Array 14 (2022) 100153

Contents lists available at ScienceDirect

Array

journal homepage: www.sciencedirect.com/journal/array

A real-time collaborative machine learning based weather forecasting system with multiple predictor locations

Tulsi Pawan Fowdur *, Rosun Mohammad Nassir-Ud-Diin Ibn Nazir

Department of Electrical and Electronic Engineering, University of Mauritius, Reduit, Mauritius

ARTICLE INFO

Keywords:
Weather forecasting
Machine learning
Collaborative
Real-time
Cloud
Mobile

ABSTRACT

Weather forecasting is an important application in meteorology and has been one of the most scientifically and technologically challenging problems around the world. As the drastic effects of climate change continue to unfold, localised short term weather prediction with high accuracy has become more important than ever. In this paper, a collaborative machine learning-based real-time weather forecasting system has been proposed whereby data from several locations are used to predict the weather for a specific location. In this work, five machine learning algorithms have been used and tests have been performed in four different locations in Mauritius to predict weather parameters such as Temperature, Wind Speed, Wind Direction, Pressure, Humidity, and Cloudiness. The weather data were collected using the OpenWeather API from a mobile as well as a desktop edge device. The data were stored as a JSON file in both the IBM Cloudant database and a local MySQL database. Analytics were performed on both a local server that captures the incoming data from the edge device and via a servlet deployed on the IBM cloud platform. Five machine learning algorithms namely Multiple Linear Regression (MLP), Multiple Polynomial Regression (MPR), K-Nearest Neighbours (KNN), Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN) were tested using both collaborative and non-collaborative methods. The experiments showed that the collaborative regression schemes achieved 5% lower Mean Absolute Percentage Error (MAPE) than non-collaborative ones and the Multiple Polynomial Regression (MLR) algorithm outperformed all the other algorithms with errors ranging from 0.009% to 9% for the different weather parameters. In general, the results showed that collaborative based weather forecasting with multiple predictor locations can potentially increase the accuracy of the predictions in machine learning algorithms.

Credit author statement

The authors confirm that the work presented is their own work and all information taken from other sources have been duly referenced.

1. Introduction

Weather forecasting aims to determine the state of the atmosphere over a given forecast period at a specific location. The weather can be described as variations in atmospheric variables such as temperature, wind speed and direction, humidity, sun-shine, cloudiness and precipitation [1]. With the advent of the Internet of Things (IoT), it is possible to obtain real-time weather predictions. Multiple weather sensors can be deployed at a low cost to collect weather data in different locations. Machine learning algorithms can then be applied to the collected data to perform predictions with a high level of accuracy [2]. In addition to IoT, the Infrastructure as a Service (IaaS) model of commercial cloud platforms is well suited for applications such as weather forecasting that require computational resources sporadically. Renting high-end servers to execute applications that require bursty computational demands has the potential to be more cost-effective than purchasing and maintaining dedicated hardware. This is because cloud users pay only for the time the resources have been used [3]. Recently, several research have focused on the application of IoT, machine learning and cloud computing to develop real-time weather forecasting systems. An overview of some of these schemes is given next.

In [4], David Irwin et al. proposed CloudCast, a mobile application that provides personalized short-term weather forecasting. The aims of this work were to investigate the hypothesis that current cloud platform connectivity and diversity could mitigate the impact of data staging and computational latency for Nowcasting. The architecture consisted of linking the radar sensor network of the Collaborative Adaptive Sensing

* Corresponding author.
E-mail addresses: p.fowdur@uom.ac.mu (T.P. Fowdur), mohammad.rosun2@umail.uom.ac.mu (R.M. Nassir-Ud-Diin Ibn Nazir).

https://doi.org/10.1016/j.array.2022.100153
Received 27 October 2021; Received in revised form 19 February 2022; Accepted 1 April 2022
Available online 14 April 2022

of the Atmosphere (CASA) centre with the Meteorological Command and Control (MC & C) to regulate the radar scanning. DiCloud was used to control access to the Amazon's Elastic Compute Cloud (EC2) resources to track the cost associated with the individual Nowcast requests. A nowcasting algorithm was used to predict high impact weather events for short time intervals. The authors observed high accuracy for 10 min Nowcasts. Further testing showed a 2 min delay for the delivery of a 15 min Now-casts image to a mobile client. The study revealed that commercial and research cloud services can be used for real-time nowcasting applications.

In [5], Bobby D. Gerardo et al. presented a Flash Flood Warning System using SMS with advanced warning information. The system architecture consisted of a remote monitoring device made of an ultrasonic sensor, pulse detector, a wireless modem, a FEZ Domino board and a 6-V battery charged by a solar panel. The water level and speed were measured and sent to a server. The server application and the modem which was used to issue SMS messages to subscribers were both simulated. To test the framework, water levels and velocity were recorded for seven days. The collected data was used to train an MLR model to predict future water flow. Using the predicted values, a risk value was calculated which was then used to determine a threshold to issue warnings to users.

An experimental analysis of a cloud-based real-time weather forecasting system was made by S. G. Totad et al. to investigate how the Cloud and IoT can be used for weather data collection and prediction [6]. The experiment architecture consisted of two DHT11 sensors to measure temperature and humidity and an Arduino board connected to a Raspberry pi. Data from the sensor were sent to the Arduino board, the Raspberry pi fetched the data and stored them in a public cloud through ThingSpeak. To perform analysis and prediction based on the collected data, one year of prior data was fetched from the cloud and correlation analysis of the parameters was then performed. An ARIMA model was generated for predicting the values for the upcoming year. The results showed satisfactory prediction results for the first two successive years. Beyond that, the prediction accuracy started to drop.

T.P. Fowdur et al. [7] made use of an IoT based weather monitoring system to develop a cloud-based real-time weather forecasting system. The aim was to investigate the accuracy of different data mining techniques and developed three adaptive schemes to select the appropriate algorithms when performing prediction for a given Nowcast. The architecture consisted of a weather monitoring node, a Raspberry pi 3 and a database server. The monitoring system was made up of weather sensors that measure weather conditions, a weather shield that was connected to the sensors and an Arduino board. An XBee shield with a WiFi module was stacked onto the Arduino board allowing the monitoring module to be connected to the Raspberry pi 3 wirelessly. Recorded data were sent from the monitoring module to the Raspberry pi for processing and storage. Several data mining algorithms including MLR, KNN, ARIMA were used for forecasting the different weather variables. To test the system, data for the previous 11 days were used, a stepwise regression using the aforementioned algorithms was performed using a window of 2 h to predict for the next intervals of 20, 40 and 60 min. The results showed a general increase in error as the prediction interval increases. The error using the adaptive algorithms were lower compared to non-adaptive ones. The lowest error achieved was 0.454% for luminosity and the highest error was 45.56% for precipitation.

In [8], the authors proposed using a six-layer Convolutional Neural Network (CNN) for forecasting severe convective weather (SCW) conditions. The study aim was to compare the CNN model to other machine learning algorithms including Support Vector Machine (SVM), Linear Regression (LR) and Random Forest Classifiers. To train the models, five years of severe weather observations were used from the National Centers for Environmental Prediction (NCEP) final analysis data. A large sample of data for each severe weather type was selected for training. The CNN model used consisted of six layers using a softmax classifier. After training the model, a sample of 50 unlabeled NCEP data was fed to the network to calculate the probability of an SCW occurring. Four skills scores were used to determine the performance of the network forecasts. When the skills scores were compared to other data mining algorithms, the CNN model showed an increase of 33.2% in SCW detection. The CNN model also scored higher in the skill scores compared to the other algorithms.

In [9] a weather forecasting system which employed simple machine learning models such as Random Forest Regression (RFR), Ridge Regression (Ridge), Support Vector (SVR), Multi-layer Perceptron (MLPR), and Extra-Tree Regression (ETR), were used to predict the temperature of the next day at any particular hour of the city of Nashville, Tennessee, based on the weather data of the current day of this city and a couple of its surrounding cities. It was found that using different predictor locations to predict the temperature of a target location, lead to better prediction accuracy as compared to using data from a single region.

Other interesting machine learning-based weather forecasting systems have been proposed in Refs. [10–15], and [16].

Although, the previous works on weather forecasting are very promising, there are several gaps that can still be exploited to enhance them. For example the analysis of the work in Ref. [4] considers only the cloud computing parameters such as throughput and memory but not the performance of the prediction algorithms in terms of mean squared error. The work in Ref. [5] was limited to the use of SMS for alerting users and did not provide a complete weather report. Only ARIMA was used in Ref. [6] to perform predictions and the work did not consider more advanced machine learning models. The work in Refs. [7,8] considered several algorithms but collaborative predictions based on data from several locations were not considered. Finally, despite the fact that collaborative forecasting was considered in Ref. [9], the architecture of the system and formulation of the collaborative algorithms used were not explicitly described and only temperature predictions were considered.

Motivated by previous research on cloud-based real-time weather forecasting using machine learning, this work develops a collaborative machine learning technique for weather forecasting. Essentially, weather conditions from different regions are combined to predict the weather parameters for a given region but in contrast to Ref. [9] where only temperature was considered, in this work six different weather parameters are predicted. Moreover, in addition to using data from different regions as in Ref. [9], a complete system with a both local and cloud-based servers as well as web and mobile client applications have been employed by extending the system in Ref. [17]. Detailed mathematical formulations of the collaborative models have also been provided for the five different machine learning algorithms that have been used. It was observed that collaborative forecasting particularly enhances the performance of the Multiple Polynomial Regression (MPR) and Multiple Linear Regression (MLR) algorithms. Moreover, a system that incorporates several machine learning algorithms to perform analytics in real-time on the cloud, has been developed. The results of the analytics are then sent to the user from the cloud. A mobile, desktop and web application have also been developed for users to observe the current weather conditions, download weather data and perform analytics.

The main contribution of this work is the development of a ubiquitous and highly flexible weather forecasting system that can perform collaborative weather forecasting both on a local server and the cloud while at the same time providing several user interfaces on desktop, mobile and web platforms for seamlessly accessing real-time weather forecasts. The novelty of this work as compared to other works are as follows:

(i) In contrast with most previous works [4–8] in which weather forecasting for a given target region is performed only with data for that region, this work considers a model which uses data from

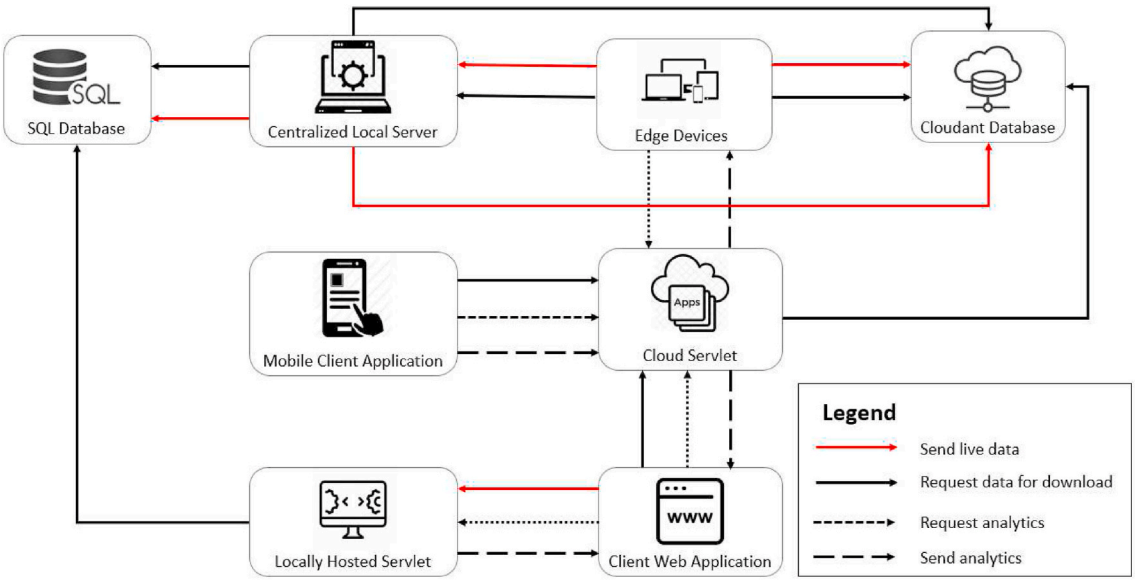


Fig. 1. System data flow.

several predictor regions to predict the data for a target region to achieve better prediction accuracy.

(ii) To the best of our knowledge a complete system with local and cloud servers, interactive mobile, desktop and web applications to access weather data forecasts seamlessly with collaborative weather forecasting, has not been proposed in any other work.

(iii) Compared to Ref. [9], this work provides a performance analysis on six different weather parameters for collaborative forecasting with detailed treatment on the algorithms and architecture developed.

This work therefore provides an important architectural and mathematical framework for state of the real-time weather forecasting systems. Moreover, it investigates the aspect of collaborative weather forecasting which has a lot of potential to significantly improve forecasts in correlated regions. The proposed architectures and multi-platform applications also provide ease of deployment and scalability.

The rest of the paper is organized as follows. Section 2 describes the system model and the proposed collaborative machine learning algorithms. Section 3 gives the results obtained with a detailed analysis. Section 4 concludes the paper.

Fig. 2. Centralized local java server app.

2. System model and algorithms for collaborative weather forecasting

In this section, the complete system model for weather forecasting consisting of the local server, mobile and desktop edge devices, cloud-hosted servlet and web application is first described. After that, the modifications that have to be made to the conventional machine learning algorithms, in order to perform collaborative forecasting are described.

2.1. System architecture for weather data collection and analytics

Fig. 1 shows a diagram detailing the data flow in the entire system. The system extends the one described in Ref. [17] by adding a mobile client with an interactive interface specifically designed for weather forecasting in Mauritius. Moreover, the centralized local server in Ref. [17] is replaced by a local server that can perform collaborative weather forecasting using weather parameters for different regions to predict the weather for a given region. Same applies to the cloud servlet and locally hosted servlet.

The system consists of edge devices (mobile and desktop) that will send a request to the OpenWeather API [18] to request real-time weather parameters for a given region. These parameters are sent to the Local Server or directly to the cloud. The centralized local server can either store the values on a local MySQL database or send them to an IBM cloudant database. Analytics can be performed on the local server, the cloud-hosted servlet or the locally hosted servlet. A Web application has also been developed to allow users to monitor weather parameters and request predictions. Moreover, a dedicated android client for weather forecasting in Mauritius has been developed which shows an interactive map of the country with weather forecasts obtained from the cloud-hosted servlet. More details on each component of this system are given in the following subsections.

2.1.1. Centralised local server for collaborative weather forecasting

To implement the centralised local server, an application written in Java was created. Fig. 2 shows a screen capture of the centralized local server layout. The desktop application has the following functionalities:

- Monitors incoming socket connections from edge devices (android & desktop), captures the incoming data and stores the weather data in the local SQL database and cloud NoSQL database.
- Displays a real-time graph showing the variation of the different weather variables.
- Downloads data from either the local or cloud database for a specific date.
- Downloads the latest data from either the local or cloud database for a specific sample size.
- Performs prediction for a specific location using the five machine learning algorithms.
- Performs a time-series cross-validation for the different algorithms.
- Displays the MAPE of the cross-validation in a table and graph for easy comparison.

To predict the weather for a given time interval, the user clicks on the "Predict" button to perform prediction using the five possible machine learning algorithms. To download data, the user chooses a date and a sample size and then clicks on the "Download" button. The required data will then be fetched from the appropriate source and saved to the local storare in CSV format.

To perform collaborative regression, the user must enter the response location and choose at least two predictor locations. The training parameters are then set using the provided textboxes. Then, the "Collaborative" button is clicked which invokes the algorithms for collaborative regression described in Section 2.2. The MAPE of the prediction for each weather variable is then shown in the output table located at the bottom of Fig. 2 table and plotted in a bar chart shown in the center of Fig. 2.

```
api.openweathermap.org/data/2.5/weather? //endpoint
q={city name}     //city name here
&appid={API key}  //API Key here
&mode={mode}      //response type here - json/xml
&units={units}    //measurement units - imperial/metric
```

Fig. 3. OpenWeather query URL.

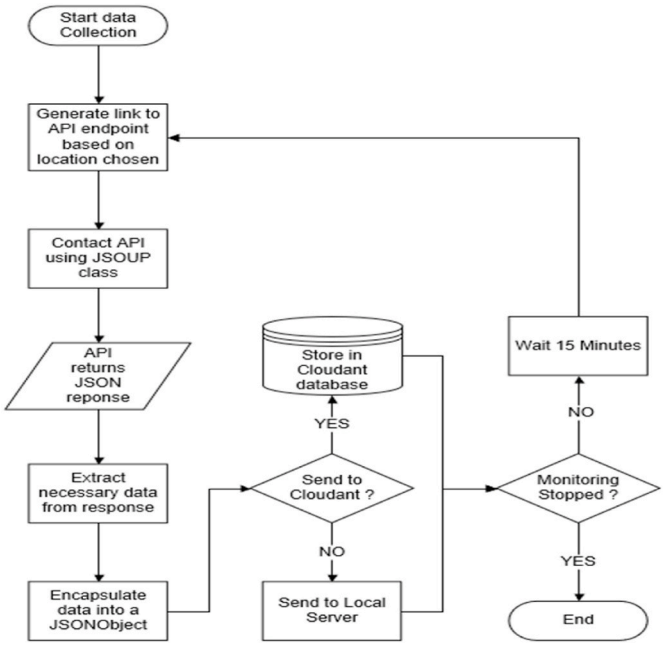


Fig. 4. Data collection flowchart.

The monitoring process is started by clicking the "Monitor" button. It starts a ServerSocket connection on port 6666 and monitors incoming socket connections from edge devices.

An analytics backend which is comprised of the machine learning algorithms and functionalities for fetching and processing the data, has been implemented for the local server and web applications. The backend is responsible for the following:

- Fetching data either from the local database or Cloudant database.
- Pre-processing the raw weather data.
- Training the different machine learning models.
- Predicting the expected weather for the next n minutes.
- Cross-validation of each algorithm using a sliding window regression.

2.1.2. Edge devices for weather data capture

This block represents the mobile and desktop applications responsible for collecting data from the OpenWeather API and sending it to either the Cloudant database or the local server. The mobile and desktop applications can also request data for download from either the local server or directly from the Cloudant database. The edge performs the following main functions [17]:

- Collects data from the OpenWeather API using the REST paradigm.
- Sends the data either to the local server application for local storage or stores it directly into the appropriate cloudant database using the Cloudant API.
- Displays a graph that shows the real-time variation of the weather condition for a specific location.
- Requests the Java servlet application for prediction results.

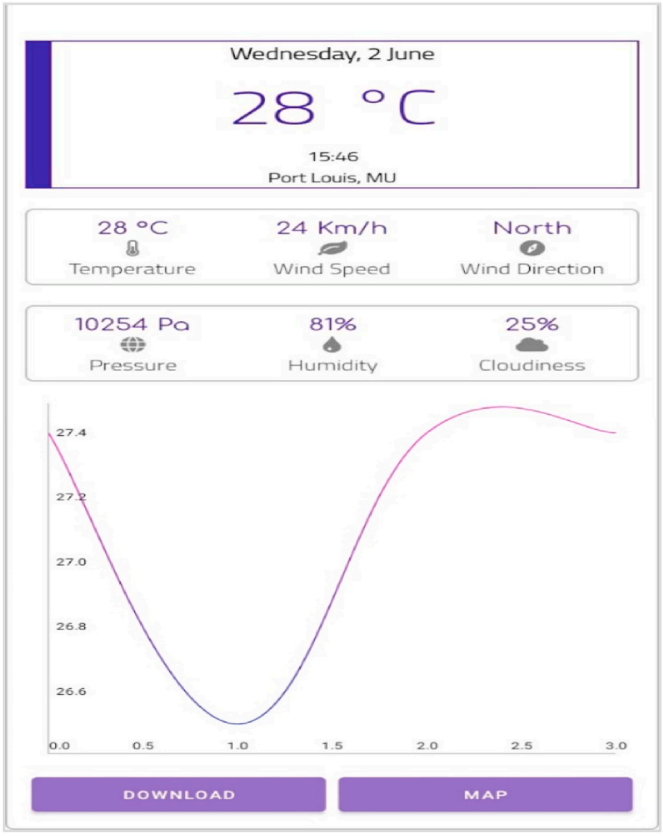


Fig. 5. Mobile client Main activity screen.

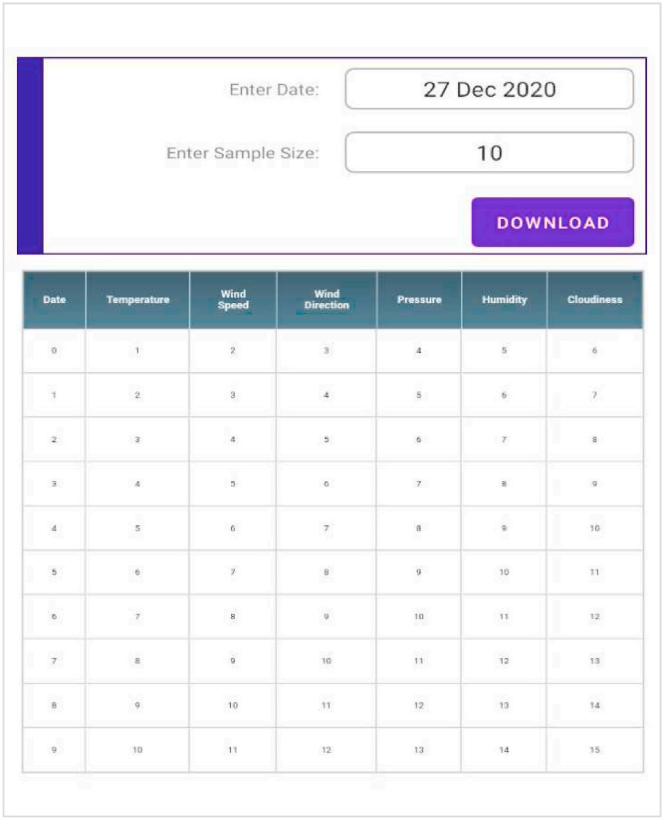


Fig. 6. Download data screen.

- Requests the Java servlet application for weather data for a specific date or fixed sample size.

The OpenWeather API [18] was used to collect data about current weather conditions every 15 min and store it in a database. To obtain the current weather condition for a specific location, the URL for the API call has to be properly formulated by specifying the city name, response type, units of measurement & API key as shown in Fig. 3.

To connect to the API, the JSOUP class was used. A connection to the URL was opened by using the connect() method of the static JSOUP class and then the get() method fetched and parsed the HTML data into a Document object. The Document object was converted to a String using the text() method. The response String was then type-casted to a JSONObject. The necessary reading was extracted from the response and encapsulated in a JSONObject. The JSONObject was then sent either to the Local Server app or Cloudant platform for storage.

A detailed description of all source codes for the mobile and desktop edge devices is given in Ref. [17]. Fig. 4 shows the data collection process that has been implemented.

Every 15 min a new link following the format in Fig. 3 is generated based on the chosen location. The OpenWeather API is contacted using the JSOUP library which contacts the link generated and downloads the JSON response from the server. The necessary data from the JSON String is extracted and encapsulated in another JSON object. A check is performed whether the "Cloud" checkbox is ticked. If this is the case, the data is sent directly to the Cloudant database, else it is sent to the local server for storage in the SQL database.

2.1.3. *Mobile client application*

A mobile application was developed using the Android Studio IDE. The application has the following functionalities:

- Displays the latest weather condition recorded.
- Displays a live graph that shows the variation of the different weather variables.
- Allows the users to download weather information for a specific date or the last N weather observations.
- Displays a GoogleMap fragment from which the user can choose a location to obtain the expected weather conditions for the next N minutes.

When the mobile client is launched, the screen in Fig. 5 is displayed. The latest recorded weather conditions are shown on the screen. There is also a graph that displays the variation of a specific weather condition in real-time. To change the variable being display, the user just has to click on the desired variable from the 2nd and 3rd rows.

To download weather data, the "Download" button is clicked. This launches the screen which is shown in Fig. 6. The user can enter either the date for which data is required or the number of samples that are required. The "Download" button is then clicked to download the data. the downloaded data is also shown in a table.

To get the expected weather condition for another location, from the Main Screen of Fig. 5, the "MAP" button is clicked. This launches the "MapActivity" screen. The screen shows a map fragment with several markers in different locations. The user then clicks on the location for which prediction is needed, the screen in Fig. 7 is then shown.

If the user wants to perform prediction using the centralized local server, the IP address of the server is entered and a prediction time is chosen. Then, the "Predict" button is clicked. The prediction is then displayed to the user. To perform predictions using the cloud java servlet, the desired location is chosen and a prediction time is chosen. Then, the "Predict" button is clicked but no IP address is entered in this case as the prediction will be obtained by querying a cloud server. The prediction is then displayed to the user.

The results returned by the servlet will be displayed in the respective field on the screen shown at the bottom of Fig. 7 which shows the

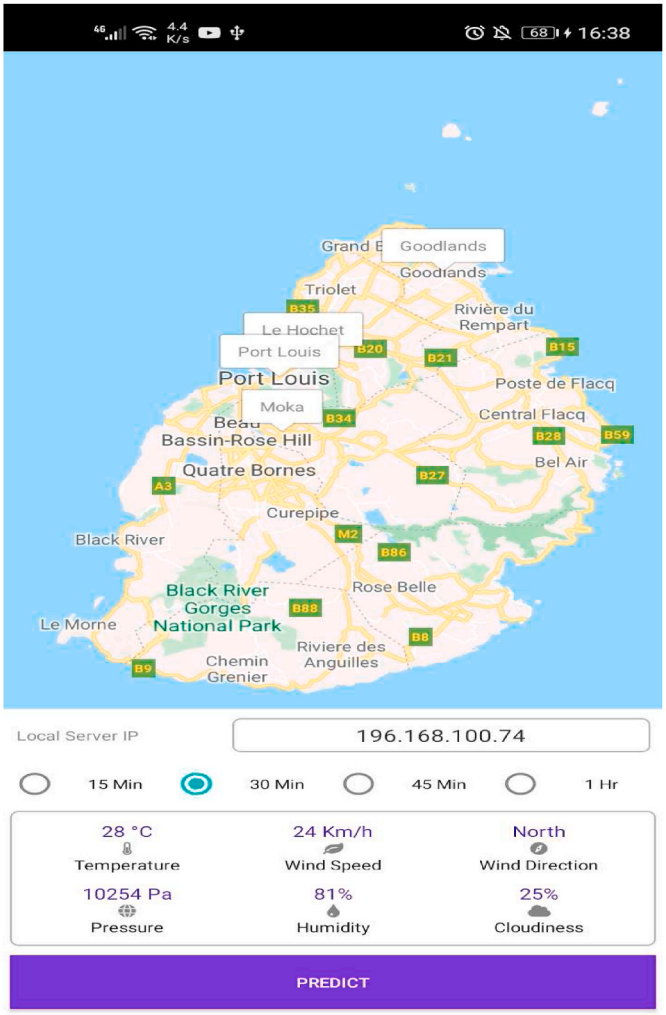


Fig. 7. MapActivity screen.

predicted values for the Temperature, Wind Speed, Wind Direction, Pressure, Humidity and Cloudiness after 30 min in this case.

2.1.4. Web application

The web application is a webpage that is displayed to users when they visit the servlet link. The webpage provides a graph that displays the variation of the weather parameters in real-time. The user can also download weather data for a specific date or only the last N readings. The page is also responsible for contacting either the cloud-hosted or the locally hosted servlet for analytics results and displaying it to the user.

A dynamic web application was written in the Eclipse IDE and deployed on the Bluemix server on IBM Cloud. The web app presents the users with a webpage where they can perform the following actions:

- Perform prediction on the cloud and get the results on their device.
- Download weather data for a specific date or only the last N weather observations.
- Observe the variation of the weather parameters in a graph updated every 15 min. The update interval is chosen as 15 min because the values from the OpenWeather API are refreshed in the same time interval.

2.1.5. Cloud hosted servlet

The main function of the cloud-hosted servlet is to serve requests coming from end-users that visit the webpage. The servlet is responsible for the following functions:

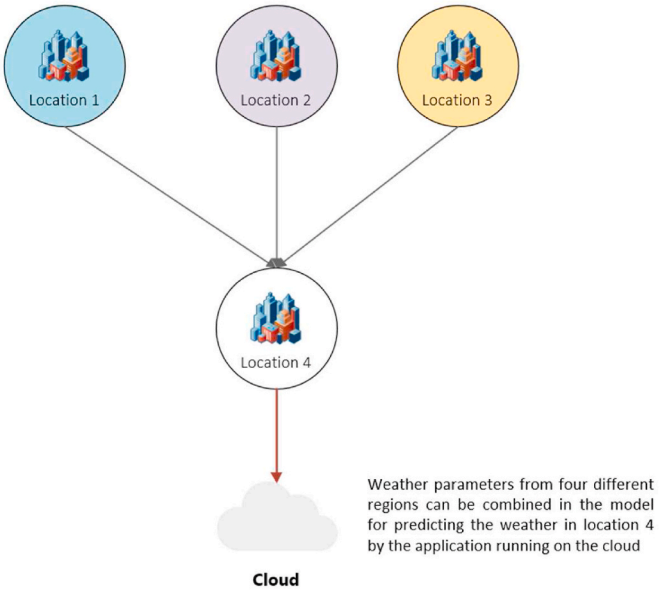


Fig. 8. Collaborative machine learning.

- Updates the live graph on the webpage presented to the user.
- Serves requested weather data to the user by fetching data from the Cloudant database.
- Performs analytics and returns the result to the user.

2.1.6. Locally hosted servlet

The locally hosted servlet performs the same functions as the cloud-hosted servlet but instead of being hosted on the IBM Cloud platform, it is hosted on a local Apache Tomcat server. The user must specify the IP Address of the server to be able to connect to the servlet.

2.1.7. MySQL database

To store the incoming data from the mobile and desktop applications locally, the MariaDB database in XAMMP is used. Data from the local database is used to perform prediction locally on the local server without the need to contact the cloud-hosted servlet.

2.1.8. Cloudant database

The Cloudant database is a service offered by the IBM Cloud platform. It is used in the system to store weather data in the cloud so that the data is accessible globally. To store the collected data in the Cloudant database, a new Cloudant resource must be added to the IBM Cloud Resource lists. This sets up a Cloudant instance that allows users to create as many databases as needed. A partitioned database in Cloudant allows logical partitioning of data by providing a second key called the partition key. The possibility to use a partitioned database was investigated, however, the queries were capped at 2000 results per query with a timeout of 5 s. For real-time applications, the result size cap resulted in multiple queries being made which increased the processing time. To address this issue, eight databases were created to store data from each location into a specific database. A detailed description of how to configure this database is given in Ref. [17].

2.2. Collaborative machine learning algorithms for weather forecasting

Collaborative machine learning in this paper is defined as the process of using data from multiple locations to predict the expected weather condition for a specific region. Fig. 8 shows the architecture for the collaborative machine learning model.

While collaborative machine learning enhances the prediction accuracy, this enhancement comes at the cost of higher complexity as will

Table 1
Weather variables notations.

Parameter	Symbol
Temperature	T_t
Wind Speed	WS_t
Wind Direction	WD_t
Pressure	P_t
Humidity	H_t
Cloudiness	C_t

be seen in the modified mathematical models of the different machine learning techniques used for collaborative prediction in the following subsections.

2.2.1. Collaborative multiple linear regression (MLR)

In this paper, with multiple linear regression, equations involving a general variable V1 to be predicted at time t + 1 with respect to the previously recorded value of V1 at time t and other related parameters Vx(1), Vx(2), …, Vx(n) were formulated as per the following generic combination:

$$\mathrm{V1}_{t+1} = \beta_0(\mathrm{w}) + \beta_{(1)}(\mathrm{w})\ \ \mathrm{Vx}(1) + \beta_{(2)}(\mathrm{w})\ \ \mathrm{Vx}(2) + \ldots + \beta_{(\mathrm{n})}(\mathrm{w})\ \ \mathrm{Vx}(\mathrm{n}) \tag{1}$$

where.

β_0(w), β_1(w), …, β_n(w), are coefficients determined for a training window of size w.

A combination of six multiple linear regression equations was derived experimentally for the six weather parameters. To define the equations, the notations in Table 1 are used to denote each weather parameter.

The MLR conventional equations i.e. without collaborative data from different locations, can be expressed as follows:

$$T_{t+1} = \beta_0(\mathrm{w}) + \beta_{(1)}(w)\ T_t + \beta_{(2)}(w)\ H_t + \varepsilon \tag{2}$$

$$WS_{t+1} = \beta_0(\mathrm{w}) + \beta_{(1)}(w)\ WS_t + \beta_{(2)}(w)\ T_t + \varepsilon \tag{3}$$

$$WD_{t+1} = \beta_0(\mathrm{w}) + \beta_{(1)}(w)\ WD_t + \beta_{(2)}(w)\ T_t + \varepsilon \tag{4}$$

$$P_{t+1} = \beta_0(\mathrm{w}) + \beta_{(1)}(w)\ P_t + \beta_{(2)}(w)\ T_t + \varepsilon \tag{5}$$

$$H_{t+1} = \beta_0(\mathrm{w}) + \beta_{(1)}(w)\ H_t + \beta_{(2)}(w)\ T_t + \varepsilon \tag{6}$$

$$C_{t+1} = \beta_0(\mathrm{w}) + \beta_{(1)}(w)\ C_t + \beta_{(2)}(w)\ H_t + \varepsilon \tag{7}$$

The MLR collaborative equations are expressed as summations as follows:

$$T_{t+1} = \beta_0(\mathrm{w}) + \sum_{i=1}^{n} \beta_{((i*2)-1\ )}(w)\ T_{t(i)} + \beta_{(i*2)}(w)\ H_{t(i\ )} \tag{8}$$

$$WS_{t+1} = \beta_0(\mathrm{w}) + \sum_{i=1}^{n} \beta_{((i*2)-1\ )}(w)\ WS_{t(i)} + \beta_{(i*2)}(w)\ T_{t(i\ )} \tag{9}$$

$$WD_{t+1} = \beta_0(\mathrm{w}) + \sum_{i=1}^{n} \beta_{((i*2)-1\ )}(w)\ WD_{t(i)} + \beta_{(i*2)}(w)\ T_{t(i\ )} \tag{10}$$

$$P_{t+1} = \beta_0(\mathrm{w}) + \sum_{i=1}^{n} \beta_{((i*2)-1\ )}(w)\ P_{t(i)} + \beta_{(i*2)}(w)\ T_{t(i\ )} \tag{11}$$

$$H_{t+1} = \beta_0(\mathrm{w}) + \sum_{i=1}^{n} \beta_{((i*2)-1\ )}(w)\ H_{t(i)} + \beta_{(i*2)}(w)\ T_{t(i\ )} \tag{12}$$

$$C_{t+1} = \beta_0(\mathrm{w}) + \sum_{i=1}^{n} \beta_{((i*2)-1\ )}(w)\ C_{t(i)} + \beta_{(i*2)}(w)\ H_{t(i\ )} \tag{13}$$

Table 2
Regression response and predictor relationship.

Response Variable	Predictor Variables	
Temperature	Temperature	Humidity
Wind Speed	Wind Speed	Temperature
Wind Direction	Wind Direction	Temperature
Pressure	Pressure	Temperature
Humidity	Humidity	Temperature
Cloudiness	Cloudiness	Humidity

where.

- n represents the number of predictor locations.

It is observed that the complexity of collaborative MLR represented by equations (8)–(13) is greater than that of conventional MLR and increases as the number of predictor locations i.e. n increases.

2.2.2. Collaborative multiple polynomial regression (MPR)

In this paper, with multiple polynomial regression, equations involving a general variable y to be predicted at time $t + 1$ with respect to the previously recorded value of y at time t and other related parameters $x1(t)$, $x2(t)$, $x3(t)$, …, $xn(t)$ were formulated as per equation (14) [19].

$$y(t+1) = \beta_0 + \sum_{p1=1}^{n} \beta_{P1} \mathrm{X}_{P1(t)} + \sum_{P1=1}^{n} \sum_{p2=p1}^{n} \beta_{p1p2} \mathrm{X}_{P1(t)} \mathrm{X}_{P2(t)} + \varepsilon \tag{14}$$

where.

- represents the dependent variable at time $t + 1$.
- $\mathrm{X}_{P2(t)}$ denotes the values of the independent variable at time t.
- $\beta_{P1}\beta_{P2}$ denotes the polynomial coefficients.
- - model error i.e., variation in the estimate of $Y(t + 1)$
- – number of predictor locations

The equation for a second-order MPR model can be represented as follows, let Y be presented by $x1(t)$ and other predictor variables be represented by $x2(t)$, $x3(t)$, …, $xn(t)$, the 2nd order MPR model is as follows:

$$\begin{aligned} y_{t+1} = {} & \beta_0 + \beta_1 \mathrm{x1(t)} + \beta_2 \mathrm{x2(t)} + \ldots + \beta_{\mathrm{n}} \mathrm{xn(t)} + \beta_{11} \mathrm{x1}^2\mathrm{(t)} + \beta_{12} \mathrm{x1(t)x2(t)} \\ & + \ldots + \beta_{1n} \mathrm{x1(t)xn(t)} + \beta_{22} \mathrm{x2}^2\mathrm{(t)} + \beta_{23} \mathrm{x2(t)x3(t)} + \ldots + \beta_{2n} \mathrm{x2(t)xn(t)} \\ & + \ldots + \beta_{nn} \mathrm{xn}^2\mathrm{(t)} \end{aligned} \tag{15}$$

To generate a model for a weather parameter, the variables $x1(t)$, $x2(t)$, $x3(t)$, …, $xn(t)$ can be substituted by the response and predictor variables for each weather parameter as given in Table 2.

The conventional equations of MPR for a second order model without collaborative data, i.e. from a single location, can be expressed as follows:

$$\mathrm{T}_{t+1} = \beta_0 + \beta_1 \mathrm{T}_1\mathrm{(t)} + \beta_2 \mathrm{H}_1\mathrm{(t)} + \beta_{11} \mathrm{T}_1{}^2\mathrm{(t)} + \beta_{12} \mathrm{T}_1\mathrm{(t)}\ \ \mathrm{H}_1\mathrm{(t)} + \beta_{22} \mathrm{H}_1{}^2\mathrm{(t)} \tag{16}$$

$$\mathrm{WS}_{t+1} = \beta_0 + \beta_1 \mathrm{WS}_1\mathrm{(t)} + \beta_2 \mathrm{T}_1\mathrm{(t)} + \beta_{11} \mathrm{WS}_1{}^2\mathrm{(t)} + \beta_{12} \mathrm{WS}_1\mathrm{(t)}\ \ \mathrm{T}_1\mathrm{(t)} + \beta_{22} \mathrm{T}_1{}^2\mathrm{(t)} \tag{17}$$

$$\mathrm{WD}_{t+1} = \beta_0 + \beta_1 \mathrm{WD}_1\mathrm{(t)} + \beta_2 \mathrm{T}_1\mathrm{(t)} + \beta_{11} \mathrm{WD}_1{}^2\mathrm{(t)} + \beta_{12} \mathrm{WD}_1\mathrm{(t)}\ \ \mathrm{T}_1\mathrm{(t)} + \beta_{22} \mathrm{T}_1{}^2\mathrm{(t)} \tag{18}$$

$$\mathrm{P}_{t+1} = \beta_0 + \beta_1 \mathrm{P}_1\mathrm{(t)} + \beta_2 \mathrm{T}_1\mathrm{(t)} + \beta_{11} \mathrm{P}_1{}^2\mathrm{(t)} + \beta_{12} \mathrm{P}_1\mathrm{(t)}\ \ \mathrm{T}_1\mathrm{(t)} + \beta_{22} \mathrm{T}_1{}^2\mathrm{(t)} \tag{19}$$

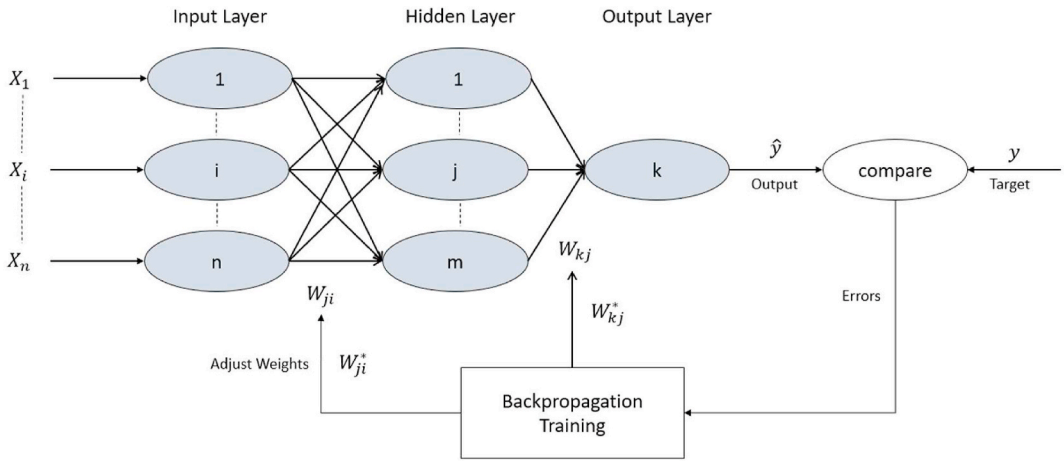


Fig. 9. Mlp network [23].

$$H_{t+1} = \beta_0 + \beta_1 H_1(t) + \beta_2 T_1(t) + \beta_{11} {H_1}^2(t) + \beta_{12} H_1(t)\ \ T_1(t) + \beta_{22} {T_1}^2(t) \tag{20}$$

$$C_{t+1} = \beta_0 + \beta_1 C_1(t) + \beta_2 H_1(t) + \beta_{11} {C_1}^2(t) + \beta_{12} C_1(t)\ \ H_1(t) + \beta_{22} {H_1}^2(t) \tag{21}$$

The conventional MPR equations for a second order model with collaborative data, i.e. from different locations, can be expressed as the summation in equation (14), where the independent variables X1 and X2 are replaced by the predictor locations as decribed in Table. Again the collaborative MPR model given in equation (14) will lead to an increase in complexity as compared to MPR that uses a single predictor location.

2.2.3. KNN collaborative regression

The K-Nearest Neighbours (KNN) algorithm is one of the simplest machine learning algorithms. It is based on a lazy learning method where the first K samples nearest to each sample test data are selected [20]. In this paper, to measure the distance between the test instance and the dataset the Euclidean distance function was used. For regression, the outcome was taken as the mean of the K nearest observations.

The formula to calculate Euclidean distance is:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{22}$$

For each dimension, we subtract one point's value from the other's to get the length of that side of the triangle in that dimension, square it, and add it to our running total. The square root of that running total is our Euclidean distance.

The Euclidean distance between the test data and the data in the training window denoted, $dX(test\ data,\ window)$, for each parameter can be summurised as follows:

$$dT(test\ data,\ window) = \sum_{i=1}^{n} \sqrt{i)\,(i)\,(i)_{window})^2 + i)\,(i)\,(i)_{window})^2} \tag{23}$$

$$dWS(test\ data,\ window) = \sum_{i=1}^{n} \sqrt{i)\,(i)\,(i)_{window})^2 + i)\,(i)\,(i)_{window})^2} \tag{24}$$

$$dWD(test\ data,\ window) = \sum_{i=1}^{n} \sqrt{i)\,(i)\,(i)_{window})^2 + i)\,(i)\,(i)_{window})^2} \tag{25}$$

$$dP(test\ data,\ window) = \sum_{i=1}^{n} \sqrt{i)\,(i)\,(i)_{window})^2 + i)\,(i)\,(i)_{window})^2} \tag{26}$$

$$dH(test\ data,\ window) = \sum_{i=1}^{n} \sqrt{i)\,(i)\,(i)_{window})^2 + i)\,(i)\,(i)_{window})^2} \tag{27}$$

$$dC(test\ data,\ window) = \sum_{i=1}^{n} \sqrt{i)\,(i)\,(i)_{window})^2 + i)\,(i)\,(i)_{window})^2} \tag{28}$$

where.

- n – number of predictor locations
- w – window size
- represents a weather parameter from test data
- represents a weather parameter from an observation for the i_{th} predictor location in the window dataset

Setting n to 1, returns the equations for non-collaborative regression. Obviously when n > 1, the complexity of computing the Euclidean distances in equations (23)–(28) increases. The predicted value can be calculated as the average of the k rows with the smalles distance with the test data as given in equation (29).

$$Y(t) = \frac{1}{K} \sum_{j=1}^{k} Y_j \tag{29}$$

where.

- Y(t) represents the dependent variable at time t.
- K is the number of nearest neighbours to consider.
- is the jth nearest neighbours.

2.2.4. MLP collaborative regression

Multi-layered perceptron (MLPs) is a type of feed-forward neural network that is used extensively to solve several different problems including regression and classification. It has a three-layer architecture which consists of an input layer, hidden layer and output layer [21]. The input layer receives the input signal, the hidden layer is where most of the computations are performed, and the output layer performs classification or prediction.

The MLP learning algorithm optimizes the weights by using backpropagation [22] which is a quite straightforward process. The hidden layer nodes in the network are first assigned with random weights between +1 and −1. The first training batch is fed to the network and the output is observed. The network output is compared with the actual values by calculating the difference between them. The error is propagated backwards through the network and the weights are readjusted. This operation is performed until an acceptable error is reached. Fig. 9

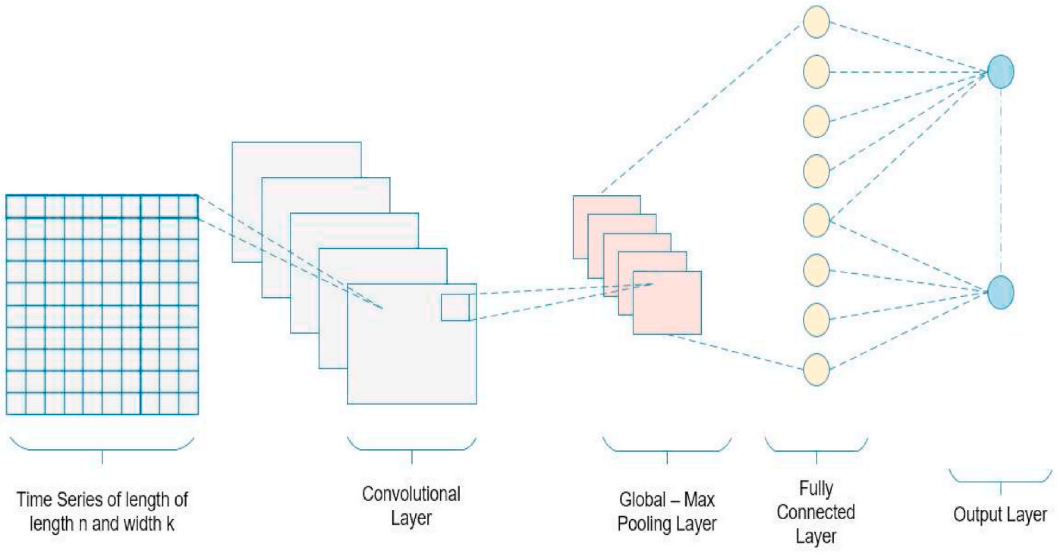


Fig. 10. CNN network architectures.

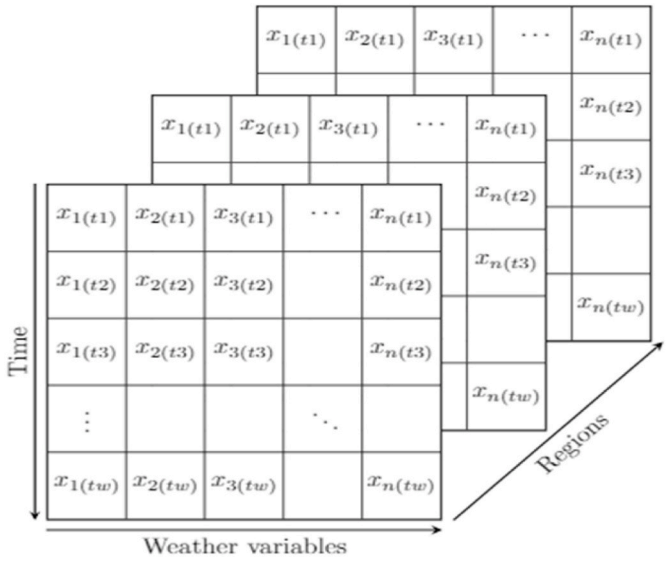


Fig. 11. CNN input tensor.

shows the architecture of an MLP network and how it is trained.

Before feeding data to a neural network, the data must first be normalized to enhance the computations and achieve more accurate results [24]. In this paper, the min-max normalization given in equation (30) has been used.

$$X_{normalised} = \frac{X_{current} - X_{min}}{X_{max} - X_{min}} \tag{30}$$

where, $X_{current}$ is the current value, X_{min} is the minimum value of X and X_{max} is the maximum value of X.

The MLP input which consists of inputs for different time steps can be summarised as follows:

$$MLP_{input} = \left[X1_{t1}, X2_{t1} \ldots X(n)_{t1}, X1_{t2}, X2_{t2}, \ldots, X(n)_{t2}, \ldots . X(n)_{tn}\right]$$

where, X1, X2, …, X(n) are the predictor variables for timestep t(n). In the non-collaborative case, the complexity is less as there is only one input layer as compared to collaborative MLP regression which has n input layers for n predictor locations.

2.2.5. CNN collaborative regression

A Convolutional Neural Network (CNN) is an advanced neural network designed to work with two-dimensional data, although one-dimensional and three-dimensional data can also be fed to the network. This type of network automatically extracts features from the input data grid and has been used mainly for object detection. The CNN architecture consists of an input layer, several hidden layers (convolutional, pooling and fully connected layers), and an output layer as shown in Fig. 10 [25].

The input to the CNN network is shown in Fig. 11.

The input tensor consists of a 3D matrix where each row represents

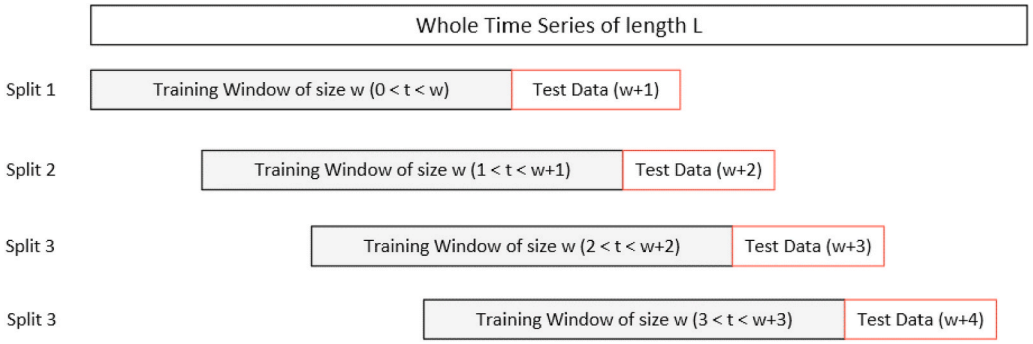


Fig. 12. Sliding window validation.

Fig. 13. Locations on google maps.

the predictor variables at a specific time step for a region. For non-collaborative regression, the number of predictor reigions is one. For collaborative regression, data from different regions are combined to form the model input tensor which results into a more complex 3D tensor input matrix as compared to the conventional non –collaborative case.

3. System testing and performance analysys

In this section, the performance of each machine learning model for each weather variable has been tested and the effect of using collaborative regression to predict the weather condition for a specific location is analyzed.

Table 3
Dataset details for simulation.

Location Details		
Location Name	Altitude (m)	Area [km^2]
Curepipe	561	24
Vacoas	430	110.45
Quatres Bornes	313	25.45
Moka	203	231
Dataset Details		
Observation Period	Samples per Location	Weather Parameters
1st May to 31st May 2021	2976	1. Temperature (°C) 2. Wind Speed (ms^{-1}) 3. Wind Direction – (°) 4. Pressure – (Pa) 5. Humidity – (%) 6. Cloudiness – (%)

3.1. Evaluation models and simulation dataset

Collaborative sliding window cross-validation for each algorithm has been performed in this work. In most cases when evaluating a machine learning model performance, the data is divided into a training and testing set. The testing set is obtained by randomly choosing a portion of the original data unseen by the model during training.

However, when dealing with time series, or any other type of data where the characteristics of the environment change with time, keeping the order of the reading is important. To cross-validate a time series, a sliding window analysis is performed. In this approach, a window of fixed size *w* is used to extract a sequence from the dataset. Then, the window is moved one sample forward and the next w samples are used to create another model to predict for t = t + w + 1. To evaluate the model, the Mean Average Percentage Error (MAPE) between the predicted values and actual values is calculated. The sliding window process validation is shown in Fig. 12.

Data were collected from the 1st to the 31st of May 2021 for four regions, namely, Moka, Curepipe, Vacoas and, Quatres Bornes. Fig. 13 shows the locations on GoogleMaps and the distance between each location.

Weather observations were collected at a rate of four samples per hour for the following parameters:

1. Temperature
2. Wind Speed
3. Wind Direction
4. Pressure
5. Humidity
6. Cloudiness

Table 4
Algorithms parameters details for simulation.

Machine Learning Algorithm Parameters		
Algorithms Name	Parameter Set	
MLR	Window size = 36	
	Ridge parameter = 0.0001	
MPR	Window size = 36	
	Model order = 2nd order	
	Ridge parameter = 0.0001	
KNN	Window size = 36	
	K = 3	
	Distance function = Euclidean distance	
Deep Learning Algorithm Parameters		
Parameter Name	MLP	CNN
Input Size	[1,72]	[32,2, n_locations]
Activation Function	Sigmoid	Tanh
Max Epochs	1000	1000
Max Error	0.0001	0.0001
Momentum	0.9	0.9
Learning Rate	0.01	0.01
L2 Regularization	0.0001	0.0001

Table 5
Experiment details.

Experiment	Predictor Locations	Response Location	Prediction Time
Experiment 1	Curepipe	Curepipe	15 Mins
Experiment 2	Moka	Moka	15 Mins
Experiment 3	Curepipe Quatres Bornes Vacoas	Curepipe	15 Mins
Experiment 4	Moka Quatres Bornes Vacoas	Moka	15 Mins

The Mean Absolute Percentage Error (MAPE) between the actual and predicted values for each parameter was then computed over the whole dataset as given in equation (31).

$$MAPE_{param} = \frac{1}{N}\sum_{i=1}^{N}\frac{|Actual(i) - Predicted(i)|}{|Actual(i)|} \quad (31)$$

where, N represents the total number of points in the dataset.

The overall percentage error was then calculated for all the algorithms used as given in equation (32).

$$Average\ Error_{model} = \frac{1}{N}\sum_{param=1}^{N} MAPE_{param} \quad (32)$$

Table 3 below shows a summary of the regions and dataset being used.

3.2. Simulation algorithms

For the simulation, cross-validation was performed using the following five algorithms:

- Multiple Linear Regression (MLR)
- Multiple Polynomial Regression (MPR)
- K-Nearest Neighbours (KNN)
- Multilayer Perceptron (MLP)
- Convolutional Neural Network (CNN)

The parameters for each algorithm are given in Table 4.

3.3. Simulation procedures

Four experiments were performed to investigate the effect of collaborative regression on the accuracy of the sliding window validation. The predictor locations were chosen based on geographical adjacency as given in Table 5.

Experiments 1 and 2 are the baseline experiments whrereby the five machine learning algorithms (MLR, MPR, MLP, KNN and CNN) are used without the parameters for collaborative predictions i.e. with only one predictor location and with their conventional equations. For MLR the conventional equations are given in equations (2)–(7). For MPR the conventional equations are (16),(17),(18),(19),(20) and (21). For KNN the conventional equations are (23) to (28) with n = 1. The conventional forms of MLP and CNN simply use one predictor location.

For experiments 3 and 4, previous data from two other locations will be combined to predict the weather condition using collaborative regression. The average error for each algorithm's model after each experiment will be recorded and then plotted in a bar chart for comparison.

There are two main limitations of the proposed collaborative ML

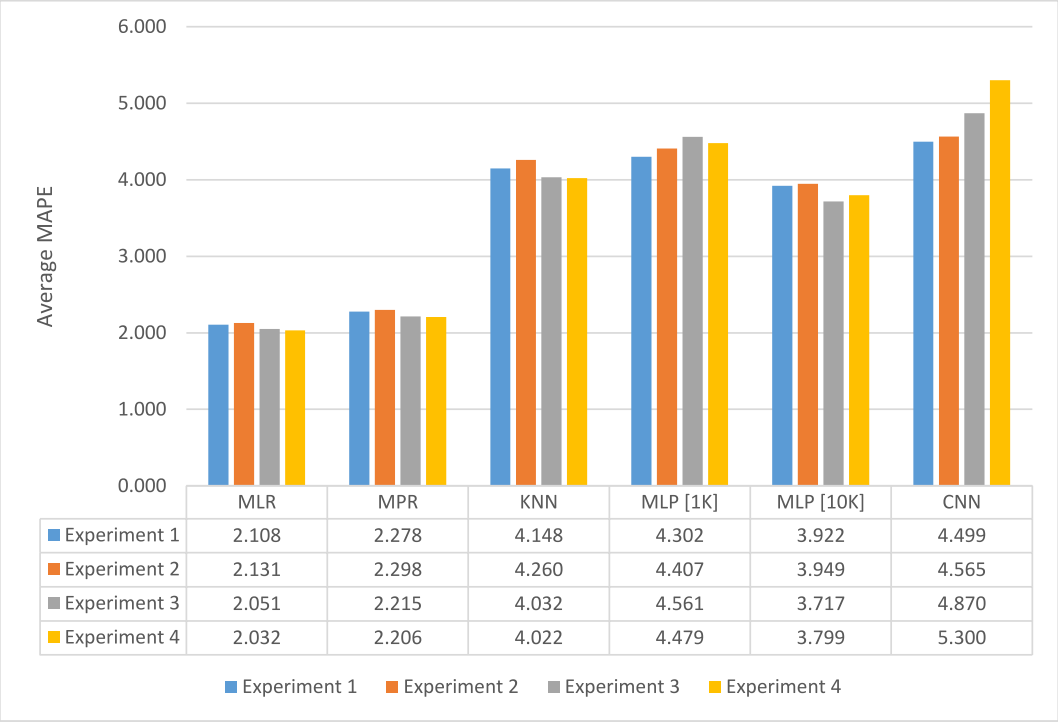
Table 6
MLP network parameters optimization.

Hyper Parameters	Training MLP[1k]	Training MLP [10k]
Max Epochs	1000	10,000
Learning Rate	0.01	0.001
Training Sample Size	2976	4464

Table 7
Experimental results.

Experiment 1						
Variables	**MLR**	**MPR**	**KNN**	**MLP [1K]**	**MLP [10K]**	**CNN**
Temperature	0.413	0.531	1.007	0.731	0.699	0.803
Wind_Speed	1.696	1.814	4.125	5.056	4.912	5.937
Wind_Direction	1.177	1.305	1.848	2.702	2.135	2.853
Pressure	0.009	0.009	0.024	0.040	0.035	0.032
Humidity	0.747	0.726	1.722	1.453	0.995	1.797
Cloudiness	8.607	9.280	16.163	15.829	14.756	15.569
Average Error	2.108	2.278	4.148	4.302	3.922	4.499
Experiment 2						
Variables	**MLR**	**MPR**	**KNN**	**MLP [1K]**	**MLP [10K]**	**CNN**
Temperature	0.409	0.524	0.977	0.752	0.684	0.892
Wind_Speed	1.753	1.865	4.195	5.039	4.851	6.065
Wind_Direction	1.117	1.190	1.798	2.679	2.012	2.795
Pressure	0.009	0.009	0.024	0.038	0.038	0.032
Humidity	0.752	0.740	1.764	1.504	1.125	1.836
Cloudiness	8.744	9.463	16.804	16.432	14.985	15.769
Average Error	2.131	2.298	4.260	4.407	3.949	4.565
Experiment 3						
Variables	**MLR**	**MPR**	**KNN**	**MLP [1K]**	**MLP [10K]**	**CNN**
Temperature	0.370	0.415	0.927	0.704	0.680	1.395
Wind_Speed	1.667	1.753	3.885	5.770	3.985	4.935
Wind_Direction	1.137	1.184	1.867	2.722	1.986	2.495
Pressure	0.008	0.008	0.024	0.033	0.032	0.030
Humidity	0.705	0.735	1.591	1.467	1.351	3.560
Cloudiness	8.416	9.194	15.896	16.672	14.267	16.804
Average Error	2.051	2.215	4.032	4.561	3.717	4.870
Experiment 4						
Variables	**MLR**	**MPR**	**KNN**	**MLP [1K]**	**MLP [10K]**	**CNN**
Temperature	0.346	0.392	0.896	0.680	0.585	1.380
Wind_Speed	1.614	1.733	3.956	5.736	4.875	4.960
Wind_Direction	1.103	1.189	1.804	2.697	1.765	3.008
Pressure	0.008	0.008	0.024	0.034	0.032	0.032
Humidity	0.682	0.685	1.614	1.462	1.321	3.598
Cloudiness	8.441	9.228	15.836	16.267	14.215	18.822
Average Error	2.032	2.206	4.022	4.479	3.799	5.300

From Table 7, it can be observed that using multiple predictor locations decreases the average model's error for the two test cases. The MAPE performance of the non-collaborative regression varies between 0.009 and 16.80 (highlighted in green), whereas the collaborative regression MAPE varies between 0.008 and 18.82 (highlighted in purple).



	MLR	MPR	KNN	MLP [1K]	MLP [10K]	CNN
Experiment 1	2.108	2.278	4.148	4.302	3.922	4.499
Experiment 2	2.131	2.298	4.260	4.407	3.949	4.565
Experiment 3	2.051	2.215	4.032	4.561	3.717	4.870
Experiment 4	2.032	2.206	4.022	4.479	3.799	5.300

Fig. 14. Performance of the algorithms for the four experiments.

approach:

1. The complexity increases with the number of predictor locations, hence only the most significant predictor locations should be used based on experimental data.
2. The training window should be kept relatively small as real-time forecasts have to be supported. This can lead to a loss in prediction accuracy in certain cases, although this can be compensated for with collaborative forecasting and better tuning of the hyperparameters.

The deep learning networks can be optimized by tuning the hyper-parameters. For a demonstration, the parameters for the MLP network were modified and each experiment was repeated with the new hyper-parameters. Table 6 shows the hyperparameters that were modified.

3.4. Experimental results

The results for experiments 1–4 has been tabulated in Table 7.

A bar chart is plotted to compare the error performance of each algorithm with and without collaborative regression. Fig. 14 shows the average error of each algorithm for experiments 1 to 4.

As can be observed from the bar chart, the average MAPE obtained by the machine learning algorithms for the collaborative experiments (Experiments 3 & 4) was generally lower when compared to the non-collaborative ones (Experiments 1 & 2). This decrease in error can be attributed to the increasing number of correlated explanatory variables. The linear regression models such as MLR and MLP are able to better exploit the correlated variables, therefore, decreasing the overall error.

The deep learning networks, however, experienced a mean increase of 8% for the average MAPE. There was a 3.79% increase in average error for the MLP network and a 12.2% increase for the CNN network. However, the increase in error can be attributed to the small training dataset and unoptimized network. The deep learning networks can be optimized by tuning the hyper-parameters. The experiment labelled MLP [10K] demonstrates that with the necessary parameter tuning, the overall error for the deep learning networks can be further minimised.

It can be observed from the above graph, that the optimized network

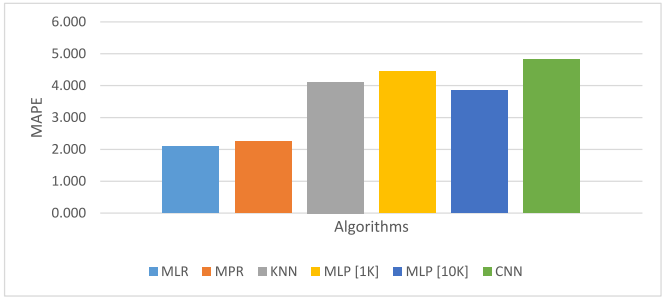


Fig. 15. Algorithms Perfomance for the four experiments.

provides the same performance as the KNN algorithm. The small learning rate allows the network to learn an optimal set of weights at the cost of higher iterations. This shows that by using more predictor locations, the complexity of the neural network increases and must, therefore, be allowed to train for larger iterations with a smaller learning rate so that the network is allowed to converge with optimal weights. The overall error decreased by 0.6% when the hyper parameters were tuned.

To compare the overall performance of the algorithms, the average MAPE for each algorithm was calculated using the following equation:

$$Algorithm\ Average\ MAPE = \frac{1}{4}\sum_{i=1}^{4} Mape(i) \tag{33}$$

Then, the average MAPE for each algorithm was plotted on a barchart as shown in Fig. 15.

When comparing the algorithms, it was observed that the average error for machine learning algorithms (MLR, MPR and KNN) is much lower than that of the deep learning networks (MLP and CNN). The lowest overall error achieved was 2.11% from the MLR algorithm while the highest error encountered was 4.81% by the CNN network. This indicates that the MLR algorithm was able to better extract the non-linear relationship between the explanatory variables. From the above graph, it can be shown that tuning the hyper parameters [MLP 10K] of

deep learning networks can further decrease the overall error obtained.

4. Conclusions

In this paper a collaborative weather forecasting system to perform data analytics on weather data using machine learning algorithms and cloud computing facilities was developed. Essentially, two edge devices were used to collect data from the OpenWeather API. Analytics were performed both on a local server and a cloud hosted servlet using MLR, MPR, KNN, CNN and MLP. An interactive android app was developed to provide mobile clients with the latest weather conditions in some regions of Mauritius. The algorithms were tested using sliding window cross-validation for a prediction interval of 15 min. A series of four experiments were performed consisting of two collaborative regression and two non-collaborative regression. The experiments showed that the machine learning algorithms were 65% more accurate than the deep learning networks. However, after tuning some hyperparameters, the MLP network was operating on par with the classical learning algorithms. The collaborative schemes gave a 5% lower average MAPE when compared to non-collaborative schemes. The most performing algorithm was MLR with a MAPE ranging between 0.009 and 6% and the worst performing algorithm was CNN with MAPE ranging between 8 and 16%. An important implication of the results obtained is that using collaborative regression can increase the accuracy of the prediction for a location. The testing of the different algorithms also demarcates from previous works [19,26] whereby data for only a single location was used. Similar to Ref. [9], significantly better accuracy was obtained by collaborative forecasting. However, in contrast to Ref. [9] where only the performance analysis was presented for temperature data, in this work six weather parameters are considered and detailed mathematical models for the collaborative machine learning are given. Moreover, a complete weather forecasting system with both local and cloud servers as well as different client devices have been developed by extending the work in Ref. [17]. This system allows testing using any number of predictor locations desired. The most interesting contribution of this work is that collaborative regression can be used to increase the accuracy of weather predictions when using machine learning. However, to achieve good accuracy with neural networks, the hyperparameters must be tuned to provide a network for optimal operation. An interesting future work would be to use a stacking ensemble learning machine learning model to perform predictions. It involves combining the predictions from multiple regression models on the same dataset [27].

Declaration of competing interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Acknowledgments

The authors would like to thank the University of Mauritius for providing the necessary facilities to conduct this research.

References

[1] Wiston M, Km M. Weather forecasting: from the early weather wizards to modern-day weather predictions. J Climatol Weather Forecast 2018. https://doi.org/10.4172/2332-2594.1000229. 06 02.
[2] Han J, Kamber M, Pei J. Data mining: concepts and techniques. In: Han J, Kamber M, Pei J, editors. Data mining: concepts and techniques. San Francisco, CA, itd: Morgan Kaufmann; 2012. https://doi.org/10.1016/B978-0-12-381479-1.00001-0. 2012.
[3] Siuta D, West G, Modzelewski H, Schigas R, Stull AR. Viability of cloud computing for real-time numerical weather prediction. Weather Forecast 2016;31(6). https://doi.org/10.1175/WAF-D-16-0075.1.
[4] Krishnappa DK, Irwin D, Lyons E, Zink M. CloudCast: cloud com-puting for short-term mobile weather forecasts. 2012. https://doi.org/10.1109/PCCC.2012.6407739.
[5] de Castro JT, Salistre GM, Byun YC, Gerardo BD. Flash flood prediction model based on multiple regression analysis for decision support system. In: Lecture notes in engineering and computer science, vol. 2; 2013.
[6] Kothapalli S, Totad SG. A real-time weather forecasting and analysis. In: IEEE international conference on power, control, signals and instrumentation engineering (ICPCSI), vol. 2017; 2017. p. 1567–70. https://doi.org/10.1109/ICPCSI.2017.8391974.
[7] Beeharry Y, Fowdur TP, Sunglee JA. A cloud-based real-time weather forecasting application. 2019. https://doi.org/10.1109/TELSIKS46999.2019.9002327.
[8] Zhou K, Zheng Y, Li B, Dong W, Zhang X. Forecasting different types of convective weather: a deep learning approach. J Mete-orol Res 2019;33(5). https://doi.org/10.1007/s13351-019-8162-6.
[9] Jakaria AHM, Hossain Md Mosharaf, Rahman Mohammad Ashiqur. Smart weather forecasting using machine learning: a case study in Tennessee. In: Proceedings of ACM mid-southeast conference (Mid-Southeast'18). New York, NY, USA: ACM; 2018. p. 4. https://doi.org/10.1145/nnnnnnn.nnnnnnn.
[10] Verma G, Mittal P, Farheen S. Real time weather prediction system using IOT and machine learning. In: 6th international conference on signal processing and communication (ICSC); 2020. p. 322–4. https://doi.org/10.1109/ICSC48311.2020.9182766. 2020.
[11] Huang ZQ, Chen YC, Wen CY. Real-time weather monitoring and prediction using city buses and machine learning. Sensors 2020;20(18):5173. https://doi.org/10.3390/s20185173. Published 2020 Sep. 10.
[12] Hewage P, Trovati M, Pereira E, et al. Deep learning-based effective fine-grained weather forecasting model. Pattern Anal Appl 2021;24:343–66. https://doi.org/10.1007/s10044-020-00898-1.
[13] Mimboro P, Lumban Gaol F, Lesie Hendric Spits Warnars H, Soewito B. Weather monitoring system AIoT based for oil palm plantation using recurrent neural network algorithm. In: IEEE 5th international conference on information technology, information systems and electrical engineering (ICITISEE); 2021. p. 283–7. https://doi.org/10.1109/ICITISEE53823.2021.9655818. 2021.
[14] Fente DN, Kumar Singh D. Weather forecasting using artificial neural network. In: Second international conference on inventive communication and computational technologies (ICICCT); 2018. p. 1757–61. https://doi.org/10.1109/ICICCT.2018.8473167. 2018.
[15] Dadhich Shruti, Pathak Vibhakar, Mittal Rohit, Doshi Ruchi. Chapter 10 machine learning for weather forecasting". Machine learning for sustainable development. In: Kamal Kant Hiran, Deepak Khazanchi, Ajay Kumar Vyas, Sanjeevikumar Padmanaban, editors. Berlin, Boston: De Gruyter; 2021. p. 161–74. https://doi.org/10.1515/9783110702514-010.
[16] Chao Zeyi, Pu Fangling, Yin Yuke, Han Bin, Chen Xiaoling. Research on real-time local rainfall prediction based on MEMS sensors. J Sens 2018. https://doi.org/10.1155/2018/6184713. Article ID 6184713, 9 pages2018.
[17] Fowdur TP, Babooram L, Nazir Rosun MNI, Indoonundon M. REAL-TIME cloud computing and machine learning applications". Nova Science Publishers; July 2021, ISBN 978-1-53619-813-3. Computer Science, Technology and Applications Book Series.
[18] OpenWeather Weather API. OpenWeather. 2021. https://openweathermap.org/api. [Accessed 28 June 2021]. accessed.
[19] Zaw WT, Naing TT. Modeling of rainfall prediction over Myanmar using polynomial regression. International Conference on Computer Engineering and Technology; 2009. p. 316–20. https://doi.org/10.1109/ICCET.2009.157. 2009.
[20] Zhang S, Cheng D, Deng Z, Zong M, Deng X. A novel kNN algo-rithm with data-driven k parameter computation. Pattern Recogn Lett 2018;109. https://doi.org/10.1016/j.patrec.2017.09.036.
[21] Raj P, Lin J-W. The digital twin paradigm for smarter systems and environments: the industry use cases. In: Advances in computers. first ed., vol. 117; 2020. vol. 117 1.
[22] Noriega Leonardo. Multilayer perceptron tutorial. Nov. Beaconside Staffordshire ST18 0DG; 2005. Accessed: Jun. 28, 2021. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.608.2530&rep=rep1&type=pdf.
[23] Ghorbani S, Barari M, Hoseini M. Presenting a new method to improve the detection of micro-seismic events. Environ Monit Assess 2018;190(8). https://doi.org/10.1007/s10661-018-6837-6. Aug.
[24] Sola J, Sevilla J. Importance of input data normalization for the application of neural networks to complex industrial problems. IEEE Trans Nucl Sci 1997;44. https://doi.org/10.1109/23.589532. 3 PART 3.
[25] Exploring Deep Learning & CNNs - RSIP Vision. RSIP vision. 2021 [Online] Available: https://www.rsipvision.com/exploring-deep-learning/.
[26] Horak HG. The effect of the atmosphere on laser range determination. SAO Spec Rep Dec. 1966;236:19.
[27] Brownlee J. Stacking ensemble machine learning with Python. Mach Learn Mast Apr 2021;17. accessed Jun. 28, 2021), https://machinelearningmastery.com/stacking-ensemble-machine-learningwith-python/.