Real-time weather prediction system

Ta Anh Khoa, Do Trong Nhan, Nguyen Duc Thinh

University of Information Technology - Vietnam National University, Ho Chi Minh City
{21522232,21521214,18521440}@gm.uit.edu.vn

Abstract. Weather prediction is an important and necessary task in many areas of life, such as agriculture, tourism, and the management and operation of transportation systems. To meet this need, we have built a weather prediction system in a specific area. In this article, we will describe the process of building this system, including the application of data processing technology using deep learning models in the BigDL-Chronos library to train data, then conduct realtime prediction by applying the Kafka framework. Prediction results will be updated on a website we have built.

Keywords: Weather-Forecasting · BigDL-Chronos · Apache kafka · Apache Pyspark.

1 Introduction

Weather forecasting has long become an extremely necessary thing for people's daily activities. Weather forecast is not only a simple forecasting tool but also an important source of information, helping the community prepare and respond effectively to fluctuations in the natural environment. Weather forecasting plays an important role in many fields such as agriculture, entertainment and tourism, outdoor sports activities, etc. It can assist an individual, organization or business in making weather-related decisions effectively.

In this article, the team will build a real-time weather prediction system of an area using apache spark, apache kafka and deep learning models of the bigdl-chronos library. My team will deploy the LSTMForecaster, Seq2SeqForecaster and TCNForecaster models in turn to evaluate and compare them through evaluation methods such as sMAPE, RMSE to choose the most optimal model. From there, apply that model into the weather forecasting system with Apache Kafka to be able to predict in real time. In this subject project, the team will perform Multi-Variate prediction with predicted attributes including temperature, humidity, precipitation (calculated from 1 hour ago), atmospheric pressure and type of rain.

2 Related works

Weather forecasting using machine learning and deep learning has been studied for a long time. On June 22, 2020, the paper "Deep learning-based effective fine-grained weather forecasting model" was published by Pradeep Hewage, Marcello Trovati, Ella Pereira and Ardhendu Behera. This article proposes a new weather forecast model, which does not require a lot of computer and data resources, by exploring the methods of long short memory (LSTM) temporal modeling and neural networks. temporal convolution (BC). The model is compared with existing classical machine learning methods, statistical forecasting methods and a dynamic ensemble method, such as the research weather forecast model and the widely recognized NWP forecast (WRF). Weather information is recorded using time series data and therefore the authors explore advanced LSTM and TCN models, which are a form of neural network specialized for weather forecasting. Experiments show that this proposed lightweight model delivers better results than the well-known and complex WRF model, demonstrating its potential for accurate and efficient weather forecasting up to 12 hours[HTPB21].

3 Data set

To carry out this topic, my team searched for appropriate data sources. The data set must contain parameters such as temperature, humidity, and rainfall and must be continuously updated over a period of time to ensure real-time calculation. After a period of searching, the team selected the Beach Weather Stations - Automated Sensors data set from the Chicago Data Portal[Por]. The dataset was collected from 2015 until now 2024. This dataset will be updated every hour. Because the dataset contained data from two different stations, the team only pulled data from a station called Oak Street Weather Station, data from this station will contain complete information and no missing data like the other station. The dataset includes the following fields:

Field Name	**Description**	**Data Type**
Measurement Timestamp	Date and time of the measurement.	Raw text
Air Temperature	Air temperature in degrees C.	Number
Wet Bulb Temperature	Wet bulb temperature in degrees C.	Number
Humidity	Percentage of relative humidity.	Number
Rain Intensity	Rain intensity in mm/hour.	Number
Interval Rain	Rain since the last hourly measurement, in mm.	Number
Total Rain	Total rain since midnight in mm.	Number
Precipitation Type	0 = No rain 60 = Liquid precipitation, e.g. rain - Ice, hail and sleet are transmitted as rain (60). 70 = Solid precipitation, e.g. snow 40 = Unidentified precipitation.	Number
Wind Direction	Wind direction in degrees.	Number
Wind Speed	Wind speed in meters/second.	Number
Max Wind Speed	Maximum wind speed in a two-minute interval.	Number
Barometric Pressure	Atmospheric pressure in hPa.	Number
Solar Radiation	Solar radiation in watts per square meter.	Number
Heading	Current heading of the wind measurement unit.	Number
Battery Life	Battery voltage, indicating remaining battery life.	Number
Measurement Timestamp Label	Last updated value in text format.	Raw text
Measurement ID	Unique record ID created from Station Name and Measurement Timestamp.	Raw text

Table 1. Weather Data

After visualizing the data, my team realized that the data was not continuous from 2015 to the present time but would be missing at certain periods in the past. This will make the data set not guaranteed to be continuous, which can cause difficulties in model training and prediction..

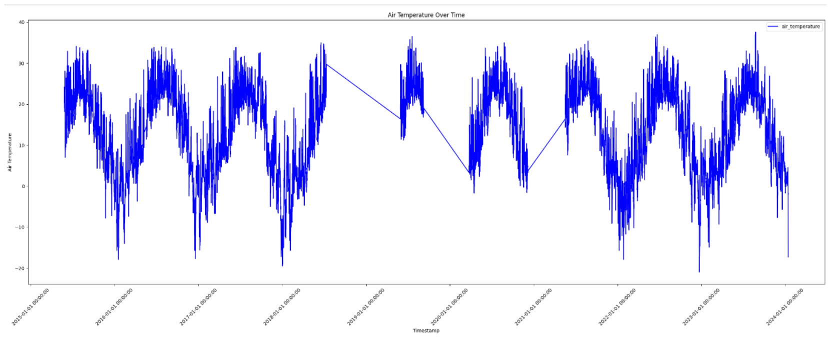


Fig. 1. Air Temperature Over Time

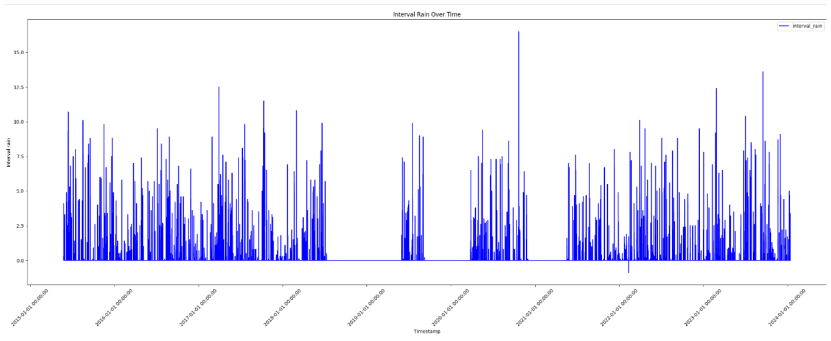


Fig. 2. Interval Rain Over Time

4 Used technologies and tools

4.1 Apache PySpark

PySpark, a Python package for Apache Spark, serves not only as a bridge between the power of Apache Spark and the flexibility of the Python programming language but also , opens up a new realm for handling large-scale data. With PySpark, the benefits of Apache Spark become more accessible to the Python programming community. The combination of Spark's high performance and Python's multitasking capabilities provides a robust programming experience, from distributed data processing to building complex machine learning models.

PySpark is not just a tool connecting Python to Spark; it also comes with flexible APIs for essential libraries such as Spark SQL, MLlib, and GraphX. This facilitates programmers in performing complex tasks seamlessly, including data querying, building machine learning models, and efficiently processing large graphs.

4.2 Apache Kafka

Apache Kafka is a cloud-based data processing system developed by the Apache Software Foundation, offering a powerful solution for real-time data processing and transmission. Built on a publish-subscribe architecture, Kafka provides the flexibility and high reliability needed to handle thousands of events per second. Apache Kafka is an ideal choice in a real-time weather forecasting system.

4.3 BigDL-Chronos

BigDL-Chronos is a framework for rapidly building accurate and scalable time series analysis applications. It is part of the BigDL project, an open-source framework for building comprehensive AI that scales across distributed Big Data. BigDL-Chronos leverages Ray and its native libraries to support advanced AI use cases such as AutoML and Automatic Time Series Analysis. It is designed to handle scalable time series analysis using AutoML

In the context of a weather forecasting system, BigDL-Chronos can be utilized to build an accurate weather prediction model based on time series data. The deep learning models in BigDL-Chronos can be trained on historical weather data to predict weather trends in the future. This can enhance the accuracy of weather forecasts and provide valuable information for weather-dependent activities such as agriculture, tourism, and disaster management. Furthermore, with the real-time data processing capability of Apache Kafka, weather predictions can be continuously updated to reflect the latest changes in weather data.

In this project, the team will deploy standalone deep learning models such as TCNForecaster, Seq2SeqForecaster, and LSTMForecaster to predict weather in the next hour. The strengths of these models lie in their ability to predict multi-step or multi-variate data, enabling the system to forecast future data more extensively and predict multiple fields simultaneously.

5 Approach Method

5.1 Data Collection

The dataset is divided into two types: training data and data used for real-time prediction. Training data will be obtained by downloading a data file from the Chicago Data Portal in csv format. Meanwhile, prediction data would be obtained through an API provided by the developer portal. Retrieving data through this API will require registering an account and creating an app token on the developer portal website. After that, the data retrieval would be handled by the Socrata module - A module used to retrieve open data from the Chicago Data Portal.

5.2 Data Preprocessing

After the data is collected, it would be preprocessed before training the model. Time series data will be converted into timestamp data, then all samples will be sorted in ascending order by the newly created timestamp column. In addition, the data will also be filled with missing values (if any), duplicate samples will be removed, additional columns related to timestamp (hour, day of the week, month, year, etc.) will be created, and the data will be normalized (in the training set) before being used to train the model. Finally, the original dataset, which has missing sets leading to uneven intervals, will be resampled to bring the interval back to 1 hour, thereby ensuring the completeness and continuity of the dataset over time.

5.3 Correlations and Variable Selection

To improve the predictive performance of the model, in addition to the timestamp variable containing time data, the team has constructed a matrix showing the correlation of variables with each other to select independent variables that affect the target variables. Through this, the model will be able to predict better based on the selected independent variables.

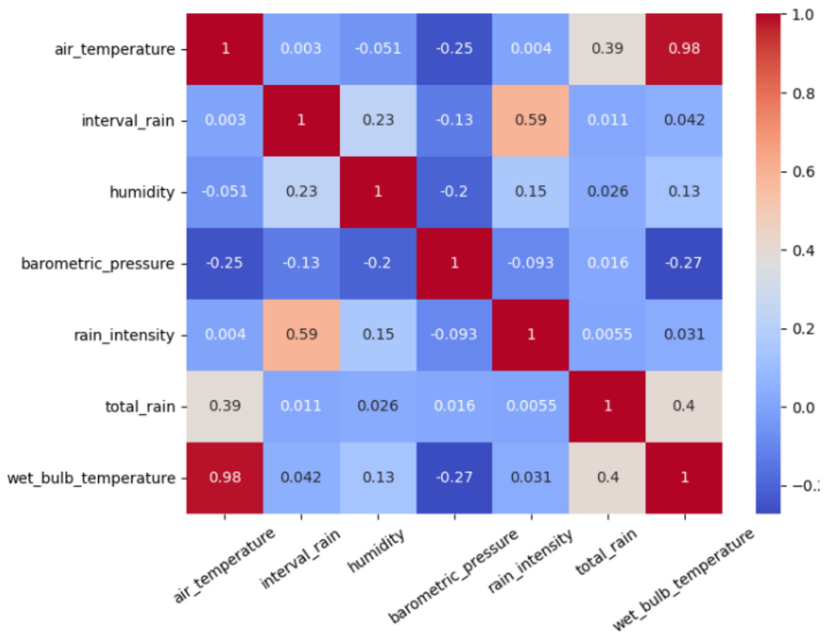


Fig. 3. Correlations between features

In the matrix above, we can see that there are very few independent variables that affect the target variables. Only rain intensity affects interval rain (0.59), wet bulb temperature

(0.98) and total rain (0.39) affect the air temperature variable. Therefore, the team will choose these variables along with the timestamp variable to train the model.

5.4 Building, Evaluating and Comparing Models

Models selection: There are two types of time series prediction:

– Traditional – Traditional Statistical (TS) Style: In this style, statistical models are employed to forecast future values based on the historical data of the time series. Statistical models such as ARIMA, SARIMAX, and BSTS are commonly used in this approach. Each model can handle only one time series and must be trained (fitted) each time the last observation point changes.

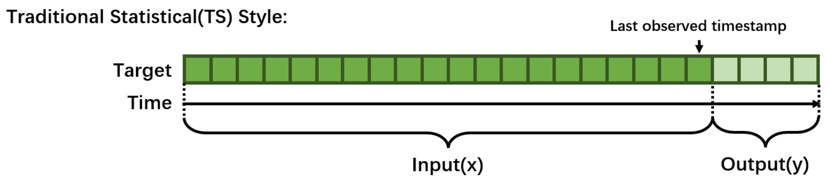


Fig. 4. Traditional Statistical Style[Aut20]

– – Regular Regression (RR) Style: In this style, forecasting is transformed into a supervised learning regression problem. A model can predict multiple time series. Deep learning models such as RNN, CNN, and Transformer are commonly used in this approach. Linear regression models can also be employed to forecast time series by treating time as an independent variable and using historical data to predict future values.

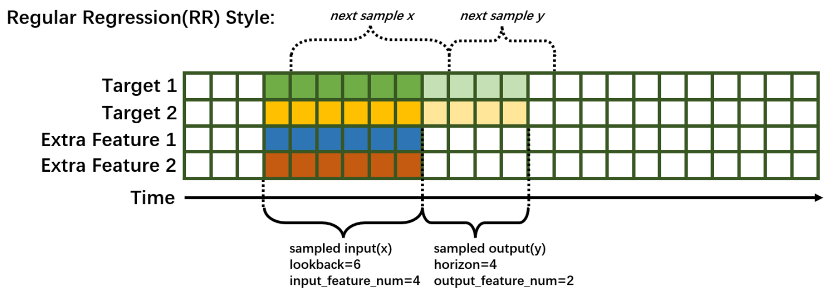


Fig. 5. Regular Regression Style[Aut20]

In this project, we will adopt the Regular Regression Style because choosing the Traditional Statistical (TS) Style for a real-time forecasting system would not be suitable. To elaborate further, each model in the TS Style would require retraining (fitting) whenever the last observation time or sample changes. This would be computationally expensive and time-consuming, especially if the data is frequently updated over short intervals. On the contrary, in the RR Style, there is no necessity to retrain the model every time new data is available. The team has also identified three models in the RR Style: LSTMForecaster, Seq2SeqForecaster, and TCNForecaster.

- **TCNForecaster** is a Time Convolutional Neural Network, or TCN, with a deep neural network (DNN) architecture specifically designed for time series data. This model utilizes historical data for a target variable, along with relevant features, to generate probability forecasts of the target for a specified forecasting horizon. The TCNForecaster consists of the following main components:

  - A pre-processing layer initially combines time series data and input feature data into an array of signal channels that the convolutional stack will process.
  - A stack of convolutional layers sequentially processes the array of channels; each layer in the stack processes the output of the previous layer to generate a new array of channels. Each channel in this output contains a mixture of convolutionally filtered signals from the input channels.
  - A set of output units aggregates the output signals from the convolutional layers and generates predictions of the target variable from this latent representation. Each output unit produces predictions up to a specified forecasting horizon for a fraction of the forecast distribution.

- **LSTMForecaster** is a Long Short-Term Memory (LSTM) recurrent neural network capable of learning and predicting long sequences. One advantage of LSTMs, in addition to learning long sequences, is their ability to generate multi-step forecasts at once, which can be beneficial for time series forecasting. The LSTM model can also be employed for forecasting multivariate time series. This involves scenarios with multiple observed sequences, and a model is required to learn from these observed sequences in the past to predict the next values for all sequences.

- **Seq2SeqForecaster** in the BigDL-Chronos library is a model based on a machine learning approach known as sequence-to-sequence learning, or Seq2Seq. This model is widely used in natural language processing and can be applied to various types of time series data. The Seq2SeqForecaster operates by transforming a sequence from one domain (e.g., an English sentence) into a sequence in another domain (e.g., the same sentence translated into French). This can be utilized for machine translation or for time series forecasting tasks.

- **RandomForestClassifier** is a machine learning algorithm in the Spark ML library, used for classification. It supports both binary and multi-class labels, as well as both continuous and categorical features. The RandomForestClassifier model works by creating a "forest" of decision trees, each tree is built from a random subset of features. When predicting, the model will take the results from all the trees in the forest and choose the class with the most votes. In this course project, the team will use RandomForestClassier to classify the type of rain (precipitation type) based on the data and the forecaster model just predicted. From there we can know whether it is raining or not, and if so, what kind of rain it is.

Model training and evaluation: We began training on each model. To get future predictions based on 12 hours prior, the team set the lookback to 12 and the horizon to 1, 3, 6 respectively for comparison, thereby getting a general view of the long-term and short-term prediction capabilities of each model. In addition, these models also support multi-variate, so the team set the target attributes to include: air temperature, humidity, interval rain, barometric pressure, and precipitation type. Among them, the first four attributes will be predicted by 3 forecaster models, while the precipitation type attribute will be classified by a RandomForestClassifier model in Spark's ML library. During the training process, the team relied on the sMAPE evaluation index to fine-tune the hyperparameters for the models to improve performance. The following are the results that the team obtained after training the models:

Forecaster	Features	Horizon	sMAPE
TCNForecaster	timestamp,rain_intensity,total_rain,wet_bulb_temperature	1	25.6
	timestamp,rain_intensity,total_rain		25.57
	timestamp,rain_intensity		25.6
	timestamp		25.7
	timestamp,rain_intensity,total_rain,wet_bulb_temperature	3	26.5
	timestamp,rain_intensity,total_rain		26.63
	timestamp,rain_intensity		26.6
	timestamp		26.5
	timestamp,rain_intensity,total_rain,wet_bulb_temperature	6	27.5
	timestamp,rain_intensity,total_rain		27.41
	timestamp,rain_intensity		27.38
	timestamp		27.56
LSTMForecaster	timestamp,rain_intensity,total_rain,wet_bulb_temperature	1	25.7
	timestamp,rain_intensity,total_rain		25.6
	timestamp,rain_intensity		25.7
	timestamp		25.64
Seq2SeqForecaster	timestamp,rain_intensity,total_rain,wet_bulb_temperature	1	25.67
	timestamp,rain_intensity,total_rain		25.66
	timestamp,rain_intensity		25.7
	timestamp		25.64
	timestamp,rain_intensity,total_rain,wet_bulb_temperature	3	26.58
	timestamp,rain_intensity,total_rain		26.62
	timestamp,rain_intensity		26.56
	timestamp		26.5
	timestamp,rain_intensity,total_rain,wet_bulb_temperature	6	27.57
	timestamp,rain_intensity,total_rain		27.44
	timestamp,rain_intensity		27.5
	timestamp		27.55

In the table above, we can see that the performance of the three models does not differ significantly from each other, even when they have been tested with different numbers of independent variables and horizons. All three models give predictions that are quite close to the actual values, and the prediction accuracy of the models is inversely proportional to the size of the horizon. Therefore, the team will choose the TCNForecaster with independent variables being timestamp, rain intensity, total rain, and horizon equal to 1, which has the lowest sMAPE (25.57), to build the prediction system.
This is an example of a prediction from the TCNForecaster model with the air temperature:

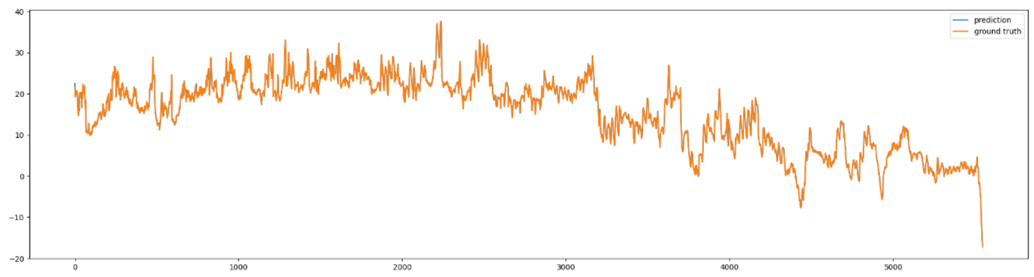


Fig. 6. Air Temperature prediction with TCNForecaster

Meanwhile, the RandomForestClassifier model also demonstrated good rain type classification capabilities, achieving an accuracy of up to 0.95 on the test set.

Building System Structure: After selecting the best model, the team will proceed to build the system structure. This includes building a producer, combining previously completed components such as data preprocessing, the trained model, and building a streaming query so that the system can make real-time predictions. In general, the system will operate as follows:

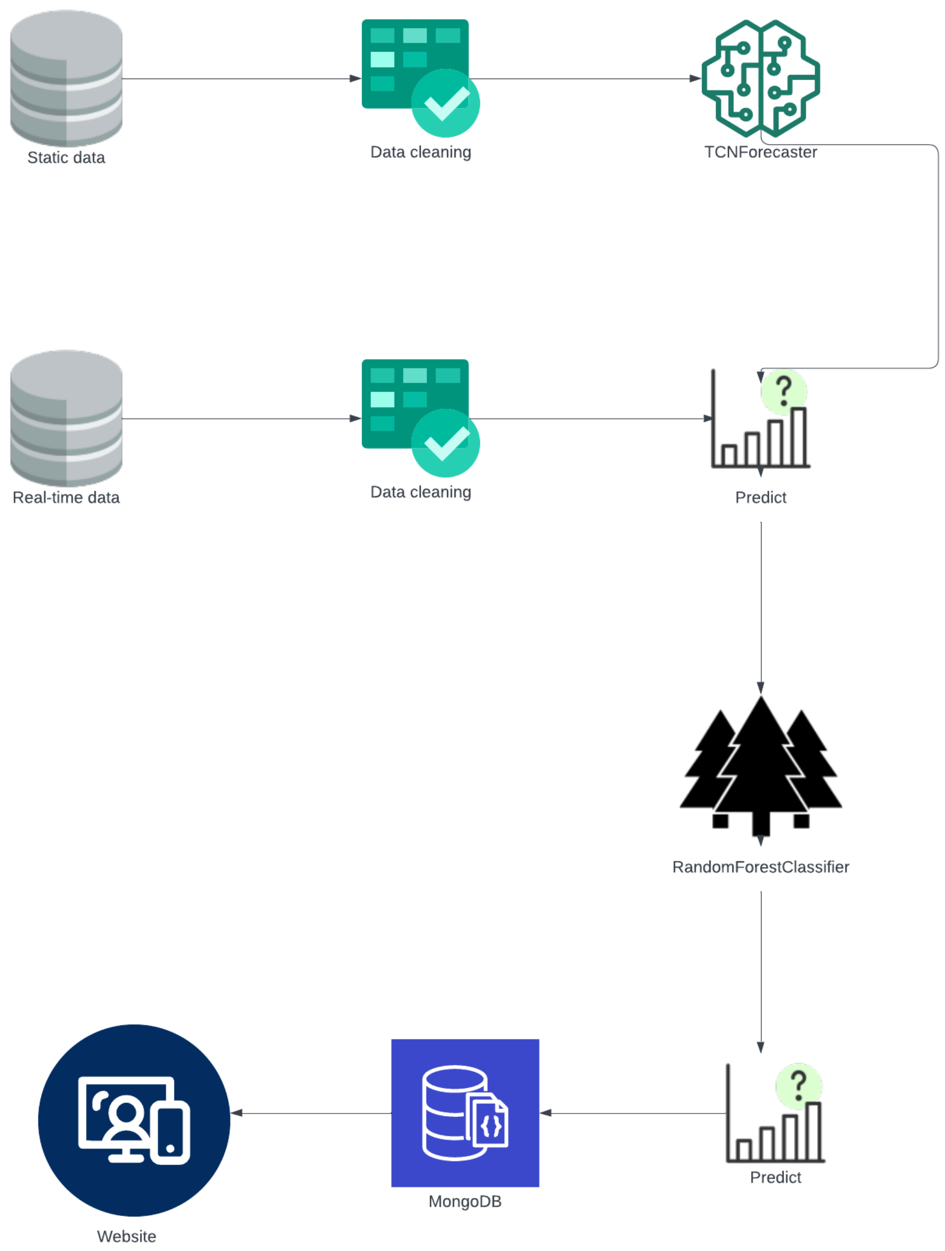


Fig. 7. System structure

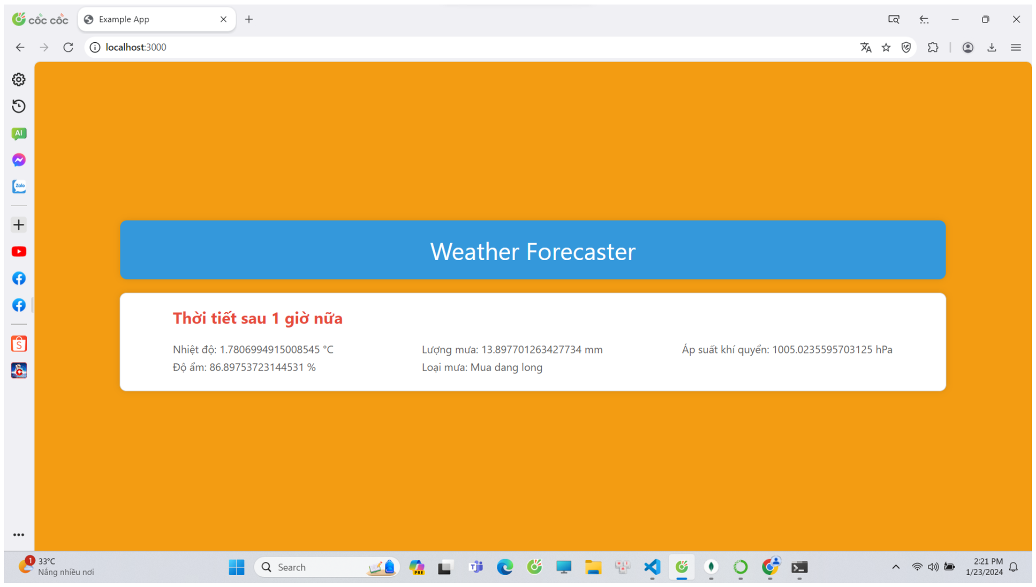
The specific process is as follows:

1. Collect static data and preprocess the static data.
2. Train the TCNForecaster model with the preprocessed static data.
3. Train the RandomForestClassifier model to classify the type of rain.

4. Build a Producer to send real-time data for prediction.
5. Preprocess the data sent from the producer.
6. Make predictions with the TCNForecaster.
7. Classify the type of rain with the RandomForestClassifier.
8. Save the predicted data into MongoDB using the MongoClient module.
9. Display the results on the built website.

6 Tests and results

After completing the system construction stages and saving the prediction results into MongoDB, our team will display the prediction results on a simple website that we has built.



Thời tiết sau 1 giờ nữa

Nhiệt độ: 1.7806994915008545 °C
Độ ẩm: 86.89753723144531 %
Lượng mưa: 13.897701263427734 mm
Loại mưa: Mua dang long
Áp suất khí quyển: 1005.0235595703125 hPa

Fig. 8. Prediction visualized on website

7 Conclusion and direction of development

We has successfully built a weather prediction system using Apache PySpark, Apache Kafka, and the BigDL-Chronos library. Additionally, throughout the implementation of this project, we also has gained valuable knowledge in machine learning, deep learning, and related fields.
In the future, we will continue to deploy, test, and research to enhance the performance of the prediction models and other components of the project. Furthermore, our team will explore deploying the project in parallel processing, distributed computing mode to improve resource utilization and computational speed.

References

Aut20. BigDL Authors. Bigdl-chronos, time series forecasting, 2020.

HTPB21. Pradeep Hewage, Marcello Trovati, Edmilson Pereira, and Ardhendu Behera. Deep learning-based effective fine-grained weather forecasting model. *Pattern Analysis and Applications*, 24:343–366, 2021.

Por. Chicago Data Portal. Beach weather stations - automated sensors.