# Deep Learning Approach For Water Management In Ethiopia Farmlands

A Thesis Presented
by
Abel Demelash

to

The Faculty of Informatics
of
St. Mary's University

In Partial Fulfillment of the Requirements
For the Degree of Master of Science

in

Computer Science

July 24, 2024

# ACCEPTANCE

**Deep Learning Approach For Water Management In Ethiopia Farmlands**

**By**

**Abel Demelash**

**Accepted by the Faculty of Informatics, St. Mary's University, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science**

**Thesis Examination Committee:**

______________________________

**Internal Examiner**
**{Full Name, Signature and Date}**

Dr. MESFIN ABEBE [illegible] 22/07/24

**External Examiner**
**{Full Name, Signature and Date}**

______________________________

**Dean, Faculty of Informatics**
**{Full Name, Signature and Date}**

**July 24, 2024**

## DECLARATION

I, the undersigned, declare that this thesis work  is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been duly acknowledged.

**Abel Demelash**
Full Name of Student

______________________________
Signature

Addis Ababa

Ethiopia

This thesis has been submitted for examination with my approval as advisor.

**D.r Alembante Mulu**
Full Name of Advisor

_________________________
Signature

Addis Ababa

Ethiopia

July 24, 2024

## Acknowledgment

First of all, I thank GOD first and foremost, without whom everything will not be possible.

I would also like to thank my adviser Dr. Alembante mulu for assisting me in this thesis ,guiding me in the right direction and giving me valuable advice and correcting me and showing me my shortcomings in the making me this thesis.

I also  want to thank the Ethiopian Agricultural Transformation Agency for providing me with the available data that was critical to our thesis paper.

I also want to thank my family for supporting me in every possible way they could throughout my masters education and thesis journey.

## Table of Contents

## List of Acronyms

| | |
|---|---|
| A.I | Artificial Intelligence |
| Al | Aluminum |
| B | Boron |
| BCM | Billion Cubic Meters |
| Ca | Calcium |
| C | Carbon |
| CEC | Cation Exchange Capacity |
| Co | Cobalt |
| Cu | Copper |
| EAIA | Ethiopian agricultural transformation agency |
| EC | Electrical Conductivity |
| ELU | Exponential Linear Unit |
| ET | Evapotranspiration |
| Fe | Iron |
| GRU | Gated Recurrent Unit |
| HP | Exchangeable Acidity |
| IOT | Internet of Things |
| Lat | Latitude |
| Lon | Longitude |
| LSTM | Long Short-Term Memory |
| Mg | Magnesium |
| Mn | Manganese |
| MLP | Multi-Layer Perceptron |
| Na | Sodium |
| N | Nitrogen |
| NGO | Non-Governmental Organization |
| NSIS | National Soil Information System |
| P | Phosphorus |
| PH | Power Of Hydrogen |
| ReLU | Rectified Linear Unit |
| RNN | Recurrent Neural Network |
| S | Sulfur |

| | |
|---|---|
| SGDM | Stochastic Gradient Descent with Momentum |
| SGD | Stochastic Gradient Descent |
| Si | Silicon |
| Zn | Zinc |

## List of Figures

## List of Tables

## Abstract

Irrigation is a critical method for managing farmland resources such as water and fertilizers. In Ethiopia, irrigation has been extensively used, and to modernize the current irrigation system in terms of water management, I  have designed a machine learning-based system that automates water management to enhance irrigation efficiency. This study utilized soil chemical data collected from farmlands in the Oromia Region, East Showa Zone, Adama Woreda, provided by the Ethiopian Institute of Agricultural Transformation (EIAT). We collected a total of 90 soil features using various preprocessing techniques to address issues that could render the data unusable by machine learning algorithms. Additionally, thresholding and weighted sum analysis were applied to prepare the data for water management purposes and to generalized decision-making.

To develop our classification model for water management, we implemented three machine learning algorithms: Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). These algorithms are capable of handling non-linear issues present in the data. We employed hyperparameters such as Adam optimizers and activation functions (sigmoid, tanh, and ReLU), along with evaluation metrics including accuracy, precision, recall, and F1 score.

By applying these parameters in combination with the three algorithms, developed machine learning models with accuracy rates of 95.4%, 95.8%, and 94.3% for MLP, LSTM, and GRU, respectively, after multiple training sessions using various parameter combinations. This study demonstrates the potential of machine learning models to significantly improve water management in irrigated farmlands, contributing to the sustainable use of water resources in agriculture.

**Keywords:** Irrigation, machine learning, MLP, LSTM, GRU, optimizers, Activation Function

# Chapter 1

# 1. Introduction

## 1.1. Background

Irrigation is a system by which agricultural resources like water and pesticides as well as fertilizers are applied in a systematic way , [1] it's designed to allow the needed crop production and Sustainability in water demanding regions, and reduce the impact of drought in Semi-desert regions or semi-wet regions. even in areas where enough rain or naturally flowing waters are in comparison to the semi-desert is more in abundance , issues like unevenly distribution of rain, soils inability to hold required amount of water arise therefore making traditional rain dependent agricultural practices highly unreliable , and currently according to [1] in Ethiopia it is being used to grow crops like Teff , wheat and other vegetation's and fruits like coffee and sugarcane , irrigation is the main strategy being Used by Ethiopia to transform Ethiopia's rain based agriculture into a more sustainable and productive agricultural development system [1] which will increase the amount of food being produced ,according to [12] the economic development of the farmers as well as the country economy in overall and decrease water loss of farmlands by up to 28% , and currently less than 3% of farm lands in Ethiopia use irrigation system most being government owned with foreign NGO partners irrigation's, and the annual water usage of irrigated farmlands in Ethiopia is estimated at 1.5 BCM , There are three types of irrigation methods used in Ethiopia those are gravity method ,drip and sprinkle system ,With Sprinkle System assisted irrigation human made rain that is limited to the irrigation region is created. the water is passed through Pipe System in which it is put under high pressure, the spraying is applied using Several Sprinkles heads that are automatically rotating, the system also has major components like pumping stations conveyance system , distribution system and drainage system, and control system including soil moisture sensors, solar irradiance detectors, temperature and atmospheric humidity measurements and a manual or computer to operate a valve and pump system.

In the near future [12] the Increased demand in food production and the climate change will lead to modernized irrigation and water management systems to be developed , and one currently being used and studied in the developing world like India [28] is Artificial intelligence or machine learning based irrigation management system ,which is being used to manage the water flow into the farmlands crops , the health of crops and fertilizer usage, even though this system is being used in other developing countries like India to feed it’s high populations , and currently in

Ethiopia which has an estimated population of 120 million people it’s not being utilized to advance the current irrigation system which we believe is an opportunity being wasted.

## 1.2. Motivations

In Ethiopia where farming is the main source of food as well as economic income for over 80% of the population and also Ethiopia being located in east Africa an area which is currently being affected by extreme drought due to climate change which makes it impossible to farm due to shortage of  ground water , shortage of rain fall and lack of fertilizers as well as lack of highly skilled and motivated labor force , and with a population of 120 million which is a lot of mouth to feed let alone export crop products to the foreign market. Water management in agriculture is a critical concern, particularly in regions with limited water resources. Efficient irrigation practices are essential to ensure sustainable crop production and conserve water. The motivation behind this research stems from the pressing need to enhance water management strategies in irrigated farmlands through the use of advanced technologies. Traditional methods of water management, which often rely on manual monitoring and expert judgment, are not only labor-intensive but also prone to inaccuracies. With the advent of machine learning, there is an unprecedented opportunity to revolutionize agricultural practices by developing predictive models that can optimize irrigation schedules based on real-time soil data. This research aims to harness the power of machine learning to provide farmers with precise, data-driven insights, ultimately leading to improved water use efficiency, higher crop yields, and sustainable agricultural practices.

## 1.3. Statement of the Problem

Efficient water management in irrigated farmlands is a complex challenge, particularly in developing regions where resources are scarce, and agricultural practices are crucial for livelihoods. The traditional methods of determining irrigation needs are often inadequate, leading to either over-irrigation, which wastes precious water resources, or under-irrigation, which hampers crop productivity. The lack of precise, real-time data further exacerbates this issue, making it difficult for farmers to make informed decisions. This research addresses the critical problem of how to accurately predict irrigation needs using machine learning algorithms, based on the analysis of soil chemical properties. By developing a predictive model with high accuracy, this study aims to provide a viable solution to optimize water usage, reduce wastage, and enhance agricultural productivity. Although according to [12] irrigation system is very effective In Ethiopia compared to conventional way of farming it still has its drawbacks compared to the goal and the resource investment the country is putting in and the return investment its getting, some of this include high level power usage for its pump stations , less productive crops both In terms of quality and quantity , excessive water usage of about 30 – 35% which also Adds to environmental pollution by transferring dissolved salt and pesticides remains into the nearby supply of water which affects the irrigated farmlands as well as its surrounding area, the environment and communities that live in the area. And overall according to the current system we believe these issues are as a result of absence of Artificial Intelligence based automation in the Ethiopia's irrigation system by comparing it to other fully or partially automated irrigation systems in counties like India in relation to ours , as India was in the same situation with their irrigation system and they were able to overcome them by integrating Their irrigation system with Artificial Intelligence based solutions to increase their productivity and decrease their wastage of resources like water and fertilizers. Which i believe we can implement in our countries irrigation fields as well with similar effect.

## 1.4. Research Question

I believe I will be able to answer these questions with our research:.

- What is the relationship between various soil chemical properties and water content in irrigated farmlands?
- How effective are AI-based predictive models in managing water resources compared to human expert-based methods?
- How accurately can machine learning models (MLP, LSTM, and GRU) predict the irrigation water needs of farmlands based on soil chemical data?
- How do the accuracies of MLP, LSTM, and GRU models compare to traditional models like kNN and SVM?
- How can hyperparameter tuning improve the performance of MLP, LSTM, and GRU models in predicting irrigation needs?
- How does the size and complexity of the dataset affect the accuracy and reliability of the predictive models?

## 1.5. Objectives

### 1.5.1. General Objective

The general objective of this research is to develop a robust and accurate predictive model for water management in irrigated farmlands using advanced machine learning algorithms. By analyzing a set of soil chemical data obtained from governmental institutes, the research aims to uncover the relationships between various soil properties and water content. This understanding will facilitate the creation of models that utilize algorithms such as Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) to determine the irrigation needs of soil with high precision. The research also seeks to demonstrate the superiority of AI-based models over traditional methods in conserving water resources while maintaining crop productivity. Paving the way for future improvements in predictive accuracy and model scalability. Ultimately, the goal is to provide a sustainable and efficient solution for water resource management in agriculture, leveraging the power of AI to address the challenges posed by water scarcity and the need for optimized irrigation practices.

### 1.5.2. Specific Objective

- To review state of the art literature on Artificial Intelligence Based Irrigation agricultural technologies.
- To prepare required Data Set for Agricultural Irrigation Water Management
- To better understand the science and understanding of the features (Chemical properties of soil in relation to water management in a farming environment) that affect the agricultural industry.
- To Design a Deep Learning Model for our Agricultural environment that is enhanced using irrigation technology.
- To measure the performance of my model.

## 1.6. Scope/Limitations

### Scope

The scope of this research encompasses the development and evaluation of a machine learning-based model for water management in irrigated farmlands. The study involves collecting and analyzing soil chemical data from governmental institutes to understand the relationship between soil properties and water content. It explores the application of several machine learning algorithms, including Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU), to predict the irrigation needs of soil. The research includes the preprocessing of soil data, hyperparameter tuning, and the assessment of model performance in terms of accuracy. Additionally, the study compares the proposed models with existing traditional methods and other machine learning algorithms to highlight their effectiveness.

## Limitations

In this research, several limitations are found. Firstly, the accuracy of the predictive model is highly dependent on the quality and quantity of the soil chemical data available. Limited or incomplete datasets can affect the model's performance. Secondly, while the research demonstrates the effectiveness of deep learning-based models, their implementation in real-world scenarios requires significant technological infrastructure, including IoT devices and reliable internet connectivity, which may not be readily available in all regions. Thirdly, the study focuses on specific machine learning algorithms (MLP, LSTM, and GRU), and while these have shown high accuracy, there may be other algorithms or combinations thereof that could yield even better results. Finally, the model's predictions need to be validated through extensive field testing, which is beyond the current scope and resources of my research scope. Future work should address these limitations by scaling up data collection efforts, exploring additional algorithms, and conducting real-time field testing to prove the practical applicability and reliability of our model.

## 1.7. Significance of the Study

We hope the AI based model being developing for the irrigation system based farmlands will be implemented in future irrigation projects, and we believe the effects of this system will help:

- Farmers have more crop yield and more profitability.
- Farmers will have to spend less on fertilizers and other resources due to less water pollution.
- Customers get high in quantity and quality crop products and with fewer prices.
- Irrigation dams have better full to semi-automated system to manage water which will increase their overall performance and decrease water loss.
- Government will have advanced Agricultural industry that helps the country to better advance the economy

- As researchers I will gain knowledge about research on deep learning concepts and how to implement this research on practical issues with real data.

## 1.8. Organization of the Thesis

Our thesis is organized by chapters which include, first chapter titled introduction ,in this part we discuss about the general understanding  of the problem we-are trying to solve, the objective we are trying to achieve, scope of our research , significance and limitations of the study , and in the second chapter we perform literature review of previous works to understand the processes , approaches and related works in the third chapter methodology we discuss about data collecting and data analysis / preprocessing methods , and about the machine learning algorithm we will implement , and how we implement them and the software and hardware tools we use during our experiment , and in the fourth chapter experiment we discuss about our experiment including how we set up our experiments and the results of our experiments and in the fifth chapter conclusion we conclude our research with some recommendations.

# Chapter 2

# 2. Literature Review

## 2.1 Introduction

In this chapter we discuss the basic theories and Current Understand to The nature of irrigated water management to make an analysis and future predictions using machine Learning Algorithms, we perform review of early literature's on the Subject matter of the nature of Irrigation, techniques and methods used to manage the natural resources and the science behind the natural relations between water as a resource in Soil, and the machine learning based techniques which can help us achieve advance the resource management on irrigation environment to help us achieve an advanced Way of managing water consumption in Irrigation and maximize productivity as a result.

## 2.2 Water - Soil Dynamic In Farmlands

water is a very scarce resource in a farming environment both in an irrigated and non-irrigated ,which is why it needs to be managed in irrigation which is why it needs to be managed properly and [12] thanks to the modern technology we have we can manage water intake of crop fields using different technologies , methods and devices like soils sensors ,weather satellites and many more, and irrigation systems need highly advanced supportive systems to help management of resources like water and other resources like fertilizers and labor to save cost.

Agriculture Is the most water demanding sector, [5] taking 2/3 of the water from area found, as the world proportion is expected to increase by at least 30%. this In Conjunction with the current climate crisis spreading in Sub-Sahara areas that is causing droughts leading to crop failure and lives stock deaths in greater numbers causing farmers to Suffer, which impacts people starting from farmers immediate families to the local community that is dependent on the farming ecosystem. and the lack of innovating solutions exacerbates the situation even more. according to [5]Ethiopia has currently over 2.7 million hector Irrigated field, which Is a total of 20% of what the countries potential land that can be irrigated for farmland use; where over 97% of the irrigation method is surface irrigation. Maze as being the primary crop being grown in the region, maze is sensitive to water volume it receives, current irrigation methods include deficit irrigation and different furrow irrigation methods.

## 2.2.1 Soil Components

Soil is a composite of minerals, organic substances, gases, liquids, and numerous organisms, all of which collectively sustain plant life. It forms a natural entity within the pedosphere and fulfills four crucial roles: acting as a medium for plant growth, serving as a system for water storage, supply, and purification, modifying the atmosphere, and providing a habitat for organisms involved in the decomposition of organic matter and the formation of new habitats.

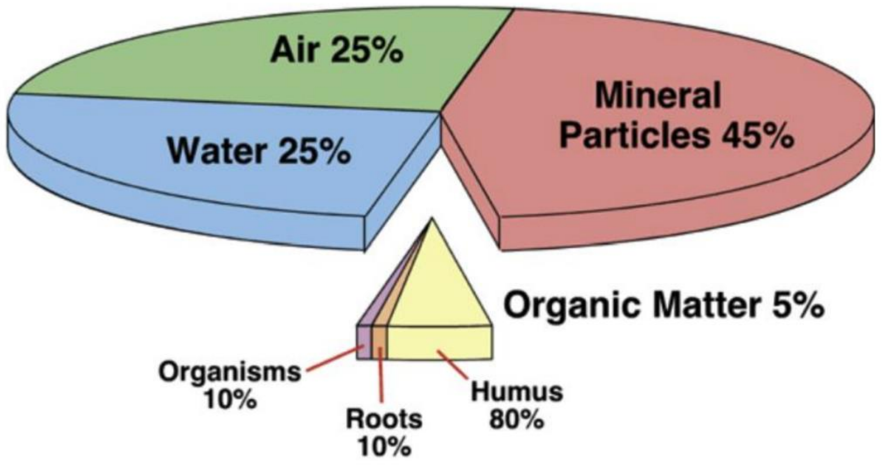


Figure 2.1 : soil components

soil is composed of [2] different minerals , and also [3] organics matters which are necessary in managing the soil prolificacy and also minimizing the loss of nutrients, water and air ,the composition of this components affect the physical properties of the soil we analyze. in order to analyze these properties in the soil [12][13] we use soil sensors which are devices we embed inside soil in an irrigated farmland to detect and report properties of soil(temperature ,PH, moisture and humidity and more) in an analog signal which then will be converted to digital signal with a specific value and be stored in a remote database like cloud services or local databases on local machines.

The other element found inside soil is water, it encompasses between 2 - 50% in volume, it helps by moving nutrients to the plants helping , their overall growth, and the soil organisms included, and helping in the process of , both biological as well as chemical decomposition. the idea of soil water availability in the extent to which the soil can hold a Certain amount of water which later will be available to the plants to Use, this Character of the soil is dictated mainly by the texture of the soil, where the condition where there is more small particles are found in The soil, the ability for the soil to hold make water increases the type of Soil with this condition called clay soil, where the opposite characteristics is displayed in soil called sand soils. Other influences that affect the soil’s capacity of holding water is called organic matter, where the more excess presence of organic matter  in the soil is the indication of the higher chance of the soils ability to hold more water.

Air is another element found in soil, occupying same Volume as water due to air ability to occupy same volume of water (2-50%), its presence in soil is important for both root and microbe respiration, supporting growth of the plant, other gases like carbon dioxide and nitrogen are vital for below ground plant functions like in nitrogen case , nitrogen-fixing bacteria and where for the case of carbon dioxide, as the level of carbon dioxide increase is determined by the depth of the Soil, because of decomposition of the organic matter it accumulated and the available amount of plant roots.

### 2.2.2 Soil Chemical Properties

soil temperature [13] is a property in a soil that is essential for many soil processes and reaction that include water and nutrient uptakes, soil temperature properties change by the radiant, thermal and latent energy exchange processes that take place in the soil, components of soil thermal properties such as specific heat capacity are affected by the water content of the soil, the flow of water and heat is an interactive process. where temperature gradients affect the moisture level of the soil , soil temperature vary due to the constant change in climate and meteorological change and the interactions of soil and atmosphere ,some of the factors that affect temperature in a soil are changes in weather, landscape , regional differences ,crop type and soil management processes.

PH  according to [3] value is measurement of the hydrogen ion concentration where it is the range of between (1-14), where it is a reverse scale in that a very acidic soil has a low PH and high hydration ion concentration , therefore at high (alkaline) PH values , the hydration ion concentration is low , most soils have a PH value between (3.5 - 10) , in higher rain fall areas the natural PH of soil typically range from (5-7) , while in dry land areas the range is (6.5 - 9).

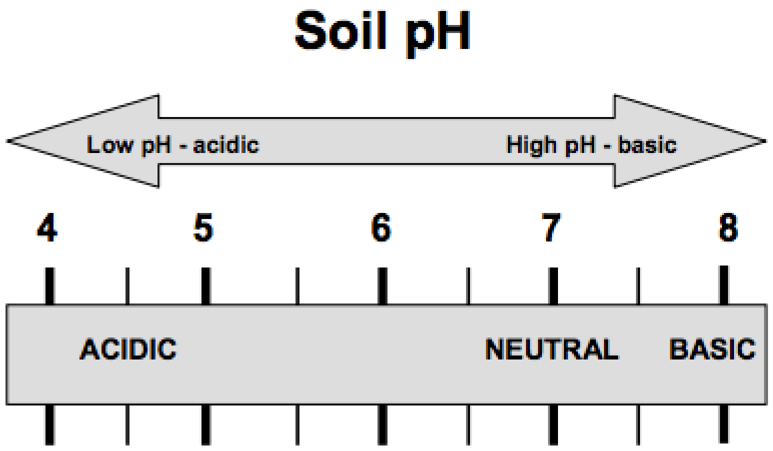
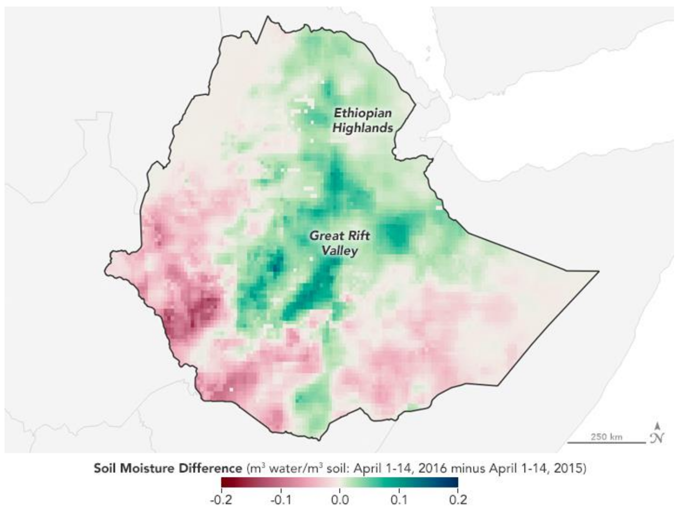


Figure 2.2 : soil PH range

soil moisture [2] is the amount of water found inside an active layer of the soil , it's the source of water that evaporates from the soil and crops into the atmosphere ,the water content of the soil is a vital element in the hydro-logical cycle and crop growth , quantification of soil moisture , valuable for understanding water resource management. In soil moisture analysis soil water content is expressed as percentage of the water by weight , volume or in other measurements as inches of water per inches of soil, and the values we get from this units of measurement will be categorized into categories like start irrigation where irrigation should started to complete most irrigation cycle before unwanted stress occurs in the irrigation , where we believe we will get to an optimum range which is a state in the soils moisture range in which crop high stress is decreased and it is required to manage soil moisture in the range of these readings.


Ethiopian Highlands
Great Rift Valley
250 km
N
Soil Moisture Difference (m³ water/m³ soil: April 1-14, 2016 minus April 1-14, 2015)
-0.2
-0.1
0.0
0.1
0.2
Figure 2.3 : soil moisture in Ethiopia

Humidity in a soil is [2] the most important factor for plant growth , which refers to the amount of water in the soil , by measuring how moisture wet the soil is.

The relationship between soil chemical properties and water availability is complex and multifaceted, [3] as each element plays a unique role in soil chemistry and plant health. Key nutrients like Potassium (K), Calcium (Ca), Magnesium (Mg), and Phosphorus (P) are crucial for water regulation and uptake efficiency in plants. Potassium enhances water use efficiency and drought resistance, while Calcium improves soil structure, facilitating better water infiltration and retention. Magnesium, a central element of chlorophyll, influences soil structure and water holding capacity, and Phosphorus is essential for root development and energy transfer, impacting water uptake. Other elements, such as Iron (Fe), Manganese (Mn), Zinc (Zn), Boron (B), and Copper (Cu), though required in smaller amounts, also significantly affect water uptake by participating in enzyme functions, photosynthesis, and nutrient transport. The soil pH (Hydrogen Potential) is another critical factor, as it affects nutrient availability and microbial activity, thereby influencing water retention and availability.

Soil structure and organic matter content, represented by [29][30] Carbon (C) and Cation Exchange Capacity (CEC), are fundamental for determining water holding capacity of soil , where the optimal range of the element Carbon are in the range of (1 - 3%) as an implicit way of telling soils healthy water level . High organic matter improves the soils structure, enhancing water retention. The CEC indicates the soil’s ability to hold and exchange nutrients with the CEC being in an optimal range of (10 - 40 Cmolc/kg) showing a healthy amount of presence of hydration, directly impacting water availability to plants. Electrical Conductivity (EC) (optimal range of less than 2dS/m ) and Sodium (Na) (optimal range of having larger than 50 ppm to avoid salinity issues )levels are [31][32]indicators of soil salinity, with high values often leading to poor soil structure and reduced water infiltration. Elements like Aluminum (Al) and Cobalt (Co), while essential in trace amounts, can be detrimental in higher concentrations, affecting root growth and water uptake where [33][34] their optimal values for both elements ranges larger than 0.2 ppm and a range of between 0.1ppm and 0.02 ppm respectively. The intricate balance of these chemical properties defines the soil's ability to retain and provide water to plants, crucial for maintaining soil health and agricultural productivity.

The weighted influence of soil chemical properties on water availability involves quantifying the impact of each element on the soil's capacity to retain and supply water to plants. Potassium (K) and Calcium (Ca) also have significant weights [35][36] due to their crucial roles in enhancing water use efficiency and improving soil structure,100 - 300 ppm and 400 - 2000 ppm in optimal ranges respectively. Magnesium (Mg), [37] being central to chlorophyll and indirectly affecting soil structure and water retention with an optimal range of 50 -100 ppm, also holds substantial weight. Phosphorus (P), essential for root development and energy transfer, is [38] weighted for its impact on water uptake where its ranged for its optimal condition of water availability of 10 - 100 ppm. Trace elements like Iron (Fe), Manganese (Mn), Zinc (Zn), Boron (B), and Copper (Cu), though required in smaller quantities, are weighted for their roles in enzyme functions, photosynthesis, and nutrient transport, all of which influence water uptake efficiency where each are measured also to have[39][36][40][41][42] a range of indicators for showing water availability with optimal ranges of (4.5 -10ppm) , (20-100ppm) ,(1.0-5.0ppm) ,(0.5-2.0ppm) , and (0.2-1.5ppm) respectively each.

| **Chemical Elements** | **Optimal Range** |
|---|---|
| PH | 6.0 - 7.5 |
| P | 10 - 60 ppm |
| K | 100 - 300 ppm |
| Ca | 400 - 2000 ppm |
| Mg | 50 - 200 ppm |
| Mn | 20 - 100 ppm |
| S | 10 -30 ppm |
| CU | 0.2 - 1.5 ppm |
| B | 0.5 - 2.0 ppm |
| Zn | 1.0 - 5.0 ppm |
| Na | > 50 ppm |
| C | 1.0 - 5.0 ppm |
| N | 1.0 - 4.0 ppm |
| Fe | 4.5 - 10 ppm |
| Al | > 0.2 ppm |
| Si | 50 - 150 mg/kg |
| Co | 0.02 - 0.1 ppm |
| Mo | 0.02 - 0.2 pm |
| ec | > 2dS/m |
| cec | 10 - 40 Cmolc/kg |

Table 2.1 : soil chemical properties optimal range

Soil pH (Hydrogen Potential) is [43] heavily weighted as it affects nutrient availability and microbial activity, crucial for water retention. The soil's organic matter content, represented by Carbon (C), is [29]weighted for its significant impact on soil structure and water holding capacity. Cation Exchange Capacity (CEC) is [30] another heavily weighted factor, indicating the soil’s ability to hold and exchange nutrients, directly affecting water availability. Electrical Conductivity (EC) and Sodium (Na) levels are weighted[31][32] for their roles in indicating soil salinity, with high values reducing water infiltration. Elements like Aluminum (Al) and Cobalt (Co), although necessary in trace amounts, are [33][34]weighted for their potential negative impact on root growth and water uptake at higher concentrations. The weighted values of these chemical properties are critical for understanding their collective influence on soil water dynamics and ensuring optimal soil health and agricultural productivity.

| **Chemical Properties** | **Weight** |
|---|---|
| PH | 1.5 |
| P | 1.2 |
| K | 1.3 |
| Ca | 1.1 |
| Mg | 1.1 |
| Mn | 1.0 |
| S | 1.0 |
| CU | 0.9 |
| B | 0.9 |
| Zn | 1.0 |
| Na | 1.3 |
| C | 1.4 |
| N | 1.2 |
| Fe | 1.0 |
| Al | 1.2 |
| Si | 0.8 |
| Co | 0.8 |
| Mo | 0.9 |

|  |  |
|---|---|
| ec | 1.3 |
| cec | 1.4 |

Table 2.2 : soil chemical properties weight [36][37][38][39][40][41][42][44][45][46][47][48]

## 2.3 Approaches to Water Management in Irrigated Farmlands

Water management in irrigated farmlands involves [4][5] a variety of approaches and techniques aimed at optimizing water use efficiency, enhancing crop yields, and ensuring sustainable agricultural practices. Here are some of the key approaches:

### 2.3.1 Scheduling Based Water Management in Irrigation

Decision made in irrigation like when to irritate in  terms of season and Where to irritate in terms of water deprived locations when the water in land is production critical. The decision is based on Considering the Seasons. Farmers also need to decide when the water at hand will be used and the type of crop to plant also . [11] unreliable rainfall pattern further makes making decisions hard to make. irrigation Scheduling techniques be applied to resolve This issues to a certain level of success.

Demand for conservational management of water resources on irrigation farmland with irrigation efficiency & Crop productivity & Soil health led to the development of Irrigation Scheduling mechanism, as described by [4] irrigation Scheduling in way of planing and decision making that irrigation operators apply to manage flow of water & other resources from dams to the farmlands. The decision making of scheduling is based on set of model like soil -water balance model ,  soil water tension, leaf water Content and entropy temperature ,Irrigation Scheduling requires requires lots of complimentary data to make the decision to determine when and how much water to apply to meet the objective

### 2.3.2 Smart Irrigation System Water Management

Smart irrigation is currently in use technology based Irrigation management system where technologies [4][12] Such as sensors and micro controllers are used, Sensors are devises used to collect analog stream of raw data's where they could collect moisture levels , temperature and PH level , as well as other chemical properties of the Soil the sensors are placed at, and this streams of data are collected [14] using micro controllers like Arduino and raspberry pi, these

micro controllers are connected to a storage source that are either found locally or remotely, where the irrigation controllers can access them and make experts based analysis & decision making system In order to manage resources like water , fertilizers and other resources.
Smart Irrigation Systems Can help [6] [12] countries that are less developed with less resource and poor resource management and high population resulting in high demand of crop production smart Irrigation's help by efficiently Utilizing their usage of their water resource in agriculture, by auto managing their water reserve, distribution and consumption of various levels, avoid over-irrigation and under irrigation problems

### 2.3.3 Deficit Irrigation System Water Management

Deficit Irrigation is according to [6] is a technique used in irrigation to manage water usage in farmlands . where it tries to minimize water consumption, by exposing the crop to a certain Level of water stress, either in an interval period or through out the growth process, with an expectation of any yield reduction will be compensated by increased production from the additional irrigated area with the water Save by difficult irrigation. Deficit irrigation is the application of less water than is required for potential ET and maximum yield.

### 2.3.4 Machine Learning Based Approaches

#### 2.3.4.1 Neural Networks

A neural network is a methodology in artificial intelligence field derived by connecting neurons into a layered structure. In computer science, a neural network is a mathematical model with multiple set of parameters, this neural network technique is based on [19] the human brains neural system and the neural networks try to imitate this natural process of our human brain. The model consists of multiple functions, neural network algorithms which are supported by mathematical equations.
A Neural network is a technique used to process set of data’s given to us to get a needed result based on our models purpose ,neurons are the fundamental building block of the artificial neural networks architecture ,each neuron in neural network is layered in a column , and three distinct parts namely ,the input layer ,the output layer ,and the hidden layers , where each neuron is interconnected between layers , where [21] each neuron takes input value and from previous neuron (except the input layer) and passes output to the next neuron , in the input and output

layers we only have single columns each , but for the hidden layer we can have between zero (adaline network) , one (madaline network) ,or multiple columns of neurons.

**Input layer**

The primary layer we find in our neural architecture is called the input layer , the input layer is where the data is first fed to the network it serves as an entrance for our data features , it's main objective is to accept and analyze the input data's before passing them to the next layers (hidden layer) of neural networks for further analysis.

The input layer serves as the link between the input data and our neural network (the hidden and output layers) , taking multiple forms of input datas like textual , numerical (integer or floating or binary for categorical data's)like in our case of data we collected , and images and passing them to the hidden layer , the input layer is composed of standalone node also called neuron , the number of these neurons is set equal to the number of input features in the given data set , each node (neuron) in the input layer is a representation of a single input from our data set , and are fully connected to the hidden layers nodes with each being assigned a weighted values ,this layer also formats the input data into a format which makes the data more suitable to be processed by the neural network we are using by scaling the input data into a range between [0 and 1] (normalization) or by scaling them zero mean and standard variance (standardization) , usually activation function functions are not applied on this layer , as there being no preprocessing being done on the neurons of this layer.

**Hidden layers**

neural networks main part is the hidden layer which is located between the input layer and the output layers ,this layer is built using layered sets of nodes(neurons) which are connected with both the input layers nodes as well as the output layer nodes fully in a feed forward fashion, this layer receives set of data's from the input layer and transforms them into a meaningful data the next layer can use ,it solves non-linear problems between the given input and the expected output by calculating the activation function where the weights of the neurons are multiplied by the given input of the previous layer plus the bias . and this results are passed to the next layer as inputs.

The activation function in the hidden layer allow [21] for it to solve non-linear problems and learn more complex patterns, while training this layers to have maximum success we adjust the weights as well as biases of this neurons using the back propagation algorithm which implements an optimization of gradient decent with a goal of minimizing loss function.

**Output layer**

The final layer in neural network is named output layer where the nodes receive out puts of the hidden layer and give an out put to a varying degree of either predictive or classification , this layers architecture ,activation function as well as back propagation, and the way we interprate the out put this layer gives depends on the nature of the task given. for classification task in this layer we use the sigmoid logistic activation function , and assigned only a single neuron pointing the probability of belonging to a positive class , and a loss function called the binary cross-entropy loss is used .for multi-class classification task the output layer uses the softmax activation function , and architecturally it’s assigned as same number as the number of classes ,and a  loss function called categorical cross-entropy loss is applied.

## 2.3.4.2 Activation Functions

In artificial neural networks activation functions are very important as they help in [21] learning, non linear and complicated mappings between inputs and output. Neural networks accuracy is mainly dependent on set of situations like number of layers as well as the activation functions used ,for the accuracy to be as high as possible the neural networks need to be at made up of at least 2 layers (the hidden layer and output layer ) with a combination of any activation function. if activation function is not applied in our neural network the output we get will be a linear function , limiting us from doing complex and detailed mapping from our data sets ,when our data is very complex and require a non-linear function because of its non-linear property , hence we use activation function that are non-linear where we are abeld to use in our hidden layer a complex neural architecture in conjunction to make an analysis and understand and make predictions based on our data sets that has a non-linear in property.

Neural networks prediction accuracy is defined by the type of activation we select, which are [21] mostly non-linear activation functions ,in real world situation errors have a non-linear properties ,hence we are required to use non-linear activation functions to address the non-linearity of the data ,and to achieve that  we assign each nodes that are interconnected nodes an activation function. Neural networks therefore need activation functions to be abeld to analyze complex information and represent non-linear convoluted random functional mapping between input and output layers, hence applying non-linearity to the network using non-linear activation function, we can map non-linearity from the input layer and passing through the hidden layer to the output layer using this functions.

**Sigmoid Activation Function**

Sigmoid is an activation function mostly used by neural networks that are trying to solve non-linear problems ,where it takes real values as inputs and transforms them into values of 0 and 1 as outputs , Sigmoid is used in models where the model has to predict probabilities as an output.

**Tanh Activation Function**

The Tanh activation function, or hyperbolic tangent, is activation function that maps input values into a range between [-1 and 1] , it's a function that is zero-centered, making it advantageous for certain neural network architectures, as it can lead to faster convergence compared to the Sigmoid function. Its output is more balanced between positive and negative values.

**ReLU (rectified linear unit) Activation Function**

ReLU activation function [7] is widely used popular function , where it's used in complex neural networks, with a range of from 0 to infinity, its major advantage over others is does not need for every neurons to be activated all by the same time , and activation only happening when the output of the linear transformation is zero, ReLU is computationally efficient and helps to mitigate the vanishing gradient problem. However, it can encounter the "dying ReLU" issue where neurons become inactive if they constantly output zero.

**maxout Activation Function**

The Maxout function is a type of function where we generalize the ReLU and leaky ReLU functions. It selects the maximum value from a set of inputs, . This technique allows the network to learn both unimodal and multimodal functions, providing greater flexibility. Maxout is particularly beneficial in mitigating the dying ReLU problem and can lead to improved performance. However, it requires more parameters, which can increase computational complexity and the risk of over fitting if we do not manage it properly.

**ELU(Exponential Linear Unit)**

The ELU activation function, is a function that aims to improve learning by addressing issues present in ReLU. Unlike ReLU, ELU has negative values which help push mean activations closer to zero, thus speeding up learning. ELUs also reduce the vanishing gradient problem by having non-zero gradients for negative inputs. Their smooth curve can lead to improved performance in deep neural networks, though they are computationally more expensive than ReLU.

**Softmax Activation Function**

The Softmax activation function is primarily used in the output layer of classification networks. It converts a vector of values into a probability distribution, where the probabilities of all possible outcomes sum to one.  it highlights the most likely class by amplifying the largest input value. Softmax is essential for multi-class classification problems, enabling the network to assign a probability to each class. It provides a clear interpretation of model predictions but can be sensitive to outliers and extreme values in the input.softmax converts real vector to vector of categorical probabilities , where the elements of the output vector are in the range (0 , 1) and sum to 1. each vector gets handled differently.

### 2.3.4.3 Back Propagation

Back propagation is algorithm, which was introduced by Rumelhart and McClelland in 1986, it is employed in layered feed-forward neural networks. In these networks, artificial neurons are structured in layers, transmitting their signals forward, while errors are propagated backward. Neurons in the input layer receive the inputs, and the output layer provides the network's results. There can be one or multiple hidden layers.

Back propagation utilizes supervised learning, which involves providing the algorithm with input-output pairs that the network should learn to compute. The error, or the difference between the actual and expected outputs, is then calculated. The primary goal of the back propagation algorithm is [21] to minimize this error, enabling the neural network to learn from the training data. Training begins with random weights, which are adjusted iteratively to minimize the error as much as possible.

In neural networks utilizing the back propagation algorithm, the activation function of the neurons is typically a weighted sum, where the inputs (x) are multiplied by their respective weights (w) and summed.

The more the Complexity of a neural networks becomes, where its learning ability Increases requiring a fitting powerful learning algorithm like the back propagation algorithm, back propagation algorithm in used in real world to train neural networks.

For a neural network process, where the input layer is fed with sample and the sample is passed through each layer, where final result in given by the output layer, then the final output error is calculated and propagated back to the hidden layer so we can adjust the

values of the weights and thresholds this process is repeated by the back propagation algorithm until the required minimal training error value is reached.
we apply two type of back propagation algorithms, where each are tailored for different conditions in the training neural network, Standard BP algorithm it updates parameters of the neural network repeatedly Compared to the accumulated BP algorithm since each update uses a Single Sample, but the accumulated BP algorithm Back propagates less frequently Since its updates once after a full scene of the training set is complete.
Bp algorithms face over-fitting issues where the training error decreases but the testing error increases, to solve this issues we use strategies like, early stoppage, where we divide our input data into training section and testing section and we use the the training to compute the gradient which in turn use to update the weight , threshold ,and the testing to calculate the error , here we look for the condition where the train error is decreasing but the testing error increases, we stop the process & joyce the parameter where the training error in the lowest. The other method is called regularization, where we add regularization term of making a tradeoff between the empirical error and the Complexity of neural network.

### 2.3.4.4 Gradient Descent

Gradient decent is [8] an iterative optimization algorithm, which minimizes loss Function of a training model, by adjusting its weight of bias, it achieves the models maximum accuracy by [21] moving throughout the model iteratively, in a direction that decreases the cost function , where it finally converges to a local minimum, by estimating the gradient of the cost function with relation to the models parameters, which are updated in the opposite direction to the gradient estimate found by the algorithm , one of the parameters it manipulates to achieve the lowest loss function is the weight of the neurons, it starts by initializing there weight But after the prediction is made in the output layer, if the loss function is higher while Comparing the result with the actual output, it calculate gradient of the loss, and adjust the initialized weights iteratively until the lowest low function is reached.

**Gradient computation using the chain rule.**

BP algorithm uses Gradient decent to update its neural network parameters by applying this chain rule, we calculate the gradient loss function , and then We use their gradients to update the parameters of the neural network. Using optimization techniques, this process is carried repeatedly through out the training of the model.

### 2.3.4.5 Model Components/Hyper Parameters

**Hyper parameter tuning.**

the performance of the model we are trying to build a perfect is based on [17] the configuration of our neural networks parameters, which we constantly have to tune to get a better accuracy the parameters we tune include ,the size and number of our hidden layers , activation function , learning rate , batch size , number of epochs and regularization.

Hyper parameter play a major role in training a neural network model , where each parameter differ in terms of impacting he models accuracy, according to [7] Hyperparameters refer to parameters that remain fixed during the training of a machine learning model. They play a crucial role in defining the model's structure, such as the number of hidden layers and the choice of activation function, and in influencing the efficiency and accuracy of the training process, therefore making our selection of those parameters both in training as well as testing phase of the model very crucial, those parameters that greatly impact this include the optimizer batch size ,learning rate and network structure , and hyper parameter like the number of neurons and width of a hidden layers greatly influence the learning capacity as well as complexity of our model , making tuning of these parameters very crucial for the overall performance of the model we are creating.

**Learning rate**

learning rate is hyper parameter that manages how the values of neural networks weights are adjusted in the training phase in a way that leads the model in the opposite direction of the loss function, it adjusts the models to small positive value in the range of [0.0 - 1.0] , it determines the speed our model is being trained, given the set of resources which include the number and width of our hidden layer and the epochs numbers.

the learning rate [7] determines the weight of each node based on loss gradient. an efficient learning rate is the the one that low enough so that convergence is possible , but also high enough so we can train our model enough times , small learning rate leads to more epochs , while larger rate lead to further changes , but may lead to sub-optimal final weights.

**Number Of Hidden Layers And Number Of Neurons Per Layer**

Number of hidden layer and the number of neurons in each of the hidden layers is one of the hyper parameters that require tuning , [7] due to it’s high impact on the accuracy of our model. the number of hidden layers is greatly dependent on the complexity of the problem we are trying to solve or the level of non- linearity. as the Complexity increases the number of layers also increases from Shallow network to deep network.
we can tune the number of layers by gradually going from Simpler design to more complex one, at least the performance of each using Validation matrix , The number of of neuron in hidden layers affect how the neural networks lean successfully, therefore tuning these parameters in essential part of the neural network training process, since there are no general governing rules for this Parameter, but there are some to suggestions We can apply Like, size of all neurons In the hidden layer should be in between the input layer and output layer we can use rules like this as a starting point and go from here based on the accuracy & success of our model , throughout this process over fitting is an issue we face as our layers complexity increases we can use techniques such as dropout or regularization to avoid over fitting.

**Batch Size And Number Of Epochs.**

Epochs and batch size dictate [21] how data is utilized during network training. The dataset is divided into training, validation, and test sets. Training data is used to adjust the neural network's weights and biases. This data is further divided into smaller batches, and for most optimizers, network weights are updated after processing each batch. Specifically, Adam optimizers adjust weights and biases post batch processing, while regular gradient descent updates weights either after processing each training sample or after the entire training set, making batch size irrelevant in this case. An epoch is defined as one complete pass of all batches, and therefore all training samples, through the network. After each epoch, the network's performance is assessed using the validation set by calculating the loss, which, while not used for weight updates, is crucial for evaluating the network's generalization to new data. This process of training on the same samples and validating with the validation set is repeated for a set number of epochs.

## 2.3.4.6 Optimizers

One of the most Challenges in neural network is achieving maximum accuracy, but these requires valuable time Consumption , this can be Solved using optimization algorithms, it's an algorithm that can be used in different neural networks, by optimizing their loss function , optimization algorithms are techniques used by neural networks to build models, where they help

the model reach minimal cost function, by [21] adjusting its parameters to reach maximum accuracy level.
Optimization algorithm we choose for our neural network, determines the learning speed and performance, but the only way we can choose the best optimization is to do a detailed analysis of each technique and tailor them to our specific model , parameters , and datasets.

**SGD**

SGD is an optimization technique , that is used when we have large dataset, it works by only selecting single Sample size to update parameters and it does this frequently , and due to SGD's highly frequent parameter update convergence will happen faster.

**Momentum**

Momentum is an optimization technique for gradient descent that incorporates a portion of the previous update vector into the current update vector, thereby accelerating the learning process. This method smoothes out updates to model parameters, helping the optimizer maintain its direction from earlier iterations, which reduces oscillations and increases convergence speed. More precisely, momentum can be described as the exponentially weighted moving average of past gradients. Rather than updating parameters based solely on the current gradient, the optimizer uses this moving average, which acts as a memory, enabling the optimizer to retain and follow its previous direction even if the current gradient suggests a different path. momentum is a strong optimization technique, that helps Speed up convergence, Minimize ossilation, avoid local minima, and make the optimization process more resistant to noisy gradients, it's usually coupled with other optimization technique like SGD and adaptive learning rate , and to get the best accuracy result for the model we are building, it's essential to adjust the parameters of our momentum , so that the momentum coefficient end the learning rate.

**SGD with Momentum**

SGD with momentum is a commonly used optimization technique, where it converges an quickly as SGD for Smooth objectives and benefits from an multistage Strategy with dynamic parameters. SGDM (Stochastic Gradient Descent with Momentum) is commonly utilized with diverse parameter configurations aimed at optimizing training efficiency. One prevalent strategy is known as "Constant and Drop," where a consistent step size is maintained initially and then reduced by a constant factor to facilitate fine-tuned training. Throughout this process, the momentum weight remains either constant or gradually increases, contributing to improved convergence and stability in the optimization process , SGDM is applied to train their large Scale neural networks and with appropriate parameter tuning we were able to achieve Superior performances. As we approach the minimum, our aim is to achieve a gradual convergence. However, prior to reaching this point, employing a very low learning rate could significantly prolong the process of reaching the minimum. Conversely, using learning rates that are too small to prevent oscillations along ridges might give the impression that the loss isn't improving, potentially causing practitioners to abandon the training algorithm prematurely.

**adam optimizer**

Adam, short for Adaptive Moment Estimation, represents an advanced optimization algorithm used in lieu of traditional stochastic gradient descent for updating neural network parameters iteratively during training. It excels in handling large datasets and numerous parameters efficiently, consuming less memory. Essentially, Adam combines elements of gradient descent with momentum and the RMSprop algorithm. It dynamically adjusts the learning rate for each parameter, ensuring effective optimization and convergence, especially in complex loss landscapes.

To counter initialization biases, Adam incorporates a bias correction mechanism in its first moment, promoting quicker convergence in early stages of training. Its overarching objective is to stabilize and expedite the training process, guiding neural networks towards optimal solutions by efficiently navigating both steep and flat regions of the loss function. By leveraging squared gradients akin to RMSprop and incorporating momentum via a moving average, Adam integrates dynamic learning rates and smoothing techniques to facilitate convergence towards global minima.

## 2.3.4.7 Loss Function

Loss Function helps in the Building & improving the performance guiding optimization process, we can apply loss function for tasks both regression and classification in order to [21] minimize loss function of a model we build. we can also measures how well the models predictions watch the actual target Values, and quantifies this difference between the actual  measure & the predicted one as an error to  minimize this difference of error, parameters of our model such as (weights and biases) are adjusted

while training M.L models we are required to assign objective functions or loss function which are used to measure our models performance, and their values are Constantly optimized for better performance. The form of loss function depends on multiple conditions including the nature of our problem we are trying to solve, the algorithm were are implementing as well as the data the are using to train our algorithm with.

A loss Function *L*, is defined as f($x_i$) with it's Corresponding $y_i$ to a real number L ЄR, which captures the similarities between f($x_i$) and $y_i$ Aggregating over all the points of the data set we find.

- the overall loss , *L*:

$$L(f|\{x_0, \ldots, x_N\}, \{y_0, \ldots, y_N\}) = \frac{1}{N}\sum_{i=0}^{N}(f(x_i) - y_i)^2$$

Figure 2.4 : loss function

**Mean Squared Error (MSE) for regression**

Mean Squared Error (MSE) in regression calculates [21] the average of the squared differences between predicted values ў and observed values y . Squaring these differences ensures all biases are positive and amplifies the impact of outliers, making MSE particularly effective in scenarios where observation noise conforms to a normal distribution.

$$\mathbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Figure 2.5 mean square error

**Cross-Entropy Loss for classification.**

Cross-entropy loss is a type of function, used for clarification Tests, where it calculates the change between the true label and the one Predicted by our model; it works by paralyzing the predicted Values that are vastly different from the actual the label.

$$L_{\mathrm{CE}} = -\sum_{i=1}^{n} t_i \log(p_i), \quad \text{for n classes,}$$

where $t_i$ is the truth label and $p_i$ is the Softmax probability for the $i^{th}$ class.

Figure 2.6 cross entropy

### 2.3.4.8 Multi Layered Perceptron(MLP)

In 1940 the idea of neural networks was conceived by computer scientists , in the early 80’s those concepts led to what is called the Multi layer perceptron (MLP) algorithm ,and this algorithm (MLP) was used to solve non-linear problems by using it’s hidden layers and activation function. The Multi layer perceptron (MLP) algorithm by it’s nature is an easy to train and implement algorithm ,it’s used to find correlation and patterns in a given dataset , by design it’s [18] a common type of feed forward neural network algorithm that’s implemented in different programs , both in classification which is why we chose it since we are using it to classify weather to irrigate or not irrigate decision based on soil chemical data, , the MLP algorithm has been used in varying degree with accurate results depending on the given data’s attributes (quality and quantity).The MLP algorithm according to [9][15] is very powerful technique within the neural network filed , especially for tasks involving very complex , non-linear relationships ,that are based on real world problems ,by designing the architecture of the algorithm ,preprocessing a given dataset and applying appropriate training and evaluation techniques , this algorithm is being currently used in fields [19] like voice recognition, visual data recognition, and machine translation and many more.

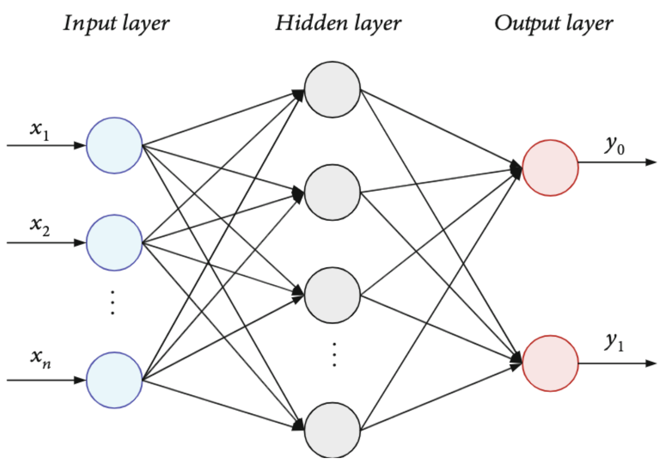


Figure 2.7 : MLP with single hidden layer

Multi layer perceptron (MLP) has it's own set of limitations being unable to compute complex architecture and big data leading to performance decline in the model at the time , and in the late 80's the algorithm back propagation was introduced ,researchers were abeld to increase the performance scale of the Multi layer perceptron (MLP) algorithm by using the back propagation algorithm , which helps the Multi layer perceptron (MLP) algorithm by adjusting it's weight and helping it minimize error.the development of optimization technique also helped the Multi layer perceptron (MLP) algorithm be more effective algorithm in the current time ,later in the 2010's future advancements in the graphics (GPU) technology also were effective in allowing the Multi layer perceptron (MLP) algorithm to create a more complex neural network architecture.

**Layer Connectivity**

A layer that is connected fully also called dense layer ,is a neural network layer design where each neuron is connected to each neuron in the previous layer and the next layer. this means that every input is used in every output computation ensuring maximum interaction between layers. This means that every input is used in every output computation ensuring maximum interaction between layers.

Fully Connected layers are parameterized by their weights and biases, which are learned during training using optimization algorithms like gradient descent ,hence they can capture complex patterns and relationships in the data due to their dense connectivity, They can be used in Various types of neural networks and for different tasks, including classification, regression, and featuring extraction. The architecture of fully connected layers is straight forward and easy to implement, making a fundamental building block in neural networks.

Fully Connected lagers are fundamental components of MLP, enabling dense connectivity between neurons across layers. this connectivity allows MLP's to learn and model complex patterns in data. While they offer significant expressive power and versatility, they also come with challenges like high parameters count and inefficiently for high dimensional data, understanding the role and functioning of fully connected layers is crucial for effectively designing and implementing neural network
models.

In a Multilayer Perceptron (MLP), all layers are fully interconnected. We denote the function of a fully connected layer as $y = fc(x, w, b)$ , where x is the input to the layer, w represents the weight matrix, b is the bias vector, and y denotes the output.

Each layer in an MLP consists of numerous neurons, with each neuron in one layer connected to every neuron in the next layer via weighted connections. The first layer is known as the input layer. The subsequent layers, termed hidden layers, do not directly interface with external inputs or outputs. The final layer is referred to as the output layer.

**Description Of The Basic Architecture**

The Multi layer perceptron (MLP) algorithm is structured according to [17] in a way the signals that are received from external environment are passed in one direction , starting from the input layer to the output layer ,where the output of every neuron doesn't affect the neuron itself this structure is called feed forward , the hidden layer is what gets this neural network the name multi layerd since the input layer is not counted in terms of it being not abeld to do anything but pass the input data to the next layer.

The Multi layer perceptron (MLP) neural network are structurally in each neuron in the hidden and output layer with an activation function (non-linear) ,and this makes them very effective neural network, making MLP act as universal approximators ,and this activations are calculated where each neurons weighed sum of its inputs which area added to a constant and this value is calculated in a non-linear function also known as activation function. and this values are passed the next neuron since its structurally feed forward mannered.

In order to solve non-linear separable problems we can use The Multi layer functional neurons , in the fig the hidden layer that is found in between the input and output layer has an activation function for each neurons ,and so does the output layer in the multilayerd neurons. structurally the neurons with in the three layers are connected fully to the next layers parallel to them , but neurons with in the same from non-adjacent layers are not connected , this type of structure implemented by this multi layerd neuron is called feed forward , in which the input layer receives a signal from the outer environment , and the hidden and output layers process this signal and finally the output layer outputs the proccessed signal , in this scenario the input layer neurons function is to pass the input signals to hidden layer neurons the learning process of neural networks is about learning from the training data(signal) to adjust the connection weights among neurons and the thresholds of functional neuron.

## 2.3.4.9 Recurrent Neural Network (RNN)

To build our water management model on irrigated farmlands on algorithm that is useful due to it's ability to handle sequential data like soil chemical properties effectively is Recurrent Neural Networks, Recurrent Neural Networks is a types of neural networks that is made up of a set of layers connected in a feed forward way that is according to [Pramita] similar to human neural system , they are made up of multiple layers designed for a set of sequence of data’s ,The RNN model has a one-way flow of information from the input units to the hidden units and a directional loop that compares the error of this hidden layer to that of the previous hidden layer, and adjusts the weights between the hidden layers. the hidden state that captures information about previous inputs in the sequence, allowing them to model temporal dependencies.

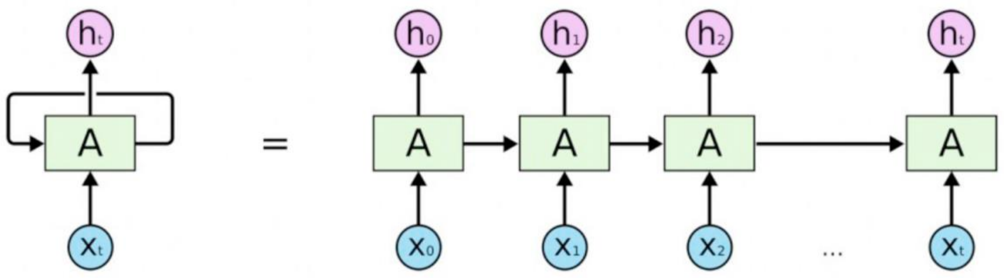


Figure 2.8 : RNN Architecture

Recurrent Neural Networks (RNN), the hidden state update formula $h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$ encapsulates the recurrent nature of the network, where each hidden state $h_t$ depends on the current input $x_t$ and the previous hidden state $h_{t-1}$. In this formula, $x_t$ is the input vector at time step $t$ , while $W_{xh}$ and $W_{hh}$ are weight matrices applied to the current input and the previous hidden state, respectively. The bias vector $b_h$ is added to introduce an offset that can improve the model's fit to the data. The activation function σ, such as the sigmoid or tanh function, is applied to the combined input and hidden state, introducing non-linearity and enabling the network to capture complex patterns. At each time step, the RNN updates its hidden state by first performing linear transformations on the current input and previous state, summing these results with the bias, and then applying the activation function. This process allows the RNN to maintain a form of memory across time steps, which is essential for tasks involving sequential data, such as time series prediction and natural language processing, where the order and context of data points are important. Understanding and implementing this formula is crucial for effectively using RNN.

$$h_t = \sigma\ (W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

Figure 8.9 : Hidden State Update Formula in RNN

However, RNN also face limitations such as difficulty in learning long-term dependencies due to issues like vanishing gradients and exploding gradients ,due to its inability to memorize during back propagation where gradient decent weight adjustment are not fully implemented , these shortcomings of regular RNN Algorithm are solved by LSTM and GRU algorithms.

**Long Short-Term Memory (LSTM) Networks**

LSTM Algorithms that is according [nigatu] designed by Hochreiter and Schmidhuber specifically designed to solve sequential non-linear data's like our soil chemical properties gathered from a farmland, which is useful in managing water prediction, LSTM are used to address the limitations of simple RNNs, particularly [nigatu] [Pramita] the vanishing gradient problem. they solve this issue by replacing all units in the hidden layer with memory cells where each memory cell according to [Pramita] has at least a single memory cell , this memory cell includes input gates, forget gates, and output gates. The memory cells maintain the cell state, which acts as a long-term memory, allowing the network to remember crucial information over extended sequences. The input gate controls the extent to which new information flows into the cell state, the forget gate regulates the removal of information no longer needed, and the output gate determines the amount of cell state information that influences the output at each time step. This gated mechanism enables LSTM to effectively handle the complexities of sequential data, such as soil chemical properties over time, and make accurate binary classification decisions on whether to irrigate or not based on the evolving patterns in the data. The ability of LSTM to preserve and utilize information over long sequences makes them a robust choice for tasks involving time-series prediction and sequential classification.

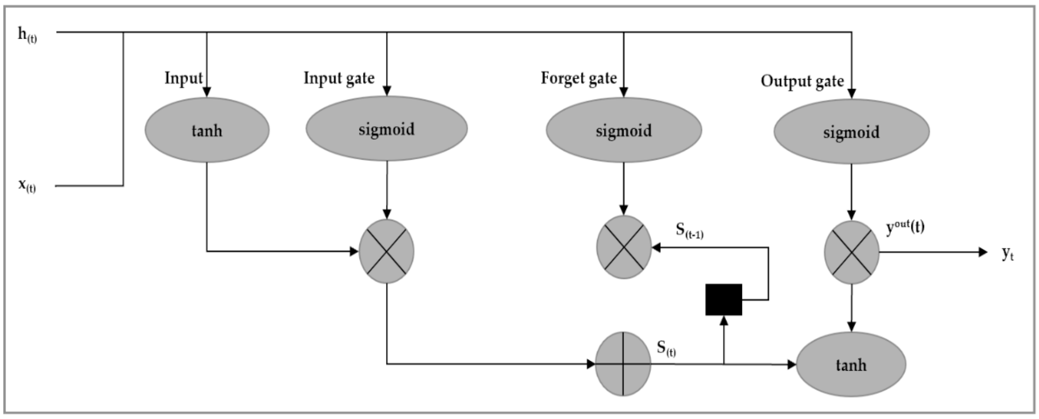


Figure 2.10 : Single Cell in LSTM

**Gated Recurrent Unit (GRU) Networks**

Gated Recurrent Unit is a simplified version of LSTM Algorithm that aim to achieve similar performance with fewer parameters.

The gated recurrent unit (GRU) is an advanced form of recurrent neural network (RNN) that builds upon the principles of long short-term memory (LSTM). While it shares similarities with the LSTM unit, particularly in combining the input and forget gates into a single update gate, the GRU is generally simpler to compute and implement. This simplicity arises from its more streamlined internal structure, which reduces the number of computations required for updating the internal state, thereby facilitating training. Like the LSTM, the GRU effectively mitigates the issue of vanishing gradients. The GRU features two gates: a reset gate, which determines the extent to which the current state should integrate with historical data, and an update gate, which manages how much information from the previous state is retained in the current state. The mathematical operations governing the GRU cell's gating mechanism are detailed as follows: [Q. Kang]. In these equations , $W_z$ , $W_r$ , and $W$ denote the weight matrices for the corresponding input vectors, while $b_z$ , $b_r$ , and $b$ represent biases. $U_z$, $U_r$ , and $U$ are the weight matrices from the preceding time step. The candidate hidden state is denoted by $h_t$ , the update gate by $z_t$ , the reset gate by$r_t$, and the logistic sigmoid function by σ.

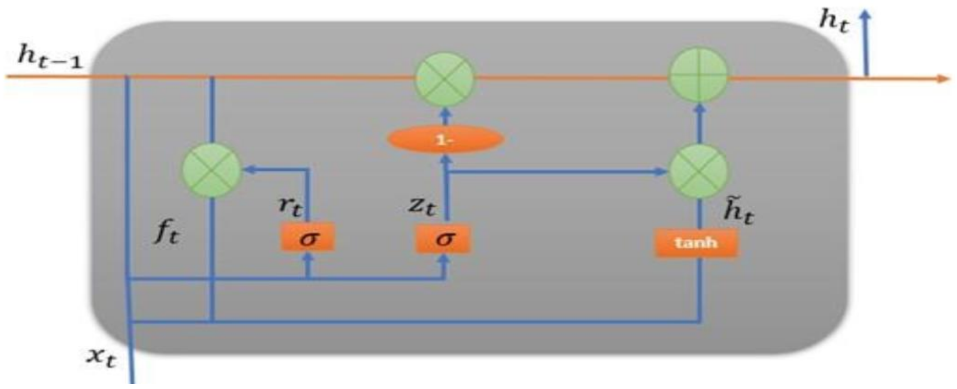


Figure 2.11 : cell structure of a gated recurrent unit

## 2.3.5 Model Evaluation

**Metrics for Evaluation**

Evaluation metrics are [9] essential components of machine learning tasks, tailored specifically to tasks such as classification and regression. While some metrics like precision-recall are versatile across different tasks, supervised learning, which includes classification and regression, dominates the field of machine learning applications. Employing diverse classification metrics for performance evaluation enhances the predictive accuracy of models before deployment on unseen data. Relying solely on accuracy without a comprehensive evaluation using diverse metrics can result in poor predictions when the model is deployed in real-world scenarios.

**Accuracy**

Accuracy measures the frequency with which a classifier predicts correctly. It is defined as the ratio of correct predictions to the total number of predictions.

**Confusion**

The confusion matrix is a pivotal tool for evaluating the performance of machine learning classification models, particularly in scenarios where there are multiple classes. It provides a table of predicted versus actual values and is crucial for assessing metrics such as Recall, Precision, Accuracy, and AUC-ROC curves.

**Precision**

Precision indicates the proportion of correctly predicted positive cases out of all predicted positive cases. It is particularly valuable in situations where false positives are more concerning than false negatives, such as in music or video recommendation systems and e-commerce platforms.

Precision for a label is defined as the number of true positives divided by the number of predicted positives.

**Recall**

Recall measures the proportion of actual positive cases correctly identified by the model. It is essential in scenarios where false negatives are more critical than false positives, such as in medical diagnostics, where detecting all positive cases is crucial.

By rephrasing each section, we aim to present the information in a unique and distinct manner while preserving the core concepts and information.

**F1-Score**

F1-Score gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall.

## 2.4 Review of Related Work

### 2.4.1 Local Related Work

**Smart and Intelligent Irrigation System (SI2S)**

The thesis titled "Smart and Intelligent Irrigation System: A Machine Learning and IOT" is are search done by the authors Samuel zeyede and asrat mulatu , inorder to create an AI model by integrating ML Algorithms with IOT sensor devices on irrigation fields found in debre-zeit Agricultural research center which is found in oromia regional state in east shoa zone , where the researchers used both primary as well as secondary data Garlic vegetation , where these data's were collected using IOT sensors by planting them on the fields , and these collected data's were collected using micro-controllers like raspberry pi and uploading them on cloud think speak, the also put their model on the cloud , the authors used two machine learning algorithms , linear regression algorithms for prediction and decision trees algorithms for categorization of data's , and they were able to save water usage by the garlic farm by percentage of 6.45% during initial period and about 6.75% during development phases.

### 2.4.2 Global Related Work

**An Intelligent Irrigation System Based On Internet of Things To Minimize Water Loss.**

This thesis was authored by Samar Amassmir and Co where they proposed which of the three machine learning Algorithm fit much more in an irrigation system where water management is crucial these Algorithms are K Nearest Neighbour , Support Vector Machine and Artificial Neural Network, the authors were able to make a comparative outcome of the three by collecting set of soil chemical data by using IOT systems called Temperature and humidity sensors where these data are sent to micro controllers like Arduino and raspberry pi where the model are stored , and based on the data which all three models which are based on the three machine learning Algorithm they were able to train models with accuracyof91%(KNN Algorithm ), 87%(SVM Algorithm) and 96.8% ( ANN) , from this accuracy results the authors were able to conclude KNN algorithm was the best result of the for more accurate model building based on the data they input.

# Chapter -3

# 3. Methodology

## 3.1 Introduction

In this section we discuss about the Deep Learning based model we prepare to predict irrigation water needs based on a set of soil chemical properties That are collected from irrigated farms in Ethiopia, thereby improving water management in controlled irrigation systems. This methodology section we describe the set of techniques and methods to gather ,select and preprocess the set of data’s needed , and also describe how the model we will build will be structurally designed using set of algorithms and In combination with optimization and activation function , we picked to train and make future predictions based on those prepared data’s.

## 3.2 Dataset Collection

For the irrigation prediction model we will be building in this experiment we got set of irrigation soil chemical data set from the region of Oromia , East Shewa Zone , Adama Woreda , which is found in the Central region of Ethiopia ,which we were provided by a governmental institute namely Ethiopian Agricultural Transformation Agency (EAIA) , the data was collected by the institute from an irrigation farmlands surrounding awash melka area ,we requested those data’s both in person and online communication where ethiopian agricultural transformation agency (EAIA) requested some legal documents to be provided for them to provide what we requested and we complied with their demand  and the data was provided to us via email , the set of data we received from the  Ethiopian Agricultural Transformation Agency (EAIA) institute include a set of chemical properties ,crop type , soil type, personnel and location detail.

## 3.3 Feature Selection

The set of data’s we received includes over 90 features which include numerical , textual / categorical values and to make these data’s usable by the model we will build we need to analyze and prepare (preprocess) it for the machine using a set of python libraries to handle the difference set of issues with our data and multiple preprocessing techniques, In the dataset we still have irrelevant and redundant features that we need to remove like long, lat and zone which are redundant as well as irrelevant to the development and performance of the model like personnel who performed the sampling and soil color and soil type as well as type of crops , and when we go once step further we also need to pick features that are contributing to the accuracy and relevancy of the model we are building and we can distinguish those features by applying which chemical features are relevant to the water threshold in a soil by the chemical property analysis., using those techniques above we selected 17 chemical feature for our model to handle.

## 3.4 Dataset Preprocessing

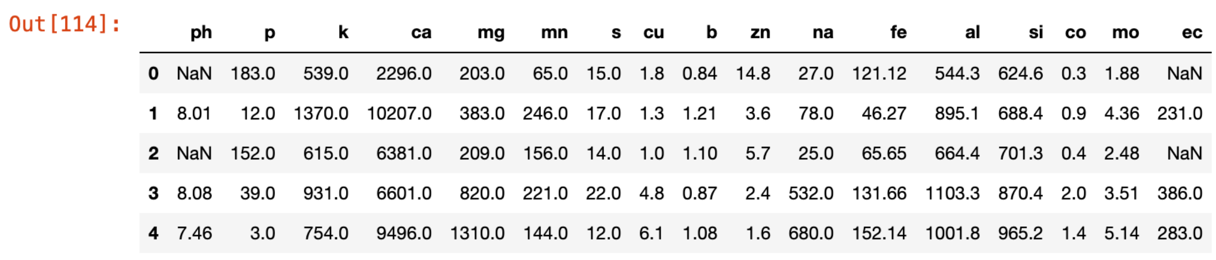
Data preprocessing is essential part of model building which involves preparing and transforming raw data before it can be used for model building and analysis.

The sets of data’s collected by The Ethiopian Agricultural Transformation Agency (EAIA) institute contains noisy , irrelevant and missing data’s ,and other expert based analysis , therefore we need to preprocess the data we have gathered , and in the next steps we’ll apply the preprocessing tasks.

### 3.4.1 Dataset Cleaning

The data collected from a different locations in farmland by the institute have missing values , and we found in order to solve these missing data’s we use an imputation method called mean imputation to replace those missing data values we did this due to the small scale of our data we will use to train our model.

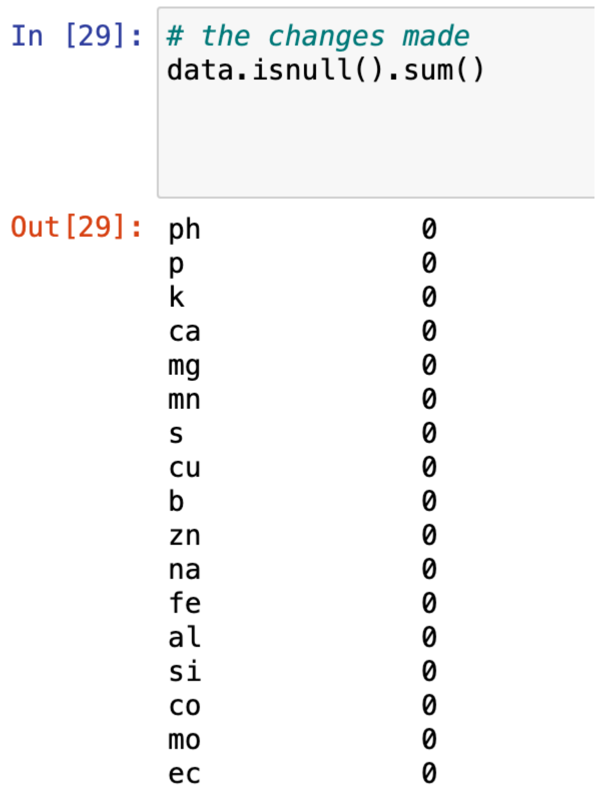
Our machine learning algorithms can’t handle missing data directly, therefore data preprocessing allows the algorithms by handling missing values by either removing instances with missing values by imputing missing values with reasonable estimates or with more advanced imputation methods like mean imputation , which we implemented for the missing values in our dataset.



Out[114]:

| | ph | p | k | ca | mg | mn | s | cu | b | zn | na | fe | al | si | co | mo | ec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NaN | 183.0 | 539.0 | 2296.0 | 203.0 | 65.0 | 15.0 | 1.8 | 0.84 | 14.8 | 27.0 | 121.12 | 544.3 | 624.6 | 0.3 | 1.88 | NaN |
| 1 | 8.01 | 12.0 | 1370.0 | 10207.0 | 383.0 | 246.0 | 17.0 | 1.3 | 1.21 | 3.6 | 78.0 | 46.27 | 895.1 | 688.4 | 0.9 | 4.36 | 231.0 |
| 2 | NaN | 152.0 | 615.0 | 6381.0 | 209.0 | 156.0 | 14.0 | 1.0 | 1.10 | 5.7 | 25.0 | 65.65 | 664.4 | 701.3 | 0.4 | 2.48 | NaN |
| 3 | 8.08 | 39.0 | 931.0 | 6601.0 | 820.0 | 221.0 | 22.0 | 4.8 | 0.87 | 2.4 | 532.0 | 131.66 | 1103.3 | 870.4 | 2.0 | 3.51 | 386.0 |
| 4 | 7.46 | 3.0 | 754.0 | 9496.0 | 1310.0 | 144.0 | 12.0 | 6.1 | 1.08 | 1.6 | 680.0 | 152.14 | 1001.8 | 965.2 | 1.4 | 5.14 | 283.0 |

Figure 3.1 :data sample with missing values



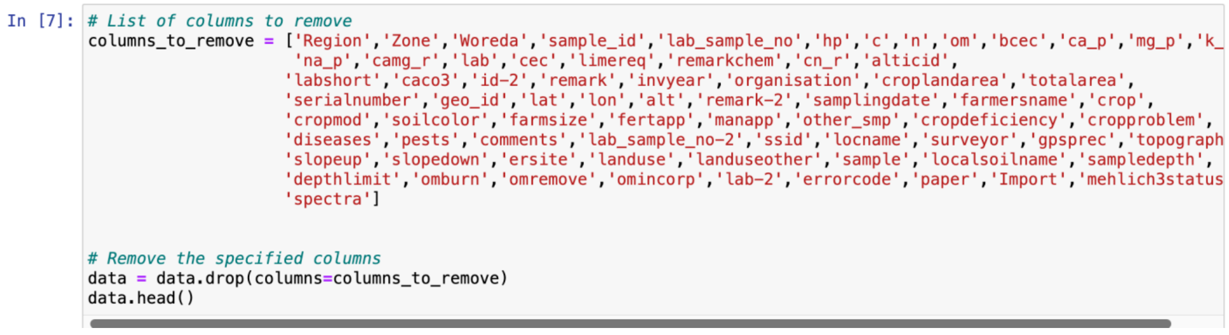
```
In [29]: # the changes made
         data.isnull().sum()

Out[29]: ph            0
         p             0
         k             0
         ca            0
         mg            0
         mn            0
         s             0
         cu            0
         b             0
         zn            0
         na            0
         fe            0
         al            0
         si            0
         co            0
         mo            0
         ec            0
```

Figure 3.2 : number of missing values after

## 3.4.2 Dataset Reduction

In this step we remove unnecessary feature that does not help our model features like sample_id , Lat , Lon, Woreda and others they are not complementary to our model therefore we remove them. , and in the end we removed over 72 features which we deemed unnecessary having no effect on our models performance .



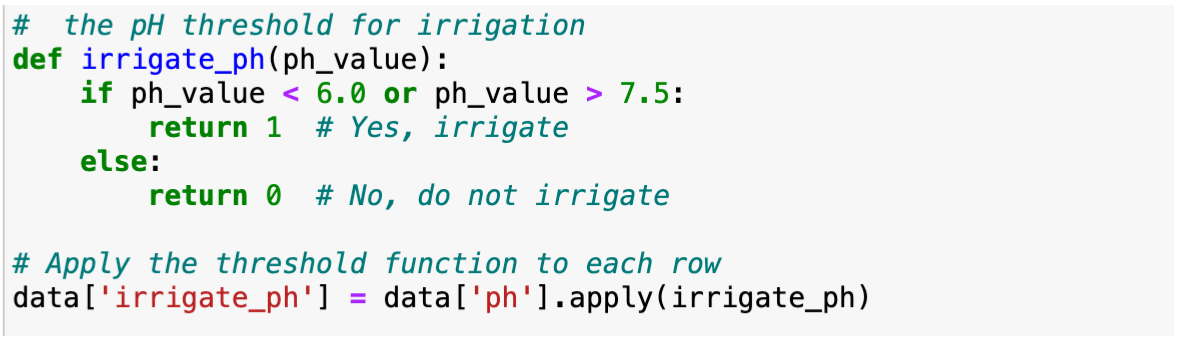
```
In [7]: # List of columns to remove
        columns_to_remove = ['Region','Zone','Woreda','sample_id','lab_sample_no','hp','c','n','om','bcec','ca_p','mg_p','k_
                              'na_p','camg_r','lab','cec','limereq','remarkchem','cn_r','alticid',
                             'labshort','caco3','id-2','remark','invyear','organisation','croplandarea','totalarea',
                             'serialnumber','geo_id','lat','lon','alt','remark-2','samplingdate','farmersname','crop',
                             'cropmod','soilcolor','farmsize','fertapp','manapp','other_smp','cropdeficiency','cropproblem',
                             'diseases','pests','comments','lab_sample_no-2','ssid','locname','surveyor','gpsprec','topograph
                             'slopeup','slopedown','ersite','landuse','landuseother','sample','localsoilname','sampledepth',
                             'depthlimit','omburn','omremove','omincorp','lab-2','errorcode','paper','Import','mehlich3status
                             'spectra']


        # Remove the specified columns
        data = data.drop(columns=columns_to_remove)
        data.head()
```

Figure 3.3 :feature reduction

### Thresholding / Weighted Sum

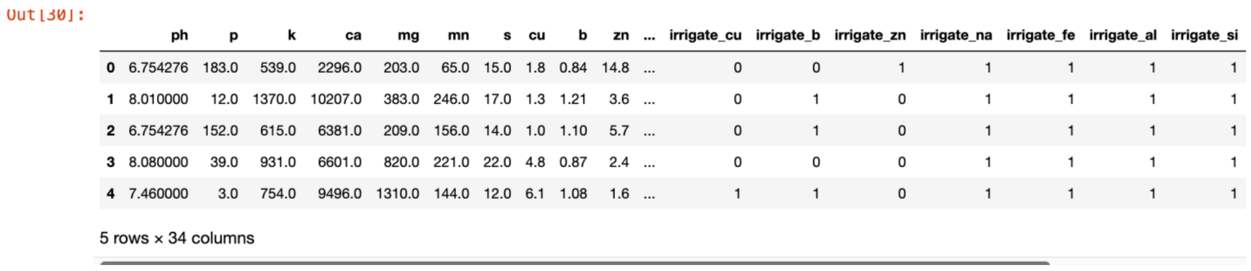
After finishing the above preprocessing steps to the three files we , we give each of our chemical properties in our dataset their own irrigate threshold using thresholding technique , since each chemical property has its own property and value range in relation to moisture level in the soil therefore we will be forced to assign each its own water threshold classification with 1 being to the decision irrigate and 0 being to the decision not to irrigate.



```
#  the pH threshold for irrigation
def irrigate_ph(ph_value):
    if ph_value < 6.0 or ph_value > 7.5:
        return 1  # Yes, irrigate
    else:
        return 0  # No, do not irrigate

# Apply the threshold function to each row
data['irrigate_ph'] = data['ph'].apply(irrigate_ph)
```

Figure 3.4: sample threshold



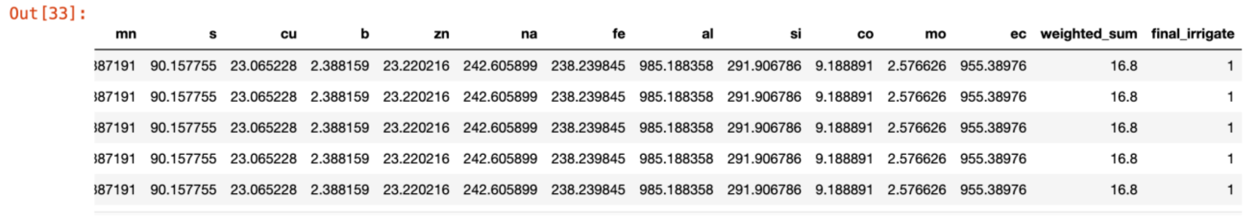
Out[30]:

|  | ph | p | k | ca | mg | mn | s | cu | b | zn | ... | irrigate_cu | irrigate_b | irrigate_zn | irrigate_na | irrigate_fe | irrigate_al | irrigate_si |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 6.754276 | 183.0 | 539.0 | 2296.0 | 203.0 | 65.0 | 15.0 | 1.8 | 0.84 | 14.8 | ... | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 8.010000 | 12.0 | 1370.0 | 10207.0 | 383.0 | 246.0 | 17.0 | 1.3 | 1.21 | 3.6 | ... | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 2 | 6.754276 | 152.0 | 615.0 | 6381.0 | 209.0 | 156.0 | 14.0 | 1.0 | 1.10 | 5.7 | ... | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 3 | 8.080000 | 39.0 | 931.0 | 6601.0 | 820.0 | 221.0 | 22.0 | 4.8 | 0.87 | 2.4 | ... | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 4 | 7.460000 | 3.0 | 754.0 | 9496.0 | 1310.0 | 144.0 | 12.0 | 6.1 | 1.08 | 1.6 | ... | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

5 rows × 34 columns

Figure 3.5 : sample chemical properties irrigate threshold

To finally have a single irrigate decision that we can use as the **y** dependent value for our model we have to apply weighted sum method where each chemical property is weighted based on its importance to measuring water moisture this weight is made by irrigation experts who value and rank these properties based on analysis of these properties in relation to their level of effectiveness in telling how moist a soil is and we have explained these in detail in chapter 2 section , for it to need be irrigated or not , and set a conservative threshold (70% of the sum of weights) so that it needs most conditions to be in dire situation to apply water so it can conserve water as much as possible. and finally we remove those chemical property thresholds since we have the weighted sum of each.



Out[33]:

| mn | s | cu | b | zn | na | fe | al | si | co | mo | ec | weighted_sum | final_irrigate |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 387191 | 90.157755 | 23.065228 | 2.388159 | 23.220216 | 242.605899 | 238.239845 | 985.188358 | 291.906786 | 9.188891 | 2.576626 | 955.38976 | 16.8 | 1 |
| 387191 | 90.157755 | 23.065228 | 2.388159 | 23.220216 | 242.605899 | 238.239845 | 985.188358 | 291.906786 | 9.188891 | 2.576626 | 955.38976 | 16.8 | 1 |
| 387191 | 90.157755 | 23.065228 | 2.388159 | 23.220216 | 242.605899 | 238.239845 | 985.188358 | 291.906786 | 9.188891 | 2.576626 | 955.38976 | 16.8 | 1 |
| 387191 | 90.157755 | 23.065228 | 2.388159 | 23.220216 | 242.605899 | 238.239845 | 985.188358 | 291.906786 | 9.188891 | 2.576626 | 955.38976 | 16.8 | 1 |
| 387191 | 90.157755 | 23.065228 | 2.388159 | 23.220216 | 242.605899 | 238.239845 | 985.188358 | 291.906786 | 9.188891 | 2.576626 | 955.38976 | 16.8 | 1 |

Figure 3.6 : weighted final result

### 3.4.3 Dataset Normalization

Normalization is a data preprocessing technique implimented by neural networks, it’s used to scale features of the input to the range of in between 0 and 1, they help by changing the inputs values of different ranges added to the neural network to similar range. this technique is applied either when the distribution of the data cannot be found or the distribution of the data input is not normal.

In our research, we employ the scikit-learn MinMaxScaler function to normalize our dataset. This function adjusts the data to a specific range, aiding in the normalization of each feature. By default, the MinMaxScaler function scales each feature independently to ensure values fall within a designated minimum and maximum range, typically 0 to 1.

## 3.5 Data Splitting

Models are shaped to make to our specific problem we are trying to solve weather it is to make prediction or categorization using a set of preprocessed data we did earlier, where this data is used both to shape as well as to validate the accuracy of our model by splitting them into two sections namely training set ( a set that is visible to the model) and testing set that is invisible to the model during training phase.

The data set we use to train our model contains 18 features ( ph, p , k , ca , mg , mn , s, cu , b , zn , na , fe , al , si , co , mo , ec  , final_irrigate ) , we divided the data set in ratio of 80/20  80 % to train and 20% to test , and catagorize the (ph, p , k , ca , mg , mn , s, cu , b , zn , na , fe , al , si , co , mo , ec  ) datas as independent and (final_irrigate) data as dependent.

## 3.6 Hyper Parameter Tuning

The performance of the model we are trying to build a perfect is based on the configuration of our neural networks parameters, which we constantly have to tune to get a better accuracy. hyper parameter play a major role in training a neural network model , where each parameter differ in terms of impacting the models accuracy , therefore making our selection of those parameters both in training as well as testing phase  of the model very crucial, those parameters that greatly impact this include the optimizer  batch size ,learning rate and network structure , and hyper parameter like the number and width of a hidden layers greatly influence the learning capacity as well as complexity of our model , making tuning of these parameters very crucial for the overall performance of the model we are creating. for our model we will be implementing set of hyper-parameters . We used selected our machine learning model, optimizer, activation, learning rate, batch size, epochs, dropout and neurons as a parameter configuration.

## 3.7 Model Design and Development

Our thesis implements MLP, LSTM and GRU algorithms, to predict when to and not to apply water to an irrigated farmland soil, we selected these algorithm due to their ability to solve non-linearity issues, and can be utilized in time series problems to achieve maximum accuracy we also implementing optimizers (SGDM and Adam) and activation functions (sigmoid , ReLU and relu) to make our model more accurate based on the best combination of the two.

## 3.8 Model Training

We will build a classification model which we will be using to predict future events in an irrigation filed water management where the model will be built using a set of chemical properties that are independent variables and a decision to be predicted, a combination of these set of data's and an a Deep Learning algorithms will give us a predictive model we require, the accuracy and success of our model is dependent on set of hyper parameters we define and manage during this training phase.

## 3.9 Evaluation Metrics

The performance of our model is assessed using cross entropy loss is a type of loss function, used for clarification Tests, where it measures the difference between the true label and the one Predicted by our model; it works by analyzing the predicted Values that are vastly different from the actual the label.

## 3.10 Proposed Model and Architecture

We will build a Deep Learning model (MLP, LSTM, GRU) with a set of soil chemical property data set and preprocess each data, where the selected soil chemical features of our data set are each given their own threshold using

Threshold-Based Labeling by defining thresholds for soil chemical properties that indicate when to apply water , and finally combine these with each soil property features are combined and each dependent feature is combined to single value using weighted sum a method by which each chemical properties threshold is given a weight of its own to decide the combined result(final irrigate) feature , the final preprocessed set with the chemical properties (ph, p , k , ca , mg , mn , s, cu , b , zn , na , fe , al , si , co , mo , ec ) and an irrigation decision result column (final_irrigate) is then split into train test and the training data is fed to The Algorithms (MLP , LSTM ,GRU), and the new trained model is then tested on the testing data to measure accuracy of the model, thus our model be helping predict future water management decisions based on set of chemical properties of a soil.

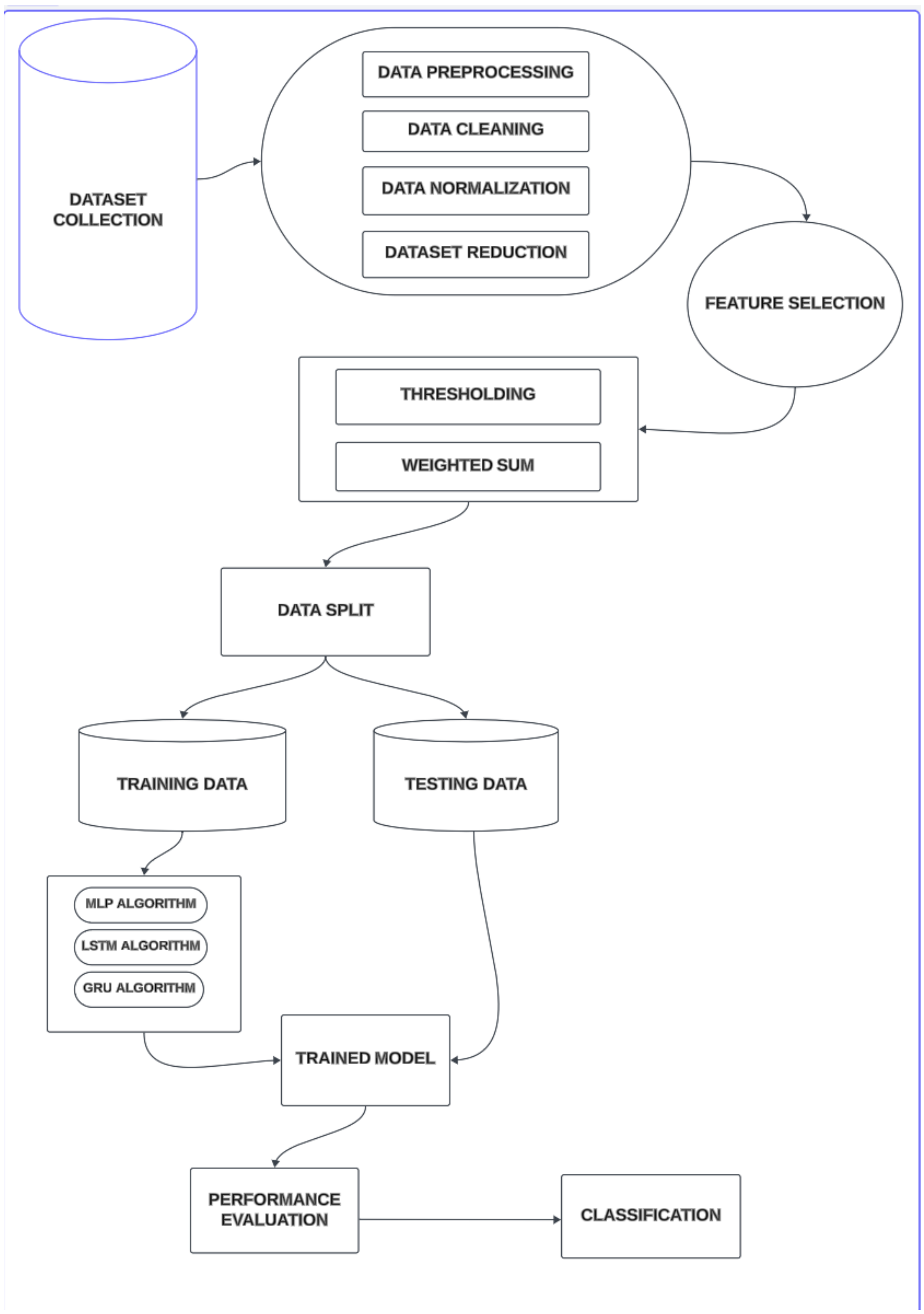


Figure 3.7 : experimental architecture of Our model

## 3.11 Software and Tools Used

We use multiple software and hardware to build our model, both local and remote tools like Google colab , we list each tool below:

**Hardware**

| **manufacturer** | **apple** |
|---|---|
| **model** | **Mac Book Pro version 12.7.5** |
| **processor** | Dual-Core Intel Core i5 |
| **processor speed** | 2.7 GHz |
| **memory** | 8 GB 1867 MHz DDR3 |
| **HDD size** | Intel Iris Graphics 6100 1536 MB |

Table 3.1 : hardware specifications

**software tools and packages/libraries**

**mac os**

we use mac operating system on our mac laptop since it's the only operating system that runs on this machine ,mac os is [52] an operating system that was created by apple company to run on their machines.

**Python**

Python is [53] an easy to learn, general-purpose, high-level, powerful programming language11. Which is widely used in the recent times, also using python different magazines and website are published. Many corporations have used and been using this tool for different function. Python offers efficient high-level data structures, is portable, open-source, and adopts a straightforward yet powerful approach to object-oriented programming. Its elegant syntax,dynamic typing, and interpreted nature collectively position it as a preferred language for scripting and rapid application development across various domains and platforms. we use python 3.10 v . Python can be integrated and supports a different technology that is why we use python, the following packages are installed on python3.

**Tensor Flow**

Tensor Flow was developed by the Google Brain team for internal Google use. It is [54] open-sourced framework for the implementation and deployment of large-scale machine learning models. Tensor Flow was introduced under the Apache License 2.0 on November 9, 2015. Renowned for its prowess in numerical computing, Tensor Flow plays a crucial role in the realm of deep learning. It offers APIs across a wide range of languages and platforms essential for deep learning projects, including Python, Android, Java, Windows, and Linux. TensorFlow stands as one of the most prominent libraries for handling Deep Neural Networks, owing to its seamless application development and deployment capabilities.

**Keras**

Keras, is [55] an open-source neural-network library implemented in Python, operates seamlessly on TensorFlow, Microsoft Cognitive Toolkit, or Theano. It serves as a high-level abstraction built atop TensorFlow or Theano, offering a Python-based API similar to scikit-learn for constructing neural networks. Developers leverage Keras to rapidly prototype neural networks, abstracting away complexities of tensor algebra, numerical techniques, and optimization methods. With its modular design, Keras fosters expressiveness, flexibility, and is well-suited for pioneering research endeavors. Being entirely Python-based, Keras facilitates ease of debugging and exploration. Unlike TensorFlow, Keras may not accommodate low-level model adjustments, necessitating TensorFlow for such tasks. Once familiar with its syntax, developers can swiftly construct models using Keras.

**Pickle**

The Python pickle module is [56] utilized for serializing and de-serializing Python object structures. It enables any Python object to be serialized for storage on disk. Pickling involves serializing the object into a character stream before writing it to file. This process converts Python objects like lists or dictionaries into a format that encapsulates all essential information for reconstructing the object in another Python script.

**Numpy**

NumPy serves as [57] the fundamental library for scientific computing in Python, encompassing a comprehensive array of tools and methodologies essential for solving mathematical models in Science and Engineering on computers. It is designed as a versatile package for array processing, featuring a high-performance multidimensional array object and a robust set of utilities tailored for manipulating these arrays. NumPy evolves from its predecessor, the Numeric array object, with the objective of establishing a robust foundation for an effective scientific computing environment.

**panda**

Pandas is [58] a versatile Python package renowned for its robust capabilities in handling labeled and time series data. Beyond offering essential statistical methods and enabling visualization through plotting, pandas excels in its capacity to seamlessly read from and write to various file formats, including Excel and CSV. Key functions like `read_csv()` empower efficient file manipulation, facilitating the storage and retrieval of data and labels from pandas objects as Series or Data Frame instances.

**Matplotlib**

Matplotlib stands [59] as a robust plotting library for Python, serving to generate static, animated, and interactive visualizations. Its core objective is to equip users with versatile tools and functionalities for visually representing data, thereby enhancing analysis and comprehension. While Matplotlib boasts an extensive codebase that may seem intimidating to newcomers, grasping its fundamental concepts and key principles can simplify its usage considerably. Matplotlib is popular due to its ease of use, extensive documentation, and wide range of plotting capabilities. It offers flexibility in customization, supports various plot types, and integrates well with other Python libraries like NumPy and Pandas.

Matplotlib is a suitable choice for various data visualization tasks, including exploratory data analysis, scientific plotting, and creating publication-quality plots. It excels in scenarios where users require fine-grained control over plot customization and need to create complex or specialized visualizations.

# Chapter 4

# 4. Experiment And Result

## 4.1 Introduction

In this chapter our model  is trained on Set of hyper parameters, We will use  Adam optimizer with multiple Activation functions, we will be testing SGDM with sigmoid , ReLU and, softmax , and so with momentum will use each of the three activation functions, sigmoid , ReLU and tanh, the results and discussions are present in our study and the experiments results are evaluated using Accuracy, precision, Recall, F1 Score and Confusion Matrix.

In this study, we use three machine learning algorithms namely Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) , to help us in water management decisions based on soil chemical properties. The MLP, which is a type of feed forward neural network, excels in capturing complex, non-linear relationships within our data, providing robust classification capabilities. LSTM and GRU, both which are specialized recurrent neural networks (RNNs), offer distinct advantages in handling temporal dependencies and sequential patterns which are  inherent in soil data, ensuring more accurate and reliable irrigation predictions.

The tests we performed was done on a dataset that included historical soil chemical data collected over a seven-year period in different parts of Adama. EIAT is the source of the data. Before providing the dataset to the prediction model , we carry out feature engineering and data preprocessing operations. The following hyper parameters were used in the model's implementation and assessment: the dropout layer, activation function, optimizer, loss function, and evaluation metrics. We examine how well state-of-the-art deep learning systems estimate water management using preprocessed soil chemical data. Ultimately, based on the original problem statement and research questions that we establish during proposal, the model's overall result as well as the experimentation result are assessed and analyzed.

## 4.2 Experimental Setup

### Dataset and Preprocessing

The dataset used in our study comprises comprehensive soil chemical properties collected from various farmlands by irrigation agricultural experts. It includes 17 distinct features related to soil composition and nutrient levels, which are crucial in determining irrigation decisions. To prepare the data for modeling, preprocessing steps involved normalization to standardize feature scales and handling missing values through median imputation techniques. And threasholding weighted sum techniques were further added to the data preprocessing step, Furthermore, categorical variables were encoded using appropriate methods to ensure compatibility with machine learning algorithms. The dataset was then split into training, validation, and test sets to facilitate robust model evaluation and performance validation.

**Experimental Environment:**

The experiments were conducted using a computational setup comprising a mach laptop equipped with a intel processor and over 128 memory capacity to handle large-scale data operations efficiently. The software environment leveraged Python as the primary programming language, supported by essential libraries such as TensorFlow and Keras for implementing and training deep learning models. These libraries provided a versatile framework for building and fine-tuning Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) models. The experiments were executed within Jupyter Notebooks, facilitating interactive development and seamless integration of data preprocessing, model training, and result analysis.

**Experimental Design**

our experimental design contains a systematic manipulation of various configurations and hyperparameters across the MLP, LSTM, and GRU architectures. For MLP models, different network depths, neuron counts per layer, and activation functions like  ReLU, Sigmoid and tanh were evaluated to optimize classification accuracy. LSTM and GRU models, known for their ability to capture temporal dependencies, were fine-tuned with varying numbers of units, and learning rates to enhance convergence speed. Each model configuration underwent rigorous evaluations using metrics such as accuracy, precision, recall, and F1 score on the validation set. Comparative analysis among the three algorithms also provided us insights into each models respective strengths and suitability.

## 4.3 Multi-Layer Perceptron (MLP)

In this section we test and train MLP neural network, using Adam and SGDM optimizers , with three activation functions each, by applying different epochs to train them on our local machine using Jupyter Notebook, using sklearn and tensor flow libraries. to train the model.

### 4.3.1 MLP Baseline Configuration

The baseline Multilayer Perceptron (MLP) model serves as the foundational framework for predicting irrigation decisions based on soil chemical properties in our study. Configured with a straightforward architecture, the MLP consists of multiple fully connected layers, each employing rectified linear unit (ReLU) activation functions to introduce non-linearity and improve model performance. The input layer constitutes about 17 soil chemical features, while subsequent hidden layers are optimized with varying numbers of neurons to capture complex relationships within the dataset. The output layer employs a sigmoid activation function, facilitating binary classification where a value close to 1 indicates a decision to irrigate and 0 suggests no irrigation. This baseline configuration where we set up our parameters to have single hidden layer , 20 epochs, 6 batch sizes and 16 neuron per hidden layers , the combination of the listed hyper parameter values listed in the table below are applied to our MLP model.

| **Hyper parameter** | **Value** |
|---|---|
| Number of Layers | 1 |
| Units per Layers | 16 |
| Activation Function | ReLU |
| Learning Rate | 0.001 |
| Epochs | 20 |
| Batch Size | 6 |
| Optimizer | Adam |

Table 4.1: MLP Baseline Configuration

The base line results are listed in the table below and we will use this evaluation to compare with the other results we will get in next analysis.

| Metric | Training Value | Validation Value |
|---|---|---|
| Accuracy | 0.969 | 0.954 |
| Loss | 0.080 | 0.139 |

Table 4.2: MLP Baseline Results

## 4.3.2 MLP Hyper parameter Tuning

In This Section We Will Try Rearranging The Hyper parameters Of our MLP model we are trying to perfect by trying to get the best accuracy results , in this section we will try to manipulate different values specifically for the number of neurons , number of hidden layers , epoch values and batch sizes , by initiating these values from the base line we mentioned above .

**Tuning number of neurons hyper parameter**

In This section we try to manipulate the value of number of neurons per layer where we input different variations including [16, 32 , 64 ,128] , and find which hyper parameter value is the one that results in better performance of the model , and we were abeld to determine the layers size , based on the our analysis , the best choice would be 16 neurons, as it has the lowest validation loss, indicating good generalization to unseen data. The slight increase in validation loss with 32 neurons suggests that it is also a reasonable choice if you prefer a bit more complexity, but 16 neurons is more optimal in this case.

| **No of neurons** | **training loss** | **validation loss** |
|---|---|---|
| **16** | **0.080** | **0.139** |
| 32 | 0.063 | 0.140 |
| 64 | 0.046 | 0.150 |
| 128 | 0.037 | 0.171 |

Table 4.3 : Performance result of tuning neuron size for MLP

**Tuning number of hidden layer hyper parameter**

In this section we try to manipulate the value of the number of of hidden layers in the hidden layer of our MLP model we are building ,by adding different variation like [1 , 2 , 3 , 4] , and find which hyperparameter value is the one that results in better performance of the model , and we were abeld to determine that , based on our analysis , 1 hidden layer is the best choice because it has the lowest validation loss, indicating the best generalization to unseen data. and adding more hidden layers will lead to over fitting , as evidenced by the increasing validation loss in the table below.

| **No of hidden layer** | **training loss** | **validation loss** |
|---|---|---|
| **1** | **0.070** | **0.155** |
| 2 | 0.052 | 0.197 |
| 3 | 0.036 | 0.193 |
| 4 | 0.036 | 0.280 |

Table 4.4 : Performance result of tuning number of hidden layers for MLP

**Tuning epoch hyper parameter**

In this section we try and perform an analysis of manipulating epoch parameter y applying multiple variations of epoch values [4 , 10 , 15 ,20,25,30] and find which epoch values is the values with the least validation loss as a viable choosing criteria

The optimal number of epochs based on this analysis appears to be 10 epochs, as it provides the lowest validation loss (0.140) while keeping the training loss reasonably low (0.097). This suggests a good balance between fitting the training data and generalizing to the validation data without over fitting , but difference between the validation losses for 10 and 20 epochs is very small, and the training loss is lower at 20 epochs. This suggests that the model is continuing to learn and improve with more epochs. Given this small difference in validation loss and the improvement in training loss, indeed it make sense to use 20 epochs as the viable option here.

| **No of epoch** | **training loss** | **validation loss** |
|---|---|---|
| 4 | 0.119 | 0.148 |
| 10 | 0.097 | 0.140 |
| 15 | 0.080 | 0.154 |
| **20** | **0.073** | **0.141** |
| 25 | 0.069 | 0.150 |
| 30 | 0.063 | 0.147 |
| 35 | 0.056 | 0.143 |
| 40 | 0.050 | 0.184 |

Table 4.5 : Performance result of tuning epoch for MLP

**Tuning batch size hyper parameter**

In this section we try and perform an analysis of manipulating batch size parameter y applying multiple variations of batch size values [6 , 16 , 32 , 64 , 128] and find which batch size values is the values with the least validation loss as a viable choosing criteria , and based on our analysis we were abeld to determine batch size 32 has the lowest validation loss (0.144), which indicates the best performance on unseen data. Although its training loss is slightly higher than batch size 6, the lower validation loss is more important for generalization to new data. batch size 32 is optimal because it offers the best balance between training and validation performance, indicating good generalization capability.

| **No of batch size** | **training loss** | **validation loss** |
|---|---|---|
| 6 | 0.074 | 0.147 |
| 16 | 0.096 | 0.152 |
| **32** | **0.105** | **0.144** |
| 64 | 0.127 | 0.159 |
| 128 | 0.145 | 0.170 |

Table 4.6: Performance result of tuning batch size for MLP

**Tuning activation function hyper parameter**

In this section we try and perform an analysis of manipulating activation function parameter by applying multiple variations of activation functions [relu , tanh , sigmoid] and find which activation function is the one on our MLP model with the least validation loss as a viable choosing criteria , and based on our analysis , ReLU has the lowest validation loss (0.135) and also the lowest training loss (0.073). This indicates that the model performs best with the ReLU activation function in terms of both training and validation performance. Using the ReLU activation function is optimal because it offers the best balance between training and validation performance.

| **Activation Function** | **training loss** | **validation loss** |
|---|---|---|
| **relu** | **0.073** | **0.135** |
| tanh | 0.083 | 0.139 |
| sigmoid | 0.102 | 0.142 |

Table 4.7: Performance result of tuning activation function for MLP

**Tuning learning rate hyper parameter**

In this section we try and perform an analysis of manipulating learning rate parameter by applying multiple variations of learning rate values [0.1 , 0.01 , 0.001 , 0.0001] and find which batch size values is the values with the least validation loss as a viable choosing criteria , and based on our analysis we were abeld to determine the learning rate of 0.001 shows the lowest validation loss (0.149), indicating better performance on unseen data compared to other learning rates tested. While a higher learning rate like 0.01 initially improves training loss significantly, it results in higher validation loss, suggesting over fitting. On the other hand, a very low learning rate like 0.0001 results in poorer training and validation performance overall. Therefore, 0.001 is optimal as it strikes a balance between effectively minimizing the loss during training and maintaining good generalization to validation data.

| **No of learning rate** | **training loss** | **validation loss** |
|---|---|---|
| 0.1 | 0.102 | 0.484 |
| 0.01 | 0.038 | 0.302 |
| **0.001** | **0.073** | **0.149** |
| 0.0001 | 0.137 | 0.156 |

Table 4.8: Performance result of tuning learning rate for MLP

### 4.3.3 MLP Analysis

In this section we will build the optimal model based on the configured values of our hyper parameters like number of neurons per layer , number of layers in a hidden layer , epoch values , learning rate , batch size and activation functions for our MLP model which are listed in the table below.

| model | neurons | layers | epochs | learning rate | batch size | activation function |
|---|---|---|---|---|---|---|
| MLP | 16 | 1 | 20 | 0.001 | 32 | relu |

Table 4.9: optimal hyper parameters for MLP

**Accuracy**

using the MLP algorithm with our hyper-parameters adjustment of 16 neurons , 20 epochs , with 80% of the data to train and 20% to test , learning rate of 0.001 , single hidden layer , and batch size 32 , and relu activation function we adjusted we got accuracy of (95.4%).

**Confusion matrix**

From our experiment we can see from the confusion matrix that out of the total data we gave it to predict 1.75% of the 0(not water) conditions are exactly predicted as 0(not water) , where 3.06% of 0(not water) conditions are incorrectly predicted as 1(water) condition ,also 1.53% of 1(water) condition are incorrectly predicted as 0(not water) condition where 93.65% of the 1(water) condition were correctly predicted as 1(water) condition.

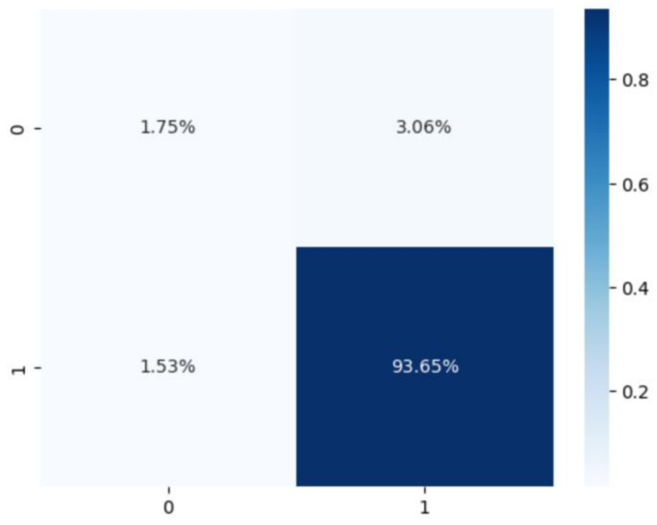
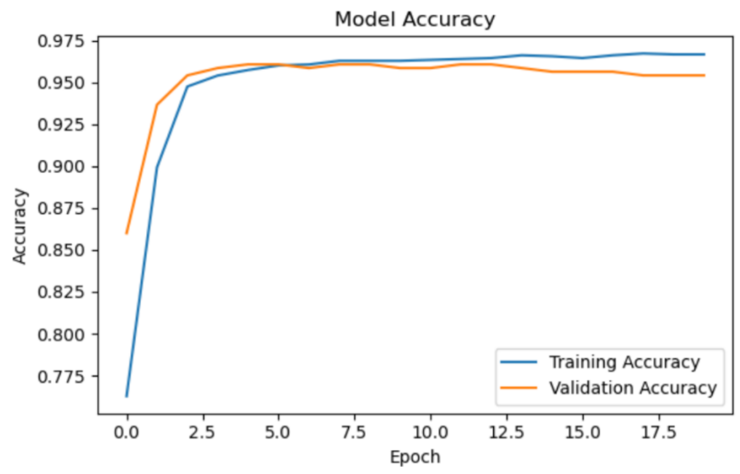


Figure 4.9 confusion matrix for MLP

**Train and test accuracy and loss**

the figure we have below shows the accuracy of the training and validation set of our model with a hyper parameter of 20 epochs , where our MLP based model with adam optimizer in relu activation function have training accuracy of 96.6% and validation accuracy of 95.4%.



.

Figure 4.10 Training and validation accuracy of attention using MLP

The figure below shows loss of our mode of adam optimizer with relu activation, describing loss of training 0.100 and validation loss 0.147.

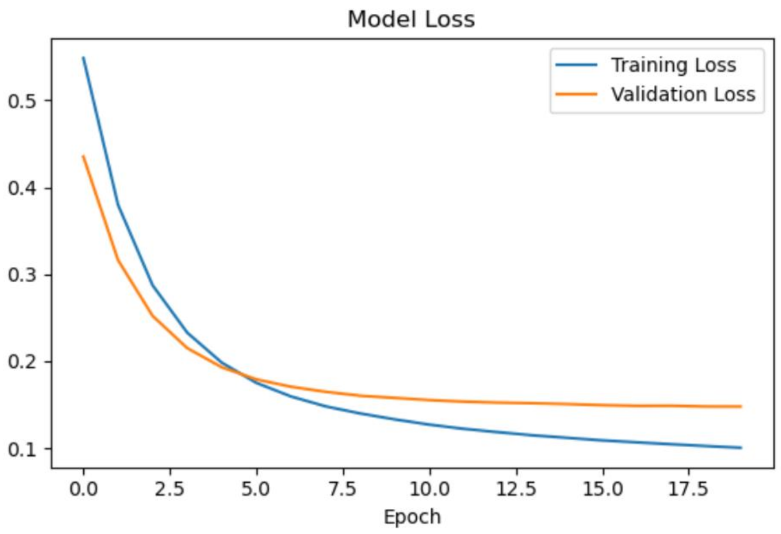


Figure 4.11 Training and validation loss of attention using MLP

.

## 4.4 Long Short-Term Memory (LSTM)

### 4.4.1 LSTM Baseline Configuration

The Long Short-Term Memory (LSTM) model is tailored to leverage the sequential nature of soil chemical data for precise irrigation decision-making. Comprising LSTM layers equipped with memory cells capable of retaining information over extended periods, this model excels in capturing temporal dependencies among the 17 soil chemical features. Each LSTM unit incorporates gates to regulate the flow of information, including input, forget, and output gates, thereby effectively managing long-term dependencies and sequential data like ours while mitigating vanishing gradient issues. The model architecture includes multiple LSTM layers followed by a dense output layer with a sigmoid activation function for binary classification.

Tuned with optimal dropout rates and learning rates, This baseline configuration where we set up our parameters to have single hidden layer , 20 epochs, 6 batch sizes and 50 neuron per hidden layers , the combination of the listed hyperparameter values listed in the table below are applied to our LSTM model.

| **Hyperparameter** | **Value** |
|---|---|
| Number of Layers | 1 |
| Units per Layers | 50 |
| Activation Function | tanh |
| Learning Rate | 0.001 |
| Epochs | 20 |
| Batch Size | 6 |
| Optimizer | Adam |

Table 4.10: LSTM Baseline Configuration

The base line results are listed in the table below and we will use this evaluation to compare with the other results we will get in next analysis.

| **Metric** | **Training Value** | **Validation Value** |
|---|---|---|
| Accuracy | 0.967 | 0.956 |
| Loss | 0.096 | 0.137 |

Table 4.11: LSTM Baseline Results

## 4.4.2 LSTM Hyperparameter Tuning

In This Section We Will Try Rearranging The Hyperparameteres Of our LSTM model we are trying to perfect by trying to get the best accuracy results , in this section we will try to manipulate different values specifically for the number of neurons , number of hidden layers , epoch values and batch sizes , by initiating these values from the base line we mentioned above .

**Tuning number of neurons hyper parameter**

In This section we try to manipulate the value of number of neurons per layer where we input different variations including [40,50,60,70,80,90,100] , and find which hyperparameter value is the one that results in better performance of the model , and we were abeld to determine the layers size , based on our analysis , neuron with size of 80 shows the lowest validation loss (0.125), indicating the best performance on unseen data. Although the training loss is slightly higher compared to some other configurations (e.g., 60 neurons), the validation loss is the most crucial metric for generalization performance.

Therefore, using 80 neurons is optimal as it offers the best balance between minimizing the training loss and maintaining a low validation loss, indicating good generalization capability.

| **No of neurons** | **training loss** | **validation loss** |
|---|---|---|
| 40 | 0.099 | 0.134 |
| 50 | 0.095 | 0.149 |
| 60 | 0.094 | 0.175 |
| 70 | 0.095 | 0.145 |
| **80** | **0.097** | **0.125** |
| 90 | 0.102 | 0.131 |
| 100 | 0.099 | 0.141 |

Table 4.12: Performance result of tuning neuron size for LSTM

**Tuning number of hidden layer hyper parameter**

In this section we try to manipulate the value of the number of of hidden layers in the hidden layer of our LSTM model we are building ,by adding different variation like [1 , 2 , 3 , 4] , and find which hyperparameter value is the one that results in better performance of the model , and we were abeld to determine that , based on our analysis , The configuration with 2 hidden layers shows the lowest validation loss (0.131), indicating the best performance on unseen data. Although the training loss is slightly higher compared to the configuration with 1 hidden layer (0.099 vs. 0.096), the validation loss is the most critical metric for generalization performance.

Therefore, using 2 hidden layers is optimal as it offers the best balance between minimizing the training loss and maintaining a low validation loss, indicating good generalization capability.

| **No of hidden layer** | **training loss** | **validation loss** |
|---|---|---|
| 1 | 0.096 | 0.132 |
| **2** | **0.099** | **0.131** |
| 3 | 0.100 | 0.138 |
| 4 | 0.101 | 0.140 |

Table 4.13: Performance result of tuning number of hidden layers for LSTM

**Tuning epoch hyper parameter**

In this section we try and perform an analysis of manipulating epoch parameter y applying multiple variations of epoch values [4 , 10 , 15 ,20,25,30] and find which epoch values is the values with the least validation loss as a viable choosing criteria

The optimal number of epochs based on this analysis The configuration with 30 epochs shows the lowest validation loss (0.127), indicating the best performance on unseen data. Although the training loss is slightly higher compared to configurations with more epochs, the validation loss is the most critical metric for generalization performance.

Therefore, using 30 epochs is optimal as it offers the best balance between minimizing the training loss and maintaining a low validation loss, indicating good generalization capability.

| **No of epoch** | **training loss** | **validation loss** |
|---|---|---|
| 4 | 0.120 | 0.144 |
| 10 | 0.108 | 0.140 |
| 15 | 0.099 | 0.151 |
| 20 | 0.098 | 0.129 |
| 25 | 0.093 | 0.133 |
| **30** | **0.091** | **0.127** |
| 35 | 0.086 | 0.138 |
| 40 | 0.085 | 0.135 |

Table 4.14: Performance result of tuning epoch for LSTM

**Tuning batch size hyper parameter**

In this section we try and perform an analysis of manipulating batch size parameter y applying multiple variations of batch size values [6 , 16 , 32 , 64 , 128] and find which batch size values is the values with the least validation loss as a viable choosing criteria , and based on our analysis we were abeld to determine The configuration with a batch size of 6 shows the lowest validation loss (0.137), indicating the best performance on unseen data. Although the training loss is not the lowest, the validation loss is the most critical metric for generalization performance.

Therefore, using a batch size of 6 is optimal as it offers the best balance between minimizing the training loss and maintaining a low validation loss, indicating good generalization capability.

| **No of batch size** | **training loss** | **validation loss** |
|---|---|---|
| **6** | **0.097** | **0.137** |
| 16 | 0.104 | 0.140 |
| 32 | 0.110 | 0.142 |
| 64 | 0.108 | 0.139 |
| 128 | 0.110 | 0.150 |

Table 4.15: Performance result of tuning batch size for LSTM

**Tuning activation function hyper parameter**

In this section we try and perform an analysis of manipulating activation function parameter by applying multiple variations of activation functions [relu , tanh , sigmoid] and find which activation function is the one  on our LSTM model with the least validation loss as a viable choosing criteria , and based on our analysis , The configuration with the ReLU activation function shows the lowest validation loss (0.136), which indicates the best performance on unseen data. Although the training loss is slightly higher than tanh, the validation loss is the most critical metric for generalization performance.

Therefore, using the ReLU activation function is optimal as it offers the best balance between minimizing the training loss and maintaining a low validation loss, indicating good generalization capability.

| **Activation Function** | **training loss** | **validation loss** |
|---|---|---|
| **relu** | **0.129** | **0.136** |
| tanh | 0.095 | 0.140 |
| sigmoid | 0.121 | 0.140 |

Table 4.16: Performance result of tuning activation function for LSTM

**Tuning learning rate hyper parameter**

In this section we try and perform an analysis of manipulating learning rate parameter by applying multiple variations of learning rate values [0.1 , 0.01 , 0.001 , 0.0001] and find which batch size values is the values with the least validation loss as a viable choosing criteria , and based on our analysis we were abeld to determine Learning Rate 0.0001 is the optimal value for this hyperparameter,  This learning rate provides the lowest validation loss (0.137), indicating the best generalization performance among the other values. Even though the training loss is not the lowest, the primary goal is to minimize validation loss to ensure good performance on unseen data.

Therefore, based on the results provided, the optimal learning rate for our LSTM model is 0.0001.

| **No of learning rate** | **training loss** | **validation loss** |
|---|---|---|
| 0.1 | 0.208 | 0.241 |
| 0.01 | 0.069 | 0.153 |
| 0.001 | 0.099 | 0.145 |
| **0.0001** | **0.117** | **0.137** |

Table 4.17: Performance result of tuning learning rate for LSTM

## 4.4.3 LSTM Analysis

In this section we will build the optimal model based on the configured values of our hyper parameters like number of neurons per layer , number of layers in a hidden layer , epoch values , learning rate , batch size and activation functions for our LSTM model which are listed in the table below.

| **model** | **neurons** | **layers** | **epochs** | **learning rate** | **batch size** | **activation function** |
|---|---|---|---|---|---|---|
| LSTM | 80 | 2 | 30 | 0.0001 | 6 | relu |

Table 4.18: optimal hyper parameters for LSTM

**Accuracy**

using the LSTM algorithm with our hyper-parameters adjustment of 80 neurons , 30 epochs , with 80% of the data to train and 20% to test , learning rate of 0.0001 , two hidden layer , and batch size 6 , and relu activation function we adjusted we got accuracy of (95.8%).

**Confusion matrix**

From our experiment we can see from the confusion matrix that out of the total data we gave it to predict 1.75% of the 0(not water) conditions are exactly predicted as 0(not water) , where 3.06% of 0(not water) conditions are incorrectly predicted as 1(water) conditon ,also 1.09% of 1(water) condition are incorrectly predicted as 0(not water) condition where 94.09% of the 1(water) condition were correctly predicted as 1(water) condition.

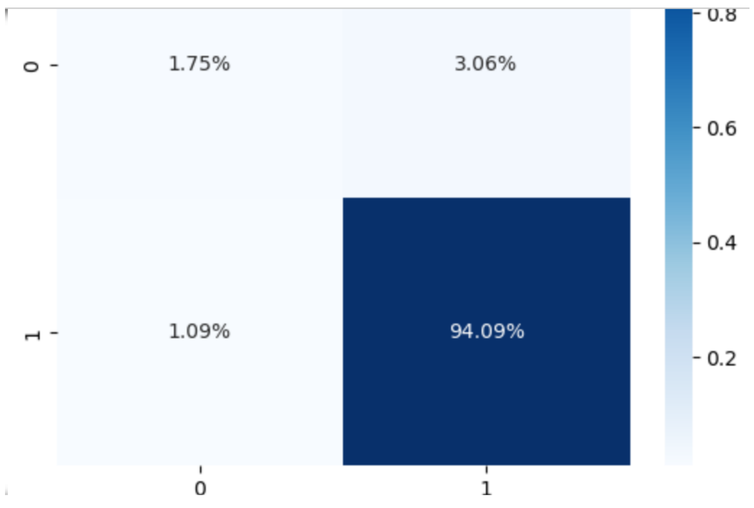
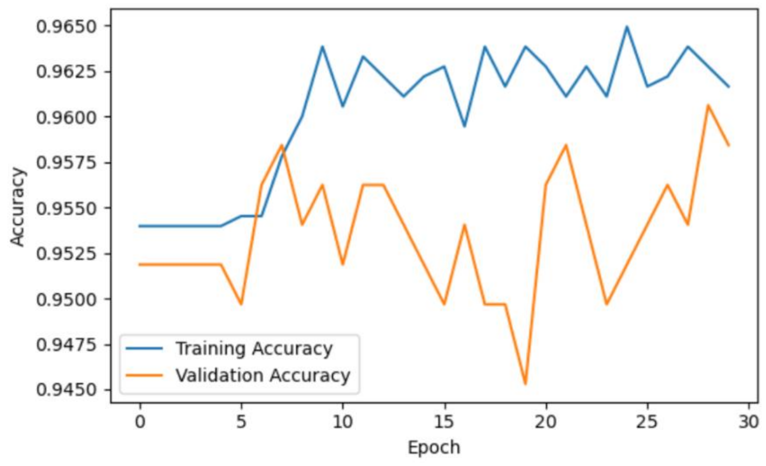


Figure 12.4: confusion matrix for LSTM

**Train and test accuracy and loss**

the figure we have below shows the accuracy of the training and validation set of our model with a hyper parameter of 30 epochs , where our LSTM based model with adam optimizer in relu activation function  have training accuracy of 96.1% and validation accuracy of 95.8%.



.

Figure 4.13: Training and validation accuracy of attention using LSTM

The figure below shows loss of our mode of adam optimizer with relu activation,Describing loss of training 0.095 and validation loss of 0.116.

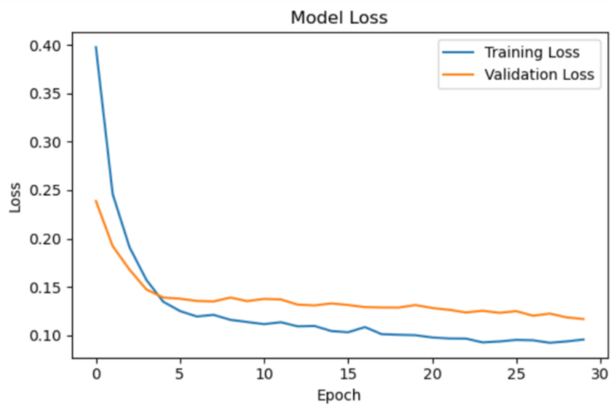


Figure 4.14 Training and validation loss of attention using LSTM

## 4.5 Gated Recurrent Unit (GRU)

### 4.5.1 GRU Baseline Configuration

The Gated Recurrent Unit (GRU) model represents a streamlined yet powerful alternative to LSTM for analyzing temporal sequences in soil chemical datasets. Featuring simplified gating mechanisms, the GRU architecture is designed to facilitate efficient training and inference while maintaining robust performance. Each GRU unit integrates reset and update gates to selectively update and reset its state, effectively capturing short-term dependencies without compromising computational efficiency. The model configuration includes multiple GRU layers stacked sequentially, culminating in a dense output layer with a sigmoid activation function for binary classification. Fine-tuned with appropriate dropout rates and learning rates, the GRU model aims to achieve comparable predictive accuracy to LSTM while offering advantages in training speed and simplicity, essential for scalable agricultural applications.

This baseline configuration where we set up our parameters to have single hidden layer , 20 epochs, 6 batch sizes and 50 neuron per hidden layers , the combination of the listed hyperparameter values listed in the table below are applied to our LSTM model.

| **Hyperparameter** | **Value** |
|---|---|
| Number of Layers | 1 |
| Units per Layers | 50 |
| Activation Function | tanh |
| Learning Rate | 0.001 |
| Epochs | 20 |
| Batch Size | 32 |
| Optimizer | Adam |

Table 4.19: GRU Baseline Configuration

The base line results are listed in the table below and we will use this evaluation to compare with the other results we will get in next analysis.

| **Metric** | **Training Value** | **Validation Value** |
|---|---|---|
| Accuracy | 0.965 | 0.956 |
| Loss | 0.109 | 0.141 |

Table 4.20: GRU Baseline Results

## 4.5.2 GRU Hyperparameter Tuning

In This Section We Will Try Rearranging The Hyperparameteres Of our GRU model we are trying to perfect by trying to get the best accuracy results , in this section we will try to manipulate different values specifically for the number of neurons , number of hidden layers , epoch values and batch sizes , by initiating these values from the base line we mentioned above .

**Tuning number of neurons hyper parameter**

In This section we try to manipulate the value of number of neurons per layer where we input different variations including [40,50,60,70,80,90,100] , and find which hyperparameter value is the one that results in better performance of the model , and we were abeld to determine the layers size , based on our analysis , 90 Neuronsis the optimal value for our hyperparameter variable number of neurons, This configuration provides the lowest validation loss (0.106), indicating the best generalization performance among the given options. The low training loss also suggests that the model fits well to the training data without overfitting.

Therefore, based on the results provided, the optimal number of neurons for the LSTM model is 90.

| **No of neurons** | **training loss** | **validation loss** |
|---|---|---|
| 40 | 0.109 | 0.144 |
| 50 | 0.105 | 0.151 |
| 60 | 0.107 | 0.143 |
| 70 | 0.109 | 0.138 |
| 80 | 0.119 | 0.140 |
| **90** | **0.106** | **0.106** |
| 100 | 0.105 | 0.140 |

Table 4.21: Performance result of tuning neuron size for GRU

**Tuning number of hidden layer hyper parameter**

In this section we try to manipulate the value of the number of of hidden layers in the hidden layer of our GRU model we are building ,by adding different variation like [1 , 2 , 3 , 4] , and find which hyperparameter value is the one that results in better performance of the model , and we were abeld to determine that , based on our analysis , The configuration with 1 hidden layer provides the lowest validation loss (0.143) among all configurations while maintaining a reasonable training loss (0.109), indicating better generalization performance, but given the minimal difference in validation loss and the slightly lower training loss,**4 hidden layers** can be used as the optimal value due to the lower training loss, which contribute to better performance during training and slightly better generalization.

So, based on your reasoning and the updated analysis, the optimal number of hidden layers for the GRU model is4 hidden layers.

| **No of hidden layer** | **training loss** | **validation loss** |
|---|---|---|
| 1 | 0.109 | 0.143 |
| 2 | 0.103 | 0.149 |
| 3 | 0.110 | 0.145 |
| **4** | **0.106** | **0.144** |

Table 4.22: Performance result of tuning number of hidden layers for GRU

**Tuning epoch hyper parameter**

In this section we try and perform an analysis of manipulating epoch parameter y applying multiple variations of epoch values [4 , 10 , 15 ,20,25,30,40s] and find which epoch values is the values with the least validation loss as a viable choosing criteria

Based on the performance results of tuning the number of epochs for the GRU model, the optimal number of epochs can be determined by analyzing both the training loss and validation loss. The results indicate that 20 epochs result in the lowest validation loss (0.136), although the training loss is higher (0.176) compared to other configurations. This suggests a model that generalizes well to new data despite having a higher training loss. On the other hand, 4 epochs and 10 epochs yield slightly higher validation losses (0.137 and 0.138, respectively) with lower training losses (0.128 and 0.114, respectively), indicating better training efficiency but a slight trade-off in validation performance. Considering both training and validation performance, 20 epochs is the most optimal configuration for the GRU model, as it provides the best generalization capability with the lowest validation loss, despite the increased training loss. This balance ensures that the model performs well on unseen data, which is crucial for practical applications in irrigation decision-making.

| **No of epoch** | **training loss** | **validation loss** |
|---|---|---|
| 4 | 0.128 | 0.137 |
| 10 | 0.114 | 0.138 |
| 15 | 0.111 | 0.145 |
| **20** | **0.176** | **0.136** |
| 25 | 0.110 | 0.151 |
| 30 | 0.118 | 0.138 |
| 35 | 0.113 | 0.140 |
| 40 | 0.101 | 0.144 |

Table 4.23: Performance result of tuning epoch for GRU

**Tuning batch size hyper parameter**

In this section we try and perform an analysis of manipulating batch size parameter y applying multiple variations of batch size values [6 , 16 , 32 , 64 , 128] and find which batch size values is the values with the least validation loss as a viable choosing criteria , and based on our analysis we were abeld to determine The configuration with a batch size of 32 provides the lowest validation loss (0.139), even though the training loss is higher (0.164) compared to other configurations. This suggests that the model generalizes well to new data with a batch size of 32, achieving the best validation performance.

While batch sizes of 6 and 16 show slightly lower training losses (0.094 and 0.100, respectively) and competitive validation losses (0.147 and 0.146, respectively), they do not outperform the batch size of 32 in terms of validation loss. Batch sizes of 64 and 128 have similar validation losses (0.142 each) but do not offer any significant improvement over batch size 32.

| **No of batch size** | **training loss** | **validation loss** |
|---|---|---|
| 6 | 0.094 | 0.147 |
| 16 | 0.100 | 0.146 |
| **32** | **0.164** | **0.139** |
| 64 | 0.111 | 0.142 |
| 128 | 0.117 | 0.142 |

Table 4.24: Performance result of tuning batch size for GRU

**Tuning activation function hyper parameter**

In this section we try and perform an analysis of manipulating activation function parameter by applying multiple variations of activation functions [relu , tanh , sigmoid] and find which activation function is the one  on our GRU model with the least validation loss as a viable choosing criteria , and based on our analysis , The ReLU activation function shows the lowest validation loss (0.126) and the lowest training loss (0.089) among the three activation functions tested. This suggests that the GRU model with the ReLU activation function not only fits the training data well but also generalizes better to the validation data compared to the tanh and sigmoid activation functions.

Therefore, ReLU is the optimal activation function for the GRU model based on the given performance results, providing the best balance between training and validation loss.

| **Activation Function** | **training loss** | **validation loss** |
|---|---|---|
| **relu** | **0.089** | **0.126** |
| tanh | 0.110 | 0.149 |
| sigmoid | 0.127 | 0.140 |

Table 4.25: Performance result of tuning activation function for GRU

**Tuning learning rate hyper parameter**

In this section we try and perform an analysis of manipulating learning rate parameter by applying multiple variations of learning rate values [0.1 , 0.01 , 0.001 , 0.0001] and find which batch size values is the values with the least validation loss as a viable choosing criteria , and based on our analysis we were abeld to determine The learning rate of 0.0001 shows the lowest

validation loss (0.137), indicating that it performs the best on the validation set. Although the training loss for this learning rate is higher than that for 0.01 (0.120 compared to 0.079), the validation performance is more important as it indicates better generalization to unseen data.
Therefore, 0.0001 is the optimal learning rate for the GRU model based on the given performance results, providing the best balance between training and validation loss and suggesting the best generalization performance.

| **No of learning rate** | **training loss** | **validation loss** |
|---|---|---|
| 0.1 | 0.140 | 0.148 |
| 0.01 | 0.079 | 0.151 |
| 0.001 | 0.108 | 0.156 |
| **0.0001** | **0.120** | **0.137** |

Table 4.26: Performance result of tuning learning rate for GRU

### 4.5.3 GRU Analysis

In this section we will build the optimal model based on the configured values of our hyper parameters like number of neurons per layer , number of layers in a hidden layer , epoch values , learning rate , batch size and activation functions for our GRU model which are listed in the table below.

| **model** | **neurons** | **layers** | **epochs** | **learning rate** | **batch size** | **activation function** |
|---|---|---|---|---|---|---|
| GRU | 90 | 4 | 20 | 0.0001 | 32 | relu |

Table 4.27: optimal hyper parameters for GRU

**Accuracy**

using the GRU algorithm with our hyper-parameters adjustment of 90 neurons , 20 epochs , with 80% of the data to train and 20% to test , learning rate of 0.0001 , four hidden layers , and batch size 32 , and relu activation function we adjusted we got accuracy of (94.3%).

**Confusion matrix**

From our experiment we can see from the confusion matrix that out of the total data we gave it to predict 1.97% of the 0(not water) conditions are exactly predicted as 0(not water) , where 2.84% of 0(not water) conditions are incorrectly predicted as 1(water) condition ,also 2.84% of 1(water) condition are incorrectly predicted as 0(not water) condition where 92.34% of the 1(water) condition were correctly predicted as 1(water) condition.

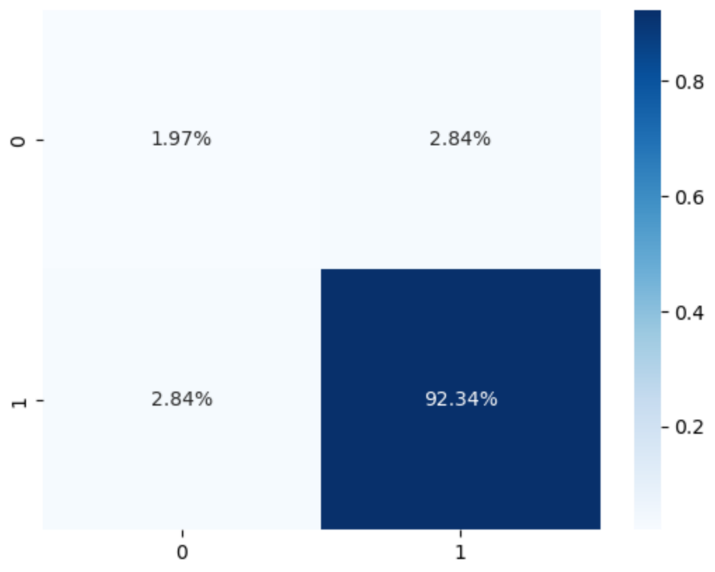
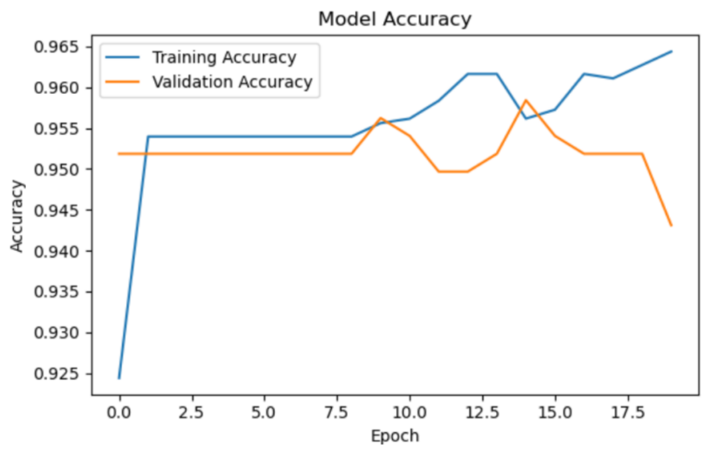


Figure 4.15 : confusion matrix for GRU

**Train and test accuracy and loss**

the figure we have below shows the accuracy of the training and validation set of our model with a hyper parameter of 20 epochs , where our GRU based model with adam optimizer in relu activation function  have training accuracy of 96.4% and validation accuracy of 94.3%.



.

Figure 4.16 :Training and validation accuracy of attention using GRU

the figure below shows loss of our mode of adam optimizer with relu activation ,describing loss of training 0.106 and validation loss of  0.141.

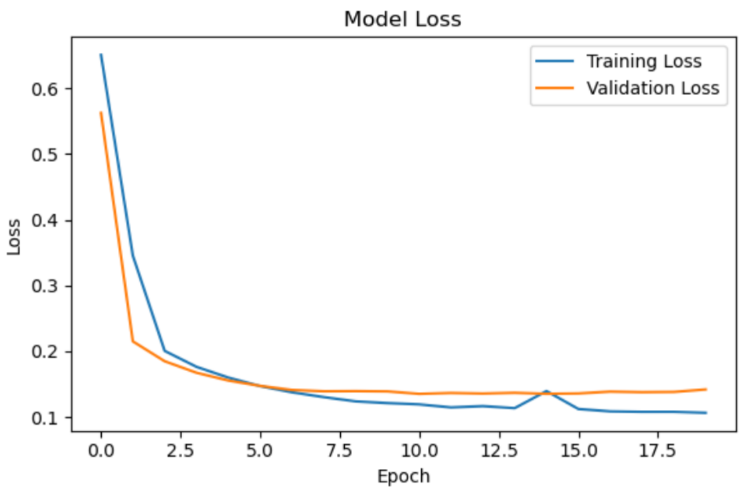


Figure 4.17 : Training and validation loss of attention using GRU

## 4.6 Comparative Analysis

In the sequence of experiments we conducted on three different neural network architectures namely the Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). With our MLP model with the optimal hyperparameters values we were abeld to achieve a training accuracy of 96.6% and a validation accuracy of 95.4%, with a training loss of 0.100 and a validation loss of 0.147. our LSTM model exhibited slightly lower training accuracy at 96.1% but outperformed in validation accuracy with 95.8%, demonstrating better generalization with a lower validation loss of 0.116 and a training loss of 0.095. Meanwhile, the GRU model achieved a training accuracy of 96.4% and matched the MLP in validation accuracy at 95.4%, with a training loss of 0.106 and a validation loss of 0.141. These results suggest that while all three models performed well, our LSTM model showed the best balance between training and validation performance, showing its superior capability in handling the time-dependent nature of the soil chemical data for our irrigation prediction requirement.

| **Model Type** | **Training Accuracy** | **Validation Accuracy** | **Training Loss** | **Validation Loss** |
|---|---|---|---|---|
| MLP | 96.6% | 95.4% | 0.100 | 0.147 |
| LSTM | 96.1% | 95.8% | 0.095 | 0.116 |
| GRU | 96.4% | 95.4% | 0.106 | 0.141 |

Table 4.28: comparative analysis for MLP , LSTM and GRU

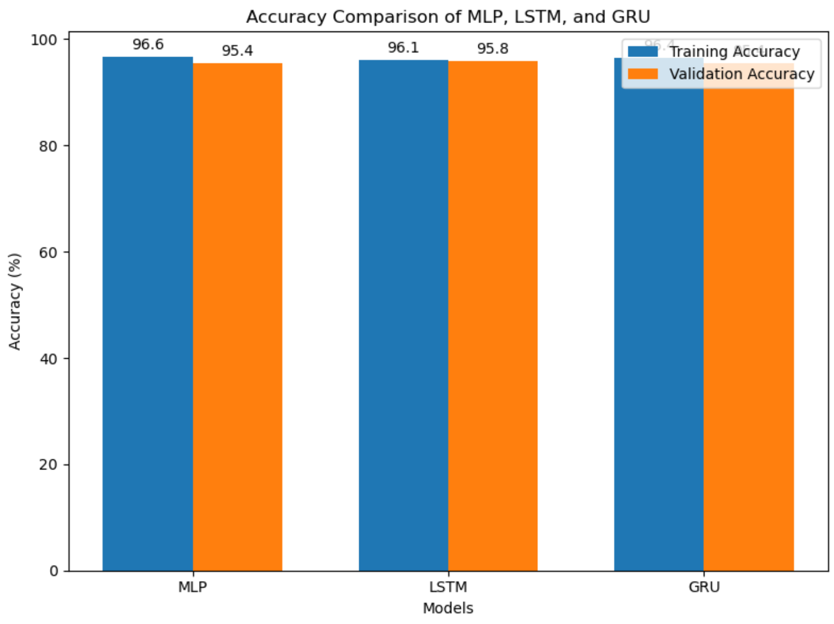


Figure 4.18 : Accuracy Comparison of MLP, LSTM, and GRU

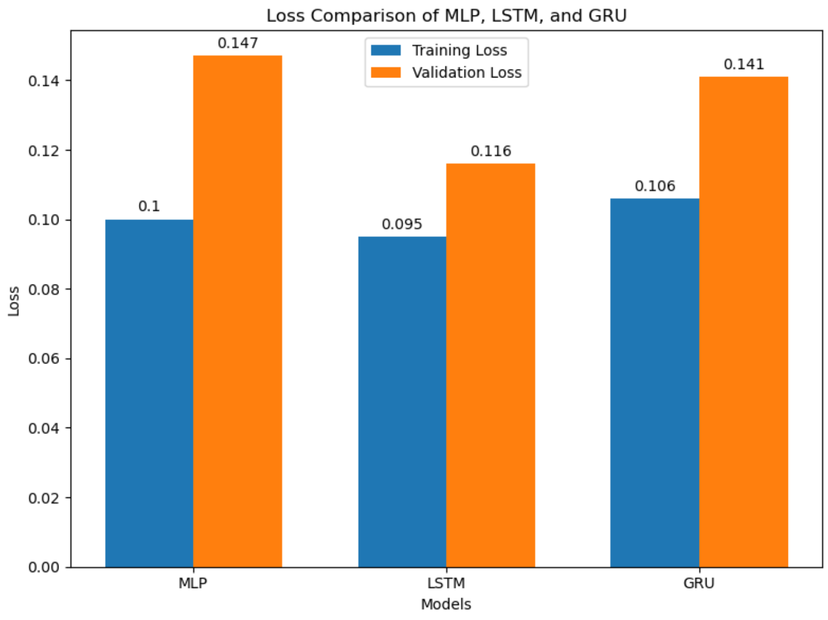


Figure 4.19 : Loss Comparison of MLP, LSTM, and GRU

# Chapter 5

# Conclusion

In my research to create a water management predictive model for irrigated farmlands I was abeld to collect and analyze set of soil chemical data which I gained access from governmental institutes , I tried to understand the relationship between these properties and how the water content in the soil are related and I was abled to specifically understand how each properties in the sample soil react to the total volume of water content and the numerical value range where each properties react when the water volume in the soil is normal range and I was abeld to use this to categorically describe this properties for the water management using preprocessing methods and feed this set of datas to a machine learning algorithms including Multi layer perceptron(MLP) , Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU)algorithms and I was abeld to predict based on these chemical properties of soil data inputs to get weather a soil needs to be irrigated or not with a maximum accuracy of 95.4 % (MLP) , 95.8% (LSTM) And 94.3% (GRU) respectively for each models , and in conclusion also there are previously done models with better prediction accuracy the research done did not use machine learning algorithm and have had more data in scale to train their model. And I was abeld to achieve my objective :

I was abeld to Analyze though articles that Deep learning based models are more effective in managing water resource based on the experience of other developing countries history of using AI based & IOT based Irrigation where they were abeld to conserve their resources without hampering their crop productivity., and we were abeld to find in our analysis and development of our model that predictive models based on machine learning algorithms were more effective compared to other human expert based conservation methods.

the models I build with different Hyperparameter combination based on the MLP, LSTM and GRU Algorithms I was abeld to achieve an accuracy 95.4 % (MLP) , 95.8% (LSTM) And 94.3% (GRU), compared to some of the papers we reviewed in terms of the local paper which lacks accuracy values but the level of water conserved we were not abeld to Compare and compare but in the other papers we reviewed which they were able to achieve an accuracy level of 91% (kNN) and 87% (SVM) and according to my analysis and comparison of these systems mine was the better in terms of accuracy level.

I was also abeld to predict future predictions using chemical data inputs with very effective predictive success.

In the future research I hope to do a better model building using much scaled up dataset to feed my model and use more complex algorithms with more complex parameter tunning adjustments that will fit the scale of my data I will procure by using IOT and cloud based technology, and test my trained model on field in real time by using field sensors and microcontrollers in combintion.

## Reference

1. N. K. Dawit, "Factors Affecting Use of Small-Scale Irrigation on Household Level in the Afar Regional State Amibera Woreda: In Case of Melka Werer Kebele," June 2023.
2. A. Balasubramanian, "Chemical Properties of Soils," Mar. 28, 2017.
3. G. Adugna, "A Review on Impact of Compost on Soil Properties, Water Use and Crop Productivity," Apr. 25, 2016.
4. T. A. Howell, "Irrigation Scheduling Research and Its Impact on Water Use," 1996.
5. W. T. Adugna, J. M. Hassen, F. R. Borena, N. A. Sori, and K. N. Tufa, "Response of Deficit Irrigation Levels and Methods on Yield and Water Productivity of Maize at Werer Agricultural Research Center, Afar, Ethiopia," 2018.
6. R. Ullah, A. W. Abbas, M. Ullah, R. U. Khan, I. U. Khan, N. Aslam, and S. S. Aljameel, "EEWMP: An IoT-Based Energy-Efficient Water Management Platform for Smart Irrigation," Apr. 2021.
7. T. Yu and H. Zhu, "Hyper-Parameter Optimization: A Review of Algorithms and Applications."
8. K. Chandra, A. Xie, J. Ragan-Kelley, and E. Meijer, "Gradient Descent: The Ultimate Optimizer," 2022.
9. P. F. Orrù, A. Zoccheddu, L. Sassu, C. Mattia, R. Cozza, and S. Arena, "Machine Learning Approach Using MLP and SVM Algorithms for the Fault Prediction of a Centrifugal Pump in the Oil and Gas Industry," Jun. 11, 2020.
10. N. Tessema, D. Yadeta, A. Kebede, and G. T. Ayele, "Soil and Irrigation Water Salinity, and Its Consequences for Agriculture in Ethiopia: A Systematic Review," 2023.
11. E. Svedberg, "Impact on Yield and Water Productivity of Wheat by Access to Irrigation Scheduling Technologies in Koga Irrigation Scheme, Ethiopia," 2019.
12. S. Z. Tefera and A. M. Beyene, "IoT and Machine Learning Based Smart and Intelligent Irrigation System."
13. M. E. Karar, M. F. Al-Rasheed, A. F. Al-Rasheed, and O. Reyad, "IoT and Neural Network-Based Water Pumping Control System For Smart Irrigation," May 8, 2020.
14. V. Ramachandran, R. Ramalakshmi, B. P. Kavin, I. Hussain, A. H. Almaliki, A. A. Almaliki, A. Y. Elnaggar, and E. E. Hussein, "Exploiting IoT and Its Enabled Technologies for Irrigation Needs in Agriculture," 2022.
15. S. Moghanian, F. B. Saravi, G. Javidi, and E. O. Sheybani, "GOAMLP: Network Intrusion Detection With Multilayer Perceptron and Grasshopper Optimization Algorithm," Nov. 26, 2020.

16. A. Odeh, A. Alarbi, I. Keshta, and E. Abdelfattah, "Efficient Prediction of Phishing Websites Using Multilayer Perceptron (MLP)," 2005.

17. R. Syah, S. Wulandari, A. Arbansyah, and A. Rezaeipanah, "Design of Ensemble Classifier Model Based on MLP Neural Network For Breast Cancer Diagnosis," 2021.

18. Y. Li, Z. Zhang, Z. Teng, and X. Liu, "PredAmyl-MLP: Prediction of Amyloid Proteins Using Multilayer Perceptron," Nov. 21, 2020.

19. A. K. Gupta, "Deep Learning: A Comprehensive Overview," , 2018.

20. V. Singh, "Normalization vs Standardization," Feb. 29, 2024.

21. T. M. E. Janssen, "Hyperparameter Tuning for Artificial Neural Networks Applied to Inverse Mapping Parameter Updating," 2022.

22. M. E. Karar, M. F. Al-Rasheed, A. F. Al-Rasheed, and O. Reyad, "IoT and Neural Network-Based Water Pumping Control System For Smart Irrigation," May 1, 2020.

23. S. Eriksson, "Water Quality in the Koga Irrigation Project, Ethiopia: A Snapshot of General Quality Parameters," Uppsala University, Department of Earth Sciences, 2013.

24. W. Seifu and E. Elias, "Soil Quality Attributes and Their Role in Sustainable Agriculture: A Review," Jan. 28, 2019.

25. T. Mulualem and B. Yebo, "Review on Integrated Soil Fertility Management for Better Crop Production in Ethiopia," Jan. 2, 2015.

26. M. Yami, "Irrigation Projects in Ethiopia: What Can Be Done to Enhance Effectiveness Under ‘Challenging Contexts’?," International Journal of Sustainable Development & World Ecology, 2016.

27. Z. A. Dejen, B. Schultz, and L. Hayde, "Water Delivery Performance at Metahara Large-Scale Irrigation Scheme, Ethiopia," Oct. 2014.

28. N. K. Nawandar and V. Satpute, "IoT Based Intelligent Irrigation Support System for Smart Farming Applications," Regular Issue, Vol. 8, No. 2, pp. 73-85, 2019.

29. R. Lal, "Soil Carbon Sequestration Impacts on Global Climate Change and Food Security," 2004.

30. M. E. Sumner and W. P. Miller, "Cation Exchange Capacity and Exchange Coefficients," Methods of Soil Analysis: Part 3—Chemical Methods, 1996.

31. J. D. Rhoades, A. Kandiah, and A. M. Mashali, "The Use of Saline Waters for Crop Production," FAO Irrigation and Drainage Paper 48, FAO, 1992.

32. E. V. Maas and S. R. Grattan, "Crop Yields as Affected by Salinity," Agricultural Drainage, vol. 55, pp. 55-108, 1999.

33. L. V. Kochian, M. A. Pineros, and O. A. Hoekenga, "The Physiology, Genetics, and Molecular Biology of Plant Aluminum Resistance and Toxicity," Plant and Soil, vol. 274, no. 1-2, pp. 175-195, 2005.

34. K. Kouno, T. Fujiwara, and Y. Yamamoto, "Essential Role of Cobalt in the Nitrogen Fixation Metabolism of Bradyrhizobium Japonicum," Soil Science and Plant Nutrition, vol. 48, no. 6, pp. 763-767, 2002.

35. K. Mengel and E. A. Kirkby, Principles of Plant Nutrition, 5th ed., Springer, 2001.

36. H. Marschner, Marschner's Mineral Nutrition of Higher Plants, 3rd ed., Academic Press, 2011.

37. U. C. Gupta and S. C. Gupta, Trace Elements in Soils and Plants, 4th ed., CRC Press, 2014.

38. A. E. Johnston, "Understanding Phosphorus and Its Use in Agriculture," European Fertilizer Manufacturers Association (EFMA), 2003.

39. W. L. Lindsay and A. P. Schwab, "The Chemistry of Iron in Soils and Its Availability to Plants," Journal of Plant Nutrition, vol. 5, no. 4-7, pp. 821-840, 1982.

40. B. J. Alloway, Zinc in Soils and Crop Nutrition, International Zinc Association, 2008.

41. U. C. Gupta, "Boron and Its Role in Crop Production," CRC Critical Reviews in Plant Sciences, vol. 12, no. 2, pp. 59-93, 1993.

42. B. J. Alloway, Heavy Metals in Soils: Trace Metals and Metalloids in Soils and Their Bioavailability, 3rd ed., Springer, 2013.

43. N. C. Brady and R. R. Weil, The Nature and Properties of Soils, 14th ed., Pearson, 2008.

44. U. C. Gupta, Molybdenum in Agriculture, Cambridge University Press, 1997.

45. J. F. Ma and N. Yamaji, "Silicon Uptake and Accumulation in Higher Plants," Trends in Plant Science, vol. 11, no. 8, pp. 392-397, 2006.

46. N. K. Fageria, Nitrogen Management in Crop Production, CRC Press, 2014.

47. J. L. Havlin, S. L. Tisdale, W. L. Nelson, and J. D. Beaton, Soil Fertility and Fertilizers: An Introduction to Nutrient Management, 8th ed., Pearson, 2013.

48. W. T. Pettigrew, "Potassium Influences on Yield and Quality Production for Maize, Wheat, Soybean, and Cotton," Physiologia Plantarum, vol. 133, no. 4, pp. 670-681, 2008. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/j.1399-3054.2008.01073.x.

49. E. J. Chen, Z. C. Li, H. B. Luo, and Y. Liu, "Attention-Based LSTM Predictive Model for the Attitude and Position of Shield Machine in Tunneling," Underground Space, vol. 13, pp. 335-350, 2023. doi: 10.1016/j.undsp.2023.05.006.

50. N. A. Abebe, "Water Consumption Analysis and Prediction Using Deep Learning Approach," Feb. 2024.

51. P. S. Muhuri, P. Chatterjee, X. Yuan, K. Roy, and A. Esterlin, "Using a Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) to Classify Network Attacks," Information, vol. 11, no. 243, 2020. doi: 10.3390/info11050243.

52. Apple Inc., "macOS - Apple," 2023. [Online]. Available: https://www.apple.com/macos/. [Accessed: 12-Jul-2024].

53. Python Software Foundation, "Welcome to Python.org," 2023. [Online]. Available: https://www.python.org/. [Accessed: 12-Jul-2024].

54. M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, Savannah, GA, USA, 2016, pp. 265-283.

55. F. Chollet, "Keras: The Python Deep Learning library," 2023. [Online]. Available: https://keras.io/. [Accessed: 12-Jul-2024].

56. Python Software Foundation, "pickle — Python object serialization," in *Python Documentation*, 2023. [Online]. Available: https://docs.python.org/3/library/pickle.html. [Accessed: 12-Jul-2024].

57. C. R. Harris, K. J. Millman, S. J. van der Walt et al., "Array programming with NumPy," *Nature*, vol. 585, pp. 357-362, 2020.

58. W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference (SciPy 2010)*, Austin, TX, USA, 2010, pp. 51-56.

59. J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, May-June 2007.

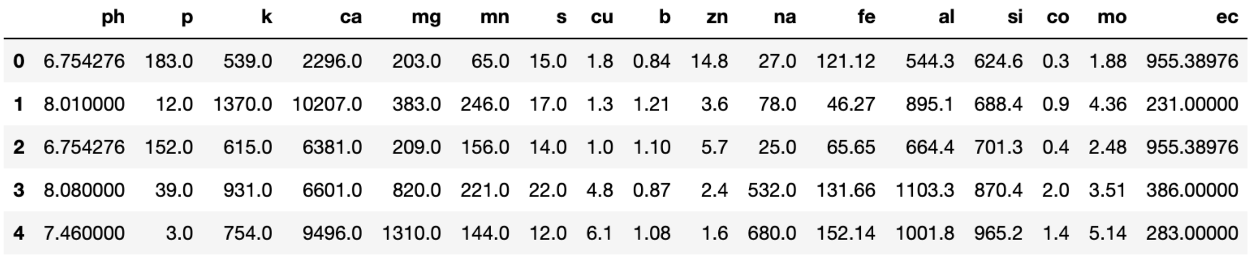
# Appendices

## Appendix A: Row dataset collected from EIAT

| Region | Zone | Woreda | sample_id | lab_sample_no | ph | p | hp |
|---|---|---|---|---|---|---|---|
| Oromia | East Shewa | Adama | 40703.A076 | NULL | NULL | 183 | NULL |
| Oromia | East Shewa | Adama | 40703.A024 | NULL | 8.01 | 12 | NULL |
| Oromia | East Shewa | Adama | 40703.A082 | NULL | NULL | 152 | NULL |
| Oromia | East Shewa | Adama | 40703.A029 | NULL | 8.08 | 39 | NULL |
| Oromia | East Shewa | Adama | 40703.A043 | NULL | 7.46 | 3 | NULL |
| Oromia | East Shewa | Adama | 40703.A095 | NULL | NULL | 61 | NULL |
| Oromia | East Shewa | Adama | 40703.A088 | NULL | NULL | 50 | NULL |
| Oromia | East Shewa | Adama | 40703.A083 | NULL | NULL | 7 | NULL |
| Oromia | East Shewa | Adama | 40703.A056 | NULL | NULL | 38 | NULL |
| Oromia | East Shewa | Adama | 40703.A103 | NULL | NULL | 20 | NULL |
| Oromia | East Shewa | Adama | 40703.A114 | NULL | 7.11 | 12 | NULL |
| Oromia | East Shewa | Adama | 40703.A116 | NULL | 7.37 | 30 | NULL |
| Oromia | East Shewa | Adama | 40703.A146 | NULL | 8.17 | 8 | NULL |
| Oromia | East Shewa | Adama | 40703.A097 | NULL | NULL | 52 | NULL |
| Oromia | East Shewa | Adama | 40703.A059 | NULL | NULL | 3 | NULL |
| Oromia | East Shewa | Adama | 40703.A038 | NULL | 8.09 | 36 | NULL |
| Oromia | East Shewa | Adama | 40703.A102 | NULL | NULL | 31 | NULL |
| Oromia | East Shewa | Adama | 40703.A045 | NULL | 7.48 | 18 | NULL |
| Oromia | East Shewa | Adama | 40703.A048 | NULL | 8.12 | 19 | NULL |
| Oromia | East Shewa | Adama | 40703.A012 | NULL | 7.86 | 14 | NULL |
| Oromia | East Shewa | Adama | 40703.A060 | NULL | NULL | 15 | NULL |
| Oromia | East Shewa | Adama | 40703.A144 | NULL | 7.97 | 13 | NULL |
| Oromia | East Shewa | Adama | 40703.A041 | NULL | 8.11 | 10 | NULL |
| Oromia | East Shewa | Adama | 40703.A143 | NULL | 6.84 | 18 | NULL |
| Oromia | East Shewa | Adama | 40703.A138 | NULL | 7.69 | 6 | NULL |
| Oromia | East | Adama | 40703.A16 | NULL | 7.8 | 31 | NULL |

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
|  | Shewa |  | 1 |  |  |  |  |
| Oromia | East Shewa | Adama | 40703.A127 | NULL | 7.56 | 25 | NULL |
| Oromia | East Shewa | Adama | 40703.A009 | NULL | 8.04 | 29 | NULL |
| Oromia | East Shewa | Adama | 40703.A137 | NULL | 7.08 | 13 | NULL |
| Oromia | East Shewa | Adama | 40703.A136 | NULL | 7.88 | 12 | NULL |
| Oromia | East Shewa | Adama | 40703.A134 | NULL | 7.64 | 7 | NULL |

## Appendix B: Processed dataset



|  | ph | p | k | ca | mg | mn | s | cu | b | zn | na | fe | al | si | co | mo | ec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 6.754276 | 183.0 | 539.0 | 2296.0 | 203.0 | 65.0 | 15.0 | 1.8 | 0.84 | 14.8 | 27.0 | 121.12 | 544.3 | 624.6 | 0.3 | 1.88 | 955.38976 |
| **1** | 8.010000 | 12.0 | 1370.0 | 10207.0 | 383.0 | 246.0 | 17.0 | 1.3 | 1.21 | 3.6 | 78.0 | 46.27 | 895.1 | 688.4 | 0.9 | 4.36 | 231.00000 |
| **2** | 6.754276 | 152.0 | 615.0 | 6381.0 | 209.0 | 156.0 | 14.0 | 1.0 | 1.10 | 5.7 | 25.0 | 65.65 | 664.4 | 701.3 | 0.4 | 2.48 | 955.38976 |
| **3** | 8.080000 | 39.0 | 931.0 | 6601.0 | 820.0 | 221.0 | 22.0 | 4.8 | 0.87 | 2.4 | 532.0 | 131.66 | 1103.3 | 870.4 | 2.0 | 3.51 | 386.00000 |
| **4** | 7.460000 | 3.0 | 754.0 | 9496.0 | 1310.0 | 144.0 | 12.0 | 6.1 | 1.08 | 1.6 | 680.0 | 152.14 | 1001.8 | 965.2 | 1.4 | 5.14 | 283.00000 |

## Appendix C: Training History

```
/Users/abel/opt/anaconda3/lib/python3.9/site-packages/keras/src/layers/rnn/rnn.py:204: UserWar
ning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential mod
els, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
Epoch 1/2058/58━━━━━━━━━━━━━━━━━━━━19s 94ms/step - accuracy: 0.8317 - loss:
0.6756 - val_accuracy: 0.9519 - val_loss: 0.5627
Epoch 2/2058/58━━━━━━━━━━━━━━━━━━━━4s 66ms/step - accuracy: 0.9532 - loss: 0.
4528 - val_accuracy: 0.9519 - val_loss: 0.2151
Epoch 3/2058/58━━━━━━━━━━━━━━━━━━━━4s 71ms/step - accuracy: 0.9622 - loss: 0.
1763 - val_accuracy: 0.9519 - val_loss: 0.1849
Epoch 4/2058/58━━━━━━━━━━━━━━━━━━━━5s 83ms/step - accuracy: 0.9480 - loss: 0.
1965 - val_accuracy: 0.9519 - val_loss: 0.1674
Epoch 5/2058/58━━━━━━━━━━━━━━━━━━━━4s 68ms/step - accuracy: 0.9543 - loss: 0.
1616 - val_accuracy: 0.9519 - val_loss: 0.1555
Epoch 6/2058/58━━━━━━━━━━━━━━━━━━━━5s 82ms/step - accuracy: 0.9630 - loss: 0.
1256 - val_accuracy: 0.9519 - val_loss: 0.1474
Epoch 7/2058/58━━━━━━━━━━━━━━━━━━━━4s 66ms/step - accuracy: 0.9582 - loss: 0.
1376 - val_accuracy: 0.9519 - val_loss: 0.1411
Epoch 8/2058/58━━━━━━━━━━━━━━━━━━━━4s 74ms/step - accuracy: 0.9595 - loss: 0.
1212 - val_accuracy: 0.9519 - val_loss: 0.1392
Epoch 9/2058/58━━━━━━━━━━━━━━━━━━━━4s 62ms/step - accuracy: 0.9488 - loss: 0.
1426 - val_accuracy: 0.9519 - val_loss: 0.1394
Epoch 10/2058/58━━━━━━━━━━━━━━━━━━━━5s 80ms/step - accuracy: 0.9534 - loss:
0.1262 - val_accuracy: 0.9562 - val_loss: 0.1390
Epoch 11/2058/58━━━━━━━━━━━━━━━━━━━━5s 79ms/step - accuracy: 0.9554 - loss:
0.1203 - val_accuracy: 0.9540 - val_loss: 0.1352
```

```
Epoch 12/2058/58━━━━━━━━━━━━━━━━━━━━7s 118ms/step - accuracy: 0.9539 - loss:
 0.1274 - val_accuracy: 0.9497 - val_loss: 0.1365
Epoch 13/2058/58━━━━━━━━━━━━━━━━━━━━7s 121ms/step - accuracy: 0.9618 - loss:
 0.1182 - val_accuracy: 0.9497 - val_loss: 0.1357
Epoch 14/2058/58━━━━━━━━━━━━━━━━━━━━5s 90ms/step - accuracy: 0.9648 - loss:
0.1020 - val_accuracy: 0.9519 - val_loss: 0.1368
Epoch 15/2058/58━━━━━━━━━━━━━━━━━━━━6s 95ms/step - accuracy: 0.9522 - loss:
0.1540 - val_accuracy: 0.9584 - val_loss: 0.1353
Epoch 16/2058/58━━━━━━━━━━━━━━━━━━━━6s 100ms/step - accuracy: 0.9558 - loss:
 0.1169 - val_accuracy: 0.9540 - val_loss: 0.1358
Epoch 17/2058/58━━━━━━━━━━━━━━━━━━━━6s 104ms/step - accuracy: 0.9647 - loss:
 0.0947 - val_accuracy: 0.9519 - val_loss: 0.1386
Epoch 18/2058/58━━━━━━━━━━━━━━━━━━━━9s 88ms/step - accuracy: 0.9646 - loss:
0.1147 - val_accuracy: 0.9519 - val_loss: 0.1378
Epoch 19/2058/58━━━━━━━━━━━━━━━━━━━━7s 122ms/step - accuracy: 0.9593 - loss:
 0.1108 - val_accuracy: 0.9519 - val_loss: 0.1382
Epoch 20/2058/58━━━━━━━━━━━━━━━━━━━━7s 112ms/step - accuracy: 0.9652 - loss:
 0.0986 - val_accuracy: 0.9431 - val_loss: 0.1419
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.
save_model(model)`. This file format is considered legacy. We recommend using instead the nat
ive Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_
model.keras')`.
Model saved as 'irrigation_prediction_gru_model.h5'15/15━━━━━━━━━━━━━━━━━━━━
━0s 20ms/step - accuracy: 0.9416 - loss: 0.1470
GRU Model Accuracy: 0.943115/15━━━━━━━━━━━━━━━━━━━━5s 203ms/step
Accuracy: 0.9431
Precision: 0.9701
Recall: 0.9701
F1 Score: 0.9701
Confusion Matrix:
[[  9  13]
 [ 13 422]]
```