**CHAPTER 1**

## INTRODUCTION

A farmer’s decision about which crop to grow is generally clouded by his intuition and other irrelevant factors like making instant profits, lack of awareness about market demand, overestimating a soil’s potential to support a particular crop, and so on. A very misguided decision on the part of the farmer could place a significant strain on his family’s financial condition. Perhaps this could be one of the many reasons contributing to the countless suicide cases of farmers that we hear from media on a daily basis. In a country like India, where agriculture and related sector contributes to approximately 20.4 per cent of its Gross Value Added (GVA) , such an erroneous judgment would have negative implications on not just the farmer’s family, but the entire economy of a region. For this reason, we have identified a farmer’s dilemma about which crop to grow during a particular season, as a very grave one. The need of the hour is to design a system that could provide predictive insights to the Indian farmers, thereby helping them make an informed decision about which crop to grow. With this in mind, we propose a system, an intelligent system that would consider environmental parameters (temperature, rainfall, geographical location in terms of state) and soil characteristics (N, P, K, pH value, soil type and nutrients concentration) before recommending the most suitable crop to the user. In addition to that a fertilizer suggestion is also made which is based on the optimum nutrients of the crops grown.

## PROJECT OVERVIEW

## The project aims to create a smart solution for managing irrigation and monitoring crop health. By combining sensor technology, automation, and IoT, the system minimizes water usage, improves crop yield, and reduces manual effort.

**Key Features**

1. **Smart Irrigation**:
   * Automated control of water supply based on soil moisture levels.
   * Integration of weather forecasts to avoid over-irrigation during rainy days.
2. **Crop Health Monitoring**:
   * Sensors for temperature, humidity, and light intensity.
   * Detection of anomalies in crop growth through camera-based monitoring.
   * Early alerts for pest infestations or potential diseases.
3. **Remote Management**:
   * Mobile or web-based platform for real-time monitoring and control.
   * Notifications and alerts for critical conditions.
4. **Energy-Efficient Design**:
   * Use of solar power for sustainability.
   * Low-energy components for long-term operation.

**System Components:**

1. **Sensors**:
   * Soil moisture sensors for irrigation needs.
   * Temperature and humidity sensors for environmental monitoring.
   * Light intensity sensors to monitor crop exposure.
2. **Microcontroller/Processor**:
   * Arduino, Raspberry Pi, or ESP32 for processing sensor data.
3. **Actuators**:
   * Solenoid valves or pumps to regulate water flow.
4. **Connectivity**:
   * Wi-Fi, GSM, or LoRa modules for IoT communication.
5. **Power Source**:
   * Solar panels or batteries for uninterrupted operation.

**Working Mechanism:**

1. **Data Collection**: Sensors gather real-time data from the field.
2. **Decision Making**: The system processes the data and determines irrigation needs.
3. **Action Execution**: Actuators manage water supply automatically.
4. **Feedback Loop**: Data is sent to a cloud-based dashboard for monitoring and analysis.

**Future Enhancements**

* AI-based analytics for predicting crop health and irrigation needs.
* Integration with drones for advanced field surveillance.
* Blockchain for data security and traceability.

Let me know if you need further customization or additional technical details.

**CHAPTER 2**

**LITERATURE SURVEY**

Low-cost IOT + ML design for smart farming with multiple application paper authors Fahad Kamraan Syed, Agniswar Paul, Ajay Kumar, Jaideep Cherukuri in paper [1] proposed system for water management systems and improve current irrigation methods. An IoT and ML- based farming system always keeps farmers aware of the upcoming weather possibilities and gives them the best suggestions about irrigation methods and crops thereby helping in better yield.

In paper [2] author’s proposed a smart system that can assist farmers in crop management by considering sensed parameters (temperature, humidity) and other parameters (soil type, location of farm, rainfall) that predicts the most suitable crop to grow in that environment.

Reference Paper [3] determines real time sampling of soil properties using MODIFIED SUPPORT VECTOR REGRESSION, a popular machine learning algorithm and four modules. The Modules include Sensor interfaced to IoT device, Agri cloud, Analyzing the real time sensor data and Agri user interface (AUI). The first module is portable IoT device (NodeMCU) with soil moisture sensor and pH sensor, environmental sensors. Agri cloud module consists of storage. Analyzing the real time data module is processing of types of crops and small plants suggested using modified support vector machine algorithm. Agri-user interface is a basic web interface. Thus, with the help of soil properties farmer will be able to get types of crops and small plants is grown in farmland with help of Modified support vector machine algorithm.

In paper [4] author’s proposed new technologies include the use of Internet of Things (IOT) and Machine Learning. The real time data from the field area can be collected using IOT system. The collected data from the field area is fed to the trained model. The trained model then makes the predictions using the data. The result produced by the model greatly helps is sowing the suitable crops in the particular field area In Reference paper [5] determines a model is proposed for predicting the soil type and suggest a suitable crop that can be cultivated in that soil. The model has been tested using various machine learning algorithms such as KNN, SVM and logistic regression. The accuracy of the present model is maximum than the existing models.

Aruul Mozhi Varman S proposed an IOT and deep learning based smart agriculture systems. This system monitors and collects the soil parameters from the field with the help of a wireless sensor network. The collected data is then uploaded in the cloud. Finally, the systems suggest best irrigation practices to the farmers by predicting the crop to be sown for next crop rotation. This information will be sent as an SMS to the farmers. The parameters include soil temperature, atmospheric temperature, and humidity [6]. This system suggests further improving the effectiveness by predicting the suitable time for applying pesticides, fertilizer, and manures.

In paper [7] proposed a system would assist the farmers in making an informed decision about which crop to grow depending on a variety of environmental and geographical factors. The ML and IoT based suggestions will significantly educate the farmer and help them minimize costs and make strategic decisions by replacing intuition and passed-down knowledge with far more reliable data-driven ML models. This allows for a scalable, reliable solution to an important problem affecting hundreds of millions of people

# R. Gutiérrez et al (2014)

proposed an IoT-based automated irrigation system, utilizing soil moisture sensors and GSM for remote monitoring. This study demonstrated a 40% reduction in water usage without compromising crop yields. Their work emphasized the importance of real-time monitoring and control for water conservation (*Sensors Journal*). Similarly, S. Patel and D. N. Doshi (2016) introduced a system that used IoT sensors to monitor environmental factors such as temperature, humidity, and soil pH. This system facilitated early detection of crop health issues, improving decision-making for pest and nutrient management

# A. Sharma et al (2018).

In this paper, integrated IoT with LoRaWAN technology to scale monitoring and irrigation systems for larger agricultural setups. Their system achieved significant cost savings and enhanced water efficiency in a real-world pilot study, addressing scalability challenges faced by traditional systems (*Journal of Precision Agriculture*). Further, in 2020, P. Kumar et al. advanced this concept by integrating weather forecasting data with soil sensors to create weather-aware irrigation systems. Their approach reduced water wastage by 30%, highlighting the benefits of combining environmental data with automated irrigation (*International Journal of Smart Agriculture Technologies*).

# M. Yadav et al. (2019).

In this paper, who designed a solar-powered irrigation system for rural areas. This renewable energy-driven system ensured uninterrupted operation in regions with unreliable electricity, significantly lowering operational costs (*Renewable Energy Journal*). In another study, V. Singh and P. Roy (2021) utilized IoT sensors and AI-based image processing for crop health monitoring. By deploying drones for aerial surveillance, they achieved early detection of diseases, improving crop yields by 15% in their trials (*IEEE Transactions on Agriculture Technology*).

# 4.T. Chen et al (2022).

In this paper, focused on greenhouse monitoring, automating irrigation and climate control using IoT and cloud analytics. Their system maintained optimal conditions for crop growth, leading to a marked increase in productivity (*International Journal of IoT in Agriculture*).Despite these advancements, challenges remain. High initial costs, limited adoption of AI for predictive analysis, and the lack of standardized communication protocols hinder widespread implementation. Future research should prioritize affordable solutions for small-scale farmers, advanced AI models for yield estimation, and universal IoT communication standards. focused on greenhouse monitoring, automating irrigation and climate control using IoT and cloud analytics. Their system maintained optimal conditions for crop growth, leading to a marked increase in productivity (*International Journal of IoT in Agriculture*).

* 1. **SUMMARY**

The Intelligent Irrigation and Crop Monitoring System integrates IoT, AI, and sensor technologies to improve agricultural productivity by optimizing water usage and monitoring crop health. Various studies have demonstrated significant advancements in this field.

IoT-based irrigation systems, such as the one developed by R. Gutiérrez et al. (2014), utilized soil moisture sensors and GSM technology for remote monitoring, reducing water consumption by 40% without affecting crop yields. Similarly, S. Patel and D. N. Doshi (2016) introduced a system that used sensors to monitor temperature, humidity, and soil pH, facilitating early detection of crop health issues and improving pest and nutrient management.

, A. Sharma et al. explored the scalability of IoT systems using LoRaWAN, achieving cost savings and water efficiency in large-scale farms. Meanwhile, P. Kumar et al. (2020) combined weather forecasting data with soil sensors to develop a weather-aware irrigation system, reducing water wastage by 30%. Energy efficiency was a focus of M. Yadav et al. who designed a solar-powered irrigation system for rural areas, offering a sustainable solution in regions with unreliable electricity.

In terms of crop health monitoring, V. Singh and P. Roy (2021) implemented AI-based image processing and drones for early disease detection, improving yields by 15%. T. Chen et al. advanced greenhouse monitoring by automating irrigation and climate control, leading to enhanced crop productivity.

Despite these advancements, challenges such as high costs, limited AI adoption, and the lack of standardized IoT communication protocols hinder widespread adoption. Addressing these gaps, future research should focus on developing affordable solutions, leveraging AI for predictive analysis, and creating standardized frameworks to make these systems accessible to both small-scale farmers and large agricultural enterprises.

In conclusion, intelligent irrigation and crop monitoring systems have the potential to revolutionize agriculture by conserving resources, enhancing productivity, and supporting sustainable farming practices. Continued innovation and scalability efforts will be critical for their broader implementation.

**CHAPTER 3**

**WORKING PRINCIPLE**

**3.1 INTRODUCTION**

The working principle of the Intelligent Irrigation and Crop Monitoring System is based on the integration of sensors, automation, and IoT technologies to optimize water usage and monitor crop health. Sensors installed in the field continuously collect real-time data on soil moisture, temperature, humidity, and light intensity, while weather forecast APIs provide additional insights into rainfall and environmental conditions. This data is processed by a microcontroller or processor, such as Arduino or Raspberry Pi, to make decisions based on pre-defined thresholds. For irrigation, actuators like solenoid valves or pumps are triggered to supply the required amount of water, ensuring precise and efficient distribution.

For crop monitoring, AI-based analysis can detect anomalies like pest infestations, diseases, or nutrient deficiencies. The processed data is sent to a cloud platform for advanced analysis and storage, allowing farmers to access real-time updates through a mobile app or web dashboard. The system also incorporates a feedback loop, continuously monitoring field conditions and refining its operations to achieve desired outcomes. Energy efficiency is enhanced by using solar power, making the system sustainable and operational in areas with limited electricity. Overall, this closed-loop system provides remote control, real-time monitoring, and timely alerts, ensuring sustainable and productive agricultural practices.

**THEORETICAL BACKGROUND**

### 3.2 Overview on Machine Learning

Machine learning is an application of artificial intelligence (AI) that gives systems the ability to automatically learn and evolve from experience without being specially programmed by the programmer. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The main aim of machine learning is to allow computers to learn automatically and adjust their actions to improve the accuracy and usefulness of the program, without any human intervention or assistance. Traditional writing of programs for a computer can be defined as automating the procedures to be performed on input data in order to create output artifacts. Almost always, they are linear, procedural and logical. A traditional program is written in a programming language to some specification, and

it has properties like:

* We know or can control the inputs to the program.
* We can specify how the program will achieve its goal.
* We can map out what decisions the program will make and under what conditions it makes them.
* Since we know the inputs as well as the expected outputs, we can be confident that the program will achieve its goal.

Traditional programming works on the premise that, as long as we can define what a **program needs to do, we are confident we can define how a program can achieve that** goal. This is not always the case as sometimes, however, there are problems that you can represent in a computer that you cannot write a traditional program to solve.

Such problems resist a procedural and logical solution. They have properties such as:

* The scope of all possible inputs is not known beforehand.
* You cannot specify how to achieve the goal of the program, only what that goal is.
* You cannot map out all the decisions the program will need to make to achieve its goal.
* You can collect only sample input data but not all possible input data for the program.
  + 1. **Supervised and Unsupervised Learning**

Machine learning techniques can be broadly categorized into the following types: Supervised learning takes a set of feature/label pairs, called the training set. From this training set the system creates a generalized model of the relationship between the set of descriptive features and the target features in the form of a program that contains a set of rules. The objective is to use the output program produced to predict the label for a previously unseen, unlabelled input set of features, i.e. to predict the outcome for some new data. Data with known labels, which have not been included in the training set, are classified by the generated model and the results are compared to the known labels. This dataset is called the test set. The accuracy of the predictive model can then be calculated as the proportion of the correct predictions the model labeled out of the total number of instances in the test set.

Unsupervised learning takes a dataset of descriptive features without labels as a training set. In unsupervised learning, the algorithms are left to themselves to discover interesting structures in the data. The goal now is to create a model that finds some hidden structure in the dataset, such as natural clusters or associations. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system does not figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data. Unsupervised learning can be used for clustering, which is used to discover any inherent grouping that are already present in the data. It can also be used for association problems, by creating rules based on the data and finding relationships or associations between them.

Semi-supervised machine learning falls somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring labeled data generally does not require additional resources.

Reinforcement machine learning algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Machine learning algorithms are tools to automatically make decisions from data in order to achieve some over- arching goal or requirement. The promise of machine learning is that it can solve

problems automatically, faster and more accurately than a manually specified solution, and at a larger scale. Over the past few decades, many machine learning algorithms have been developed by researchers, and new ones continue to emerge and old ones modified.

### Machine Learning Tools

There are many different software tools available to build machine learning models and to apply these models to new, unseen data. There are also a large number of well defined machine learning algorithms available. These tools typically contain libraries implementing some of the most popular machine learning algorithms. They can be categorized as follows :

* Pre-built application-based solutions.
* Programming languages which have specialized libraries for machine learning.

Using programming languages to develop and implement models is more flexible and gave us better control of the parameters to the algorithms. It also allows us to have a

better understanding of the output models produced. Some of the popular programming languages used in the field of machine learning are:

* **Python**: Python is an extremely popular choice in the field of machine learning and AI development. Its short and simple syntax make it extremely easy to learn
* **R**: R is one of the most effective and efficient languages for analyzing and manipulating data in statistics. Using R, we can easily produce well-designed publication-quality plot, including mathematical symbols and formulae where needed. Apart from being a general purpose language, R has numerous of packages like RODBC, G models, Class and Tm which are used in the field of machine learning. These packages make the implementation of machine learning algorithms easy, for cracking the business associated problems
* **Tensorflow:** TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications. TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide

variety of other domains, as well. TensorFlow provides stable Python and C++ APIs, as well as non-guaranteed backward compatible API for other languages.

### SciKit-learn

SciKit learn is an open source machine learning library built for python. Since its release in 2007, Scikit-learn has become one of the most popular open source machine learning libraries. Scikit-learn (also called sklearn) provides algorithms for many machine learning tasks including classification, regression, dimensionality reduction and clustering. The documentation for scikit-learn is comprehensive, popular and well maintained. Sklearn is built on mature Python Libraries such as NumPy, SciPy, and matplotlib. While languages such as R and MATLAB are extremely popular and useful for machine learning, we decided to choose Python along with its SciKit-learn libraries as our programming language of choice. The reasons for this are:

* We already have some familiarity and exposure to Python, and thus have a smaller learning curve.
* Both Python and Scikit-learn have excellent documentation and tutorials available online
* The number of classic machine learning algorithms that come with Scikit-learn, and the consistent patterns for using the different models i.e., each model can be used with the same basic commands for setting up the data, training the model and using the model for prediction. This makes it easier to try a range of machine learning algorithms on the same data.
* The machine learning algorithms included with sklearn have modifiable parameters known as hyper parameters that effect the performance of the model. These usually have sensible default values, so that we can run them without needing a detailed knowledge or understanding of their semantics.
* The IPython notebook, which is an interactive computational environment for Python, in which a user can combine code execution, rich text, mathematics and plots in a web page. This functionality allows us to provide the notebooks we used to run our experiments almost as an audit and in a presentable.

### Dataset

For the system, we are using various datasets all downloaded for government website and kaggle.

**Datasets include:-**

* Yield dataset
* Fertilizer dataset
* Soil nutrient content dataset
* Rainfall Temperature dataset A brief description of the datasets:
* **Yield Dataset:** This dataset contains yield for 16 major crops grown across all the states in kg per hectare. Yield of 0 indicates that the crop is not cultivated in the respective state.
* **Soil nutrient content dataset :** This dataset has columns with the attributes in the order-State, Nitrogen content, Phosphorous content, Potassium content and average ph
* **Rainfall Temperature dataset**: This dataset contains crops, max and min rainfall, max and min temperature, max and min rainfall and ph values.

### Data Preprocessing

This step includes replacing the null and 0 values for yield by -1 so that it does not effect the overall prediction. Further we had to encode the data-set so that it could be fed into the ML models.

**CHAPTER 4**

## SOFTWARE REQUIREMENT SPECIFICATION

### Introduction

This Software Requirements Specification provides a complete description of all the Functions and constraints of the “Precision agriculture using machine learning & IOT”. The document, describes the issue related to the agriculture system and what actions are needed to perform to grow which type of crop should be grown in your soil and your environment to get better crop production.

The basic idea of this project comes from the farmers who don’t know which type of crop will be better for their soil according to nutrition and other environment conditions. In India, agriculture is one of the most important professions. Many of the people do agriculture but are unable to determine which types of crops are more suitable to their soil. Means there are variety of crops which are only suitable for wet soil, some requires medium humidity in the soil to grow but this knowledge is less known to farmers as well as newbies who develop some interest in farming. As of now there are very less resources as well as software’s which will help them to improve quality. Such types of software are precision agriculture using machine learning.

### Project Scope

* + - * Improve farm management efficiency by adjusting field/crop treatments
      * Getting a better result for which type of crop will be growing on your agriculture field.
      * Getting more productivity from less efforts by using our application
      * Improve farm management efficiency by adjusting field/crop treatments
      * Optimise efforts and resources, reduce consumption and waste, and boost land productivity.
      * Which type of fertilizers should be used if any crop having any disease we can minimize using our app

This are some project scope of our project.

### User Classes and Characteristics

This protocol is implemented in python language .We also use flask, bootstrap frameworks. The system design has done using python flask web application.

This project is for end user who has to know about which type of crop should be better yield in their agriculture field just like farmer and newbies who don’t know how to do agriculture in their filed so our project provide all necessary thing to get better crop productivity in your field. Also we can minimize the effect of disease of your crop by identifying which type of disease it is.

### Assumptions and Dependencies

* + - * Data is an asset. It is valuable resource, as it has real and measurable value. Accurate and timely data is critical to quality and efficiency of service. Data input and its accuracy will depend on the user. Accountability of data will be defined
      * Content generation and updation will be done timely by the user
      * Portal Management Framework will be devised and user will play active role in it
      * User will provide content in local languages

### Functional Requirements

* System must be fast and efficient
* User friendly GUI
* Performance
* System Validation input
* Proper output

### System Feature 1(Functional Requirement)

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. Following are the functional requirements on the system:

1. All the data must be in the same format as a structured data.
2. The data collected will be vectorized and sent across to the classifier.

### External Interface Requirements

* + 1. **User Interfaces**
       - Front End Software: Flask Framework integrated with HTML, CSS, JS, BOOTSTRAP
       - Back End Software: Machine Learning (Python)

### Hardware Interfaces

* + - * RAM - Minimum 512 MB.
      * Processor - i3 or above and above with 2.5 GHz

### Software Interfaces

* + - * OS: Ubuntu, Windows, Mac
      * Tools: VScode or Python IDE and Jupyter Notebook.
      * Programming Language: Python flask, HTML, CSS, JS, BOOTSTRAP.
      * Dataset: A Dataset which is openly available in kaggle.
      * Libraries/Tools : Seaborn, Pandas, NumPy, SciKit-Learn, Pytorch, ResNet-9, SQLAlchemy, Pickle

### Communication Interfaces

The communication protocol started from http from UI interface in which person can see some details of the soil nutrition section by filling up details user can see which type of crop should be grown in your region and your agriculture field. And another one is person can just upload the picture of crop then user will get information about the diseases of crop.

### Nonfunctional Requirements

Nonfunctional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the

requirements arise through the user needs, because of budget constraints, organizational policies and the need for interoperability with other software and hardware systems.

### Performance Requirements

The System should be interactive and performance should be efficient. The delays of the system should be less. Also a stable internet connection is required.

### Safety Requirements

This Specification shall be sufficient detailed to allow the design and implement to achieve the required safety integrity and allow an assessment of functional safety

### Security Requirements

We provide login system by passwords for each level of access

### Software Quality Attributes

**Availability:**

The information regarding the nutrition’s must be available.

**Usability**:

System should be interactive and easy to understand.

**Maintainability**:

System should be easy to maintain.

**Robustness:**

Single application failure will not affect the system. System should be fault tolerant.

System should reliable enough to sustain in any condition. It should give Perfect results.

### System Requirements

* + 1. **Database Requirements**

MYSQL

### Software Requirements

* + - * Operating System: Windows, Linux and Mac
      * Coding Language: Python Flask, HTML,CSS,JS,BOOSTRAP
      * Tools: Seaborn, Pandas, NumPy, SciKit-Learn, Pytorch, ResNet-9, SQLAlchemy, Pickle

### Hardware Requirements

* + - * Processor: i3 and above with 2.5 GHz.
      * RAM: 512 MB.
      * Hard disk space: = 8GB

### Other Requirements:

The below hardware instruments are used for testing our model in real-time using IOT system which has following set of instruments.

* + - * Soil NPK Sensor
      * DS18B20 Temperature Sensor
      * Capacitive Soil Moisture Sensor
      * NRF24L01 Wireless Transceiver
      * Arduino Nano Board

**CHAPTER 5 METHODOLOGY**

* 1. **INTRODUCTION**

The proposed system aims to establish a comprehensive end-to-end farming solution that integrates an efficient automatic irrigation subsystem and a crop disease classifier. The system leverages IoT for data collection and control and employs a Convolutional Neural Network (CNN) to classify crop diseases.

In the Automatic Irrigation System, soil moisture sensors monitor the moisture levels in the soil. Based on these readings, the system automatically irrigates the crop according to its specific moisture requirements, ensuring optimal water use. Simultaneously, the system incorporates an automated mechanism to capture images of the crop at regular intervals. These images are synced to the Deep Learning Model, where a pre-trained CNN processes the input. The model analyzes the images to classify the leaf health into one of three categories: Healthy, Late Blight, or Early Blight.

The integration of these two subsystems—automatic irrigation and disease classification—creates a streamlined workflow. The irrigation system ensures the plant receives adequate water, while the disease classifier provides early detection and monitoring of crop health. This integration allows for real-time data flow, ensuring that the irrigation system and disease monitoring work in tandem to support efficient and sustainable farming practices. The use of IoT and deep learning enhances the system's accuracy, reliability, and scalability for modern agricultural applications.

The Intelligent Irrigation and Crop Monitoring System is designed to optimize agricultural practices by integrating IoT, sensor networks, and deep learning techniques. The system comprises two primary subsystems: an Automatic Irrigation System and a Crop Disease Classification Model, both interconnected to function as an end-to-end smart farming solution. The following methodology outlines the workflow, technologies, and operational principles of the system.

The Automatic Irrigation System uses IoT-based sensors to monitor soil moisture levels in real time. These sensors are strategically placed in the field to gather accurate data on soil conditions. The collected data is sent to a central microcontroller, such as Arduino or Raspberry Pi, which processes the inputs using predefined thresholds and decision-making algorithms. If the soil moisture falls below the desired level, the system activates water pumps via actuators, ensuring the precise amount of water is supplied to the crop. This automated irrigation process eliminates human intervention, minimizes water wastage, and ensures optimal hydration for the plants.

Simultaneously, the system includes a mechanism for capturing periodic images of the crops. These images are transferred to a cloud-based platform, where they are analyzed by the Deep Learning Model for disease detection. A Convolutional Neural Network (CNN) is employed for image classification, trained on a dataset containing images of healthy plants and plants affected by diseases such as Late Blight and Early Blight. The model processes the images and classifies the leaves into three categories: Healthy, Late Blight, or Early Blight. This classification is critical for early detection of diseases, enabling timely interventions to prevent further spread and potential crop loss.

The integration of the two subsystems is achieved through automated synchronization. The irrigation system is equipped with a camera that captures images during or after irrigation cycles. These images are sent as test data to the CNN-based model for analysis. The results from the model are relayed to a user-friendly dashboard or mobile application, allowing farmers to monitor crop health and irrigation status remotely. Notifications and alerts are generated for conditions requiring immediate attention, such as severe moisture deficits or disease detection.

To ensure energy efficiency and sustainability, the system is often powered by renewable energy sources like solar panels, making it suitable for deployment in rural areas with limited access to electricity. The entire system operates in a closed loop, where feedback from the disease classification and soil moisture monitoring continuously refines irrigation schedules and guides the farmer on necessary crop treatments.

This methodology ensures a seamless integration of water management and crop health monitoring, providing a holistic solution for precision agriculture. By leveraging IoT for real-time monitoring and deep learning for disease detection, the system enhances productivity, reduces resource wastage, and promotes sustainable farming practices.

automatic irrigation and disease classification—creates a streamlined workflow. The irrigation system ensures the plant receives adequate water, while the disease classifier provides early detection and monitoring of crop health. This integration allows for real-time data flow, ensuring that the irrigation system and disease monitoring work in tandem to support efficient and sustainable farming practices. The use of IoT and deep learning enhances the system's accuracy, reliability, and scalability for modern agricultural applications.

### 5.2Feasibility Study

Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements and evaluates the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

### 5.2.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. Since the project is Machine learning based, the cost spent in exectuating this project would not demand cost for softwares and related products, as most of the products are open source and free to use. Hence the project would consumed minimal cost and is economically feasible.

### Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Since machine learning algorithms is based on pure math there is very less requirement for any professional software. And also most of the tools are open source. The best part is that we can run this software in any system without any software requirements which makes them highly portable. Also most of the documentation and tutorials make easy to learn the technology

### Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The main purpose of this project which is based on crop prediction is to prevent the farmer from incurring losses and improve productivity. This also ensures that there is no scarcity of food as lack of production may lead to severe consequences. Thus, this is a noble cause for the sake of the society, a small step taken to achieve a secure future.

### Analysis

* + 1. **Performance Analysis**

Most of the software we use is open source and free. The models whichwe use in this software, learn only once ,i.e once they are trained theyneed not be again fed in for the training phase. One can directly predictfor values, hence time-complexity is very less. Therefore this model istemporally sound.

### Technical Analysis

As mentioned earlier, the tools used in building this software is opensource. Each tool contains simple methods and the required methodsare overridden to tackle the problem.

### Economical Analysis

The completion of this project can be considered free of cost in itsentirety. As the software used in building the model is free of cost andall the data sets used are being downloaded from kaggle and Govt. ofIndia website.

### CHAPTER 6

### SYSTEM DESIGN

* 1. **Analysis Models: SDLC Model to be applied**

The waterfall model is a sequential software development process, in which progress is seen as owing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.

**Requirement Analysis:**

This phase is concerned about collection of requirement of the system. This process involves generating document and requirement review.

**System Design:**

Keeping the requirements in mind the system specifications are translated in to a software representation. In this phase the designer emphasizes on:- algorithm, data structure, software architecture etc.

**Coding:**

In this phase programmer starts his coding in order to give a full sketch of product. In other words system specifications are only converted in to machine.

**Implementation:**

The implementation phase involves the actual coding or programming of the software.

The output of this phase is typically the library, executable, user manuals and additional software documentation.

**Testing:**

In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation

**Maintenance:**

The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer needs, adapt to accommodate changes in the external

environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

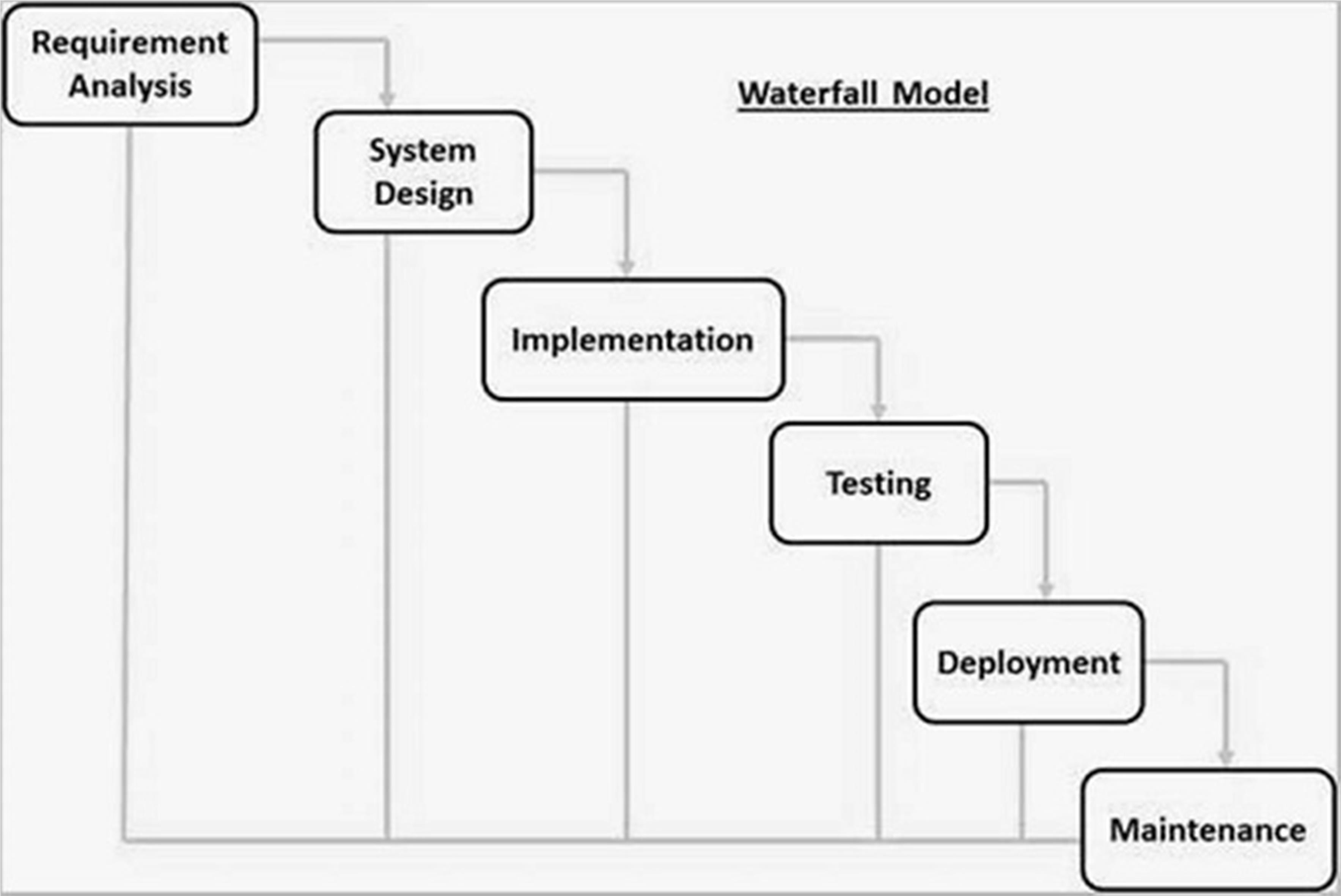


Fig-1 Waterfall Model

### Advantages of Waterfall model

* + - Clear project objective
    - Stable project requirements
    - Progress of system is measurable.
    - Logic of software development is clearly understood.
    - Better resource allocation.

### System Architecture

Fig-2 System Architecture

A system architecture is a conceptual model using which we can define the structure and behaviour of that system. It is a formal representation of a system. Depending on the context, system architecture can be used to refer to either a model to describe the system or a method used to build the system. Building a proper system architecture helps in analysis of the project, especially in the early stages. Figure 6.3 depicts the system architecture and is explined in the following section.

### Sequence Diagram

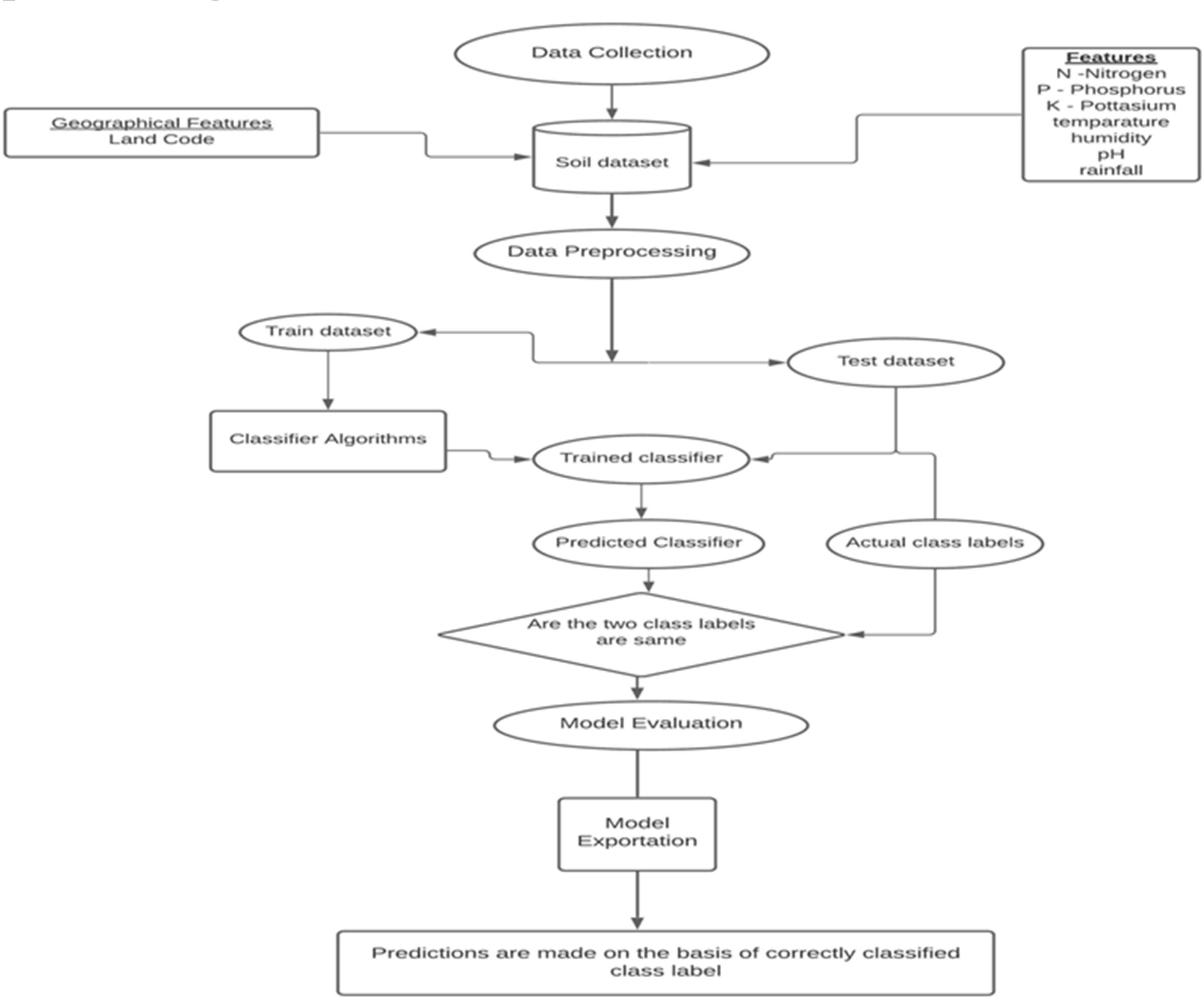


Fig-3 Sequence Diagram

### Data Flow Diagrams

* + 1. **DFD Level-0**



Fig-4 DFD Level-0

### DFD Level-1

Identification of data source

Data collections

Data processing

Training Models

Researching models

Predicted output

Fig-5 DFD Level-1

### UML Diagrams

Fig-6 UML diagram

**CHAPTER 7 HARDWARE DESCRIPTION**

## 7.1ARDUINO BOARD NANO

Fig. 7.1 ARDUINO BOARD NANO

his microcontroller is based on Atmega168 or Atmega328p. It is fairly similar to Arduino Uno board but when it comes to pin-configuration and features, this nano board has replaced [Arduino Uno](https://www.elprocus.com/atmega328-arduino-uno-board-working-and-its-applications/) due to small in size. As we know that while designing an [embedded system](https://www.elprocus.com/basics-of-embedded-system-and-applications/) small size components are preferred. Arduino boards are mainly used to build [electronic projects](https://www.elprocus.com/build-simple-electronic-projects-using-breadboard/). embedded systems, robotics, etc. But the nano boards are mainly introduced for the beginners who are not from the technical background.

Arduino Nano is one [type of microcontroller](https://www.elprocus.com/arm7-based-lpc2148-microcontroller-pin-configuration/) board, and it is designed by Arduino.cc. It can be built with a microcontroller like  Atmega328. This microcontroller is also used in [Arduino](https://www.elprocus.com/arduino-sim-for-iot-based-devices-launched-by-arduino/) UNO. It is a small size board and also flexible with a wide variety of applications. Other [Arduino boards](https://www.elprocus.com/different-types-of-arduino-boards/) mainly include Arduino Mega, Arduino Pro Mini, Arduino UNO, Arduino YUN, Arduino Lilypad, Arduino Leonardo, and Arduino Due. And other development boards are AVR Development Board, PIC Development Board, [Raspberry Pi](https://www.elprocus.com/new-raspberry-pi-3-model-a-with-wi-fi-and-bluetooth/), Intel Edison, MSP430 Launchpad, and ESP32 board.

This board has many functions and features like an Arduino Duemilanove board. However, this Nano board is different in packaging. It doesn’t have any DC jack so that the power supply can be given using a small USB port otherwise straightly connected to the pins like VCC & GND. This board can be supplied with 6 to 20volts using a mini USB port on the board.

**Arduino Nano Features**

The features of an Arduino nano mainly include the following.

* ATmega328P Microcontroller is from 8-bit AVR family
* Operating voltage is 5V
* Input voltage (Vin) is 7V to 12V
* Input/Output Pins are 22
* Analog i/p pins are 6 from A0 to A5
* Digital pins are 14
* Power consumption is 19 mA
* I/O pins DC Current is 40 mA
* Flash memory is 32 KB
* SRAM is 2 KB
* EEPROM is 1 KB
* CLK speed is 16 MHz
* Weight-7g
* Size of the printed circuit board is 18 X 45mm
* Supports three communications like SPI, IIC, & USART

### Arduino Nano Communication

The communication of an Arduino Nano board can be done using different sources like using an additional Arduino board, a computer, otherwise using microcontrollers. The microcontroller using in Nano board (ATmega328) offers [serial communication](https://www.elprocus.com/avr-microcontroller-serial-data-communication/) (UART TTL). This can be accessible at digital pins like TX, and RX. The Arduino software comprises of a serial monitor to allow easy textual information to transmit and receive from the board.

The TX & RX LEDs on the Nano board will blink whenever information is being sent out through the FTDI & USB link in the direction of the computer. The library-like SoftwareSerial allows serial communication on any of the digital pins on the board. The microcontroller also supports SPI & I2C (TWI) communication.

**7.2 SOIL NPK SENSOR**

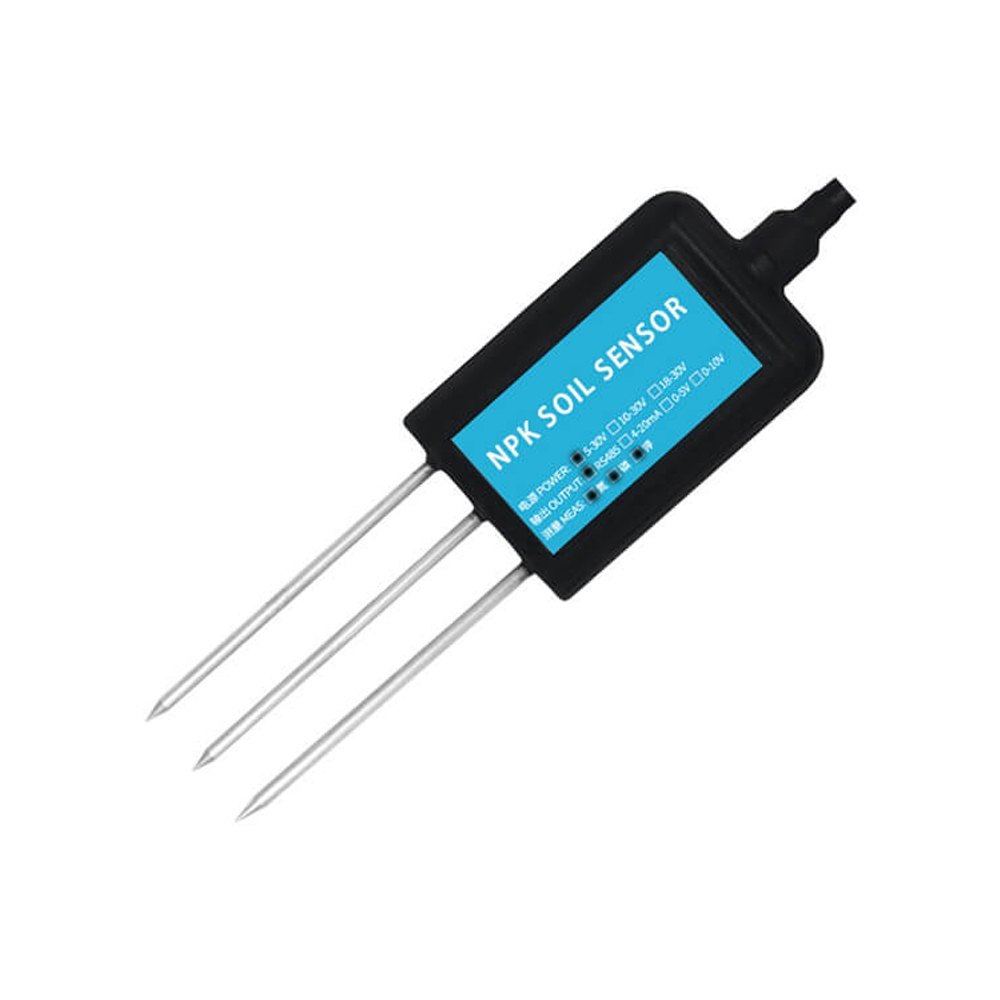


Fig. 7.2 SOIL NPK SENSOR

The soil npk sensor is suitable for detecting the content of nitrogen, phosphorus, and potassium in the soil, and judging the fertility of the soil. thereby facilitating the systematic evaluation of the soil condition. Can be buried in the soil for a long time, resistant to long-term electrolysis, corrosion resistance, vacuum potting, and completely waterproof. Soil npk sensors are widely used in soil nitrogen, phosphorus and potassium detection, precision agriculture, forestry, soil research, geological prospecting.

A Soil NPK Sensor is a specialized device used to measure the concentration of Nitrogen (N), Phosphorus (P), and Potassium (K) in soil, which are critical nutrients for plant growth. The sensor helps in determining the soil's fertility levels, enabling farmers to make informed decisions regarding fertilizer

**Components and Working Principles**

1. **Sensor Probes**:
   * These are inserted into the soil to measure electrical conductivity (EC) and ion concentration, which correlate with NPK levels.
   * The probes detect nutrient levels through chemical or electrochemical reactions.
2. **Microcontroller**:
   * Processes data from the probes and converts the raw signals into readable NPK concentration values.
   * May include calibration algorithms to ensure accuracy across various soil types.
3. **Display or Output Interface**:
   * Displays real-time nutrient levels on an LCD or sends data to a mobile app or dashboard through IoT integration.

**How It Works**

1. **Insertion**:
   * The sensor is inserted into the soil where the measurement is to be taken.
2. **Data Collection**:
   * Electrochemical or optical methods detect ion concentrations corresponding to N, P, and K.
3. **Signal Processing**:
   * The microcontroller processes the raw data into actionable values, often in milligrams per kilogram (mg/kg).
4. **Output**:
   * The data is displayed on the device or sent to an IoT platform for analysis.
5. **Actionable Insights**:
   * Farmers can use the readings to adjust fertilizer application, ensuring crops receive adequate nutrients while avoiding over-fertilization.

## Soil npk sensor parameters

* + **Power supply**: 5-30VDC
  + **Maximum power consumption**: ≤0.15W
  + **Operating temperature**: -40~80℃
  + **Range:** 0-1999 mg/kg(mg/L)
  + **Resolution**: 1 mg/kg(mg/L)
  + **Precision**: ±2%FS
  + **Response time**: ≤1S
  + **Protection grade**: IP68
  + **Probe material**: 316 stainless steel
  + **Sealing material**: Black flame-retardant epoxy resin
  + **Default cable length**: 2 meters, cable length can be customized
  + **Dimensions**: 45\*15\*123mm
  + **Output signal**: RS485/4-20ma/0-5v/0-10v

**7.3 DS18B20 TEMPERATURE SENSOR**



Fig. 7.3 DS18B20 TEMPERATURE SENSOR

The digital [temperature sensor](https://www.elprocus.com/temperature-sensors-types-working-operation/) like DS18B20 follows single wire protocol and it can be used to measure temperature in the range of -67oF to +257oF or -55oC to +125oC with +-5% accuracy. The range of received data from the 1-wire can range from 9-bit to 12-bit. Because, this sensor follows the single wire protocol, and the controlling of this can be done through an only pin of Microcontroller. This is an advanced level protocol, where each sensor can be set with a 64-bit serial code which aids to control numerous sensors using a single pin of the microcontroller. This article discusses an overview of a DS18B20 temperature sensor.

The DS18B20 is one type of temperature sensor and it supplies 9-bit to 12-bit readings of temperature. These values show the temperature of a particular device. The communication of this sensor can be done through a one-wire [bus protocol](https://www.elprocus.com/i2c-bus-protocol-tutorial-interface-applications/) which uses one data line to communicate with an inner [microprocessor](https://www.elprocus.com/8085-microprocessor-architecture/). Additionally, this sensor gets [the power supply](https://www.elprocus.com/regulated-power-supply-circuit-working-applications/) directly from the data line so that the need for an external power supply can be eliminated. The applications of the DS18B20 temperature sensor include industrial systems, consumer products, systems which are sensitive thermally, thermostatic controls, and thermometers.

**DS18B20 Pin Configuration**

* Pin1 (Ground): This pin is used to connect to the GND terminal of the circuit
* Pin2 (Vcc): This pin is used to give the power to the sensor which ranges from 3.3V or 5V
* Pin3 (Data): The data pin supplies the temperature value, which can communicate with the help of 1-wire method.

**Specifications**

The specifications of this sensor include the following.

* This sensor is a programmable and digital temperature sensor
* The [communication](https://www.elprocus.com/what-is-a-communication-system-and-its-basic-elements/) of this sensor can be done with the help of a 1-Wire method
* The range of power supply is 3.0V – 5.5V
* Fahrenheit equal s to -67°F to +257°F
* The accuracy of this sensor is ±0.5°C
* The o/p resolution will range from 9-bit to 12-bit
* It changes the 12-bit temperature to digital word within 750 ms time
* This sensor can be power-driven from the data line
* Alarm options are programmable
* The multiplexing can be enabled by Unique 64-bit address
* The temperature can be calculated from -55°C to +125°C.
* These are obtainable like SOP, To-92, and also as a waterproof sensor

**Working Principle**

The working principle of this DS18B20 temperature sensor is like a temperature sensor. The resolution of this sensor ranges from 9-bits to 12-bits. But the default resolution which is used to power-up is 12-bit. This sensor gets power within a low-power inactive condition. The temperature measurement, as well as [the conversion of A-to-D](https://www.elprocus.com/analog-to-digital-converter/), can be done with a convert-T command. The resulting temperature information can be stored within the 2-byte register in the sensor, and after that, this sensor returns to its inactive state.

If the sensor is power-driven by an exterior power supply, then the master can provide read time slots next to the Convert T command. The sensor will react by supplying 0 though the temperature change is in the improvement and reacts by supplying 1 though the temperature change is done.

**DS18B20 Temperature Sensor Applications**

The applications of DS18B20 include the following.

* This sensor is extensively used to calculate temperature within rigid environments which includes mines, chemical solutions, otherwise soil, etc.
* This sensor is used to measure the liquid temperature.
* We can use it in the thermostat controls system.
* It can be used in industries as a temperature measuring device.
* This sensor is used as a thermometer.
* It can be used in devices like which are sensitive to thermal.
* These are used in HVAC systems.
* Applications where the temperature has to be measured at multiple points.

**7.4 CAPACITIVE SOIL MOISTURE SENSOR**



Fig. 7.4 CAPACITIVE SOIL MOISTURE SENSOR

**Capacitive Soil Moisture Sensor** is an innovative and widely used device designed to measure the moisture content in soil. Unlike traditional resistive sensors, capacitive sensors rely on the dielectric constant of the soil to determine its moisture levels, offering higher durability and accuracy. Soil moisture is a crucial parameter in agriculture, landscaping, and environmental monitoring, making capacitive sensors a vital component in modern precision farming and irrigation systems. This comprehensive overview discusses the working principle, components, advantages, applications, and limitations of capacitive soil moisture sensors, highlighting their growing significance in sustainable agricultural practices.

**1. Working Principle of Capacitive Soil Moisture Sensors**

The capacitive soil moisture sensor operates on the principle of capacitance, which is the ability of a material to store electrical charge. The sensor typically consists of a pair of conductive plates or electrodes that form a capacitor. When the sensor is inserted into the soil, the dielectric constant of the soil acts as the medium between these plates. The capacitance of the system changes with the soil’s moisture content because water has a much higher dielectric constant (~80) compared to air (~1) or dry soil (~3-5).

The sensor generates an electric field between the plates, and as the soil's moisture level increases, the dielectric constant rises, causing an increase in capacitance. This change in capacitance is then converted into an electrical signal, which can be processed and interpreted as soil moisture content. The output is usually in the form of an analog or digital signal, which can be calibrated to provide accurate moisture readings.

**2. Components of a Capacitive Soil Moisture Sensor**

A capacitive soil moisture sensor typically includes the following key components:

1. **Capacitor Plates**:
   * These are usually made of corrosion-resistant materials such as stainless steel or coated with anti-corrosive layers to ensure durability in various soil conditions.
2. **Circuit Board**:
   * Contains the electronics for generating the electric field, measuring the capacitance, and converting it into a readable signal.
3. **Microcontroller**:
   * Processes the signal from the capacitor and converts it into soil moisture data, often in percentage or volumetric water content.
4. **Power Supply**:
   * Provides the energy needed for sensor operation, often through batteries or solar panels in IoT-enabled systems.
5. **Output Interface**:
   * Allows the sensor to communicate data to external devices, such as a display, data logger, or IoT platform, via analog, digital, or wireless connections.

**3. Advantages of Capacitive Soil Moisture Sensors**

Capacitive soil moisture sensors offer several advantages over traditional resistive sensors:

1. **Durability**:
   * Unlike resistive sensors that corrode over time due to prolonged contact with soil and water, capacitive sensors are non-corrosive and have a longer lifespan.
2. **Accuracy**:
   * By relying on the dielectric properties of soil, capacitive sensors provide more reliable and consistent readings, minimizing errors caused by soil conductivity.
3. **Low Power Consumption**:
   * These sensors are energy-efficient, making them suitable for battery-operated or solar-powered systems.
4. **Cost-Effectiveness**:
   * While slightly more expensive initially, their long lifespan and low maintenance requirements reduce long-term costs.
5. **Environmental Adaptability**:
   * Capacitive sensors perform well across different soil types and environmental conditions, including sandy, clayey, and loamy soils.
6. **Compatibility with IoT**:
   * Many capacitive sensors are designed for integration with IoT platforms, enabling real-time monitoring and data-driven decision-making.

**4. Applications of Capacitive Soil Moisture Sensors**

The versatility of capacitive soil moisture sensors makes them ideal for various applications:

1. **Agriculture**:
   * Used in precision farming to monitor soil moisture levels, ensuring optimal irrigation. This helps conserve water, improve crop yields, and prevent over-irrigation.
2. **Smart Irrigation Systems**:
   * Integrated into automated irrigation systems to control water supply based on real-time soil moisture data, reducing wastage.
3. **Greenhouse Monitoring**:
   * Helps maintain ideal soil moisture levels for plant growth in controlled environments, enhancing productivity.
4. **Landscaping and Gardening**:
   * Used to monitor moisture levels in gardens, parks, and golf courses, ensuring healthy vegetation with minimal water use.
5. **Environmental Research**:
   * Employed in hydrology studies, soil health assessment, and climate change research to measure soil-water interactions.
6. **Forestry and Ecology**:
   * Used to monitor soil moisture in forests and natural reserves, aiding in conservation efforts and wildfire prevention.

**5. Integration with IoT and Automation**

Modern capacitive soil moisture sensors are increasingly integrated into **IoT-based systems**, making them an essential component of smart agriculture. These sensors can transmit real-time data to cloud platforms via wireless technologies such as Wi-Fi, LoRa, or Zigbee. Farmers can access this data on mobile apps or web dashboards, enabling remote monitoring and control of irrigation systems. Additionally, the integration with AI and machine learning models allows predictive analysis, helping optimize water usage and crop growth.

Automation is another critical area where capacitive sensors excel. By connecting the sensors to actuators, such as solenoid valves or pumps, the irrigation process can be fully automated. The system activates irrigation only when soil moisture levels fall below a predefined threshold, ensuring water is applied precisely when and where it is needed. This reduces labor, saves water, and improves overall efficiency.

**6. Challenges and Limitations**

Despite their numerous advantages, capacitive soil moisture sensors face some challenges:

1. **Calibration**:
   * Sensors need calibration for different soil types to maintain accuracy, as soil composition affects dielectric properties.
2. **Sensitivity to Temperature**:
   * Extreme temperatures can slightly affect the sensor’s readings, requiring compensation algorithms for accurate results.
3. **High Initial Cost**:
   * Compared to resistive sensors, capacitive sensors have a higher upfront cost, though their longevity offsets this over time.
4. **Maintenance**:
   * While more durable than resistive sensors, capacitive sensors require occasional cleaning and recalibration for optimal performance.

**7. Future Prospects**

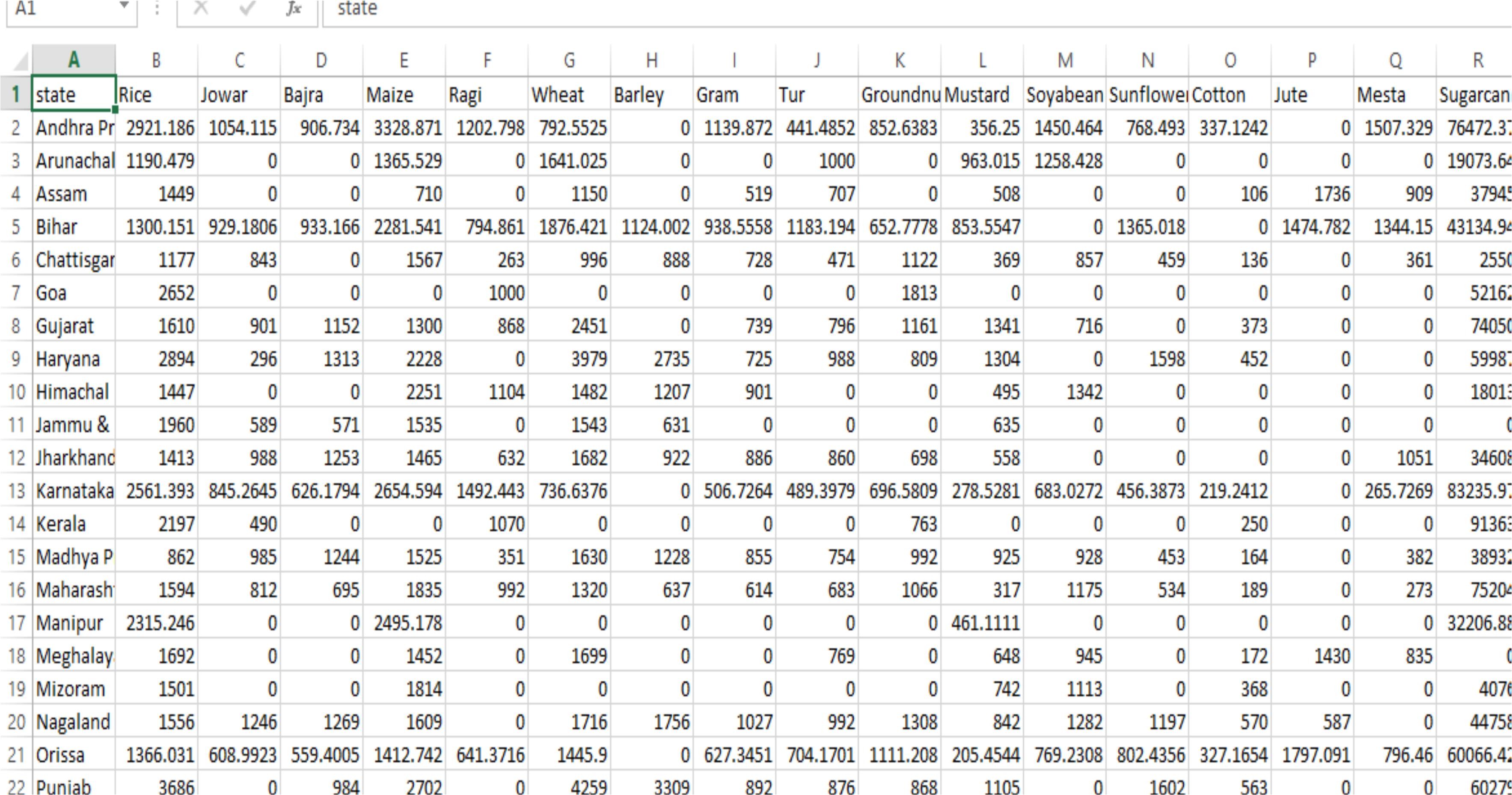
The future of capacitive soil moisture sensors lies in further integration with **smart farming technologies**. Advancements in miniaturization, AI, and IoT connectivity will make these sensors more affordable and accessible to small-scale farmers. Enhanced algorithms for self-calibration and compensation for environmental factors will further improve their accuracy and reliability. Additionally, combining these sensors with weather prediction systems and drone technology could revolutionize water management in agriculture.

**CHAPTER 8**

# Implementation

### Data Analysis

One of the first steps we perform during implementation is an analysis of the data. This was done by us in an attempt to find the presence of any relationships between the various attributes present in the dataset. Acquisition of Training Dataset: The accuracy of any machine learning algorithm depends on the number of parameters and the correctness of the training dataset. We In this project analyzed multiple datasets collected from Government website -https://data.gov.in/ and Kaggle and carefully selected the parameters that would give the best results. Many work done in this field have considered environmental parameters to predict crop sustainability some have used yield as major factor where as in some works only economic factors are taken into consideration. We have tried to combine both environmental parameters like rainfall, temperature, ph, nutrients in soil, soil type, location and economic parameters like production, and yield to provide accurate and reliable recommendation to the farmer on which crop will be most suitable for his land.



**Fig-8.**1 Yield Dataset

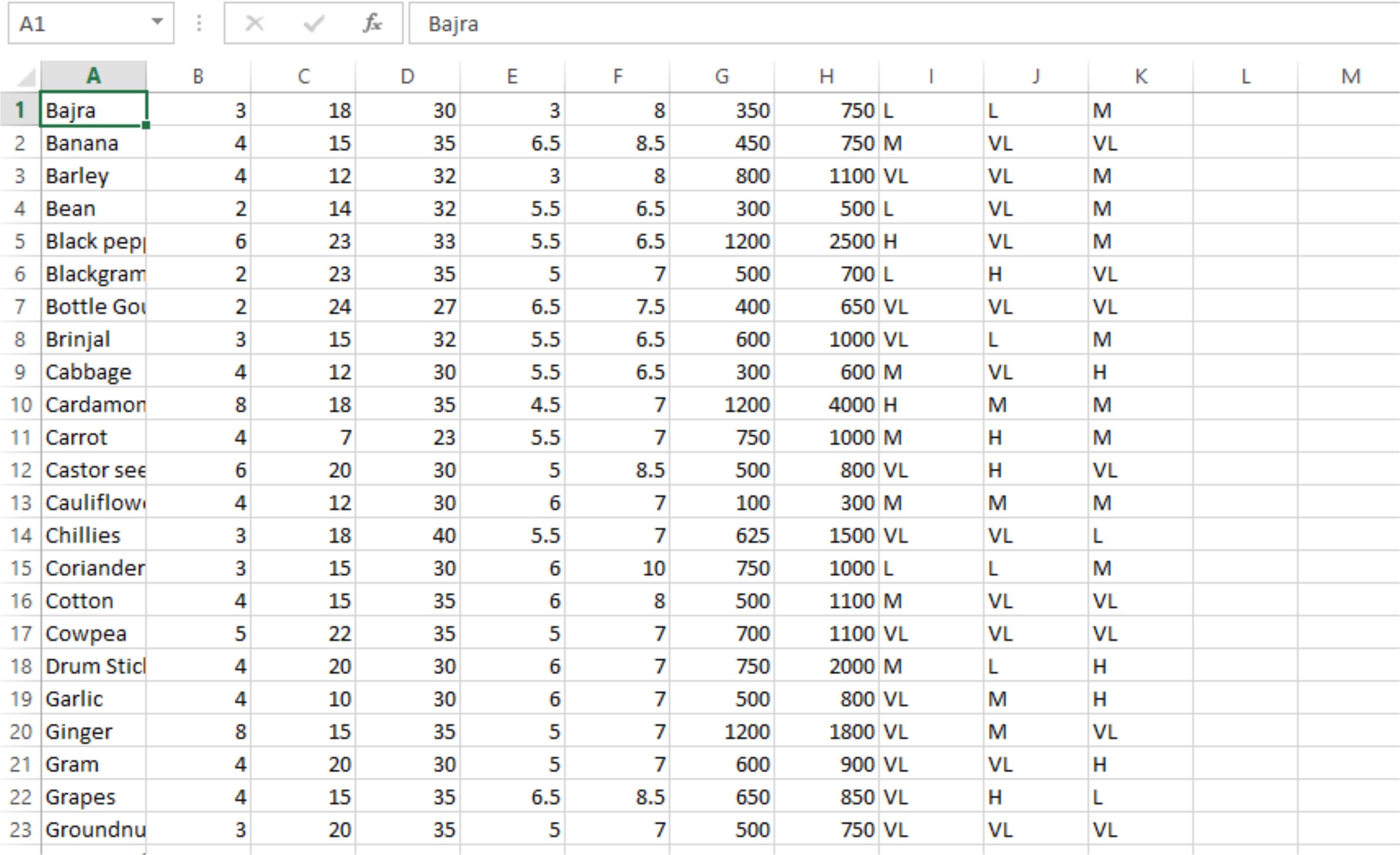


Fig-8.2 Temperature Rainfall and Nutrients dataset

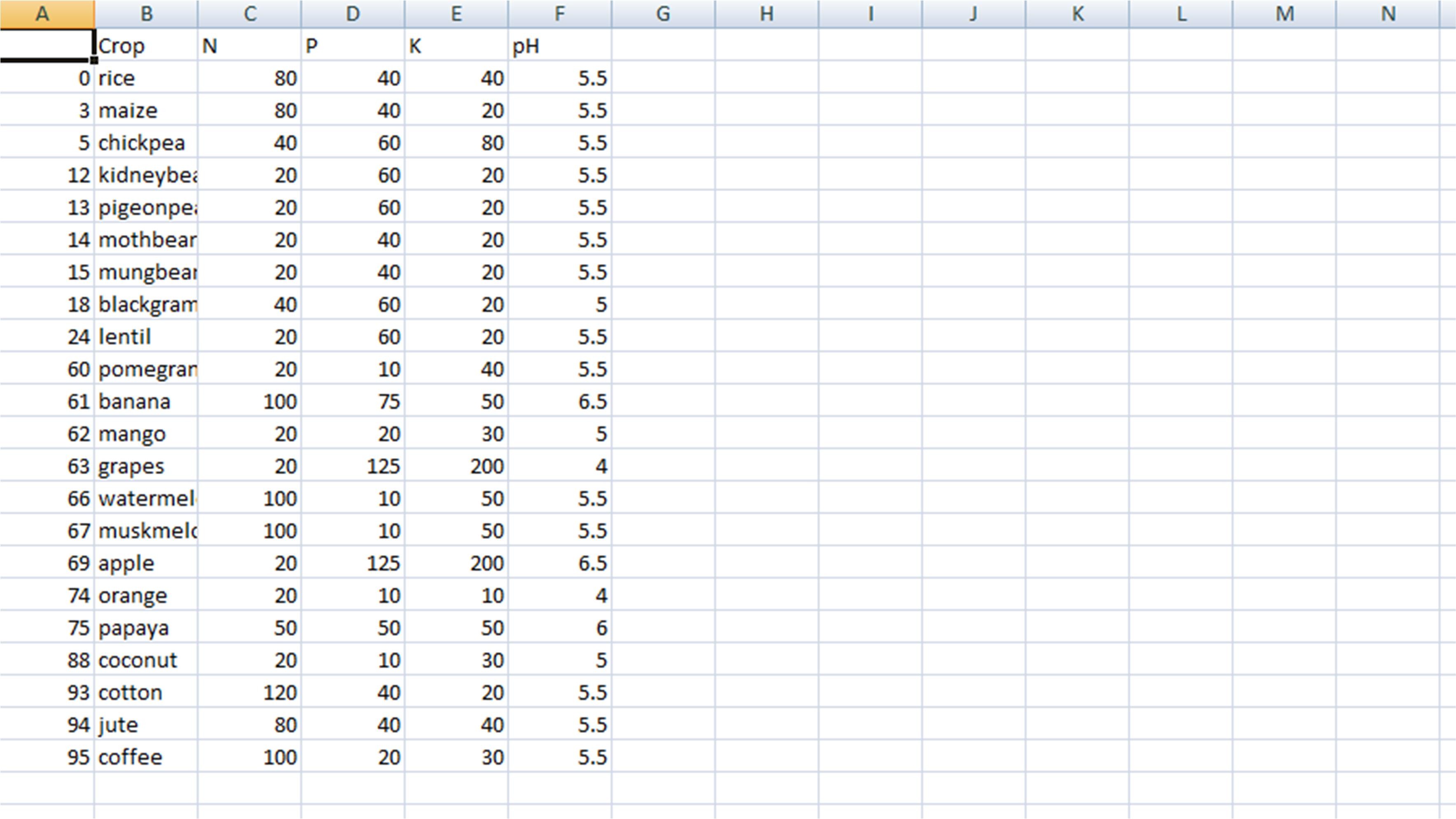


Fig-8.**3** Soil Nutrients distribution as per crop(Nitrogen,Phosphorous,Potassium).

### 8.2Data Preprocessing

After analyzing and visualizing the data, the next step is preprocessing.

Data preprocessing is an important step as it helps in cleaning the data and making it suitable for use in machine learning algorithms. Most of the focus in preprocessing is to remove any outliers or erroneous data, as well as handling any missing values.

Missing data can be dealt with in two ways. The first method is to simply remove the entire row which contains the missing or error value. While this an easy to execute method, it is better to use only on large datasets. Using this method on small datasets can reduce the dataset size too much, especially if there are a lot of missing values.

This can severely affect the accuracy of the result. Since ours is a relatively small dataset, we will not be using this method. The dataset that we used had values that were in string format so we had to transform and encode the into integer valued so as to pass as an input to the neural network. First we converted the data into pandas categorical data and then generated codes for crops and states respectively we than appended these and created separated datasets.

import pandas as pd # Reading the data

crop\_data\_path = '../Data-raw/cpdata.csv' fertilizer\_data\_path = '../Data-raw/Fertilizer.csv' crop = pd.read\_csv(crop\_data\_path)

fert = pd.read\_csv(fertilizer\_data\_path) # Function for lowering the cases defchange\_case(i):

i = i.replace(" ", "") i = i.lower()

return i

fert['Crop'] = fert['Crop'].apply(change\_case) crop['label'] = crop['label'].apply(change\_case) crop\_names = crop['label'].unique() print(crop\_names)

crop\_names\_from\_fert = fert['Crop'].unique() print(crop\_names\_from\_fert)

for i in crop\_names\_from\_fert: print(crop[crop['label'] == i])

extract\_labels = []

for i in crop\_names\_from\_fert: if i in crop\_names:

import pandas as pd # Reading the data

crop\_data\_path = '../Data-raw/cpdata.csv' fertilizer\_data\_path = '../Data-raw/Fertilizer.csv' crop = pd.read\_csv(crop\_data\_path)

fert = pd.read\_csv(fertilizer\_data\_path) # Function for lowering the cases defchange\_case(i):

i = i.replace(" ", "") i = i.lower()

return i

fert['Crop'] = fert['Crop'].apply(change\_case) crop['label'] = crop['label'].apply(change\_case) crop\_names = crop['label'].unique() print(crop\_names)

crop\_names\_from\_fert = fert['Crop'].unique() print(crop\_names\_from\_fert)

for i in crop\_names\_from\_fert: print(crop[crop['label'] == i])

extract\_labels = []

for i in crop\_names\_from\_fert: if i in crop\_names:

extract\_labels.append(i)

# using extract labesl on crop to get all the data related to those labels new\_crop = pd.DataFrame(columns = crop.columns)

new\_fert = pd.DataFrame(columns = fert.columns) for label in extract\_labels:

new\_crop = new\_crop.append(crop[crop['label'] == label]) for label in extract\_labels:

new\_fert = new\_fert.append(fert[fert['Crop'] == label].iloc[0]) print(new\_crop)

print(new\_fert)

new\_crop.to\_csv('../Data-raw/MergeFileCrop.csv') new\_fert.to\_csv('../Data-raw/FertilizerData.csv')

# 8.3Machine Learning Models

extract\_labels.append(i)

# using extract labesl on crop to get all the data related to those labels new\_crop = pd.DataFrame(columns = crop.columns)

new\_fert = pd.DataFrame(columns = fert.columns)

for label in extract\_labels:

new\_crop = new\_crop.append(crop[crop['label'] == label]) for label in extract\_labels:

new\_fert = new\_fert.append(fert[fert['Crop'] == label].iloc[0]) print(new\_crop)

print(new\_fert)

new\_crop.to\_csv('../Data-raw/MergeFileCrop.csv') new\_fert.to\_csv('../Data-raw/FertilizerData.csv')

### 8.3.1Decision Tree :

**Decision Trees (DTs)** are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the

decision rules and the fitter the model.

features = df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']] target = df['label']

#features = df[['temperature', 'humidity', 'ph', 'rainfall']] labels = df['label']

# Initialzing empty lists to append all model's name and corresponding name acc = []

model = []

# Splitting into train and test data

from sklearn.model\_selection import train\_test\_split

Xtrain, Xtest, Ytrain, Ytest = train\_test\_split(features,target,test\_size = 0.2,random\_state =2) from sklearn.tree import DecisionTreeClassifier

DecisionTree = DecisionTreeClassifier(criterion="entropy",random\_state=2,max\_depth=5) DecisionTree.fit(Xtrain,Ytrain)

predicted\_values = DecisionTree.predict(Xtest)

x = metrics.accuracy\_score(Ytest, predicted\_values) acc.append(x)

model.append('Decision Tree') print("DecisionTrees's Accuracy is: ", x\*100) print(classification\_report(Ytest,predicted\_values)) from sklearn.model\_selection import cross\_val\_score # Cross validation score (Decision Tree)



score = cross\_val\_score(DecisionTree, features, target,cv=5) print(score)

#Saving trained Decision Tree model import pickle

# Dump the trained Naive Bayes classifier with Pickle DT\_pkl\_filename = '../models/DecisionTree.pkl'

# Open the file to save as pkl file DT\_Model\_pkl = open(DT\_pkl\_filename, 'wb') pickle.dump(DecisionTree, DT\_Model\_pkl)

# Close the pickle instances DT\_Model\_pkl.close()

### 8.3.1Support Vector Machine

score = cross\_val\_score(DecisionTree, features, target,cv=5) print(score)

#Saving trained Decision Tree model import pickle

# Dump the trained Naive Bayes classifier with Pickle DT\_pkl\_filename = '../models/DecisionTree.pkl'

# Open the file to save as pkl file DT\_Model\_pkl = open(DT\_pkl\_filename, 'wb') pickle.dump(DecisionTree, DT\_Model\_pkl)

# Close the pickle instances DT\_Model\_pkl.close()

The objective of the support vector machine algorithm is to find a hyperplane in an N- dimensional space(N — the number of features) that distinctly classifies the data points.

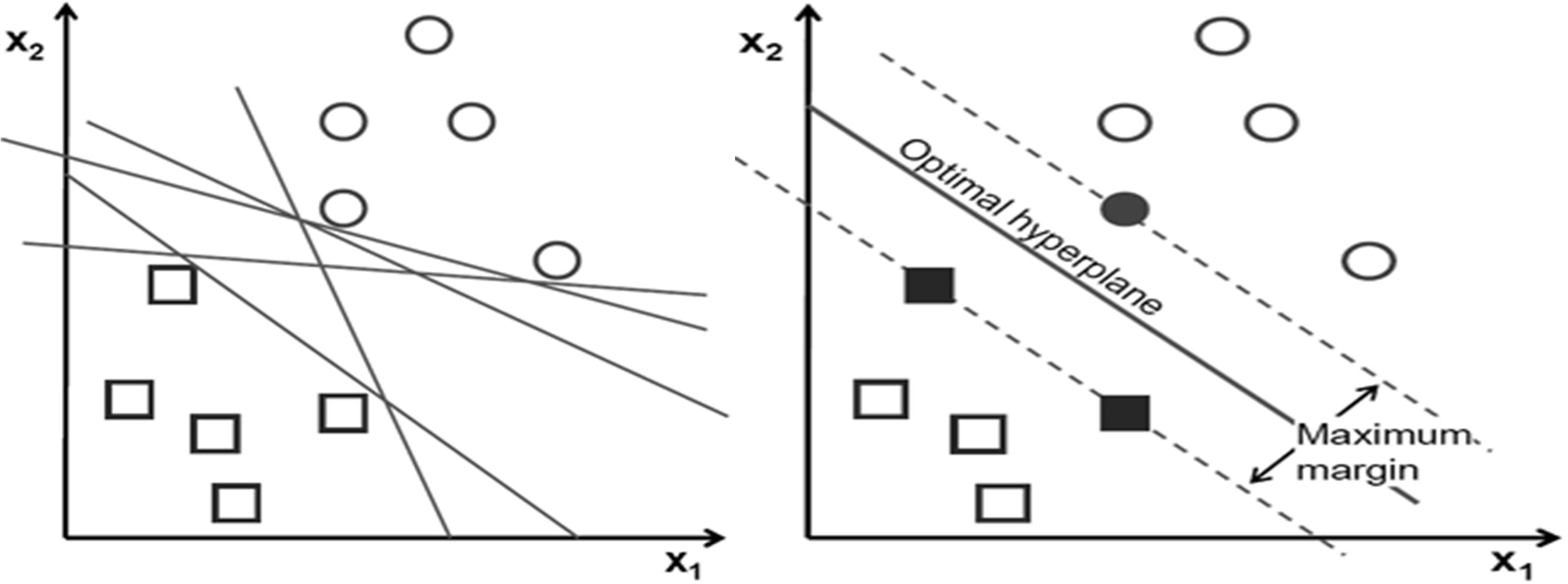
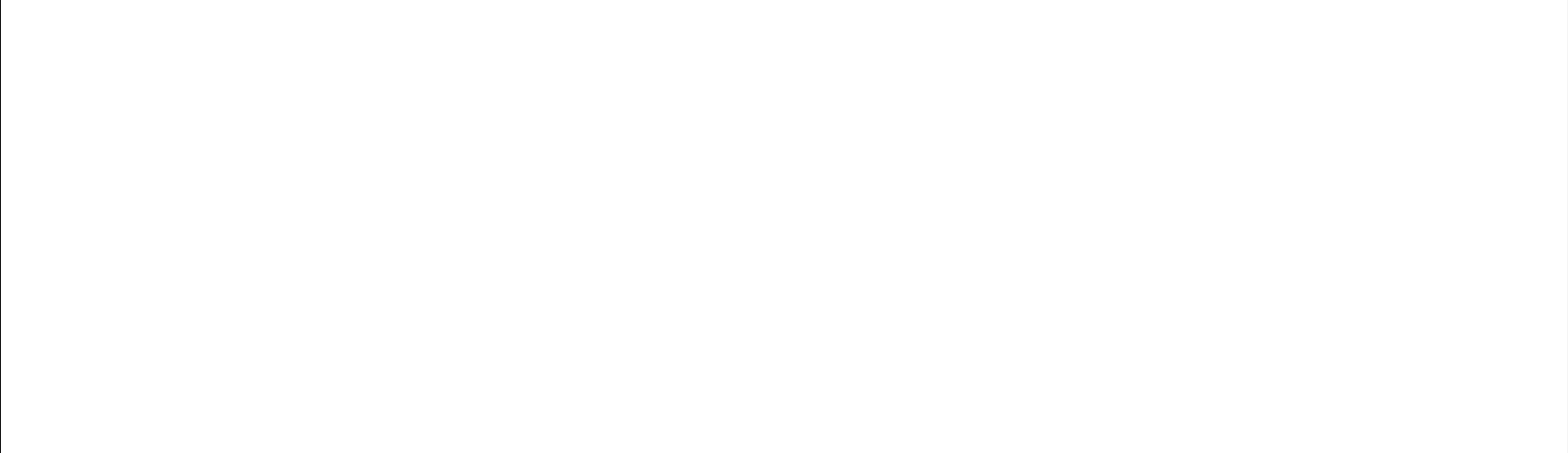


Fig-8 support vector machine

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.



from sklearn.svm import SVC SVM = SVC(gamma='auto') SVM.fit(Xtrain,Ytrain)

predicted\_values = SVM.predict(Xtest)

x = metrics.accuracy\_score(Ytest, predicted\_values) acc.append(x)

model.append('SVM') print("SVM's Accuracy is: ", x)

print(classification\_report(Ytest,predicted\_values)) score = cross\_val\_score(SVM,features,target,cv=5) print(score)

### Logistic Regression

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Some of the examples of classification problems are Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign. Logistic regression transforms its output using the logistic sigmoid function to return a probability value.

**What are the types of logistic regression**

* + - 1. Binary (eg. Tumor Malignant or Benign)
      2. Multi-linear functions failsClass (eg. Cats, dogs or Sheep's)

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.

from sklearn.linear\_model import LogisticRegression LogReg = LogisticRegression(random\_state=2) LogReg.fit(Xtrain,Ytrain)

predicted\_values = LogReg.predict(Xtest)

x = metrics.accuracy\_score(Ytest, predicted\_values) acc.append(x)

model.append('Logistic Regression') print("Logistic Regression's Accuracy is: ", x) print(classification\_report(Ytest,predicted\_values))

score = cross\_val\_score(LogReg,features,target,cv=5) print(score)

import pickle

# Dump the trained Naive Bayes classifier with Pickle LR\_pkl\_filename = '../models/LogisticRegression.pkl' # Open the file to save as pkl file

LR\_Model\_pkl = open(DT\_pkl\_filename, 'wb') pickle.dump(LogReg, LR\_Model\_pkl)

# Close the pickle instances LR\_Model\_pkl.close()

### Random Forest

Random forest is a **Supervised Machine Learning Algorithm**that is **used widely in Classification and Regression problems**. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing **continuous variables** as in the case of regression and **categorical variables** as in the case of classification. It performs better results for classification problems.

Random Forests was developed specifically to address the problem of high-variance in Decision Trees. Like the name suggests, you’re not training a single Decision Tree, you’re training an entire forest! In this case, a forest of Bagged Decision Trees.

Random Forests algorithm follows these steps:

Take the original dataset and create *N* bagged samples of size *n*, with *n* smaller than the original dataset.

Train a Decision Tree with each of the *N* bagged datasets as input. But, when doing a node split, don’t explore all features in the dataset. Randomly select a smaller number, *M* features, from all the features in training set. Then pick the best split using impurity measures, like Gini Impurity or Entropy.Aggregate the results of the individual decision trees into a single output.

Average the values for each observation, produced by each tree, if you’re working on a Regression task.

Do a majority vote across all trees, for each observation, if you’re working on a Regression task.

from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(n\_estimators=20, random\_state=0) RF.fit(Xtrain,Ytrain)

predicted\_values = RF.predict(Xtest)

x = metrics.accuracy\_score(Ytest, predicted\_values) acc.append(x)

model.append('RF') print("RF's Accuracy is: ", x)

print(classification\_report(Ytest,predicted\_values)) # Cross validation score (Random Forest)

score = cross\_val\_score(RF,features,target,cv=5) print(score)

import pickle

# Dump the trained Naive Bayes classifier with Pickle RF\_pkl\_filename = '../models/RandomForest.pkl'

# Open the file to save as pkl file RF\_Model\_pkl = open(RF\_pkl\_filename, 'wb') pickle.dump(RF, RF\_Model\_pkl)

# Close the pickle instances RF\_Model\_pkl.close()

from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(n\_estimators=20, random\_state=0) RF.fit(Xtrain,Ytrain)

predicted\_values = RF.predict(Xtest)

x = metrics.accuracy\_score(Ytest, predicted\_values) acc.append(x)

model.append('RF') print("RF's Accuracy is: ", x)

print(classification\_report(Ytest,predicted\_values)) # Cross validation score (Random Forest)

score = cross\_val\_score(RF,features,target,cv=5) print(score)

import pickle

# Dump the trained Naive Bayes classifier with Pickle RF\_pkl\_filename = '../models/RandomForest.pkl'

# Open the file to save as pkl file RF\_Model\_pkl = open(RF\_pkl\_filename, 'wb') pickle.dump(RF, RF\_Model\_pkl)

# Close the pickle instances RF\_Model\_pkl.close()

## CHAPTER 9

**TESTING**

### 9.1Testing Methodologies

The program comprises of several algorithms which are tested individually for the accuracy. we check for the correctness of the program as a whole and how it performs.

### Unit Testing

Unit tests focus on ensuring that the correct changes to the world state take place when a transaction is processed. The business login transaction processor functions should have unit tests, ideally with 100 percent code coverage. This will ensure that you do not have typos or logic errors in the business logic. The various modules can be individually run from a command line and tested for correctness. The tester can pass various values, to check the answer returned and verify it with the values given to him/her. The other work around is to write a script, and run all the tests using it and write the output to a log \_le and using that to verify the results. We tested each of the algorithms individually and made changes in preprocessing accordingly to increase the accuracy.

### System Testing

System Testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the systems compliance with the specified requirements. System Testing is the testing of a complete and fully integrated software product and White Box Testing. System test falls under the black box testing category of software testing. Different Types of System Testing:

* + 1. **Usability Testing** - Usability Testing mainly focuses on the users ease to use the application, exibility in handling controls and ability of the system to meet its objectives.
    2. **Load Testing** - Load Testing is necessary to know that a software **Migration Testing** - Migration testing is done to ensure that the software can be moved from older system infrastructures to current system infrastructures without any issues.

### Quality Assurance

Quality Assurance is popularly known as QA Testing, is defined as an activity to ensure that an organization is providing the best possible product or service to customers. QA focuses on improving the processes to deliver Quality Products to the customer. An organization

has to ensure, that processes are efficient and effective as per the quality standards defined for software products.

### Functional Test

Functional Testing is also known as functional completeness testing, Functional Testing involves trying to think of any possible missing functions. As chat-bot evolves into new application areas, functional testing of essential chatbot components. Functional testing evaluates use-case scenarios and related business processes, such as the behavior

of smart contracts.

## CHAPTER 10

**RESULTS**

# Algorithms and Their Accuracy

For the purposes of this project we have used four popular algorithms:

Decision Trees, Logistic regression, Support Vector Machine and Random Forest. All the algorithms are based on supervised learning. Our overall system is divided

into two modules:

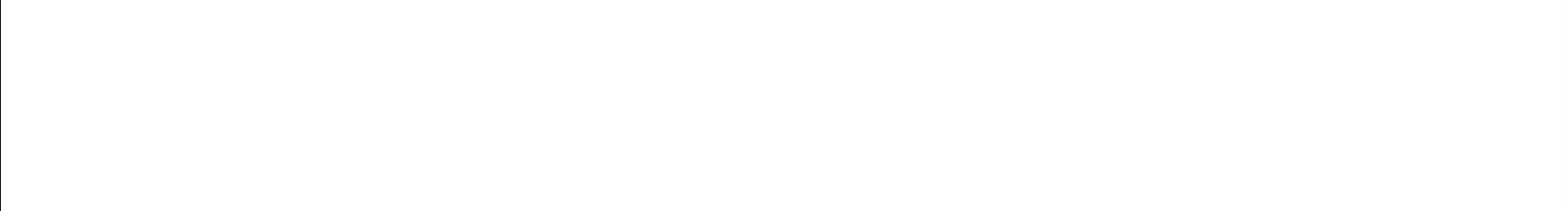
* + - Crop recommender
    - Fertilizer Recommender/Suggestion

|  |  |
| --- | --- |
| **Algorithm** | **Accuracy (%)** |
| **Decision Tree** | 90% |
| **SVM** | 97% |
| **Logistic Regression** | 95% |
| **Random Forest** | 99% |

### Accuracy Comparison of ML Models:

plt.figure(figsize=[10,5],dpi = 100) plt.title('Accuracy Comparison') plt.xlabel('Accuracy') plt.ylabel('Algorithm')

sns.barplot(x = acc,y = model,palette='dark')



plt.figure(figsize=[10,5],dpi = 100) plt.title('Accuracy Comparison') plt.xlabel('Accuracy') plt.ylabel('Algorithm')

sns.barplot(x = acc,y = model,palette='dark')

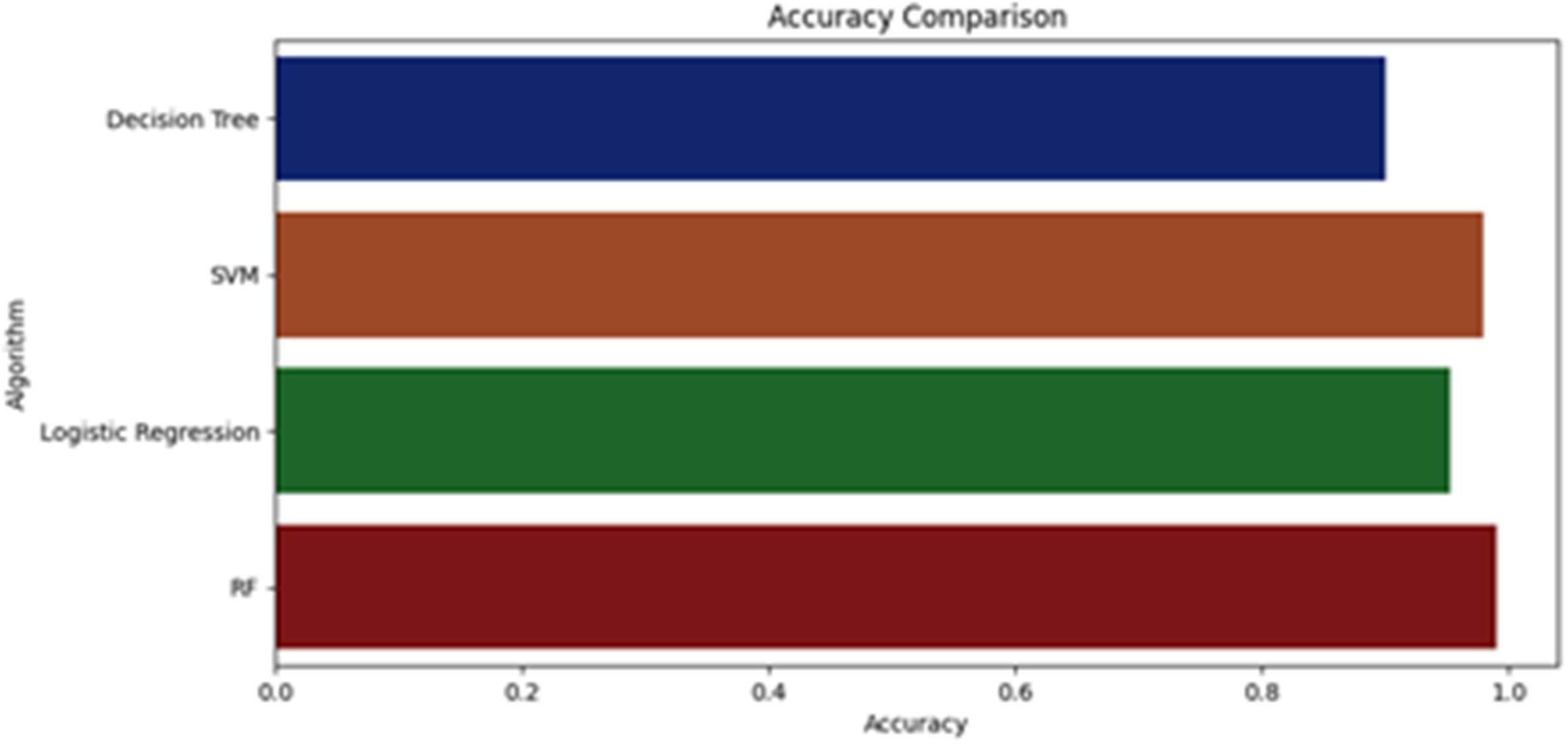


Fig-9 Accuracy Comparison

Hence, **Random Forest** is our Final efficient model.

### Output for Crop recommender:

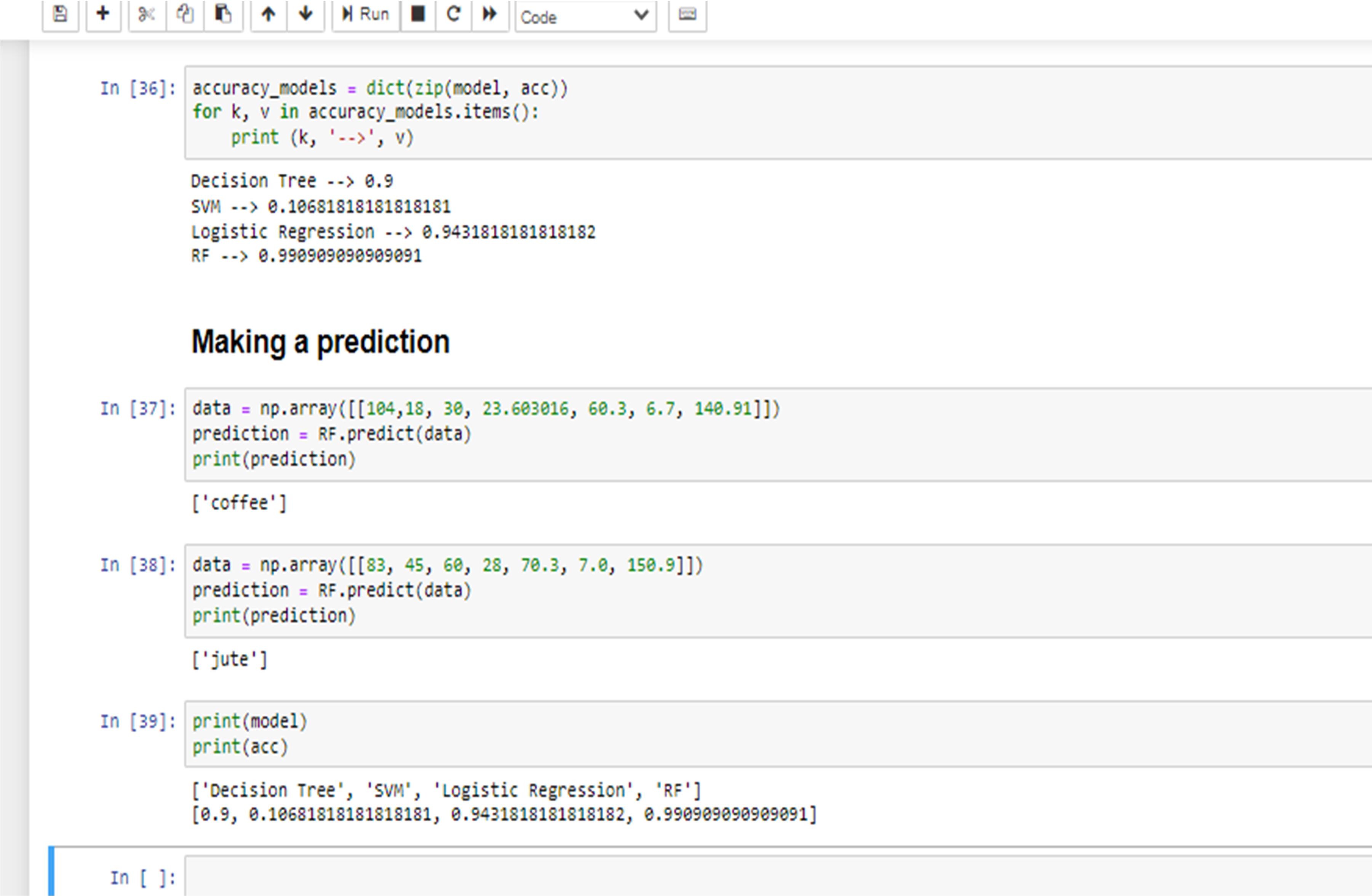


Fig-10 Output crop recommender

### Fertilizer Recommender/Suggestion

The fertilizer recommendation system is purely python logic based. In this we compare the data (optimum nutrients for growing the crop) with the user’s entered data. Then nutrient having maximum difference is made as HIGH or LOW and according to that suggestions will be fetched.

**CHAPTER 11**

# CONCLUSION

All This system helps the farmer to choose the right crop by providing insights that ordinary farmers don't keep track of thereby decreasing the chances of crop failure and increasing productivity. It also prevents them from incurring losses. The system can be extended to the web and can be accessed by millions of farmers across the country. We could achieve an accuracy of 90 percent from the Decision Trees, an accuracy of 70.6 percent from the Support Vector Machine, an accuracy of 94.30 percent from the Logistic Regression and an accuracy of 99.09 percent from the Random Forest model. Further development is to integrate the crop recommendation system with another subsystem, yield predictor that would also provide the farmer an estimate of production if he plants the recommended crop.

.

**CHAPTER 12**

# REFERENCES

1. 2019, 10th International Conference on Computing, Communication and Networking Technologies, “Low-cost IOT+ML design for smart farming with multiple applications”, Fahad Kamraan Syed, Agniswar Paul, Ajay Kumar, Jaideep Cherukuri.
2. 2019 IEEE “ Smart Management of Crop Cultivation using IoT and Machine Learning”

Archana Gupta, Dharmil Nagda, Pratiksha Nikhare, Atharva Sandbhor

1. Radhika, Narendiran, “Kind of Crops and Small Plants Prediction using IoT with Machine Learning,” International Journal of Computer & Mathematical Sciences, 2018.
2. “Crop Recommendation on Analyzing Soil Using Machine Learning” Anguraj.Ka, Thiyaneswaran.Bb, Megashree.Gc, Preetha Shri.J.Gd, Navya.Se, Jayanthi. Jf, 2020.
3. “Classification of Soil and Crop Suggestion using Machine Learning Techniques”, A. Mythili , IEEE 2019.
4. Mehta, P., Shah, H., Kori, V., Vikani, V., Shukla, S., & Shenoy, M.,2018. “Survey of unsupervised machine learning algorithms on precision agricultural data”, IEEE
5. “IOT based Crop Recommendation, Crop Disease Prediction and Its Solution” Rani Holambe, Pooja Patil, Padmaja Pawar, Saurabh Salunkhe , Mr. Hrushikesh Joshi, 2019 IRJET
6. Tom M. Mitchell, Machine Learning, India Edition 2013, McGrawHill Education.
7. https://data.gov.in/
8. Kaggle: https://[www.kaggle.com/datasets](http://www.kaggle.com/datasets)