IoT-Based Weather Station and Rainfall Prediction with ML

Akshaj Rai
Electronics and Communication Engineering
Nirma University
Ahmedabad, India
20bec099@nirmauni.ac.in

Sanchit Sharma
Electronics and Communication Engineering
Nirma University
Ahmedabad, India
20bec108@nirmauni.ac.in

***Abstract*—This paper presents a way to Integrate IoT and Machine Learning in order to create an accurate model capable of predicting the possibility of rainfall based on various parameters such as Humidity,temperature,Altitude,Atmospheric Pressure,etc. The Random Forest classifier has been used to predict the rainfall based on these parameters**

***Index Terms*—Rainfall Prediction, Random Forest, Machine Learning, ESP8266**

I. Introduction

Rainfall plays an important role in our lives and facilitates various occupations,living conditions and our livelihood.In this paper we have presented a way to combine Internet of Things with Machine Learning in order to predict rainfall.We have used ESP8266 as the microcontroller for our Project and have used various sensors such as DHT11,YL83 and BMP 280 to collect information about the weather such as temperature,humidity.etc. We have used a Kaggle Dataset to train a model using Random Forest Classifier and have used the real time data generated using the aforementioned sensors and ThingSpeak platform. We have also interfaced the realtime sensor readings with Blynk IoT, which can be effectively used as a Weather Station.We have observed that the trained model can predict with an accuracy of upto 83.21 percent.

II. Block Diagram

In order to implement a functioning weather station and prediction algorithm, we have followed the following Block diagram:

As depicted above, the sensors work in conjunction with the ESP8266 in order to measure real time environmental conditions.The wifi module of the ESP8266 is uesed to tansmit this data to cloud platform ThingSpeak which is used to generate .csv files of the observed data over a period of time.Similarly, in oder to create an effective weather monitoring system, Blynk IoT too has been interfaced which can be used to broadcast the data. Next, a preobtained dataset available from kaggle is used to train and test the model using the Random Forest classifier which predicts the probability of rainfall the next day.It has been found that the designed model's accuracy is 83.21 percent which is in accordance of industry specified standards.

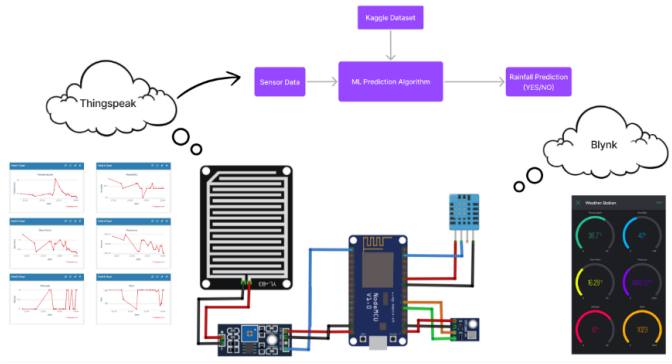

Fig. 1. Block Diagram of the implemented system

III. Working

A. Components Used

The central unit of the circuit is the NodeMCU ESP8266. The ESP8266 ESP-01 is a Wi-Fi module that allows microcontrollers access to a Wi-Fi network. This module is a self-contained SOC (System On a Chip) that doesn't necessarily need a microcontroller to manipulate inputs and outputs.Depending on the version of the ESP8266, it is possible to have up to 9 GPIOs (General Purpose Input Output). Thus, we can give a microcontroller internet access like the Wi-Fi shield does to the Arduino, or we can simply program the ESP8266 to not only have access to a Wi-Fi network, but to act as a microcontroller as well.It must be noted that the wifi connectivity offered by the ESP 8266 is essential as it helps to establish a cloud connectivity, Client-server model with ThingSpeak and Blynk IoT platforms

In order to collect the realtime data of the environment we have used various sensors - DHT11 (For Temperature and Humidity measurement), the BMP 280(used for measuring Atmospheric Pressure and Altitude) and the YL-83 rainfall sensor for collecting insights about the volume of precipitation.

The DHT 11 is used for measuring Temperature and Humidity.It Operates by using a humidity-sensitive capacitor and a thermistor to measure humidity and temperature The sensor converts analog measurements into digital readings that can be fed into and be analysed by the ESP8266 microcontroller

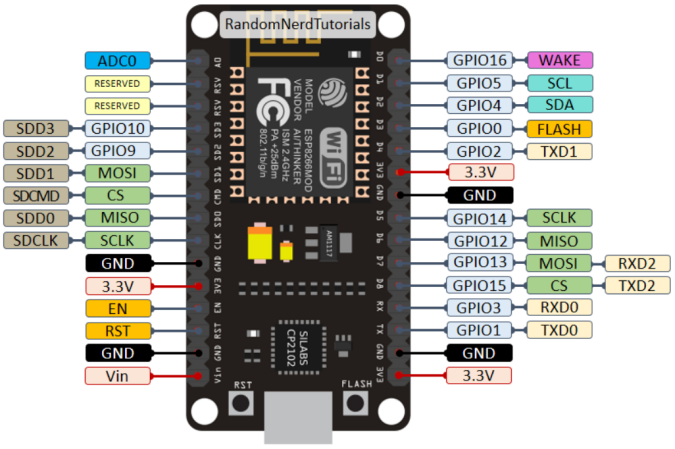

Fig. 2. The ESP8266 microcontroller and it's pin diagram

which has been interfaced with the sensor using digital GPIO pins.

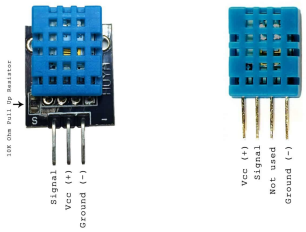

Fig. 3. The DHT-11 Sensor

The BMP 280 is a sensor used to measure the pressure and altitude of a place. Measures atmospheric pressure by monitoring the resistance changes in its piezoresistive pressure sensor as the diaphragm flexes with pressure variations and altitude calculation based on the barometric formula. The BMP can be interfaced with the NodeMCU using I2C as well as the SPI protocol.For this project we have used it by I2C interfacing.

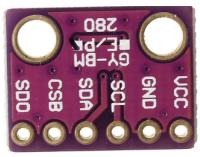

Fig. 4. The BMP 280 sensor

The YL-83 sensor is used for measuring volume of precipitatation in order to calculate the rainfall on a particular day.The sensor consists of a conductive metal plate, which when in contact with water provides a low resistance path for the currrent thus lowering its voltage drop, which can be used to determine the variation in rainfall.The module can detect this change and signal the presence of rain.

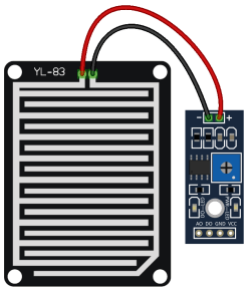

Fig. 5. The YL-83 sensor

B. ThingSpeak and BlynkIOT interfacing

In order to facilitate the design of a working weather station, it is essential that the data obtained from the sensors be transmitted to cloud platforms with minimum delay.ThingSpeak has been interfaced with the NodeMCU using various functions and the wi-fi module of the ESP8266.Thingspeak performs the very crucial task of plotting the variation of the physical parameters such as temperature,Humidity,Pressure,Altitude,Dew Point and the output voltage of the YL-83 sensor and generates a dataset of these values and updates it accordingly.

Fig. 6. Parameter variation as observed on ThingSpeak

Blynk IoT on the other hand provides the full functionality of a weather monitoring system and can be used to broadcast the real time environmental conditions and can be even used in times of natural disasters such as floods to address a large number of devices or users.Using the Blynk Application we have designed dials depicting the variance of these parameters obtained from sensor readings.

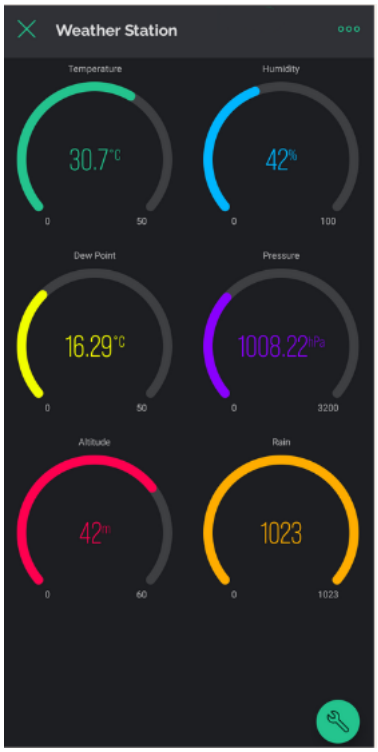

Fig. 7. Blynk IoT results

IV. Rainfall Prediction

A. Dataset

In order to create a function model that can accurately predict the possibility of rainfall, we attained the dataset available on Kaggle,This dataset comprised of many attributes such as :

- Min Temp
- Max Temp
- Humidity
- Pressure
- Temperature
- Rain Today
- Evaporation
- WindSpeed
- Cloud
- Sunshine
- Wind Direction

For our model, we have selected few attributes such as :

- Min Temp
- Max Temp
- Humidity
- Pressure
- Temperature
- Rain Today

We have dropped the remaining attributes in order to optimize the cost-performance ratio of the implemented model. Following this,we have cleaned the data and standardized it, followed by One-Hot encoding to convert non categorical data into a boolean data format that could be analyzed by the machine learning model.

B. Random Forest Classifier

Random forest is a supervised learning algorithm. The "forest" it builds is an ensemble of decision trees, usually trained with the bagging method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in a random forest classifier, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

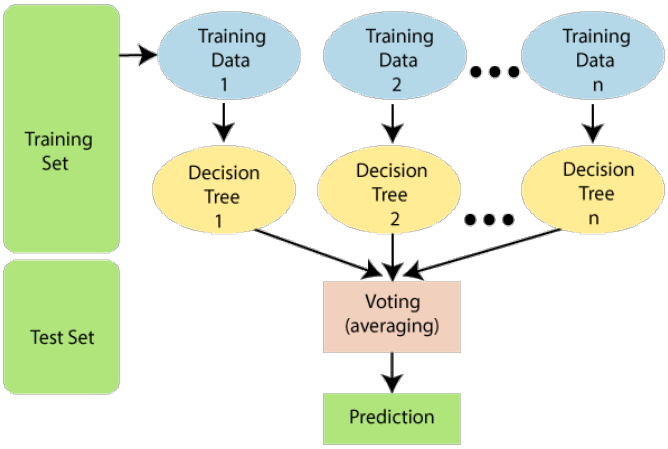

Fig. 8. Random Forest Algorithm

The Kaggle dataset has been cleaned and is then split into training samples and test samples using python libraries,the training sets are then used to create multiple decision trees which are analyzed by voting to find the majority and predict the result.The Test data is fed into the trained model to find the accuracy of the designed model.It is observed that the accuracy of the model is upto 83.21 percent.

It must be noted that the data from sensor YL-83 has been normalized to the range 0-1023, where 1023 indicates no rain (absolutely none) and the closer the digital output is to 0, the higher is the volume of the precipitation. In order to convert the non categorical data into categorical, the limit of 750 has been set to indicate the presence or absence of rain.

C. Random Forest Classifier vs Decision Trees

Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random

forest. With random forest, you can also deal with regression tasks by using the algorithm's regressor.

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in a random forest classifier, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

D. Final Results

In order to predict the rainfall probability based on realtime data generated by aforementioned sensors and ThingSpeak, we extract the data from the CSV file and give it as an input to the Random Forest Classifier which returns the appropriate result based as the following image:

Fig. 9. Prediction Flow

V. Future Scope

While the aforementioned model has demonstrated a commendable accuracy of up to 83.21 percent, there exists an opportunity to enhance its predictive capabilities by incorporating superior sensors and expanding the range of collected parameters. The integration of advanced sensors measuring additional environmental factors, including evaporation, wind speed, cloud cover, sunshine duration, and wind direction, would contribute to a more comprehensive dataset. By incorporating these nuanced variables, our model could capture a more intricate understanding of the atmospheric conditions, potentially resulting in a refined and more accurate rainfall prediction. This expansion not only allows for a more sophisticated analysis of weather patterns but also opens avenues for broader applications across sectors like agriculture, climate research, and urban planning, where a nuanced understanding of various environmental parameters is critical for informed decision-making.

VI. Conclusion

We have successfully developed and implemented an IoT model designed for environmental data measurement with a focus on predicting the probability of rainfall for the following day. Utilizing a Random Forest Classifier, our model demonstrates high accuracy in forecasting. Leveraging Cloud Services like ThingSpeak and Blynk, we've established seamless remote access to real-time and historical data. This integration allows users to remotely monitor environmental conditions, providing valuable insights and aiding in decision-making processes related to weather-sensitive activities. The cloud-based architecture enhances accessibility and ensures the reliability of our IoT system, making it a versatile and effective tool for various applications, from agriculture to urban planning.

Acknowledgement

We extend our deepest appreciation to Dr. Viranchi Pandya, our esteemed mentor, for their unwavering guidance and invaluable assistance throughout the duration of this project. Their knowledge and insights helped en- sure for the successful completion of this project. Furthermore, we would like to express our gratitude to all the supporting personnel who provided us with the necessary equipment and resources, which were essential in executing this project efficiently. Without their contributions, it would have been impossible to achieve the desired results. We acknowledge and value the efforts of all those who played a part in this project, and we are truly grateful for their support and cooperation. Their contributions have enabled us to produce a paper of exceptional quality and significance, which we are proud to present as a testament to our collective efforts.

References

[1] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[2] https://www.instructables.com/Blynk-With-ESP8266/
[3] https://www.instructables.com/ThingSpeak-Using-ESP8266/
[4] https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf
[5] https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors/bmp280/
[6] https://www.scribd.com/document/403555524/YI-83-Rain-Detector-Datasheet-Low

APPENDIX

ESP8266-NodeMCU Code

```
#include <DHT.h>  // HEADER FILES
#include <ESP8266WiFi.h>
#include <Adafruit_BMP280.h>
#include <Wire.h>
#define BLYNK_TEMPLATE_ID "TMPL3asbf1X2b"
#define BLYNK_TEMPLATE_NAME "Weather Station"
#define BLYNK_AUTH_TOKEN "2MAjMdY2TDdh2BrynHuoMGE4rz3MJS12"
#include <Blynk.h>
#include <BlynkSimpleEsp8266.h>



#define DHTPIN 0
DHT dht(DHTPIN, DHT11);
Adafruit_BMP280 bmp;
#define SEALEVELPRESSURE (1013.25)
int sensorPin = A0;
int sensorValue2 = 0;  // variable to store the value coming from sensor Rain sensor
#define ALTITUDE 80.0 // Altitude of SparkFun's HQ in Boulder, CO. in meters
String apiKey = "IUTG165MGR8T65OL";//  Enter your Write API key from ThingSpeak
long myChannelNumber = 2336546;

const char *ssid = "Hey,GetYourOwnWi-Fi";
const char *pass = "12345678";
const char* server = "api.thingspeak.com";

WiFiClient client;
```

```
void setup()
{
  Serial.begin(115200); // open serial port, set the baud rate to 9600 bps
  delay(10);
  dht.begin();
  Serial.println("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);
  while(WiFi.status() != WL_CONNECTED)
  {
  delay(200);
  Serial.print("..");
  }
  Serial.println();
  Serial.println("ESP8266 is connected!");
  Serial.println(WiFi.localIP());

  // Initialize the sensor (it is important to get calibration values stored on the device).

  if (bmp.begin(0x76))
  {
    Serial.println("BMP280 init success");
  }
  else
  {
    // Oops, something went wrong, this is usually a connection problem,
    // see the comments at the top of this sketch for the proper connections.
```

```
    Serial.println("BMP280 init fail\n\n");
  }
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

}

void loop()
{
  Blynk.run();

      float h = dht.readHumidity();
      float t = dht.readTemperature();
      float dewPoint = dewPointFast(t, h);
      Serial.println("*********************");
      Serial.println("DHT11 sensor values:");
                              Serial.print("Temperature: ");
      Serial.print(t);
      Serial.print(" Degrees Celcius, Humidity: ");
      Serial.print(h);
      Serial.print(" %, DewPoint: ");
      Serial.print(dewPoint);
      Serial.print(" degrees Celsius ");


      const int sensorMin = 150; //0;     // sensor minimum
      const int sensorMax = 440; //1024;  // sensor maximum

      delay(500);
      float sensorValue2;
```

```
sensorValue2 = analogRead(sensorPin);
sensorValue2 = constrain(sensorValue2, 150, 440); //150, 400
sensorValue2 = map(sensorValue2, sensorMin, sensorMax, 0, 1023);  //150, 440
Serial.println("**********");
Serial.println("Rain sensor values:");
Serial.print("Rain value: ");
Serial.print(sensorValue2);
Serial.println();
delay(100);

float p,a;

Serial.println("**********");
Serial.println("BMP280 sensor values:");

p = bmp.readPressure() / 100.0F;
Serial.print("Pressure: ");
Serial.print(p,2);
Serial.print("hPa, ");

a = bmp.readAltitude(SEALEVELPRESSURE);
Serial.print("Altitude: ");
Serial.print(a,0);
Serial.print(" meters, ");
Serial.println();


if (client.connect(server,80))   //   "184.106.153.149" or api.thingspeak.com
  {
```

```
 String postStr = apiKey;
 postStr +="&field1=";
 postStr += String(t);//temperature
 postStr +="&field2=";
 postStr += String(h);//humidity
 postStr += "&field3=";
 postStr += String(dewPoint);//dew point
 postStr += "&field4=";
 postStr += String(p);//pressure
 postStr += "&field5=";
 postStr += String(a,0);//altitude
 postStr +="&field6=";
 postStr += String(sensorValue2);//rain
 postStr += "\r\n\r\n\r\n\r\n";

 client.print("POST /update HTTP/1.1\n");
 client.print("Host: api.thingspeak.com\n");
 client.print("Connection: close\n");
 client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
 client.print("Content-Type: application/x-www-form-urlencoded\n");
 client.print("Content-Length: ");
 client.print(postStr.length());
 client.print("\n\n");
 client.print(postStr);

 }
client.stop();
Serial.print("Waiting...");
Serial.println("");
Serial.println("==========================");
```

```
  delay(5000);

  Blynk.virtualWrite(V0, t);  //V0 is for Temperature
  Blynk.virtualWrite(V1, h);  //V1 is for Humidity
  Blynk.virtualWrite(V2, dewPoint);  //V3 is for Dew Point
  Blynk.virtualWrite(V3, p);  //V4 is for Pressure
  Blynk.virtualWrite(V4, a);  //V5 is for Altitude
  Blynk.virtualWrite(V5, sensorValue2);  //V6 is for Rainfall
}

  double dewPointFast(double celsius, double humidity)
 {
   double a = 17.271;
   double b = 237.7;
   double temp = (a * celsius) / (b + celsius) + log(humidity * 0.01);
   double Td = (b * temp) / (a - temp);
   return Td;
 }
```

Rainfall Prediction using Random Forest Classifier

Importing Libraries

In [1]:

```
import urllib.request as urllib2
import json
import time
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [2]:

```
import warnings
warnings.filterwarnings("ignore")
```

Reading the Dataset

In [3]:

```
ds=pd.read_csv("weatherAUS.csv")
```

In [4]:

```
ds.head()
```

Out[4]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Hum
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	

5 rows × 23 columns

Cleaning the Dataset

In [5]:

```
ds = ds.drop(columns = ['Location','Evaporation', 'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am','WindDir3pm','WindSpeed9am', 'WindSpeed3pm', 'Cloud9am', 'Cloud3pm'])
```

In [6]:

```
ds.head()
```

Out[6]:

	Date	MinTemp	MaxTemp	Rainfall	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Temp9am	Temp3pm	RainT
0	2008-12-01	13.4	22.9	0.6	71.0	22.0	1007.7	1007.1	16.9	21.8	
1	2008-12-02	7.4	25.1	0.0	44.0	25.0	1010.6	1007.8	17.2	24.3	
2	2008-12-03	12.9	25.7	0.0	38.0	30.0	1007.6	1008.7	21.0	23.2	
3	2008-12-04	9.2	28.0	0.0	45.0	16.0	1017.6	1012.8	18.1	26.5	
4	2008-12-05	17.5	32.3	1.0	82.0	33.0	1010.8	1006.0	17.8	29.7	

In [7]:

```
ds['Humidity'] = ds[['Humidity9am', 'Humidity3pm']].mean(axis=1)
ds['Pressure'] = ds[['Pressure9am', 'Pressure3pm']].mean(axis=1)
ds['Temperature'] = ds[['Temp9am', 'Temp3pm']].mean(axis=1)
ds = ds.drop(columns = ['Humidity9am', 'Humidity3pm','Pressure9am', 'Pressure3pm','Temp9
am', 'Temp3pm'])
new_cols = ['Date','MinTemp', 'MaxTemp', 'Humidity', 'Pressure', 'Temperature', 'RainToda
y', 'RainTomorrow']
ds = ds.reindex(columns=new_cols)
ds.head()
```

Out[7]:

	Date	MinTemp	MaxTemp	Humidity	Pressure	Temperature	RainToday	RainTomorrow
0	2008-12-01	13.4	22.9	46.5	1007.40	19.35	No	No
1	2008-12-02	7.4	25.1	34.5	1009.20	20.75	No	No
2	2008-12-03	12.9	25.7	34.0	1008.15	22.10	No	No
3	2008-12-04	9.2	28.0	30.5	1015.20	22.30	No	No
4	2008-12-05	17.5	32.3	57.5	1008.40	23.75	No	No

In [8]:

```
ds.isnull().sum()
```

Out[8]:

```
Date                0
MinTemp          1485
MaxTemp          1261
Humidity         1887
Pressure        14804
Temperature      1129
RainToday        3261
RainTomorrow     3267
dtype: int64
```

In [9]:

```
ds = ds.dropna()
```

In [10]:

```
ds.isnull().sum()
```

Out[10]:

```
Date           0
MinTemp        0
MaxTemp        0
Humidity       0
Pressure       0
```

```
Temperature     0
RainToday       0
RainTomorrow    0
dtype: int64
```

Splitting the Dependent and Independent Variables

In [11]:

```
x = ds.iloc[:,1:7]
display(x.head())
```

	MinTemp	MaxTemp	Humidity	Pressure	Temperature	RainToday
0	13.4	22.9	46.5	1007.40	19.35	No
1	7.4	25.1	34.5	1009.20	20.75	No
2	12.9	25.7	34.0	1008.15	22.10	No
3	9.2	28.0	30.5	1015.20	22.30	No
4	17.5	32.3	57.5	1008.40	23.75	No

In [12]:

```
y = ds.iloc[:,7]
display(y.head())
```

```
0    No
1    No
2    No
3    No
4    No
Name: RainTomorrow, dtype: object
```

One Hot Encoding

In [13]:

```
x_encoded = pd.get_dummies(x, columns=['RainToday'], drop_first=True)
x_encoded.rename(columns = {'RainToday_Yes':'RainToday'}, inplace = True)
x_encoded.head()
```

Out[13]:

	MinTemp	MaxTemp	Humidity	Pressure	Temperature	RainToday
0	13.4	22.9	46.5	1007.40	19.35	0
1	7.4	25.1	34.5	1009.20	20.75	0
2	12.9	25.7	34.0	1008.15	22.10	0
3	9.2	28.0	30.5	1015.20	22.30	0
4	17.5	32.3	57.5	1008.40	23.75	0

In [14]:

```
y_encoded = pd.get_dummies(y, columns=['RainTomorrow'], drop_first=True)
y_encoded.rename(columns = {'Yes':'RainTomorrow'}, inplace = True)
y_encoded.head()
```

Out[14]:

	RainTomorrow
0	0
1	0

	RainTomorrow
2	0
3	0
4	0

Splitting the Train and Test Data

In [15]:

```
x_train, x_test, y_train, y_test = train_test_split(x_encoded, y_encoded, test_size = 0.
20, random_state=0)
```

Model Building

In [16]:

```
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier()
model_rf.fit(x_train,y_train)
```

Out[16]:

RandomForestClassifier

RandomForestClassifier()

Predicting using Test Data

In [17]:

```
pred = model_rf.predict(x_test)
```

In [18]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, pred, normalize = True)
```

Out[18]:

0.8310695758919733

Predicting for the Actual Data from the Sensors

In [19]:

```
sensor_data = pd.read_csv("feeds.csv")
```

In [20]:

```
sensor_data.head()
```

Out[20]:

	created_at	entry_id	field1	field2	field3	field4	field5	field6	latitude	longitude	elevation	status
0	2023-11-07T18:13:33+00:00	1	31.0	42	16.56	1008.66	42	1023.0	NaN	NaN	NaN	NaN
1	2023-11-07T18:15:41+00:00	2	36.2	32	16.85	1008.62	42	1023.0	NaN	NaN	NaN	NaN
2	2023-11-07T18:20:47+00:00	3	31.0	43	16.93	1008.48	42	1023.0	NaN	NaN	NaN	NaN
3	2023-11-07T18:21:13+00:00	4	31.0	43	16.93	1008.65	42	1023.0	NaN	NaN	NaN	NaN
4	2023-11-07T18:22:10+00:00	5	31.0	43	16.93	1008.54	42	1023.0	NaN	NaN	NaN	NaN

In [21]:

```
sensor_data = sensor_data.drop(columns = ['latitude','longitude', 'elevation', 'status']
)
```

In [22]:

```
sensor_data.head()
```

Out[22]:

	created_at	entry_id	field1	field2	field3	field4	field5	field6
0	2023-11-07T18:13:33+00:00	1	31.0	42	16.56	1008.66	42	1023.0
1	2023-11-07T18:15:41+00:00	2	36.2	32	16.85	1008.62	42	1023.0
2	2023-11-07T18:20:47+00:00	3	31.0	43	16.93	1008.48	42	1023.0
3	2023-11-07T18:21:13+00:00	4	31.0	43	16.93	1008.65	42	1023.0
4	2023-11-07T18:22:10+00:00	5	31.0	43	16.93	1008.54	42	1023.0

In [23]:

```
sensor_data.rename(columns = {'created_at':'Data and Time','entry_id':'Sr No.','field1':
'Temperature', 'field2':'Humidity','field3':'Dew Point','field4':'Pressure','field5':'Alt
itude','field6':'Rainfall'},inplace = True)
sensor_data.head()
```

Out[23]:

	Data and Time	Sr No.	Temperature	Humidity	Dew Point	Pressure	Altitude	Rainfall
0	2023-11-07T18:13:33+00:00	1	31.0	42	16.56	1008.66	42	1023.0
1	2023-11-07T18:15:41+00:00	2	36.2	32	16.85	1008.62	42	1023.0
2	2023-11-07T18:20:47+00:00	3	31.0	43	16.93	1008.48	42	1023.0
3	2023-11-07T18:21:13+00:00	4	31.0	43	16.93	1008.65	42	1023.0
4	2023-11-07T18:22:10+00:00	5	31.0	43	16.93	1008.54	42	1023.0

In [24]:

```
min_temp = sensor_data['Temperature'].min()
print('Min_Temp',min_temp)

max_temp = sensor_data['Temperature'].max()
print('Max_Temp:',max_temp)

humidity = sensor_data['Humidity'].mean()
print('Humidity:', humidity)

pressure = sensor_data['Pressure'].mean()
print('Pressure:',pressure)

temp = sensor_data['Temperature'].mean()
print('Temperature:',temp)

if(sensor_data['Rainfall'].mean()<750):
    rain_today = 1
    print('Rainfall Today: Yes')
else:
    rain_today = 0
    print('Rainfall Today: No')
```

```
Min_Temp 30.8
Max_Temp: 36.2
Humidity: 42.84444444444444
Pressure: 1008.0068888888886
Temperature: 31.164444444444445
Rainfall Today: No
```

In [25]:

In [25]:

```
# test_data = [[13.4,30.4,35.0,1010.25,24.6,0]]
test_data = [[min_temp,max_temp,humidity,pressure,temp,rain_today]]
rain_predict = model_rf.predict(test_data)
if(rain_predict == 1):
    print("Yes it will rain tomorrow")
else:
    print("No it won't rain tomorrow")
```

```
No it won't rain tomorrow
```