**Thundercast: AI-Powered Weather Intelligence Platform**

**1: Introduction and Project Overview**

**1.1 Introduction**

Thundercast is an AI-powered weather intelligence platform designed to deliver fast, actionable predictions for thunderstorms and gale-force winds. Unpredictable and localized weather events pose significant risks to critical sectors like aviation, marine operations, urban infrastructure, and public safety. Traditional forecasting methods often lack the fine spatial and temporal granularity required for effective early warnings. Thundercast addresses this challenge by leveraging advanced machine learning models and a modern web interface to transform complex meteorological data into clear, reliable risk insights.

**1.2 Project Goal**

The primary goal of this project is to develop a robust, end-to-end AI/ML system that accurately predicts the occurrence of thunderstorms and the speed of gales. The system integrates a powerful Python/Flask backend with a responsive React frontend, providing a seamless user experience for monitoring, predicting, and acting upon severe weather threats.

**1.3 Target Audience**

This documentation is intended for a diverse audience, including developers, data scientists, project managers, and system administrators involved in the design, implementation, and maintenance of the Thundercast platform.

**2: System Architecture**

**2.1 High-Level Architecture**

The Thundercast platform is composed of two primary components: a robust machine learning backend and a user-centric, interactive frontend. Data flows from various meteorological sources through a sophisticated pipeline for preprocessing, model training, and inference. The results are then exposed via a clean API to power the user-facing application.

**2.2 System Flow**

The system operates on a clear, staged flow:

1. **Data Ingestion**: Historical and real-time weather data are collected from multiple sources.
2. **ETL & Feature Store**: Data is cleaned, transformed, and engineered into a feature store.
3. **Model Training**: Machine learning models for thunderstorms and gale speed are trained and tuned.
4. **Model Registry**: Trained models are versioned and stored for deployment.
5. **Inference**: Batch or streaming inference is performed to generate predictions.
6. **Visualization & Alerts**: Predictions are delivered to the frontend for visualization and used to trigger alerts.

**2.3 Component Breakdown**

* **Backend**: Built with Python/Flask, responsible for data handling, machine learning inference, and exposing API endpoints.
* **Frontend**: A premium React application that provides a real-time, interactive map experience.
* **ML Models**: Trained using frameworks like scikit-learn and XGBoost for robust performance.
* **Data Stores**: Includes data warehouses for historical data and in-memory caches for real-time predictions.

**3: Data Collection & Preprocessing**

**3.1 Data Sources**

The models are trained on a rich variety of meteorological data from reputable sources, ensuring comprehensive and reliable feature inputs.

* **IMD (India Meteorological Department)**: Synoptic and METAR bulletins, storm reports.
* **NOAA (National Oceanic and Atmospheric Administration)**: GHCN/ISD surface observations, NEXRAD radar.
* **ECMWF (European Centre for Medium-Range Weather Forecasts)**: ERA5/ERA5-Land reanalysis for gridded variables.
* **NASA**: GPM precipitation data, GOES/INSAT-3D/3DR satellite products (cloud top temperature, brightness).
* **Other Sources**: Real-time APIs (Open-Meteo, OpenWeatherMap), lightning networks (WWLLN/GLD360).

**3.2 Core Features**

The following features are extracted for each station/grid point at a specific time:

* **Atmospheric Variables**: Temperature, dew point, humidity, mean sea-level pressure (MSLP).
* **Wind Data**: Wind speed/direction, u/v components, gust, and wind shear.
* **Convective Potential**: CAPE (Convective Available Potential Energy) and CIN (Convective Inhibition).
* **Cloud/Precipitation**: Cloud fraction, cloud-top temperature, precipitation rate.
* **Temporal/Spatial Context**: Hour-of-day, day-of-year, season, latitude/longitude, elevation.

**3.3 Data Cleaning & Imputation**

A rigorous preprocessing pipeline ensures data quality:

* **Time Alignment**: All data streams are resampled to a consistent cadence (e.g., 10-minute or hourly) and aligned to UTC.
* **Outlier Removal**: Outliers are identified using range checks and Hampel filters.
* **Imputation**: Short gaps are filled using forward-fill, while longer gaps are addressed with more sophisticated methods like KNN or iterative imputation.

**4: Feature Engineering and Methodology**

**4.1 Feature Engineering**

Raw meteorological variables are enhanced through a series of feature engineering steps to provide more predictive power to the models.

* **Rolling Statistics**: Features like mean, standard deviation, minimum, and maximum are calculated over rolling windows (e.g., 1-hour, 3-hour) to capture short-term trends.
* **Lags**: Lagged values (t-1, t-2, etc.) are included to capture the temporal dependency of weather patterns.
* **Derivative Features**: New variables are created from existing ones, such as pressure fall rate over 3 hours and temperature/humidity tendencies.
* **Directional Encoding**: Wind direction and hour-of-day are encoded using sine and cosine transformations to preserve their cyclical nature.

**4.2 Machine Learning Tasks**

The platform tackles two distinct but related prediction tasks:

* **Thunderstorm Occurrence**: A **binary classification** problem to predict the presence or absence of a thunderstorm. The output is a probability and a confidence score.
* **Gale Speed Prediction**: A **regression** problem to forecast maximum sustained wind or gust speed over a specified time horizon (e.g., next 1-3 hours).

**4.3 Model Selection**

* **Primary Models**: **Tree ensembles** such as **Random Forest, XGBoost, and LightGBM** are chosen as strong, reliable baselines for tabular data.
* **Advanced Models**: For future work, **Temporal CNNs or LSTMs** could be explored to better capture long-term sequential dependencies.

**5: Thunderstorm Prediction Model**

**5.1 Model: RandomForestClassifier**

The thunderstorm predictor utilizes a **RandomForestClassifier**, a powerful ensemble learning method that builds multiple decision trees and merges their predictions to improve accuracy and control overfitting. It is trained on a curated set of features including CAPE, Lifted Index, wind shear, and multi-level wind fields.

**5.2 Model Output**

The model exposes a simple **JSON API** that provides a rich set of outputs for each prediction query:

* prediction: A binary value (0 or 1) indicating the presence of a thunderstorm.
* probability: The model's probabilistic output for the positive class.
* confidence: A measure of prediction confidence.
* risk\_level: A human-readable risk level, mapped to Green, Yellow, or Red, based on the probability score.
* alerts: Text-based alerts corresponding to the risk level.

**5.3 Performance Metrics**

Due to the class imbalance inherent in thunderstorm data, the following metrics are prioritized for evaluation:

* **Precision**: The proportion of positive identifications that were actually correct.
* **Recall**: The proportion of actual positives that were identified correctly.
* **F1 Score**: The harmonic mean of Precision and Recall.
* **PR-AUC**: The Area Under the Precision-Recall Curve, which is more informative for imbalanced datasets than ROC-AUC.
* **Brier Score**: Measures the accuracy of probabilistic predictions.

**6: Gale Speed Prediction Model**

**6.1 Model: RandomForestRegressor**

The gale speed predictor employs a **RandomForestRegressor**, an ensemble model that averages the predictions of multiple decision trees to produce a continuous output. This model is adept at handling the complex, non-linear relationships between meteorological variables and wind speed. It is trained on time-series features like lags, moving averages, and variability, in addition to core weather variables.

**6.2 Forecasting Near-Term Wind Conditions**

The model provides forecasts for near-term wind conditions, which include:

* **Forecasted Wind Speed**: A regression output for the maximum sustained wind or gust over a defined horizon.
* **Wind Category**: A classification of the forecasted wind into categories (e.g., strong winds, gale, severe gale) to provide context.
* **Alert Guidance**: Tailored guidance or alerts for operational safety based on the predicted wind category.

**6.3 Performance Metrics**

The effectiveness of the gale speed model is measured using standard regression metrics:

* **RMSE (Root Mean Squared Error)**: Measures the square root of the average of the squared differences between predicted and actual values.
* **MAE (Mean Absolute Error)**: Measures the average of the absolute differences between predicted and actual values.
* **R² (R-squared)**: Represents the proportion of the variance in the dependent variable that is predictable from the independent variables.
* **Operational Metrics**: Lead-time hit rate and false alarm rate are also monitored for real-world application effectiveness.

**7: Backend API & Microservices**

**7.1 API Endpoints**

The backend is designed for clean integration and automation, exposing three key endpoints:

* **/api/ml/predict**: Accepts a JSON payload of meteorological features and returns the thunderstorm prediction details (risk level, probability, etc.).
* **/api/windspeed/predict**: Accepts a JSON payload and returns the forecasted wind speed and associated alert guidance.
* **/api/health**: A diagnostic endpoint that provides the status of the backend services, including model availability and data sources.

**7.2 Model Packaging and Stability**

To ensure stable and reproducible inference, the machine learning models are carefully packaged with their associated scalers and feature order. This prevents data drift between the training and inference environments and guarantees that predictions are based on the correct input structure. Fallback logic is also included to maintain service when specific models are temporarily unavailable.

**7.3 Technical Stack**

The backend is built on a robust, scalable stack:

* **Core Language**: Python
* **Web Framework**: Flask or FastAPI
* **ML Libraries**: scikit-learn, XGBoost, PyTorch
* **Data Handling**: Pandas, Redis/Kafka for caching and messaging
* **Model Management**: MLflow for model registry and tracking

**8: Frontend: User Interface & Experience**

**8.1 Interactive Map & Overlays**

The Thundercast frontend is a premium **React** application inspired by **Zoom Earth**. It features an interactive **dark map** and provides a suite of real-time weather overlays:

* **Wind Vectors**: Animated wind vectors showing direction and speed.
* **Radar**: Live radar reflectivity to show precipitation.
* **Cloud Cover**: Satellite-based cloud cover and cloud top temperatures.
* **Temperature**: Air temperature at various altitudes.

**8.2 Airfield Risk Assessment**

The UI provides a simple, color-coded risk assessment for airfields, allowing for immediate visual understanding of potential weather threats. Airfield markers on the map change color based on the predicted risk level (Green/Yellow/Red).

**8.3 Design & Technology**

The user interface is designed for clarity and speed.

* **Technical Stack**: React 19.1.1, Tailwind CSS 3.4.0, Leaflet.js, Framer Motion, Lucide React icons.
* **Design System**: A dark theme with accent colors for risk levels, glassmorphism effects, and smooth animations using Framer Motion.
* **Responsive Design**: The app is fully responsive, adapting to all screen sizes with touch-friendly controls.

**9: Conclusion, Limitations & Future Scope**

**9.1 Conclusion**

Thundercast is a comprehensive, end-to-end AI/ML solution for predicting thunderstorms and gale speed. By combining robust data pipelines, proven machine learning models, and a user-centric design, the platform transforms complex meteorological data into confident, actionable insights. This project demonstrates a practical application of data science to enhance public safety and operational efficiency in weather-sensitive sectors.

**9.2 Limitations**

* **Data Gaps**: The project's accuracy is limited by the sparsity and quality of available data, particularly for remote or less-monitored regions.
* **Label Noise**: The labels for thunderstorm events can be noisy, as they are often derived from indirect proxies like lightning density.
* **Generalization**: Spatiotemporal generalization to new, un-seen regions or climates can be a challenge.
* **Non-Stationarity**: Climate trends and sensor biases can affect long-term model performance.

**9.3 Future Scope**

* **Deep Learning**: Exploring advanced architectures like LSTMs or Transformers to better model long-term temporal dependencies.
* **Multi-modal Fusion**: Integrating diverse data types, such as radar reflectivity and satellite imagery, with tabular data for more accurate predictions.
* **Real-Time IoT Integration**: Connecting to real-time IoT sensors and edge stations to enhance nowcasting capabilities.
* **Probabilistic Forecasting**: Moving from point predictions to full probabilistic forecasts (e.g., using quantile regression) to provide uncertainty estimates.
* **Full Production Deployment**: Implementing a complete CI/CD pipeline, advanced monitoring, and automated retraining for a robust, self-maintaining system.