IOT project

Smart Weather Monitoring System

Submitted by	CMS ID
Lamees	347289
Amna Ahmad	335246
Fatima Hussain	350328
Ayesha Shamim Naime	356760
Muhammad Abdullah Khan Abbasi	339550

BESE-11A

Table of Contents

Project Description:

Introduction:

The function of a weather monitoring system is to measure and record several meteorological conditions, including temperature, humidity, wind direction and speed, and precipitation. These systems are essential for providing meteorologists, farmers, pilots, sailors, and other organizations who need an in-depth understanding of weather phenomena with up-to-date weather information.

Weather monitoring systems come in various configurations, from simple installations with one weather station and a few sensors to more complex ones that include satellite data, computer modeling, and several weather stations. Weather monitoring systems have a wide range of uses, such as:

- Forecasting weather patterns and issuing timely weather warnings
- Investigating climate change and monitoring long-term weather trends
- Facilitating agricultural planning and irrigation strategies
- Supporting shipping and transportation planning
- Managing energy production and distribution
- Assisting in military operations

There are many different kinds of weather monitoring systems, including:

- Automatic weather stations: These gather and record meteorological data using sensors, then send it to a central location for study.
- Remote sensing systems: Using satellite data and additional technology, these systems collect extensive weather information worldwide.
- Computer modeling systems: Based on historical data, these systems simulate and anticipate weather patterns using complex computer algorithms.

The accuracy and reliability of the sensors used to gather data determine how effective weather monitoring systems are, regardless of the particular kind. These sensors could include rain gauges, anemometers, barometers, hygrometers, and thermometers, among others.

Objectives:

The primary goal of a weather monitoring system is to furnish dependable and precise information about the weather, enabling individuals and organizations to make well-informed decisions and safeguard themselves and their assets from the impacts of unfavorable weather conditions. For instance, in agriculture, accurate weather data is crucial for farmers as they plan their crop production and irrigation schedules. With knowledge of optimal planting, irrigating, and harvesting times, farmers can enhance their crop yields and minimize wastage.

Expected Outcomes:

The anticipated results of a weather monitoring system include the capability to precisely measure and predict weather conditions, issue warnings for severe weather events, and contribute to research on the Earth's climate and weather patterns. Additionally, the system aims to assist in agricultural planning, facilitate disaster response and recovery efforts, and furnish essential information for industries such as transportation that are impacted by varying weather conditions.

Application Requirements:

1. Sensors:

A weather monitoring system typically relies on a network of sensors to measure various weather variables. In this project, we used the following sensors:

- TMP36 for Temperature: Used to measure temperature.
- Ultrasonic Distance Sensor for Rain: Utilized for rain detection.
- Gas Sensor: Employed to measure gas levels.
- Multimeter with Potentiometer for Wind Speed: Calculated wind speed using a multimeter and potentiometer.

2. Data Storage and Processing:

A weather monitoring system must be able to store and process large amounts of data generated by its sensors. In this project, we used ThingSpeak for the data storage measured by our system.

3. Communication Infrastructure:

A reliable communication infrastructure is essential to transmit data from sensors to the central processing and analysis system. In this project, we utilized the following components:

- Arduino Uno R3: Used for interfacing with sensors and controlling the system.
- ESP8266 Wi-Fi Module: Enabled communication and data transmission to ThingSpeak.

4. Power Requirements:

The voltages required for the sensors and modules involved in our system are as follows:

- TMP36: Requires a voltage within the range of 2.7V to 5.5V.
- Ultrasonic Distance Sensor: Voltage requirements specific to the sensor model.
- Gas Sensor: Voltage requirements specific to the sensor model.
- Multimeter with Potentiometer: Powered by standard multimeter batteries.
- Arduino Uno R3: Operates at 5V.
- ESP8266 Wi-Fi Module: Operates within the specified voltage range.

5. Security:

A weather monitoring system must be secure to protect against unauthorized access to the data it collects and processes. This may involve the use of encryption, authentication protocols, and other security measures. Consider implementing security measures for communication with ThingSpeak and safeguarding sensitive weather data.

Tools(sensors) Used:

The weather monitoring system is a digital apparatus designed to measure various atmospheric conditions. It utilizes a collection of sensors interfaced with an Arduino Uno microcontroller for data acquisition and processing. The system measures temperature, gas concentration, rainfall, and wind speed. Below is a detailed description of how each component works and their role in measuring and monitoring these quantities:

1. **Breadboard**: prototyping tool used to connect different components together.
2. **Arduino Uno R3** : The microcontroller serves as the brain of the system. It reads data from the sensors, processes it, and controls the display on the LCD. It interfaces with the ESP8266 Wi-Fi module to send data to a server, in this case ThingSpeak, for remote monitoring.
3. **Wi-Fi Module (ESP8266)** :
   - **Description:** ESP8266 is a low-cost, low-power microcontroller with built-in Wi-Fi connectivity. It has a 32-bit microprocessor and can be programmed using the Arduino programming language. It has a few input/output (I/O) pins that can be used to connect sensors and other devices, as well as a built-in Wi-Fi module that can be used to connect to the internet or other Wi-Fi networks.
   - **How it works:** We use this module to connect the Arduino to the internet, allowing the system to transmit data to a central server for analysis and storage. It's set up in the code to communicate over serial, connect to Wi-Fi, and make HTTP requests to send sensor data to ThingSpeak. It can also be used to receive updates and alerts from the server and trigger alarms or other actions in response to certain weather conditions.
   - **Role in system:** It enables data transmission, allowing the system to contribute to a larger weather monitoring system.
4. **Temperature Sensor (TMP36):**
   - **Description:** This analog sensor operates on a low voltage range, and the voltage output changes linearly proportional to the Celsius temperature.
   - **How it works:** The Arduino reads the analog voltage, scales it to determine the temperature in Celsius, and displays the value.
   - **Role in system:** it provides real-time temperature data, which is essential for monitoring and understanding temperature variations.
5. **Gas Sensor (GAS):**
   - **Description:** This sensor detects the concentration of gas in the air. It provides a voltage output proportional to the gas concentration.
   - **How it works:** The Arduino reads its value and triggers the piezo buzzer if the gas level is too high, 200 ACD units in this case.
   - **Role in system:** It monitors air quality. An audible alert is triggered if gas levels are above the threshold, offering a warning system and ensuring safety.
6. **Ultrasonic Distance Sensor:**
   - **Description:** This sensor typically consists of a transmitter and a receiver. The transmitter emits ultrasonic pulses, and the receiver detects the echoes.

- **How it works:** Ultrasonic pulses are sent, and the time taken for echoes to return is used to calculate rainfall (in inches) using the formula: **`rain = 0.01723 * duration.`**
- **Role in system:** Provides data on rainfall, which is important for weather monitoring.

7. **Potentiometer:**
   - **Description:** It is a variable resistor with three terminals. It's commonly used to control the electrical resistance in a circuit.
   - **How it works:** The potentiometer's value is read by the Arduino and used to calculate the wind speed. The potentiometer's analog output is converted to voltage and then read by the Arduino and used to calculate the wind speed. The formula used to convert analog readings to wind speed is: **`wind speed = analog reading  * (5.0 / 1023.0)`**
   - **Role in system:** It monitors wind speed, important for understanding weather patterns, and displays measurement on an LCD. This data can also be sent to ThingSpeak for further analysis.
8. **Piezo Buzzer (PIEZO):** It is an electronic device that can be used to detect vibrations or to produce sound. In this system, the buzzer is triggered when the gas sensor detects a value above a certain threshold. This can alert users to the presence of potentially harmful gasses.
9. **LCD 16 x 2:** The LCD is used to display the rainfall and wind speed data measured by the system. The LiquidCrystal library in the code initializes the LCD and sets the cursor to display the text at the correct location.
10. **Voltage Multimeter:** This device measures voltage. In this system, it is monitoring the voltage levels of various components to ensure they are within operational parameters for the sensors and microcontroller.
11. **Resistors:** These resistors are used for various purposes, including current limiting to protect components or as part of voltage dividers to read sensor values correctly.

The system works as a cohesive unit where each sensor gathers specific environmental data, the Arduino processes and compiles this data, the LCD displays the current readings, and the Wi-Fi module transmits the data to a remote server. This allows for both local monitoring through the LCD and remote monitoring via internet connectivity. This ensures continuous and real-time updates of the weather conditions.

Power Consumption and Optimization:

1. **Arduino Uno R3:**

   Power Consumption: The Arduino Uno R3 typically consumes around 50-60 mA when active. The power consumption can vary based on the tasks it performs, such as reading sensors, processing data, and communicating with the Wi-Fi module.

   Optimization Approach: To optimize power consumption with Arduino Uno, you can:

   - **Utilize sleep modes:** Put the microcontroller to sleep when it's not actively processing data or communicating. This significantly reduces power consumption during idle periods.
   - **Disable unnecessary peripherals:** Turn off unused peripherals (e.g., ADC, timers) to minimize power consumption.
2. **ESP8266 Wi-Fi Module:**

   Power Consumption: The ESP8266's power consumption can range from tens of milliamps to over 200 mA during data transmission, depending on the mode and transmission power.

   Optimization Approach: Optimize power consumption with the ESP8266 by:

   - **Using low-power modes:** Switch the module to sleep mode when it's not actively transmitting data.
   - **Adjusting transmission power:** Lower the transmission power if the distance to the Wi-Fi router allows, as higher power levels consume more energy.
   - **Implementing efficient communication protocols:** Minimize the time the Wi-Fi module spends in active mode by using efficient protocols and minimizing the amount of data transmitted.
3. **Temperature Sensor (TMP36):**

   Power Consumption: The TMP36 is a low-power sensor, typically drawing less than 50 μA during measurements.

   Optimization Approach: As the TMP36 is already low-power, optimization options may be limited. However, you can:

   - **Read data at longer intervals:** Increase the time between temperature readings to reduce the frequency of sensor activations.
   - **Power down when not in use:** If the temperature sensor is not required constantly, power it down between readings.
4. **Gas Sensor (GAS):**

Power Consumption: Gas sensors usually have a higher power consumption during active sensing, often ranging from 50 mA to a few hundred mA.

Optimization Approach: Optimize power consumption with the gas sensor by:

- **Adjusting sensing frequency:** Increase the time between gas concentration measurements to reduce power usage.
- **Powering down during idle periods:** Turn off the gas sensor when air quality monitoring is not required.

5. **Ultrasonic Distance Sensor:**

   Power Consumption: Ultrasonic sensors typically consume a few milliamps during operation.

   Optimization Approach: Optimize power consumption with the ultrasonic sensor by:

   - **Increasing measurement intervals:** Space out the time between ultrasonic pulses to reduce power consumption.
   - **Powering down during idle periods:** Turn off the ultrasonic sensor when rainfall data is not needed.

6. **Piezo Buzzer (PIEZO):**

   Power Consumption: Piezo buzzers are low-power devices, typically drawing a few milliamps during operation.

   Optimization Approach: As the buzzer is usually triggered intermittently, power optimization may not be a primary concern.

7. **LCD 16 x 2:**

   Power Consumption: A typical 16x2 LCD might consume around 1-2 mA without backlight and 20-25 mA with backlight.

   Optimization Approach: Optimize power consumption with the LCD by:

   - Turning off the backlight when not needed.
   - Updating the display only when necessary, rather than continuously refreshing.

8. **Voltage Multimeter:**

   Power Consumption: Voltage multimeters typically have low power consumption.

   Optimization Approach: No specific optimization is required for power consumption with a voltage multimeter.

9. **Resistors:**

   Power Consumption: Resistors do not consume power in the usual sense, as they are passive components.

   Optimization Approach: No specific optimization is required for power consumption with resistors.

10. **Potentiometer:**

    Power Consumption: Potentiometers are passive components and do not consume power.

    Optimization Approach: No specific optimization is required for power consumption with a potentiometer.

Communication Technologies/Protocols:

Physical Layer:

Communication Technology: Wi-Fi (Wireless Fidelity)

Description: The physical layer involves the transmission of data over the air using Wi-Fi technology. The ESP8266 module is equipped with a built-in Wi-Fi module, allowing it to establish a wireless connection to the internet.

Internet Layer:

Communication Protocol: TCP/IP (Transmission Control Protocol/Internet Protocol)

Description: The internet layer is responsible for routing data packets between devices across the internet. The ESP8266 uses the TCP/IP protocol suite to communicate with ThingSpeak over the internet. This involves addressing, routing, and delivering data packets reliably.

Application Layer:

Communication Protocol: HTTP (Hypertext Transfer Protocol)

Description: The application layer is where high-level communication protocols operate. In this case, the ESP8266 communicates with ThingSpeak using HTTP, a protocol commonly used for web communication. The Arduino sends HTTP GET requests to ThingSpeak to update data fields in your ThingSpeak channel.

Possible Topologies to Scale-Up:

Currently, we are using a star topology where each sensor node communicates directly with a central hub.i.e. ESP8266. The central hub manages the data transmission to ThingSpeak, providing a straightforward and easily manageable structure. This topology is simple and easy to manage.

But in case of failure, the central hub becomes a single point of failure, meaning that if it malfunctions, the entire network may be affected. To scale up the system, we can use following topologies

1. **Mesh Topology:**

   Description: Mesh topology involves interconnected sensor nodes, allowing data to be routed through multiple nodes to reach the destination, such as ThingSpeak. This topology provides redundancy and flexibility in data transmission pathways.

   Advantages: Mesh networks offer redundancy, self-healing capabilities, and flexibility in adding or removing sensor nodes without disrupting the entire network.

   Consideration: Managing interactions between nodes can become complex as the number of nodes increases.

2. **Hierarchical Topology:**

   Description: Hierarchical topology organizes sensor nodes into multiple levels, with each level having a designated leader or coordinator. This approach is suitable for managing large-scale deployments with clear hierarchical structures.

   Advantages: The hierarchical topology offers scalability, efficient management, and a defined structure that can help in organizing and maintaining the network.

   Consideration: Designing and maintaining a hierarchical structure may introduce additional complexity.

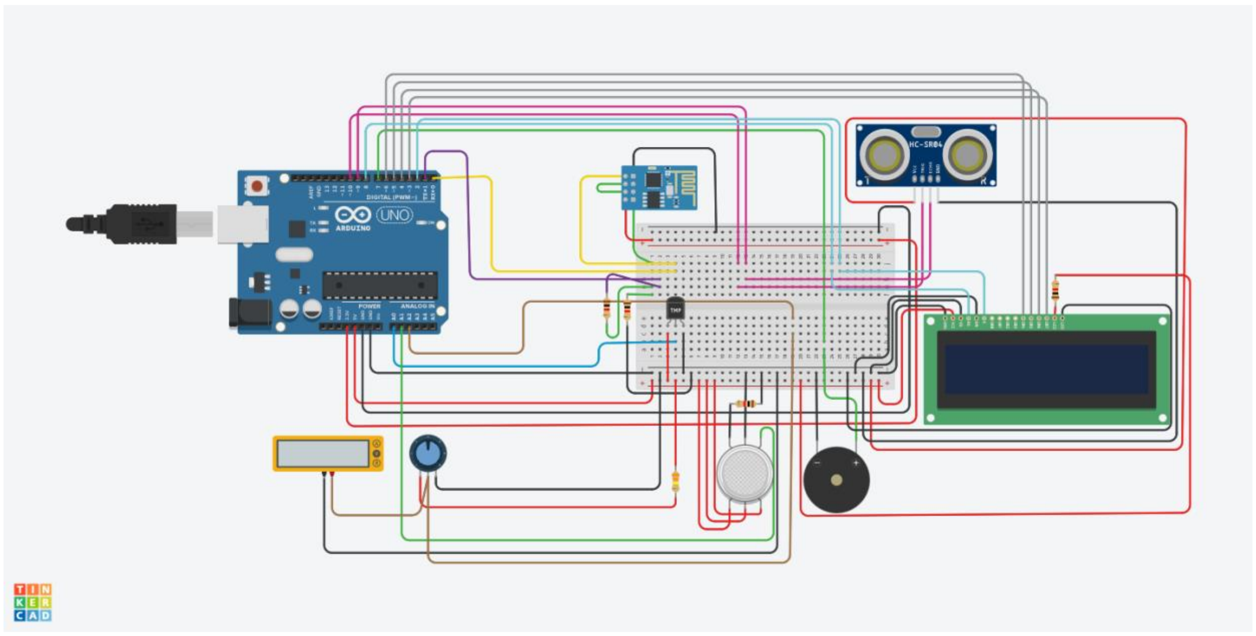
3. **Cloud-based Architecture:**

   Description: In a cloud-based architecture, data from sensor nodes is sent directly to a cloud server, which can then interact with platforms like ThingSpeak. This approach simplifies data management and analysis by leveraging cloud services.

   Advantages: Cloud-based architectures offer scalability, centralized data storage, and seamless integration with various cloud services for advanced data processing and analysis.

   Consideration: Dependence on internet connectivity and the reliability of cloud services are important considerations.

Circuit Diagram:

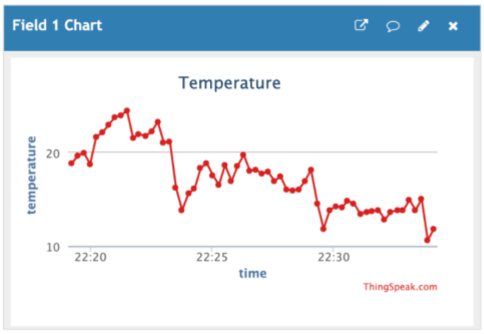


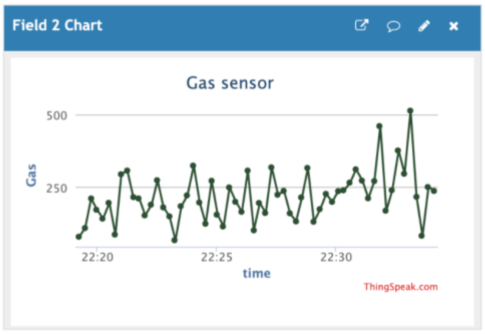
Simulation link: https://www.tinkercad.com/things/63uT0urji4L-smart-weather-monitoring-system/editel?returnTo=%2Fdashboard%3Ftype%3Dcircuits%26collection%3Ddesigns&sharecode=EOPQ3UTtL-zTfysde11DOLlvgQ4gDv-XTJ3oMCzPezU

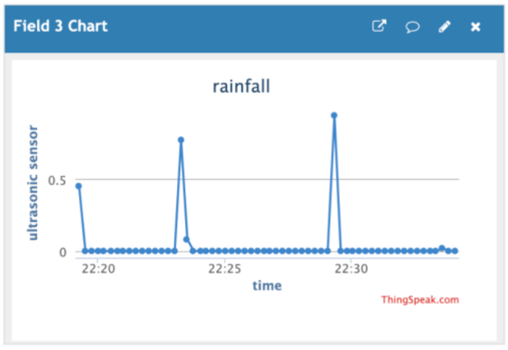
Data on Thingspeak:

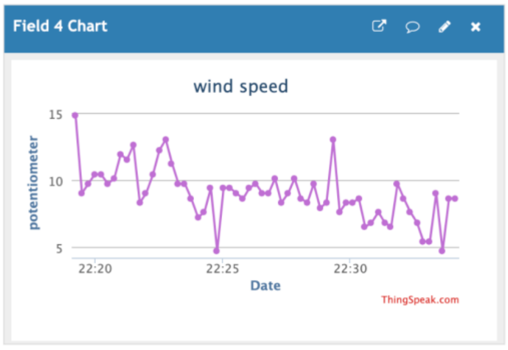
Data Analysis:

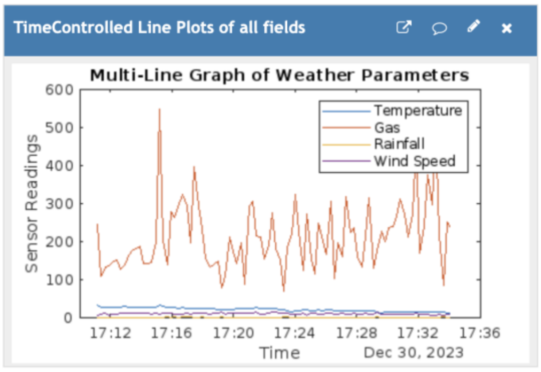
These are the graphs of the data we measured from our weather monitoring system and stored it on Thingspeak

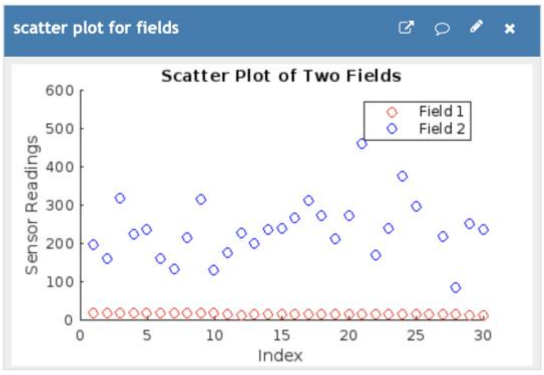






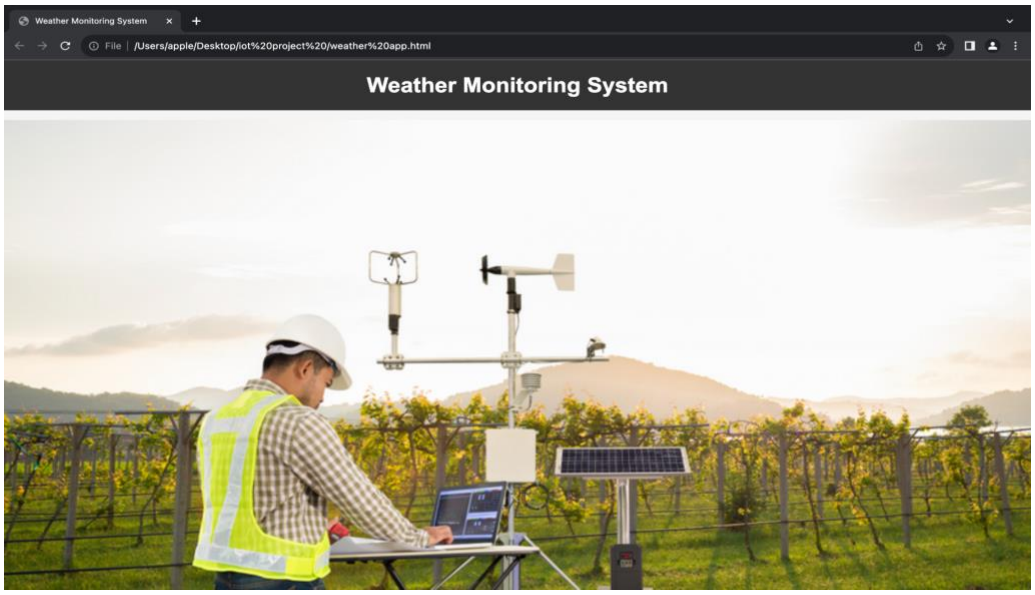


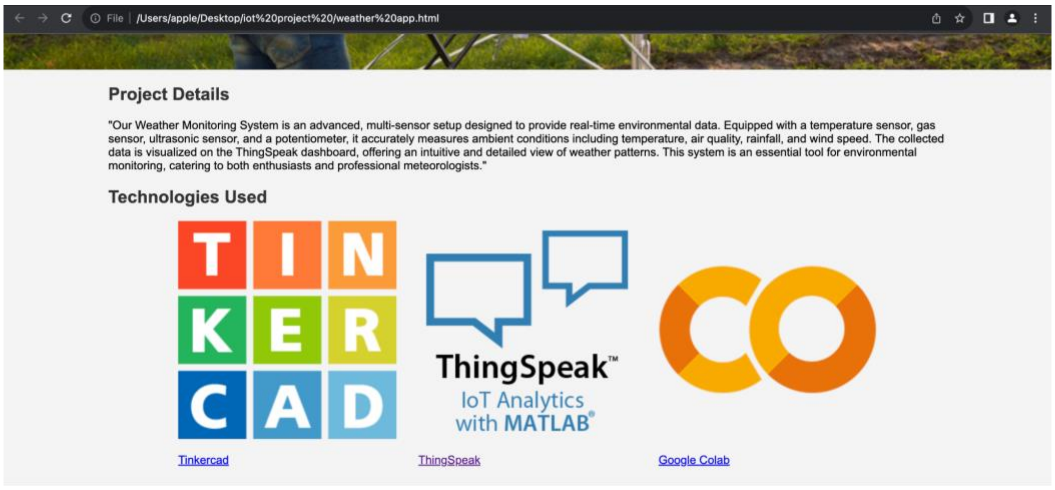




Dashboard link : https://thingspeak.com/channels/2392076

Our Web Portal:


Weather Monitoring System
File | /Users/apple/Desktop/iot%20project%20/weather%20app.html
Weather Monitoring System


File | /Users/apple/Desktop/iot%20project%20/weather%20app.html

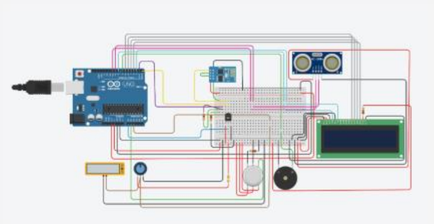
Project Details

"Our Weather Monitoring System is an advanced, multi-sensor setup designed to provide real-time environmental data. Equipped with a temperature sensor, gas sensor, ultrasonic sensor, and a potentiometer, it accurately measures ambient conditions including temperature, air quality, rainfall, and wind speed. The collected data is visualized on the ThingSpeak dashboard, offering an intuitive and detailed view of weather patterns. This system is an essential tool for environmental monitoring, catering to both enthusiasts and professional meteorologists."

Technologies Used

Tinkercad ThingSpeak Google Colab

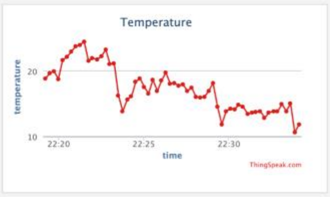
Simulation

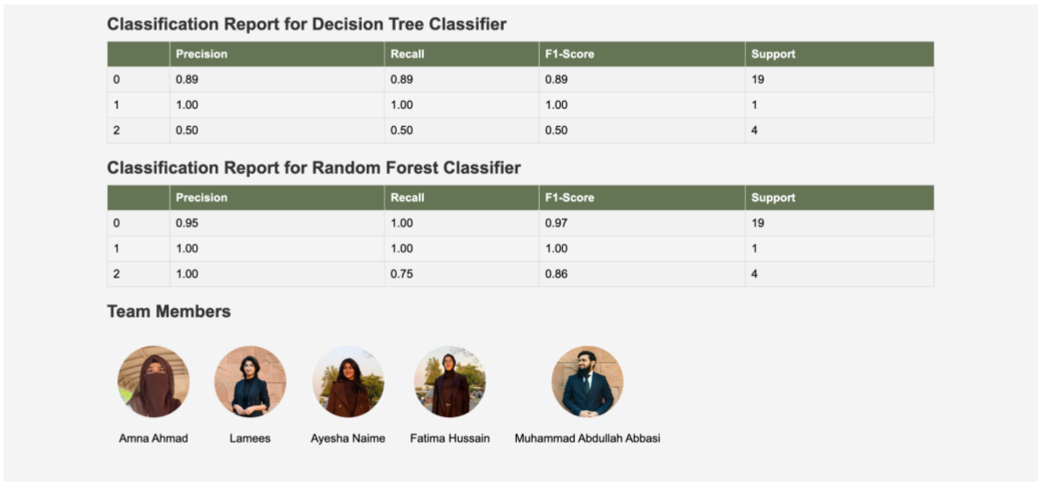


Link to Tinkercad Simulation

Statistics

Temperature





Classification Report for Decision Tree Classifier

	Precision	Recall	F1-Score	Support
0	0.89	0.89	0.89	19
1	1.00	1.00	1.00	1
2	0.50	0.50	0.50	4

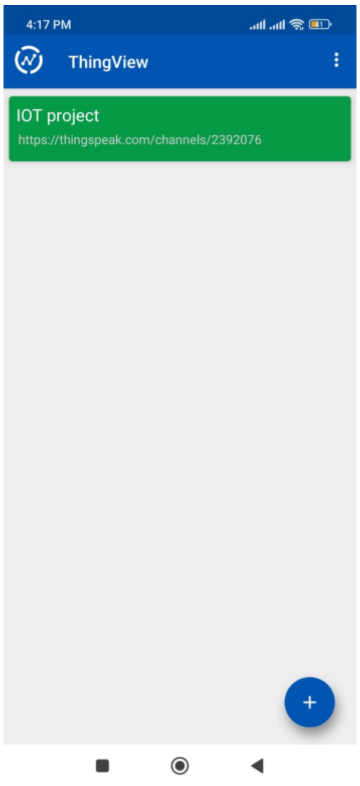
Classification Report for Random Forest Classifier

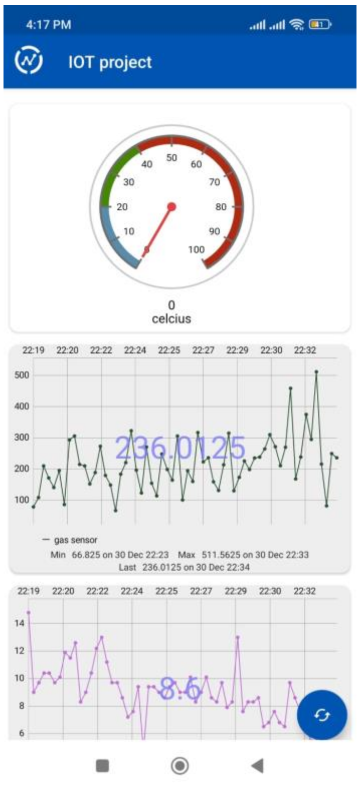
	Precision	Recall	F1-Score	Support
0	0.95	1.00	0.97	19
1	1.00	1.00	1.00	1
2	1.00	0.75	0.86	4

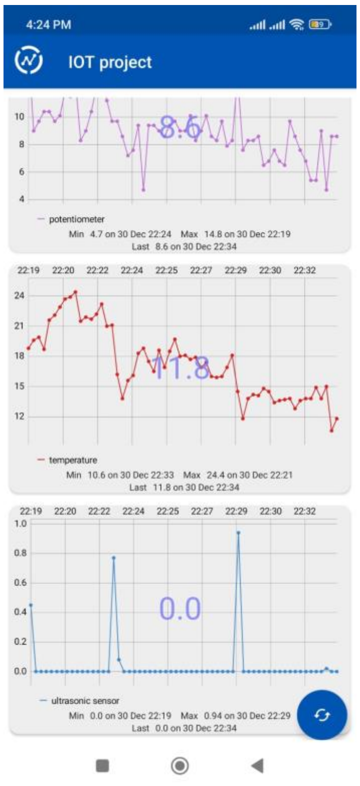
Team Members

Amna Ahmad Lamees Ayesha Naime Fatima Hussain Muhammad Abdullah Abbasi

Android application to monitor:




4:17 PM
IOT project
0
celcius
236.0125
gas sensor
Min 66.825 on 30 Dec 22:23 Max 511.5625 on 30 Dec 22:33
Last 236.0125 on 30 Dec 22:34
8.6


4:24 PM
IOT project
8.6
potentiometer
Min 4.7 on 30 Dec 22:24 Max 14.8 on 30 Dec 22:19
Last 8.6 on 30 Dec 22:34
11.8
temperature
Min 10.6 on 30 Dec 22:33 Max 24.4 on 30 Dec 22:21
Last 11.8 on 30 Dec 22:34
0.0
ultrasonic sensor
Min 0.0 on 30 Dec 22:19 Max 0.94 on 30 Dec 22:29
Last 0.0 on 30 Dec 22:34

Code:

GitHub Link:

https://github.com/amnaahmad20/Smart-weather-monitoring-system

Simulation Code:

```
//Smart weather reporting system
#include <LiquidCrystal.h>

// Temperature sensor variables
float temp_vout;
float temp;
float voltage;

// Gas sensor variables
int gas_sensor_port = A1;
int gas_sensor_value = 0;

// Rainfall measurement variables
float rain;
```

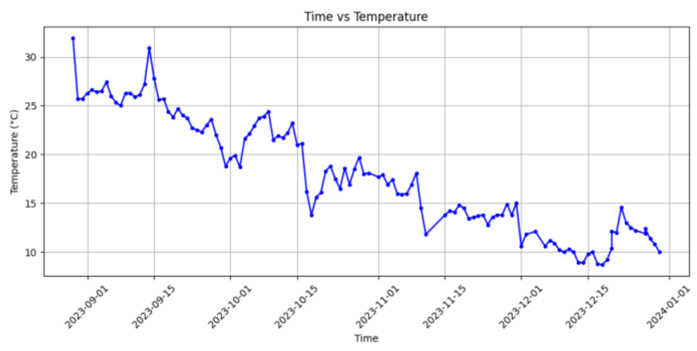
```
const int triggerPin = 10;
const int echoPin = 9;
long duration;

// Wind speed measurement variables
float V_wind = 0;
float Windspeedfloat;
int Windspeedint;

// LCD
LiquidCrystal lcd(8, 2, 6, 5, 4, 3); // Parameters: (rs, enable, d4, d5, d6, d7)

// WIFI module variables
String ssid = "Simulator Wifi";  // SSID to connect to
String password = "";            // Virtual wifi has no password
String host = "api.thingspeak.com";
const int httpPort = 80;
String apiKey = "991K6D6ET5LJY84X";  // Replace with your ThingSpeak channel's
write API key

void setupESP8266() {
  Serial.begin(115200); // Baud rate
  Serial.println("AT");
  delay(300); // Wait for ESP8266 to respond

  Serial.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\"");
  delay(500); // Wait for connection

  Serial.println("AT+CIPMUX=0"); // Single connection mode
  delay(300);

  Serial.println("AT+CIPSTART=\"TCP\",\"" + host + "\"," + httpPort);
  delay(300);
}

void send_data() {
  String httpRequest = "GET /update?api_key=" + apiKey + "&field1=" +
String(temp) + "&field2=" + String(gas_sensor_value) + "&field3=" + String(rain)
+ "&field4=" + String(Windspeedfloat) + " HTTP/1.1\r\nHost: " + host +
"\r\n\r\n";
  int length = httpRequest.length();

  Serial.print("AT+CIPSEND=");
  Serial.println(length);
  delay(100); // Wait for ESP8266 to respond

  Serial.print(httpRequest);
  delay(500); // Wait for data to be sent
}

void setup() {
  pinMode(A1, INPUT);
  pinMode(7, OUTPUT);
  pinMode(A0, INPUT);
  pinMode(A2, INPUT);
  setupESP8266();
  lcd.begin(16, 2);
  pinMode(triggerPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  //for temperature sensor
  temp_vout = analogRead(A0);
  voltage = temp_vout * 0.0048828125; //convert analog value between 0 to 1023
with 5000mV/5V ADC
  temp = (voltage - 0.5) * 100.0;
  Serial.println("Current temperature: " + String(temp));
  //---------------------------------------------------------------------------------
----------------------
  //for gas sensor
  gas_sensor_value = analogRead(gas_sensor_port);
  Serial.println("Gas sensor value: " + String(gas_sensor_value));
  if (gas_sensor_value > 200)
  {
    tone(7,523,1000);
  }

  //---------------------------------------------------------------------------------
--------------------------
  //for rainfall measurement
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  // Sets the trigger pin to HIGH state for 10 microseconds
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  // Reads the echo pin, and returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  rain = 0.01723 * duration;
  delay(3); // Delay a little bit to improve simulation performance

  lcd.setCursor(0,0); // Sets the location at which subsequent text written to
the LCD will be displayed
  lcd.print("Rainfall: "); // Prints string "Rainfall" on the LCD
  lcd.print(rain); // Prints the distance value from the sensor
  Serial.println("Rainfall: " + String(rain));
  delay(3);
 //--------------------------------------------------------------------------------
-----------------------
  //for wind speed measurement
  float V_wind = analogRead(A2) * (5.0 / 1023.0);
  // Voltage converted to MPH
  Windspeedint = (V_wind - 0.4) * 10 * 2.025 * 2.237;     // For LCD screen
output
  Windspeedfloat = (V_wind - 0.4) * 10 * 2.025 * 2.237; // For Serial monitor
output
  //wind speed LCD output
  lcd.setCursor(0,1);        // adjust cursor
  lcd.print("Wind speed");
  lcd.print(" ");
  if (V_wind < 0.4)
  {
    lcd.print("0");
  }
  else
  {
    lcd.print(Windspeedint);
  }
  lcd.print("MPH");
  //wind speed serial monitor output
  Serial.print("Wind Speed: ");
  if (Windspeedfloat <= 0)
  {
    Serial.print("0.0");}
  else{
    Serial.print(Windspeedfloat);}      // Output Wind speed value
  Serial.println(" MPH");
  delay(100);

  Serial.print("Anemometer Voltage: ");
  if (V_wind > 2){
    Serial.println("Out of range!");
  }
  else if (V_wind < 0.4)
  {
    Serial.println("Out of range!");
  }
  else{
    Serial.print(V_wind);
    Serial.println(" V");}


  // Send data to ThingSpeak
  send_data();
  delay(5000); // Delay for 5 seconds before sending data again
}
```

Descriptive Analysis:

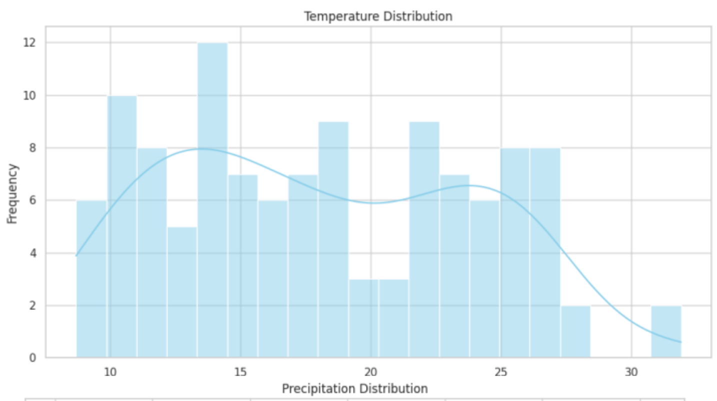
Utilizing prominent data analytics libraries such as Pandas, NumPy, Matplotlib, and Seaborn, we conducted a comprehensive visualization of the dataset to elucidate its behavioral patterns. Subsequently, thorough preprocessing steps were applied, laying the groundwork for the development of a robust machine learning model

Analytics:

Islamabad's temperature in the year 2023's winter.

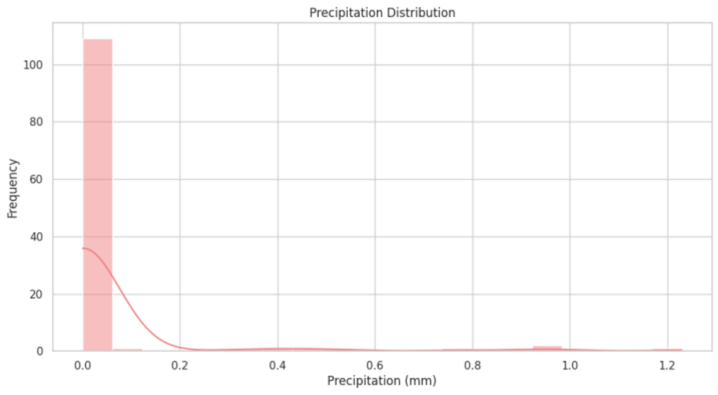


Histogram of the temperature in Islamabad:



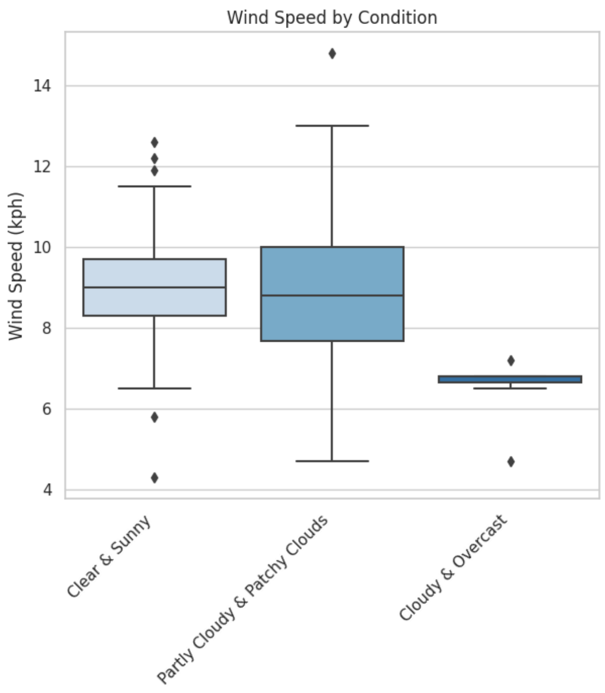
Due the season being winter, the frequency of lower temperature is more than the frequency of higher temperature

Histogram of the precipitation in Islamabad:



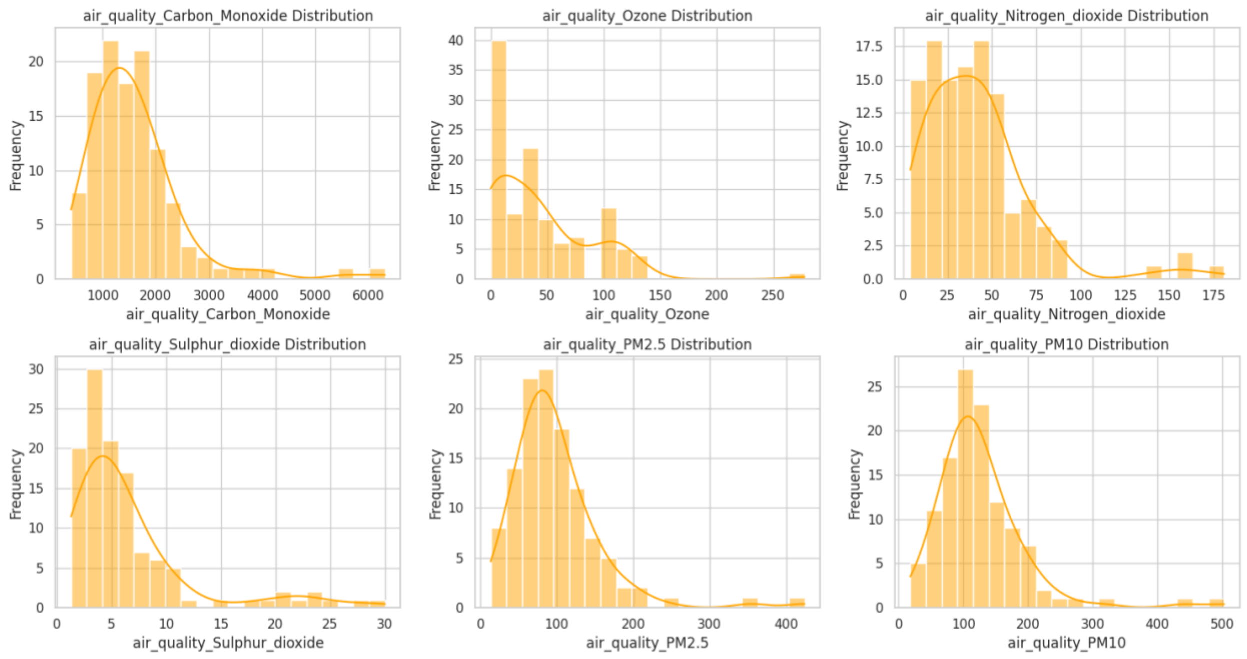
The histogram shows that this winter there was very little rain.

Bivariate Analysis between the day's outlook and wind speed:

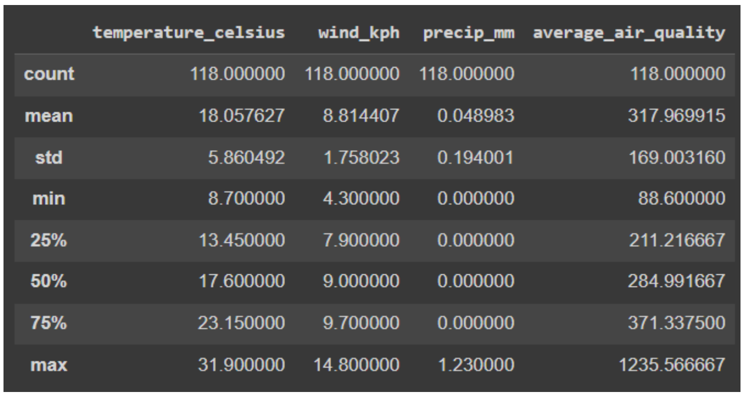


The boxplot highlights distinct wind speed patterns across different weather conditions. Cloudy and overcast days exhibit lower wind speeds, consistent with expectations due to higher humidity and relatively still air. Partly cloudy days showcase the widest range of wind speeds, whereas sunny days feature the highest number of outliers

Histogram of different air pollutants:



Carbon monoxide exhibits the widest range, indicating concentrations spanning from 1000 to 2000. Conversely, sulfur dioxide (SO2) displays a narrower range (1 to 30), emphasizing its relatively lower impact on atmospheric acidity compared to nitrogen dioxide (NO2), which presents a range of 10 to 175. This observation suggests that measures to mitigate air quality issues should prioritize addressing NO2 emissions. It's crucial to acknowledge that the higher concentrations of gasses in Islamabad may not necessarily originate within the city, given the influence of wind direction on air quality. Additionally, the concentrations of particulate matter (PM2.5 and PM10) show distinct ranges, with PM2.5 ranging from 0 to 400 and PM10 ranging from 0 to 500. Notably, occurrences beyond 300 are infrequent.
Statistical Description of numerical Features:



	temperature_celsius	wind_kph	precip_mm	average_air_quality
count	118.000000	118.000000	118.000000	118.000000
mean	18.057627	8.814407	0.048983	317.969915
std	5.860492	1.758023	0.194001	169.003160
min	8.700000	4.300000	0.000000	88.600000
25%	13.450000	7.900000	0.000000	211.216667
50%	17.600000	9.000000	0.000000	284.991667
75%	23.150000	9.700000	0.000000	371.337500
max	31.900000	14.800000	1.230000	1235.566667

Preprocessing:

1. Column Selection:

- Out of the extensive list of columns, only the relevant ones were retained for analysis.
- Selected columns: 'temperature_celsius', 'condition_text', 'wind_kph', 'precip_mm', 'gust_kph', 'air_quality_Carbon_Monoxide', 'air_quality_Nitrogen_dioxide', 'air_quality_Sulphur_dioxide', 'air_quality_PM2.5', 'air_quality_PM10', 'air_quality_us-epa-index', 'air_quality_gb-defra-index'.

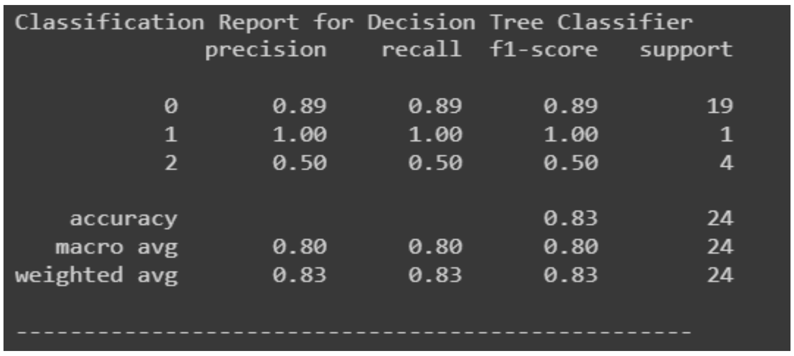
2. **Air Quality Processing:**
   - Air quality columns were consolidated into a single column, 'average_air_quality'.
   - The remaining non-air quality columns were dropped from the dataset.
3. **Normalization:**
   - The columns 'temperature_celsius', 'wind_kph', and 'average_air_quality' were normalized to ensure uniform scales for accurate modeling.
4. **Class Merging:**
   - Eight original weather classes were merged into three more generalized categories:'Clear & Sunny','Partly Cloudy & Patchy Clouds' and 'Cloudy & Overcast'.
5. **Label Encoding:**
   - The merged classes were label encoded into numerical values (0, 1, 2) for use as target variables in machine learning models.

Predictive Analysis:

Decision Trees:

A Decision Tree Classifier is a supervised machine learning algorithm that makes decisions based on a tree-like graph structure. It recursively splits the dataset into subsets based on the most significant feature at each step, forming a tree where each leaf node represents a class label.

Results for the decision tree:



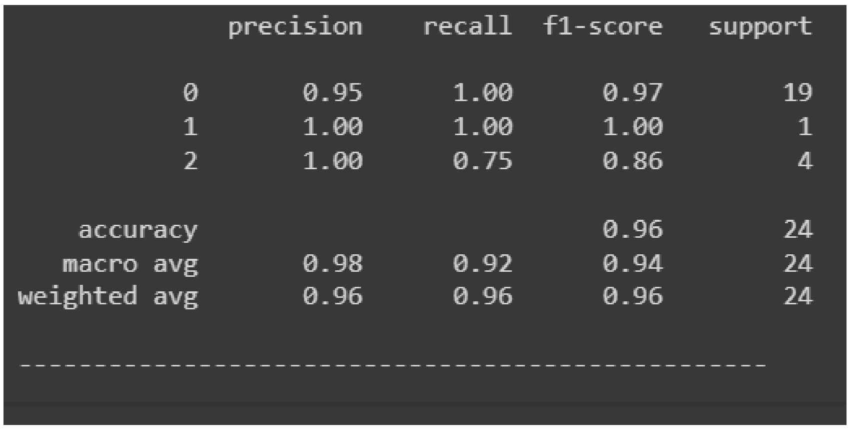
```
Classification Report for Decision Tree Classifier
              precision    recall  f1-score   support

           0       0.89      0.89      0.89        19
           1       1.00      1.00      1.00         1
           2       0.50      0.50      0.50         4

    accuracy                           0.83        24
   macro avg       0.80      0.80      0.80        24
weighted avg       0.83      0.83      0.83        24

--------------------------------------------------
```

Random Forest:

Decision tree did very well in 'Clear & Sunny' and 'Partly Cloudy & Patchy Clouds' classes but needs improvement in 'Cloudy & Overcast' due to the imbalance in target classes. Therefore we trained a Random Forest Classifier as well. Random Forest Classifier is an ensemble learning method that builds multiple decision trees and merges their predictions. It introduces randomness by training each tree on a random subset of features and a random subset of the training data. The final prediction is determined through a voting mechanism.

Random Forest's ensemble approach mitigates biases towards majority classes found in imbalanced datasets. By aggregating predictions from multiple trees, it creates a more robust and accurate model, particularly effective in handling imbalanced class distributions.

Results for the random forest:



```
              precision    recall  f1-score   support

           0       0.95      1.00      0.97        19
           1       1.00      1.00      1.00         1
           2       1.00      0.75      0.86         4

    accuracy                           0.96        24
   macro avg       0.98      0.92      0.94        24
weighted avg       0.96      0.96      0.96        24

------------------------------------------------
```

Impact on society:

1. Advancements in Weather Forecasting:

Through the collection and analysis of real-time weather data from multiple weather stations, meteorologists can generate more precise forecasts regarding future weather conditions. This enables individuals and organizations to effectively plan and prepare for adverse weather events, such as storms or extreme temperatures.

2. Deepened Understanding of Climate Change:

The accumulation of long-term weather data empowers scientists to examine trends in weather patterns, fostering a comprehensive understanding of how the climate is evolving over time. This valuable information aids in comprehending the causes and impacts of climate change, facilitating the development of strategies to mitigate its effects.

3. Optimized Agricultural Practices:

Leveraging weather data to fine-tune crop production and irrigation schedules enables farmers to enhance crop yields and minimize wastage. This heightened agricultural efficiency is a direct result of informed decision-making based on accurate weather information.

4. **Improved Energy Production and Distribution:**

   Integrating weather data into the optimization of energy production and distribution processes allows energy companies to meet demand with greater efficiency, thereby reducing waste. This strategic use of weather information contributes to more effective energy management.

References:

[1] https://www.tinkercad.com/

[2] https://geekflare.com/smart-weather-stations/

[3] BulipeSrinivasRao , Prof. Dr. K. SrinivasaRao , Mr. N. Ome, Internet of Things (IOT) Based Weather Monitoring system, IJARCCE Journal,vol. 5

[4] https://colab.research.google.com