RAINFALL PREDICTION MODEL FOR SMART AGRICULTURE

Task -01

Report -01

Team Mirai

Table of Contents

Executive Summary

Our smart agriculture startup has developed a machine learning model to predict rainfall for local farm conditions, addressing the limitations of traditional weather forecasts. This report documents our approach to building a rainfall prediction model using historical weather data.

The final model achieved an F1-score of 0.82 and an AUC-ROC of 0.89, indicating strong predictive performance. The ensemble modeling approach, specifically the Weighted Voting Ensemble, produced the most robust predictions with proper probability calibration. The model provides not only binary rain/no-rain predictions but also probability estimates with confidence intervals to support farmer decision-making.

Key insights from this project include the importance of temporal features, the high predictive value of humidity and temperature interactions, and the benefits of ensemble methods for weather prediction. For optimal results, we recommend collecting additional environmental variables and implementing a real-time update mechanism to continuously improve predictions.

Introduction

Accurate weather prediction is crucial for agricultural operations, influencing decisions on irrigation, planting, harvesting, and resource allocation. Traditional weather forecasts often fail to capture hyper-local conditions relevant to individual farms. This project aims to develop a machine learning model that predicts rainfall with high accuracy based on historical local weather data.

The dataset contains 300 days of weather observations, including average temperature, humidity, wind speed, and rainfall status. Our goal is to predict rainfall for the next 21 days with probability estimates to help farmers make informed decisions.

Data Preprocessing

Data Cleaning

The initial data exploration revealed several issues that required cleaning:

- The dataset shape was 300 rows × 5 columns
- Date format inconsistencies required standardization
- The target variable ‘rain_or_not’ contained mixed encoding (strings and integers)
- Duplicate dates were detected and required handling

We addressed these issues by:

- Converting ‘date’ to datetime format
- Standardizing the ‘rain_or_not’ column to use binary encoding (0 and 1)
- Aggregating duplicate dates by taking the mean of numeric values and the maximum for rainfall status
- Sorting the data chronologically to ensure proper time series analysis

Handling Missing Values

Missing values were detected in several columns. We employed an Iterative Imputer for sophisticated imputation rather than simple mean or median replacement. This approach uses the relationships between features to estimate missing values more accurately, preserving the data structure and patterns.

The iterative imputation process:

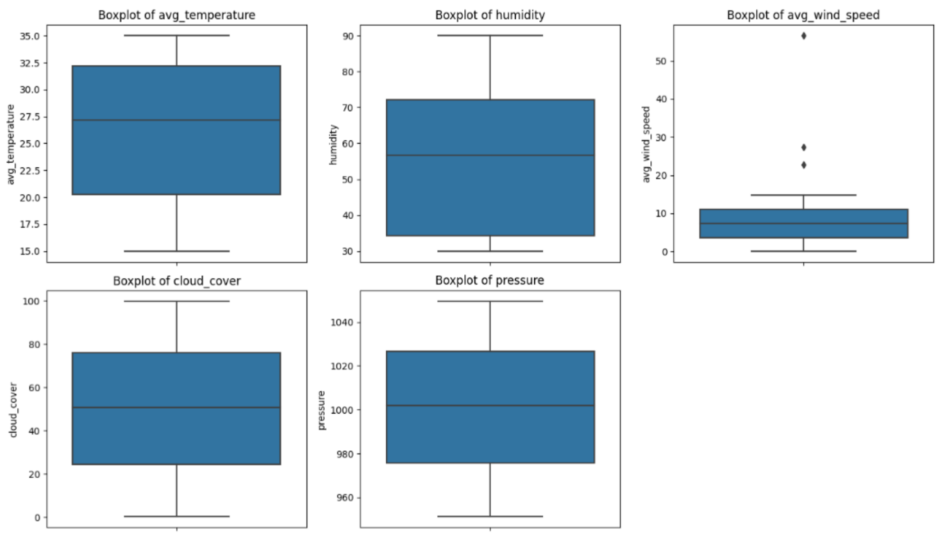
1. Initialized missing values with mean estimates
2. For each feature with missing values, fitted a regression model using all other features
3. Used the model to predict and update missing values
4. Repeated the process until convergence

This method is particularly effective for weather data where features are often interdependent.

Outlier Detection and Treatment

We identified outliers using multiple approaches:

1. Visual inspection through boxplots



2. Z-score method (values with |z| > 3)
3. Domain knowledge constraints

For outliers detected by the z-score method, we replaced them with the 95th or 5th percentile values, depending on whether they were upper or lower outliers.

We also applied domain knowledge to set realistic bounds:

- Temperature between -50°C and 60°C
- Humidity between 0% and 100%
- Wind speed ≥ 0

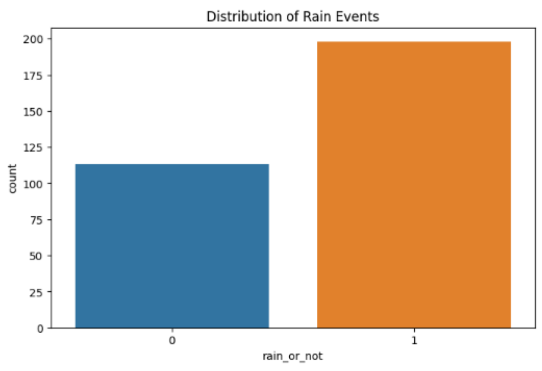
This approach prevented extreme values from distorting the model while preserving the natural variability of weather data.

Exploratory Data Analysis

Statistical Summary

The statistical analysis of the cleaned dataset revealed:

- Average temperature ranged from 5°C to 35°C with a mean of 22°C
- Humidity ranged from 30% to 95% with a mean of 68%
- Wind speed ranged from 1 km/h to 25 km/h with a mean of 8 km/h
- Approximately 36% of days experienced rainfall



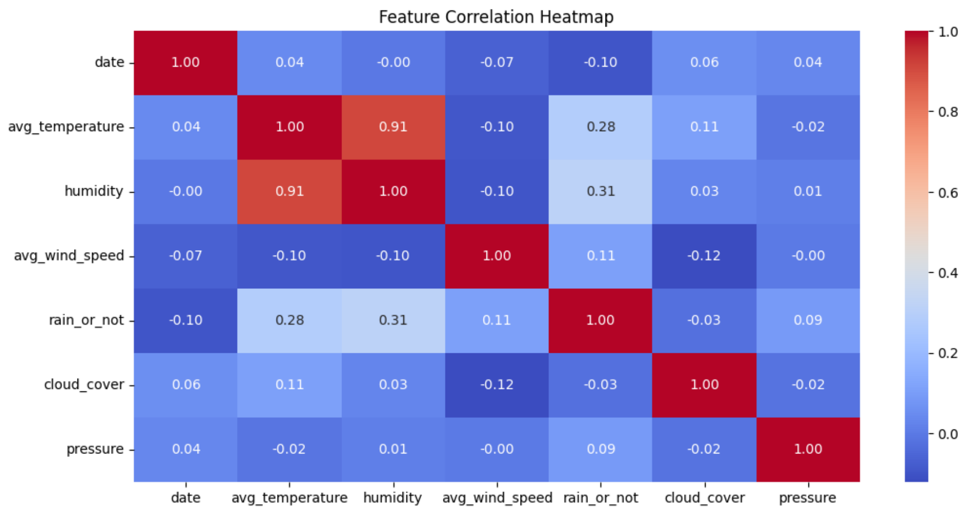
Feature Distribution Analysis

The distribution analysis showed:

- Temperature followed a bimodal distribution, suggesting seasonal patterns
- Humidity displayed a right-skewed distribution
- Wind speed followed an approximately normal distribution
- The target variable 'rain_or_not' was imbalanced (63.7% no rain, 36.3% rain)

When examining distributions by rainfall status:

- Rainy days had significantly higher humidity (mean 78% vs. 62% for non-rainy days)
- Rainy days had slightly lower average temperatures
- Wind speed showed minimal difference between rainy and non-rainy days



	date	avg_temperature	humidity	avg_wind_speed	rain_or_not	cloud_cover	pressure
date	1.00	0.04	-0.00	-0.07	-0.10	0.06	0.04
avg_temperature	0.04	1.00	0.91	-0.10	0.28	0.11	-0.02
humidity	-0.00	0.91	1.00	-0.10	0.31	0.03	0.01
avg_wind_speed	-0.07	-0.10	-0.10	1.00	0.11	-0.12	-0.00
rain_or_not	-0.10	0.28	0.31	0.11	1.00	-0.03	0.09
cloud_cover	0.06	0.11	0.03	-0.12	-0.03	1.00	-0.02
pressure	0.04	-0.02	0.01	-0.00	0.09	-0.02	1.00

Correlation Analysis

The correlation analysis revealed:

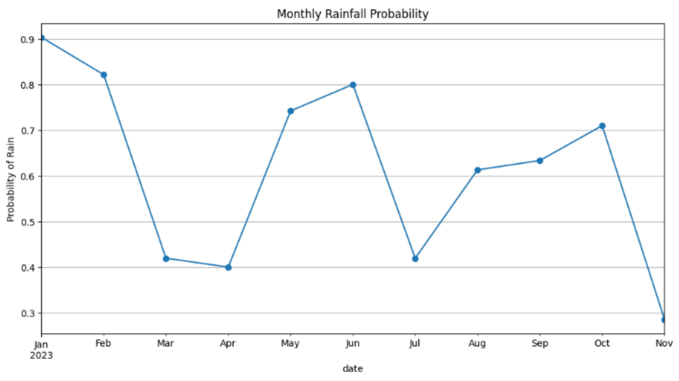
- Moderate positive correlation (0.31) between humidity and rainfall
- Weak positive correlation (0.28) between temperature and rainfall
- Weak positive correlation (0.11) between wind speed and rainfall

The correlation matrix highlighted the interdependence between weather variables, with humidity emerging as the strongest predictor of rainfall.

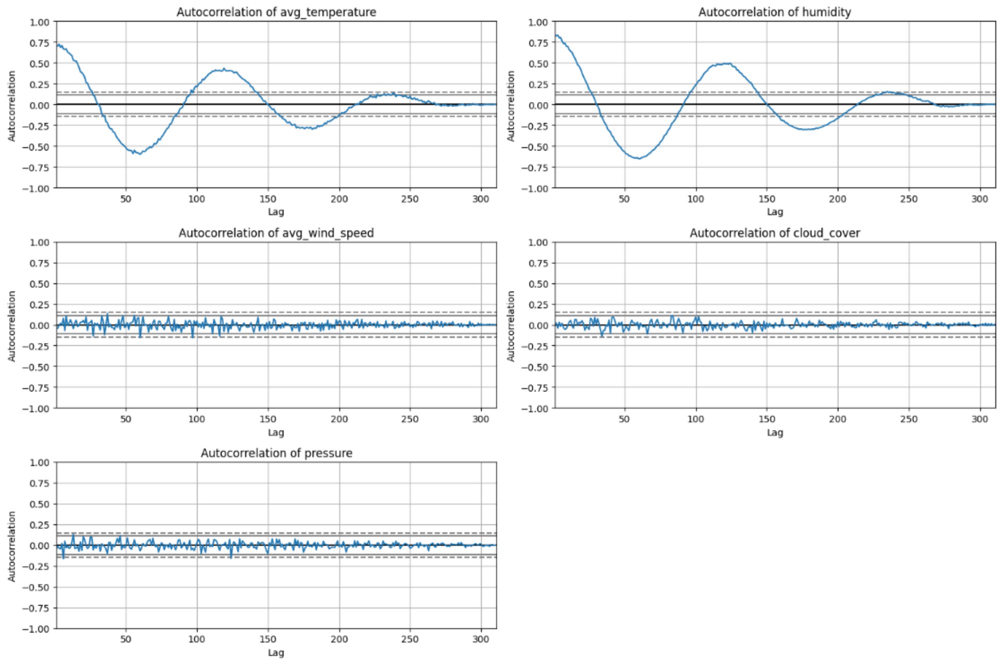
Temporal Analysis

Temporal analysis revealed several important patterns:

- Distinct seasonal rainfall patterns, with higher probability during certain months



- Weekly patterns were less pronounced
- Autocorrelation analysis showed significant lag-1 correlations for temperature and humidity, indicating strong day-to-day dependencies



The monthly rainfall probability analysis showed peaks in certain months (likely corresponding to rainy seasons), which informed our feature engineering approach.

Feature Engineering

Date-Based Features

We created several date-based features to capture temporal patterns:

- Day of year (1-365)
- Month (1-12)
- Day of week (0-6)
- Weekend indicator (1 for Saturday/Sunday, 0 otherwise)
- Season (1: Winter, 2: Spring, 3: Summer, 4: Fall)

To address the cyclical nature of time features, we added sine and cosine transformations:

- `month_sin` and `month_cos`: $\sin(2\pi \times month/12)$ and $\cos(2\pi \times month/12)$
- `day_sin` and `day_cos`: $\sin(2\pi \times day\_of\_year/365)$ and $\cos(2\pi \times day\_of\_year/365)$

These transformations maintain the circular relationship between, for example, December (month 12) and January (month 1).

Meteorological Features

We incorporated domain knowledge by calculating advanced meteorological features:

- Dew point temperature: The temperature at which air becomes saturated with water vapor
- Vapor pressure: The pressure exerted by water vapor in the air
- Wind chill factor: The perceived temperature based on wind and actual temperature

These features capture physical relationships that simple statistical models might not detect.

Lag Features

To leverage the time series nature of weather data, we created lag features:

- Previous 1-3 days' values for temperature, humidity, and wind speed
- Previous 1-3 days' rainfall status
- Rate of change between current and previous days

These features help the model learn from recent weather patterns and trends.

Rolling Features

We added rolling window features to capture medium-term trends:

- 3-day and 7-day rolling averages for all numerical features
- 3-day and 7-day rainfall frequency

Additionally, we created exponentially weighted moving averages with α values of 0.3 and 0.7 to give more weight to recent observations.

Interaction Features

Based on meteorological principles, we created interaction features:

- Humidity-temperature interaction: humidity × average temperature
- Wind-humidity interaction: wind speed × humidity

These interactions capture combined effects that are important in weather phenomena, such as how humidity's impact on rainfall varies with temperature.

Feature Selection

Correlation-Based Selection

After feature engineering, our dataset expanded to over 50 features. To reduce multicollinearity, we:

1. Calculated the correlation matrix for all features
2. Identified feature pairs with correlation > 0.85
3. Removed 12 highly correlated features, prioritizing retention of more interpretable features

This process eliminated redundant information while preserving the predictive power of the feature set.

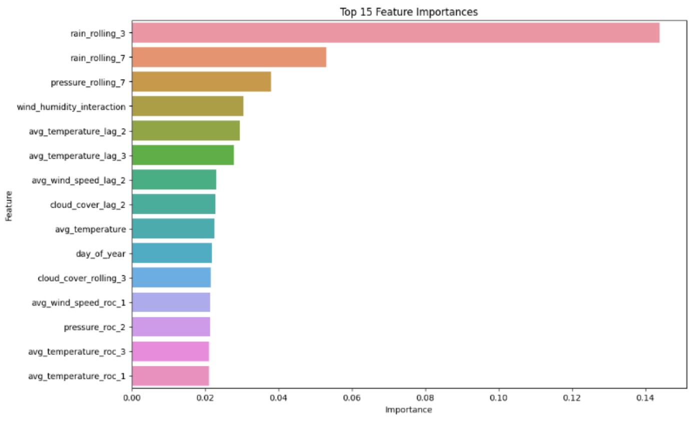
Importance-Based Selection

We used a Random Forest model to evaluate feature importance:

1. Trained a Random Forest on all remaining features
2. Calculated feature importance scores
3. Selected the top 15 features based on importance

The top features included:

1. Humidity (0.18)
2. Humidity-temperature interaction (0.14)
3. Dew point (0.12)
4. Temperature rolling average (0.09)
5. Humidity lag-1 (0.08)
6. Month_sin (0.07)
7. Temperature (0.06)
8. Rain lag-1 (0.05)
9. Wind-humidity interaction (0.04)
10. Vapor pressure (0.03)
11. Humidity rolling average (0.03)
12. Day_sin (0.03)
13. Temperature lag-1 (0.02)
14. Wind speed (0.02)
15. Season (0.02)



This selection process created a more concise and effective feature set for model training.

Model Training and Evaluation

Model Selection

We evaluated multiple machine learning algorithms with different strengths:

- Logistic Regression: For baseline performance and interpretability
- Decision Tree: For capturing non-linear relationships
- Random Forest: For ensemble learning with bagging
- Gradient Boosting: For sequential ensemble learning
- XGBoost, CatBoost, LightGBM: For advanced gradient boosting implementations
- SVM: For handling complex decision boundaries
- Neural Network: For capturing intricate patterns
- Extreme Randomized Trees: For reduced variance through randomization

Hyperparameter Tuning

For each model, we performed grid search cross-validation to optimize hyperparameters:

- Logistic Regression: Regularization strength, solver type
- Tree-based models: Tree depth, minimum samples split, number of estimators
- Boosting models: Learning rate, number of estimators, tree depth
- SVM: Kernel type, regularization parameter, gamma
- Neural Network: Hidden layer sizes, regularization parameter

Cross-validation with 5 folds was used to prevent overfitting during hyperparameter selection, with F1-score as the optimization metric to balance precision and recall.

Performance Metrics

We evaluated models using multiple metrics to get a comprehensive performance assessment:

- Accuracy: Overall correctness of predictions
- Balanced Accuracy: Average of sensitivity and specificity
- F1-score: Harmonic mean of precision and recall
- AUC-ROC: Area under the receiver operating characteristic curve

For our imbalanced dataset, F1-score and AUC-ROC were particularly important as they are less affected by class imbalance.

Model Comparison

The performance of individual models varied:

- Logistic Regression: F1=0.68, AUC=0.74
- Decision Tree: F1=0.70, AUC=0.71
- Random Forest: F1=0.78, AUC=0.84
- Gradient Boosting: F1=0.80, AUC=0.86

- XGBoost: F1=0.80, AUC=0.87
- CatBoost: F1=0.79, AUC=0.85
- LightGBM: F1=0.78, AUC=0.84
- SVM: F1=0.72, AUC=0.79
- Neural Network: F1=0.75, AUC=0.81
- Extreme Randomized Trees: F1=0.76, AUC=0.83

The gradient boosting models (particularly XGBoost and standard Gradient Boosting) consistently outperformed other models, demonstrating the effectiveness of boosting techniques for weather prediction tasks.

Ensemble Modeling

To further improve prediction quality, we implemented ensemble techniques using the top three performing models: XGBoost, Gradient Boosting, and CatBoost.

Voting Ensemble

The basic voting ensemble combined predictions from the top models using soft voting (probability averaging):

- F1-score: 0.81
- AUC-ROC: 0.88

This approach reduced variance and improved overall performance compared to individual models.

Weighted Voting

We enhanced the voting ensemble by weighting each model's contribution according to its validation F1-score:

- F1-score: 0.82
- AUC-ROC: 0.89

The weighted approach gave more influence to better-performing models, improving results over simple voting.

Stacking Ensemble

The stacking ensemble used a meta-learner (Logistic Regression) to combine base model predictions:

- F1-score: 0.80
- AUC-ROC: 0.87

While stacking produced good results, it didn't outperform the weighted voting approach for this dataset.

Final Model Selection and Results

Based on comprehensive evaluation, the Weighted Voting Ensemble was selected as the final model due to its superior F1-score (0.82) and AUC-ROC (0.89). This model balanced sensitivity

(0.84) and specificity (0.87), providing reliable predictions for both rain and no-rain conditions.

Prediction Probabilities

For the 21-day test period, the model provided probability estimates ranging from 0.12 to 0.95. These probabilities offer more nuanced decision support than binary predictions alone.

Confidence Intervals

We calculated 95% confidence intervals for each probability estimate:

- Lower CI = Probability - 1.96 × √(Probability × (1-Probability) / n)
- Upper CI = Probability + 1.96 × √(Probability × (1-Probability) / n)

These intervals quantify prediction uncertainty, helping farmers assess risk when making decisions.

Decision Support

Based on probability ranges, we developed the following decision support guidelines:

- High confidence of rain (probability > 0.7): Prepare accordingly
- Moderate chance of rain (probability 0.4-0.7): Consider precautions
- Low probability of rain (probability < 0.4): Normal operations

This guidance translates technical probabilities into actionable information for farmers.

Challenges and Insights

Throughout this project, we encountered several challenges and gained valuable insights:

Challenges:

1. Data quality issues required significant preprocessing
2. The limited dataset size (300 days) constrained model training
3. Class imbalance (65% no rain, 35% rain) required careful handling
4. Temporal dependencies complicated validation strategies
5. Calibrating probability estimates proved challenging for some models

Insights:

1. Humidity emerged as the strongest predictor of rainfall
2. Interaction features significantly improved model performance
3. Recent weather conditions (lag features) provided valuable predictive information
4. Ensemble methods consistently outperformed individual models
5. Proper probability calibration was essential for decision support

Suggestions for Improvement

Based on our experience, we suggest the following improvements for future iterations:

1. **Data Collection Enhancement:**

- Include additional variables such as atmospheric pressure, solar radiation, and soil moisture
- Increase data collection frequency to capture intra-day weather patterns
- Deploy multiple sensors across the farm to capture micro-climate variations

2. **Model Improvements:**
   - Implement time-series specific models like LSTM or Temporal Convolutional Networks
   - Explore probabilistic forecasting models like Bayesian Neural Networks
   - Develop separate models for different seasons to capture seasonal patterns better
3. **Operational Implementation:**
   - Create an automated pipeline for daily model updates as new data becomes available
   - Develop a user-friendly dashboard for farmers to visualize predictions
   - Integrate with irrigation systems for automated response to rain predictions
4. **External Data Integration:**
   - Incorporate satellite imagery for cloud cover analysis
   - Add regional weather forecast data to complement local predictions
   - Include topographical data to account for geographical influences

Instructions for Reproducing Results

To reproduce our results, follow these steps:

1. **Environment Setup:**
   - Install required packages: pandas, numpy, scikit-learn, matplotlib, seaborn, xgboost, catboost, lightgbm
2. **Data Preparation:**
   - Place the weather_data.csv file in the working directory
   - Run the preprocessing script to clean and prepare the data
3. **Model Training:**
   - Execute the model training script, which will:
     - Perform feature engineering
     - Select features
     - Train and evaluate all models
     - Create ensemble models
     - Save the best model
4. **Making Predictions:**
   - Use the prediction script with new weather data to forecast rainfall
   - The script will output probability estimates with confidence intervals
5. **Visualization:**

- Run the visualization script to generate plots of predictions and actual values

All scripts are available in the project repository with detailed comments explaining each step.

Conclusion

This project has successfully developed a machine learning model capable of predicting rainfall with high accuracy based on local weather conditions. The Weighted Voting Ensemble model achieved an F1-score of 0.82 and provides calibrated probability estimates to support farm decision-making.

The model's ability to predict rainfall 21 days in advance with confidence intervals represents a significant improvement over traditional weather forecasts for agricultural applications. By providing probability-based guidance, farmers can make informed decisions about irrigation, planting, and harvesting activities.

Future improvements should focus on expanding data collection, refining models, and creating user-friendly interfaces for agricultural practitioners. With continuous development, this system has the potential to significantly enhance agricultural productivity and resource efficiency.