Smart Weather Station

Kasun Thushara

B.Sc. Eng (Hons) Electronic and Electrical Engineer

Research Scientist - SLINTEC

Introduction

The goal of weather prediction is to provide information people and organizations can use to reduce weather-related losses and enhance societal benefits, including the protection of life and property, public health and safety, and support for economic prosperity and quality of life. In simple words, weather data is the state of Earth's atmosphere at a given geographical location at a given time. Weather data provides invaluable insights through meteorological parameters like

- Temperature
- Cloud Cover
- Pressure
- Humidity
- Dew Point
- Precipitation
- Visibility
- Wind Speed

Weather data is aggregated by combining multiple weather sources—on-ground sensors, satellite imagery, and statistical inference—to ensure the accuracy and availability of the highest order. But there are a few constraints. The atmosphere is changing all the time, and due to that, the above parameters can be changed. As an example, the temperature can change rapidly in space due to land-cover heterogeneity and changing altitude over complex mountainous terrain. This means that a weather station ten kilometers away may measure conditions that have little relevance to your location, making it hard to make informed local decisions. So those estimates are less reliable the further you get into the future. It is important to predict short-term, long-term, and medium-

term weather for industries that are interested in weather data. A numerical weather prediction model, or NWP, solves a complex set of mathematical equations that are based on the physics that drives how the air moves and how heat and moisture are exchanged throughout the atmosphere. The two best-known NWP models are the National Weather Service's Global Forecast System, or GFS, and the European Center for Medium-Range Weather Forecast, known as the ECMWF model. They are also known as the American and European models, respectively. Generally speaking, the European model has produced the most accurate global weather forecasts. Attempts can be made to supplement weather information with satellite data, but these infer rainfall and near-surface temperature at spatially averaged scales exceeding 10 km and are subject to considerable uncertainties despite advances in sensor technology; moreover, some data such as near-surface winds or solar flux are not available at all. Local weather information can have a huge impact on communities and applications enabled by granular weather data, including agriculture, changes in the prices of commodities, and early warning systems.

Background

Background Using review papers, I could find out what the recent machine learning approaches to predicting the weather are. A. Mahabub [1] makes a model to predict weather parameters using maximum and minimum temperatures, humidity, wind speed, etc. The work was carried out on many machine-learning models. It used several ML techniques like Support Vector Regression (SVR), Linear Regression, Bayesian Ridge, Gradient Boosting (GB), Extreme Gradient Boosting (XGBoost), Category Boosting (CatBoost), Adaptive Boosting (AdaBoost), k-Nearest Neighbors (KNN), and Decision Tree Regressor (DTR). The performance was analyzed for each model. It can be said that DTR and CatBoost methods were almost equivalent, but the adaptability of DTR was better for nonlinear data. A. Parashar's [2] work carried on to predict the next day's minimum, maximum, and mean temperatures. A multiple linear regression model is used. Results can be categorized as follows:94% accuracy was achieved for the next-day minimum temperature.

- 93% accuracy was achieved for the next day's maximum temperature.
- 95% accuracy was achieved for the next day's Mean temperature.

Singh et al. [3] used the random forest classifier to predict rainfall. The accuracy of the model is 87.8%. The input data were temperature, humidity, and pressure. [4] had predicted rainfall based on hourly meteorological data such as pressure, temperature, humidity, wind speed, wind direction, sea level, and rainfall using deep learning techniques such as the echo state network (ESN) and the deep echo state network (DeepESN). The accuracy of the predicted rainfall by using the DeepESN was improved compared with that of the ESN, the BPN, and the SVR.

Hardware architecture

Microcontroller- By using resources there are two constraints to choosing a suitable microcontroller. Those are

1. Ability to deploy the tiny-ML model
2. IoT enabled SoC

So, in this case, I have to go through some internet resources to find the best suitable Microcontroller to deploy the tiny-ML model and IoT to enable the device to gather the data on the cloud and further improvements of the system. So, I chose ESP 32 Dev board which has both capabilities.

Choosing sensors

To identify temperature and humidity -DHT 11

To identify pressure -BMP280 and BMP180

Since there is no moving part in the hardware component. The wind speed was achieved by using Bernoulli's equation. I have installed those two sensors outside of the enclosure and also inside the enclosure. I assumed that the internal wind speed is zero.

$$v2 = \left[\frac{2(p1 - p2)}{\rho}\right]^{\frac{1}{2}}$$

V_2 is the speed of the wind and p1 and p2 are the pressure difference. ρ is air density.

To identify rain – Rain Sensor

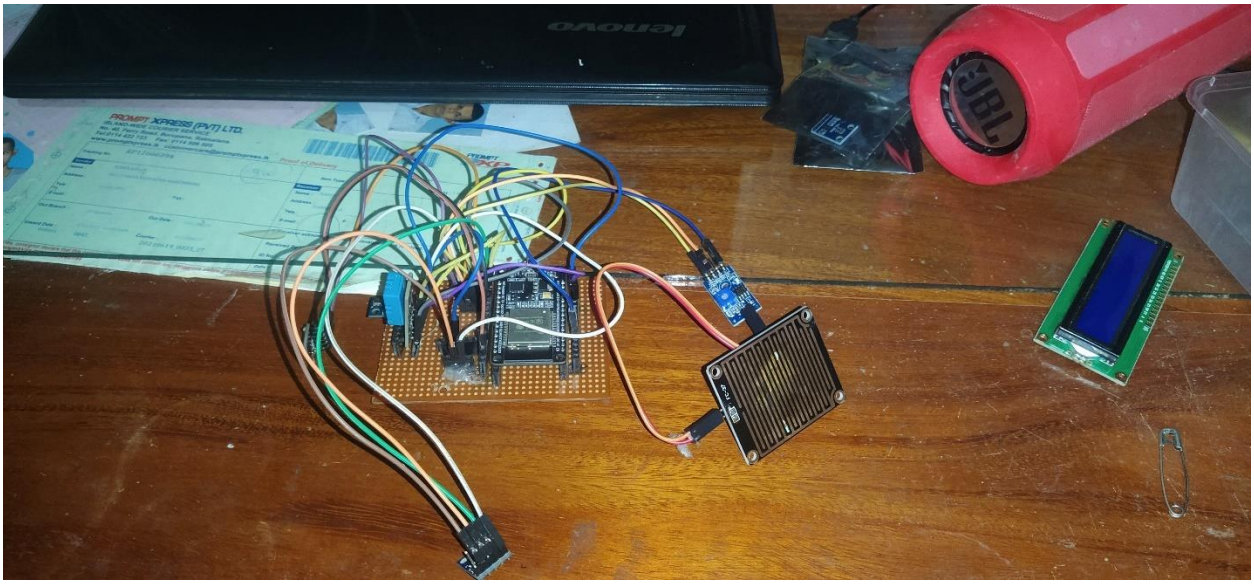


Figure 01: Esp32 microcontroller and sensor attachments

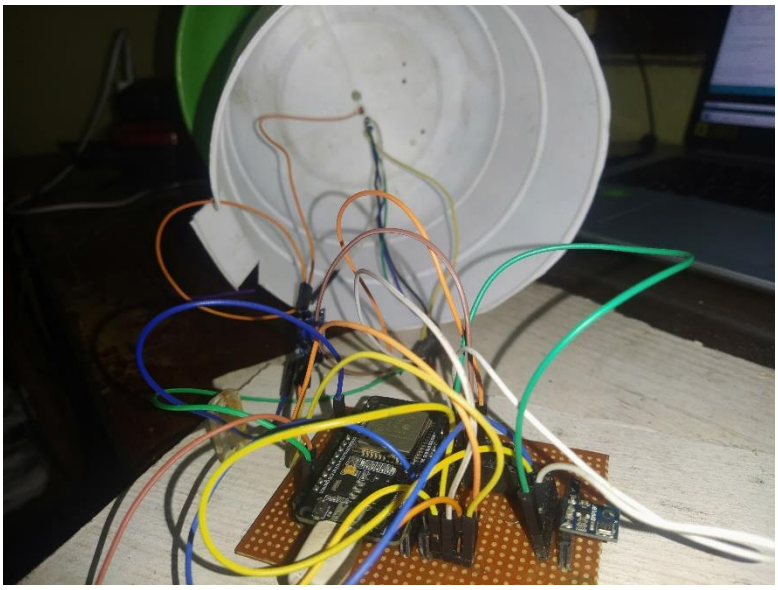


Figure 02: Embedded system and enclosure

The enclosure is a 4-liter ice cream cup and I put a plastic plate. All of those are waste materials in Sri Lanka and it's common.

Figure 03: Weather Station

Data aquation

Temperature, humidity, air pressure, wind speed, and precipitation factor are the points that I figured out to predict the rain status. So I used an IoT-based microcontroller since I could collect the data in a Google Spreadsheet. I used Google Script to create an event when the payload arrived as an HTTP request, and then it splits into columns. Every 3600 seconds, the data will be transmitted to the sheet. I collected data from October 1st to November 10th. Overall, there were 985 data sets. I added a new column to show whether the next hour had rain or not by checking the previous hour. 20% of data was used for testing and 80% of data was used for training. For your reference, I have attached the data set.

Model Selection

For training and testing different models, I used the Python programming language and Jupyter Notebook. I used the Scikit-Learn Python library to train the model. The models that I used here are logistic regression, decision tree, neural network, LGBM, CatBoost, Random Forest regression, and XGBoost. Then I calculate the accuracy and time taken for execution.

After making the model or models predict the forecast, it is better to optimize the model. The improvement in energy efficiency is an effective way to increase the availability of edges and expand the radius of life. Therefore, improving energy efficiency at the edge is indispensable to realizing edge intelligence.

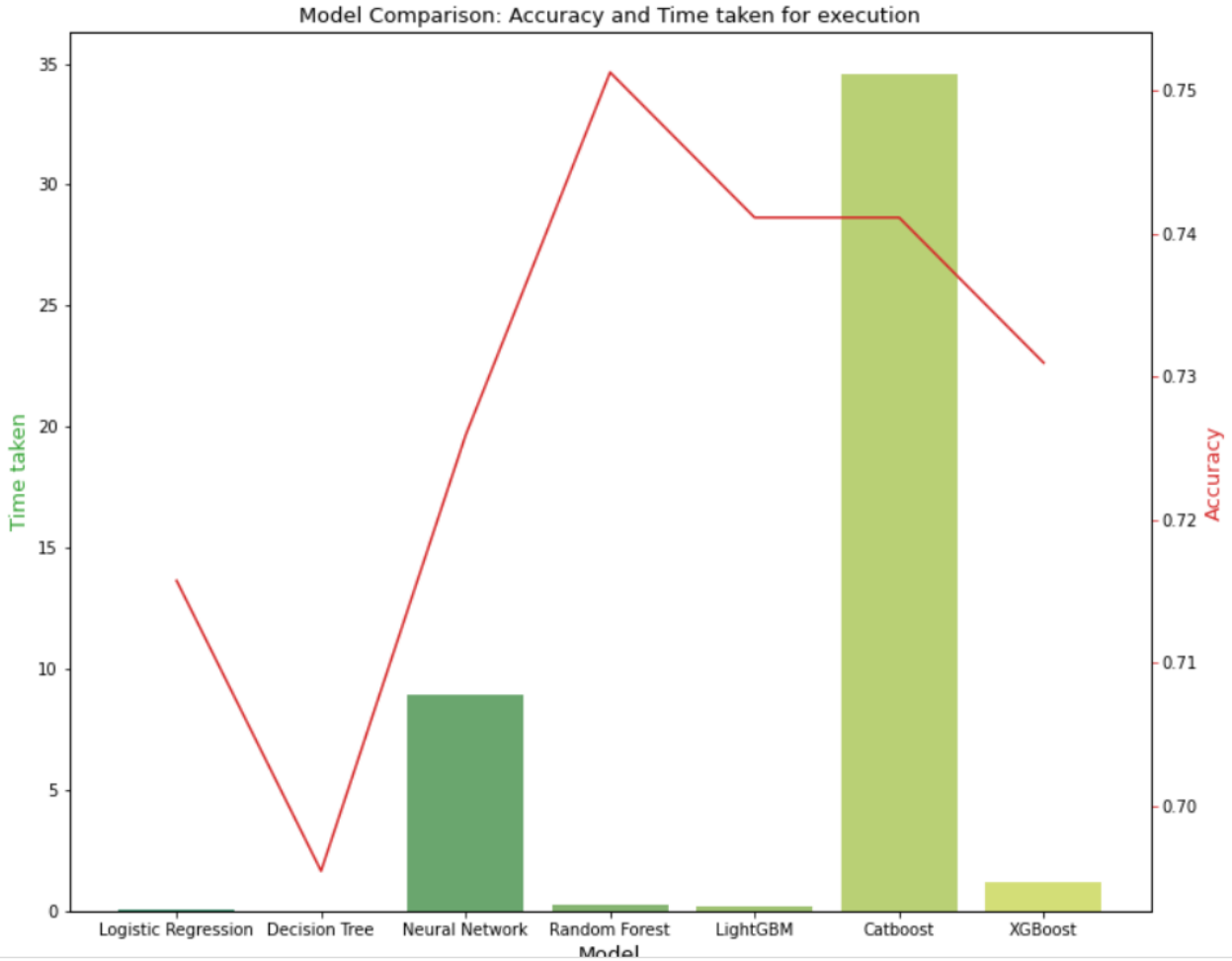
All of the models reached between 69 and 75% accuracy. Logistic regression, decision trees, random forest regressors, and LGBM have shown less time in execution. The random forest regression has a highly accurate value, and since it has a shorter execution time, this model can be chosen for deployment on MCU. (i.e., the correlation between an algorithm's complexity and execution time).

```
from sklearn.ensemble import RandomForestClassifier

params_rf = {'max_depth': 5,
             'min_samples_leaf': 1,
             'min_samples_split': 2,
             'n_estimators': 100,
             'random_state': 12345}

model_rf = RandomForestClassifier(**params_rf)
model_rf, accuracy_rf, roc_auc_rf, coh_kap_rf, tt_rf = run_model(model_rf, X_train, y_train, X_test, y_test)
```

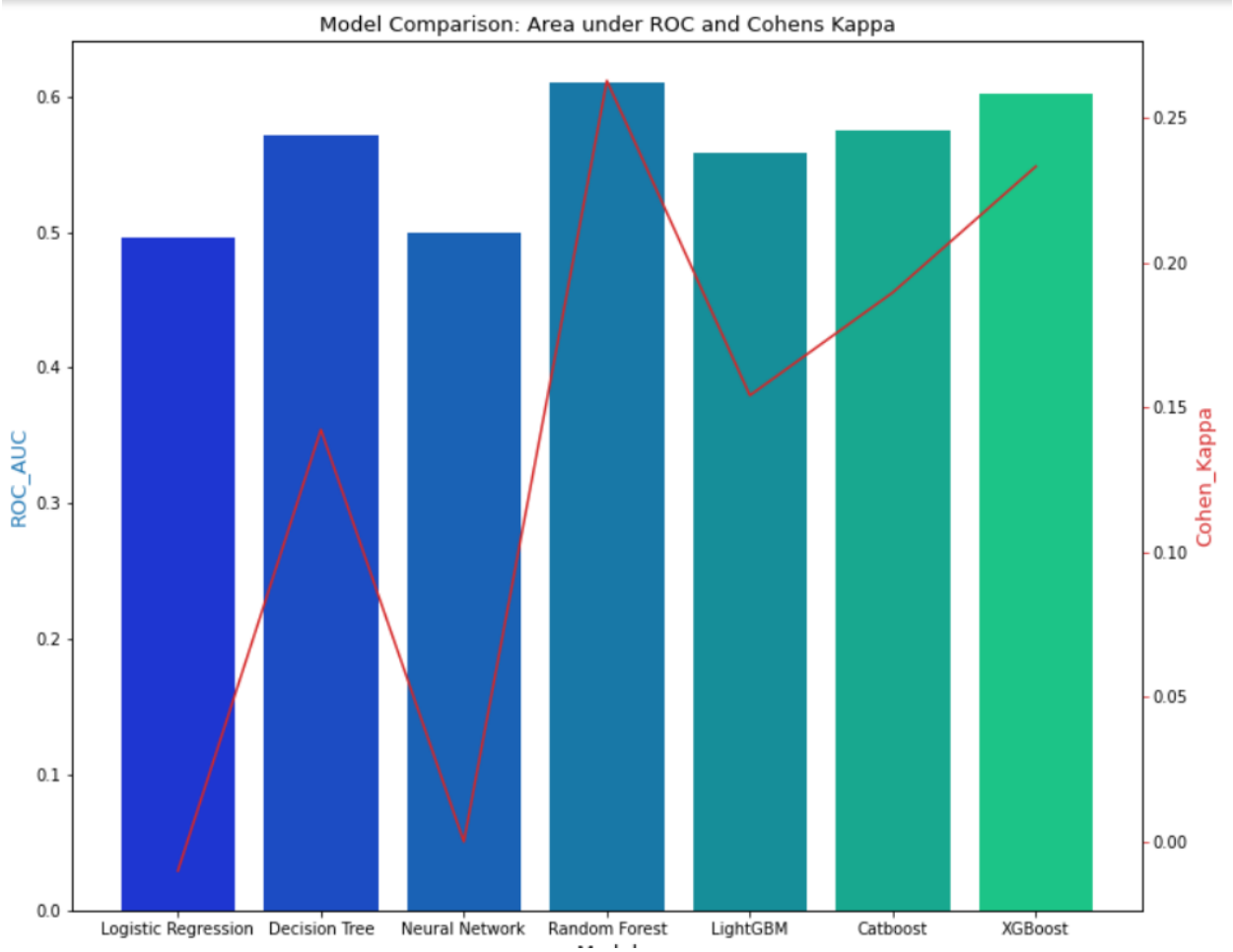
Figure 04: Random forest regression parameters

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import seaborn as sns
params_lr = {'penalty': 'l1', 'solver':'liblinear'}

model_lr = LogisticRegression(**params_lr)
model_lr, accuracy_lr, roc_auc_lr, coh_kap_lr, tt_lr = run_model(model_lr, X_train, y_train, X_test, y_test)
```

Figure 05: Logistic regression parameters



Cohen's kappa coefficient is a statistic that measures inter-rater agreement for qualitative (categorical) items. It is generally thought to be a more robust measure than a simple percent agreement calculation since k takes into account the agreement occurring by chance. Cohen's kappa measures the agreement between two raters who each classify N items into C mutually exclusive categories. The area under the curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the model's performance at distinguishing between the positive and negative classes. The Cohen Kappa value is high for the decision tree, and the ROC area under the curve is also high for the decision tree. So in this case, we can conclude that the best model we can get is the decision tree. Since all the ROC_AUC values are pretty much the same, the Cohen kappa value comes into play in this situation.



Tiny ML model

With the above parameters, it was obvious to choose the random forest regressor. MicroML library is a library in which you can convert the model into C++ code to deploy on microcontrollers. Using that I convert the random forest classifier model into C++ code and I deploy it on the Esp 32 board. Not only I converted the random forest classifier. I also tried Logistic regression and the Decision tree algorithm. I tested with the test data set. For your reference, I have attached the tiny ML models with this. If you want to change the model include the relevant model.h file to weather_predict.ino file.

Discussion and conclusion

The primary goal of this project is to create an edge device capable of measuring weather parameters and using those parameters to predict the short-term forecast using an energy-efficient tiny ML model. I believe that this data set is fair enough to make predictions, and it shows 75% accuracy. By increasing the small amount of data, I believe that this will increase to 85%. The ML model's execution time of a few milliseconds demonstrates the model's low power consumption and efficiency. For that, I can operate this for another month. Using the ESP-32 module gives me more flexibility and possibilities to improve this in an IoT manner. I hope these small weather stations can locate nearby villages to acquire data for analyzing patterns of the weather and predicting precipitation. This will be useful to generate hydropower for next year, for short-term trip planning, for agricultural industries to think about yields, etc. Adding APIs to mobile applications can improve the next user experience. Whether he or she goes outside, the weather alert will be useful to schedule his or her day.

References

[1] A. Mahabub and A. S. Bin Habib, "An Overview of Weather Forecasting for Bangladesh Using Machine Learning Techniques," pp. 1–36, 2019

[2] A. Parashar, "IoT-based automated weather report generation and prediction using machine learning," 2019 2nd Int. Conf. Intell. Commun. Comput. Tech. ICCT 2019, pp. 339–344, 2019, doi: 10.1109/ICCT46177.2019.8968782

[3] N. Singh, S. Chaturvedi, and S. Akhter, "Weather Forecasting Using Machine Learning Algorithm," 2019 Int. Conf. Signal Process. Commun. ICSC 2019, pp. 171–174, 2019, doi: 10.1109/ICSC45622.2019.8938211.

[4] M. H. Yen, D. W. Liu, Y. C. Hsin, C. E. Lin, and C. C. Chen, "Application of the deep learning for the prediction of rainfall in Southern Taiwan," Sci. Rep., vol. 9, no. 1, pp. 1–9, 2019, doi: 10.1038/s41598-019- 49242-