*Rainfall Prediction: Model Refinement and Test Submission*

### Team Members

1. Selam Kiflay
2. Hilina Amare
3. Abdi Gonfa
4. Ketim Teklu

# Model Refinement

## Overview

The model refinement phase enhances the rainfall prediction model’s performance by addressing initial evaluation shortcomings and optimizing its predictive accuracy. This phase is crucial for improving reliability in forecasting daily rainfall probability, ensuring the system supports agricultural planning and water management effectively. Refinement involves tuning parameters, adjusting features, and exploring alternative techniques to boost metrics like accuracy and F1-score, building on the foundation laid in model exploration.

## Model Evaluation

Initial evaluation of the Random Forest model yielded an accuracy of 84.77%, F1-score of 0.82, and AUC-ROC of 0.89 on the validation set. Visualizations (e.g., ROC curve, feature importance plot) showed strong performance but highlighted issues: the model struggled with non-rainy days (14% of data) due to class imbalance, and some features (e.g., wind direction) had low importance. Areas for improvement included enhancing minority class prediction and reducing feature redundancy to improve efficiency and generalization.

## Refinement Techniques

Refinement techniques included:

* + *Class Weight Adjustment*: Applied balanced class weights to prioritize non-rainy days, addressing the 86% rainy day imbalance.
  + *Ensemble Exploration*: Tested a voting classifier combining Random Forest and lo- gistic regression to leverage complementary strengths, but it underperformed (ac- curacy 83.5%).
  + *Feature Pruning*: Removed low-importance features (e.g., wind direction) based on feature importance scores, reducing model complexity.

These techniques improved the F1-score for non-rainy days by 5%, enhancing overall robustness.

## Hyperparameter Tuning

Additional tuning was performed using grid search (Cell 19):

* + *Parameters*: Expanded n*estimators* = [100*,* 200]*, maxdepth* = [10*,* 15*,* 20]*, minsamplessplit* = [2*,* 5]*.Insights* : *Increasingnestimatorsto*200*andmaxdepthto*15*improvedaccuracyto*85*.*92%*andF* 1*− scoreto*0*.*84*, butfurtherincreasescausedoverfitting.*
  + *Impact*: The tuned model reduced false negatives for non-rainy days, critical for bal- anced predictions.

## Cross-Validation

The initial 5-fold time-series cross-validation was modified to a 10-fold expand- ing window approach to better capture temporal patterns and reduce variance in val- idation scores. This change ensured training sets grew sequentially, preventing data leakage and improving model stability. The refined strategy increased the average validation AUC-ROC from 0.89 to 0.91, confirming better generalization across time periods.

## Feature Selection

Feature selection used Random Forest’s feature importance scores:

* + *Method*: Dropped features with importance < 0.05 (e.g., wind direction, sunshine hours).
  + *Impact*: Reduced feature count from 12 to 8, decreasing training time by 15% while maintaining accuracy (85.90%).
  + *Rationale*: Eliminating redundant features streamlined the model, focusing on high- impact predictors like humidity and pressure.

# Test Submission

## Overview

The test submission phase prepared the refined model for evaluation on the test dataset (*test.csv*) and real-world application via WeatherAPI predictions. This phase ensured the model’s robustness by applying consistent preprocessing, generating predictions, and assessing performance, paving the way for potential deployment in agricultural decision-making systems.

## Data Preparation for Testing

The test dataset (*test.csv*) was prepared by:

* + Aligning features with *train.csv* (e.g., temperature, humidity, pressure).
  + Applying identical preprocessing : forward fill for missing values ( 3%), Stan- dardScaler for numerical features, and one-hot encoding for wind direction.
  + Adding engineered features (e.g., humidity-pressure interaction, cyclical day en- coding) to match training data.

Considerations included ensuring temporal consistency and handling class imbalance in predictions.

## Model Application

The refined Random Forest model (Cell 24) was applied to *test.csv* to predict rainfall probabilities, saved as *submission.csv*. For real-world use, Cell 25 fetched WeatherAPI data for user-specified locations and dates, mapping features to the model’s input format. Predictions were generated as probabilities (e.g., 68.5%) and binary outcomes (Rain/No Rain).

**Code Snippet (Model Application)**:

**import** pandas as pd

**from** sklearn.ensemble **import** RandomForestClassifier

**import** pickle

# Load test data

test\_df = pd.read\_csv(’test.csv’)

# Apply preprocessing (same as training)

test\_df[’humidity’] = test\_df[’humidity’].fillna(method=’ffill’) scaler = pickle.load(**open**(’scaler.pkl’, ’rb’))

test\_df[[’temperature’, ’humidity’, ’pressure’]] = scaler.transform(test\_df [[’temperature’, ’humidity’, ’pressure’]])

# Predict

model = pickle.load(**open**(’model\_rf.pkl’, ’rb’))

predictions = model.predict\_proba(test\_df.drop([’id’], axis=1))[:, 1] submission = pd.DataFrame({’id’: test\_df[’id’], ’rainfall’: predictions}) submission.to\_csv(’submission.csv’, index=False)

## Test Metrics

Metrics mirrored training: accuracy, F1-score, AUC-ROC. Test results: accuracy = 85.10%, F1-score = 0.83, AUC-ROC = 0.90, compared to training (85.92%, 0.84, 0.91) and validation (85.77%, 0.84, 0.91). The slight accuracy drop reflects test set variability, but high AUC- ROC confirms robust generalization. Visualizations (e.g., ROC curve) showed consistent performance.

## Model Deployment

After completing all project tasks, we plan to develop a user-friendly web interface using Flask and deploy it on cloud platforms such as modal or Render to deliver rainfall predictions to agricultural users, like farmers. The interface will extend the prototype script , enabling users to input a country, city, and date to receive rainfall probabilities (e.g., 68.5%, Rain/No Rain) powered by WeatherAPI. Steps include building a Flask-based web app with a front-end form, integrating the Random Forest model , and ensuring seamless WeatherAPI connectivity. Challenges include maintaining 3 API reliability and ensuring accessibility for non-technical users, addressed through response caching and an intuitive interface. Initial testing is underway to validate local server execution, with cloud deployment planned to ensure accessibility and scalability

## Code Implementation

### Code Snippet (Refinement and Hyperparameter Tuning):

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.model\_selection **import** GridSearchCV, TimeSeriesSplit

# Prepare data

X\_train = df.drop([’rainfall’], axis=1) y\_train = df[’rainfall’]

# Define parameter grid param\_grid = {

’n\_estimators’: [100, 200],

’max\_depth’: [10, 15, 20],

’min\_samples\_split’: [2, 5]

}

# Initialize model and cross-validation

rf = RandomForestClassifier(random\_state=42, class\_weight=’balanced’) tscv = TimeSeriesSplit(n\_splits=10)

grid\_search = GridSearchCV(rf, param\_grid, cv=tscv, scoring=’f1’)

# Fit model grid\_search.fit(X\_train, y\_train)

**print**(f’Best parameters: {grid\_search.best\_params\_}’)

**print**(f’Best F1-score: {grid\_search.best\_score\_:.2f}’)

# Save best model

best\_model = grid\_search.best\_estimator\_ pickle.dump(best\_model, **open**(’model\_rf.pkl’, ’wb’))

# Conclusion

The model refinement phase improved the rainfall prediction model’s accuracy from 84.77% to 85.92% and F1-score from 0.82 to 0.84 through class weight adjustments, hyper- parameter tuning, and feature selection. The test submission phase confirmed robust performance (85.10% accuracy, 0.90 AUC-ROC) on *test.csv* and enabled real-world predictions via WeatherAPI. Challenges included handling class imbalance and API reliability, addressed through balanced weights and caching. The final model supports agricultural planning with reliable, user-friendly forecasts.

# References References

1. Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Ma- chine Learning Research*, 12, 2825–2830.
2. WeatherAPI. (n.d.). Real-Time Weather and Forecast Data.