PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Report on

'Weather Station with ML capabilities'

Submitted by

Sujal Yatin Vaidya (PES1UG22EC299)
Thithikshaa S (PES1UG22EC319)

Jan – May 2024

Under the guidance of

Internal Guide

Prof. Prajeesha
Assistant Professor
Department of ECE PES
University
Bengaluru -560085

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGG

PROGRAM B.TECH

CERTIFICATE

This is to certify that the Report entitled

'Weather Station with ML capabilities'

is a bona fide work carried out by

Sujal Yatin Vaidya (PES1UG22EC299)
Thithikshaa S (PES1UG22EC319)

In partial fulfillment for the completion of course work in the Program of Study B.Tech in Electronics and Communication Engineering, under rules and regulations of PES University, Bengaluru during the period Jan - May 2024. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The report has been approved as it satisfies the academic requirements in respect of JoEL Lab project work.

Signature with date & Seal
Prof. Prajeesha
Internal Guide

Signature with date & Seal
Dr. Anuradha M
Chairperson

Signature with date & Seal
Dr. B. K. Keshavan
Dean - Faculty of Engg. &Technology

DECLARATION

We, Sujal Yatin Vaidya (PES1UG22EC299) and Thithikshaa S (PES1UG22EC319)**,** hereby declare that the report entitled, '***Weather Station with ML capabilities',*** is an original work done by us under the guidance of **Prof. Prajeesha**, *Assistant Professor*, ECE Department and is being submitted in partial fulfillment of the requirements for completion of course work in the Program of Study, B.Tech in Electronics and Communication Engineering.

PLACE: PES University
DATE: 25-7-2023

NAME AND SIGNATURE OF THE CANDIDATES

1. Sujal Yatin Vaidya
2. Thithikshaa S

ABSTRACT

This project presents the design and implementation of a Weather Station with Machine Learning (ML) capabilities. The system utilizes an ESP8266 NodeMCU microcontroller to collect real-time environmental data using sensors such as DHT11, BMP280, and a rain sensor. The data is transmitted to a web server, where it is displayed on a dashboard with graphical representations and predictions. Machine Learning algorithms are applied to forecast future weather conditions based on the collected data. The project demonstrates the integration of IoT, web development, and ML techniques to create a comprehensive weather monitoring and prediction system.

METHODOLOGY

The methodology involved designing a hardware setup for environmental data collection, developing firmware for the ESP8266 NodeMCU, and creating a web dashboard for data visualization. The collected data includes temperature, humidity, pressure, and rainfall, which are processed and stored on a server. A Python backend processes the data and uses machine learning models for weather prediction. The frontend, built with HTML, CSS, and JavaScript, provides a user-friendly interface to display the data and predictions.

Software Required

- **Arduino IDE**: For programming the ESP8266 NodeMCU.
- **Python:** For data processing and machine learning.
- **Python Libraries:** TensorFlow, scikit-learn, requests, pandas, numpy, Openmeteo API
- **Web Development Tools**: HTML, CSS, JavaScript (Chart.js for graphs), Bootstrap
- **NodeMCU Libraries**: ESP8266WiFi, ESP8266WebServer, ArduinoJson, Adafruit BMP280, DHTesp

Hardware Required

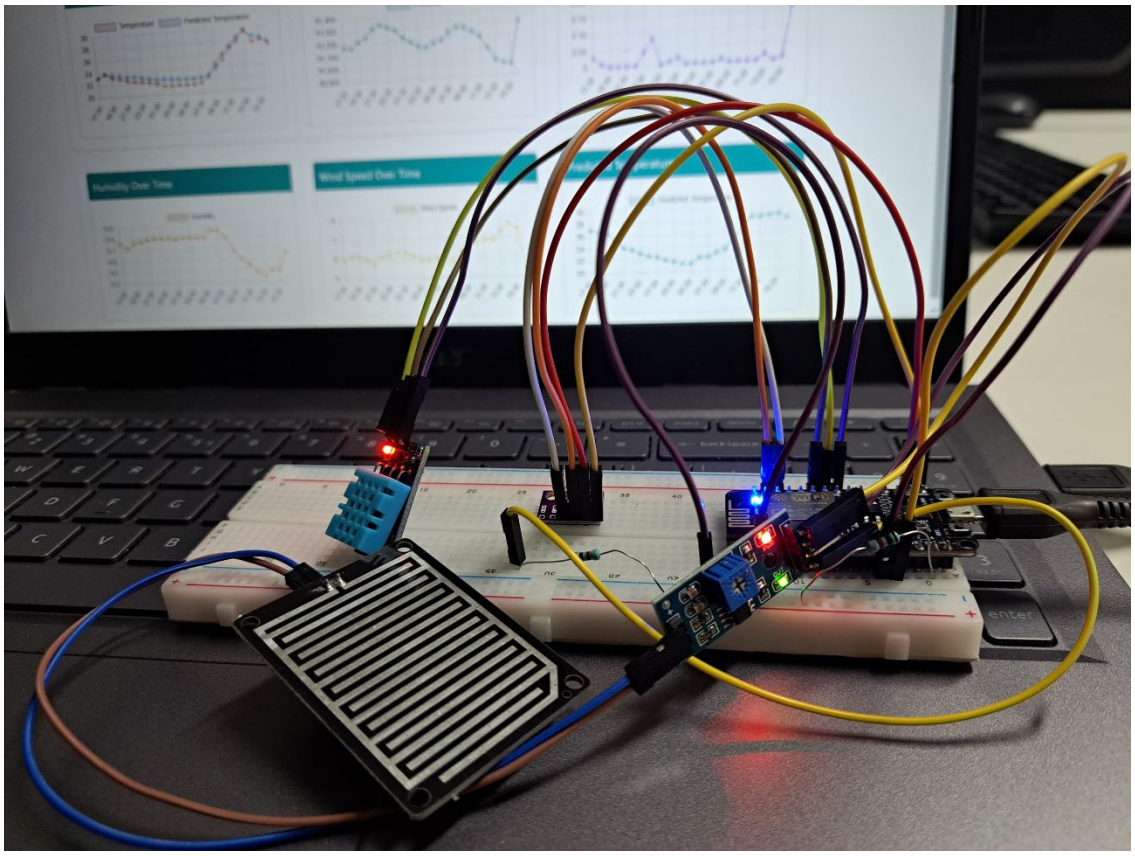
- **ESP8266 NodeMCU:** Microcontroller for data acquisition and WiFi connectivity.
- **DHT11 Sensor**: For measuring temperature and humidity.
- **BMP280 Sensor:** For measuring atmospheric pressure.
- **Rain Sensor**: For detecting rainfall.
- **Power Supply**: To power the NodeMCU and sensors.

DATA RETRIEVAL AND PROCESSING

The data for this project was retrieved from various sources, including a local weather station equipped with ESP8266 NodeMCU and sensors like DHT11, BMP280, and a rain sensor. Additionally, wind speed data was obtained using a third-party API, which provided real-time information. Also, historical weather data of the past 7 years was downloaded from a third-party site. The data collection process involved sending HTTP GET requests to the API and storing the responses in a structured format. The data was logged in a CSV file, which included columns for temperature, humidity, pressure, wind speed, and other relevant weather parameters.

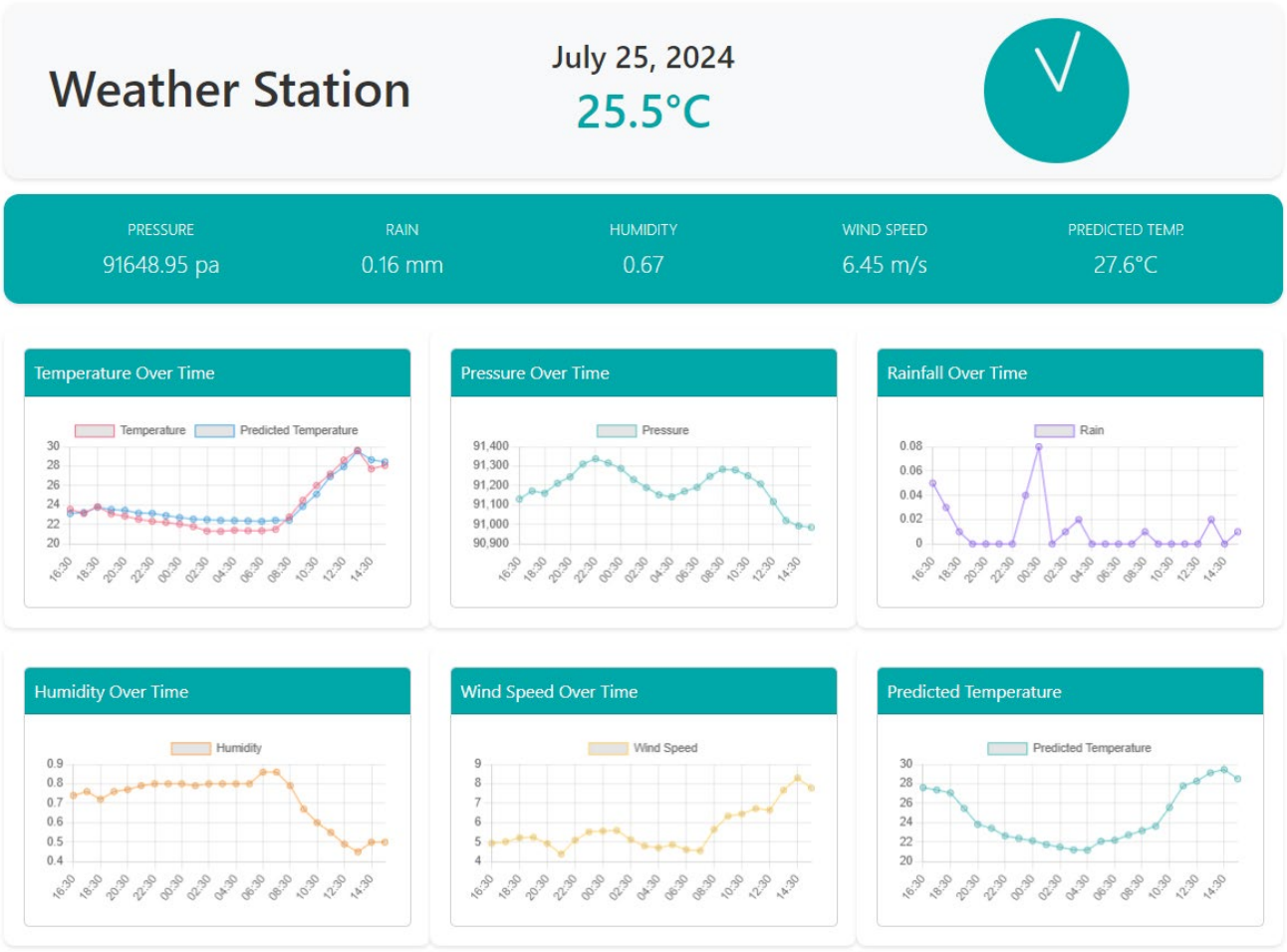
SERVER SETUP AND DATA ACCESS

The weather data collection involves a physical setup with an ESP8266 NodeMCU microcontroller. This microcontroller serves as a web server using the ESP8266WebServer library, enabling the collection and distribution of sensor data. The server receives data sent from a Python script using the requests library along with data received from other sensors, ensuring real-time updates and seamless integration with the data processing workflow.



DASHBOARD IMPLEMENTATION

The weather data was displayed on a web dashboard built with HTML, CSS, and JavaScript, using Bootstrap for responsive design. The dashboard featured interactive line charts created with the Chart.js library, showing real-time and historical data for temperature, humidity, pressure, wind speed, and rain. Users could access the dashboard via a web browser, where they could view visualizations and trends over time. The dashboard also includes predicted temperature values, leveraging machine learning models.



MACHINE LEARNING IMPLEMENTATION

Data Preparation

The machine learning component involved predicting future temperatures based on historical weather data. The data was preprocessed by first loading it from a CSV file. The 'datetime_utc' column was converted into a datetime object, and rows with missing values were dropped. To capture temporal dependencies, lag features for temperature were added, ranging from 1 to 24 hours. Additionally, cyclical features for hour, month, and day of the year were created to account for periodic variations.

Model Selection and Training

Various machine learning models were evaluated for predicting temperature, including Random Forest, XGBoost, Neural Networks, Linear Regression, and Gradient Boosting. After a comprehensive analysis, the Random Forest Regressor was selected for its superior performance. The model was trained using the most relevant features, with a training set obtained by splitting the data into 80% training and 20% testing.

The best hyperparameters for the Random Forest model were identified using GridSearchCV. The chosen parameters were:

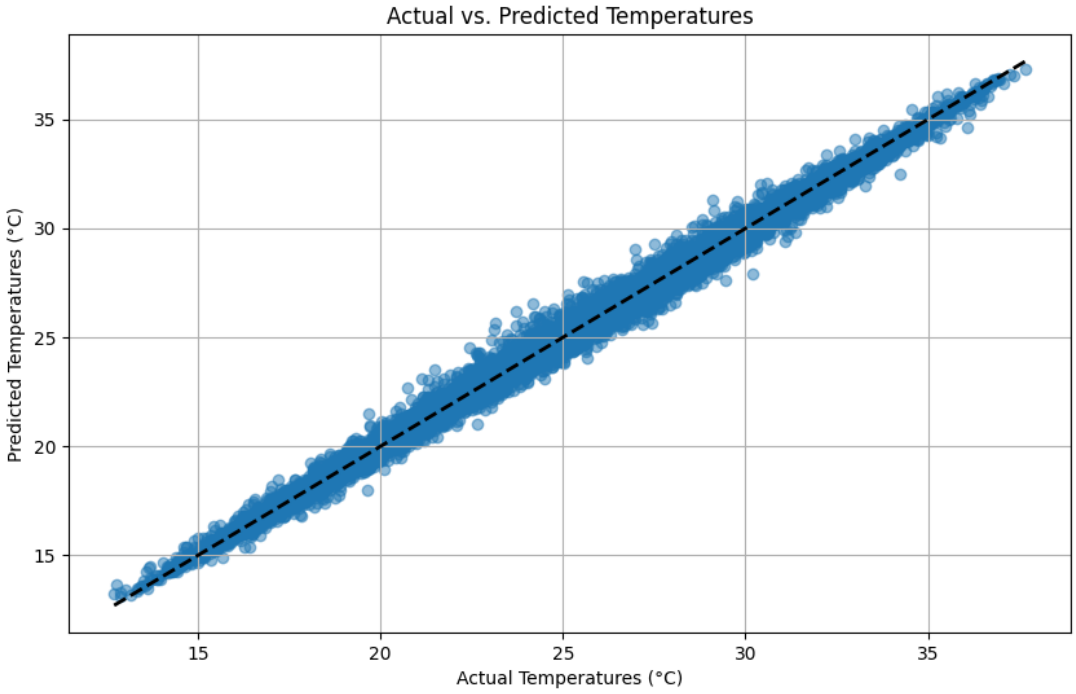
- max_depth: 20
- min_samples_leaf: 1
- min_samples_split: 2
- n_estimators: 300

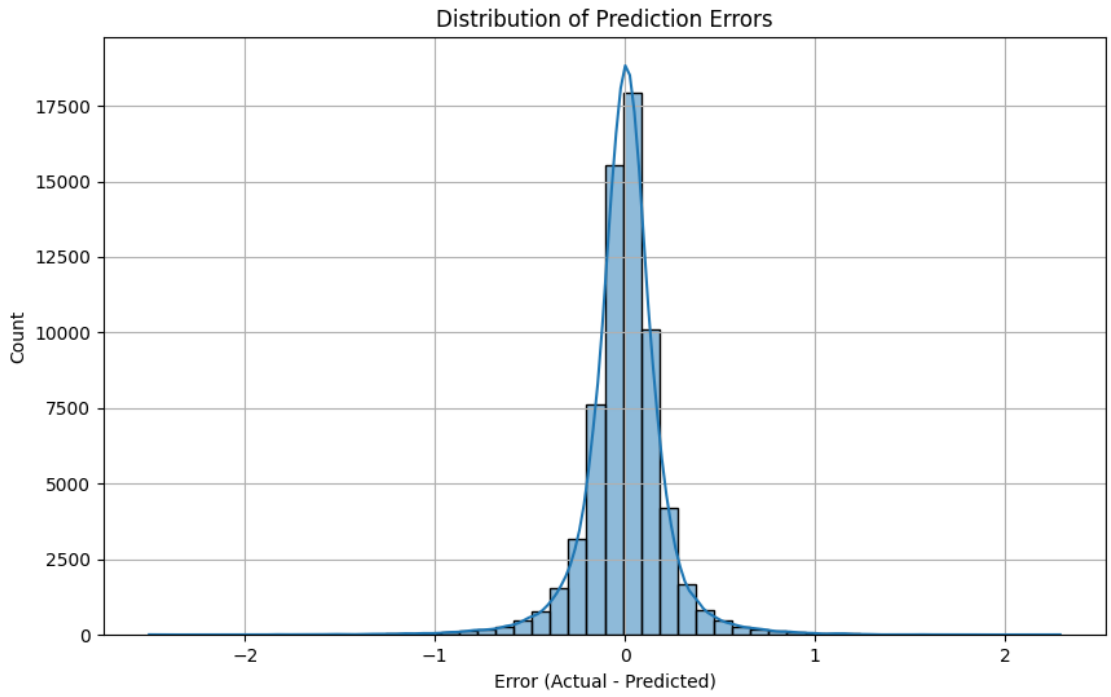
The model was trained on scaled data using StandardScaler, and the trained model was saved using Joblib for future use.

Model Evaluation and Results

The model's performance was evaluated using metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), R^2 Score, and Explained Variance Score. The Random Forest model achieved an MSE of 0.136, an MAE of 0.269, an R^2 Score of 0.990, and an Explained Variance Score of 0.990. These results indicated a high level of accuracy in predicting temperature, demonstrating the model's effectiveness in capturing the underlying patterns in the data.

The predictions were saved and compared with the actual values, confirming the model's reliability and precision. The saved model and scaler were integrated into the dashboard for real-time predictions and user interaction.





RESULTS

The weather dashboard effectively integrates seven years of historical weather data from Bangalore, utilizing a variety of sensors and an API for wind speed data. This extensive dataset, combined with the ESP8266 NodeMCU as a server, enables real-time data collection and display on the dashboard. Users can access the dashboard via a browser to view up-to-date weather conditions and trends, presented through intuitive line charts powered by Chart.js.

The machine learning model, a Random Forest Regressor, was selected after thorough testing and comparison with other algorithms such as XGBoost, Neural Networks, Linear Regression, and Gradient Boosting. The model, optimized through hyperparameter tuning, demonstrated outstanding predictive capabilities, achieving a Mean Squared Error (MSE) of 0.136, a Mean Absolute Error (MAE) of 0.269, an R^2 Score of 0.990, and an Explained Variance Score of 0.990. These metrics indicate the model's high accuracy and reliability in forecasting temperature changes.

Overall, the project successfully combines hardware, software, and advanced analytics to deliver a comprehensive weather monitoring and prediction system. The dashboard provides valuable insights into weather patterns, assisting users in planning and decision-making based on accurate and timely weather data.

REFERENCES

[1] https://docs.oikolab.com

[2] https://pypi.org/project/openmeteo-requests/

[3] https://github.com/adafruit/Adafruit_BMP280_Library

[4] https://github.com/beegee-tokyo/DHTesp

[5] https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WebServer

[6] https://www.chartjs.org/docs/latest/

[7] https://scikit-learn.org/stable/