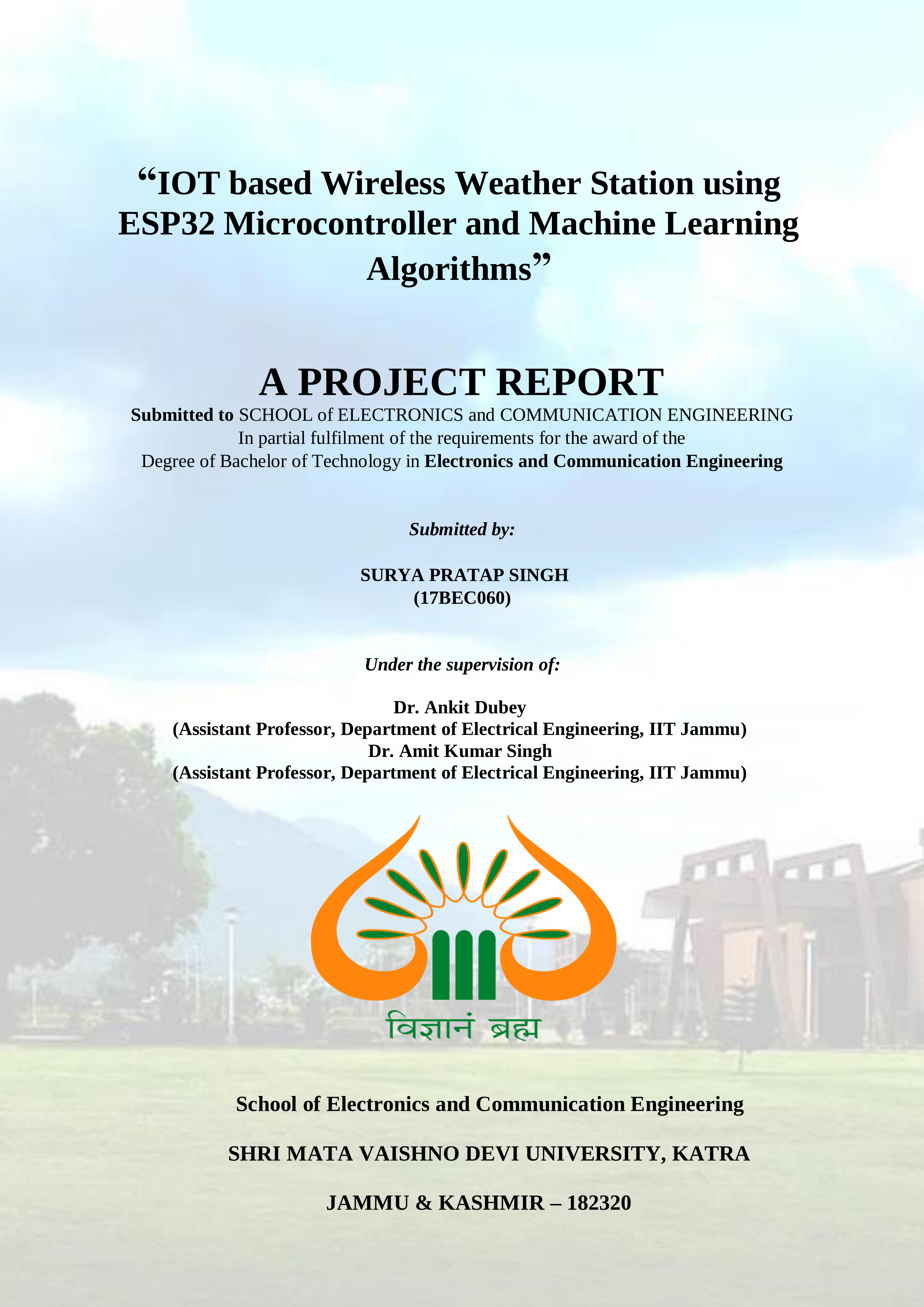


“IOT based Wireless Weather Station using ESP32 Microcontroller and Machine Learning Algorithms”

A PROJECT REPORT

Submitted to SCHOOL of ELECTRONICS and COMMUNICATION ENGINEERING
In partial fulfilment of the requirements for the award of the
Degree of Bachelor of Technology in **Electronics and Communication Engineering**

Submitted by:

SURYA PRATAP SINGH
(17BEC060)

Under the supervision of:

Dr. Ankit Dubey
(Assistant Professor, Department of Electrical Engineering, IIT Jammu)
Dr. Amit Kumar Singh
(Assistant Professor, Department of Electrical Engineering, IIT Jammu)

विज्ञानं ब्रह्म

School of Electronics and Communication Engineering

SHRI MATA VAISHNO DEVI UNIVERSITY, KATRA

JAMMU & KASHMIR – 182320

SHRI MATA VAISHNO DEVI UNIVERSITY
School of Electronics & Communication Engineering

STUDENT DECLARATION

I hereby declare that the work which is presented in the B. Tech Project Report entitled "**IOT based Wireless Weather Station using ESP32 Microcontroller and Machine Learning Algorithm**", in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering and submitted to the School of Electronics & Communication Engineering, Shri Mata Vaishno Devi University, Katra, J&K is an authentic record of our own work which has carried out during a period from January, 2021 to May, 2021 at Indian Institute Of Technology, Jammu under the guidance of Dr. Ankit Dubey (Assistant Professor, Department of Electrical Engineering, IIT Jammu) and Dr. Amit Kumar Singh (Assistant Professor, Department of Electrical Engineering, IIT Jammu). The matter presented in this report has not been submitted elsewhere by me for the award of any other degree.

[illegible]

Surya Pratap Singh
(17BEC060)

विज्ञानं ब्रह्म

SHRI MATA VAISHNO DEVI UNIVERSITY
School of Electronics & Communication Engineering

CERTIFICATE

This is to certify that the Project entitled **"IOT based Wireless Weather Station using ESP32 Microcontroller and Machine Learning Algorithm"** being submitted by Surya Pratap Singh (17BEC060) to the Department of Electrical Engineering, Indian Institute of Technology, Jammu under the supervision and guidance of Dr. Ankit Dubey (Assistant Professor) and Dr. Amit Kumar Singh (Assistant Professor) And to the School of Electronics And Communication Engineering, Shri Mata Vaishno Devi University, Katra is completed under the guidance of Mr. Ashish Suri. The report has reached the standard of fulfilling of the requirement of the regulation related to degree.

We wish the best of his endeavor.

[illegible]

Surya Pratap Singh
(17BEC060)

Project Supervisors

[illegible]

Dr. Ankit Dubey
Assistant Professor
Department of Electrical Engineering,
IIT Jammu

Amit Kumar

Dr. Amit Kumar Singh
Assistant Professor
Department of Electrical Engineering,
IIT Jammu

Project Guide

Ashish Suri

Mr. Ashish Suri
Assistant Professor
School of Electronics and Communication
SMVDU, Katra

Head of Department

Dr. Manish Sabraj
Head of Department
School of Electronics and Communication
SMVDU, Katra

विज्ञानं ब्रह्म

SHRI MATA VAISHNO DEVI UNIVERSITY
School of Electronics & Communication Engineering

<u>PLAGIARISM CERTIFICATE</u>

It is certified that Mr. Surya Pratap Singh S/o Sh. Manjit Singh Entry. No. 17BEC060 has carried out her project work on the topic entitled "**IOT based Wireless Weather Station using ESP32 Microcontroller and Machine Learning Algorithm**". The complete report has been checked by the Turnitin Software. I have reviewed the report of the plagiarism software and the similarity index is 8% i.e., below the accepted norms of the University. The thesis may be considered for the award of the degree.

Ashish Suri

Mr. Ashish Suri
Supervisor
Assistant Professor, School of Electronics & Communication Engineering
Shri Mata Vaishno Devi University, Katra-182320, Jammu and Kashmir (India)

Forwarded by: Surya Pratap Singh

Signature of Chairperson with Seal

ACKNOWLEDGEMENT

“An endeavor over long period can be successful only with advice and guidance of many well-wishers.”

I feel immense pleasure in expressing my sincere thanks and deep sense of gratitude to mu supervisors **Dr. Ankit Dubey (Assistant Professor)** and **Dr. Amit Kumar Singh (Assistant Professor)**, **Department of Electrical Engineering, Indian Institute of Technology, Jammu** for providing us the golden opportunity to conduct our Project Work.

I am highly indebted to **DR. Manish Sabraj, Head Of Department, School of Electronics and Communication Engineering, SMVDU** for his assistanceand constant source of encouragement.

I wish to express my profound and deep sense of gratitude to **MR. Ashish Suri (Assistant Professor), School of Electronics and Communication Engineering, SMVDU** for sparing his valuable time and ideas to extend ahelping hand in every step of our project work.

Last but not the least we would like to thank our friends and family for their help in every way for the success of this project.

[illegible]

SURYA PRATAP SINGH

ABSTRACT

Weather Station are built for the purpose of collecting quantitative data about the weather conditions of a place. Monitoring weather Conditions as of our environment today is considered as very important because of the uncertainty in weather every day. This report intends to make a smart weather station which is used to publish ours recorded data on a Digital Dashboard using Network. We have used IOT because IOT provides us the ability to check the various weather changes even being physically absent. The design uses ESP32 as microcontroller to measure weather parameters including Temperature, Humidity, Pressure using BME280, Light using BH1750 Light Sensor, Rain using Rain Sensor. This Report presents a guide on Machine Learning, API and ESP32 interfacing and publishing on a digital dashboard using MQTT. The gathered data is then used in Machine Learning making a prediction model to predict various weather changes. The application of this weather station is that it provides us real time weather of the few meters rather than of the whole city as seen in any weather application as climate changes differs within few meters.

CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF CODE SNIPPETS

LIST OF GRAPHS

LIST OF SYMBOLS AND ABBREVIATIONS

MQTT	Message Queuing Telemetry Transport
API	Application Programming Interface
IOT	Internet of Things
IDE	Integrated Development Environment

1. INTRODUCTION

The Weather Station proposed in this project is an advanced solution for monitoring weather conditions for a particular place and make the information visible anywhere in the world [1]. The advancement over older weather stations is that, older weather stations used to use modules which were connected to balloons and used to gather data from far above the ground so which make this method of weather sensing bulkier and costly with the main flaw of lower accuracy. With the advancement in technology, we are now able to work on smaller sensors which are cost-efficient and with higher accuracy as such made in this project. The technology behind this is Internet of Things which is an advanced and efficient solution for connecting the electronic things to the Internet and to connect the entire world of things in a network, here things can be anything like electronic gadgets, sensors, automotive electronic equipment, etc. This Weather Station works in three phases which are:

1. Real Time Monitoring of the environmental conditions like Temperature, Pressure, Humidity and Rain with Sensors and sending the information to Cloud and then displaying different parameters. For this we use ESP32 Microcontroller to interface our sensors and publish the recorded data to digital dashboard i.e., Adafruit IO.[4]
2. Using gathered data to make a Machine Learning Model for predicting different weather parameters. Python provides us with metapackages like Pandas, numpy, sklearn, matplotlib, tensorflow, requests, etc. which are used in our project to work on Weather Dataset implementing various techniques like data filtering, data splitting, training model used for better predictions with higher accuracy.[2]
3. Lastly learning about API using them to request data and working with different API. Here we request data from an open-source Weather API called Openweathermap to harvest and forecast data. Other alternatives are darksky API, Accuweather API, weatherbit API, Climacell API, etc.

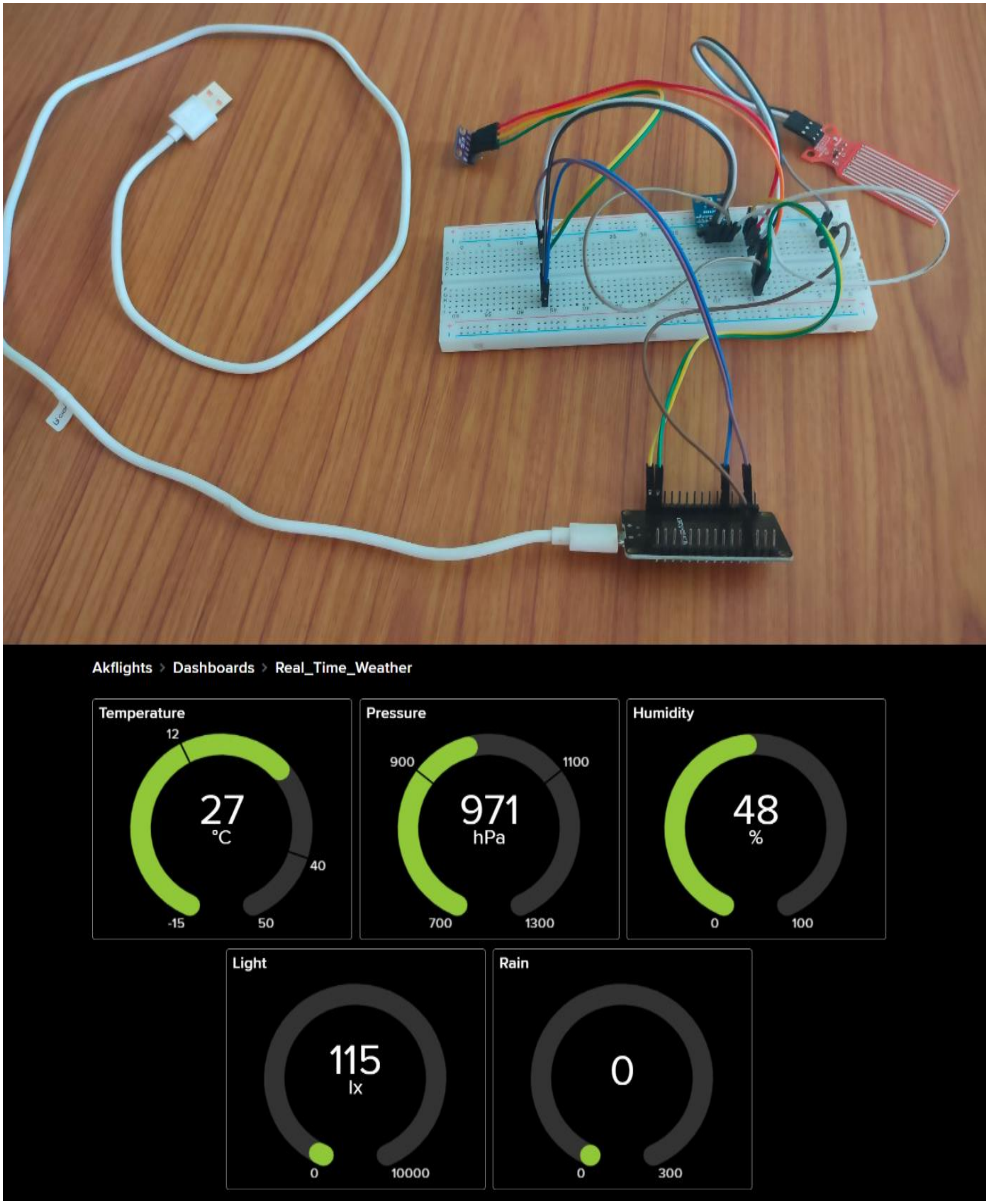


Figure 1: TOP: Sensor and Microcontroller   Bottom: Cloud Output

2. THEORY

The proposed weather station is for monitoring weather parameters which are Temperature, Pressure, Humidity, Rain and displaying data over cloud using MQTT. The MQTT stands for **Message Queuing Telemetry Transport** which is a lightweight, publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP. However, any network protocol that provides ordered, lossless, bi-directional connections can support MQTT. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited.

For data prediction we use Machine Learning. Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. Machine Learning provides user with different training models like Linear Regression, Polynomial Regression, Decision Tree Regressor, Random Decision Forest, etc. In our weather station we will be using Linear Regression and Decision Tree Regressor Model.

To forecast weather, we use Weather API. API stands for Application Programming Interface which is an interface that defines interactions between multiple software applications or mixed hardware-software intermediaries. It defines the kinds of calls or requests that can be made, how to make them, the data format that should be used, the conventions to follow, etc. It can also provide extension mechanisms so that users can extend existing functionality in various ways and to varying degrees. An API can be entirely custom, specific to a component, or designed based on an industry-standard to ensure interoperability.

3. HARDWARE USED

3.1 ESP 32 WIFI and Bluetooth Microcontroller:

ESP32 is a series of low-cost, low-power system on a chip microcontroller with integrated WIFI and dual-mode Bluetooth. The ESP32 series employs a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules.

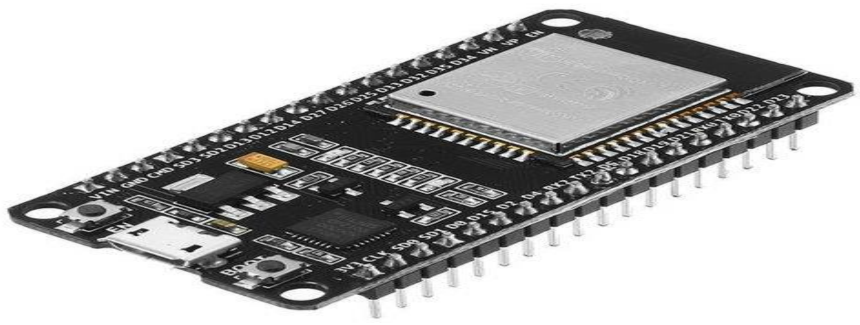


Figure 2: ESP 32 MCU

3.2 Rain Sensor:

A rain sensor is one kind of switching device which is used to detect the rainfall. It works like a switch and the working principle of this sensor is, whenever there is rain, the switch will be normally closed. This sensor is a resistive dipole, and based on the moisture only it shows the resistance. For example, it shows more resistance when it is dry and shows less resistance when it is wet.

Figure 3: Rain Sensor

3.3 BME 280:

BME 280 is a next-generation digital temperature, humidity, pressure sensor manufactured by Bosch. It is a small sensor which makes it highly preferred by mobile manufacturers, etc. It is successor to sensors like BMP180, BMP183, BMP0185.

Table 1: BME 280 Specification

Operating Range	Temperature: -40°C - 85°C (accuracy ±1.0°C)
	Pressure: 300hPa - 1100hPa (accuracy ±1.0hPa)
	Humidity: 0% - 100% (accuracy ±3.0°%)
Power Requirement	1.71 – 3.6 V
Interface	I^2C and SPI

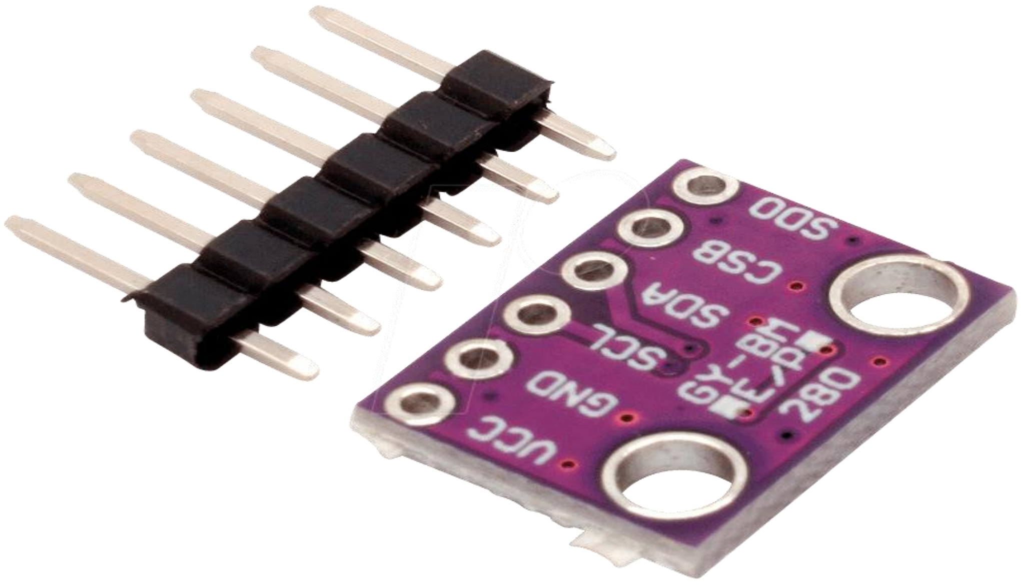


Figure 4: BME 280

3.4 BH1750 Light Sensor:

BH1750 is a digital ambient light sensor that is used commonly used in mobile phones to manipulate the screen brightness based on the environment lighting. This sensor can accurately measure the LUX value of light up to 65535lx. It has built in A/D converter for converting analog illuminance in the digital data. It has very small effect of IR Radiation and highly responsive near to human eye.

Table 2: BH1750 Specification

Operating Range	1 – 65535 lx
Voltage	2.4 – 3.6 V
Interface	I^2C

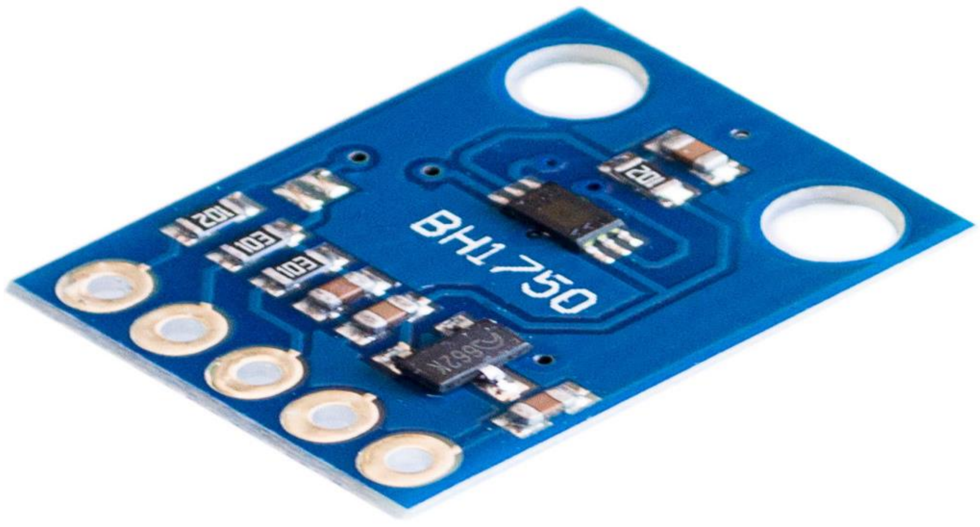


Figure 5: BH1750 Light Sensor

4. BLOCK DIAGRAM

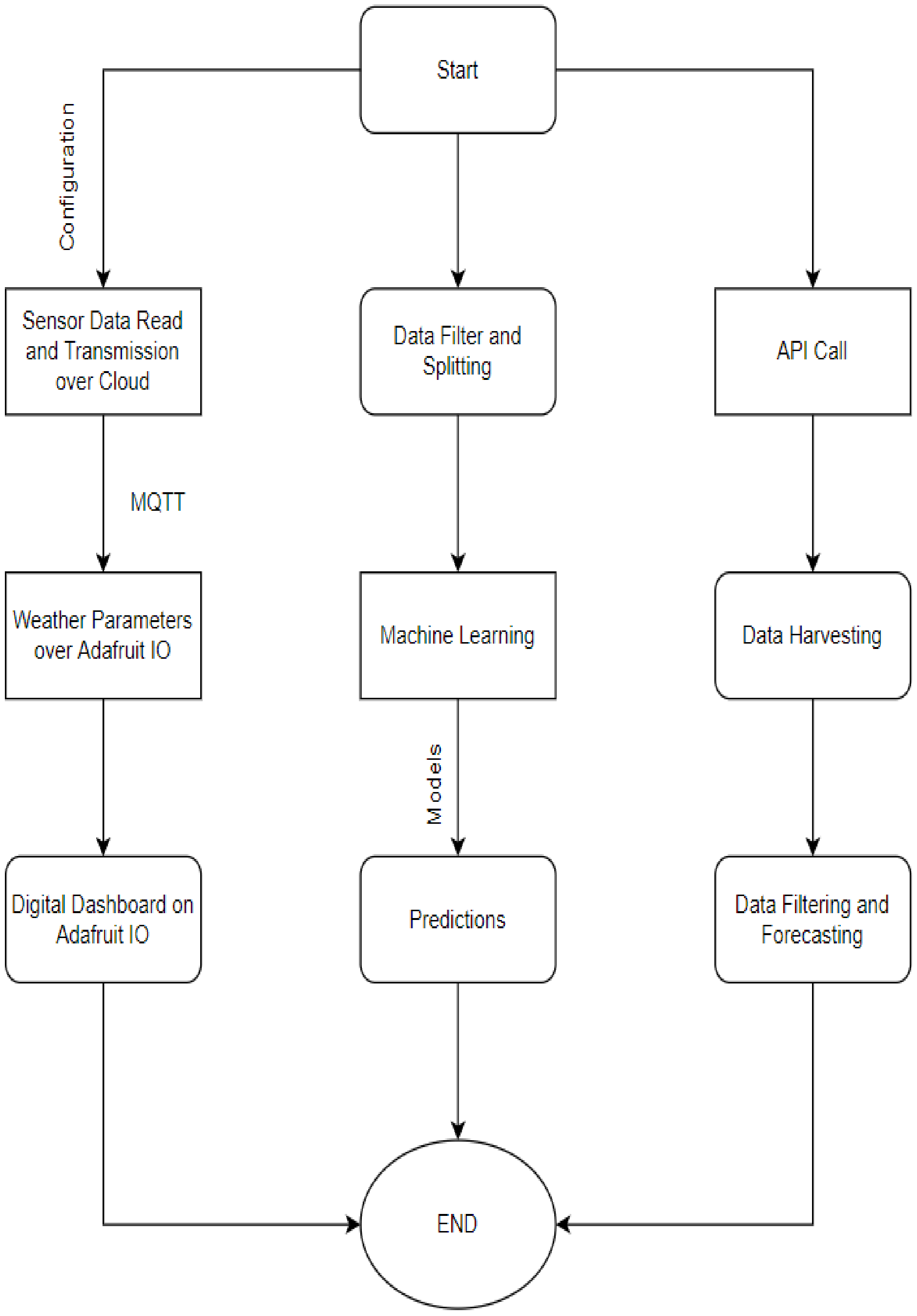

Start
Configuration
Sensor Data Read and Transmission over Cloud
Data Filter and Splitting
API Call
MQTT
Weather Parameters over Adafruit IO
Machine Learning
Data Harvesting
Models
Digital Dashboard on Adafruit IO
Predictions
Data Filtering and Forecasting
END

Figure 6: Block Diagram

5. CIRCUIT DIAGRAM

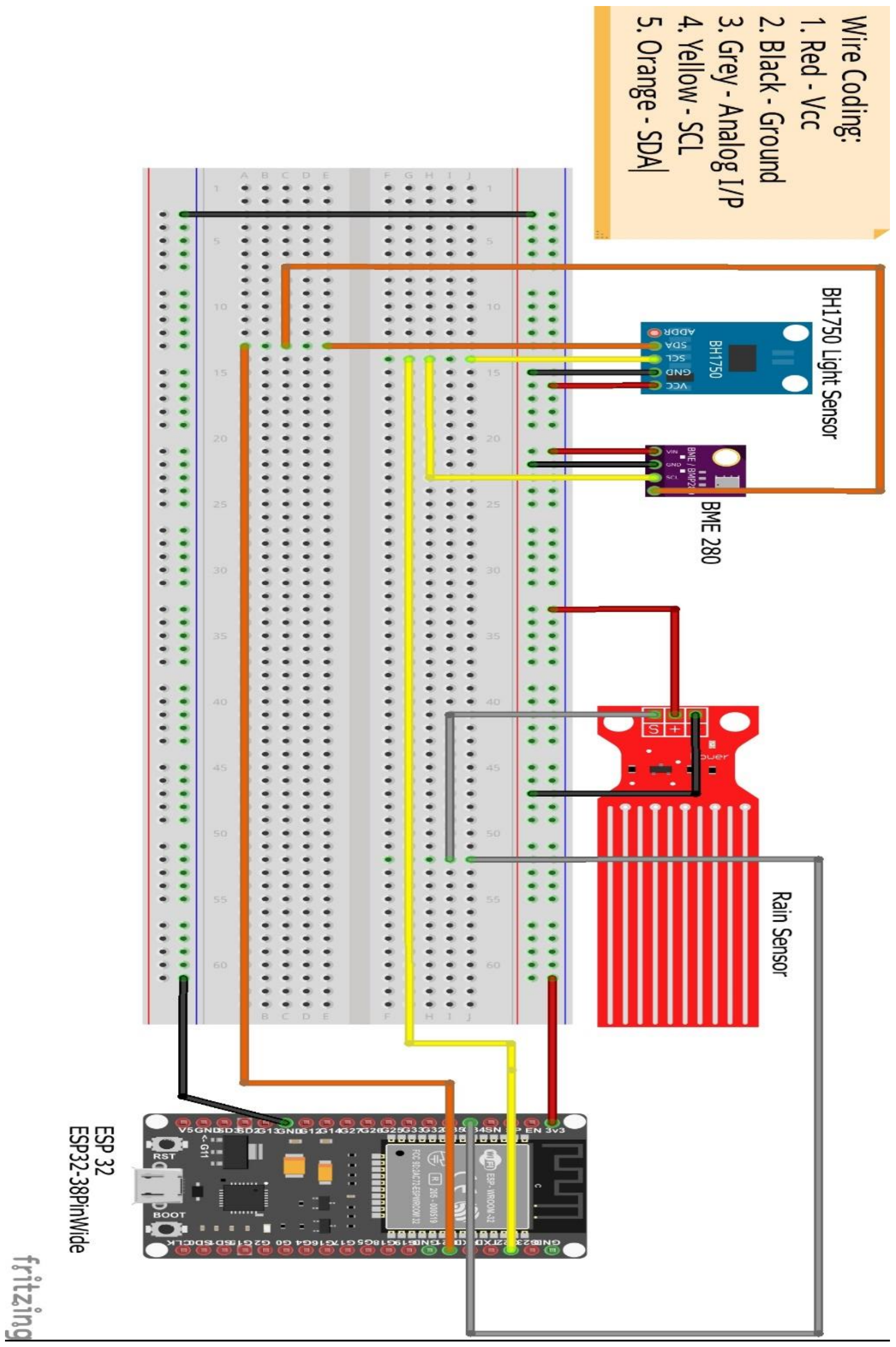


Figure 7: Circuit Diagram

6. BUILDING WEATHER STATION

1. Make the circuit as shown in figure and prepare a printed circuit board as shown.
2. Install Libraries used like Adafruit IO, Adafruit Sensor, BH1750, etc. either from zip or library manager.
3. Adafruit IO comes with inbuilt libraries for MQTT. Sign up on Adafruit.io and create feeds for our weather parameters. In Arduino code configure WIFI, AIO Username and Key provided by Adafruit.io
4. Create the required dashboard using created feeds on Adafruit.IO.
5. Write program for collecting data from sensors and sending to Adafruit.io and displaying on dashboard.

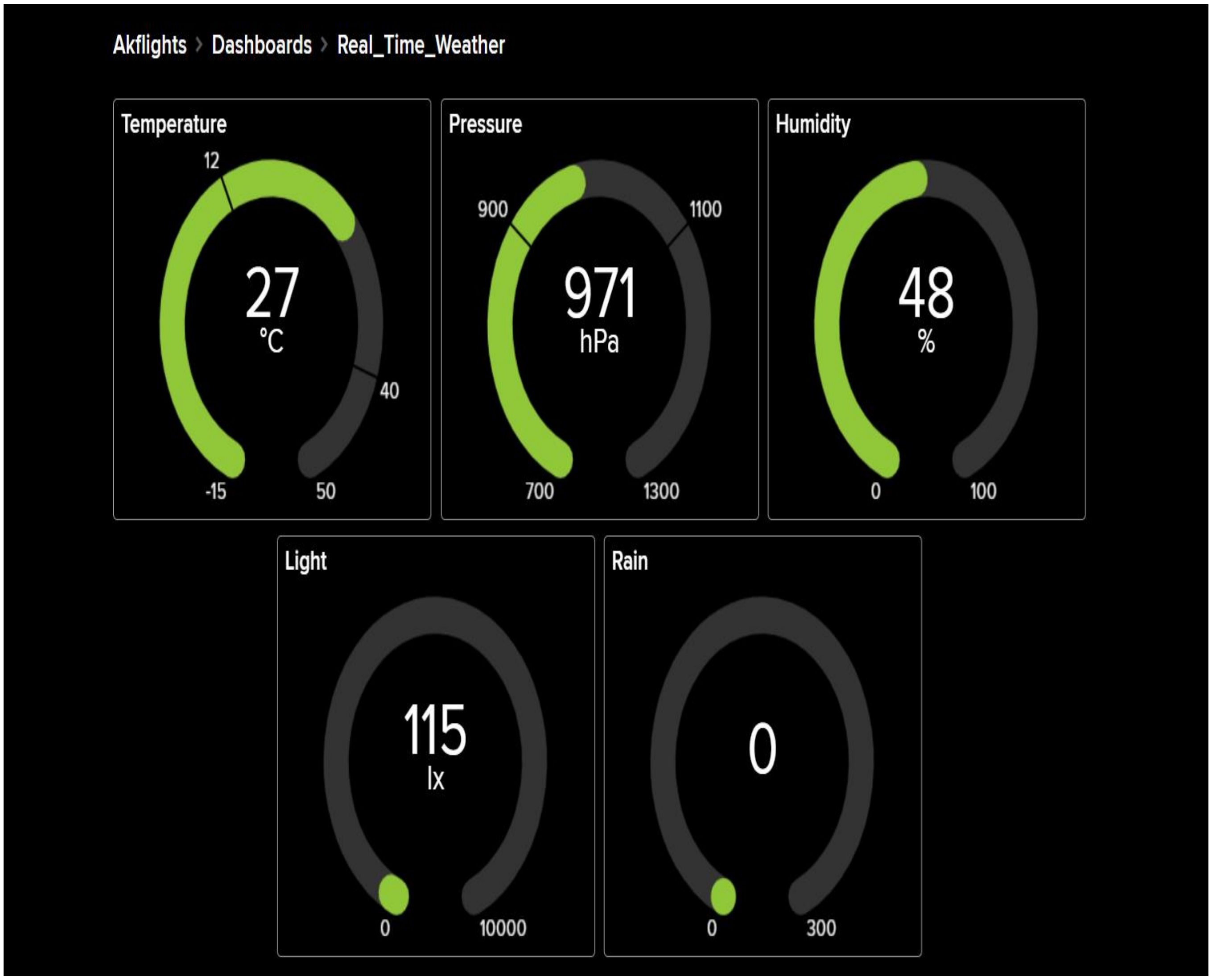
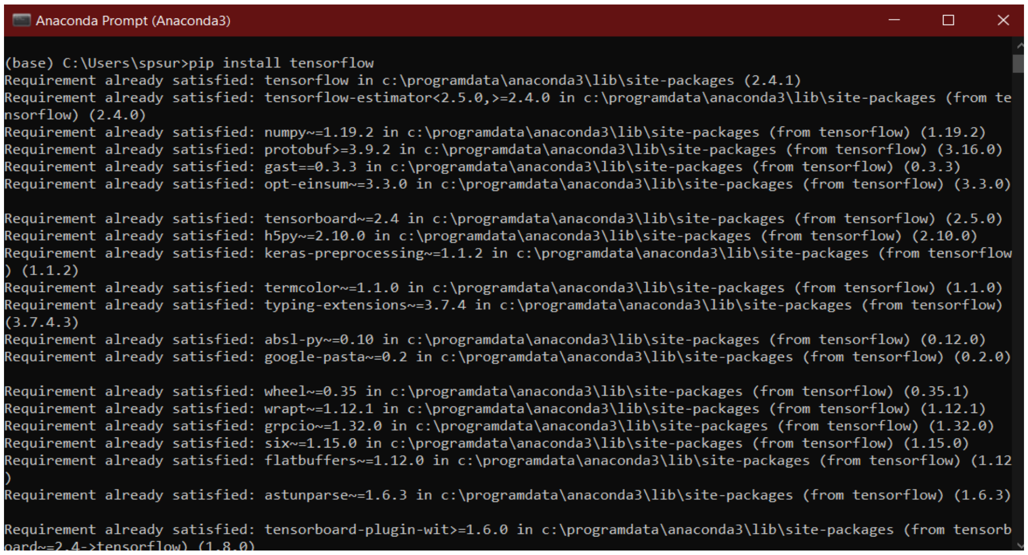


Figure 8:Digital Dashboard

7. COLLECTING DATA

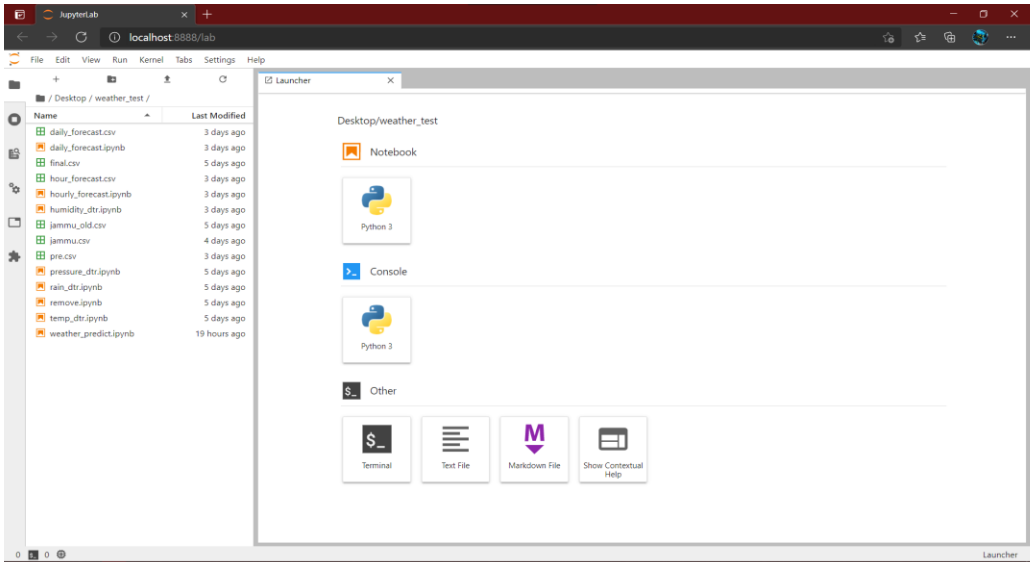
7.1 Installing Dependencies:

For our Weather Model we use Anaconda Individual Edition which is a Python Distribution Platform which comes with pre-installed IDE, Command Prompt, Jupyter Lab, etc. For our Base Station we use Jupyter Lab for implementing weather model and prediction. Most of the metapackage and dependencies are already installed but if you want a specific package, you can install using the command "pip install <package name>".



```
(base) C:\Users\spsur>pip install tensorflow
Requirement already satisfied: tensorflow in c:\programdata\anaconda3\lib\site-packages (2.4.1)
Requirement already satisfied: tensorflow-estimator<2.5.0,>=2.4.0 in c:\programdata\anaconda3\lib\site-packages (from te
nsorflow) (2.4.0)
Requirement already satisfied: numpy~=1.19.2 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (1.19.2)
Requirement already satisfied: protobuf>=3.9.2 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (3.16.0)
Requirement already satisfied: gast==0.3.3 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (0.3.3)
Requirement already satisfied: opt-einsum~=3.3.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (3.3.0)

Requirement already satisfied: tensorboard~=2.4 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (2.5.0)
Requirement already satisfied: h5py~=2.10.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (2.10.0)
Requirement already satisfied: keras-preprocessing~=1.1.2 in c:\programdata\anaconda3\lib\site-packages (from tensorflow
) (1.1.2)
Requirement already satisfied: termcolor~=1.1.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: typing-extensions~=3.7.4 in c:\programdata\anaconda3\lib\site-packages (from tensorflow)
(3.7.4.3)
Requirement already satisfied: absl-py~=0.10 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (0.12.0)
Requirement already satisfied: google-pasta~=0.2 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (0.2.0)

Requirement already satisfied: wheel~=0.35 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (0.35.1)
Requirement already satisfied: wrapt~=1.12.1 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: grpcio~=1.32.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (1.32.0)
Requirement already satisfied: six~=1.15.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (1.15.0)
Requirement already satisfied: flatbuffers~=1.12.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (1.12
)
Requirement already satisfied: astunparse~=1.6.3 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (1.6.3)

Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\programdata\anaconda3\lib\site-packages (from tensorb
oard~=2.4->tensorflow) (1.8.0)
```

Figure 9: Anaconda Prompt

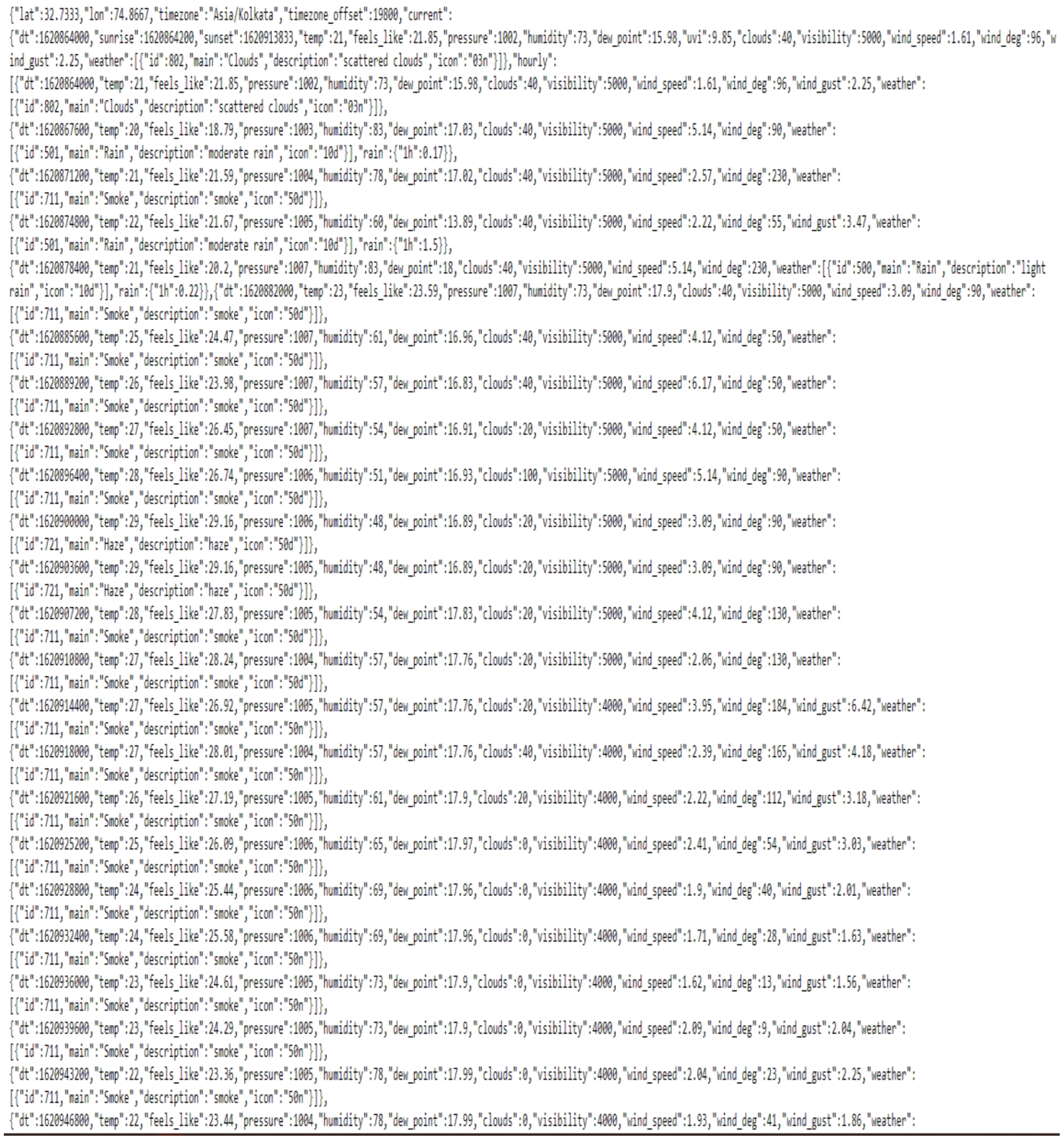


```
/ Desktop / weather_test /
Name                      Last Modified
daily_forecast.csv        3 days ago
daily_forecast.ipynb      3 days ago
final.csv                 5 days ago
hour_forecast.csv         3 days ago
hourly_forecast.ipynb     3 days ago
humidity_dtr.ipynb        3 days ago
jammu_old.csv             5 days ago
jammu.csv                 4 days ago
pre.csv                   3 days ago
pressure_dtr.ipynb        5 days ago
rain_dtr.ipynb            5 days ago
remove.ipynb              5 days ago
temp_dtr.ipynb            5 days ago
weather_predict.ipynb     19 hours ago
```

Figure 10: Jupyter Lab

7.2 Data Collection:

To implement Machine Learning the first Task is to collect data. We can either use data collected from our weather station or use official datasheet of any open weather platform here we use openweathermap.org for historical data of current location. The data is in the . json format with data type as dictionary as shown:

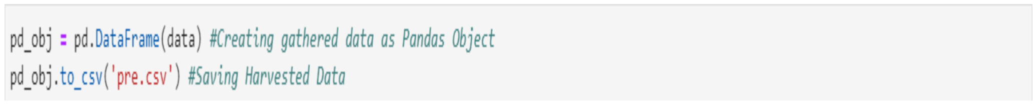


```
{"lat":32.7333,"lon":74.8667,"timezone":"Asia/Kolkata","timezone_offset":19800,"current":
{"dt":1620864000,"sunrise":1620864200,"sunset":1620913833,"temp":21,"feels_like":21.85,"pressure":1002,"humidity":73,"dew_point":15.98,"uvi":9.85,"clouds":40,"visibility":5000,"wind_speed":1.61,"wind_deg":96,"w
ind_gust":2.25,"weather":[{"id":802,"main":"Clouds","description":"scattered clouds","icon":"03n"}]},"hourly":
[{"dt":1620864000,"temp":21,"feels_like":21.85,"pressure":1002,"humidity":73,"dew_point":15.98,"clouds":40,"visibility":5000,"wind_speed":1.61,"wind_deg":96,"wind_gust":2.25,"weather":
[{"id":802,"main":"Clouds","description":"scattered clouds","icon":"03n"}]},
{"dt":1620867600,"temp":20,"feels_like":18.79,"pressure":1003,"humidity":83,"dew_point":17.03,"clouds":40,"visibility":5000,"wind_speed":5.14,"wind_deg":90,"weather":
[{"id":501,"main":"Rain","description":"moderate rain","icon":"10d"}],"rain":{"1h":0.17}},
{"dt":1620871200,"temp":21,"feels_like":21.59,"pressure":1004,"humidity":78,"dew_point":17.02,"clouds":40,"visibility":5000,"wind_speed":2.57,"wind_deg":230,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50d"}]},
{"dt":1620874800,"temp":22,"feels_like":21.67,"pressure":1005,"humidity":60,"dew_point":13.89,"clouds":40,"visibility":5000,"wind_speed":2.22,"wind_deg":55,"wind_gust":3.47,"weather":
[{"id":501,"main":"Rain","description":"moderate rain","icon":"10d"}],"rain":{"1h":1.5}},
{"dt":1620878400,"temp":21,"feels_like":20.2,"pressure":1007,"humidity":83,"dew_point":18,"clouds":40,"visibility":5000,"wind_speed":5.14,"wind_deg":230,"weather":[{"id":500,"main":"Rain","description":"light
rain","icon":"10d"}],"rain":{"1h":0.22}},{"dt":1620882000,"temp":23,"feels_like":23.59,"pressure":1007,"humidity":73,"dew_point":17.9,"clouds":40,"visibility":5000,"wind_speed":3.09,"wind_deg":90,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50d"}]},
{"dt":1620885600,"temp":25,"feels_like":24.47,"pressure":1007,"humidity":61,"dew_point":16.96,"clouds":40,"visibility":5000,"wind_speed":4.12,"wind_deg":50,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50d"}]},
{"dt":1620889200,"temp":26,"feels_like":23.98,"pressure":1007,"humidity":57,"dew_point":16.83,"clouds":40,"visibility":5000,"wind_speed":6.17,"wind_deg":50,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50d"}]},
{"dt":1620892800,"temp":27,"feels_like":26.45,"pressure":1007,"humidity":54,"dew_point":16.91,"clouds":20,"visibility":5000,"wind_speed":4.12,"wind_deg":50,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50d"}]},
{"dt":1620896400,"temp":28,"feels_like":26.74,"pressure":1006,"humidity":51,"dew_point":16.93,"clouds":100,"visibility":5000,"wind_speed":5.14,"wind_deg":90,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50d"}]},
{"dt":1620900000,"temp":29,"feels_like":29.16,"pressure":1006,"humidity":48,"dew_point":16.89,"clouds":20,"visibility":5000,"wind_speed":3.09,"wind_deg":90,"weather":
[{"id":721,"main":"Haze","description":"haze","icon":"50d"}]},
{"dt":1620903600,"temp":29,"feels_like":29.16,"pressure":1005,"humidity":48,"dew_point":16.89,"clouds":20,"visibility":5000,"wind_speed":3.09,"wind_deg":90,"weather":
[{"id":721,"main":"Haze","description":"haze","icon":"50d"}]},
{"dt":1620907200,"temp":28,"feels_like":27.83,"pressure":1005,"humidity":54,"dew_point":17.83,"clouds":20,"visibility":5000,"wind_speed":4.12,"wind_deg":130,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50d"}]},
{"dt":1620910800,"temp":27,"feels_like":28.24,"pressure":1004,"humidity":57,"dew_point":17.76,"clouds":20,"visibility":5000,"wind_speed":2.06,"wind_deg":130,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50d"}]},
{"dt":1620914400,"temp":27,"feels_like":26.92,"pressure":1005,"humidity":57,"dew_point":17.76,"clouds":20,"visibility":4000,"wind_speed":3.95,"wind_deg":184,"wind_gust":6.42,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50n"}]},
{"dt":1620918000,"temp":27,"feels_like":28.01,"pressure":1004,"humidity":57,"dew_point":17.76,"clouds":40,"visibility":4000,"wind_speed":2.39,"wind_deg":165,"wind_gust":4.18,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50n"}]},
{"dt":1620921600,"temp":26,"feels_like":27.19,"pressure":1005,"humidity":61,"dew_point":17.9,"clouds":20,"visibility":4000,"wind_speed":2.22,"wind_deg":112,"wind_gust":3.18,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50n"}]},
{"dt":1620925200,"temp":25,"feels_like":26.09,"pressure":1006,"humidity":65,"dew_point":17.97,"clouds":0,"visibility":4000,"wind_speed":2.41,"wind_deg":54,"wind_gust":3.03,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50n"}]},
{"dt":1620928800,"temp":24,"feels_like":25.44,"pressure":1006,"humidity":69,"dew_point":17.96,"clouds":0,"visibility":4000,"wind_speed":1.9,"wind_deg":40,"wind_gust":2.01,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50n"}]},
{"dt":1620932400,"temp":24,"feels_like":25.58,"pressure":1006,"humidity":69,"dew_point":17.96,"clouds":0,"visibility":4000,"wind_speed":1.71,"wind_deg":28,"wind_gust":1.63,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50n"}]},
{"dt":1620936000,"temp":23,"feels_like":24.61,"pressure":1005,"humidity":73,"dew_point":17.9,"clouds":0,"visibility":4000,"wind_speed":1.62,"wind_deg":13,"wind_gust":1.56,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50n"}]},
{"dt":1620939600,"temp":23,"feels_like":24.29,"pressure":1005,"humidity":73,"dew_point":17.9,"clouds":0,"visibility":4000,"wind_speed":2.09,"wind_deg":9,"wind_gust":2.04,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50n"}]},
{"dt":1620943200,"temp":22,"feels_like":23.36,"pressure":1005,"humidity":78,"dew_point":17.99,"clouds":0,"visibility":4000,"wind_speed":2.04,"wind_deg":23,"wind_gust":2.25,"weather":
[{"id":711,"main":"Smoke","description":"smoke","icon":"50n"}]},
{"dt":1620946800,"temp":22,"feels_like":23.44,"pressure":1004,"humidity":78,"dew_point":17.99,"clouds":0,"visibility":4000,"wind_speed":1.93,"wind_deg":41,"wind_gust":1.86,"weather":
```

Figure 11: Collected data

7.3 Data Filtering:

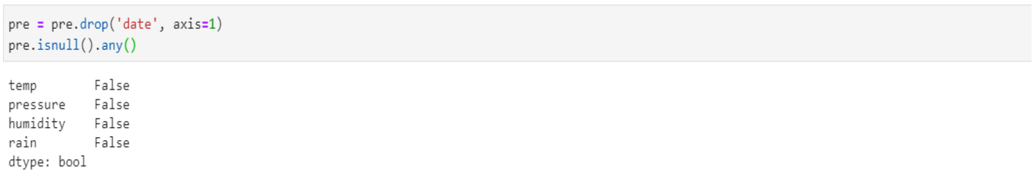
Next step is to collect data from the Json and convert to csv for further usage. For this we use pandas. Pandas is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language. The above json is then converted to pandas data frame as an object. Then this pandas object is saved to csv using "<object name>.to_csv('<filename>.csv')".



```
pd_obj = pd.DataFrame(data) #Creating gathered data as Pandas Object
pd_obj.to_csv('pre.csv') #Saving Harvested Data
```

Code Snippet 1: Gathered Data to csv

Then we need to filter the gathered data by removing unnecessary columns using "pd.drop('column name', axis=1)" and checking for null values which decreases the accuracy of our model using "isnull()" command.

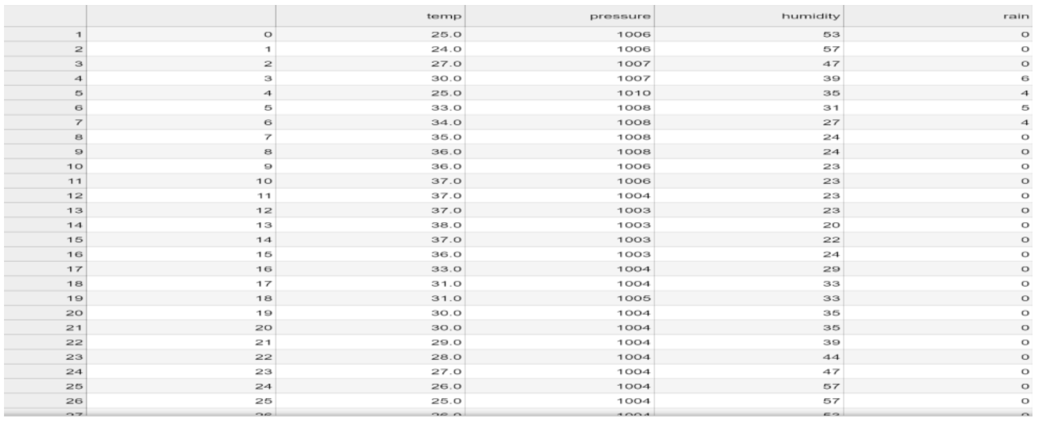


```
pre = pre.drop('date', axis=1)
pre.isnull().any()
```

```
temp        False
pressure    False
humidity    False
rain        False
dtype: bool
```

Code Snippet 2: Filtering Data

If there are null values replace them with values and then again save this csv which is the final data set on which our model predicts. Our csv looks as follow:



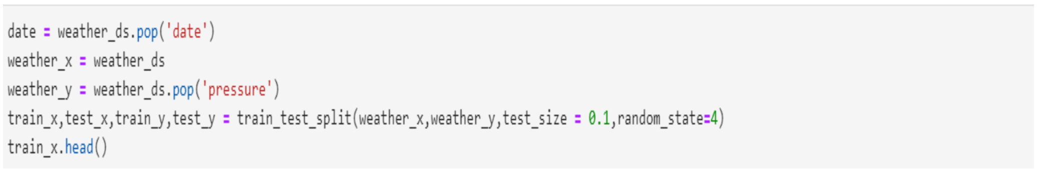
		temp	pressure	humidity	rain
1	0	25.0	1006	53	0
2	1	24.0	1006	57	0
3	2	27.0	1007	47	0
4	3	30.0	1007	39	6
5	4	25.0	1010	35	4
6	5	33.0	1008	31	5
7	6	34.0	1008	27	4
8	7	35.0	1008	24	0
9	8	36.0	1008	24	0
10	9	36.0	1006	23	0
11	10	37.0	1006	23	0
12	11	37.0	1004	23	0
13	12	37.0	1003	23	0
14	13	38.0	1003	20	0
15	14	37.0	1003	22	0
16	15	36.0	1003	24	0
17	16	33.0	1004	29	0
18	17	31.0	1004	33	0
19	18	31.0	1005	33	0
20	19	30.0	1004	35	0
21	20	30.0	1004	35	0
22	21	29.0	1004	39	0
23	22	28.0	1004	44	0
24	23	27.0	1004	47	0
25	24	26.0	1004	57	0
26	25	25.0	1004	57	0

Figure 12: Filtered CSV

8. BUILDING PREDICTION MODEL

8.1 Data Selection and Splitting:

Before building machine learning model we need to select the parameter we want to predict using our prediction model and for that we select the parameter, remove from our dataset and assign it to a variable (y used) and then assigning x as rest of data set. Now sklearn is a package of python which gives us the flexibility of splitting data into our train and test with desired split ratio.



```
date = weather_ds.pop('date')
weather_x = weather_ds
weather_y = weather_ds.pop('pressure')
train_x,test_x,train_y,test_y = train_test_split(weather_x,weather_y,test_size = 0.1,random_state=4)
train_x.head()
```

Code Snippet 3: Data Selection and Splitting

8.2 Training Model:

Once above steps are complete now, we need to train our model using different models like Linear Regression, Polynomial Regression, Decision Tree Regression, etc. here we use Linear Regression and Decision Tree Regressor.

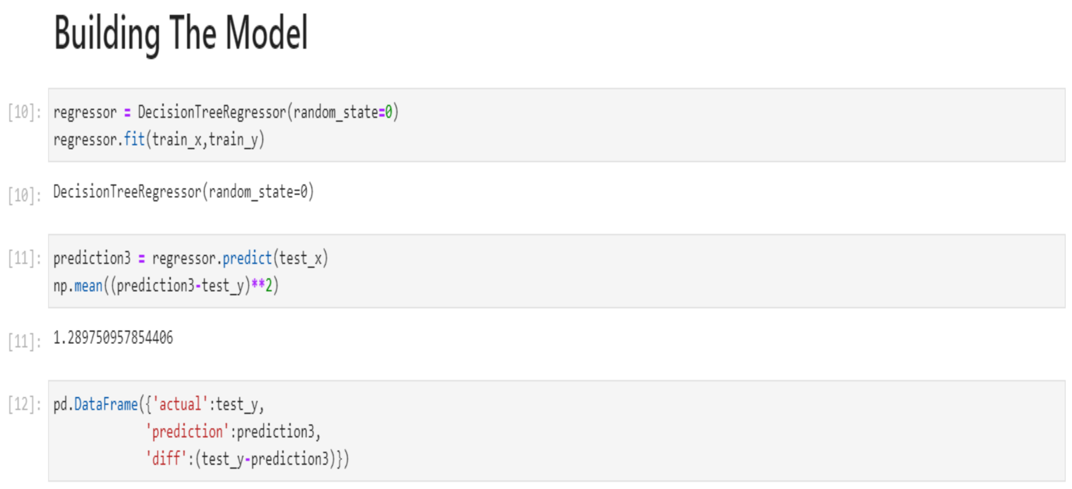
Linear Regression Model:

Linear regression is a linear approach to modelling the relationship between a dependent variable and one or more independent variables.

Decision Tree Regressor:

For each attribute in the dataset, the decision tree algorithm forms a node, where the most important attribute is placed at the root node. For evaluation we start at the root node and work our way down the tree by following the corresponding node that meets our condition or "decision". This process continues until a leaf node is reached, which contains the prediction or the outcome of the decision tree.

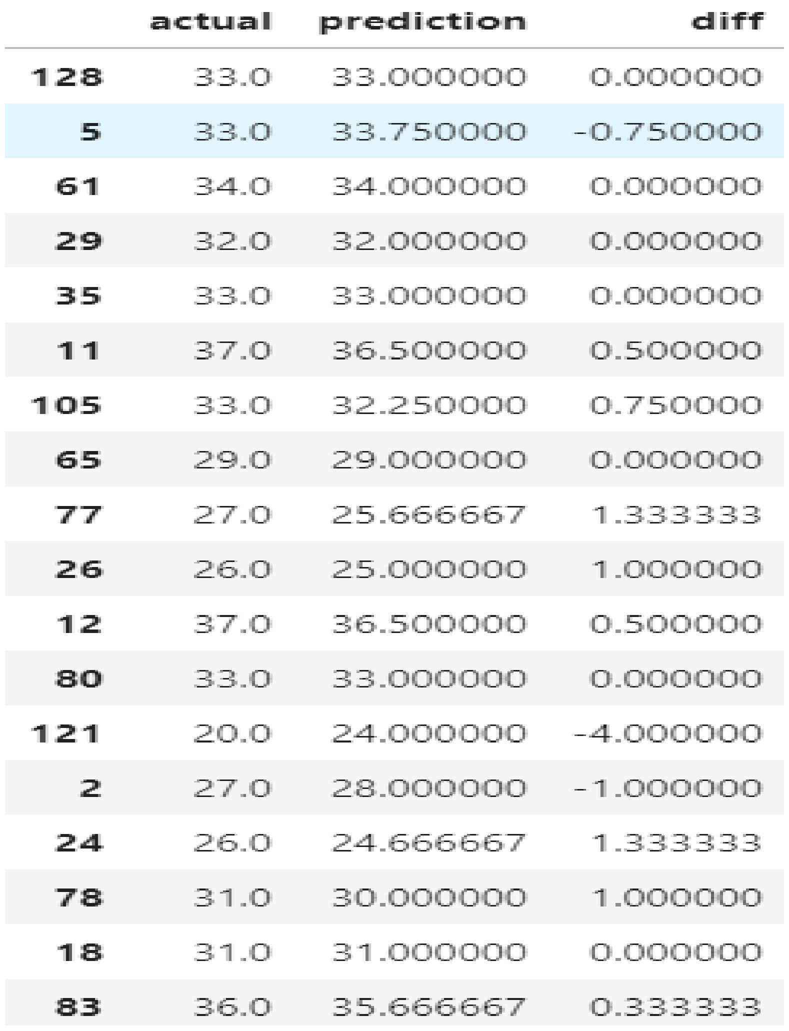
Firstly, we import the model we want to use from sklearn and then fit into our model the train split data set and then finally predict the weather for the test split.



Building The Model

```
[10]: regressor = DecisionTreeRegressor(random_state=0)
      regressor.fit(train_x,train_y)

[10]: DecisionTreeRegressor(random_state=0)

[11]: prediction3 = regressor.predict(test_x)
      np.mean((prediction3-test_y)**2)

[11]: 1.289750957854406

[12]: pd.DataFrame({'actual':test_y,
                    'prediction':prediction3,
                    'diff':(test_y-prediction3)})
```

Code Snippet 4: Building the Model

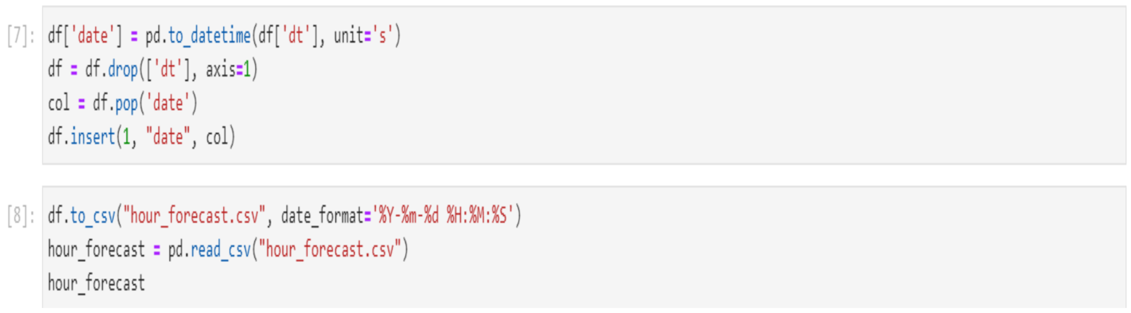


	actual	prediction	diff
128	33.0	33.000000	0.000000
5	33.0	33.750000	-0.750000
61	34.0	34.000000	0.000000
29	32.0	32.000000	0.000000
35	33.0	33.000000	0.000000
11	37.0	36.500000	0.500000
105	33.0	32.250000	0.750000
65	29.0	29.000000	0.000000
77	27.0	25.666667	1.333333
26	26.0	25.000000	1.000000
12	37.0	36.500000	0.500000
80	33.0	33.000000	0.000000
121	20.0	24.000000	-4.000000
2	27.0	28.000000	-1.000000
24	26.0	24.666667	1.333333
78	31.0	30.000000	1.000000
18	31.0	31.000000	0.000000
83	36.0	35.666667	0.333333

Figure 13: Result

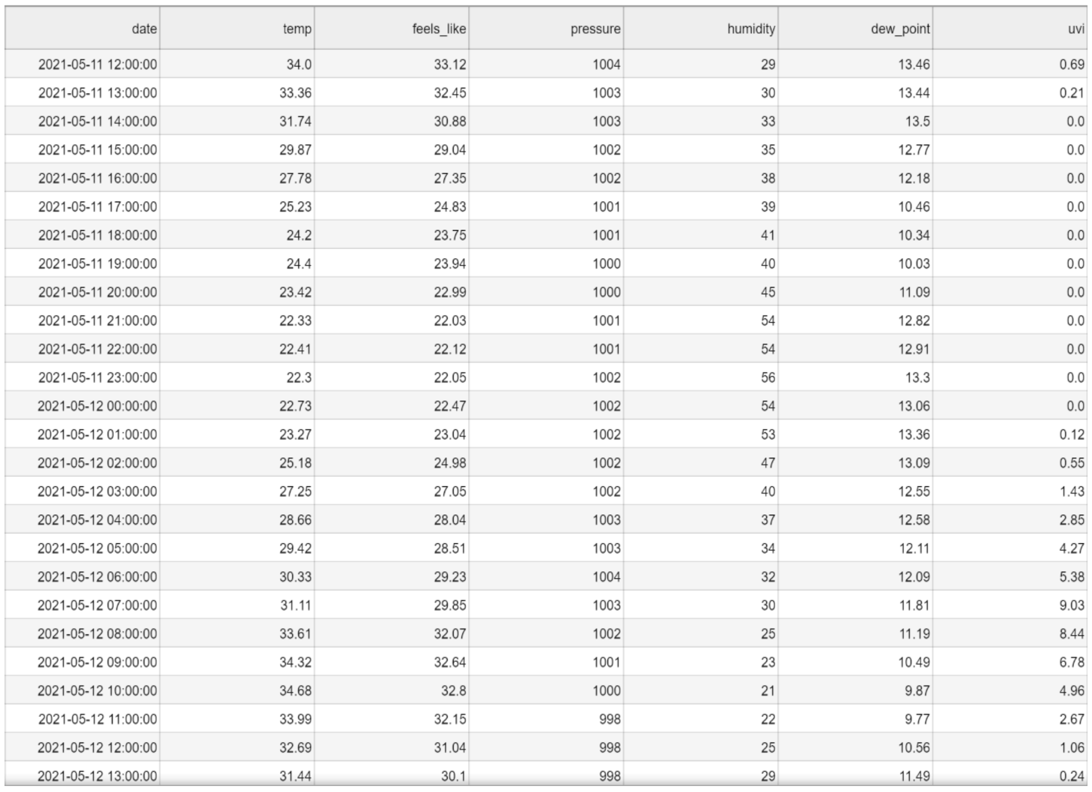
9. API WEATHER FORECAST

For getting a API key we sign up on weather API website like darksky, openweathermap.org, etc. we used openweathermap and then we request the data using API key once gathered we harvest the necessary data and drop the unrequired parameters. Then we refine our result by converting date in Epochs to Standard time and finally save the data to csv file.



```
[7]: df['date'] = pd.to_datetime(df['dt'], unit='s')
     df = df.drop(['dt'], axis=1)
     col = df.pop('date')
     df.insert(1, "date", col)

[8]: df.to_csv("hour_forecast.csv", date_format='%Y-%m-%d %H:%M:%S')
     hour_forecast = pd.read_csv("hour_forecast.csv")
     hour_forecast
```

Code Snippet 5: Refining Data

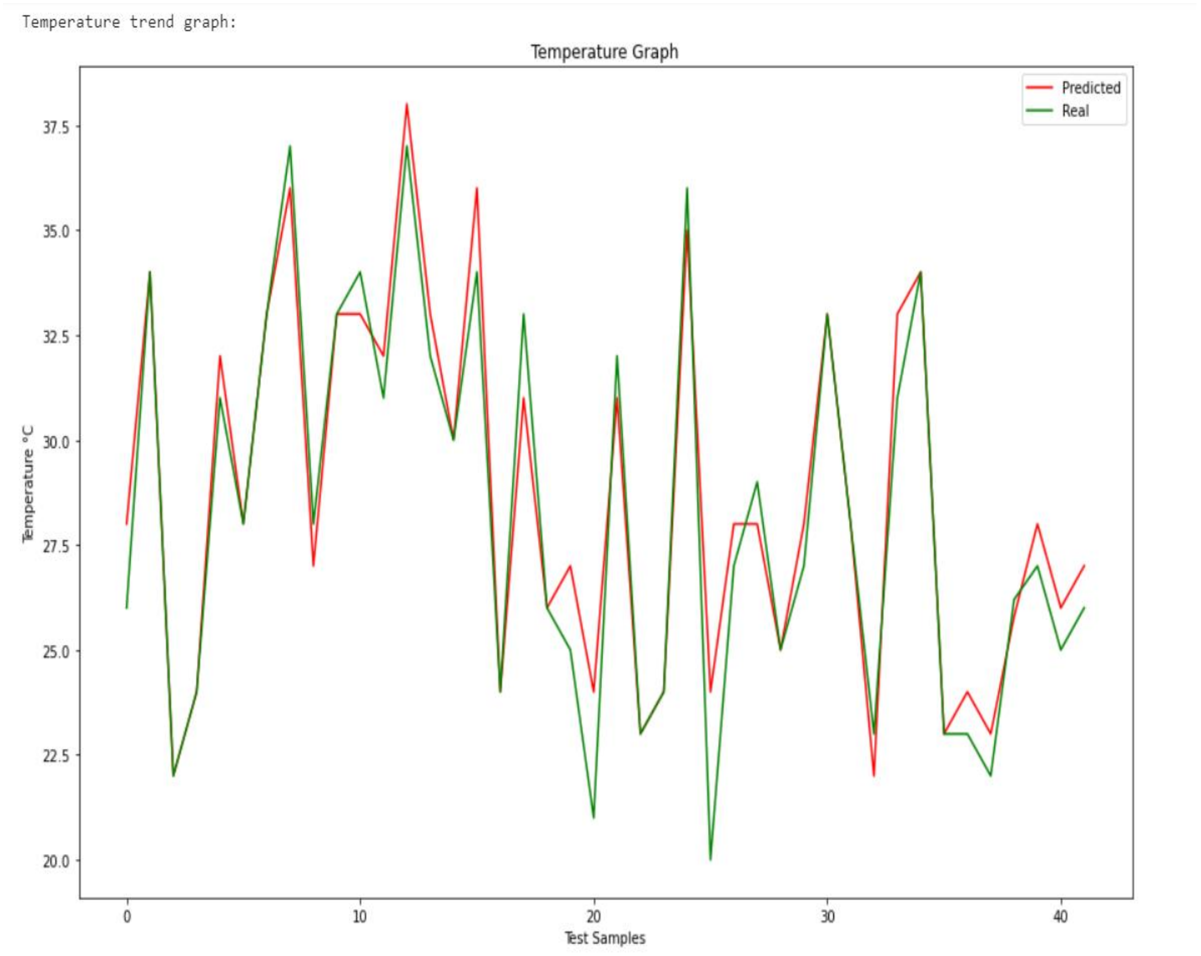


date	temp	feels_like	pressure	humidity	dew_point	uvi
2021-05-11 12:00:00	34.0	33.12	1004	29	13.46	0.69
2021-05-11 13:00:00	33.36	32.45	1003	30	13.44	0.21
2021-05-11 14:00:00	31.74	30.88	1003	33	13.5	0.0
2021-05-11 15:00:00	29.87	29.04	1002	35	12.77	0.0
2021-05-11 16:00:00	27.78	27.35	1002	38	12.18	0.0
2021-05-11 17:00:00	25.23	24.83	1001	39	10.46	0.0
2021-05-11 18:00:00	24.2	23.75	1001	41	10.34	0.0
2021-05-11 19:00:00	24.4	23.94	1000	40	10.03	0.0
2021-05-11 20:00:00	23.42	22.99	1000	45	11.09	0.0
2021-05-11 21:00:00	22.33	22.03	1001	54	12.82	0.0
2021-05-11 22:00:00	22.41	22.12	1001	54	12.91	0.0
2021-05-11 23:00:00	22.3	22.05	1002	56	13.3	0.0
2021-05-12 00:00:00	22.73	22.47	1002	54	13.06	0.0
2021-05-12 01:00:00	23.27	23.04	1002	53	13.36	0.12
2021-05-12 02:00:00	25.18	24.98	1002	47	13.09	0.55
2021-05-12 03:00:00	27.25	27.05	1002	40	12.55	1.43
2021-05-12 04:00:00	28.66	28.04	1003	37	12.58	2.85
2021-05-12 05:00:00	29.42	28.51	1003	34	12.11	4.27
2021-05-12 06:00:00	30.33	29.23	1004	32	12.09	5.38
2021-05-12 07:00:00	31.11	29.85	1003	30	11.81	9.03
2021-05-12 08:00:00	33.61	32.07	1002	25	11.19	8.44
2021-05-12 09:00:00	34.32	32.64	1001	23	10.49	6.78
2021-05-12 10:00:00	34.68	32.8	1000	21	9.87	4.96
2021-05-12 11:00:00	33.99	32.15	998	22	9.77	2.67
2021-05-12 12:00:00	32.69	31.04	998	25	10.56	1.06
2021-05-12 13:00:00	31.44	30.1	998	29	11.49	0.24

Figure 14: Forecast Data

10.GRAPHS

10.1 Temperature vs Test Samples:

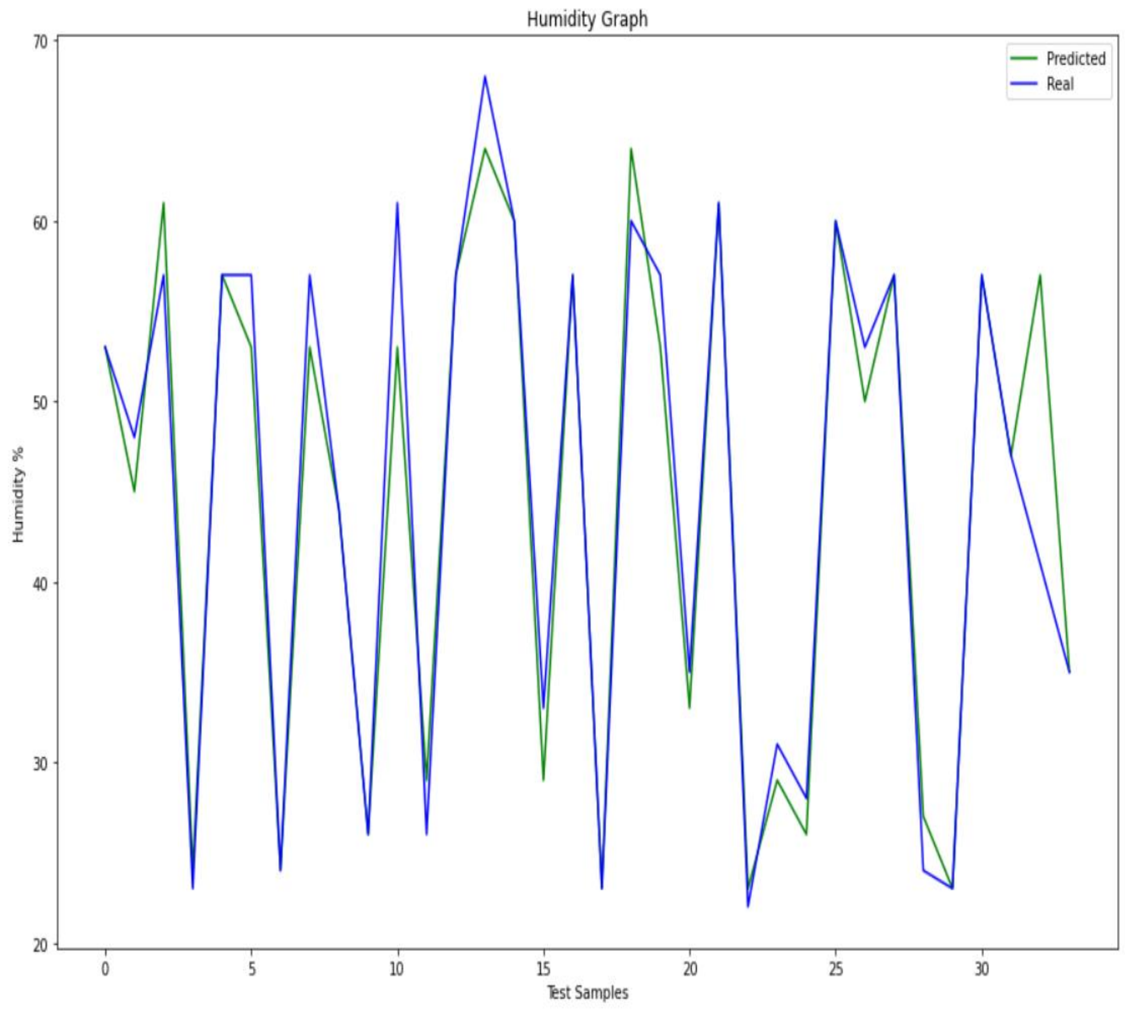

```
Temperature trend graph:
```

Graph 1: Temperature vs Test Samples

The above graph shows the Temperature vs test samples result where x-axis shows Test Samples and y-axis shows Temperature in Degree Celsius. The Green line shows the real temperature values for the Random Test Samples and Red line shows the predicted temperature by our model. As the most of the graph is either overlapping or minimal difference between the real and predicted values, we can say that our model has high accuracy.

10.2 Humidity vs Test Samples:

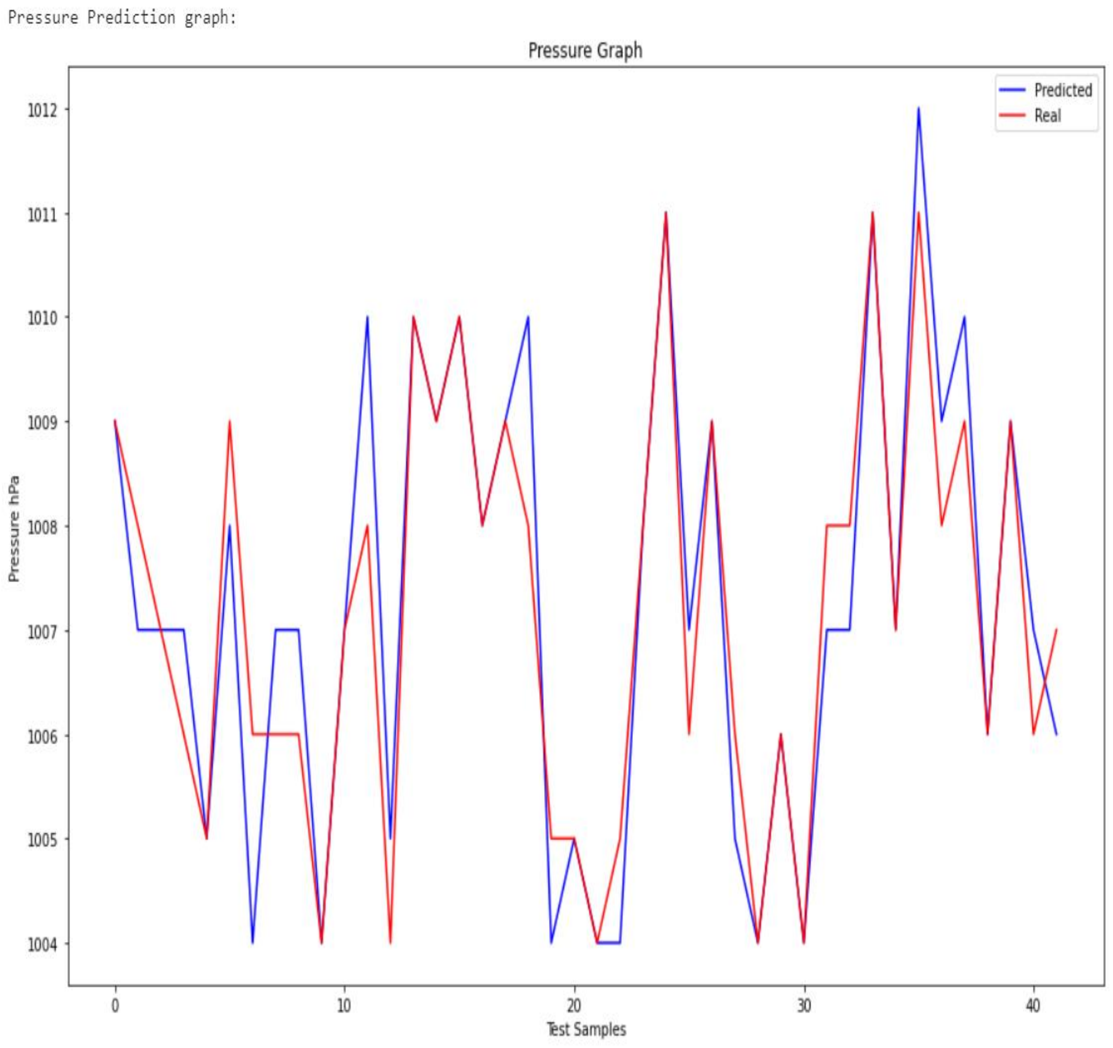
Humidity Prediction graph:



Graph 2: Humidity vs Test Samples

The above graph shows the Humidity vs test samples result where x-axis shows Test Samples and y-axis shows Humidity in %. The Blue line shows the real humidity values for the Random Test Samples and Green line shows the predicted humidity by our model. As the most of the graph is either overlapping or minimal difference between the real and predicted values, we can say that our model has high accuracy.

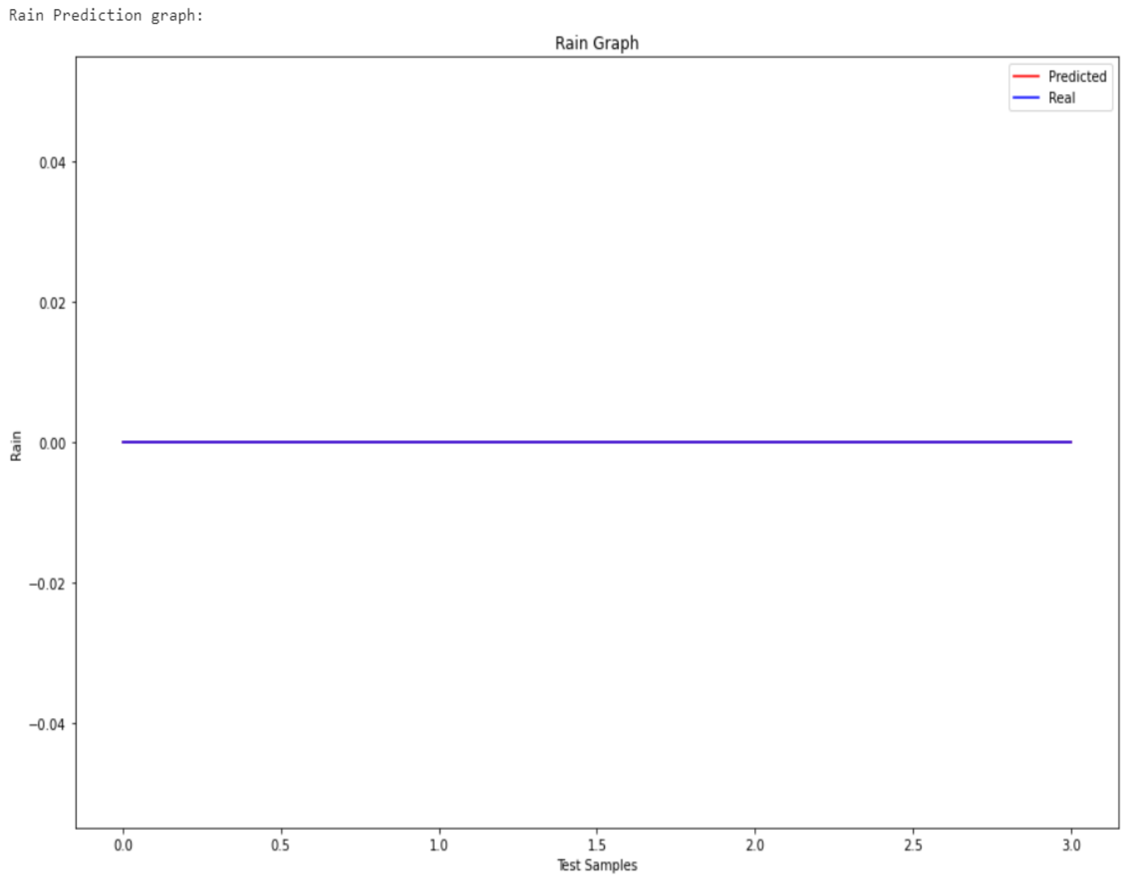
10.3 Pressure vs Test Samples:



Graph 3: Pressure vs Test Samples

The above graph shows the Pressure vs test samples result where x-axis shows Test Samples and y-axis shows Pressure in hPa. The Red line shows the real pressure values for the Random Test Samples and Blue line shows the predicted pressure by our model. As the most of the graph is either overlapping or minimal difference between the real and predicted values, we can say that our model has high accuracy.

10.4 Rain vs Test Samples:

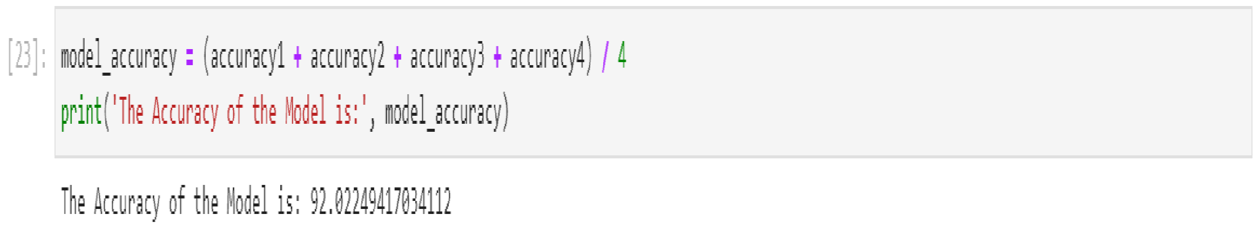


Graph 4: Rain vs Test Samples

The above graph shows the Rain vs test samples result where x-axis shows Test Samples and y-axis shows Rain. The Blue line shows the real rain values for the Random Test Samples and Red line shows the predicted rain by our model. As the entire graph is at a fixed 0.0 for both predicted and real values it shows that for the test samples selected there has not been any rain and so for accuracy of rain for these test samples is 100%.

11.RESULT AND CONCLUSION

We were able to successfully create a weather station and implemented a weather model to predict various weather parameters like Temperature, Humidity, Pressure, Rain using a data set with an overall accuracy of 92.02%.



```
[23]: model_accuracy = (accuracy1 + accuracy2 + accuracy3 + accuracy4) / 4
      print('The Accuracy of the Model is:', model_accuracy)

      The Accuracy of the Model is: 92.02249417034112
```

Figure 15: Accuracy

What makes this project so special is its pricing and size. Using ESP32 we can connect and control various sensors from anywhere in the world as long as it is connected to internet. ESP32 has inbuilt WIFI and Bluetooth which reduces use of extra hardware like ethernet shield, WIFI modules, etc. Being cost efficient and compact size, it can be easily placed and provides Real Time Parameters rather than of the whole City and could also be used in Agricultural areas providing Weather Analysis and Forecast. The Project could be further improved by adding wind sensor, direction sensor, gas or particle sensors. The future scope of this project is making a Wireless Sensor Network which contains a module for every 10 meters so that we can get more area specific results and could be used for earlier Natural Disaster Alarms example in case of a fire outbreak in forest the sudden rise in parameters can send an alarm rather than getting heat signatures from satellite when the fire would have consumed much of the forest and would be difficult to extinguish.

12.ANNEXURE

Arduino Code:

```
//including necessary libraries

#include <WiFi.h>

#include "Wire.h"

#include "Adafruit_Sensor.h"

#include "Adafruit_BME280.h"

#include "BH1750.h"

#include "Adafruit_MQTT.h"

#include "Adafruit_MQTT_Client.h"

#include "config.h"


WiFiClient client;


//MQTT Client Setup

Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

Adafruit_MQTT_Publish Temperature = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/weather-station.temperature");

Adafruit_MQTT_Publish Pressure = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/weather-station.pressure");

Adafruit_MQTT_Publish Humidity = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/weather-station.humidity");

Adafruit_MQTT_Publish Light = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/weather-station.light");
Adafruit_MQTT_Publish Rain = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/weather-station.rain");

Adafruit_BME280 bme;
BH1750 lightMeter;
const byte rainPin = 34;

struct sensorData
  {
     byte humidity;
     int temperature;
     byte rain;
     int  pressure;
     int light;
  };

sensorData sensors;

void setup()
{
    Serial.begin(115200);
    Serial.print(F("Connecting to "));
    Serial.println(WIFI_SSID);
    WiFi.begin(WIFI_SSID, WIFI_PASS);
    while (WiFi.status() != WL_CONNECTED)
    {delay(500);
    Serial.print(F("."));}
    Serial.println();
    Serial.println(F("WiFi connected"));
    Serial.println(F("IP address: "));
    Serial.println(WiFi.localIP());
    // connect to adafruit io
    connect();
}

void connect() {
  Serial.print(F("Connecting to Adafruit IO... "));
  int8_t ret;
  while ((ret = mqtt.connect()) != 0) {
    switch (ret) {
      case 1: Serial.println(F("Wrong protocol")); break;
      case 2: Serial.println(F("ID rejected")); break;
      case 3: Serial.println(F("Server unavail")); break;
      case 4: Serial.println(F("Bad user/pass")); break;
      case 5: Serial.println(F("Not authed")); break;
      case 6: Serial.println(F("Failed to subscribe")); break;
      default: Serial.println(F("Connection failed")); break;
    }
    if(ret >= 0)
      mqtt.disconnect();

    Serial.println(F("Retrying connection..."));
    delay(10000);
  }
  Serial.println(F("Adafruit IO Connected!"));
}

void loop()
{   if(! mqtt.ping(3)) {
    // reconnect to adafruit io
    if(! mqtt.connected())
      connect();
  }
    updateSenzors();
    transmitData();
    delay(10000);
}

void updateSenzors()
{
    bme.begin(0x76);
    lightMeter.begin();
    delay(300);
    sensors.temperature = bme.readTemperature();
    sensors.pressure = bme.readPressure() / 100.0F;
    sensors.humidity = bme.readHumidity();
    sensors.light = lightMeter.readLightLevel();
    sensors.rain = readRain();
}

void transmitData()
{
    Serial.print("Temp: ");Serial.print(sensors.temperature);Serial.println(" C");
    Serial.print("Humid: ");Serial.print(sensors.humidity);Serial.println(" %");
    Serial.print("Pressure: ");Serial.print(sensors.pressure);Serial.println(" hPa");
    Serial.print("Light: ");Serial.print(sensors.light);Serial.println(" lx");
    Serial.print("Rain: ");Serial.println(sensors.rain);
    if (! Temperature.publish(sensors.temperature)) {          //Publish to Adafruit
      Serial.println(F("Failed"));
    }
    if (! Pressure.publish(sensors.pressure)) {          //Publish to Adafruit
      Serial.println(F("Failed"));
    }
    if (! Humidity.publish(sensors.humidity)) {          //Publish to Adafruit
      Serial.println(F("Failed"));
    }

    if (! Light.publish(sensors.light)) {                    //Publish to Adafruit

      Serial.println(F("Failed"));

    }

    if (! Rain.publish(sensors.rain)) {                      //Publish to Adafruit

      Serial.println(F("Failed"));

    }

    else {

      Serial.println(F("Sent!"));

    }

    delay(2000);

}


byte readRain()

{

    byte level = analogRead(rainPin);


    return map(level, 0, 1023, 0, 100);

}
```

13.REFERENCES

[1] H. Üçgün and Z. K. Kaplan, "Arduino based weather forecasting station," *2017 International Conference on Computer Science and Engineering (UBMK)*, 2017, pp. 972-977.

[2] Shriya Kaneriya, Sudeep Tanwar, Srushti Buddhadev, Jai Prakash Verma, Sudhanshu Tyagi, Neeraj Kumar, Sudip Misra, "A Range-Based Approach for Long-Term Forecast of Weather Using Probabilistic Markov Model", *Communications Workshops (ICC Workshops) 2018 IEEE International Conference on*, pp. 1-6, 2018.

[3] Abhinav Malhotra, Subhranil Som, Sunil Kumar Khatri, "IoT Based Predictive Model for Cloud Seeding", *Artificial Intelligence (AICAI) 2019 Amity International Conference on*, pp. 669-773, 2019.

[4] Aishwarya Karra, Bhuvana Kondi, Ramesh Jayaraman, "Implementation of Wireless Communication to Transfer Temperature and Humidity Monitoring Data using Arduino Uno", *Communication and Signal Processing (ICCSP) 2020 International Conference on*, pp. 1101-1105, 2020.

Surya

ORIGINALITY REPORT

8%	8%	7%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	archive.org Internet Source	5%
2	csvlint.io Internet Source	1%
3	www.pubmedcentral.nih.gov Internet Source	1%
4	hdl.handle.net Internet Source	1%
5	noemata.anart.no Internet Source	1%
6	M.J. Shalkhauser, W.A. Whyte. "Digital codec for real-time processing of broadcast quality video signals at 1.8 bits/pixel", IEEE Global Telecommunications Conference, 1989, and Exhibition. 'Communications Technology for the 1990s and Beyond, 1989 Publication	<1%
7	Roy G. Jones. "III. RHYTHMIC UNITS COMPOSING THE LINES", Walter de Gruyter GmbH, 1972	<1%

Publication

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		