This document aims to provide a visualization of the project and allow readers to understand how the simulation operates in detail. All figures are screenshots of Node-RED nodes and MongoDB documents.

The overall steps of this project are as follows:

1. Simulate an IoT network which generates weather information
2. Store simulated IoT device data in an external database
3. Train a machine learning model using data collection in external database
4. Test the machine learning model and asses its accuracy
5. Test the machine learning model by generating new data points and verify its accuracy

Step 1: Setup a collection of 25 MongoDB documents. The nodes in Figure 1, when executed, will generate 25 documents through the code in Figure 2, then pass the array of documents to the local database.

Figure 1: Node-RED which sends an array of JSON templates to the local MongoDB database

```
var jsonTemplate = {
    "State": "California",
    "City": "Los Angeles",
    "ID": null,
    "X": null,
    "Y": null,
    "Date": [],
    "Temperature": [],
    "Cloudy": [],
    "Season": [],
    "Humidity": [],
    "WindSpeed": [],
    "Neighbors": []
};
var GRID_X = 5;
var GRID_Y = 5;
msg.payload = [];
var docs = [];

for(let i = 0; i < GRID_X; i++) {
    for(let j = 0; j < GRID_Y; j++) {
        let initialCondition = JSON.parse(JSON.stringify(jsonTemplate));
        initialCondition.X = i;
        initialCondition.Y = j;
        initialCondition.ID = (i*GRID_X)+j;
        if(i - 1 >= 0) {
            initialCondition.Neighbors.push(initialCondition.ID - GRID_X);
        }
        if(i + 1 < GRID_X) {
            initialCondition.Neighbors.push(initialCondition.ID + GRID_X);
        }
        if(j - 1 >= 0) {
            initialCondition.Neighbors.push(initialCondition.ID - 1);
        }
        if(j + 1 < GRID_Y) {
            initialCondition.Neighbors.push(initialCondition.ID + 1);
        }
        docs.push(initialCondition);
    }
}

msg.payload.push(docs);
return msg;
```

Figure 2: Fill Collection node code

Figure 3 below demonstrates an empty document within MongoDB which will be associated with a simulated IoT device. The template will store new entries as array pushes, which will permit data entry in any singular document per simulated device.

Figure 3: Example of Document in MongoDB

Step 2: Store simulated IoT device data in an external database. In this step, we create 25 virtual IoT devices to begin the data generating process. Figure 4 represents a single IoT device which passes its ID to the sub flow in Figure 5. The flow in Figure 5 will retrieve the document from the database, generate new weather information by the code in Figure 6, then update the document in the database. The result will be an entry in the database as represented in figure 7.

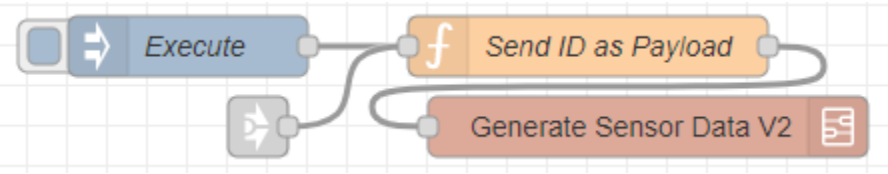

Figure 4: Individual IoT device which connects to sub flow in Figure 5

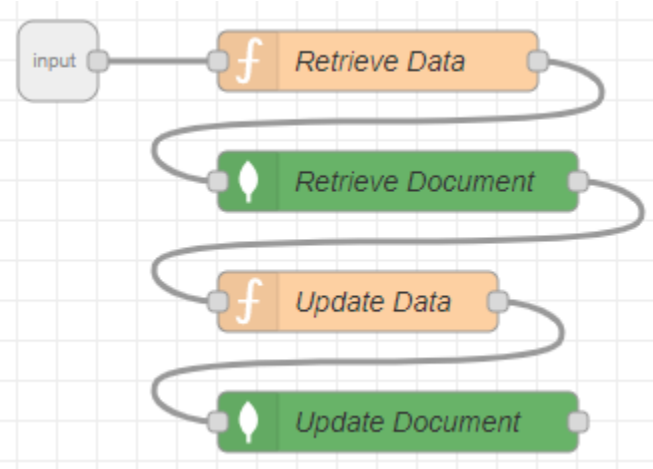

Figure 5: Sub flow which receives sensor ID, retrieves a copy of the document, then generates new weather information and updates the document in the database

```
const AVG_COOLING_BY_CLOUDS = 9;
const AVG_SPRING_TEMP = 67;
const AVG_SUMMER_TEMP = 83;
const AVG_FALL_TEMP = 72;
const AVG_WINTER_TEMP = 56
const HIGHEST_WIND_SPEED = 6;
const MAX_HUMIDITY_EFFECT = 6;

var updateDoc = {};
updateDoc.ID = msg.payload.ID;
updateDoc.Temperature = null;
updateDoc.Cloudy = Math.random().toFixed(2) * AVG_COOLING_BY_CLOUDS;
updateDoc.Season = Math.floor(Math.random() * 4 + 1);
updateDoc.Humidity = (Math.random().toFixed(2) * 2 - 1) * MAX_HUMIDITY_EFFECT;
updateDoc.WindSpeed = Math.floor(Math.random() * HIGHEST_WIND_SPEED + 1);
updateDoc.Date = new Date();

var currentTemp = 0;
switch(updateDoc.Season) {
    case 1:
        currentTemp = AVG_SPRING_TEMP;
        break;
    case 2:
        currentTemp = AVG_SUMMER_TEMP;
        break;
    case 3:
        currentTemp = AVG_FALL_TEMP;
        break;
    case 4:
        currentTemp = AVG_WINTER_TEMP;
        break;
}

currentTemp -= updateDoc.Cloudy;
currentTemp -= updateDoc.WindSpeed;
currentTemp += updateDoc.Humidity;

if(currentTemp < 65) {
    updateDoc.Temperature = "COLD";
}
else if(currentTemp < 77) {
    updateDoc.Temperature = "MODERATE";
}
else {
    updateDoc.Temperature = "HOT";
}

msg.payload = [
    {"ID": updateDoc.ID},
    {"$push":
        {
            Temperature: updateDoc.Temperature,
            Date: updateDoc.Date,
            Humidity: updateDoc.Humidity,
            Cloudy: updateDoc.Cloudy,
            Season: updateDoc.Season,
            WindSpeed: updateDoc.WindSpeed
        }
    }
];
return msg;
```

Figure 6: Weather generating code within the Update Data node

```
_id: ObjectId("612ec7745f0d7823bc42b769")
State: "California"
City: "Los Angeles"
ID: 0
X: 0
Y: 0
Date: Array
    0: 2021-09-01T20:36:07.765+00:00
Temperature: Array
    0: "MODERATE"
Cloudy: Array
    0: 0.45
Season: Array
    0: 2
Humidity: Array
    0: -4.92
WindSpeed: Array
    0: 6
Neighbors: Array
    0: 5
    1: 1
```


Figure 7: Example data entry by Sensor 0 which pushes new data to each array.

The result is 25 virtual IoT devices which each generates its own weather information, as seen in Figure 8. Each device will have an associated MongoDB document and are organized to execute once per second. Of course, this configuration is specific to the simulation.

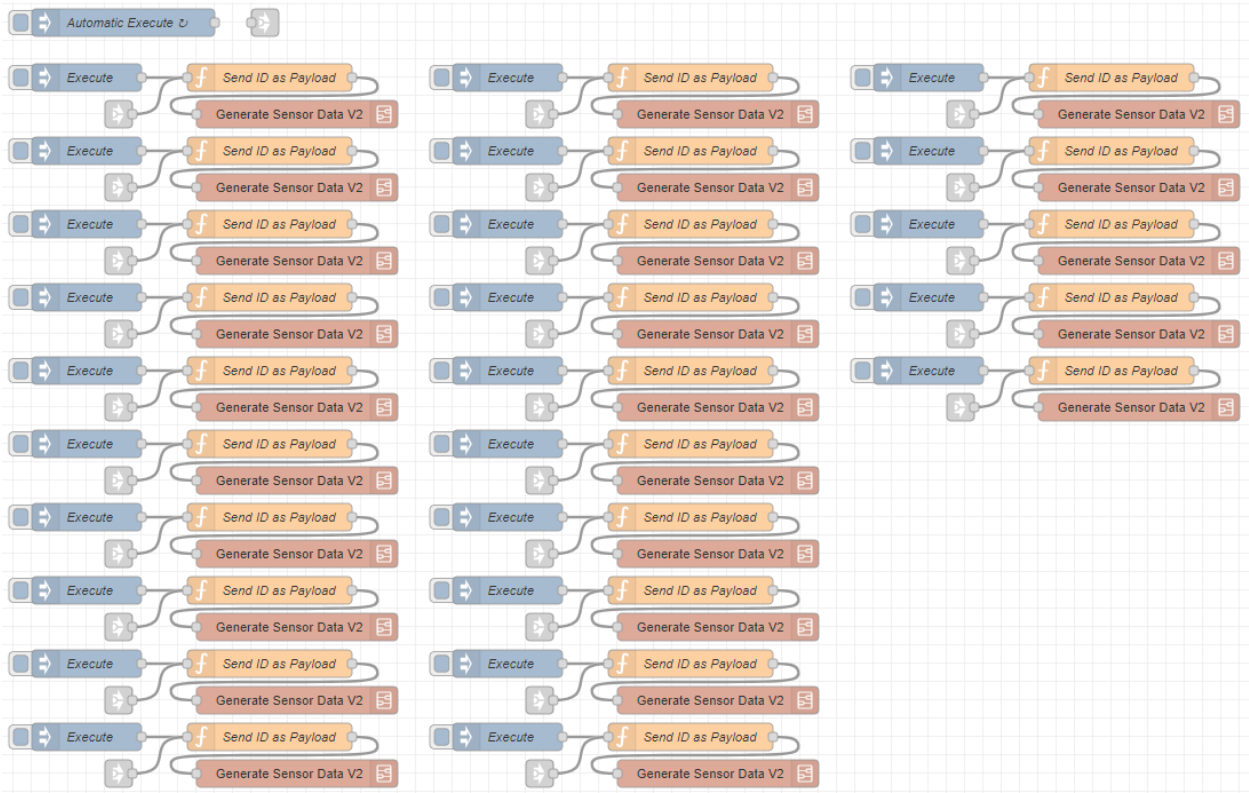

Figure 8: 25 virtual IoT devices

Step 3: Train a machine learning model using data collection from the external database. In this step, we first obtain a copy of every document within the database and obtain the sensory information. Next, the data is cleaned and pre-processed to obtain only the necessary data, then partitioned and stored into a training and testing dataset.

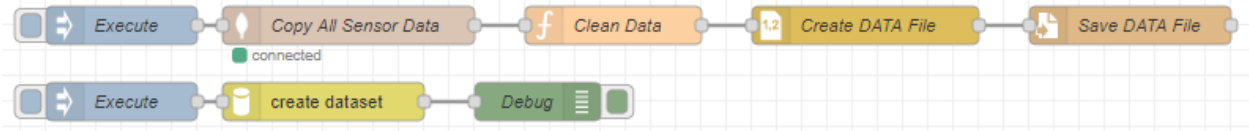

Figure 9: Flow which obtains a copy of all the data, performs pre-processing, then partitions the data into a training and testing dataset

Once the datasets are created, we train the decision tree classifier with the training dataset.

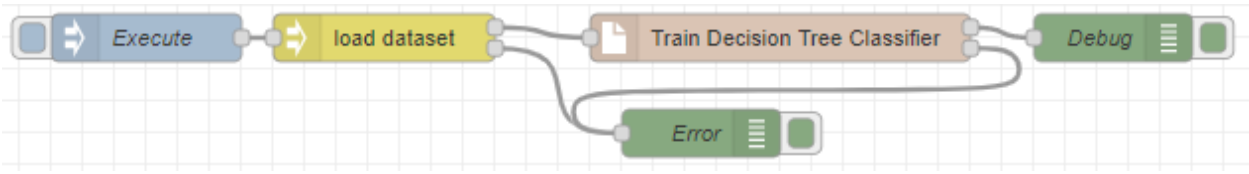

Figure 10: Loading training dataset into decision tree classifier to train it

Step 4: Test the machine learning model and asses its accuracy. We test the machine learning model with the testing dataset to verify its accuracy. This step will involve tweaking the machine learning model if the accuracy rate provided is unsatisfactory for your application.

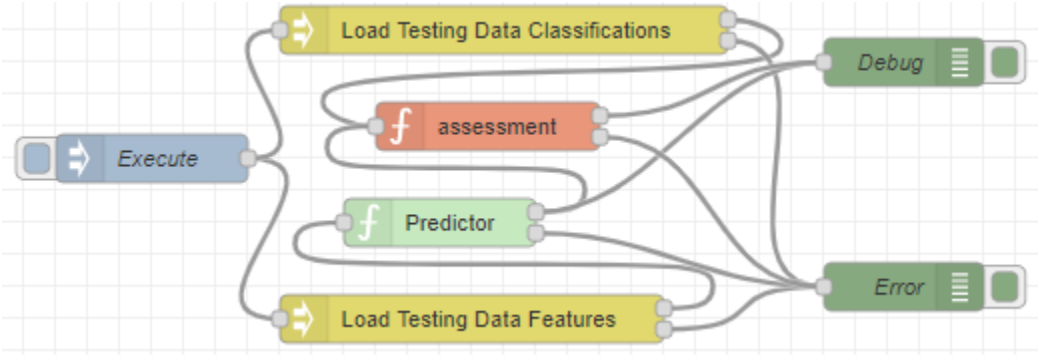

Figure 11: Flow which loads the test dataset into the model as well as within an assessment node to verify its accuracy.

Step 5: Test the machine learning model by generating new data points and verify its accuracy. In this simulation, the node generated new weather information unseen by the machine learning model and have the model predict the climate as cold, moderate, or warm. In real world applications, the predictor will need to receive information passed by the offline node's neighbors to make an assessment, as information at the device's location will not be received.