Smart Irrigation Applying ML and IoT

A report submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

in

Electronics & Communication and Engineering

by

Ayush Singh (ECE19U001)
Krishna Laddha (ECE19U010)
Nikhil Chourase (ECE19U017)
M V Pavan Kumar (ECE19U012)

under the guidance of

Dr. R. Krishnamurthy

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING.

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY TIRUCHIRAPPALLI

SETHURAPATTI, TIRUCHIRAPPALLI – 620012

MAY 2023

BONAFIDE CERTIFICATE

This is to certify that the project work titled "**Smart Irrigation Applying ML and IoT**" is a bonafide record of the work done by

Ayush Singh - ECE19U001
Krishna Laddha - ECE19U010
Nikhil Chourase - ECE19U017
M V Pavan Kumar - ECE19U012

in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Electronics and Communication Engineering** of the **Indian Institute of Information Technology Tiruchirappalli** during the year 2022-2023. The contents of this report, in full or in parts, have not been submitted to any other institute or university for the award of any degree or diploma.

Dr. R. Krishnamurthy
Asst. Professor
Supervisor

Dr. G. Seetharaman
Head of the Department

Project viva-voce held on ___________________

Internal Examiner

External Examiner

ABSTRACT

Farms can be upgraded with electronic technology to continuously monitor crop and soil conditions, allowing crops to be watered as needed. This can be controlled and monitored online using IoT applications. You can create a device that connects to a water pump controlled by an Arduino UNO R3 microcontroller. The water pump automatically controls based on the values of various environmental factors such as temperature, humidity, soil moisture and light intensity, which can be measured via sensors such as DHT-11 temperature and humidity sensor, humidity sensor, LDR sensor. Our research work focuses on making the model intelligent by storing previously scanned values in a database and performing pre-recognition based on the stored historical values during the training phase of the working model. The model is trained using the Random Forest Classifier algorithm. Agriculture plays an important role in the economy, and its contribution is based on quantifiable crop yields that heavily rely on irrigation. In countries like India, where agriculture is largely based on an unorganized sector, irrigation techniques and patterns are often inefficient, leading to unnecessary water wastage. An automated irrigation system based on artificial vision and the Internet of Things can autonomously irrigate fields using soil moisture data. The system is based on forecasting algorithms that use historical weather data to identify and forecast precipitation patterns and climate change. This creates a sophisticated system that selectively irrigates crop fields only when needed at the right time based on weather and soil moisture conditions. The system has been tested with 89% accuracy in a controlled environment and offers an efficient solution to your dilemma.

Keywords: Machine Learning, Internet of Things, Arduino, Neural Networks, Rainfall Prediction

ACKNOWLEDGEMENT

We wish to record my deep sense of gratitude and profound thanks to our project supervisor **Dr. R. Krishnamurthy,** Assistant Professor, Electronics and Communication Engineering Department, Indian Institute of Information Technology Tiruchirappalli for his keen interest, inspiring guidance, constant encouragement with our project work during all stages, to bring this project report into fruition.

We thank **Prof. NVSN. Sarma,** Director, Indian Institute of Information Technology Tiruchirappalli and **Dr. G. Seetharaman**, Head, Department of ECE, Indian Institute of Information Technology Tiruchirappalli for providing us all the facilities to complete our project work.

We also thank the faculty and non-teaching staff members of the Department of ECE, Indian Institute of Information Technology Tiruchirappalli for their valuable support throughout the course of our project work.

AYUSH SINGH

KRISHNA LADDHA

NIKHIL CHOURASE

M V PAVAN KUMAR

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In India, traditional agricultural methods are followed for irrigation, which heavily relies on personal supervision and experience of farmers. This results in highly inefficient and outdated irrigation practices, leading to poor crop yield and wastage of water resources. It is essential to provide farmers with reliable and adaptable irrigation solutions that can accurately determine the amount of water their crops require based on local climatic conditions. The main challenge in India is not water scarcity, but water wastage and poor resource utilization due to lack of infrastructure and facilities. Due to this issue, the country is facing significant economic losses from drought conditions, variable rainfall patterns, and crop eradication.

Automated irrigation systems that follow traditional methods are not suitable for India as they are unable to adapt to the country's changing rainfall patterns and are not responsive to geographical variations. India requires an innovative and adaptable irrigation system that can handle these challenges and provide effective water management for agriculture.

The system developed for irrigation in India is unique as it studies local rainfall patterns and adapts to changing weather conditions to predict water requirements for irrigation. The system uses microcontrollers and soil moisture sensors placed in waterproof boxes distributed evenly over the area to be irrigated, connected to the cloud via Wi-Fi. The system analyzes soil moisture levels via sensors and requires little to no human intervention once deployed. The system is designed to update continuously and uses the Desultory Forest Regressor to estimate weather patterns. The developed system is energy-efficient, water-efficient, and low maintenance. Nodes are scattered throughout the farm grounds to minimize water wastage and increase efficiency. The system also works with a replication prediction system, allowing easy identification of failures. Node status can be monitored via a mobile app based on the mapping of farms and areas designated for irrigation. Overall, the system promotes low maintenance and has proven to be effective in reducing water wastage and increasing yield.

MOTIVATION

In India, agriculture heavily relies on irrigation for crop production and water availability significantly impacts crop yield. However, the traditional irrigation methods used in the country are inefficient and result in significant water wastage. Farmers rely on their personal experience and supervision to irrigate their fields, which leads to poor utilization of resources and economic losses due to drought and crop eradication.

Absolutely, efficient and effective irrigation solutions are crucial for sustainable agriculture in India. By implementing innovative technologies like IoT-based irrigation systems and smart water management practices, farmers can optimize water usage, reduce waste, and increase crop yields. This not only benefits the farmers' livelihoods but also contributes to the overall economic development of the country.

Modern irrigation systems that incorporate advanced technologies such as microcontrollers, soil moisture sensors, and cloud computing can help address these challenges. By using these technologies, farmers can accurately determine the amount of water required for their crops based on soil moisture levels and weather conditions. This can help minimize water wastage and increase crop yields. Additionally, these systems can adapt to changing weather patterns and adjust water usage accordingly, making them more efficient and effective than traditional irrigation methods.

Investing in modern technologies and innovative solutions can greatly benefit India's agriculture sector. The government can play a vital role in facilitating the adoption of new technologies by providing financial incentives and policy support to farmers and other stakeholders. The private sector can also contribute by developing and commercializing new irrigation solutions that meet the needs of Indian farmers. Additionally, research institutions and universities can collaborate with farmers and other stakeholders to identify and develop new technologies that are tailored to the unique needs of Indian agriculture. With a collaborative effort, India can improve its irrigation systems and achieve sustainable and efficient crop production.

OBJECTIVES

The use of electronic technology has revolutionized the way irrigation is carried out in farms. By continuously monitoring crop and soil conditions, it is now possible to water crops precisely when they need it. The process can be remotely controlled and monitored using IoT applications, making it convenient for farmers to operate the system.

To achieve this, a device can be created that connects to a water pump and is controlled by an Arduino UNO R3 microcontroller. The water pump can be programmed to operate automatically based on various environmental factors such as temperature, humidity, soil moisture, and light intensity, which are measured using sensors like DHT-11 temperature and humidity sensor, humidity sensor, and LDR sensor.

To make the device more intelligent and efficient, our research work focuses on storing previously scanned values in a database and performing pre-recognition based on the stored historical values during the training phase of the working model. This helps to increase the accuracy of the model's predictions, leading to improved irrigation efficiency and better use of resources. The Random Forest algorithm is used to train the model, which is a widely used algorithm for regression analysis.

The use of electronic technology in agriculture can have a significant impact on India's food security and economic growth. By reducing water wastage and improving crop yields, farmers can generate more income and contribute to the country's economy. Additionally, the conservation of water resources is crucial for addressing water scarcity issues in the country, and the use of electronic irrigation systems can help achieve this goal. Therefore, it is essential to continue investing in and promoting the use of electronic technology in agriculture in India.

CHAPTER 2
LITERATURE REVIEW

1. INTELLIGENT IRRIGATION SYSTEM USING ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Research paper key points:

- The microcontroller of Arduino receives regular data from temperature and moisture sensors through its built-in analog to digital converters.
- The KNN algorithm, implemented in the hardware, processes the sensor data to activate the irrigation pump through Raspberry Pi3.
- The system updates a database with sensor values and irrigation processes for future reference.
- The proposed system aims to combine ANN and KNN methods to provide farmers with data on crop selection, growth, and yield.

Drawback or limitations:

- The system lacks a rainfall prediction system, which can lead to over-irrigation.
- Over-irrigation due to rainfall immediately after the irrigation process can negatively impact crop yield.

2. PREDICTION OF RAINFALL USING INTENSIFIED LSTM BASED RECURRENT NEURAL NETWORK WITH WEIGHTED LINEAR UNITS

Research Paper Key points:

- It is about using deep learning techniques, specifically an Intensified LSTM-based Recurrent Neural Network (ILSTM RNN) with Weighted Linear Units (WLUs), for predicting rainfall. The authors note that accurate rainfall prediction is important for a variety of applications, including agriculture, hydrology, and disaster management.
- The paper presents the ULSTERMEN-WLU model, which combines the strengths of both LSTMs and WLUs. LSTMs are a type of recurrent neural network that are particularly good at modeling sequences of data, while WLUs are a type of activation function that can help prevent the vanishing gradient problem that can occur in deep networks. The authors argue that by using these two techniques in combination, they can achieve better performance in rainfall prediction.
- Overall, the paper presents a novel approach to rainfall prediction using deep learning techniques and demonstrates its effectiveness on a real-world dataset.

3. MACHINE LEARNING: APPLICATIONS IN INDIAN AGRICULTURE

Research paper key points:

- It provides an overview of various applications of machine learning techniques in the field of agriculture in India. The paper highlights the need for modernization of agriculture and the role of technology in achieving the same. It discusses the potential of machine learning in solving various problems related to agriculture such as crop yield prediction, disease detection, soil analysis, weather forecasting, and farm management.
- The paper provides an in-depth analysis of different machine learning algorithms that can be used for agriculture applications and their strengths and weaknesses. It also discusses the challenges faced in implementing machine learning techniques in agriculture and suggests possible solutions to overcome these challenges.

Drawback or limitations:

- The paper primarily focuses on the use of machine learning in predicting pest attacks and disease outbreaks in crops like tomato, paddy, and grapes. However, the techniques used may not necessarily be applicable to other crops, which limits the generalizability of the findings.

- Machine learning models require large amounts of data to train and make accurate predictions. However, the availability of relevant and high-quality data in Indian agriculture can be limited, which may impact the effectiveness of the proposed methods.
- The paper does not explicitly consider the social and economic factors that affect agricultural practices in India, such as farmer education and access to resources. These factors may impact the adoption and success of machine learning applications in Indian agriculture.

4. MACHINE LEARNING CLASSIFICATION TECHNIQUE FOR FAMINE PREDICTION

Research paper key points:

- The research suggests a machine learning-based categorization method for famine forecasting that combines meteorological and satellite imagery data.
- The suggested method uses the Random Forest (RF) algorithm to categorize satellite imagery and meteorological data into several hunger severity levels.
- Only accuracy can be reported in the study, which may not give a complete picture of the model's performance.

5. AUTOMATED IRRIGATION SYSTEM USING A WIRELESS SENSOR NETWORK AND GPRS MODULE

Research Paper Key Points:

- A gateway device also manages sensor data, activates actuators, and sends data to the web application. The system had a dispersed wireless network of soil-moisture and temperature sensors installed in the root zone of the plants.
- To regulate the amount of water used, a microcontroller-based gateway was programmed with an algorithm based on temperature and soil moisture threshold values.
- The system used photovoltaic panels for power and included a duplex communication link based on a cellular-Internet interface that allowed users to schedule irrigation and review data via a web page.

Drawback or Limitations:

- The project's findings were published in a study where the water motor's ability to turn on and off is controlled by the temperature of the soil and a temperature sensor inserted into plant roots. Their lack of a method to inform the user of the status of the agricultural field is a flaw in their project.

6. AN OVERVIEW OF SMART IRRIGATION SYSTEMS USING IOT

Research Paper Key Points:

- The Microcontroller Node MCU ESP8266 is used in the intelligent irrigation system.
- Using the soil sensor, the irrigation system measures the soil's temperature and humidity before watering the plants accordingly.
- Additionally, the system will email you whenever the plants are watered and include all the pertinent plant information, such as temperature and humidity.
- The idea behind this project is to give landowners the ability to oversee and monitor the development of the plants in their farms. The idea we've employed is really straightforward and simple to put into practice. Both labor and time will be saved.

Drawback or limitations:

- Their lack of a method to inform the user of the status of the agricultural field is a flaw in their project.
- This project's disadvantage is that the system can't assess the worth of the plants' nutrients.
- Measurement of soil moisture is the only aspect of the "Automatic Irrigation System on Sensing Soil Moisture Content." But in addition to the soil moisture sensor, our proposed system also includes a temperature sensor.

7.ARDUINO BASED MACHINE LEARNING AND IOT SMART IRRIGATION SYSTEM

Key Points:

- The system is designed to optimize irrigation by using machine learning algorithms to predict soil moisture levels based on various factors such as temperature, humidity, and precipitation.
- The system uses an Arduino board to collect data from sensors and send it to the cloud for processing.
- Machine learning algorithms are used to analyze the data and make predictions about soil moisture levels. The system can then adjust irrigation levels accordingly.
- The system is intended to reduce water consumption and increase crop yields by ensuring that plants receive the optimal amount of water.

Limitations or Drawbacks:

- The accuracy of the machine learning algorithms may depend on the quality and quantity of the data collected. Inaccurate or incomplete data could lead to incorrect predictions and ineffective irrigation.
- The system may require frequent calibration and adjustment to ensure optimal performance.
- The cost of implementing the system, including the cost of sensors and other hardware, may be prohibitive for some farmers or agricultural organizations.
- The system may require a reliable internet connection and access to cloud services, which may not be available in all areas.

CHAPTER 3
PROPOSED METHODOLOGY:

3.1 COMPONENTS AND TECHNOLOGY USED

3.1.1 Arduino Uno R3:

One type of ATmega328P-based microcontroller board is the Arduino Uno R3. It comes with everything needed to support the microcontroller; all you need to do is use a USB cable to connect it to a computer and provide power using an AC-DC adapter or a battery to get things going. The word "Uno" was chosen to signify the launch of the Arduino IDE 1.0 software since it is a "Italian" word that signifies "one". The third and latest version of the Arduino Uno is called the R3. The reference versions of the Arduino board and IDE software are currently being updated. The first in a line of USB-Arduino boards, the Uno-board serves as the platform's reference design.

Arduino Uno R3 configuration

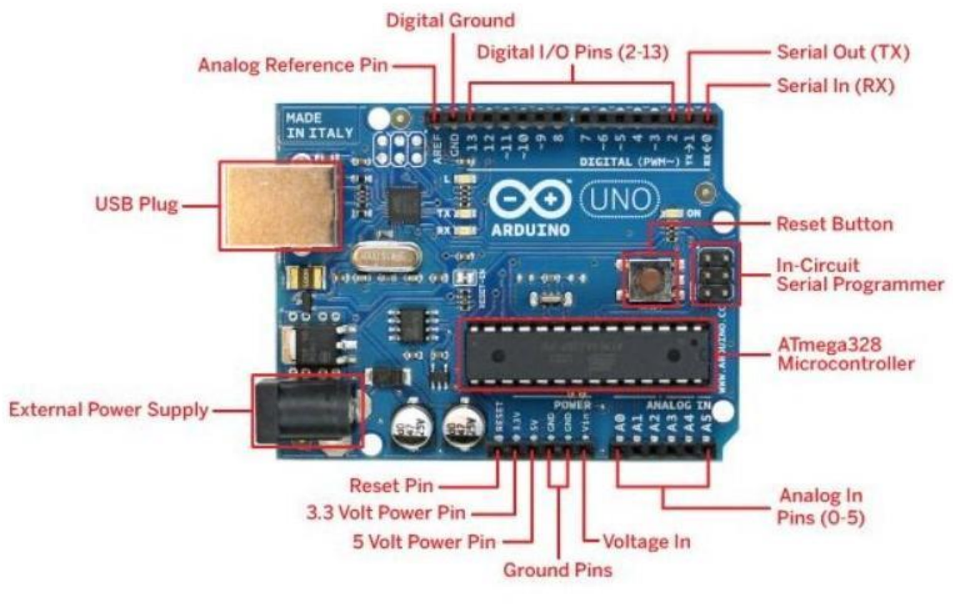


Fig. 3.1.1 Arduino Uno R3 configuration

The Arduino Uno R3 board includes the following specifications.

- It is an ATmega328P based Microcontroller
- The Operating Voltage of the Arduino is 5V
- The recommended input voltage ranges from 7V to 12V
- The i/p voltage (limit) is 6V to 20V
- Digital input and output pins-14
- Digital input & output pins (PWM)-6
- Analog i/p pins are 6
- DC Current for each I/O Pin is 20 mA
- DC Current used for 3.3V Pin is 50 mA
- Flash Memory -32 KB, and 0.5 KB memory is used by the boot loader
- SRAM is 2 KB
- EEPROM is 1 KB
- The speed of the CLK is 16 MHz
- In Built LED
- Length and width of the Arduino are 68.6 mm X 53.4 mm
- The weight of the Arduino board is 25 g

3.1.2 DHT11 Sensor

The DHT11 digital temperature and humidity sensor is a composite sensor that outputs temperature and humidity as calibrated digital signals. The device has great dependability and outstanding long-term stability thanks to the technology of a dedicated digital modules collection as well as the temperature and humidity sensor technology. The sensor is coupled to a high-performance 8-bit microprocessor and has an NTC temperature and a resistive sense of wetness measurement mechanism.

3.1.3 Soil moisture sensor

The volumetric water content of the soil is measured by soil moisture sensors. Soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons as a proxy for the moisture content, because the direct gravimetric measurement of free soil moisture requires removing, drying, and weighing of a sample.

It is necessary to calibrate the relationship between the measured property and soil moisture since it can change based on the environment, including the soil type, temperature, and electric conductivity. The soil moisture has an impact on the reflected microwave radiation, which is employed for remote sensing in agriculture and hydrology. Farmers and gardeners can both employ portable probing tools. Sensors that assess the volumetric water content are commonly referred to as soil moisture sensors. Tensiometers and gypsum blocks are examples of another class of sensors that measure the water potential property of soil moisture. These sensors are also known as soil water potential sensors.

3.1.4 Relay Module

Relays are electrically powered machines. It has a controlled system that is also known as an output circuit or an output contactor, as well as a control system that is also known as an input circuit or an input contactor. Circuits for automatic control typically employ it. To put it simply, it is an automatic switch to a low-current signal regulating a high-current circuit. A relay's lower moment of inertia, stability, long-term dependability, and small volume are its advantages. It is widely used in power protection, automation, sports, remote control, intelligence gathering, and communication equipment, as well as electromechanical and power electronic ones. A relay typically has an induction component that can reflect inputs like current, voltage, power, resistance, frequency, temperature, pressure, speed, and light, among other things. Additionally, it has an actuator module (output) that has the ability to energize or de-energize a controlled circuit's connection. Between the input and output parts, there is an intermediary component that is utilized to couple and isolate input current as well as actuate the output. When the specified input parameters (voltage, current, and temperature etc.) are above the critical value, the controlled output circuit of the relay will be energized or de-energized.

3.1.5 Motor Pump (DC 12V)

A DC motor is any of a class of rotary electrical motors that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor. DC motors were the first form of motor widely used, as they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight brushed motor used for portable power tools and appliances. Larger DC motors are currently used in propulsion of electric vehicles, elevators and hoists, and in drives for steel rolling mills.

3.1.6 Breadboard

The Breadboard serves as a foundation for serving electronics Due to the availability of solderless breadboards, also known as plugboards or terminal array boards, the phrase "breadboard" is now frequently used to describe these. The solderless breadboard is reusable because soldering is not necessary. This makes it simple to use for developing temporary prototypes and conducting circuit design experiments. Solderless breadboards are therefore common among students and in technological education. Earlier breadboard models lacked this characteristic. A variety of electronic systems may be prototyped by using breadboards, from small analog and digital circuits to complete central processing units (CPUs). It is difficult to reuse a stripboard (Veroboard) or other prototyping printed circuit board used to create one-off or semi-permanent soldered prototypes.

3.1.7 Connecting wires

An electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them - simply 'tinned') is known as a jump wire (also known as jumper wire, or jumper), and it is typically used to connect the parts of a breadboard or other prototype or test circuit, internally or with other machinery or components, without soldering. Jump wires are installed individually by placing their "end connections" into a breadboard's slots, a circuit board's header connector, or a piece of test equipment. Male-to-male, male-to-female, and female-to-female jumper wires are the most common types. The wire's termination tip distinguishes each one from the other. Male ends have a pin protruding and can plug into things, while female ends do not and are used to plug things into. Male-to-male jumper wires are the most common and what you likely will use most often. When connecting two ports on a breadboard, a male-to-male wire is what you'll need.

3.2 RAINFALL PREDICTION USING ARTIFICIAL INTELLIGENCE

The rainfall presage involved a one-phase solution:

3.2.1 PREDICTION OF PROBABILITY OF RAINFALL IN THE NEXT 30 MINUTES

The device continuously checks the status at regular intervals when it is turned on periodically. The first phase is to make the network available to determine whether there is a likelihood of rainfall to occur in the next 30 minutes or not.

The average temperature, pressure, wind speed, and air humidity are only a few of the variables that affect rainfall. The dataset utilized is reliable for local areas and regions since it includes parameters that can be used to predict rainfall in general and because all of the parameters had low values. The information used comes from local region-specific rainfall statistics on Weather Underground. The data was split into three sets: training, testing, and a percentage of 70 and 30 respectively.

3.2.2 FLOWCHART OF MODEL METHODOLOGY

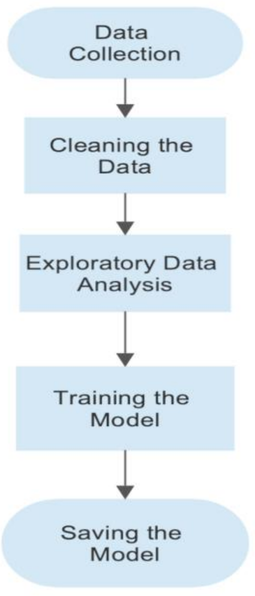


Fig 3.2.2.1 Flow Chart of ML

3.2.3 DATA COLLECTION

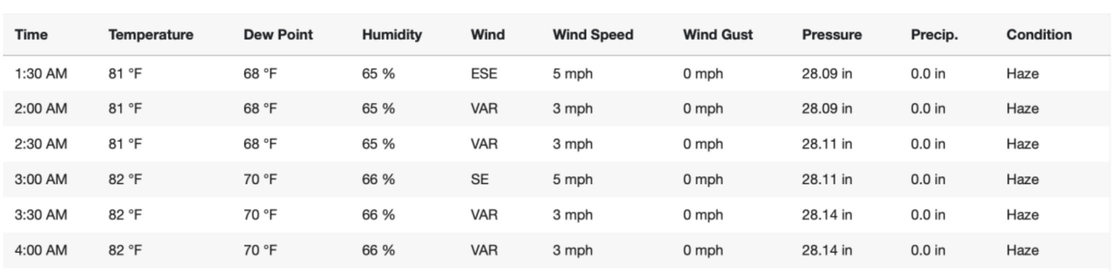
3.2.3.1 Web Scraping:

Web scraping is a method for obtaining data from websites using computer programmes or automated scripts. It entails accessing, downloading, and processing website material and data in order to extract pertinent information. Web scraping is frequently employed for data mining, market research, price monitoring, and other uses that require the collection and analysis of vast amounts of data.

We initially looked for the pertinent data but were unable to find it. They therefore thought about two possibilities. Either purchase the government's data or get it from a website. In the end, we chose the web scraping option and learned how to perform web scraping with Python tools like Selenium and Beautiful Soup 4.

Writing code was required to scrape the webpage and obtain the data about the rainfall. The website's design, however, required that the code wait 15 seconds for the page to fully load before extracting the data. As a result, data collection took longer than expected; it took us 7–10 days to collect the required data.

Cleaning the data is the first stage in the data preparation process. This entails eliminating any columns or data items that are superfluous or not pertinent to the issue at hand. For instance, to ensure that a number column is entirely numerical, any letters in the column should be eliminated.



Time	Temperature	Dew Point	Humidity	Wind	Wind Speed	Wind Gust	Pressure	Precip.	Condition
1:30 AM	81 °F	68 °F	65 %	ESE	5 mph	0 mph	28.09 in	0.0 in	Haze
2:00 AM	81 °F	68 °F	65 %	VAR	3 mph	0 mph	28.09 in	0.0 in	Haze
2:30 AM	81 °F	68 °F	65 %	VAR	3 mph	0 mph	28.11 in	0.0 in	Haze
3:00 AM	82 °F	70 °F	66 %	SE	5 mph	0 mph	28.11 in	0.0 in	Haze
3:30 AM	82 °F	70 °F	66 %	VAR	3 mph	0 mph	28.14 in	0.0 in	Haze
4:00 AM	82 °F	70 °F	66 %	VAR	3 mph	0 mph	28.14 in	0.0 in	Haze

Fig 3.2.3.1.1 Dataset

After that, it's crucial to delete any blank or missing values from the dataset. This can be accomplished in one of two ways: either by deleting the rows that contain the missing information or by imputed values such as the mean or median. The next stage is to look for any outliers in the dataset after the data has been cleaned. It is crucial to find and eliminate outliers since they have a considerable negative impact on the performance of machine learning models.

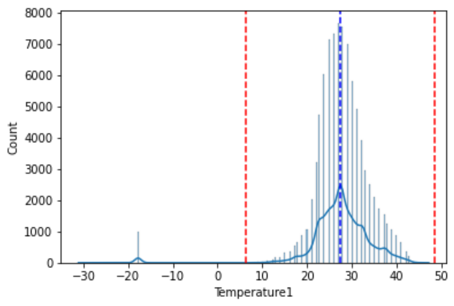


Fig a

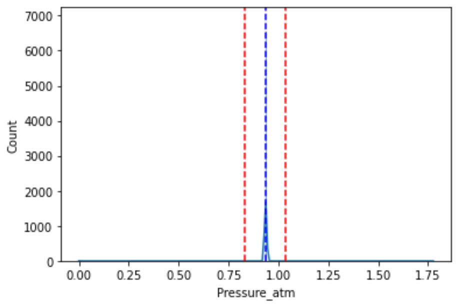


Fig b

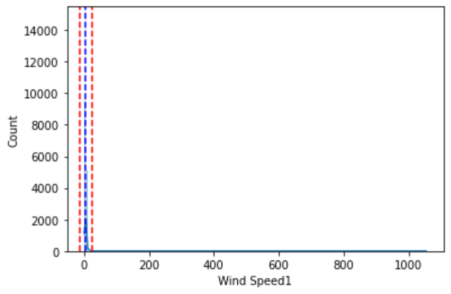


Fig c

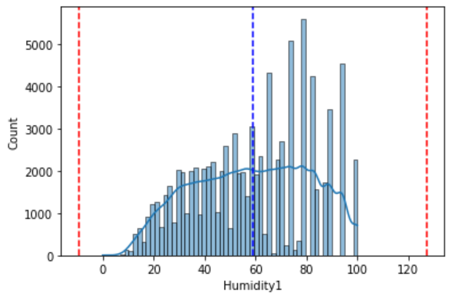


Fig d

Fig 3.2.3.1.2 Histograms before outliers (a. Temperature, b. Pressure (in atm), c Wind Speed(mph), d. Humidity)

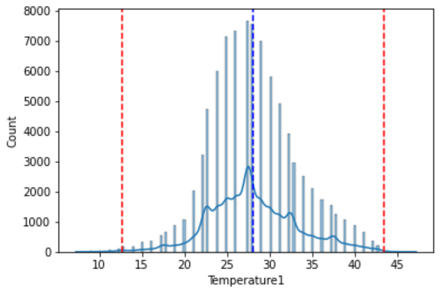


Fig a

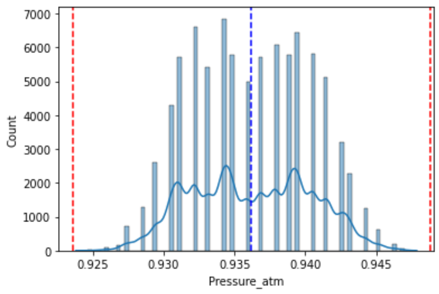


Fig b

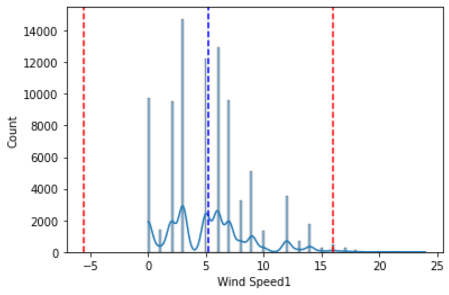


Fig c

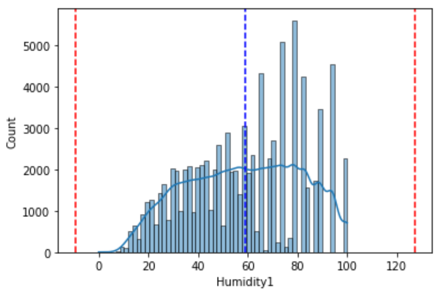


Fig d

Fig 3.2.3.1.3 Histograms after Outliers (a. Temperature, b. Pressure (in atm), c. Wind Speed(mph), d. Humidity)

The red line indicates the 3 standard deviations from the mean. The next step in data preparation is labeling the target variable, which in this case is whether or not it will rain. This is necessary for supervised learning algorithms to be trained on the data.

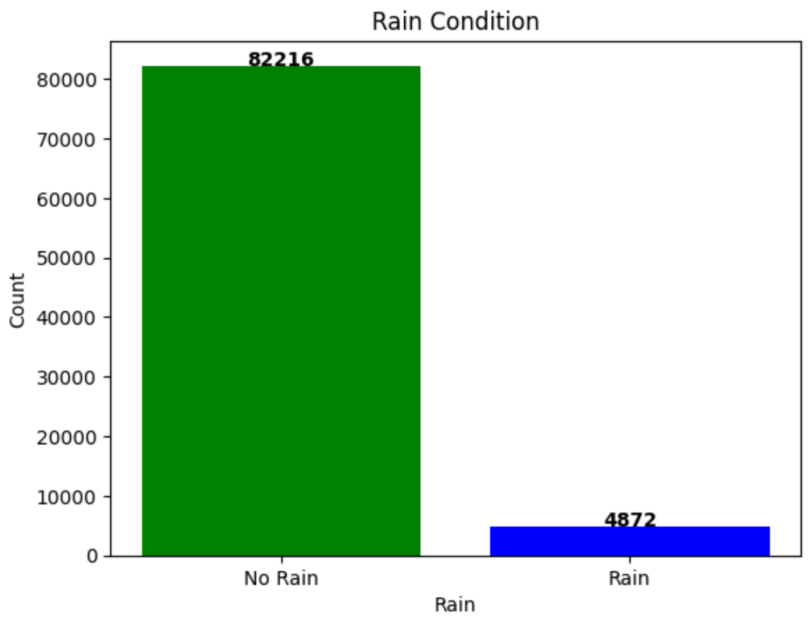


Fig 3.2.3.1.4 Count of Rain and No Rain

The data must now be divided into training and testing sets after being cleaned and labeled. In order to make sure that both the training and testing sets have a representative mix of data points, this is often done using a stratified sampling technique. The dataset needs to be balanced using oversampling techniques because it is unbalanced with regard to the goal variable (rain or no rain). To balance out the distribution of data in the dataset, this entails producing fictitious data points.

3.2.4 Logistic Regression

It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, True or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Then we set a threshold value for the classification.

- So, here we generally try to separate the data points with the line or the plane.
- The equation of the plane is $Y = M^T X + C$ (here consider M as the slope)
- The sum of product of the distance between the point & the plane and +ve and -ve data points should be minimum. We take such kind of line into consideration.

- $$P = max(\sum^{n} Y_i(M^T X_i))$$

- In the above equation Y is either +1 or -1 based on the data points. If a point is from the left side of the plane, then the distance will be +ve and the other side of the points are -ve distance.

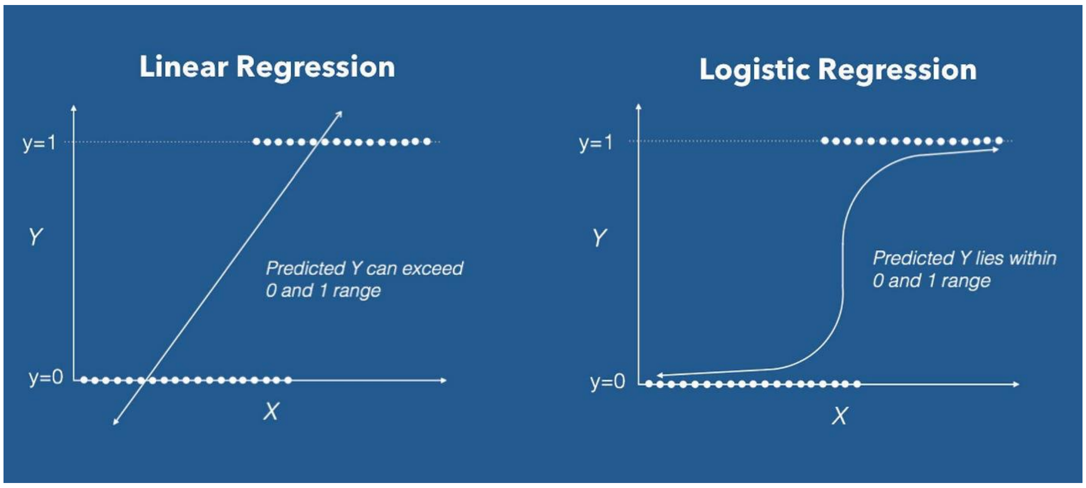
- If there is any outlier added to the dataset then the best line may be ignored unknowingly. To avoid that we use a special function after the product of the distance and Y called sigmoid function.

- $$P = max(\sum^{n} \frac{1}{1+e^{-Y_i(M^T X_i)}})$$

- We will use the sigmoid function to optimize our equation. If the distance of Xi is increased from the higher plane, then our sigmoid function squishes that distance into the value between 0 – 1. It provides probabilistic interpretations.

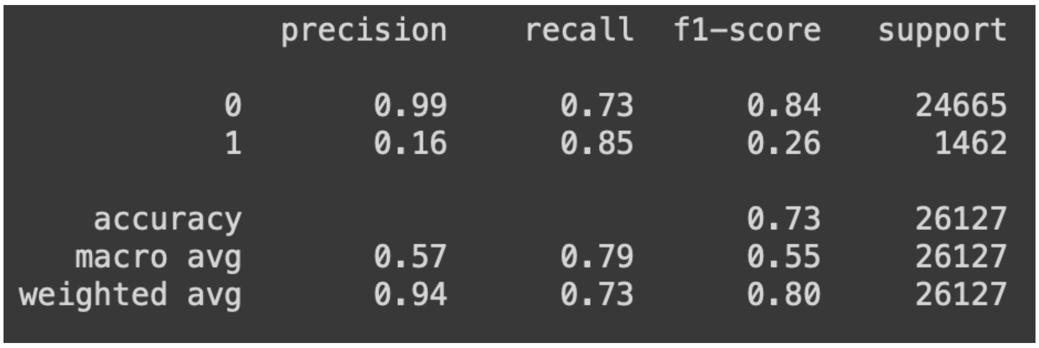
- And, if the distance of point Xi from the plane is 0 then its probability will be 0.5.

- So, this is our optimal sigmoid function which will help for preserving the optimal equation from the outlier.



Linear Regression
y=1
Y
Predicted Y can exceed 0 and 1 range
y=0
X
Logistic Regression
y=1
Y
Predicted Y lies within 0 and 1 range
y=0
X

Fig 3.2.4.1 Linear Regression and Logistic Regression Graphs



```
              precision    recall  f1-score   support

           0       0.99      0.73      0.84     24665
           1       0.16      0.85      0.26      1462

    accuracy                           0.73     26127
   macro avg       0.57      0.79      0.55     26127
weighted avg       0.94      0.73      0.80     26127
```

Fig 3.2.4.2 Metrics of Logistic Regression Model

3.2.5 Artificial Neural Network (ANN)

Artificial Neural Networks (ANN) are a type of machine learning algorithm that mimic the structure and functioning of the human brain. They comprise interconnected nodes arranged in layers that process information, with each node executing a basic mathematical calculation based on the input received.

When using an ANN, data is first inputted into the input layer, which then passes through one or more hidden layers of nodes. Each node in a hidden layer performs a mathematical operation on the input, and the resulting output is then transferred to the next layer until it arrives at the output layer, which produces the final result. The learning process in ANNs involves modifying the weights assigned to each connection between nodes. These weights govern the level of influence each node has on the output of the network, and are adjusted during the training process based on the difference between the predicted and actual output.

ANNs are widely utilized in diverse applications, including speech and image recognition, natural language processing, and prediction tasks such as weather prediction and stock market forecasting. In agriculture, ANNs are used to predict crop yields, soil moisture, and detect pests.

One of the key advantages of ANNs is their capacity to identify intricate patterns in data, even if these patterns are not explicitly defined. They also possess strong generalization abilities, enabling them to perform well on prediction tasks with new data. However, ANNs can be computationally demanding and require substantial amounts of training data to produce accurate results.

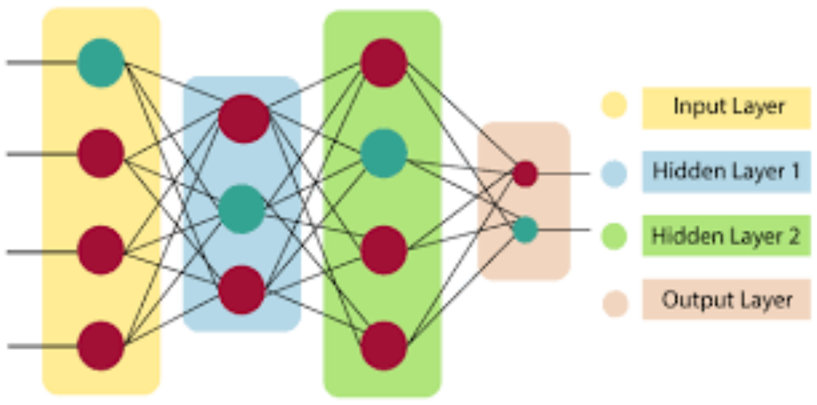
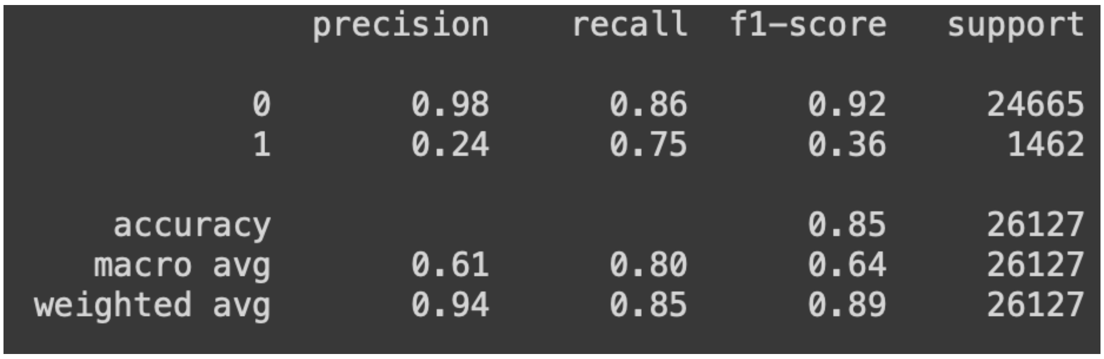


Fig 3.2.5.1 ANN Visualization



```
              precision    recall  f1-score   support

           0       0.98      0.86      0.92     24665
           1       0.24      0.75      0.36      1462

    accuracy                           0.85     26127
   macro avg       0.61      0.80      0.64     26127
weighted avg       0.94      0.85      0.89     26127
```

Fig 3.2.5.2 ANN Model Metrics

3.2.6 Random Forest Classifier

The Random Forest Classifier is a type of ensemble learning algorithm that utilizes decision trees. It operates by creating numerous decision trees during the training phase, and then selecting the mode of the classes (for classification) or mean prediction (for regression) of the individual trees as the output. In constructing each tree, a random subset of the input features and training data is employed.

The algorithm works as follows:

- Create a new dataset by randomly choosing a subset of the training data with replacement.
- Randomly select a subset of features to be employed in constructing the decision tree.
- Construct the decision tree using the chosen data and features.
- To form a decision tree forest, repeat steps 1-3.
- For classification of new data points, pass them through every decision tree in the forest and determine the majority vote of predicted class (classification) or average prediction (regression).

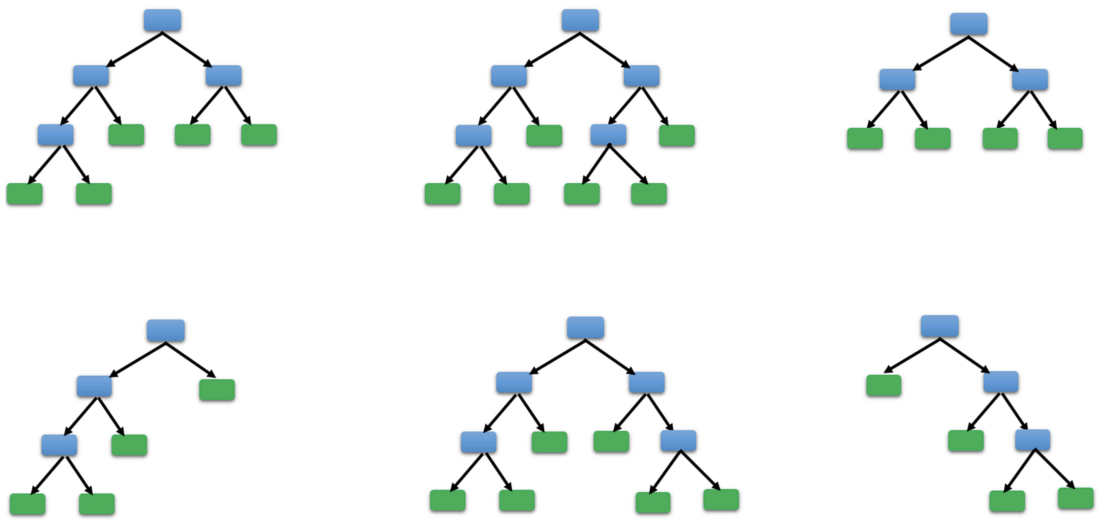
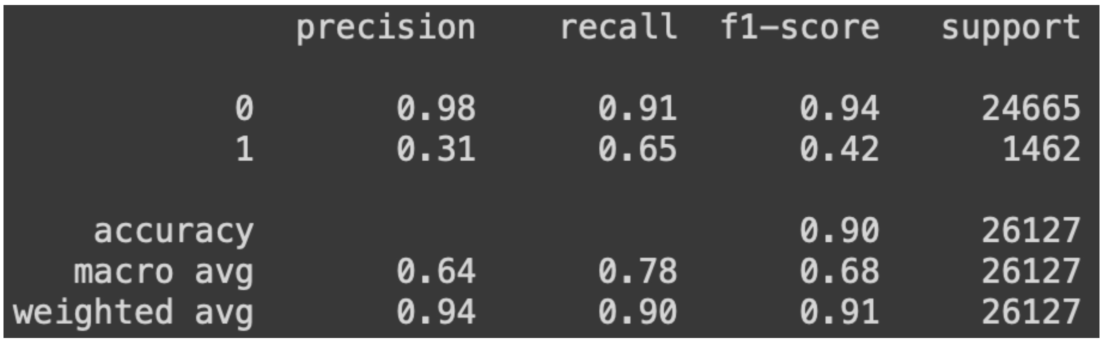


Fig 3.2.6.1 Different Decision Trees (Random Forest)

The Random Forest Classifier can be expressed mathematically as follows: Assuming a dataset D = {(x1, y1), (x2, y2), ..., (xn, yn)}, where xi is the feature vector for the i-th instance, and yi is the corresponding class label. The goal of the algorithm is to build a forest F of decision trees T that can forecast the class label of a new instance x.

1. To create each tree T in forest F
   - Randomly choose a subset of the training data D' (with replacement).
   - Randomly choose a subset of features F' to construct the decision tree T.
   - Build decision tree T using the chosen data and featues.
2. To forecast the class label of a new instance x
   - Pass x through every decision tree T in the forest F.
   - Compute the predicted class label yi for each tree T.
   - Generate the majority vote of the predicted class labels (classification) or average prediction (regression) across all trees T.



```
              precision    recall  f1-score   support

           0       0.98      0.91      0.94     24665
           1       0.31      0.65      0.42      1462

    accuracy                           0.90     26127
   macro avg       0.64      0.78      0.68     26127
weighted avg       0.94      0.90      0.91     26127
```

Fig 3.2.6.2 Random Forest Model Metrics

3.2.7 Confusion Matrix

A confusion matrix is a table that assesses the effectiveness of a machine learning model in classification problems. It's a 2x2 matrix that contrasts the predicted and actual values of a binary classifier. The matrix's four cells represent:

True Positive (TP): The number of accurate positive predictions made by the model.

False Positive (FP): The number of inaccurate positive predictions made by the model.

True Negative (TN): The number of accurate negative predictions made by the model.

False Negative (FN): The number of inaccurate negative predictions made by the model.

The matrix's rows represent the actual values (positive or negative), and the columns represent the predicted values (positive or negative).

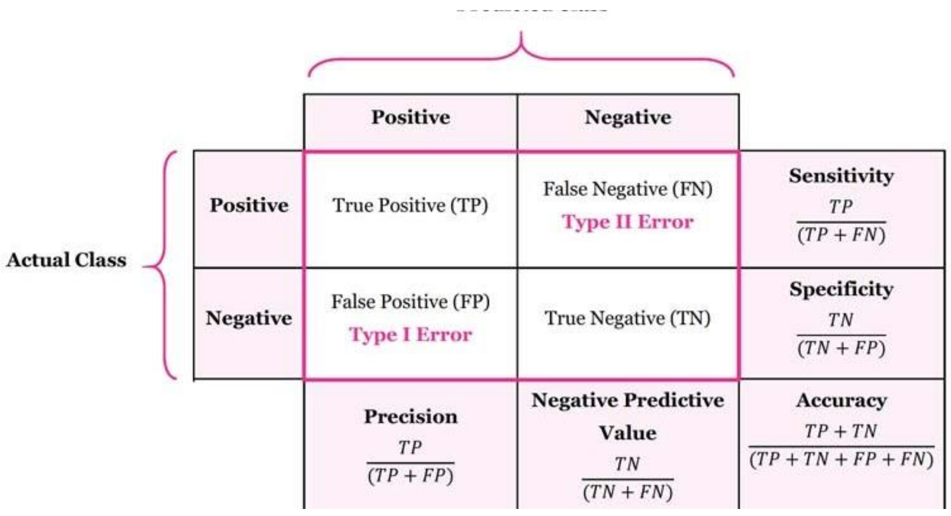


Fig 3.2.7.1 Confusion Matrix

To assess the performance of a machine learning model, various metrics such as accuracy, precision, recall, and F1 score are used. Accuracy is the ratio of correctly predicted instances to the total number of instances, and it can be calculated by dividing the sum of true positives (TP) and true negatives (TN) by the total number of predictions.

Precision is the ratio of correctly predicted positive instances to the total number of positive instances predicted by the model, which can be calculated as TP divided by the sum of TP and false positives (FP). Recall is the ratio of correctly predicted positive instances to the total number of actual positive instances, which can be calculated as TP divided by the sum of TP and false negatives (FN).

F1 score is the harmonic mean of precision and recall, which can be calculated as 2 times the product of precision and recall divided by their sum. A confusion matrix is a 2x2 matrix that compares the predicted and actual values of a binary classifier and provides information on TP, TN, FP, and FN. It can be used to evaluate the model's performance and identify errors, and to determine if the model is overfitting or underfitting.

3.2.8 Optional Algorithms

Recurrent Neural Network (RNN):

A Recurrent Neural Network (RNN) is a neural network that is specifically designed to handle sequential data. Unlike traditional neural networks, which process inputs and outputs independently, RNNs are able to capture the temporal dependencies present in sequential data by incorporating loops that allow information to persist.

Consider the example of sentiment analysis on a sequence of words. Each word in the input sequence of n words has a corresponding word embedding vector x t, and the objective is to predict the sentiment of the entire sentence. If we were to use a traditional neural network, we would simply feed the word embedding vectors x t one at a time into a feedforward network to obtain a final prediction. However, this approach does not account for the temporal dependencies between words. By contrast, an RNN can capture these dependencies by using its recurrent loops to process the entire sequence of word embeddings, thereby enabling it to make more accurate predictions about the sentiment of the sentence.

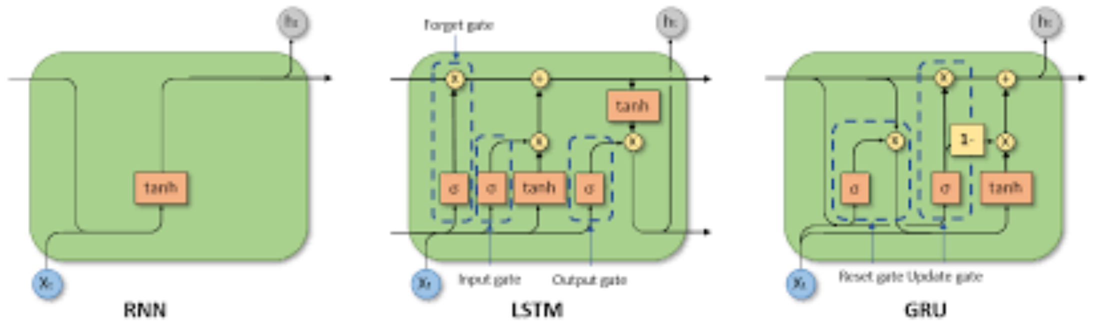


Fig 3.2.8.1 RNN LSTM GRU Working Block Diagrams

Recurrent Neural Network (RNN) is a type of neural network that is designed to process sequential data by maintaining an internal state that depends on the current input and the previous state. Unlike traditional neural networks, RNNs can capture temporal dependencies between inputs.

For example, when analyzing the sentiment of a sentence, a traditional neural network processes each word independently. However, an RNN processes each word in sequence by updating its internal state with the current input and the previous state. This allows the network to capture the contextual information from previous words and make a prediction about the sentiment of the whole sentence. RNNs are trained using a method called backpropagation through time (BPTT), which takes into account the temporal dependencies between inputs. During training, the internal state is updated at each time step, and the error is propagated backwards through time to update the weights of the network.

Overall, RNNs are useful for processing sequential data and capturing temporal dependencies. They are trained using BPTT, which allows for efficient computation of gradients over long sequences.

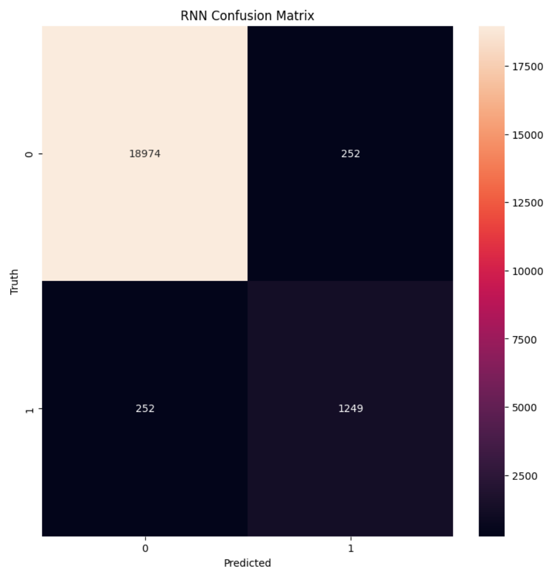
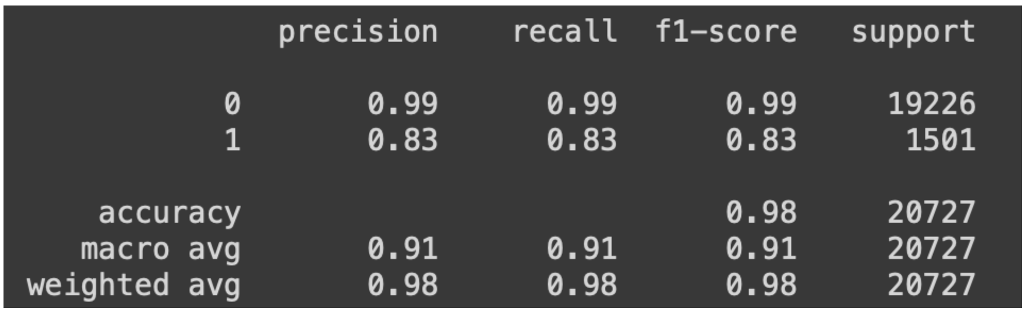


Fig 3.2.8.2 RNN Model Confusion Matrix



```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     19226
           1       0.83      0.83      0.83      1501

    accuracy                           0.98     20727
   macro avg       0.91      0.91      0.91     20727
weighted avg       0.98      0.98      0.98     20727
```

Fig 3.2.8.3 RNN Model Metrics

3.2.9 Result of ML:

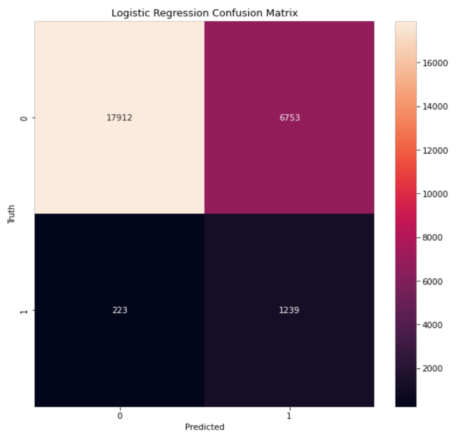


Fig 3.2.9.1 Logistic Regression Model Confusion Matrix

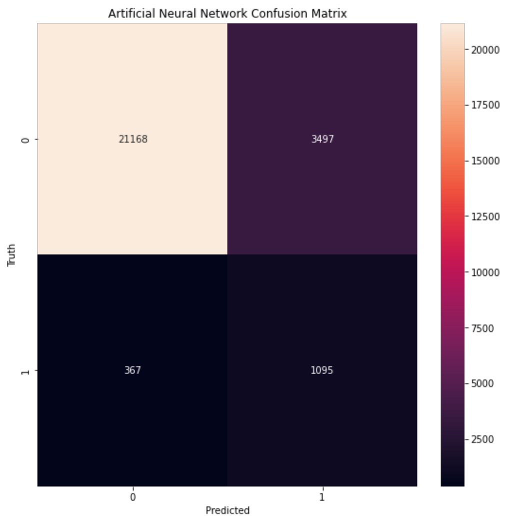


Fig 3.2.9.2 ANN Model Confusion Matrix

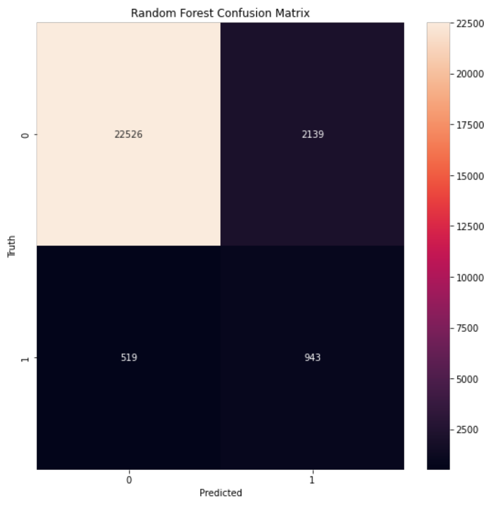
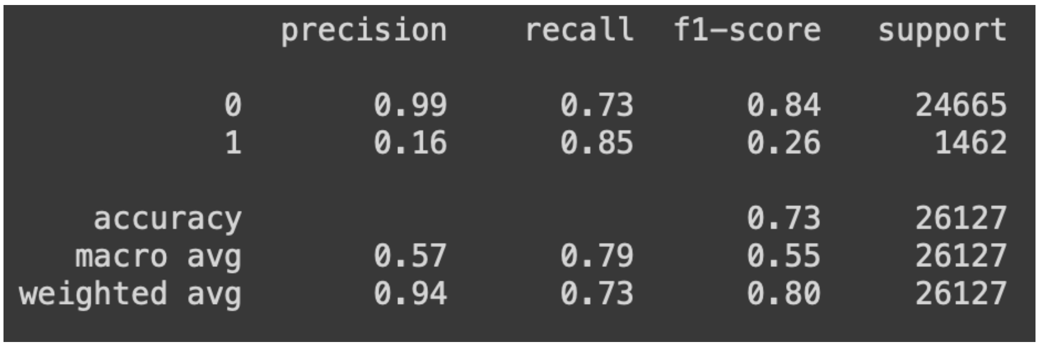


Fig 3.2.9.3 Random Forest Classifier Model Confusion Matrix

Performances of models (Accuracy):

Logistic Regression	---> 73.2%
Artificial Neural Network	---> 85.0%
Random Forest Classifier	---> 90%

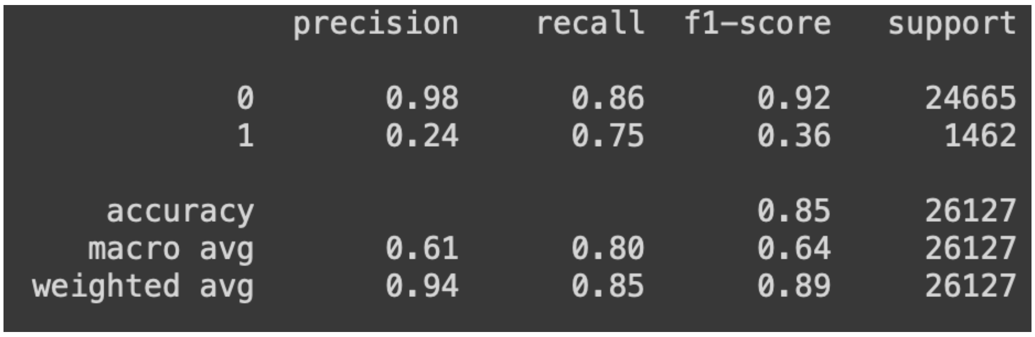


```
              precision    recall  f1-score   support

           0       0.99      0.73      0.84     24665
           1       0.16      0.85      0.26      1462

    accuracy                           0.73     26127
   macro avg       0.57      0.79      0.55     26127
weighted avg       0.94      0.73      0.80     26127
```

Fig 3.2.9.4 Metrics of Logistic Regression Model

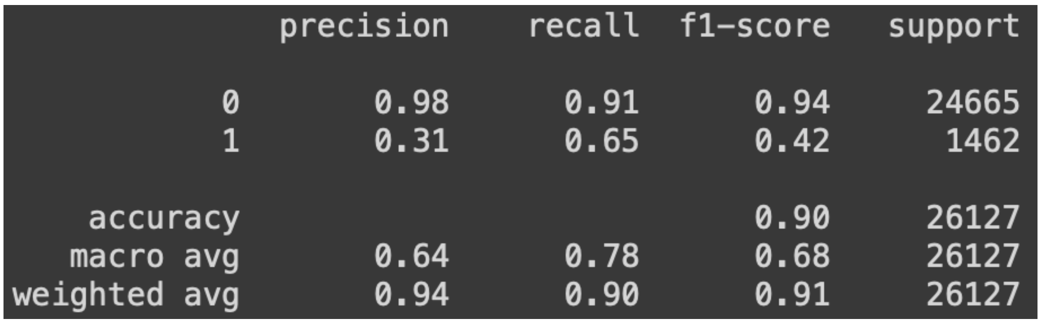


```
              precision    recall  f1-score   support

           0       0.98      0.86      0.92     24665
           1       0.24      0.75      0.36      1462

    accuracy                           0.85     26127
   macro avg       0.61      0.80      0.64     26127
weighted avg       0.94      0.85      0.89     26127
```

Fig 3.2.9.5 ANN Model Metrics



```
              precision    recall  f1-score   support

           0       0.98      0.91      0.94     24665
           1       0.31      0.65      0.42      1462

    accuracy                           0.90     26127
   macro avg       0.64      0.78      0.68     26127
weighted avg       0.94      0.90      0.91     26127
```

Fig 3.2.9.6 Random Forest Model Metrics

3.3 Platform Used

3.3.1 Anaconda Jupyter

Anaconda Navigator is a graphical user interface (GUI) tool that comes bundled with the Anaconda distribution. It provides a convenient way to set up, manage, and launch various tools, including Jupyter Notebook. Meanwhile, a Conda Python environment creates a separate and isolated environment for Python, which enables users to install packages without affecting their system's Python installation.

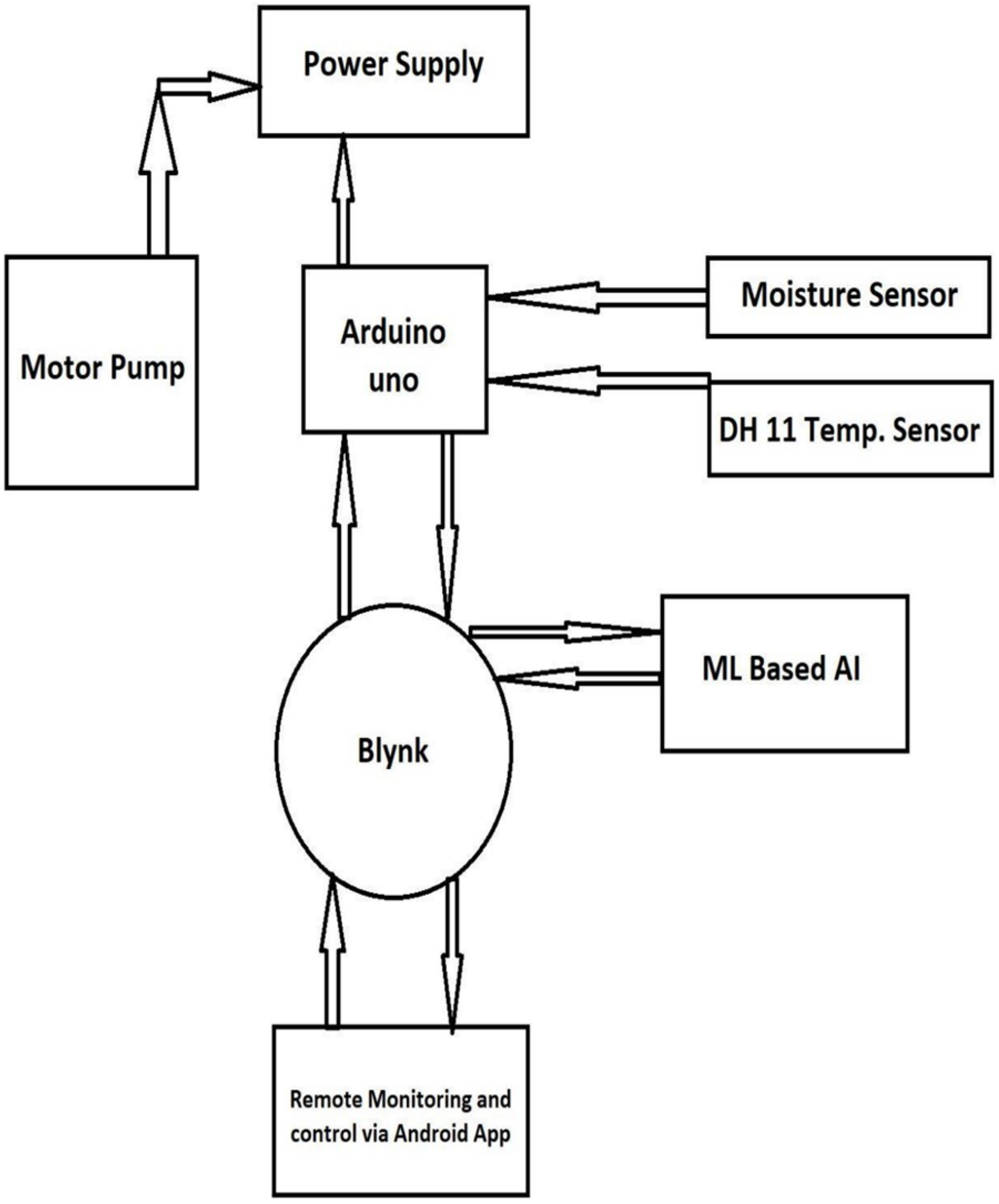
3.3.2 Google Colaboratory

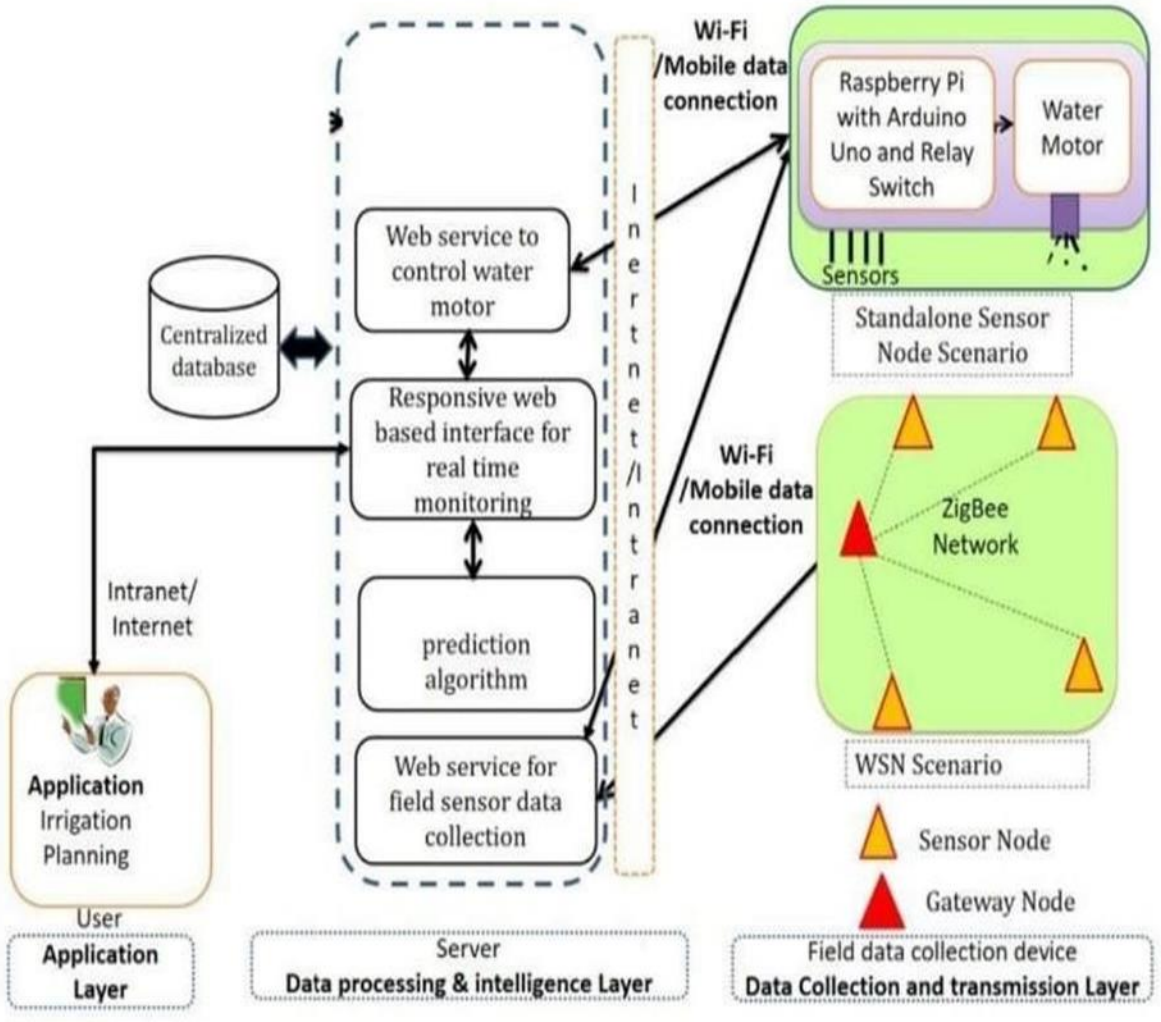
Google Colaboratory is a cloud-based Jupyter notebook environment that provides free access to powerful backend hardware such as GPUs and TPUs. This allows users to perform all the tasks that can be done in a locally hosted Jupyter notebook, without the need for local installations or setups. Since Google Colaboratory is hosted on cloud servers, users can easily share their notebooks with others and collaborate in real-time.

3.3.3 PyCharm CE

PyCharm is an Integrated Development Environment (IDE) specifically designed for Python development, offering a comprehensive set of tools and features that facilitate productive Python, web, and data science development in a convenient and tightly integrated environment.

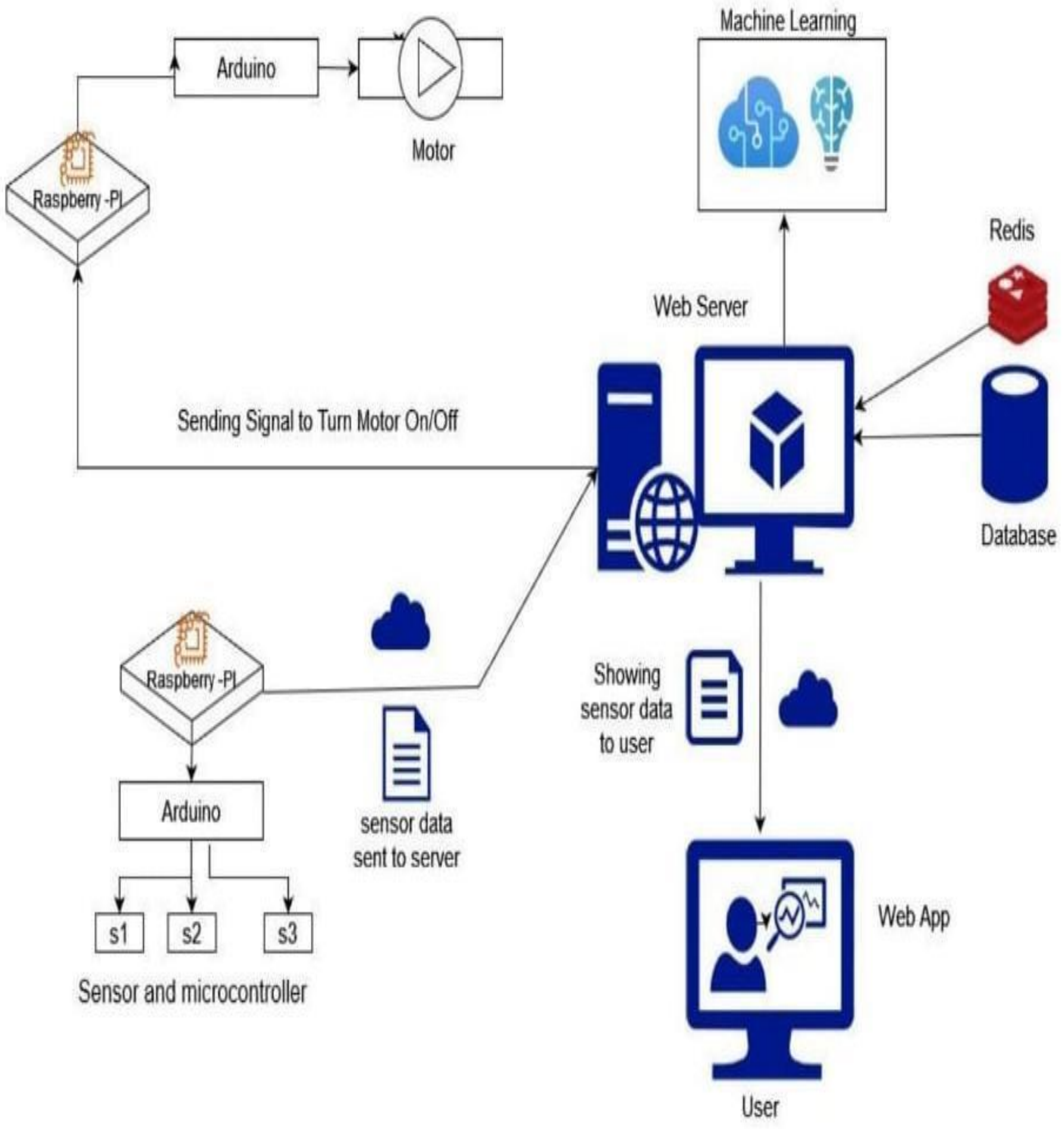
3.4 FLOWCHART OF MODEL METHODOLOGY:





Wi-Fi /Mobile data connection
Raspberry Pi with Arduino Uno and Relay Switch
Water Motor
Sensors
Standalone Sensor Node Scenario
Web service to control water motor
Centralized database
Responsive web based interface for real time monitoring
Internet/Intranet
Wi-Fi /Mobile data connection
ZigBee Network
Intranet/ Internet
prediction algorithm
WSN Scenario
Web service for field sensor data collection
Application Irrigation Planning
Sensor Node
Gateway Node
User Application Layer
Server Data processing & intelligence Layer
Field data collection device Data Collection and transmission Layer

3.5 FLOWCHART OF THE PROCESS



Arduino
Motor
Raspberry -PI
Machine Learning
Redis
Web Server
Sending Signal to Turn Motor On/Off
Database
Raspberry -PI
Showing sensor data to user
Arduino
sensor data sent to server
s1
s2
s3
Web App
Sensor and microcontroller
User

Fig 3.5.1 FLOWCHART OF THE PROCESS

3.6 ARCHITECTURE OF THE MODEL:

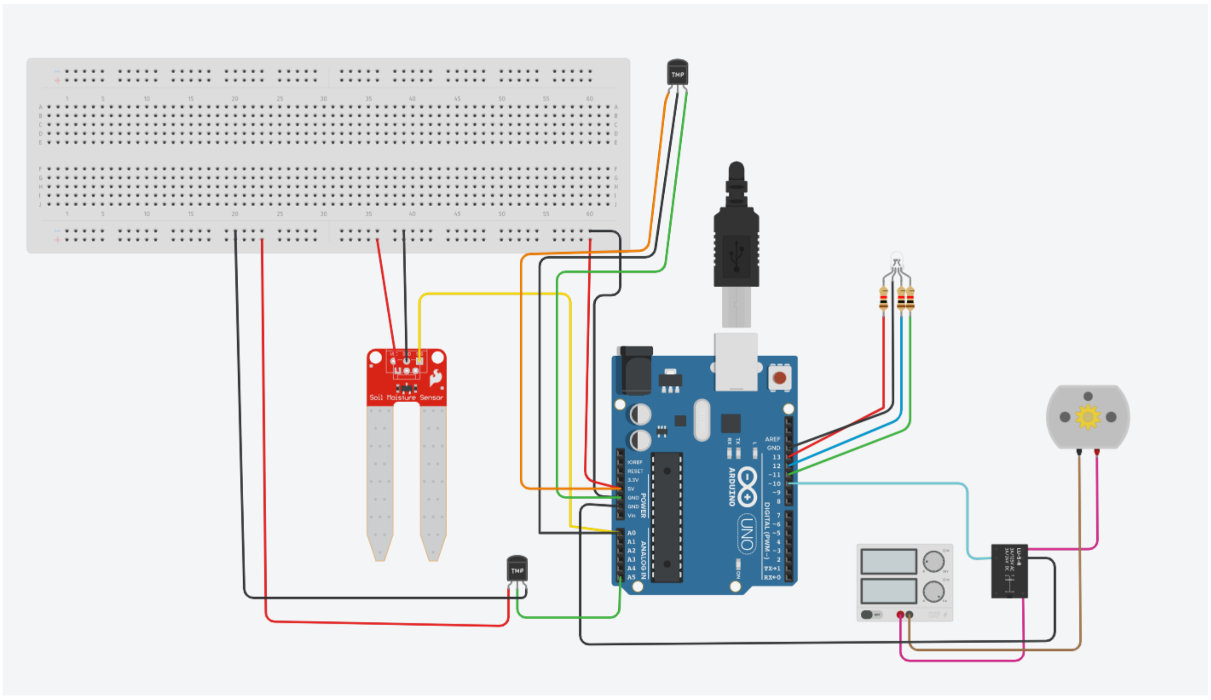


Fig 3.6.1 ARCHITECTURE OF THE MODEL

3.7 BLYNK WEB USER INTERFACE:

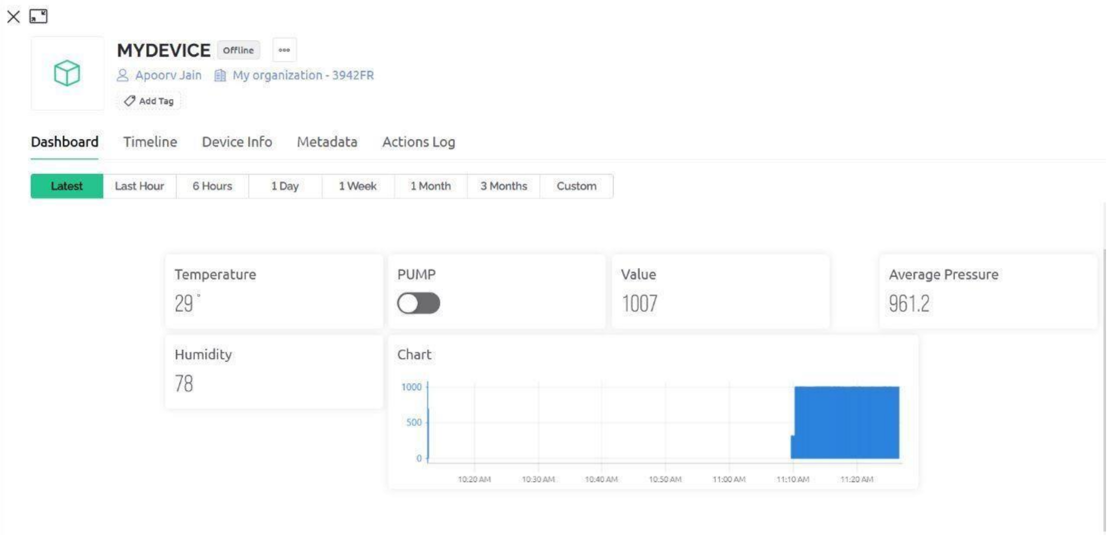


Fig 3.7.1 BLYNK WEB USER INTERFACE:

3.8 ANDROID APP INTERFACE

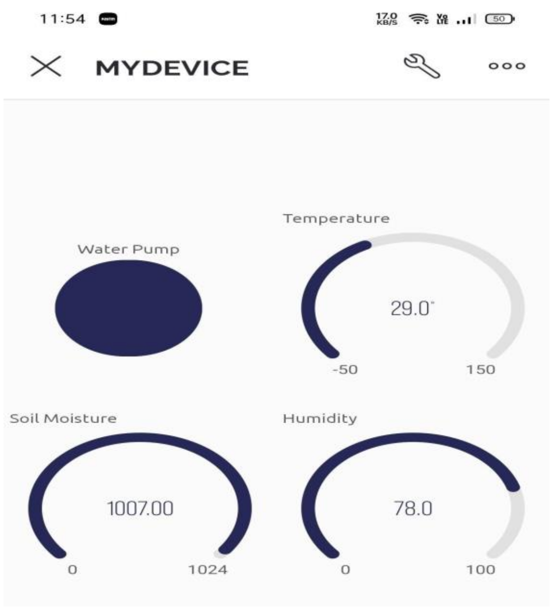
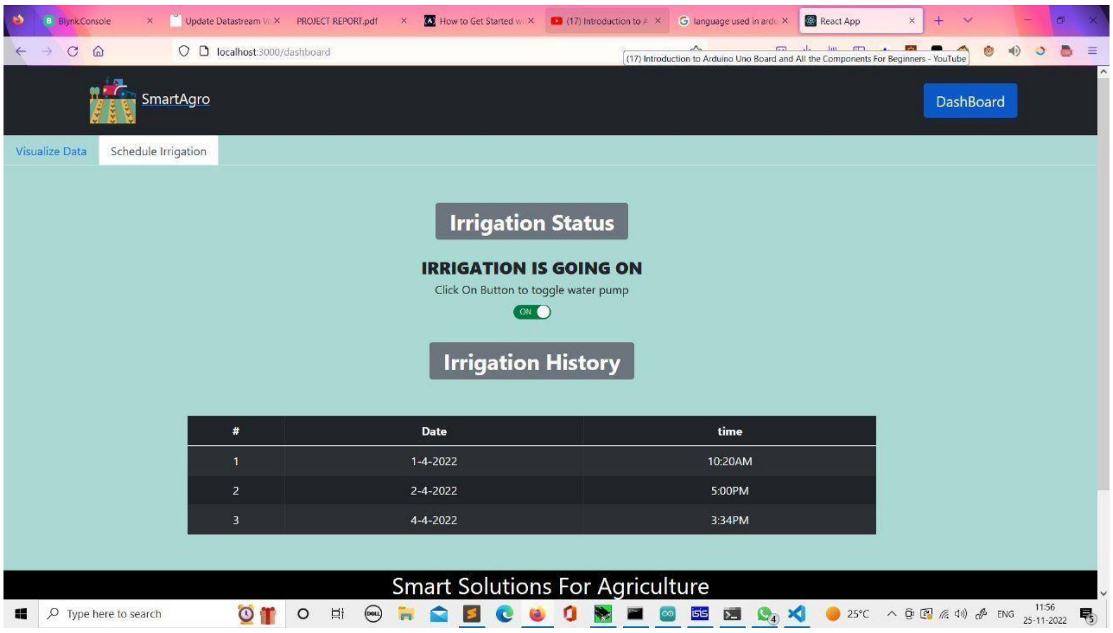


Fig 3.8.1 ANDROID APP INTERFACE

3.9 USER WEB INTERFACE



Irrigation Status

IRRIGATION IS GOING ON

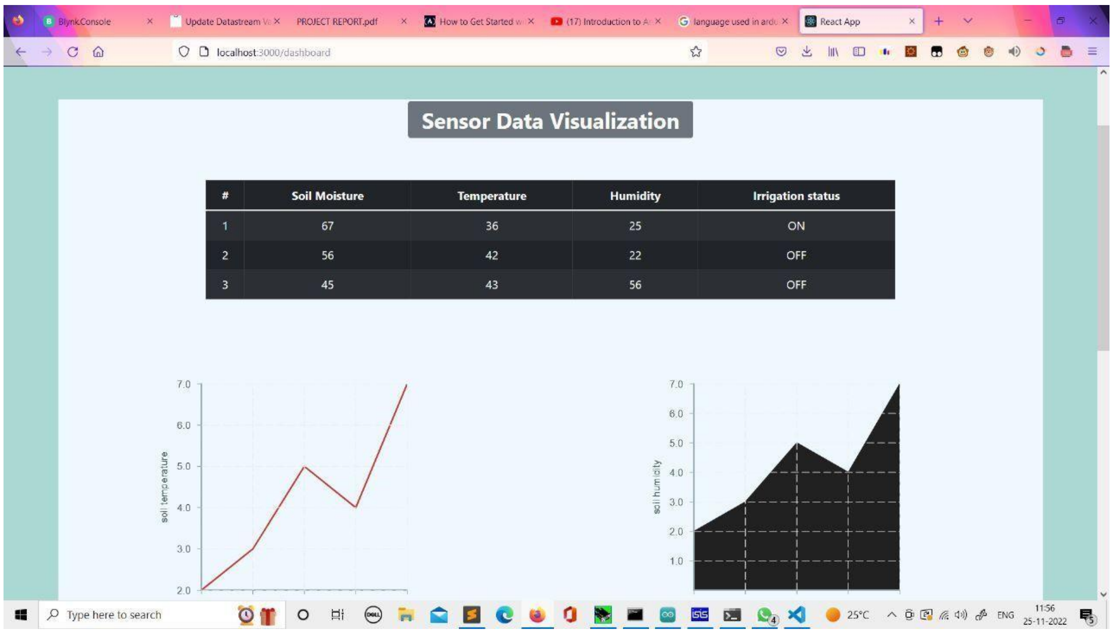
Click On Button to toggle water pump

Irrigation History

#	Date	time
1	1-4-2022	10:20AM
2	2-4-2022	5:00PM
3	4-4-2022	3:34PM

Smart Solutions For Agriculture

Fig 3.9.1 Irrigation Status



Sensor Data Visualization

#	Soil Moisture	Temperature	Humidity	Irrigation status
1	67	36	25	ON
2	56	42	22	OFF
3	45	43	56	OFF

Fig 3.9.2 Sensor Data Visualization

CHAPTER 4

4. RESULTS

The proposed system is an intelligent irrigation solution based on artificial intelligence that utilizes the soil moisture content and crop moisture requirements to automate the irrigation process. The main advantage of the system is its efficiency and economic feasibility. The main idea behind the proposed rainfall estimation is to obtain accurate rainfall estimates for a specific local region and annual rainfall data to aid in future estimation of rainfall in different states.

When the soil moisture level falls below the required threshold value and there is no rainfall predicted by the machine learning algorithm, the model activates the motor pump for crop irrigation.

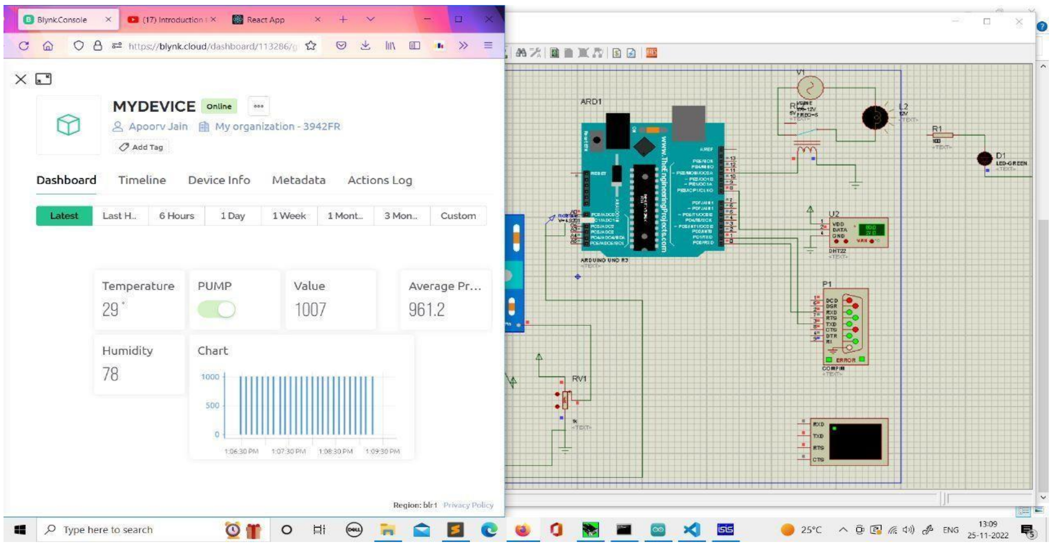


Fig 4.1 Blynk Web User Interface

When irrigation is needed, the circuit is on as we can see LED is on.

CHAPTER 5

5. CONCLUSION

Our team has created an autonomous irrigation system that incorporates Artificial Intelligence learning and predictive algorithms to enhance the capabilities of existing automatic irrigation systems. By leveraging these technologies, our system is capable of making informed decisions and adjusting its operations based on changing environmental conditions.

The methods outlined in this paper have the potential to significantly improve irrigation efficiency while reducing the amount of effort required and conserving water resources compared to traditional irrigation methods. Our system is currently reliant on weather station data to make its calculations. However, we recognize that weather stations may not be readily available in certain regions, particularly in rural areas of the Indian subcontinent and arid regions where water resources are scarce.

To address this issue, we propose the deployment of on-premise sensors that can provide accurate and timely information to the irrigation system. This approach would enable the system to make informed decisions and adjust its operations based on localized conditions, even in areas where weather station data is not available.

Overall, our autonomous irrigation system represents a significant advancement in the field of irrigation technology. By leveraging the power of Artificial Intelligence and predictive algorithms, we have developed a highly effective solution that has the potential to improve crop yields and conserve water resources. We are confident that this technology will prove to be a valuable asset in supporting sustainable agricultural practices in regions where water is scarce and traditional irrigation methods are no longer sufficient.

6. REFERENCES

[1] Edordu C. and Sacks L., "Self Organizing Wireless Sensor Networks as a Land Management Tool in Developing Countries: A Preliminary Survey," In Proceedings of the 2006 London Communications Symposium, September 2006, Communications Engineering Doctorate Centre, London, UK.

[2] Kim Y., Evans R.G. and Iversen W.M., "Remote Sensing and Control of an Irrigation System Using a Distributed Wireless Sensor Network," Instrumentation and Measurement, IEEE Transactions on, vol.57, no.7, pp.1379-1387, July 2008. http://dx.doi.org/10.1109/TIM.2008.917198

[3] Joaquín G, Juan F , Alejandra N.G, and Miguel Ángel, "Automated Irrigation System Using a Wireless Sensor Network and GPRS Module", IEEE Transactions On Instrumentation and Measurement, Vol.63, no.1, pp.166-176, 2013

[4] Karandeep K, "Machine Learning : Applications in Indian Agriculture", International Journal of Advanced Research in Computer and Communication Engineering, Vol.5, no.4, pp.342- 344, 2016.

[5] Washington O and Joseph O, "Machine Learning Classification Technique for Famine Prediction". Proceedings of the World Congress on Engineering 2011 Vol II WCE 2011, July 6 - 8, London, U.K, 2011.

[6] Snehal S, Sandeep V.R, "Agricultural Crop Yield Prediction Using Artificial Neural NetworkApproach". International Journal of Innovative Research in Electrical, Electronic, Instrumentation and Control Engineering, Vol. 2, Issue 1, January 2014.

[7] Kumar R, Singh M .P, Prabhat K, and Singh J.P. "Crop Selection Method to Maximize Crop Yield Rate Using Machine Learning Technique." Proceedings of International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM) ,2015.

[8] Rumpf, T., A.-K. Mahlein, U. Steiner, E.-C. Oerke, H.-W. Dehne, and L. Plümer. "Early Detection and Classification of Plant Diseases with Support Vector Machines Based on Hyperspectral Reflectance." Computers and Electronics in Agriculture, Vol. 74, no.1, pp.91-99, 2010.

[9] Carles Antón-Haro, Thierry Lestable, Yonghua Lin, Navid Nikaein, Thomas Watteyne, Jesus AlonsoZarate, “Machine-to-machine: An emerging communication paradigm. Trans. on Emerging Telecommunications Technologies”, Volume 24, Issue 4, pages 353–354, June 2013. DOI: http://dx.doi.org/10.1002/ett.2668

[10] Koushik Anand, Jayakumar C, Mohana Muthu and Sridhar A, “Automatic Drip Irrigation using Fuzzy Logic and Mobile Technology”, Proceedings of Technological Innovation in ICT for Agriculture and Rural Development, 2015.

[11] Kait L.K., Kai C.Z., Khoshdelniat R., Lim S.M., and Tat E.H., “Paddy Growth Monitoring with Wireless Sensor Networks,” Proceedings of Intelligent and Advanced Systems, Kuala Lumpur, Malaysia, pp.966- 970, 2007. http://dx.doi.org/10.1109/ICIAS.2007.4658529

[12] Kim Y., Evans R.G. and Iversen W.M., “Remote Sensing and Control of an Irrigation System Using a Distributed Wireless Sensor Network,” Instrumentation and Measurement, IEEE Transactions on, vol.57, no.7, pp.1379-1387, July 2008. http://dx.doi.org/10.1109/TIM.2008.917198

[13] Wall R.W. and King B.A., “Incorporating plug and play technology into measurement and control systems for irrigation management”, presented at the ASAE/CSAE Annu. Int. Meeting, Ottawa, ON, Canada, Aug. 2004

[14] Yang W., Liusheng H., Junmin W. and Hongli X., “Wireless Sensor Networks for Intensive Irrigated Agriculture,” Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE, pp.197- 201, Las Vegas, Nevada, Jan. 2007. http://dx.doi.org/10.1109/CCNC.2007.46

[15] Konstantinos K., Apostolos X., Panagiotis K. and George S., “Topology Optimization in Wireless Sensor Networks for Precision Agriculture Applications,” Sensor Technologies and Applications, 2007. SensorComm 2007. International Conference on, pp.526-530, Valencia, Spain, 14-20 Oct. 2007. http://dx.doi.org/10.1109/SENSORCOMM.2007.101

[16] Masuki, K. F. Ga, Kamugisha, Rb, Mowo, J. Gc, Tanui, Jc, Tukahirwa, Ja. Mogoi, Jc. and Adera E.O, “Role of mobile phones in improving communication and information delivery for agricultural development”, ICT and Development - Research Voices from Africa. International Federation for Information Processing (IFIP),

Technical Commission 9 – Relationship Between Computers and Society. Workshop at Makerere University, Uganda. 22-23 March 2010

[17] Suyash S P and Sandeep A T, "Early Detection of Grapes Disease using Machine Learning and IoT", Proceedings of Second International Conference on Cognitive Computing and Information Processing, Mysore, India, 2016

[18] Yue L, Long M and Ooi S K, "Prediction of Soil Moisture based on Extreme Learning Machine for an Apple Orchard", Proceedings of 3rd IEEE International Conference on Cloud Computing and Intelligent System", Shenzhen, Hong Kong, 2014.

[19] Biswas S, Saunshi A, Sarangi S and Pappula S, "Random Forest based Classification of Diseases in Grapes from Images Captured in Uncontrolled Environment", Proceedings of IET International Conference on Signal Processing, 2016.

[20] Rakesh K, Singh M.P, Prabhat K and Singh J.P, "Crop Selection Method to maximize Crop Yield Rate using Machine Learning Technique", Proceedings of International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials, Chennai, India, 2015.

[21] Intelligent Irrigation System Using Artificial Intelligence and Machine Learning: A Comprehensive Review, DOI:10.21474/IJAR01/7959

[22] Intelligent Irrigation System Using Artificial Intelligence and Machine Learning: A Comprehensive Review, DOI:10.21474/IJAR01/795

[23] Intelligent Irrigation System Using Artificial Intelligence and Machine Learning: A Comprehensive Review, DOI:10.21474/IJAR01/7959

[24] Intelligent Irrigation System Using Artificial Intelligence and Machine Learning: A Comprehensive Review, DOI:10.21474/IJAR01/7959

[25] Joaquín G, Juan F , Alejandra N.G, and Miguel Ángel, "Automated Irrigation System Using a Wireless Sensor Network and GPRS Module", IEEE Transactions On Instrumentation and Measurement, Vol.63, no.1, pp.166-176,2013

[26] Kim Y., Evans R.G. and Iversen W.M., "Remote Sensing and Control of an Irrigation System Using a Distributed Wireless Sensor Network," Instrumentation and Measurement, IEEE Transactions on, vol.57, no.7

[27] Arduino Based Machine Learning and IoT Smart Irrigation System. DOI:10.35940/ijsce.D3481.031042

7. Source Code:

Arduino code:

```
#include <DHT.h>
/* Fill-in your Template ID (only if using Blynk.Cloud) */
#define BLYNK_TEMPLATE_ID "TMPLYXtOg0jT"
#define BLYNK_DEVICE_NAME "BTPPROJECT"
#define BLYNK_AUTH_TOKEN
"lIBPUqiLkEt3R0kDt0cMn3p5X5suse_k"
// You could use a spare Hardware Serial on boards that have it
(like Mega)
#include <SoftwareSerial.h>
//SoftwareSerial DebugSerial(2, 3); // RX, TX
#include <BlynkSimpleStream.h>
#define PUMP 11
#define SENSOR V1
// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = BLYNK_AUTH_TOKEN;
float soilMoisture;
float temperature;
float humidity;
int var;
int cnt=0;
// Blynk Timer timer;
void syncValues()
{
humidity=80+random(6)-3;
temperature=27+random(6)-3;
}
BLYNK_WRITE(V0){
var=param.asInt();
if(var==0)
{
digitalWrite(PUMP,LOW);
}
else
{
digitalWrite(PUMP,HIGH);
}
}
void myTimer()
{
```

```
Serial.println(soilMoisture);
Blynk.virtualWrite(V1, soilMoisture);
}
#define DHTPIN 8 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
void setup()
{
cnt=0;
analogReference(DEFAULT);
//DebugSerial.begin(9600);
pinMode(PUMP,OUTPUT);
Serial.begin(9600);
Blynk.begin(Serial, auth);
dht.begin();
//timer.setInterval(1000L, myTimer);
}
void loop() {
soilMoisture = analogRead(A0);
humidity = dht.readHumidity();
temperature = dht.readTemperature();
Blynk.run();
// Serial.println(soilMoisture);
// Serial.println(temperature);
// Serial.println(humidity);
syncValues();
Blynk.virtualWrite(V1, soilMoisture);
//Blynk.virtualWrite(V2, temperature);
// Blynk.virtualWrite(V3, humidity);
cnt++;
delay(1000);
//timer.run(); }
```

Backend code:

```
# importing the requests library
import requests
import time
import pickle;
token="lIBPUqiLkEt3R0kDt0cMn3p5X5suse_k"
pumpPin="v0";
soilPin="v1"
temperaturePin="v2"
humidityPin="v3";
weather_api="http://api.weatherbit.io/v2.0/forecast/agweather?lat=23.27
56&lon=77.4560&key=c0cebec5f6594632acdfc4b83347014f9";
loaded_model=pickle.load(open("finalized_model.sav", 'rb'));
def
```

```
required_soilMoisture(temperature,humidity,avg_pres,evapotranspiration
):
return
loaded_model.predict([[temperature,humidity,avg_pres,evapotranspiratio
n]])[0][0]+50;
def isThePumpNeedsToBeOn(curSoilMoisture,requiredSoilMoisture):
if(curSoilMoisture/10>requiredSoilMoisture):
return False
else:
return True
while True:
URL =
"https://blynk.cloud/external/api/get?token="+token+"&"+pumpPin+
"&"+soilPin+"&"+temperaturePin+"&"+humidityPin;
URLupdate="https://blynk.cloud/external/api/update?token="+toke
n+"&"
# sending get request and saving the response as response object
r = requests.get(url = URL)
u=requests.get(url=weather_api);
# extracting data in json format
data = r.json()
data2=u.json();
evapotranspiration=data2['data'][0]['evapotranspiration'];
avg_pres=data2['data'][0]['pres_avg'];
print(avg_pres);
requests.get(url=URLupdate+"V4="+str(avg_pres));
requests.get(url=URLupdate+"V5="+str(evapotranspiration));
print(data);
print(required_soilMoisture(data['v2'], data['v3'], avg_pres,
evapotranspiration))
time.sleep(10);
```

For Web Scraping:

```
from datetime import date, timedelta
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from bs4 import BeautifulSoup
import csv
# Initialize the Chrome driver
driver = webdriver.Chrome()
# Define the start and end date range
start_date = date(2023, 1, 14)
end_date = date(2023, 1, 14)
# Define the location and URL template
```

```
location = "secunderabad"
url_template =
"https://www.wunderground.com/history/daily/in/{location}/VOHY/date/{year}-
{month}-{day}"
bool_ = True
while start_date <= end_date:
   k, n = 1, 1
   try:
      # Create an empty list to store the table data
      data = []
      # Loop through the dates
      delta = timedelta(days=1)
      while start_date <= end_date:
         # Format the date in the required format
         date_str = start_date.strftime("%Y-%m-%d")
         year_str = start_date.strftime("%Y")
         month_str = start_date.strftime("%m")
         day_str = start_date.strftime("%d")
         # Construct the URL
         url = url_template.format(location=location, year=year_str, month=month_str,
day=day_str)
         # Navigate to the URL
         driver.get(url)
         # Wait for the table to load
         wait = WebDriverWait(driver, 15)
         table = wait.until(
            EC.presence_of_element_located((By.CSS_SELECTOR, ".mat-table.cdk-
table.mat-sort.ng-star-inserted")))
         # Parse the web page content as HTML using BeautifulSoup
         soup = BeautifulSoup(driver.page_source, "html.parser")

         # Find the table element by its tag and class
         table = soup.find("table", class_="mat-table cdk-table mat-sort ng-star-
inserted")
         # Loop through the table rows
         for row in table.find_all("tr"):
            # Create an empty list to store the row data
            row_data = []
            # Loop through the row cells
            for cell in row.find_all("td"):
               # Append the cell text to the row data list
               row_data.append(cell.text.strip())
            # Append the row data list to the table data list
            if len(row_data) > 0:
               # Add the date to the row data list
               row_data.insert(0, date_str)
```

```
                data.append(row_data)
            # Increment the date by 1 day
            start_date += delta
    except:
        # Save the data to a CSV file
        with open("data_web_scrap.csv", "a", newline="") as csvfile:
            writer = csv.writer(csvfile)
            # Write the header row
            if bool_:
                writer.writerow(
                    ["Date", "Time", "Temperature", "Dew Point", "Humidity", "Wind",
"Wind Speed", "Wind Gust", "Pressure",
                     "Precipitation", "Condition"])
                bool_ = False
            # Write the data rows
            writer.writerows(data)
            k += 1
    if k == n:
        # Save the data to a CSV file
        with open("data_web_scrap.csv", "a", newline="") as csvfile:
            writer = csv.writer(csv file)
            # Write the header row
            if bool_:
                writer.writerow(
                    ["Date", "Time", "Temperature", "Dew Point", "Humidity", "Wind",
"Wind Speed", "Wind Gust",
                     "Pressure",
                     "Precipitation", "Condition"])
                bool_ = False
            # Write the data rows
            writer.writerows(data)
        start_date += delta
# Close the Chrome driver
driver.quit()
```

For ML Part:
RAINFALL PREDICTION

```
# ** Importing all Required Libraries**

import pandas as pd
import matplotlib.pyplot as pt
import matplotlib
import seaborn as sb
import numpy as np
import tensorflow as tf
import pickle
```

```
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import Normalizer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
get_ipython().run_line_magic('matplotlib', 'inline')

df = pd.read_csv('/content/data_web_scrap_hyderabad_12_23_new2.csv')
df.head()
df.shape
df.Temperature.unique()

df[df.Temperature == 'Temperature']

df1 = df[df.Date != 'Date']

df1['Temperature1'] = df1['Temperature'].apply(lambda x: float(x[0:-2]))
df1.head()
 **Changing Fahrenheit (°F) to Celsius (°C)**
df1['Temperature 1'] = 5/9*(df1['Temperature 1'] - 32)
df1.head()

# For wind check link
# https://www.weather.gov/pqr/wind

df1.isnull().sum()

sb.heatmap(df1.isnull(), yticklabels = False, cbar = False, cmap = 'viridis')
null_rows = df1[df1.isna().any(axis = 1)]
def wind(x):
   try:
      return float(x[0:-4])
   except:
      a = x[0:-4].replace(',', '')
      return float(a)

df1['Wind Speed1'] = df1['Wind Speed'].apply(wind)

Wind_Speed1 = df1['Wind Speed1'].tolist()

l = []
for i in Wind_Speed1:
   if i < 30:
    l.append(i)

sb.histplot(x = l, kde = True)
sb.boxplot(Wind_Speed1)

df1['Pressure1'] = df1['Pressure'].apply(lambda x: float(x[0:-3]))
df1.head()

df1['Humidity1'] = df1['Humidity'].apply(lambda x: int(x[0:-2]))

df1.corr()

df1.groupby('Condition')['Condition'].agg('count')

df2 = df1.copy()

df2.reset_index(level = 0, inplace = True)

df2.rename(columns = {'index':'Index'}, inplace = True)

Cond_df = df2.groupby('Condition')['Index'].agg('count').reset_index()
Cond_df
# Rain ---> 1
# Not Rain ---> 0

condi = Cond_df.Condition.to_list()
condi

df2['Rain_01'] = df2['Condition'].replace(['Cloudy', 'Cloudy / Windy', 'Fair', 'Fair /
Windy', 'Fog', 'Fog / Windy',
                              'Haze', 'Haze / Windy', 'Mist', 'Mostly Cloudy', 'Mostly
Cloudy / Windy',
                              'Partly Cloudy', 'Partly Cloudy / Windy', 'Smoke',
'Widespread Dust',
                              'Widespread Dust / Windy'], 0)

df2['Rain_01'] = df2['Rain_01'].replace(['Light Drizzle', 'Thunder', 'Light Rain with
Thunder', 'Rain',
     'Light Rain', 'Showers in the Vicinity', 'T-Storm', 'Drizzle',
     'Thunder / Windy', 'Rain / Windy', 'T-Storm / Windy',
     'Heavy T-Storm', 'Light Rain Shower', 'Heavy Rain', 'Rain Shower',
     'Drizzle / Windy', 'Drizzle and Fog', 'Light Rain / Windy',
     'Light Drizzle / Windy'], 1)

df2.Rain_01.unique()
# **inHg to atm**
# 1 atm = 29.92 inHg ----> inHg is inches in Hg(Mercury)

df2['Pressure_atm'] = df2['Pressure1']/29.92
# # Features units
# Temperature is in C
# Wind Speed is in mph(Meter per Hour)
# Pressure_atm is in atmospheric pressure
# 1 atm = 29.92 inHg ----> inHg is inches in Hg(Mercury)
# Humidity is in percentage
# Rain:
#     0 ---> No Rain
#     1 ---> Raining

df2.groupby('Rain_01')['Index'].agg('count').reset_index()
df2.dropna(inplace = True)

df2.describe()

def graph(df, clm):
    # Generate some random data that follows a normal distribution
    data = df[clm]

    # Create a histogram of the data
    sb.histplot(data, kde=True)

    # Calculate the mean and standard deviation of the data
    mean = np.mean(data)
    std_dev = np.std(data)

    left_bound = mean - (3 * std_dev)
    right_bound = mean + (3 * std_dev)


    pt.axvline(x=left_bound, color='r', linestyle='--')
    pt.axvline(x = mean, color = 'b', linestyle = '--')
    pt.axvline(x=right_bound, color='r', linestyle='--')

    # Show the plot
    pt.show()

graph(df2, 'Pressure_atm')


# **We can see there are outliers**

pt.scatter(x = df2.Pressure_atm, y = df2.Rain_01)
```

```
df3 = df2.copy()

# **Outliers Removal using IQR**

def removal_outliers(df, clm):
    q1, q3 = np.percentile(df[clm], [25, 75])
    iqr = q3 - q1
    low = q1 - 1.5*iqr
    upp = q3 + 1.5*iqr
    lst = df[clm]
    for i in lst:
        if i < low or i > upp:
            df = df[df[clm] != i]
    return df

df4 = removal_outliers(df3, 'Pressure_atm')

pt.scatter(x = df4.Pressure_atm, y = df4.Rain_01)
 # **Visualizing the Distribution**
# Here the visualization is 3 standard deviation of left and right side from the mean
graph(df4, 'Temperature1')

def outlier_detc(df, clm):
    m = np.mean(df[clm])
    sd = np.std(df[clm])
    l = df[clm]
    for i in l:
        z = (i - m)/sd
        if z > 3 or z < -3:
            df = df[df[clm] != i]
    return df

df4 = outlier_detc(df4, 'Temperature1')


graph(df4, 'Wind Speed1')
df4 = outlier_detc(df4, 'Wind Speed1')

sb.histplot(data = df4, x = 'Wind Speed1', kde = True)

graph(df4, 'Humidity1')

# **Visualisation after removal of Outliers**
```

```
graph(df4, 'Humidity1'), graph(df4, 'Wind Speed1'), graph(df4, 'Temperature1'),
graph(df4, 'Pressure_atm')

pt.scatter(x = df4.Humidity1, y = df4.Rain_01)

rainfall_count = df4.groupby('Rain_01')['Index'].agg('count').reset_index()
rainfall_count
x = rainfall_count.Rain_01
y = rainfall_count.Index

# **Visualizing the target Variable**

x = ['No Rain', 'Rain']
counts = y # example data

fig, ax = pt.subplots()
ax.bar(x, counts, color = ['g', 'b'])

# Adding count labels to each bar
for i, v in enumerate(counts):
    ax.text(i, v+0.5, str(v), ha='center', fontweight='bold')

# Set title and axis labels
ax.set_title('Rain Condition')
ax.set_xlabel('Rain')
ax.set_ylabel('Count')

# Show the graph
pt.show()

df_final = df4[['Temperature1', 'Humidity1',  'Rain_01']]
df_final.head()

x = df_final.iloc[:, :-1].values
x
y = df_final.iloc[:, -1].values
y
# In independent variable:
# The Inputs(x) are Temperature in (C), Wind Speed in mph(meter per hour),
Humidity in %, pressure in atm (atmosphere)
# The Output(y) is Rain (1), No Rain (0)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, stratify = y,
random_state = 42)

norma = Normalizer()
```

```
x_train = norma.fit_transform(x_train)
x_test = norma.transform(x_test)

x_train
x_test

os =  RandomOverSampler(sampling_strategy = 0.65)

x_train_res, y_train_res = os.fit_resample(x_train, y_train)

y_train_res

print('Original dataset shape {}'.format(Counter(y_train)))
print('Resampled dataset shape {}'.format(Counter(y_train_res)))

ann = tf.keras.models.Sequential()

ann.add(tf.keras.layers.Dense(units = 20, activation = 'relu'))
ann.add(tf.keras.layers.Dense(units = 7, activation = 'relu'))


ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

ann.fit(x_train_res, y_train_res, batch_size = 32, epochs = 20)

y_ann_predicted = ann.predict(x_test)
ann_lst = []
for i in y_ann_predicted:
  if i[0] >= 0.5:
    ann_lst.append([1])
  else:
    ann_lst.append([0])

ann_lst_predicted = np.array(ann_lst)
cr_ann = classification_report(y_test, ann_lst_predicted)
print(cr_ann)
cm_ann = confusion_matrix(y_test, ann_lst_predicted)
cm_ann

pt.figure(figsize = (9, 9))
sb.heatmap(cm_ann, annot = True, fmt='d')
pt.xlabel('Predicted')
pt.ylabel('Truth')
pt.title('Artificial Neural Network Confusion Matrix')
```

```
LogReg_model = LogisticRegression(solver='lbfgs',class_weight='balanced',
max_iter=10000)
LogReg_model.fit(x_train_res, y_train_res)
LogReg_model.score(x_test, y_test)

y_log_reg_predicted = LogReg_model.predict(x_test)
cr_log = classification_report(y_test, y_log_reg_predicted)
print(cr_log)
cm_lg = confusion_matrix(y_test, y_log_reg_predicted)
cm_lg

pt.figure(figsize = (9, 9))
sb.heatmap(cm_lg, annot = True, fmt='d')
pt.xlabel('Predicted')
pt.ylabel('Truth')
pt.title('Logistic Regression Confusion Matrix')

RFC_model = RandomForestClassifier()
RFC_model.fit(x_train_res, y_train_res)
RFC_model.score(x_test, y_test)

y_rfc_predicted = RFC_model.predict(x_test)
cr_rfc = classification_report(y_test, y_rfc_predicted)
print(cr_rfc)
cm_rfc = confusion_matrix(y_test, y_rfc_predicted)
cm_rfc

pt.figure(figsize = (9, 9))
sb.heatmap(cm_rfc, annot = True, fmt='d')
pt.xlabel('Predicted')
pt.ylabel('Truth')
pt.title('Random Forest Confusion Matrix')


# **Performances of models**
# Logistic Regression   ---> 73.2%
# Artificial Neural Network ---> 83.0%
# Random Forest Classifier ---> 85.5%

with open('RFCmodel.pkl', 'wb') as file:
   pickle.dump(RFC_model, file)
import os
print(os.getcwd())
```

PROJECT-Smart Irrigation using IOT and ML

by IIIT Tiruchirappalli

General metrics

68,795	9,557	941	38 min 13 sec	1 hr 13 min
characters	words	sentences	reading time	speaking time

Score

84

This text scores better than 84% of all texts checked by Grammarly

Writing Issues

418	95	323
Issues left	Critical	Advanced

Plagiarism

15 %

79 sources

15% of your text matches 79 sources on the web or in archives of academic publications

Writing Issues

212	**Clarity**
14	Intricate text
45	Unclear sentences
67	Passive voice misuse
13	Unclear paragraphs
68	Wordy sentences
4	Hard-to-read text
1	Word choice
142	**Correctness**
30	Improper formatting
5	Text inconsistencies
6	Comma misuse within clauses
6	Confused words
15	Determiner use (a/an/the/this, etc.)
2	Conjunction use
3	Wrong or missing prepositions
5	Misplaced words or phrases
3	Faulty subject-verb agreement
29	Misspelled words
4	Punctuation in compound/complex sentences
1	Faulty tense sequence
20	Unknown words
1	Incomplete sentences
1	Incorrect noun number
10	Misuse of semicolons, quotation marks, etc.

1	Closing punctuation	
50	**Engagement**	
50	Word choice	
14	**Delivery**	
11	Tone suggestions	
3	Incomplete sentences	

Unique Words

Measures vocabulary diversity by calculating the percentage of words used only once in your document

21%

unique words

Rare Words

Measures depth of vocabulary by identifying words that are not among the 5,000 most common English words.

40%

rare words

Word Length

Measures average word length

4.9

characters per word

Sentence Length

Measures average sentence length

10.2

words per sentence