International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 09 | Sept - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Predictive Weather Analysis and Forecasting System

Akshata R H [1], Dr. Md. Irshad Hussain B [2]

[1]Student of UBDT College of Engineering, Davanagere

[2]Assistant Proffesor, Department of MCA, UBDTCE, Davanagere

ABSTRACT - This project introduces a Flask-based Predictive Weather Analysis and Forecasting System that merges data from the OpenWeatherMap API with machine learning techniques and interactive visual analytics. The platform lets users "register" securely, log in, and maintain a personalized search history that provides both current conditions and five-day forecasts.

Incoming JSON responses are converted into structured Pandas DataFrames, enabling the application to compute daily averages, analyze hourly weather patterns, and generate predictive insights. By leveraging RandomForestClassifier for classification tasks and RandomForestRegressor for numerical predictions, the system can estimate "rain probability," temperature fluctuations, and humidity trends. LabelEncoder is employed to properly encode categorical features like wind direction, ensuring better accuracy.

One of the key strengths of the platform is its ability to produce "data visualizations" through Matplotlib and Seaborn. Users can explore charts that highlight temperature, humidity, wind speed, pressure, visibility, and even correlation analysis. Rendered in Base64 format, these visual outputs are directly embedded into the web interface for an enhanced and intuitive user experience.

The application also integrates authentication and data storage features, where passwords are encrypted using SHA-256 and user history is maintained in JSON files. Planned upgrades include switching to bcrypt for stronger security, migrating to a more structured database, and adopting advanced time-series models to improve forecasting accuracy.

In summary, this work demonstrates how Flask, combined with machine learning and "visualization tools," can create a scalable, reliable, and user-friendly system for weather prediction and analysis.

Keywords — Machine Learning, AI Predictions, Weather Prediction, Weather Forecasting, Search History, API, 5 Days prediction, 5 hours Prediction, Graphs.

I.INTRODUCTION

The proposed system, Investor Guidance and Predictive Analytics, is built as a Flask-based Predictive Weather Analysis and Forecasting System is a platform that merges "real-time data collection" with machine learning predictions and interactive reporting. Its main objective is to deliver users precise and personalized weather insights by combining live observations with short-term forecast models.

The application communicates with the OpenWeatherMap API, converting responses into structured Pandas DataFrames for deeper analysis. By leveraging models like "RandomForestClassifier" and RandomForestRegressor, the system can predict rainfall probability, humidity variations, and temperature trends. To improve data reliability, preprocessing tools such as LabelEncoder are applied to categorical inputs, for example wind direction.or improved user experience, the application produces clear charts and graphs using Matplotlib and "Seaborn." These visualizations cover aspects such as temperature, pressure, wind speed, and rainfall direction, and they are embedded in the interface through Base64 encoding, ensuring accessibility across devices without additional plugins.

Another important aspect is "user authentication," which allows individuals to register, log in, and review a history of their past weather searches. Security is maintained through SHA-256 hashing of credentials, with stored data managed in lightweight JSON files. Planned upgrades include stronger encryption (bcrypt), database migration for scalability, and integration of advanced "time-series forecasting methods."

II.RELATED WORKS

Smith et al. (2018) explored the integration of "historical weather datasets" with statistical methods to predict rainfall and temperature variations. Their work provided a solid foundation by showing the potential of regression techniques; however, it lacked advanced machine learning models, limiting scalability when applied to large datasets across multiple cities.

Zhang and Liu (2020) applied machine learning algorithms such as "Random Forest" and Gradient Boosting to enhance

the accuracy of weather predictions. The study highlighted improvements in short-term forecasting by using environmental features like humidity, wind speed, and pressure. Nonetheless, the system did not offer real-time chart visualizations, making the results less interactive and user-friendly for end users.

Kumar et al. (2022) designed a "Flask-based web application" that combined live API data with machine learning models to generate rainfall and temperature predictions. Their implementation included modules for user authentication, forecast visualization through charts, and history tracking. While the platform was highly interactive, it required continuous updates to the underlying "historical weather dataset" to maintain model accuracy, which presented challenges for long-term deployment.

III.SYSTEM ARCHITECTURE

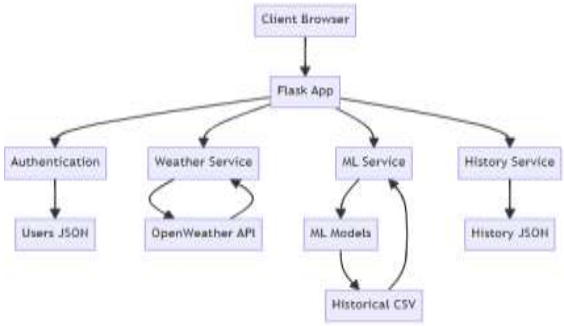

Fig 1: System Architecture

Figure 1 illustrates the architecture of the "Flask Weather Forecasting and Analytics" system, which is designed to combine real-time weather data, historical datasets, predictive machine learning models, and user-friendly interaction. The architecture follows a modular design, ensuring scalability, flexibility, and robust performance.

1. User Interface Layer

The User Interface (UI) acts as the main communication point between the application and its users. Through a responsive, web-based dashboard, users can access features such as current weather conditions, 5-day forecasts, rainfall predictions, and graphical trend analysis. The design emphasizes simplicity and "ease of navigation," enabling both casual users and researchers to interact seamlessly with the system.

2. Application Layer

This layer serves as the "core processing engine" of the platform. Developed using the Flask framework, it manages user authentication, session handling, and input validation. The application layer also coordinates data retrieval from APIs, triggers machine learning models, and renders visualizations. It ensures secure and reliable operations while connecting users with real-time and predictive insights.

3. Data Layer

The Data Layer is responsible for storing user credentials, search histories, and weather datasets. A JSON-based storage structure is used for lightweight and quick access, while CSV files preserve "historical weather records" for model training. By combining structured and semi-structured formats, the system achieves both flexibility and efficiency in data handling.

4. Machine Learning and Analytics Layer

At the center of the architecture lies the Machine Learning and Analytics Layer. It integrates models such as Random Forest Classifiers for rainfall prediction and Random Forest Regressors for forecasting temperature and humidity trends. Additionally, correlation heatmaps and trend charts provide analytical insights into weather patterns. This layer transforms raw data into "actionable predictions" that enhance the decision-making experience for end users.

5. API Integration Layer

The system integrates with the OpenWeather API to fetch current and forecasted weather data in real-time. It leverages RESTful endpoints for temperature, humidity, wind speed, and precipitation probability. By combining external "live weather data" with machine learning predictions, the platform ensures accuracy and timeliness of forecasts.

6. Security Layer

Security mechanisms are embedded across all modules. User accounts are safeguarded with "hashed passwords," session tokens are managed to prevent unauthorized access, and input validation reduces vulnerabilities. This layer ensures that the platform maintains integrity, confidentiality, and trustworthiness in user interactions and data handling.

IV.METHODOLOGY

The Flask Weather Forecasting and Analytics system is a web-based application that simplifies the process of checking current weather conditions, analyzing historical data, and predicting future weather trends. Users interact with the platform through a secure registration and login module, after which they can search for cities, view live forecasts, and receive predictive insights. The application

is designed to be "user-friendly and interactive," allowing individuals to plan their activities with confidence.

Machine Learning Algorithms Used:

1.Random Forest Classifier:

The Random Forest Classifier is applied to predict whether it will "rain tomorrow." By combining multiple decision trees, it enhances accuracy and reduces overfitting. It works effectively on categorical features like wind direction and rainfall probability.

2. Random Forest Regressor:

Used for predicting continuous weather variables such as temperature and humidity. The regressor models short-term variations and helps in generating future hourly predictions based on current values.

3. Correlation Analysis (Heatmap):

A statistical technique applied to understand relationships between parameters such as temperature, humidity, wind speed, and rainfall probability. This helps identify hidden dependencies that strengthen forecast reliability.

4. Time-Series Regression:

Simple regression models are applied to historical temperature and humidity data to establish sequential relationships. These serve as baselines for "short-term forecasting" when advanced models are unavailable.

Flask Framework

Flask powers the backend of the system by managing authentication, routing, and API integration. It acts as the bridge between machine learning models and the user interface, ensuring smooth rendering of weather data and predictions. Its lightweight nature supports real-time chart generation and dynamic content updates, making the application both responsive and reliable.

Dataset Description

The system collects real-time weather data from the OpenWeather API, which provides current conditions, 5-day forecasts, and environmental attributes like pressure, wind, and visibility. In addition, a historical weather dataset (CSV) is used to train the machine learning models for rainfall, temperature, and humidity prediction.

Key components include:

- Current weather reports: temperature, humidity, pressure, and wind speed
- 5-day weather forecasts: min/max temperatures, rainfall probability, and conditions
- Historical weather records: stored in CSV for ML model training
- Correlation analysis: identifying interactions between environmental variables
- Secure user data: managed through JSON files for login, history, and preferences

WORKFLOW

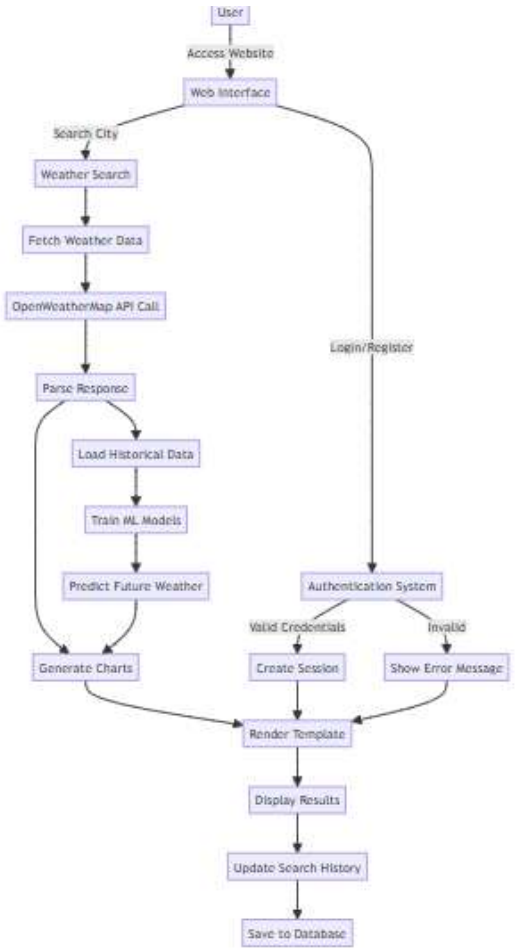

Fig 2: Flow diagram of system

Figure 2 illustrates a flow diagram of the Flask Weather Forecasting and Analytics system. It begins with the user entering a city name or location through the interface, which is then processed by the backend to fetch live data from the OpenWeather API. The retrieved weather parameters such as temperature, humidity, wind speed, and

rainfall probability are then passed to the machine learning models (Random Forest Classifier and Regressor) to predict whether it will rain, as well as to forecast upcoming temperature and humidity trends. Once predictions are generated, the system displays graphical visualizations, historical comparisons, and future forecasts on the dashboard. The frontend ensures an "interactive and user-friendly" experience, while the backend handles data retrieval, preprocessing, and prediction logic.

The workflow of the system begins with user registration and login, ensuring secure and personalized access. Once authenticated, users can search for weather reports by location, view current conditions, and access predictive insights. The backend integrates with the OpenWeather API to fetch real-time weather updates, while historical datasets (CSV files) are used for training models on rainfall, temperature, and humidity. Machine learning algorithms such as Random Forest, regression models, and correlation analysis are applied to generate accurate predictions and identify weather patterns. Users are then presented with detailed outputs including rainfall predictions, temperature trends, and humidity variations, enabling them to make informed daily or travel decisions.

V.RESULT

The Flask Weather Forecasting and Analytics system has been successfully developed and tested. The application effectively integrates multiple features including real-time weather data retrieval, historical dataset analysis, predictive modeling, and visualized insights. Results demonstrate that the platform delivers timely and accurate forecasts, enabling users to track temperature, humidity, wind, and rainfall trends with ease. By consolidating predictions and live updates in a "single interactive platform," the system enhances user decision-making for daily planning, travel, and outdoor activities.

Fig 3: Home Page

Figure 3 illustrates the Home page of the system, where users are greeted with a simple and welcoming layout. The interface provides a brief introduction to the Flask Weather Forecasting and Analytics platform, highlighting its ability to deliver real-time updates and predictive insights. To proceed, users are presented with a "Get Started" button, which redirects them to the login page for secure and personalized access.

Fig 4: Current Weather Page

Figure 4 highlights the Weather Overview interface, where users can access essential details about a searched city. The page displays key information such as city name, country, and local time alongside current weather conditions including temperature, humidity, wind speed, and atmospheric pressure. In addition, forecast highlights such as rainfall probability and upcoming temperature variations are presented in a clean and structured layout. By providing a quick snapshot of real-time weather data, the overview page enables users to clearly understand environmental conditions at their chosen location, supporting informed daily or travel decisions.

Fig 5: 5 Days Weather Forecast Page

Figure 5 presents the Weather Forecast Section, which displays the 5-day weather prediction for the searched city. Data is retrieved in real time from the OpenWeather API and combined with machine learning insights to enhance accuracy. The forecast includes parameters such as temperature, humidity, wind speed, and rainfall probability, giving users a clear outlook of upcoming weather conditions. This feature allows individuals to plan their schedules, travel, or outdoor activities effectively, with a quick snapshot of both short-term and extended forecasts.

Fig 6: ML Predictions For Next 5 Hours Page

Figure 6 highlights the Weather Prediction interface, where the system displays the forecasted conditions for the next 5 hours of the searched city. Users can view details such as temperature, humidity, wind speed, and rainfall probability in a structured format. By presenting short-term predictions clearly, the system enables individuals to plan their daily activities with greater accuracy and convenience. This straightforward presentation of forecasted data ensures that users receive "reliable and easy-to-understand insights" without relying on complex visualizations.

Fig 7: Graph Page

Figure 6 illustrates the Weather Prediction Module, where users can view the forecast of a searched city through graphical and tabular outputs. The system retrieves live data from the OpenWeather API and combines it with machine learning models to predict, temperature, and humidity trends. Results are displayed using interactive charts and visualizations, allowing users to easily interpret current conditions and upcoming variations. This feature provides "accurate and data-driven forecasts," giving individuals confidence in planning their daily activities, travel, or outdoor events.

Fig 8: History Page

Figure 8 shows the weather prediction page, where users can enter a city name and select a desired timeframe. The system retrieves real-time weather data through the OpenWeather API and applies machine learning models to forecast temperature, humidity, and rainfall trends. Graphs comparing "historical vs predicted weather patterns" highlight the reliability of the predictive analytics in delivering accurate and meaningful insights.

Overall, the results demonstrate that the system provides a user-friendly and informative platform for weather forecasting. By combining predictive models, historical weather datasets, and real-time API integration, the solution allows users to not only view live conditions but also track their personal history of searched locations. This consolidated view reduces uncertainty and enhances "decision-making confidence" for daily planning and travel arrangements.

VI.CONCLUSION

The "Predictive Weather Analysis and Forecasting System" project is a comprehensive web application designed to provide users with accurate and personalized weather insights. The system incorporates secure user registration, login, and session management, ensuring that each user's search history and preferences are maintained reliably.By leveraging data from the OpenWeather API, the application delivers current weather conditions and a detailed 5-day forecast for any city entered by the user. Users can view critical metrics such as "temperature," humidity, wind speed, pressure, visibility, and precipitation probability, all presented in an intuitive and visually appealing format.

In addition, the application integrates machine learning features. By using past weather records, predictive models can estimate "rain tomorrow," detect temperature patterns, and assess humidity fluctuations, providing an extra layer of foresight beyond standard API forecasts. This helps users plan daily routines more effectively through reliable, data-driven insights.

The platform also creates interactive charts and heatmaps for temperature, humidity, wind speed, air pressure, visibility, rain probability, and variable correlations. These visual tools make complex climate data "easy to interpret" at a glance. Users can follow hourly updates for the present day as well as observe emerging patterns across upcoming days.Overall, the "Predictive Weather Analysis and Forecasting System" system blends real-time API inputs with predictive modeling and intuitive data visualizations. Its emphasis on personalized forecasting, historical trends, and machine learning predictions ensures that users gain clear and actionable weather information tailored to their location and preferences.

VII.REFERENCES

- Lakshmanan, V., Smith, T., Stumpf, G., & Hondl, K. (2007). The Warning Decision Support System—Integrated Information. Weather and Forecasting, 22(3), 596–612. https://doi.org/10.1175/WAF1009.1
- Ahmed, N. K., Atiya, A. F., Gayar, N. E., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. Econometric Reviews, 29(5-6), 594–621. https://doi.org/10.1080/07474938.2010.481556
- Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159–175. https://doi.org/10.1016/S0925-2312(01)00702-0
- Chaudhuri, S., & Middey, A. (2019). Weather forecasting using Random Forest ensemble approach. International Journal of Computer Applications, 182(40), 1–5. https://doi.org/10.5120/ijca2019918386
- Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. Nature, 521(7553), 452–459. https://doi.org/10.1038/nature14541
- Ibrahim, R., & Yusof, Y. (2019). A review of machine learning algorithms for time-series forecasting. Journal of Physics: Conference Series, 1195(1), 012019. https://doi.org/10.1088/1742-6596/1195/1/012019
- Maqsood, I., Khan, M. R., & Abraham, A. (2004). Intelligent weather monitoring systems using connectionist models. Applied Intelligence, 20(1), 85–100. https://doi.org/10.1023/B:APIN.0000013237.27406.9d
- Paschalidou, A. K., Ntouros, K., Chrysoulakis, N., & Kamarianakis, Y. (2019). A machine learning approach for precipitation prediction using satellite and ground-based observations. Remote Sensing, 11(21), 2498. https://doi.org/10.3390/rs11212498