IOT AND AI BASED PREDICTION SYSTEM FOR ENVIRONMENTAL PARAMETERS USING NODEMCU AND LINEAR REGRESSION

INTRODUCTION

The necessity in monitoring the temperature and humidity variation becomes very crucial in almost all the fields, aiming to serve certain purposes from agriculture to industries. These two main environmental parameters play a major and critical role since they can affect the well-being and security of the personnel and also the cost of the product.

With the increase in electronic devices and circuitry, their heat management becomes a challenge. Most electronic components used for commercial purposes are designed to operate over a specified range of temperatures and humidity. Their variations may affect the system's performance. Hence, it is necessary to monitor and control them.

The technologies such as IoT and wireless networks have made the monitoring of the environment simple, smart and AI-controlled. So, the main objective is to facilitate an AI-based IoT system that predicts the temperature and humidity based on the real-time data collected by our sensors. This will help us to analyze the future environmental conditions and thus it can be used in various applications.

The backbone of the hardware of our system is a Wireless Sensor Networks (WSNs) that is establishing the actual interface between IoT devices and data captured through various types of smart sensors. The WSNs provide data connectivity, captured by employing various sensors and IoT devices, used to record, monitor and control various environmental conditions. To collect the real-time data, we have used a DHT11 sensor and NodeMCU, an open-source firmware and development board that has wifi capability. We programmed NodeMCU using Arduino IDE. Our system is a cloud based SEM system. So here, ThingSpeak cloud has been used to store the data. Then web scraping is done to collect data and send it to a python notebook.

To predict, we use Linear regression which is a Machine Learning Algorithm based on Supervised learning. Using the least square method we draw the regression line, a line that has the minimum vertical distance to all the points. We train our system with real-time data and then, regression models a target prediction value depending on the independent variables. Here, we have imported an ML library, scikit-learn, and other basic libraries like Numpy, Pandas, and matplotlib to perform the process. Finally, we explore the results and use them for our applications.

LITERATURE REVIEW

Temperature and humidity are two related parameters that reflect the environmental conditions. Changes in temperature causes the humidity levels to fluctuate too. As temperature increases, humidity increases as warm air holds more moisture and cool air is dry. Studies on temperature variations reveal that high temperatures are recorded typically from 3 to 4 p.m. After 4 p.m., the temperature begins to cool, as the amount of outgoing heat is more than that incoming. The lowest temperatures tend to happen just before sunrise.

The current study on advances in IoT and sensor technologies used for environmental monitoring systems provides insights in developing a framework of appropriate methods to monitor environments that are affected by poor air quality, pollutants, radiation, etc. These factors affect agriculture, industries and a lot more.

Holmstrom et al. used a linear regression model and also a variation of a functional linear regression model that outperformed professional weather forecasting services for the prediction of up to seven days. However, they work well only for forecasting later days or longer time scales.

Hybrid approaches combining discriminatively trained predictive models with deep neural networks were used for predictions by *Grover et al.* Most of the present prediction models heavily depend on complex physical models and require large computer systems that involve several High-performance Computing nodes. In our work we use a simple Linear Regression model that would provide reliable predictions.

We use real time data to train the model rather than using the available datasets, which also poses an advantage.

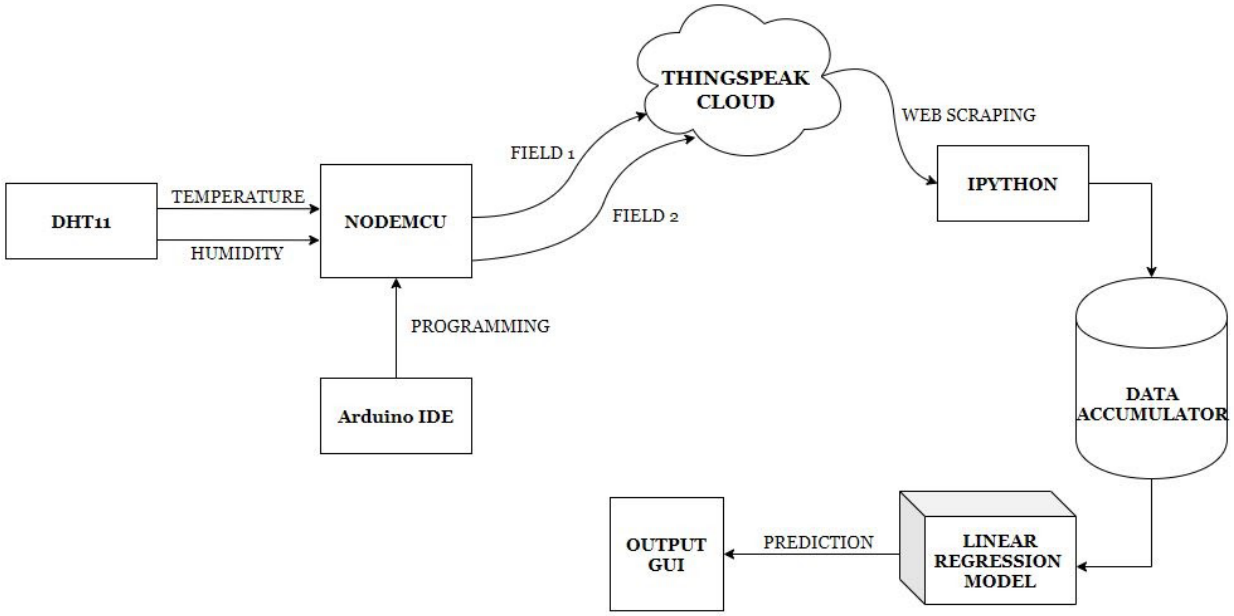
EXISTING COMPLEXITY

There are a lot of Smart Environment Monitoring (SEM) Projects out there to monitor the environmental parameters like temperature, humidity, gas level, light intensity, air pollution etc. But there are very few researches based on predicting the environmental parameters. Prediction of those parameters is crucial to detect any anomaly or abnormality and take necessary precautions in order to avoid damages. Moreover, the existing real time systems are costly and comprises invasive sensory networks.

Due to global warming and some irresponsible actions of us, there are drastic changes that occur in the climatic conditions. So, prediction of these parameters with the help of annual/monthly data would not be accurate. Hence, we propose a system that uses an accumulator to accumulate the real time data within a very short time duration so that the prediction would be reasonable and the changes that we face in the environment will be properly taken into account.

There are a few other important studies on various applications of SEM systems for different applications. But we aimed at studying the status of research on SEM using IoT, sensors and AI techniques.

PROPOSED METHODOLOGY (BLOCK DIAGRAM)



HARDWARE AND SOFTWARE USED

Hardware

- NodeMCU - ESP8266
- DHT11

Software

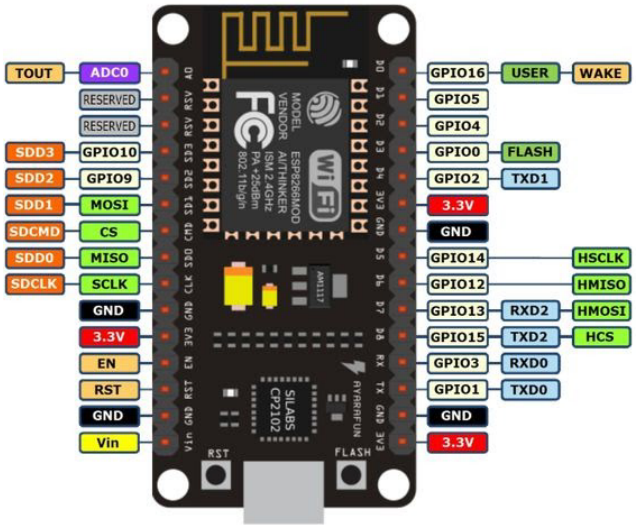
- Arduino IDE
- ThingSpeak cloud
- Google Colaboratory

Python Libraries used

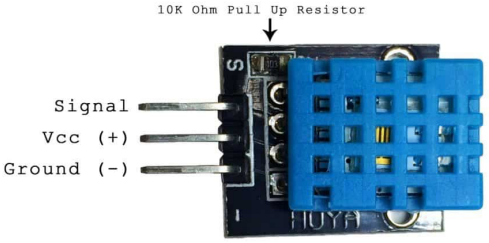
- Numpy
- Pandas
- Matplotlib
- Time
- Scikit-learn
- Urllib
- re

Arduino Libraries used

- ThingSpeak
- DHT11

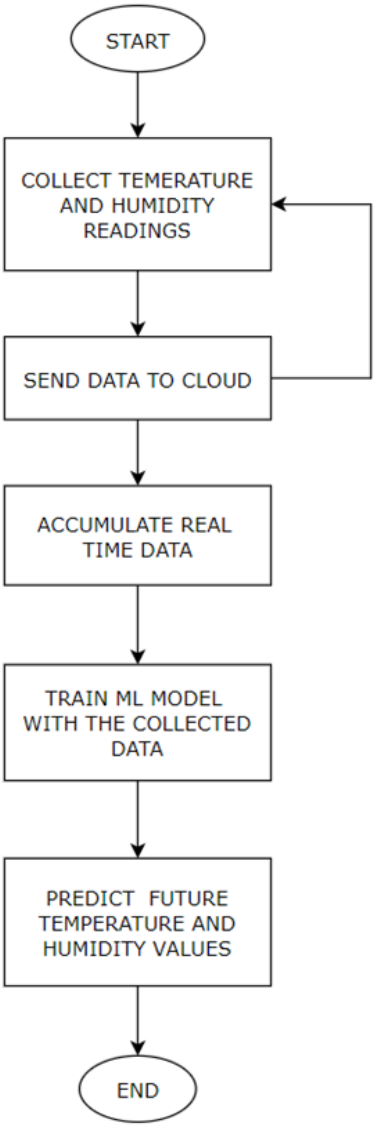


PIN DIAGRAM OF NODEMCU



PIN DIAGRAM OF DHT11

FLOWCHART



ALGORITHM

STEP 1: Get the temperature and humidity values from the sensor.

STEP 2: Send the data to ThingSpeak API through WiFi

STEP 3: Plot the data in ThingSpeak in two separate fields

STEP 4: Scrap the data using Python in Google Colab

STEP 5: Accumulate the data for a fixed duration

STEP 6: Apply pre-processing techniques to the data if necessary

STEP 7: Train the Linear Regression ML Model with the data.

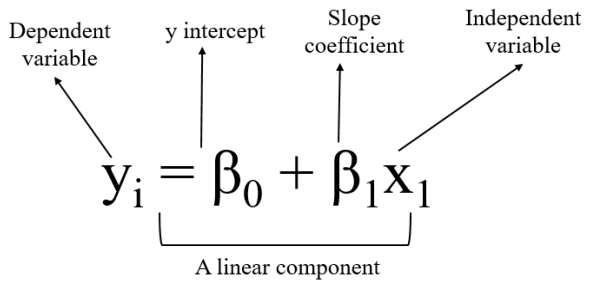
STEP 8: Test the accuracy and R2 Score

STEP 9: Predict the data in future

STEP 10: Print the predicted

WORKING

The project can be considered as two separate subprojects. The first part is interfacing a DHT11 Temperature and Humidity sensor with the NodeMCU and sending the temperature and humidity values to cloud. The other part is scraping the data from the cloud and using them to predict the temperature and humidity in future. Firstly, The DHT11 Sensor is interfaced with the ESP8266 NodeMCU. Using the Arduino IDE and the suitable libraries such as ESP8266, ThingSpeak, DHT11, the NodeMCU is programmed to send the temperature and humidity values from the sensor DHT11 to the ThingSpeak API. The NodeMCU does this job of sending the data to the cloud since it has an inbuilt WiFi connectivity. ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. Two separate fields are created in ThingSpeak for both temperature and humidity. A Python notebook has been created in Google Colab. Using a web scraping library called urllib, the data in both the fields of the ThingSpeak API is collected in the Python notebook. Accumulation of real time data is done by running a 'for loop' for a particular duration of time and the time can be set using a time library in Python. Two Machine Learning Models have been created using the Scikit-learn Library in Python. The models created are of Univariate Linear Regression type.



The models will be trained with the data that has been collected from the cloud once the loop that has been mentioned before stopped executing. The first model will be trained with temperature data and the other one will be trained with Humidity. In both the cases, the independent variable will be time. Hence, using Linear Regression approach, the models will be able to perform regression and predict the value of temperature and humidity for the future. The models are deployed in the Google Colab itself through a Graphical User Interface. The idea is to integrate the fields of Internet of Things and Artificial Intelligence.

CODE

Arduino IDE:

```
#include <ESP8266WiFi.h>;
#include <WiFiClient.h>;
#include <ThingSpeak.h>;

// Including library of DHT11 temperature and humidity sensor
#include "DHT.h"
#define DHTTYPE DHT11   // DHT 11
#define dht_dpin 0

DHT dht(dht_dpin, DHTTYPE);

// Network SSID
const char* ssid = "Redmi 2";
// Network Password
const char* password = "123456789";

WiFiClient client;

// Channel Number
unsigned long myChannelNumber = 1346212;
//Write API Key
```

```
const char * myWriteAPIKey = "OQWDX032UM0RDNRJ";



void setup()
{
    dht.begin();
    Serial.begin(9600);

    delay(10);
    // Connect to WiFi network
    WiFi.begin(ssid, password);
    ThingSpeak.begin(client);
}

void loop()
{
    float h = dht.readHumidity();
    float t = dht.readTemperature();

    //Read Analog values and Store in variable
    Serial.print("Current humidity = ");
    Serial.print(h);
    Serial.print("%  ");
    Serial.print("temperature = ");
    Serial.print(t);
    Serial.println("C  "); //Print on Serial Monitor
    delay(100);

    //Update in ThingSpeak
    ThingSpeak.writeField(myChannelNumber,1,t,myWriteAPIKey);
    ThingSpeak.writeField(myChannelNumber, 2,h, myWriteAPIKey);
    delay(100);
```

```
}
```

IPython Notebook:

1. Importing all the libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import sklearn.metrics as sn
import matplotlib.pyplot as plt
import urllib
import time
from datetime import datetime
import urllib.request
import re
```

2. Scraping the data

```
t_end = time.time() + 60 * 1
x_y1={}
x_y2={}
while time.time() < t_end:
  Time=datetime.now()
  t=Time.strftime("%H:%M:%S")
  with urllib.request.urlopen("https://api.thingspeak.com/channels/
   1346212/feeds.json?api_key=X9U2XG5ORTWDGKFG&results=2") as url:
    s1 = url.read()
    select=repr(s1)

    #Fetching Temperature Data from the API
```

```
    select=select[250:]
    print(select)
    pick1=re.search('field1":"(.+?)",',select)
    if pick1:
      x_y1[t]=pick1.group(1)

    #Fetching Humidity Data from the API
    select=select[50:100]
    pick2=re.search('field2":"(.+?)"}',select)
    if pick2:
      x_y2[t]=pick2.group(1)

#Temperature
x_1=np.array(list(x_y1.keys()))
y_1=np.array(list(x_y1.values()))

x1=[]
for i in x_1:
      x1.append(i[0]+i[1]+'.'+i[3]+i[4]+i[6]+i[7])

x_1=[float(i) for i in x1]  ##Time
x1=x_1
y1=[float(i) for i in y_1]  ##Temperature


#Humidity
x_2=np.array(list(x_y2.keys()))
y_2=np.array(list(x_y2.values()))

x2=[]
for j in x_2:
  x2.append(j[0]+j[1]+'.'+j[3]+j[4]+j[6]+j[7])
x_2=[float(j) for j in x2] ##Time
x2=x_2
```

```
y2=[float(j) for j in y_2] ##Temperature
print(x1,y1)
print(x2,y2)
len(list(x_y2.keys()))
len(y2)
```

3. Plotting Temperature vs Time

```
plt.plot(list(x_y1.keys()),y1)
plt.scatter(list(x_y1.keys()),y1)
plt.xlabel("Time")
plt.ylabel("Temperature")
plt.title("Temperature vs Time Plot")
plt.rcParams['figure.figsize'] = [1, 1]
```

4. Training the Model

```
X = np.array(x1).reshape(-1, 1)
Y = np.array(y1).reshape(-1, 1)
#df_binary.dropna(inplace = True)
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size = 0.25)

regr = LinearRegression()

regr.fit(X_train, y_train)
print(X_train, y_train)
print(X_test, y_test)
print(regr.score(X_test, y_test))
Time=184 #@param{type:"slider",min:1,max:500,step:1}
q=Time//60
r=Time%60
q,r
X[-1]=X[-1]+(0.0001*r)
X[-1]=X[-1]+(0.01*q)
```

```
X[-1]
```

5. Prediction

```
regr.predict([X[-1]])
```

6. Plotting humidity vs time

```
plt.plot(list(x_y2.keys()),y2)
plt.scatter(list(x_y2.keys()),y2)
plt.xlabel("Time")
plt.ylabel("Humidity")
plt.title("Humidity vs Time Plot")
plt.rcParams['figure.figsize'] = [2,2]
```

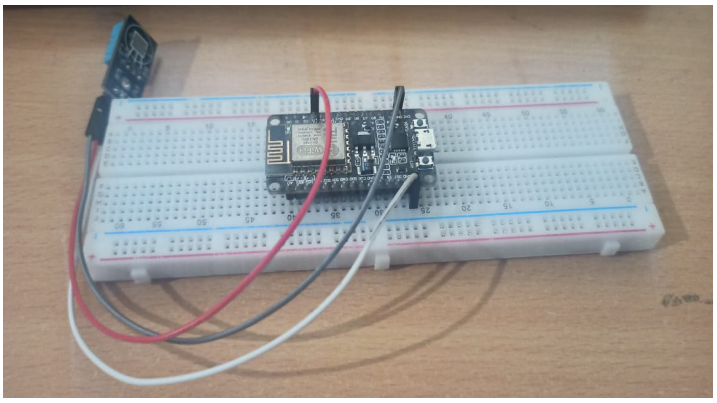
7. Training the model

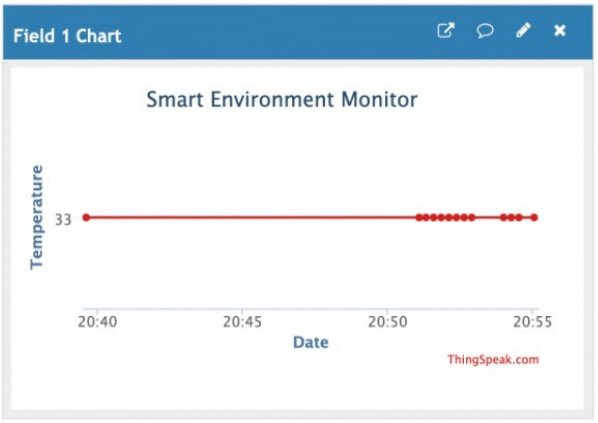
```
X = np.array(x2).reshape(-1, 1)
Y = np.array(y2).reshape(-1, 1)
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size = 0.25)
regr = LinearRegression()
regr.fit(X_train, y_train)
print(X_train, y_train)
print(X_test, y_test)
print(regr.score(X_test, y_test))
Time=180 #@param{type:"slider",min:1,max:500,step:1}
q=Time//60
r=Time%60
q,r
X[-1]=X[-1]+(0.0001*r)
X[-1]=X[-1]+(0.01*q)
X[-1]
```

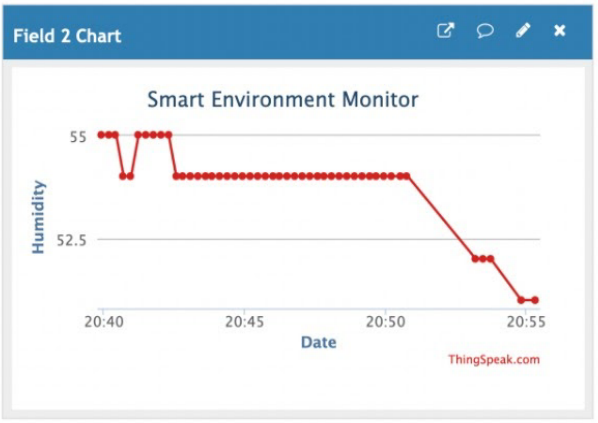
8. Prediction

```
regr.predict([x[-1]])
```

PHOTO OF THE RESULTS



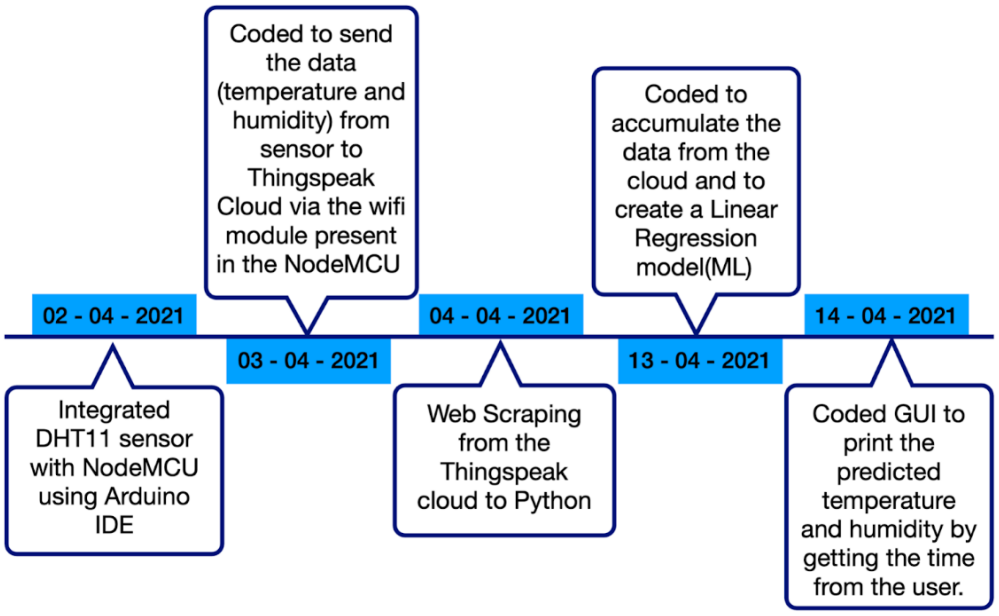




RESULTS AND INFERENCES

The real time data collected by the DHT11 sensor is used by the Machine Learning model to predict the temperature and humidity at a later time. The Linear Regression model predicts by using least squares regression line, a best fit line that depicts the linear relationship. Such models are less resource-hungry and can easily be run on almost any computer or mobile devices. It is simple, reliable and the predicted values are found to be nearly accurate. Thus these can be used in several applications to monitor the parameters and predict any abnormal variations beforehand, thereby giving us time to take precautionary measures.

TIMELINE OF THE PROJECT



APPLICATIONS

The machine learning model would be able to predict the temperature pattern and when it encounters abnormal changes in temperature or seems to exceed the permissible limits, suitable counter mechanisms can be initiated to control the temperature.

→ **Electronic Appliances**

Appliances like desktop/laptops experience increase in temperature, when there is an overload. So before the temperature goes high we can predict and make the fan in the desktop/laptops to rotate. Our model can be applied in all the electronics to prevent from overheating and to avoid severe damage.

→ **Autonomous AC**

In houses and working places, when the humidity is predicted to reduce in the future or temperature is predicted to increase in future we can turn on the Air Conditioning system before the room goes hotter.

→ **Monitor the temperature of engines in vehicles**

Can be placed in the Engines of vehicles like cars, bikes, bus, etc. to do when there is a small hype in temperature is predicted, we can make the engine coolant system to activate by which we can avoid overheating and some chances of engine failure(engine seized)

→ **Motor related appliances**

Mostly many appliances like DC motors used in water pumping systems, Mixie, Grinder, etc. use motors for its functionality. Since it rotates continuously due to its physical properties there are several chances of temperature hype so we can predict and turn off the motor before the motor is damaged.

→ **Weather forecasting**

We don't know what the weather will be in the next minute or next second, even though we know the temperature of the next day. By our model we can predict the temperature and humidity with foremost accuracy.

→ **Humidity monitoring in Godown**

In food storing godowns, the products should be maintained in a proper temperature and humidity, when the humidity goes high it may lead to decay of food products. So our model can be helpful in predicting the high humidity before it reaches high by which we can preserve the food products.

FUTURE SCOPE OF THE PROJECT

- Our model can be built as a PCB so that it can be easily attached to all the electronic and other devices.

- Miniaturisation of our model can be placed in a wristband or wristwatch. Consider when the temperature of the body is increased form 98C to 99C, when our model predicts the further rise in temperature so that we can consult a doctor in advance before the situation gets worse.

- In weather forecasting, by constructing many models we can place them in many places/cities from which we can calculate the accurate temperature and humidity.

REFERENCES

- https://www.comptus.com/blog

- Aditya Grover, Ashish Kapoor, and Eric Horvitz. 2015. A deep hybrid model for weather forecasting. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 379–386

- Mark Holmstrom, Dylan Liu, and Christopher Vo. 2016. Machine Learning Applied to Weather Forecasting

- A. Hammami, "Smart Environment Data Monitoring," 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 2019, pp. 1-6, doi: 10.1109/ICCISci.2019.8716469.