convolutional Network”*

**[5]** *R. Garg, S. Barpanda, G. R. Salanke N. S., and R. Shivamadegowda, "Machine Learning Algorithms for Time Series Analysis and Forecasting”*

**[6]** *R. Li, B. Yin, Y. Cong, and Z. Du, "Simultaneous Prediction of Soil Properties Using Multi\_CNN Model"*

**[7]** *A. Liang, C. Xie, J. Wang, and S. Che, "Daily Dynamics of Soil Heat Flux and Its Relationship with Net Radiation in Different Urban Riparian Woodlands"*

**[8]** *T. A. Adjuik and S. C. Davis, "Machine Learning Approach to Simulate Soil CO2 Fluxes under Cropping Systems"*

**[9]***Z. P. Mabunga and J. C. Dela Cruz, "An Optimized Soil Moisture Prediction Model for Smart Agriculture Using Gaussian Process Regression"*

**[10]** *A. Liang, C. Xie, J. Wang, and S. Che, "Raising Root Zone Temperature Improves Plant Productivity and Metabolites in Hydroponic Lettuce Production"*

**[11]** <https://github.com/bayesian-optimization/BayesianOptimization>

**[12]** <https://unit8.com/resources/temporal-convolutional-networks-and-forecasting/>

## CHAPTER 8

## APPENDIX

**Ensemble model of TCN and ANN**

| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Dense  from sklearn.preprocessing import StandardScaler  from sklearn.linear\_model import LinearRegression  from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score  from darts import TimeSeries  from darts.models import TCNModel  # Function to load data  def load\_data(train\_file, val\_file):  train\_df = pd.read\_csv(train\_file)  val\_df = pd.read\_csv(val\_file)  # Convert 'DateTime' column to datetime and set as index  for df in [train\_df, val\_df]:  df['DateTime'] = pd.to\_datetime(df['DateTime'])  df.set\_index('DateTime', inplace=True)    return train\_df, val\_df  # Load CSV data files (adjust paths as needed)  train\_file = '/kaggle/input/forecastingtcn/train\_transformed\_10.csv'  val\_file = '/kaggle/input/forecastingtcn/val\_transformed\_10.csv'  train\_df, val\_df = load\_data(train\_file, val\_file)  # Select features and target  features = ['Month', 'VolSWC\_20cm', 'Hour', 'SoilTemp\_10cm', 'VolSWC\_10cm',  'SoilTemp\_40cm', 'NET\_Radiation', 'SoilTemp\_20cm']  target = 'SoilHeatFlux\_DiffVolt1'  X\_train = train\_df[features].values  y\_train = train\_df[target].values  X\_val = val\_df[features].values  y\_val = val\_df[target].values  # Normalize the data for ANN  scaler = StandardScaler()  X\_train\_scaled = scaler.fit\_transform(X\_train)  X\_val\_scaled = scaler.transform(X\_val)  # Build and train ANN model  ann\_model = Sequential()  ann\_model.add(Dense(64, activation='relu', input\_dim=X\_train\_scaled.shape[1]))  ann\_model.add(Dense(32, activation='relu'))  ann\_model.add(Dense(1, activation='linear')) # Output for regression  ann\_model.compile(optimizer='adam', loss='mse')  ann\_model.fit(X\_train\_scaled, y\_train, validation\_data=(X\_val\_scaled, y\_val), epochs=100, batch\_size=32, verbose=1)  # Get ANN predictions for the training and validation sets  ann\_train\_predictions = ann\_model.predict(X\_train\_scaled).flatten()  ann\_val\_predictions = ann\_model.predict(X\_val\_scaled).flatten()  # Prepare TimeSeries objects for TCN including ANN predictions  train\_df['ANN\_Predictions'] = ann\_train\_predictions  val\_df['ANN\_Predictions'] = ann\_val\_predictions  train\_series\_tcn = TimeSeries.from\_dataframe(train\_df[features + [target, 'ANN\_Predictions']], freq='10T')  val\_series\_tcn = TimeSeries.from\_dataframe(val\_df[features + [target, 'ANN\_Predictions']], freq='10T')  # Initialize and train TCN model with ANN predictions as input  tcn\_model = TCNModel(  input\_chunk\_length=60,  output\_chunk\_length=1,  n\_epochs=50,  dropout=0.05,  dilation\_base=3,  weight\_norm=True,  kernel\_size=5,  num\_filters=16,  random\_state=0  )  # Fit the TCN model using the training series with ANN predictions  tcn\_model.fit(series=train\_series\_tcn[target], val\_series=val\_series\_tcn[target], verbose=True)  # Get TCN predictions  tcn\_predictions = tcn\_model.predict(n=len(y\_val), series=train\_series\_tcn[target])  tcn\_predictions\_np = tcn\_predictions.values()  # Stack predictions  stack\_X = np.column\_stack((ann\_val\_predictions, tcn\_predictions\_np.flatten()))  stack\_y = y\_val # Actual target values  # Train a simple linear regression model on the stacked predictions  meta\_model = LinearRegression()  meta\_model.fit(stack\_X, stack\_y)  # Predict using the meta-model  final\_predictions = meta\_model.predict(stack\_X)  # Calculate metrics  mae = mean\_absolute\_error(y\_val, final\_predictions)  mse = mean\_squared\_error(y\_val, final\_predictions)  r2 = r2\_score(y\_val, final\_predictions)  # Print the metrics  print(f'Mean Absolute Error (MAE): {mae:.4f}')  print(f'Mean Squared Error (MSE): {mse:.4f}')  print(f'R-squared (R²): {r2:.4f}')  # Plot actual vs final predictions  plt.figure(figsize=(12, 6))  plt.plot(val\_df.index, y\_val, label='Actual', color='blue')  plt.plot(val\_df.index, final\_predictions, label='Final Predictions (Stacked)', color='green', linestyle='--')  plt.title('Actual vs Final Predictions using Stacking')  plt.xlabel('Date')  plt.ylabel('Soil Heat Flux (Diff Volt 1)')  plt.legend()  plt.xticks(rotation=45)  plt.tight\_layout()  plt.show() |
| --- |

| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import matplotlib.dates as mdates  from matplotlib.ticker import MaxNLocator  from sklearn.linear\_model import LinearRegression  # Ensure the 'DateTime' column in train\_df is a DateTimeIndex  if not isinstance(train\_df.index, pd.DatetimeIndex):  train\_df['DateTime'] = pd.to\_datetime(train\_df['DateTime'])  train\_df.set\_index('DateTime', inplace=True)  # Define the number of past observations to plot  n\_past = 144\*30 # Adjust this value as needed, e.g., 144 for 24 hours of 10-minute data  # Number of future steps to predict  n\_future\_steps = 144 \* 45 # For example, 30 days of 10-minute intervals  # Get TCN predictions  tcn\_predictions = tcn\_model.predict(n=n\_future\_steps, series=train\_series[target])  tcn\_predictions\_np = tcn\_predictions.values()  # Get ANN predictions (ensure the correct shape)  ann\_predictions = ann\_model.predict(X\_val[:n\_future\_steps]).flatten() # Flatten if needed  # Create a stacked dataset for training the meta-model  stacked\_X = np.column\_stack((ann\_predictions, tcn\_predictions\_np.flatten()))  stacked\_y = y\_val[:n\_future\_steps] # Use actual target values for training  # Train the meta-model (Linear Regression)  meta\_model = LinearRegression()  meta\_model.fit(stacked\_X, stacked\_y)  # Make predictions using the meta-model for future data  future\_ann\_predictions = ann\_model.predict(X\_val[:n\_future\_steps]).flatten() # Use same input structure  future\_tcn\_predictions = tcn\_model.predict(n=n\_future\_steps, series=train\_series[target]).values().flatten()  # Stack future predictions for meta-model  stacked\_future\_X = np.column\_stack((future\_ann\_predictions, future\_tcn\_predictions))  final\_future\_predictions = meta\_model.predict(stacked\_future\_X)  # Generate future time index based on the last datetime in train\_df  last\_date = train\_df.index[-1]  future\_dates = pd.date\_range(last\_date, periods=n\_future\_steps + 1, freq='10min')[1:]  # Check predictions range  print("Final Future Predictions Range:", final\_future\_predictions.min(), final\_future\_predictions.max())  # Plot past values (from training set) in blue and combined future predictions in green  plt.figure(figsize=(14, 7))  # Plot past values  plt.plot(train\_df.index[-n\_past:], train\_df[target][-n\_past:], color='blue', label='Past (Train)')  # Plot combined future predictions  plt.plot(future\_dates, final\_future\_predictions, label='Stacked Future Predictions', color='green', linestyle='--')  plt.title('Past and Future Soil Heat Flux Prediction using Stacked ANN + TCN Models')  plt.xlabel('Date')  plt.ylabel('Soil Heat Flux (Diff Volt 1)')  # Set legend location to the top right corner  plt.legend(loc='upper right')  # Format the x-axis to show the date and time  plt.gca().xaxis.set\_major\_formatter(mdates.DateFormatter('%Y-%m-%d'))  # Adjust x-axis ticks and limits  plt.gca().xaxis.set\_major\_locator(mdates.MinuteLocator(interval=60)) # Tick every hour  plt.gca().xaxis.set\_major\_locator(MaxNLocator(nbins=10, integer=True))  plt.xticks(rotation=45)  plt.ylim(-4, 4)  plt.tight\_layout()  plt.show( |
| --- |

**Plant Root Zone Temperature Optimization**

| import pandas as pd  import numpy as np  from scipy import stats  from skopt import gp\_minimize  import matplotlib.pyplot as plt  from tqdm import tqdm  # Load your dataset with error handling  def load\_data(file\_path):  try:  data = pd.read\_excel(file\_path)  data['DateTime'] = pd.to\_datetime(data['DateTime'])  data.set\_index('DateTime', inplace=True)  return data  except Exception as e:  print(f"Error loading data: {e}")  return None  # Load the dataset  file\_path = 'EnsemblePredictions.xlsx' # Update this path  data = load\_data(file\_path)  if data is not None:  # Define the depths and decay constant  depths = [10, 20, 40, 80]  alpha = 0.3 # Decay rate  # Calculate exponential decay weights and normalize them  raw\_weights = [np.exp(-alpha \* d) for d in depths]  normalized\_weights = [w / sum(raw\_weights) for w in raw\_weights]  # Map normalized weights to depth columns  weights = {  'SoilTemp\_10cm': normalized\_weights[0],  'SoilTemp\_20cm': normalized\_weights[1],  'SoilTemp\_40cm': normalized\_weights[2],  'SoilTemp\_80cm': normalized\_weights[3]  }  # Calculate RZT using the exponential decay weights  def calculate\_rzt(data, weights):  """Calculates the Root Zone Temperature (RZT) based on soil temperatures and their weights."""  data['RZT'] = (  data['SoilTemp\_10cm'] \* weights['SoilTemp\_10cm'] +  data['SoilTemp\_20cm'] \* weights['SoilTemp\_20cm'] +  data['SoilTemp\_40cm'] \* weights['SoilTemp\_40cm'] +  data['SoilTemp\_80cm'] \* weights['SoilTemp\_80cm']  )  return data  # Calculate RZT and update the dataset  data = calculate\_rzt(data, weights)  # Define time intervals based on the index  conditions = [  (data.index.hour < 6),  (data.index.hour >= 6) & (data.index.hour < 12),  (data.index.hour >= 12) & (data.index.hour < 17),  (data.index.hour >= 17)  ]  choices = ['Night', 'Morning', 'Afternoon', 'Evening']  # Create TimeOfDay column  data['TimeOfDay'] = np.select(conditions, choices)  # Calculate min and max RZT values before optimization  rzt\_before\_optimization = data.groupby('TimeOfDay')['RZT'].agg(['min', 'max'])  print("Before Optimization:")  for time\_of\_day, row in rzt\_before\_optimization.iterrows():  print(f"{time\_of\_day} - Min RZT: {row['min']:.2f}°C, Max RZT: {row['max']:.2f}°C")  # Define dynamic bounds for optimization based on the data  lower\_bound = max(10, data['RZT'].min()) # Lower bound can be reduced  upper\_bound = min(35, data['RZT'].max() + 5) # Upper bound can be increased  bounds = [(lower\_bound, upper\_bound)] # Bounds for RZT  # Define an objective function that evaluates each entry's RZT value  def comprehensive\_objective(rzt):  """Objective function to minimize based on individual entries."""  penalties = []  rzt\_value = rzt[0] # Extract single value from list  for index, row in data.iterrows():  ideal\_temp\_low = 18  ideal\_temp\_high = 25  if rzt\_value < lower\_bound:  penalties.append((lower\_bound - rzt\_value) \*\* 2 \* 2) # Strong penalty for being too low  elif rzt\_value > upper\_bound:  penalties.append((rzt\_value - upper\_bound) \*\* 2 \* 2) # Strong penalty for being too high  elif rzt\_value < ideal\_temp\_low:  penalties.append((ideal\_temp\_low - rzt\_value) \*\* 2 \* 1.5) # Penalty for being too low  elif rzt\_value > ideal\_temp\_high:  penalties.append((rzt\_value - ideal\_temp\_high) \*\* 2 \* 1.5) # Penalty for being too high  else:  penalties.append(0) # No penalty if within the ideal range  # Handle potential NaN values  penalties = [penalty if not np.isnan(penalty) else 0 for penalty in penalties]  return np.mean(penalties)  # Function to run optimization for a single row  def optimize\_row(row):  res = gp\_minimize(  comprehensive\_objective,  bounds,  n\_calls=20, # Reduced number of calls  acq\_func='EI',  random\_state=42,  x0=[row['RZT']] # Start with current RZT value as initial point  )  return res.x[0] # Return optimized value  # Run Bayesian optimization without parallel processing with tqdm progress bar  optimized\_rzts = []  for \_, row in tqdm(data.iterrows(), total=data.shape[0], desc="Optimizing RZT"):  optimized\_rzt = optimize\_row(row)  optimized\_rzts.append(optimized\_rzt)  # Add optimized values back to DataFrame for comparison  data['Optimized\_RZT'] = optimized\_rzts  print("\nAfter Optimization:")  # Display optimized RZT values along with min and max values after optimization.  for timeframe in data['TimeOfDay'].unique():  current\_min = data[data['TimeOfDay'] == timeframe]['RZT'].min()  current\_max = data[data['TimeOfDay'] == timeframe]['RZT'].max()  optimized\_min\_after\_opt = min(current\_min, data[data['TimeOfDay'] == timeframe]['Optimized\_RZT'].min())  optimized\_max\_after\_opt = max(current\_max, data[data['TimeOfDay'] == timeframe]['Optimized\_RZT'].max())  print(f"{timeframe} - Optimized Min RZT: {optimized\_min\_after\_opt:.2f}°C, "  f"Optimized Max RZT: {optimized\_max\_after\_opt:.2f}°C")  if len(optimized\_rzts) < len(data):  optimized\_rzts += [None] \* (len(data) - len(optimized\_rzts))  # Add optimized values back to the original DataFrame for comparison  data['Optimized\_RZT'] = optimized\_rzts  print("\nAfter Optimization:")  # Display optimized RZT values along with min and max values after optimization.  for timeframe in data['TimeOfDay'].unique():  optimized\_min\_after\_opt = data[data['TimeOfDay'] == timeframe]['Optimized\_RZT'].min()  optimized\_max\_after\_opt = data[data['TimeOfDay'] == timeframe]['Optimized\_RZT'].max()  print(f"{timeframe} - Optimized Min RZT: {optimized\_min\_after\_opt:.2f}°C, "  f"Optimized Max RZT: {optimized\_max\_after\_opt:.2f}°C")  # Save the updated DataFrame to a CSV file (optional)  data.to\_csv('optimized\_results\_1910.csv', index=False) |
| --- |