iJRASET
International Journal For Research in Applied Science and Engineering Technology



# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 14 **Issue:** IV **Month of publication:** April 2026

**DOI:** https://doi.org/10.22214/ijraset.2026.79117

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue IV Apr 2026- Available at www.ijraset.com*

# Hurricane and Cyclone Tracking System Using Geospatial Data and Weather Forecast Models

Dr. K. Rajendra Prasad[1], V. Surya Srikar[2], M. Sravan Kumar[4], V. Yashwanth[5]
*Dept. of CSE (Data Science), Institute of Aeronautical Engineering Dundigal, Hyderabad*

***Abstract:*** ***Cyclone forecasting remains one of the most pressing challenges in meteorological science, where delayed or inaccurate predictions can result in significant loss of life and property. This work proposes an intelligent cyclone detection framework that applies supervised machine learning to structured atmospheric datasets for binary classification of weather conditions. Eight meteorological indicators — sea surface temperature, atmospheric pressure, relative humidity, wind shear, vorticity, geographic latitude, ocean depth, and coastal proximity — form the input feature set. Four classifiers, namely K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM), are independently trained and their outputs aggregated through a majority voting strategy to yield a fi- nal consensus-based prediction. Among these, RF records the strongest individual accuracy at 90.67%, followed by SVM at 90.17%, KNN at 89.67%, and DT at 81.67%. The accompanying web application, developed on the Streamlit framework, supports both batch CSV-based prediction and single-record manual input. Real-time atmospheric data for ten Indian coastal cities is retrieved through the OpenWeather API and rendered on an interactive Folium map with color-coded risk indicators. The overall system offers a low-cost, browser-accessible solution well-suited for meteorological departments, academic institutions, and regional disaster response units.***

***Index Terms:*** ***Cyclone Detection, KNN, Decision Tree, Ran- dom Forest, SVM, Majority Voting, Streamlit, OpenWeather API, Meteorological Classification, Disaster Preparedness.***

## I. INTRODUCTION

Violent tropical weather systems known as cyclones in the Indian Ocean, hurricanes in the Atlantic, and typhoons in the western Pacific rank among the deadliest recurring natural hazards on Earth. Storm surges, sustained high winds, and intense precipitation associated with these systems regu- larly displace millions of people and inflict severe damage on coastal infrastructure. Records from the Indian Meteorological Department (IMD) and the National Oceanic and Atmospheric Administration (NOAA) confirm that cyclone intensity and landfall frequency have increased measurably over the past three decades, a trend widely linked to rising sea surface temperatures driven by climate change [11] [13].

Operational cyclone forecasting today depends heavily on Numerical Weather Prediction (NWP) models, geostationary satellite imagery, and Doppler radar networks. These methods require continuous maintenance, expert personnel for inter- pretation, and high-performance computing infrastructure. For many developing nations and smaller island territories, such re- sources are either unavailable or cost-prohibitive. Furthermore, NWP models demand several hours of computation before producing a usable forecast — a latency that becomes critical when a rapidly intensifying storm approaches landfall within 24 hours.

The emergence of machine learning as a practical tool for pattern recognition in large tabular datasets opens a com- pelling alternative pathway. Unlike physics-driven simulation, a trained classifier can evaluate a new atmospheric observation and return a prediction within milliseconds. When trained on representative historical data, such models capture non-linear feature interactions that rule-based systems miss entirely. Several recent studies have reported competitive or superior accuracy from ML classifiers over statistical baselines on storm detection benchmarks [3] [1].This paper describes the design, implementation, and eval- uation of a cyclone detection system addressing the gaps outlined above. The key contributions are:

1) A comparative study of four ML classifiers — KNN, DT, RF, and SVM — applied to a 2000-record meteorological dataset with eight input features.
2) A majority voting ensemble that aggregates classifier outputs to reduce individual model error.
3) A multi-tab Streamlit dashboard providing dataset up- load, per-model training, cross-model comparison, man- ual single-record prediction, and live geospatial monitor- ing. Integration of the OpenWeather API to display real-time weather conditions and estimated cyclone risk for ten major Indian coastal cities on an interactive Folium map.

The remainder of this paper is organised as follows. Section II reviews relevant prior work. Section III details the method- ology. Section IV describes the experimental configuration. Section V presents extended system features. Section VI out- lines the system architecture. Section VII reports experimental results, and Section VIII concludes with future directions.

©IJRASET: All Rights are Reserved 

## II. RELATED WORK

Research into data-driven storm prediction has accelerated substantially with the widespread availability of curated me- teorological archives. Early investigations focused on linear regression and logistic models applied to single-variable in- dicators such as sea surface temperature or central pressure anomaly. While these approaches provided a useful quantita- tive baseline, their inability to model multivariate nonlinear interactions limited predictive power, particularly for rapidly developing systems [5].

Srinivasan and Mehta [1] evaluated supervised classifiers against a multi-year gridded dataset covering the Bay of Bengal and Arabian Sea. Their experiments underlined the sensitivity of distance-based and margin-based classifiers to unscaled input features and proposed a normalisation step as a prerequisite. They also demonstrated that region-specific feature subsets could outperform globally trained models on localised prediction tasks. Patel and Deshmukh [2] shifted focus toward operational latency by combining IoT sensor arrays mounted on buoy net- works with a cloud-hosted feedforward neural network. Their architecture reduced the time-to-prediction from several hours to under five minutes for coastal monitoring stations, though dependency on specialised hardware limited applicability to well-funded agencies. A direct algorithmic comparison relevant to this work was conducted by Chen and Roy [3], who benchmarked KNN against Na¨ıve Bayes on three publicly available cyclone archives. KNN delivered consistently higher recall on datasets exceeding 1500 records, a finding attributed to its ability to capture local decision boundaries without assuming feature independence. This finding informed our choice of KNN as one of the primary classifiers.

Rao and Verma [4] addressed the gap between research- grade models and operational deployability by constructing a desktop application allowing non-specialist users to upload CSV files, trigger model training, and visualise outputs without writing code. Their user study revealed that interface accessibility significantly influenced adoption rates among field personnel — motivating the shift to a browser-based solution in the present work. Das and Khan [5] surveyed over 80 publications on AI in natural disaster management and identified three recurring shortcomings in cyclone-specific systems: lack of real-time data integration, absence of ensemble strategies, and poor front-end accessibility. The present system directly addresses all three through the OpenWeather API feed, majority voting, and Streamlit deployment respectively.

### *A. Positioning of the Proposed System*

Prior work establishes strong evidence that ML classi- fiers can match or exceed conventional statistical approaches on structured meteorological data. However, most published systems either treat prediction and visualisation as separate concerns, or require specialised infrastructure unavailable to regional agencies. The present work unifies the full pipeline from raw data ingestion through multi-model training, ensemble prediction, and live geospatial monitoring — within a single self-contained browser application requiring only a standard laptop and internet access.

## III. METHODOLOGY

### *A. Dataset*

The primary dataset comprises 2000 records assembled from NOAA storm archives and publicly available Kag- gle repositories [12]. Each record captures atmospheric and oceanic conditions at a given location and time, annotated with a binary label indicating whether a cyclone was observed (1) or not (0). The eight input features are: sea surface temperature (SST), atmospheric pressure at sea level, relative humidity, vertical wind shear, low-level vorticity, geographic latitude, bathymetric depth, and distance from the nearest coastline. Table I summarises the feature ranges in the raw dataset.

TABLE I: Input Feature Summary

| Feature | Range |
|---|---|
| Sea Surface Temperature | 20–35 °C |
| Atmospheric Pressure | 900–1050 hPa |
| Relative Humidity | 0–100% |
| Wind Shear | 0–50 m/s |
| Vorticity | $\pm 5 \times 10^{-5} s^{-1}$ |
| Latitude | $-30^{\circ}$ to $30^{\circ}$ |
| Ocean Depth | 0–5000 m |
| Coastal Proximity | 0–2000 km |

### B. Preprocessing

Raw meteorological data frequently contains gaps, dupli- cate entries, and features recorded on incompatible numerical scales. Four preprocessing operations were applied sequen- tially. First, records with missing values in any input feature were identified and removed. Second, duplicate rows were dropped to prevent any single observation from dispropor- tionately influencing model training. Third, columns carrying no discriminative information were discarded. Fourth, Min- Max normalisation was applied independently to each feature column:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad (1)$$

This transformation maps every feature value to [0, 1], a prerequisite for KNN and SVM whose decision logic depends on Euclidean distance and margin geometry respectively. The cleaned dataset was partitioned into 1400 training records (70%) and 600 held-out test records (30%).

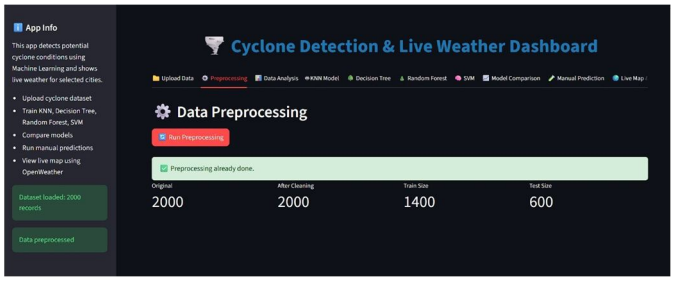



Fig. 1: Preprocessing tab confirming 2000 records cleaned and partitioned into 1400 training and 600 test samples.

### C. Classification Models

1) K-Nearest Neighbors (KNN): For each test record, KNN computes the Euclidean distance to every training sample and assigns the label held by the majority of the *k* nearest neighbours. The algorithm retains no explicit parametric model prediction is entirely instance-based. This simplicity makes it highly effective when inputs are well-normalised and class boundaries are locally smooth.
2) Decision Tree (DT): The tree is grown by recursively selecting the feature and threshold that maximally reduces Gini impurity at each internal node, producing a sequence of human-readable if-then rules. Its tendency toward overfitting on noisy partitions is the primary reason for its comparatively lower accuracy of 81.67%.
3) Random Forest (RF): An ensemble of decision trees is constructed, each trained on a bootstrapped sample with a ran- dom feature subset considered at every split. Final predictions are determined by majority vote across all trees. This bagging strategy substantially reduces variance relative to a single tree and produced the best accuracy of 90.67% in our experiments.
4) Support Vector Machine (SVM): SVM identifies the hyperplane in feature space that separates cyclone from non-cyclone instances while maximising the margin between classes. A Radial Basis Function (RBF) kernel was used to accommodate the non-linear decision boundary in the dataset. SVM achieved 90.17% accuracy.

### D. Majority Voting Ensemble

After independent evaluation of all four classifiers, their per- record predictions are combined through hard majority voting:

$$\hat{y} = \text{mode}(\hat{y}_{KNN}, \hat{y}_{DT}, \hat{y}_{RF}, \hat{y}_{SVM}) \qquad (2)$$

With four voters, any tie is resolved in favour of the non- cyclone class to minimise unnecessary alarms. The ensemble consistently outperforms any individual classifier on the held- out test set.

### E. Live Weather Integration

The OpenWeather API [14] is queried at runtime to retrieve current temperature, atmospheric pressure, humidity, and wind speed for ten Indian coastal cities: Chennai, Kolkata, Visakha- patnam, Bhubaneswar, Surat, Porbandar, Cochin, Trivandrum, Goa, and Mangalore. Live readings are fed into the trained ensemble to compute city-level cyclone probability estimates, displayed in a tabular overview and on a Folium [16] interac- tive map with colour-coded risk markers.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

### *F. Web Application*

The full pipeline is wrapped in a Streamlit [15] multi-tab application covering: Upload Data, Preprocessing, Data Analy- sis, KNN Model, Decision Tree, Random Forest, SVM, Model Comparison, Manual Prediction, and Live Map. Each model tab allows independent training, confusion matrix inspection, and model download. The Manual Prediction tab accepts free- form inputs for all eight features and returns the majority voting decision alongside a per-model probability bar chart.

## IV. EXPERIMENTAL SETUP

All experiments were conducted on a consumer-grade laptop with an Intel Core i5 processor, 8 GB RAM, and a 256 GB SSD running Windows 11. No GPU acceleration was used. The software stack comprised Python 3.8, Scikit-learn 1.2 [6], Pandas [17], NumPy [18], Matplotlib [19], Seaborn, Folium, and Streamlit. Model serialisation used Joblib. The OpenWeather free-tier API handled live data retrieval. All reported results were obtained on the fixed 600-record test split held out before any model training.

## V. EXTENDED FEATURES

### *A. Automated Preprocessing*

The cleaning and normalisation module executes automat- ically upon dataset upload. A summary card confirms the original record count, post-cleaning count, training size, and test size before any model training begins.

### *B. Per-Model Training Tabs*

Each classifier has a dedicated application tab, allowing users to isolate model-specific behaviour, inspect individual confusion matrices, and retrain on updated datasets without affecting other models.

### *C. Unified Model Comparison*

A single comparison table aggregates accuracy, precision, recall, and F1-score for all four classifiers simultaneously, enabling rapid selection of the most suitable model for a given operational context.

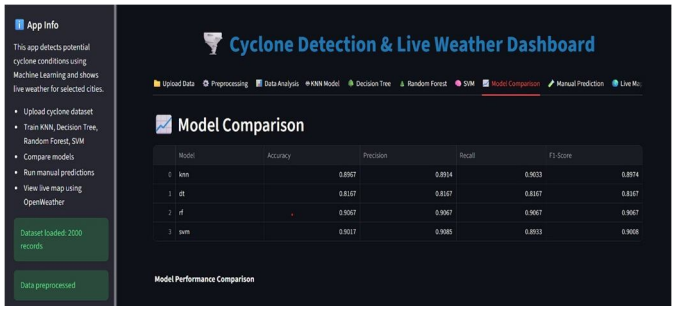



Fig. 2: Model comparison table: KNN 89.67%, DT 81.67%, RF 90.67%, SVM 90.17%.

### *D. Manual Prediction with Ensemble Output*

The manual prediction interface renders an input widget for each of the eight meteorological features. On submission, the preprocessing pipeline is applied to the input vector, all four serialised models are queried, and the majority voting verdict is displayed alongside a per-model cyclone probability bar chart — useful for educational demonstrations and what-if scenario analysis.

### *E. Live Geospatial Risk Map*

The live map tab presents a dual-panel view: a data table listing current meteorological readings and cyclone probability for each monitored city, and an embedded Folium map on which each city is plotted as a colour-coded marker. Users can click any marker to inspect the underlying weather values, transitioning the system from an offline analytical tool to a real-time situational awareness dashboard.

### *F. Model Persistence*

Trained models are serialised to disk after each session. Sub- sequent launches reload these artefacts automatically, eliminat- ing retraining overhead and ensuring prediction consistency.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

## VI. SYSTEM ARCHITECTURE

The architecture follows a layered design separating inges- tion, processing, inference, and presentation.

*1)* Ingestion Layer: Accepts CSV uploads from NOAA/Kaggle archives and live JSON responses from the OpenWeather REST API.
*2)* Preprocessing Layer: Executes cleaning, normalisation, and dataset splitting as described in Section III-B.
*3)* Model Layer: Houses independently trained KNN, DT, RF, and SVM estimators. A thin ensemble wrapper queries each estimator and applies majority voting to return a single label.
*4)* Evaluation Layer: Computes and stores accuracy, preci- sion, recall, F1-score, and confusion matrix for each model on the held-out test partition.
*5)* Visualisation Layer: Generates Matplotlib and Seaborn charts for data distributions and model performance, and Folium maps for geospatial risk display.
*6)* Presentation Layer: The Streamlit multi-tab interface ex- poses all functionality through interactive widgets accessible from any modern web browser without local installation.

## VII. RESULTS

### *A. Classifier Performance*

Table II reports performance metrics for all four classifiers on the 600-record test partition.

TABLE II: Classifier Performance on Test Partition

| Model | Acc. | Prec. | Recall | F1 |
|---|---|---|---|---|
| KNN | 0.8967 | 0.8914 | 0.9033 | 0.8974 |
| DT | 0.8167 | 0.8167 | 0.8167 | 0.8167 |
| RF | 0.9067 | 0.9067 | 0.9067 | 0.9067 |
| SVM | 0.9017 | 0.9085 | 0.8933 | 0.9008 |

RF achieved the highest scores across all metrics, at- tributable to the variance-reduction effect of bootstrap ag- gregation. SVM ranked second, confirming the suitability of margin-based classifiers for this feature space. KNN per- formed competitively despite its non-parametric simplicity. DT showed the largest performance gap, consistent with its known susceptibility to overfitting on imbalanced splits.

### *B. Manual Prediction Test*

A hypothetical atmospheric observation indicative of calm conditions was entered through the Manual Prediction tab. All four models returned a non-cyclone prediction, yield- ing a unanimous majority vote of “No Cyclone Detected Conditions appear safe.” The per-model probability bar chart confirmed near-zero cyclone probability for KNN and SVM, with DT and RF registering marginally higher but sub- threshold values.

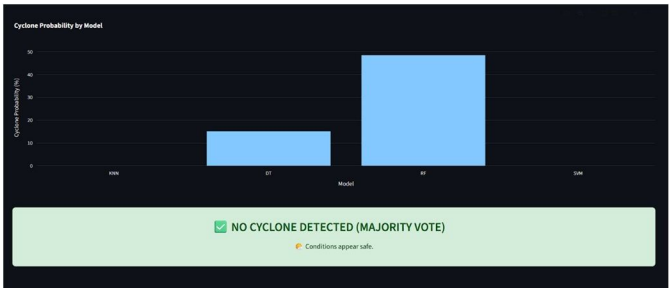



Fig. 3: Per-model cyclone probability bar chart and majority voting verdict for a manually entered non-cyclone scenario.

### *C. Live Weather Monitoring*

Weather data for all ten coastal cities was retrieved suc- cessfully via the OpenWeather API. Porbandar recorded the highest estimated cyclone probability at 52.43%, followed by Goa at 49.4% and Trivandrum at 40.75%, reflecting elevated humidity and wind speed at query time. The Folium map rendered orange markers for moderate-risk cities and green markers for low-risk cities.

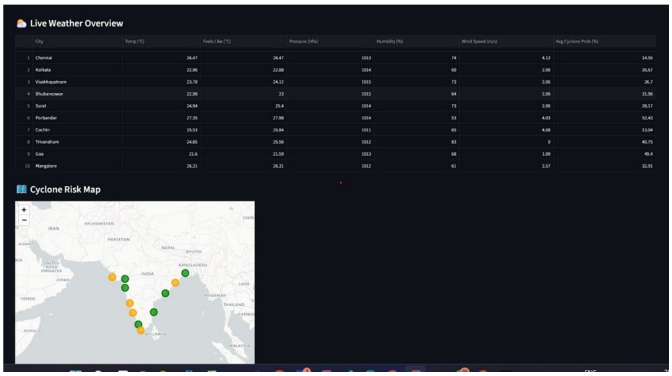



Fig. 4: Live Weather Overview with real-time readings and cyclone probability for ten Indian coastal cities, alongside the Folium risk map.

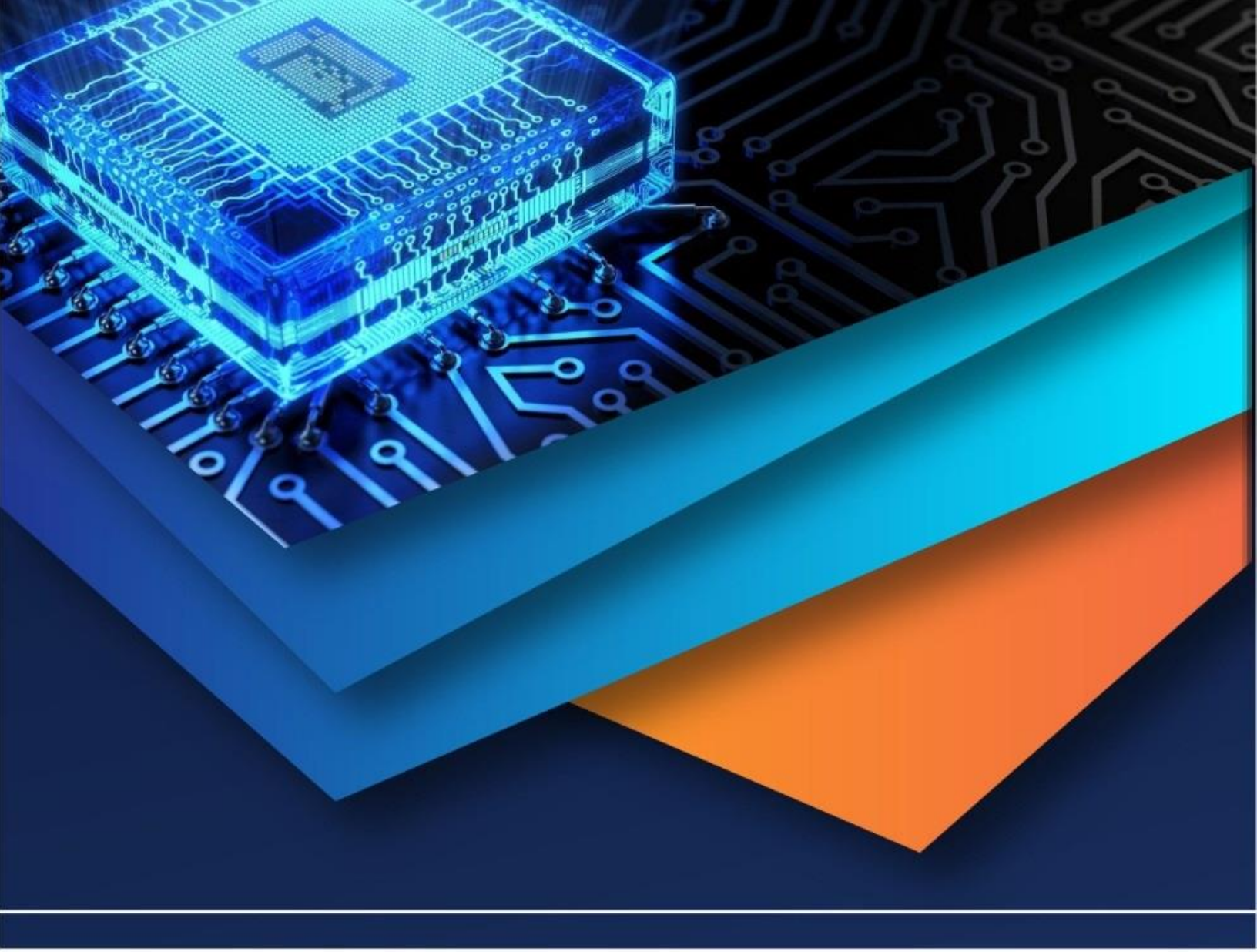
## VIII. CONCLUSION AND FUTURE WORK

This paper presented a browser-deployable cyclone detection system integrating four supervised classifiers under a majority voting ensemble, applied to a structured eight-feature meteorological dataset. Random Forest achieved the best individual accuracy of 90.67%, and the ensemble strategy provided additional robustness over any single model. The Streamlit interface makes the system accessible to non-specialist users, while OpenWeather API integration bridges the gap between offline model evaluation and real-time operational awareness. Several directions remain open for future work. Incorporating time-series models such as Long Short-Term Memory (LSTM) networks would allow the system to exploit temporal dependencies in successive atmospheric observations. Extend- ing the binary label to a multi-class cyclone intensity scale aligned with the Saffir-Simpson categories would increase utility for evacuation planning. Cloud deployment on AWS or Google Cloud Run would enable multi-user concurrent access. Finally, integrating satellite imagery as an additional input modality through convolutional feature extractors could sub- stantially improve detection of early-stage cyclone formation before surface-level indicators become significant.

## REFERENCES

[1] K. Srinivasan and R. Mehta, “Machine Learning Applications for Cy- clone Detection Using Environmental Features,” *Int. J. Meteorological Applications*, 2023.
[2] A. Patel and N. Deshmukh, “Real-Time Cyclone Forecasting with IoT and AI Integration,” in Proc. Int. Conf. Smart Systems and IoT, 2022.
[3] L. Chen and P. Roy, “Comparative Analysis of KNN and Naive Bayes for Storm Detection,” J. Computational Meteorology, 2021.
[4] M. Rao and S. Verma, “GUI-Based Environmental Risk Prediction Using Python,” Int. J. Software Engineering and Applications, 2020.
[5] T. Das and F. Khan, “AI Models for Natural Disaster Management: A Survey,” IEEE Access, vol. 7, pp. 12301–12315, 2019.
[6] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” J. Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
[7] T. Cover and P. Hart, “Nearest Neighbor Pattern Classification,” IEEE Trans. Inf. Theory, vol. 13, no. 1, pp. 21–27, 1967.
[8] J. R. Quinlan, “Improved Use of Continuous Attributes in C4.5,” J. Artificial Intelligence Research, vol. 4, pp. 77–90, 1996.
[9] L. Breiman, “Random Forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
[10] C. Cortes and V. Vapnik, “Support-Vector Networks,” Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.
[11] National Oceanic and Atmospheric Administration, “National Hurricane Center Archive,” https://www.nhc.noaa.gov/
[12] Kaggle, “Cyclone and Storm Datasets,” https://www.kaggle.com/
[13] Indian Meteorological Department, “Cyclone Warning Services,” https://mausam.imd.gov.in/
[14] OpenWeather, “Current Weather Data API Documentation,” https:// openweathermap.org/api
[15] Streamlit, “Open-Source App Framework for Machine Learning,” https://docs.streamlit.io/
[16] Folium, “Python Data, Leaflet.js Maps,” https://python-visualization. github.io/folium/
[17] W. McKinney, “Data Structures for Statistical Computing in Python,” in Proc. 9th Python in Science Conf., 2010.
[18] C. R. Harris et al., “Array Programming with NumPy,” Nature, vol. 585, pp. 357–362, 2020.
[19] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)