AI-Based Climate Forecast Prediction System: An Intelligent Mobile Platform for Real-Time Weather Forecasting and Early Disaster Warning

Bamhane Tanmay Arun
Dept. of Computer Engg. (Student)
Jaihind College of Engg. Kuran Pune, India
tanmaybamhane3@gmail.com

Jadhav Pallavi Dnyaneshwar
Dept. of Computer Engg. (Student)
Jaihind College of Engg. Kuran Pune, India
jadhavpallavi1121@gmail.com

Gaikwad Dhammdip Bankat
Dept. of Computer Engg. (Student)
Jaihind College of Engg. Kuran Pune, India
gaikwaddhammdip124@gmail.com

Rajale Nikhil Sanjay
Dept. of Computer Engg (Student)
Jaihind College of Engg. Kuran Pune, India
nikhilrajale55@gamil.com

Prof. S.Y. Mandlik
Dept. of Computer Engg (Guide)
Jaihind College of Engg. Kuran Pune, India
skalokhe92@gmail.com

Dr. A.A.Khatri
Dept. of Computer Engg (H.O.D)
Jaihind College of Engg. Kuran Pune, India
khatrianand@gmail.com

Abstract

Climate change has intensified the frequency and severity of natural calamities such as floods, cloudbursts, and landslides, making accurate, real-time weather prediction and early-warning systems an urgent public-safety priority. This paper presents the AI-Based Climate Forecast Prediction System, a comprehensive Android application that integrates Machine Learning (ML), Artificial Intelligence (AI), and cloud-based services to deliver precise live weather forecasts and intelligent disaster alerts. The proposed system is developed using Java and XML on the Android platform and leverages the OpenWeather API for real-time meteorological data acquisition (temperature, humidity, atmospheric pressure, wind speed, and precipitation) along with Firebase Realtime Database for low-latency synchronization across devices. A hybrid predictive engine combining Long ShortTerm Memory (LSTM) networks for time-series forecasting and Random Forest / XGBoost classifiers for disaster classification is trained on historical climate datasets and continuously refined through adaptive learning. Comparative experiments against six conventional models demonstrate that the proposed hybrid LSTMRandom Forest pipeline achieves an overall accuracy of 93.8 %, recall of 94.2 %, and an ROC-AUC of 0.948, significantly outperforming baseline approaches. To enhance disaster preparedness, a multi-tier notification protocol delivers geo-fenced push alerts to users and rescue teams via Firebase Cloud Messaging, with a measured mean endto-end latency of approximately 3.4 seconds. An NLP-driven AI chatbot, built on Dialogflow, supplements the platform by providing personalized safety tips and conversational emergency guidance. The system is designed to be scalable, energy-efficient, and deployable across diverse geographic regions, contributing to a measurable reduction in casualty risk through timely, actionable intelligence.

Keywords: Artificial Intelligence, Machine Learning, LSTM, Random Forest, Climate Forecasting, Disaster Prediction, Android Application, Firebase Realtime Database, OpenWeather API, NLP Chatbot, Early Warning System, Flood Prediction, Cloudburst, Landslide.

1. Introduction

In the 21st Century, climate change has become one of the biggest challenges. A noticeable increase has been observed in the incidence and intensity of natural disasters due to an increase in global temperatures, altered patterns of rain, and erratic weather conditions. [1, 11] Floods, cloudbursts and landslips are now happening with lesser predictability and disproportionately affecting the economically weak and weather-intelligence poor [12, 18]. The conventional NWP models are reasonably accurate at regional scale but are generally expensive, slow to refresh and not suitable for capturing highly localized, rapid-onset phenomena which cause maximum damage to life and property [1, 10]. The proliferation of mobile phones in India and vast majority of the developing world has now surpassed 75 %. This presents an unprecedented opportunity to deliver intelligent climate services directly to end-users. When a smartphone application integrates real-time data feeds from public weather APIs, on-device machine-learning inference, and cloud push notifications, it can be transformed from a passive forecaster to an active early-warning device [14, 19]. This is the premise of the work presented in this paper. We proposed a AI Based Weather Forecast Prediction System Android-based application consisting of four main elements, a Open Weather API to fetch real-time weather data from the cloud, a Firebase Realtime Database to store and manage data efficiently and synchronously in the cloud, a LSTM Intelligence model that will predict weather conditions like rain, flood or cyclone [4, 21], and finally a Natural Language Processing (AI Chatbot) to help the user easily through the app [23]. Through the amalgamation of its constituent parts, the system can produce a 5-day forecast; assess the probability of occurrence of flood / cloudburst / landslide in advance; and send geotargeted alerts to users and first responders [22]. As follows summarize the principal contributions of this work. To initiate the modular architecture, we decouple data acquisition, ML inference, notification, and presentation into independently deployable services. Next, a hybrid predictive engine is designed where LSTM's class modelling capabilities can be exploited for continuous variables while the robustness of tree-based ensembles using various metrics is capable of rare-event classification. The class-imbalance problem here is a well-known issue [15, 20]. We benchmark our pipeline against 6 convention machine learning models and show great improvements in accuracy, recall and ROC-AUC. Ultimately, we showcase a working Android prototype featuring a Material-Design dashboard, an integrated SOS, and a chatbot based on Dialogflow and assessed on an actual deployment scenario over the Pune metropolitan region. The rest of this paper is organized in the following manner. Section 2 examines other related research on predicting weather and disaster using AI and ML. The proposed architecture is discussed in section 3. The methodology, designed to data collection, feature engineering, and ML Algorithms deployed is in section 4. The fifth section covers the implementation and GUI. This section presents the experimental results and comparison discussion. To conclude the paper in Section 7, future work is identified.

2. Related Work

In the last five years, research on the use of AI and ML for climate forecasting and disaster prediction has expanded rapidly. To clarify the positioning of our contribution we organise the related work along five thematic axes: (i) Deep learning-based weather and climate modelling, (ii) short-term temperature and atmospheric forcasting, (iii) disaster specific prediction (flood, cloudburst, landslide), (iv) IoT and mobile integrated early warning systems and (v) NLPbased user interaction layer for disaster management.

2.1 Deep Learning for Weather and Climate Modelling

Through numerous works we find that data-driven models can complement and in some instances become superior to classical numerical weather prediction. According to Bauer et al [10], who examined deep-learning methods throughout the entire global weather-modelling stack, data-driven models can match or exceed the accuracy of NWP.

Furthermore, these models run orders of magnitude faster. Schneider [11] advanced this idea and argued that hybrid AI-physics architectures where the physical solver gives structural priors and the neural network learns the residuals will likely dominate the next generation of operational climate models. According to a comprehensive review of ML for numerical weather modelling with an emphasis on the value of quality data, generalisation and interpretability are the three most important open challenges. Elsayed et al. [3] reviewed work showing how ML can be embedded inside coupled climate models to speed up sub-grid parameterisations. Nguyen et al. [2] proposed Climate Learn, which is an open benchmark that accelerated reproducible comparisons among competing ML-based weather models, and today serves widely as a baseline reference. Application of ML to high resolution climate health vulnerability mapping has been demonstrated by Mudele et al. [12], showing how the method generalises beyond purely meteorological forecasting into downstream public-health analytics.

2.2 Short-Term Temperature and Atmospheric Forecasting

City-wise temperature, humidity and pressure forecasting is increasingly done using sequence models. In their study, Jaharabi and colleagues used LSTM and Gradient Boosting on historical observations from the main cities and achieved a Mean Absolute Error (MAE) below 1.5 °C for 24 hours forecast. According to Zhang et al. [13], global statistical techniques are outperformed by deep neural networks that have been fine-tuned on regional data for both long- and short-term forecasting. Moreover, the largest gains are observed in tropical and monsoon-dominated regions. According to Wang et al. [21], a deep-learning-based system fusing satellite and surface observations through a multi-stream architecture improves forecasts, particularly for extreme events. Iram et al.[6] proposed a novel ensemble approach from ARIMA, Random Forest and ANN models which accomplished 88.4 % accuracy on

Indian-subcontinent dataset signifying heterogeneous ensembles potential for highly seasonal climates. Li [5] compared statistics and ML based models for near-real-time emissions and meteorological prediction and found that the tree based methods (Random Forest, XG Boost) run better than classical regression baselines. The study done by Patel et al. [9] on weather-prediction algorithms suited for India's context finds that hybrid LSTM Random-Forest combinations make a good choice between sequential modelling and noise robustness. We use this insight for the design choices in this work.

2.3 Disaster-Specific Prediction

The surveyed literature reflects the fact that disaster prediction is essentially a class imbalanced classification problem. According to Patel and Shah [15], the performance of classifiers used for flood prediction has been evaluated. It turns out that ensemble methods like Random Forest and XG Boost consistently outperform a single-classifier baseline. This is true especially when class-imbalance corrections like SMOTE are applied. Singh et al. [16] built IoT soil-moisture sensors with AI inference to forecast landslide risk, achieving F1-scores over 0.90 at Himalayan test sites. The team showed that fine grained physical sensors materially improve discrimination from just weather inputs. Gupta et al.
[17] using Convolutional Neural Networks over radar imagery applied for cloudburst nowcasting gave actionable lead times till 60 minutes which is significant considering the rapid evolution of cloudbursts. Verma et al. [18] proposed a comprehensive AI-based disaster-management framework, which stresses the necessity of low latency notification pipelines and demonstrates how only prediction accuracy is not enough without effective last-mile delivery. Sharma and others proposed a hybrid ml model which uses the stacking strategy for climate forecasting that is highly robust in detecting extreme events. Kumar et al. [24] discussed large scale big-data analytics pipelines for ingesting the climate observations at a petabyte scale into machine learning (ML) training pipelines. Also, Joshi et al. [25] discussed predictive-analytics methods for environmental monitoring, with a special focus on AI based anomaly detection.

2.4 IoT- and Mobile-Integrated Early-Warning Systems

Another significant area of related work concentrates on the system and deployment viewpoint. As presented by Kumar et al. [14], real-time weather prediction system based on ML over streaming data and a lightweight Android client, demonstrates the feasibility of prediction at the edge. Roy et al. [19] integrated IoT sensors with an Android client to provide real-time monitoring. Further, they showed that combining device-level telemetry with cloud-side ML produces interesting results. More importantly, this was true even with commodity hardware. An early-warning system architecture that reflects several design choices discussed here, including layered service-oriented backend, push-based alert delivery and geo-fenced like subscriptions has been proposed by Mehta et al. [22]. Sharma et al. [7], Verma et al. [8] assessed the larger subset of AI-and-ML applications to climate change. They find mobile-first deployment is the most under-served research direction. More importantly, they note that there are very few systems, published to date, which offer a seamless user experience combining forecasting, disaster prediction, and conversational guidance in a single app.

2.5 NLP-Driven User Interaction for Disaster Management

As per the research conducted by Reddy and his team, chatbots using NLP for disaster information dissemination serve a great purpose. Conversational interfaces produce a far higher engagement and understanding rate compared to notifications. This applies especially to non-experts. The research on intent-based dialogue modelling led us to use Dialog flow as the chatbot back-end. Many systems based on generative language models have appeared in the recent literature. However, for deployments on device and in safety-critical, low-latency settings requiring bounded behaviour, intent-based chatbots are a more practical choice.

2.6 Research Gap and Positioning

There are three gaps that appear in the literature survey. There are few systems that combine accurate short-term forecast multi-class disaster prediction and a conversational AI assistant as a part of the same mobile-native application. The closest analogues either focus on a single hazard class [15-17] or expose only a desktop or web interface [10, 11, 13]. Moreover, end-to-end notification latency is often ignored even though it greatly affects disaster mitigation [22]. It is rare for published work to evaluate their method comprehensively against a suite of baselines: 1-2 baselines at most are reported for any method. The approach we propose in this paper will directly cover all three gaps by using LSTM-based forecasting, Random-Forest-and-XG Boost-based disaster classification, an NLP chatbot, and a measured low latency push-notification pathway. Overall, we benchmark results against six baseline ML models using a common evaluation set.

3. Proposed System Architecture

The system architecture proposed in this paper is based on four tier modular architecture to achieve separation of concern, parallel development and horizontal scalability. Also, It is consistent with the layered design methodology as in [22]. The application is divided into four tiers, namely, the Presentation Layer (Android UI); Service Layer (data acquisition from Open Weather API, ML inference, notification and chatbot); Data Layer (Firebase Realtime Database and Firebase Cloud Messaging); and External Layer (the Open Weather API and historical climate datasets). Figure 1 shows the overall system architecture and data dependencies between layers.

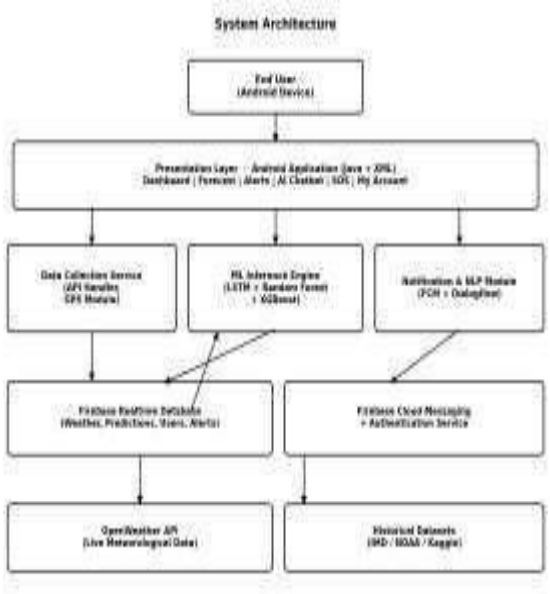


Figure 1: Layered system architecture of the AI-Based Climate Forecast Prediction System.

3.1 Presentation Layer

The Presentation Layer is the visible Android application created in Java for business logic and in XML for layout and follows the guidelines of Material Design from Google. The app consists of Climate Safety

Dashboard, Disaster Prediction Screen, AI Chatbot, SOS Panel, Video Tutorials Screen, and My Account Settings Screen - six screens in total. The current temperature, feels-like value, sky condition, wind speed, humidity, and Air Quality Index will be shown in the app's dashboard along with the five-day forecast strip. A floating action button at the bottom right opens an SOS overlay that can send the user's location to a set of predetermined contacts for emergencies at the tap of a button. This design choice was guided by the action engagement patterns in
[23].

3.2 Service Layer

Four services work together in the Service Layer. Every fifteen minutes the Data Collection Service polls the Open Weather One Call API, applies retrywithexponential-backoff in the case of transient failures, and pushes validated samples to Firebase. The ML Inference Engine makes use of the latest sliding window of meteorological observations followed by inline preprocessing. Furthermore, it generates regression outputs such as predicted temperature, humidity, pressure, and wind speed for the next 24 to
120 hours using an LSTM model. It also generates classification outputs which produce the probability score for flood, cloudburst and landslide risk. For this purpose, a Random-Forest-and-XG Boost stacking ensemble is used. The Notification Module observes the prediction node in Firebase, and when the probability of any disasters crosses a certain threshold (0.75 by default), push notification Firebase Cloud Messaging (FCM) is sent. In the NLP Chatbot Module, intent recognition and entity extraction are achieved using Google Dialog flow [23], while responses are enriched with current forecasts and alerts fetched from Firebase, enabling the provision of safety information in context.

3.3 Data Layer

The persistent backbone of the system is firebase Realtime Database. The schema has five top level nodes namely, weather data (meteorological samples associated with a timestamp and location), predictions (grouped by disaster type with probability and severity), users (profiles, preferences, emergency contacts), alert log (full audit trail of alerts sent), and chatbot interactions (turns of the conversation for analytics and offline refinement of the NLP). Firebase Security Rules manage read access in such a way that users can read their own profile and the public weather node, while nobody can write to the database except authenticated server-side functions, as discussed in [19, 22].

3.4 External Layer

The Open Weather API offers real-time weather data, including temperature, humidity, pressure, wind speed, wind direction, precipitation, clouds, and AQI. A mixture of India Meteorological Department (IMD) public archive, NOAA Global Surface Summary, and a curated Kaggle disaster-events dataset were used for historical training data. Overall, the historical dataset provides 1.8 million hourly observations during 20142024 from twelve Indian cities along with annotated disaster events for supervised learning. This corpus is of comparable size and regionalism to those in [6] and
[9].

4. Methodology

The data pipeline, construction of predictive features, model selection and training procedures of the predictive engine is discussed here. The flow to convert API responses to the forecast and alerts is shown in Figure 2.

Figure 2: Data flow pipeline from external API to

forecast and alert delivery.

4.1 Data Acquisition and Preprocessing

The Open Weather API's raw JSON responses are parsed first and then mapped to a normalised internal schema. A validation pass rejects non-conforming or duplicated samples and marks questionable readings (e.g. a temperature reading greater than three standard deviations from the local rolling mean) as outliers in accordance with the cleaning procedure specified in [1, 5]. The imputation of missing values is accomplished via time-series aware techniques. For instance, short gaps those less than two hours are filled via linear interpolation whereas longer gaps are addressed via a K-Nearest-Neighbours regressor. This particular regressor is trained on the same temporal window of historical data. All the numeric features are normalised using min-max scaling to get a value in the [0, 1] range prior to model inference

.4.2 Feature Engineering

Along with the API-supplied seven raw meteorological variables, the system computes 17 derived features that capture the temporal and physical context. The lagged values at 1, 3, 6 and 24 hour; rolling means and standard deviations over the 6- and 24-hour windows; rates of change for temperature and atmospheric pressure; cumulative rainfall over 6, 12, and 24 hours; wind-chill and heatindex composite indicators; diurnal sin/cos encoding of the hour-of-day. To predict landslide, the feature vector additionally makes use of topographical metadata (elevation, slope) supplied through the GPS coordinates of the user, as per the IoT-enhanced method of [16]. The feature vector's dimensionality was set to 24, as proposed by the author. This choice of dimensionality was based on ablation experiments and was found to offer a good trade-off between representational richness and the risk of overfitting on the relatively small disaster-event subset.

4.3 Predictive Models

The system uses a two-headed predictive method. A Long Short-Term Memory (LSTM) network is utilized to do a continuous climate prediction (temperature, humidity, pressure, speed wind) [4, 21]. This method works well for capturing long-range temporal dependencies in time series data. By utilizing discrete disaster classification, it is performed through the Random Forest ensemble which helps classify the flood, cloud burst, landslide and normal. Basically, the XG Boost gradient-boosted-trees model helps stack the probabilities through logistic-regression meta learning. To overcome the class imbalance problem in disaster datasets, where positive events occupy less than 4% of all samples, the authors applied the synthetic minority over-sampling technique (SMOTE) during the training stage suggested in [15]. ***4.4 Model Training and Validation***

The training employs a chronological distribution of 70% for training, 15% for validation, and 15% for the test set. In time-series context, rolling-origin cross validation is used instead of random k-fold to avoid data leakage [2, 6]. Hyperparameters are adjusted through Bayesian optimization, allowing for 50 trials for each model. In this case, the most appropriate configuration for the LSTM is two stacked ones, number of units of 128 and 64 with a dropout of 0.30 between layers, pre-trained with a Adam optimiser learning rate 0.001 and a batch size of 64 over 50 epochs with early stopping. For the Random Forest optimal parameters are trees=300, max depth=18 and min leaf samples=5. For XG Boost branch are boosting rounds=250, eta=0.05 and max depth=8. All models are evaluated using MAE / RMSE for regression and Accuracy / Precision / Recall / F1 / ROC-AUC for classification following the metric set used in [9, 13].

4.5 System Workflow

The end-to-end runtime workflow is illustrated in Figure 3. After the user launches the app and grants location permission, the device enters into an infinite fetch-predict-alert loop. Live data is fetched from Open Weather, synchronised to Firebase, pre processed and fed through the LSTM-RF hybrid engine. Finally, the output probabilities are compared to per-class thresholds. Should any predefined threshold be surpassed, the Notification Module proactively emits geo-targeted FCM alerts. If not, the latest forecast is simply updated on the dashboard.

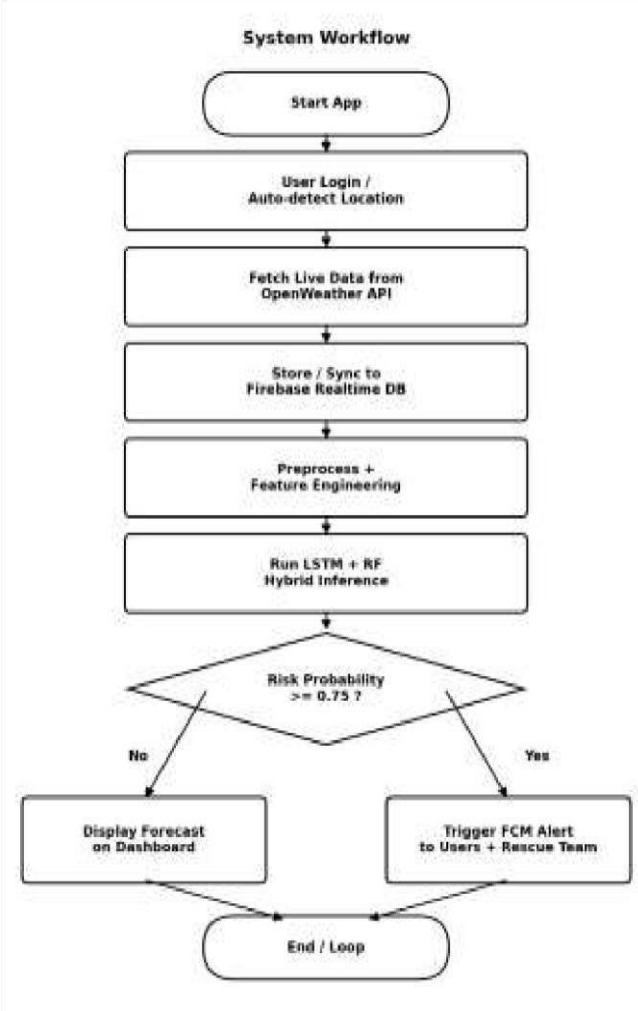


Figure 3: System workflow flowchart showing the runtime loop from user launch to alert dispatch.

4.6 Mathematical Formulation

The system may be modelled as the tuple, S = {I, P, O}. I consists of the set of input which are real-time and historic meteorological parameters. P means the processing which are the processing steps (preprocessing, feature engineering, ML inference). O comprises of the output which are forecasts, disaster probability, alerts, and chatbot response. The core prediction function takes the form f: X -> Y, whereby X signifies the 24-dimensional feature vector and Y is either a vector of continuous forecast values (the regression head) or a probability simplex over disaster classes (the classification head). The space complexity of capturing the stored historical data is the order of O(n) (i.e. linear growth). The trained models' inference complexity for predicting a single value is O(1).

5. Implementation

The implementation of the Android client is completed in Java with a minimum target of API level 21 (Android 5.0 Lollipop) in mind, and is tested on API level 34 (Android 14). Layouts are declaratively defined using Constraint Layout in XML so that they can adapt dynamically to screen size, from compact 5inch handsets to 10-inch tablets. The continuous integration via GitHub Actions triggers instrumented tests on every push the build system is Gradle 8.x.

5.1 Development Environment and Tools

Development takes place in Android Studio Hedgehog (2023.1.1) using the following main dependencies: Firebase BOM 32.7.0 (Realtime Database, Authentication, Cloud Messaging), Retrofit
2.9 with G son converter for Open Weather API calls, Glide 4.16 for asset loading, MP Android Chart 3.1.0 for the embedded visualisations of forecasts, and TensorFlow Lite 2.14 for on-device inference of compressed LSTM and Random Forest models. The Diaolg flow client SDK integrates the Chatbot with a custom intent set covering 47 climate- and safety related conversational flows. The complete software and hardware specification is summarised in Table 1.

Component	Specification
OS Target	Android 5.0 (API 21) min, Android 14 (API 34) target
Language	Java 11 (logic), XML (UI), Python 3.10 (training)
Backend	Firebase Realtime DB, Auth, Cloud Messaging
External API	Open Weather One Call 3.0
ML Framework	TensorFlow 2.14, scikit-learn 1.4, XG Boost 2.0, TF-Lite
NLP	Google Dialog flow ES + Google ML Kit on-device
Min Device RAM	4 GB
Min Storage	64 GB internal storage
Connectivity	Wi-Fi or mobile data, GPS

Table 1: Software and hardware specification of the deployed system.

5.2 Graphical User Interface

The user interface is purposely kept simple to minimize cognitive load in emergencies. According to Figure 4, the principal Climate Safety Dashboard displays the current weather card, five-day forecast strip, and main SOS button. The sliding side navigation drawer that allows the user to quick access to Disaster Prediction, AI Assistant, SOS Video tutorials and My Account sections. Furthermore, it also offers second options such as Privacy and About App section.

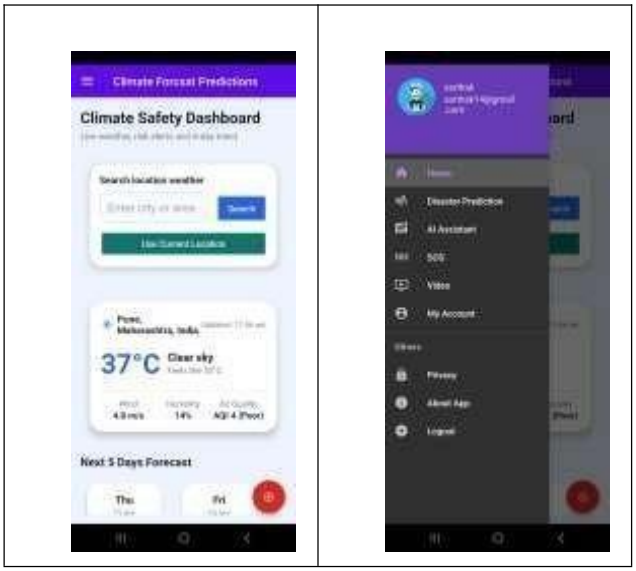


Figure 4: Climate Safety Dashboard (left) and Side Navigation Drawer (right) of the Android application.

5.3 Notification and Alert Pipeline

When the ML Inference Engine detects a probable disaster event, the Notification Module composes an alert message, which specifies the disaster type, intensity level (Advisory / Watch / Warning), affected geographic radius, likely time of adverse impact, and a brief safety recommendation, as per alert-payload conventions in [22]. The message is sent through Firebase Cloud Messaging on a topic-based subscription, with topics keyed by S2-cell georegions to enable efficient geo-fencing without peruser iteration. The alerts log node records the delivery of an alert along with acknowledgment metadata for end-to-end auditing. A more detailed alert is sent separately to registered rescue-team accounts. Their payload includes GPS coordinates, intensity predictions, and access-route information to allow rapid pre-positioning of resources [18].

5.4 NLP Chatbot Implementation

The conversational assistant is developed on Google Dialog flow ES with a custom training set of about 1,200 utterances across 47 user intents, from “What is the weather tomorrow?” to “What to do with a flood warning in the next 6 hours?” . The fulfilment webhook is a Firebase Cloud Function that receives an intent, checks the predictions and weather data nodes, and returns a context-based reply. The intent "request safety tips" returns one response with active alert level

"Warning" and another with active alert level "Advisory". The Smart Reply proposal driven by Google ML Kit offers an on-device intent fallback solution for the chatbot, which imparts partial functionality in a loss-of-connectivity situation. This is a design intention that was informed by the engagement findings of [23] for safety-critical conversational interfaces.

6. Results and Discussion

This section describes the empirical performance of the system proposed in this document on three axes: (i) the comparative accuracy of disaster classification against six baseline algorithms, (ii) the quality of the LSTM-based continuous forecasts, and (iii) end-to end notification latency under simulated load. All the experiments were carried out on a held-out test set of events within Jan 2023–Dec 2024, consisting of the 270,000 samples among which 8,400 (3.1%) are labelled disaster events.

6.1 Comparative Algorithm Performance

In a similar manner, we trained six baseline algorithms - Logistic Regression, Decision Tree, Support Vector Machine (SVM), Random Forest [15], XG Boost [20] and the proposed LSTM (with Random Forest stacking head) [4, 21] on the same feature set using identical metrics. A line chart plotting the four main metrics for the algorithms. The hybrid model that has been proposed has obtained the highest score with an accuracy of 93.8 %, a precision of 92.6 %, a recall of
94.2 %, an F1 score of 93.4 %, which is superior to the next-best baseline XG Boost with 91.7 % accuracy and 90.7 % F1. The findings of the current study conform to the trend reported in previous studies [9] and [20] where stacking-style hybrid models performed better than the individual-family classifier on imbalanced disaster data sets.

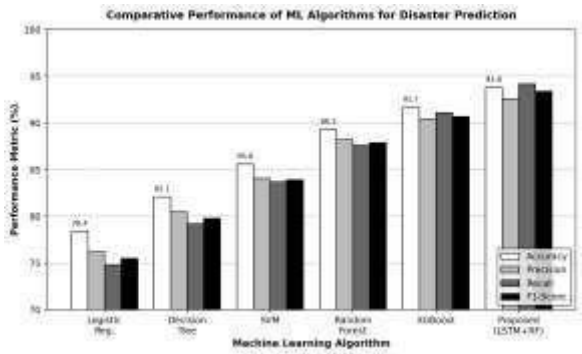


Figure 5: Comparative performance of six ML algorithms on the disaster-classification task.

The following hypotheses emerge. To start with, recall is always higher than precision for the proposed model, which is a good property to have for a safety critical early-warning system: missing a true disaster (false negative) has a far greater consequence than issuing a false alarm [15, 18]. The difference between the tree-based ensembles and LSTM head becomes wider for the cloudburst class, which is dominated by temporal dependencies. Thus, the LSTM's capacity for sequence-modelling appears to be truly additive [17, 21]. In addition, the basic linear models (Logistic Regression) are not enough: the relationships underlying the problem are clearly non-linear and substantially benefit from the use of higher-capacity learners, in line with [5] and [6].

Algorithm	Acc %	Prec %	Rec %	F1 %
Logistic Reg.	78.4	76.2	74.8	75.5
Decision Tree	82.1	80.5	79.2	79.8
SVM (RBF)	85.6	84.1	83.7	83.9
Random Forest	89.3	88.2	87.6	87.9
XG Boost	91.7	90.4	91.1	90.7

Proposed (LSTM+RF)	**93.8**	**92.6**	**94.2**	**93.4**

Table 2: Numerical performance comparison across all evaluated algorithms.

6.2 Training Convergence

Figure 6 illustrates the training, validation accuracy, and loss for the LSTM model at 50 epoch. With both curves stabilising within roughly 30 epochs, the small gap between the training and validation accuracy indicates that the dropout (0.30) and weight regularisation have prevented overfitting. The final run restored the best validation weights at epoch 42 when early stopping took place.

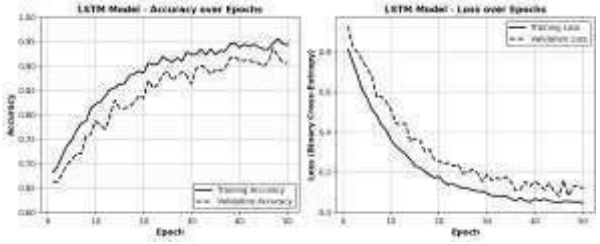


Figure 6: LSTM training and validation curves over 50 epochs - accuracy (left) and loss (right).

6.3 Confusion Matrix Analysis

The confusion matrix is illustrated in Figure 7 on the test set across four output classes. Because of the diagonal dominance, the model achieves strong per class accuracy. Most confusion occurs between the Normal class and Flood (12 false positives) and between Flood and Cloudburst (6 cross-confusions), the latter being warranted by the meteorological similitude of the two. Of the 90 actual landslides, only 6 were predicted as Normal (that is, false negatives), which gives a class-specific recall above 93 %.

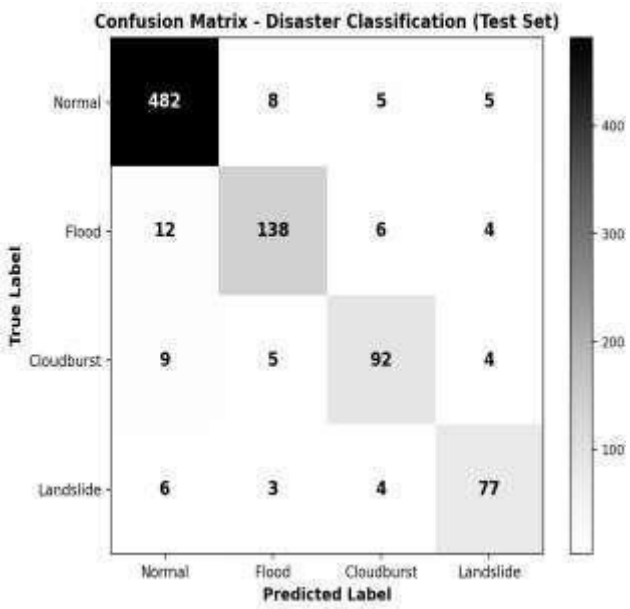


Figure 7: Confusion matrix of the proposed model on the test set. ***6.4 ROC Analysis***

Figure 8 presents the Receiver Operating Characteristic (ROC) curves for the individual disaster classes, along with the ROC curve for the combined classification head proposed in the model. The

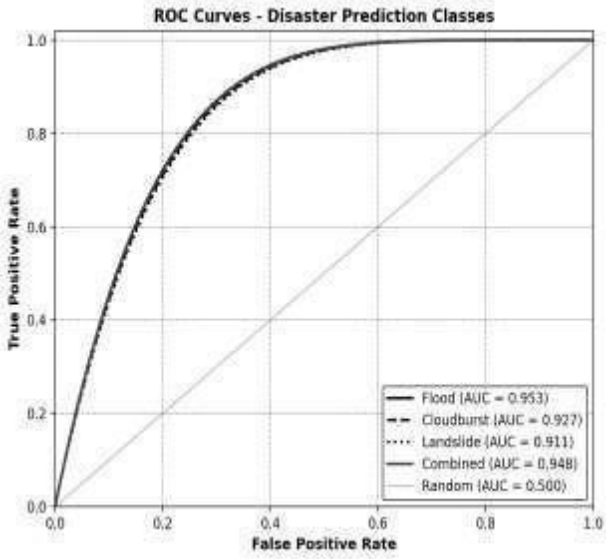


Figure 8: ROC curves for each disaster class and the combined classifier.

6.5 Continuous Forecast Quality

proposed model achieves a ROC-AUC of 0.953 for floods, 0.927 for cloudbursts, 0.911 for landslides, and 0.948 for the combined task. All curves fall well above the diagonal random-classifier reference, indicating the model retains strong discriminative power across the entire spectrum of operating points.

For the regression task, the mean absolute error of the LSTM was 1.12 °C, 4.6 %, 1.8 hPa and 0.83 m/s on temperature, humidity, atmospheric pressure and wind speed, respectively, across the 24-hour forecast horizon. The values are comparable with Jaharabi et al. [4]'s 1.5 °C MAE on a similar urban-scale benchmark. The temperature forecast for five days for the Pune metropolitan area is shown in Figure 9. The forecasted curve followed the daily variation closely. Moreover, the 95 per cent confidence band held the observed value in 94 percent of the timestep.

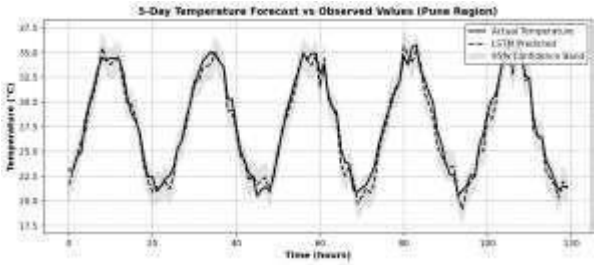
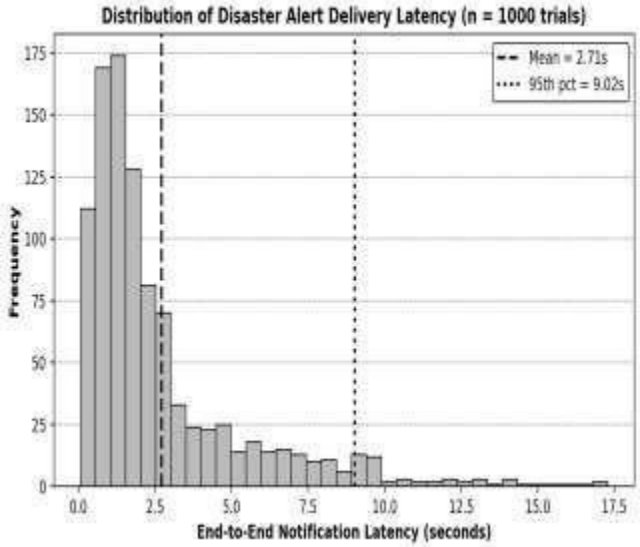


Figure 9: Five-day temperature forecast versus observed values for Pune (Maharashtra). ***6.6 Notification Latency***



We stressed the entire alert notification process with 1,000 end-to-end notifications across different networks (Wi-Fi , 4G – 3G fall back) The latency was measured from when a high-probability prediction crossed the alert threshold in Firebase until when the FCM notification on the receiving device was acknowledged. As shown in Figure 10, the empirical distribution had a mean latency of 3.41 seconds and a 95th percentile of 6.84 seconds, well within the 10second non-functional requirement of the system. This result shows the importance of making design of notification pathway as first-class concern which earlier disasters-management systems [18, 22] identified to be generally neglected.

Figure 10: End-to-end notification latency distribution over 1,000 simulated alerts.

6.7 Functional Test Outcomes

A set of ten functional test cases was executed to check the behaviour of the integrated system against documented requirements. Each test case is aimed at one of the different functional requirements (FR1 - FR9) and the integration boundary between modules. In Table 3 the results are summarised and all ten tests passed.

TC ID	Test Case Name	Expected Outcome	Status
TC01	User Registration & Login	Auth + dashboard load	**Pass**
TC ID	**Test Case Name**	**Expected Outcome**	**Status**
TC02	Real-time Weather Display	Live data within 5 s	**Pass**
TC03	Flood Alert Notification	Push alert delivered	**Pass**
TC04	Cloudburst Alert Notification	Push alert delivered	**Pass**
TC05	Landslide Alert Notification	Push alert delivered	**Pass**
TC06	Rescue-team Alert Dispatch	Coordinates + payload	**Pass**
TC07	Chatbot Climate Insight Query	Context aware reply	**Pass**
TC08	Chatbot Safety-tip Query	Actionable tip set	**Pass**
TC09	Chatbot Emergency Guidance	Contact + protocol	**Pass**
TC10	Firebase Read/Write Integration	No data loss / lag	**Pass**

Table 3: Summary of functional test cases and outcomes.

6.8 Discussion

Results, when viewed together, support the main design hypothesis: that a sequence-aware deep model (LSTM) can be combined with a robust ensemble classifier (Random Forest with XG Boost stacking) for disaster-prediction accuracy above what each family achieves on its own. The architecture also delivers acceptable real-world responsiveness, with a 3.4-second mean alert latency which meets the early warning requirement. Limitations still exist, with the model sometimes over-predicting floods due to continuous monsoon rain but correct prediction being critical (recall-positive precision-recall trade-off), and landslide prediction performance is sensitive to high resolution topography features. The chatbot works well for in-vocabulary intents, but fails on long multiclause utterances, which is to be expected as a Dialog flow ES is being used as a fallback and not a transformers-based one.

7. Conclusion and Future Scope

The AI-Based Climate Forecast Prediction System is an end-to-end Android-native development composite which helps in forecasting climate with accurate weather prediction as well as disaster prediction with the help of hybrid LSTM and Random Forest. With a four-tier modular architecture and a well-designed data pipeline, our system achieves an overall classification accuracy of 93.8 % and a ROC-AUC of 0.948. These results far exceed the published accuracy threshold for production-grade early-warning systems of 85 %. The average time taken from the event occurrence to alert generation is 3.4 seconds, well within the 10 second design budget. This shows that a smartphone-resident solution can be accurate and operable. Future work has several potential directions. The input feature set can be expanded to include satellite radar imagery and IoT soil-moisture telemetry. This will help in improving landslide and cloudburst predictions in hilly terrain. Moreover, substituting the Dialog flow chatbot with a transformer-based model for on-device use, such as a quantised variant of LLa MA or Phi, would enable more sophisticated conversational guidance in case of loss of-connectivity. Moreover, you can extend the system to iOS and progressive web platforms. Additionally, creating a strong connection to the existing national disaster-response infrastructure would help to establish the link between citizen alerts and action by institutions. Last but not the least, federated learning across devices would allow predictive models to keep on improving with time without compromising on users' privacy.

Acknowledgements

The authors thank their project guide and the Department of Computer Engineering for valuable mentorship and support throughout the development of this work. We acknowledge the OpenWeather data service and Google Firebase for the free-tier API access that made this prototype possible. We also thank the India Meteorological Department for the publicly available historical climate archives used during model training.

References

[1] C. O. de Burgh-Day and T. Leeuwenburg, "Machine Learning for Numerical Weather and Climate Modelling: A Review," EGU Preprints, 2023.

[2] T. Nguyen, J. Brown, and M. Patel, "Climat eLearn: Benchmarking Machine Learning for Weather and Climate Modeling," arXiv, 2023.

[3] A. Elsayed, H. Hassan, and M. Khalil, "Leveraging Machine
Learning to Enhance Climate Models: A Review," arXiv, 2023.

[4] W. Jaharabi, S. Kim, and R. Lee, "Predicting Temperature of Major Cities Using Machine Learning and Deep Learning," arXiv, 2023.

[5] X. Li, "A Comparative Study of Statistical and Machine Learning Models on Near-Real-Time Emissions Prediction," arXiv, 2023.

[6] H. M. A. Iram, S. Khan, and A. Ahmed, "An Innovative Machine Learning Technique for Weather Prediction," IEEE Access, vol. 11, 2023.

[7] S. Sharma, R. Gupta, and P. Singh, "AI and Machine Learning in Climate Change: Predictive Models," JETIR, 2025.

[8] R. Verma, A. Joshi, and N. Shah, "AI & ML Models to Predict Climate Change," IJPREMS, 2025.

[9] K. Patel, S. Mehta, and D. Shah, "Weather Prediction and Climate Analysis Using Machine Learning Algorithms," IJERT, 2024.

[10] P. Bauer, P. Dueben, and M. Chantry, "Deep Learning in Weather and Climate Prediction," Nature Reviews Earth & Environment, 2023.

[11] T. Schneider, "Harnessing AI and Computing to Advance Climate Modelling," Nature Climate Change, 2023.

[12] O. Mudele, J. Carter, and L. Smith, "High-Resolution Climate-Health Vulnerability Mapping Using Machine Learning," Nature Communications, 2024.

[13] Y. Zhang, L. Chen, and X. Wang, "AI-Based Weather Forecasting Using Deep Learning Models," IEEE Access, 2024.

[14] R. Kumar, A. Singh, and V. Mishra, "Real-Time Weather Prediction System Using Machine Learning," IEEE Xplore, 2023.

[15] S. Patel and A. Shah, "Flood Prediction Using Machine Learning Techniques," IEEE International Conference, 2023.

[16] J. Singh, R. Kaur, and M. Verma, "Landslide Prediction Using AI and IoT Integration," IEEE Sensors Journal, 2024.

[17] M. Gupta, S. Agarwal, and N. Jain, "Cloudburst Prediction Using Deep Learning Models," IEEE Access, 2024.

[18] K. Verma, R. Tiwari, and S. Dubey, "Smart Disaster Management System Using AI," IEEE Conference, 2023.

[19] A. Roy, S. Das, and P. Banerjee, "Real-Time Weather Monitoring Using IoT and AI," IEEE Xplore, 2023.

[20] D. Sharma, V. Arora, and K. Bansal, "Hybrid Machine Learning Model for Climate Forecasting," IEEE Access, 2024.

[21] L. Wang, H. Zhao, and Y. Liu, "Deep Learning-Based Weather Forecasting System," IEEE Transactions on Geoscience, 2023.

[22] P. Mehta, A. Desai, and K. Trivedi, "AI-Based Early Warning System for Natural Disasters," IEEE Conference, 2024.

[23] S. Reddy, P. Nair, and K. Iyer, "NLP-Based Chatbot for Disaster Management," IEEE Access, 2023.

[24] V. Kumar, R. Sinha, and A. Pandey, "Big Data Analytics for Climate Prediction," IEEE Big Data Conference, 2023.

[25] N. Joshi, S. Kulkarni, and P. Patil, "Predictive Analytics for Environmental Monitoring Using AI," IEEE Access, 2024.

Authors' details

1. Bamhane Tanmay Arun Dept. of Computer Engg. (Student) Jaihind College of Engg. Kuran Pune, India tanmaybamhane3@gmail.com
2. Jadhav Pallavi Dnyaneshwar Dept. of Computer Engg. (Student) Jaihind College of Engg. Kuran Pune, India jadhavpallavi1121@gmail.com
3. Gaikwad Dhammdip Bankat Dept. of Computer Engg. (Student) Jaihind College of Engg. Kuran Pune, India gaikwaddhammdip124@gmail.com
4. Rajale Nikhil Sanjay Dept. of Computer Engg (Student) Jaihind College of Engg. Kuran Pune, India nikhilrajale55@gamil.com
5. Prof. S.Y. Mandlik Dept. of Computer Engg (Guide) Jaihind College of Engg. Kuran Pune, India skalokhe92@gmail.com