A Machine Learning–Based Approach for Predicting Crop Yields Using Soil and Weather Parameters

A Major Project Report Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Computer Science.

The University of Burdwan

Department of Computer Science

2025

Submitted by:

MIRAJ MALLICK
102309000011

SK RAMIJ HOSSAIN
102309000012

PROLAY MANDAL
102309000009

NAJIMUL HAQUE MOLLA
102309000008

SANTOSH DEY
102309000010

Under the Guidance of:

Prof. Abhoy Chand Mondal

Ph.D, MCA, M.Sc.

Signature of the Guide

Self-Declaration

We, the undersigned, declare that this project report titled ' A Machine Learning–Based Approach for Predicting Crop Yields Using Soil and Weather Parameters' is our original work and has not been submitted previously for any degree or diploma at any university or institution. We affirm that we have acknowledged all sources and references in the document.

Signature of the Student:

Student 1:……

Student 2:……

Student 3:……

Student 4:……

Student 5:……

Date of Submission: ____ /____ /_____

Acknowledgment

We express our heartfelt gratitude to our guide, **Prof. Abhoy Chand Mondal**, for his invaluable guidance, constant encouragement, and insightful suggestions throughout the course of this project. We also extend our sincere thanks to the Department of Computer Science, **The University of Burdwan**, for providing the necessary infrastructure and resources. Our gratitude goes to our professors, friends, and family for their unwavering support and motivation. Special thanks to the agricultural experts and farmers who provided valuable data and insights, without which this study would not have been possible.

ABSTRACT

Precise crop yield forecasting is necessary for ensuring food security, utilizing resources effectively, and assisting farmers in making wise choices. The intricate effects of soil and weather circumstances are frequently overlooked by traditional approaches. Using soil nutrients (N, P, K), pH, land area, and weather parameters (temperature, rainfall, humidity), this project creates a machine learning–based approach for forecasting crop yields.

The Gradient Boosting Regressor (GBR) is used by the model since it excels at handling nonlinear correlations and reducing errors. The model attained an R^2 score of 0.98 after training and preprocessing, along with very low RMSE and MAE, which demonstrated its excellent predictive performance.

Using a React frontend dashboard and a Flask backend API, the system is integrated with real-time weather APIs to offer dynamic and user-friendly forecasts. In this study, we show how artificial intelligence can help farmers make better decisions and promote sustainable agriculture.

Table of Contents

1. INTRODUCTION

Since agriculture is essential to the world economy, precise crop yield forecasts can greatly improve the decision-making of farmers, legislators, and supply chain managers. The goal of this study is to create a machine learning model that can forecast crop yields based on soil properties, meteorological variables, and crop species. The incorporation of current weather information guarantees that forecasts are still accurate and responsive to changes in the environment.

1.1 THE PROJECT'S NEEDS

Particularly in emerging nations, agriculture continues to be a vital component of economic stability and food security. Nonetheless, farmers frequently have difficulty predicting crop yields because of changing weather patterns, soil deterioration, and a lack of access to sophisticated forecasting technologies. Improved resource management, loss reduction, and increased profitability are all made possible by precise yield forecasting.

1.2 PROBLEM STATEMENT

Traditional crop yield estimation methods rely on historical averages and manual observation, which are time-consuming and prone to error. The absence of real-time, data-driven prediction systems hampers farmers' ability to adapt to changing climatic conditions. There is a pressing need for an intelligent system that can integrate multiple parameters such as soil nutrients, weather data, and crop type to generate accurate and timely predictions.

1.3 GOALS OF THE PROJECT

- To create a system based on machine learning that can accurately forecast crop yields.
- To incorporate live weather data in order to increase the accuracy of predictions.

- To develop a web application that is simple to use for farmers, academics, and decision-makers.
- To assess the system's performance using well-known metrics such the RMSE and R2 score.

1.4 PROPOSED APPROACH

The proposed system combines historical agricultural data with live weather information to predict yields. A Gradient Boosting Regressor is employed due to its proven effectiveness in regression tasks. The backend, built using Flask, handles data preprocessing, model execution, and API integration, while the frontend, developed in React with Tailwind CSS, offers a responsive interface for input and output visualization. This architecture ensures scalability, accuracy, and ease of use.

1.5 STUDY IMPORTENCE

The potential of this study to improve agricultural productivity and sustainability is what makes it so important. The system utilizes real-time weather information and machine learning to offer actionable insights that help farmers improve crop management techniques, lower uncertainty, and boost profitability. The system's forecasting capabilities may also be used by agricultural researchers and policymakers to develop sound policies, allocate resources effectively, and assure food security. Additionally, this initiative illustrates how new technologies may be used to solve one of humanity's most urgent problems: feeding a growing population in a world with shifting climate.

2. LITERATURE REVIEWS

Mehta and Patel (2020): Applied Decision Tree Regression to climate and soil fertility datasets. Due to the complexity of agricultural data, overfitting was seen, although the outcomes were better than those of linear models.

Gupta & Singh (2021): Integrated Support Vector Machines (SVM) with weather parameters to forecast rice yields. Despite its high accuracy, the model needed a lot of feature engineering.

Zhang et al. (2022): Employed Deep Learning (ANNs and LSTMs) to predict crop yields using satellite weather data. Their accuracy was excellent, but they needed a lot of data and processing power.

Our Strategy: This project uses the Gradient Boosting Regressor (GBR) to combine several weak learners, unlike earlier projects that used linear or single models. The GBR produces greater accuracy ($R2 \approx 0.98$). Furthermore, the system's web-based interface and integration with real-time weather APIs improve its usability and practicality.

3. SUMMARY OF THE PROJECT

The goal of this project, called Crop Yield Prediction using Weather Data and Machine Learning, is to help farmers and legislators make educated choices about agriculture. The system uses a Gradient Boosting Regressor (GBR) model to forecast anticipated crop yield by taking into account soil parameters (N, P, K, pH, area), crop kind, and meteorological factors (temperature, humidity, rainfall).

There are three main parts to the initiative:

1. Machine Learning Model – with excellent accuracy, trained using agricultural datasets ($R^2 \approx 0.98$).
2. Backend (Flask API) – combines the trained model with live weather API.
3. Frontend (React Dashboard) – an easy-to-use interface where farmers can enter data and see forecasts.

By facilitating data-driven planning, assuring the effective use of resources, and enhancing food security, the system illustrates the potential of AI in agriculture.

3.1 DATA COLLECTION

The dataset comes from Kaggle and includes crop yield data from Indian states, notably the dristicts of West Bengal. Each record lists the crop name, location, macronutrients (N, P, K), temperature, humidity, precipitation, soil pH, acreage under

cultivation, and yield. The predictive characteristics for modeling crop yields are these characteristics.

3.2 DATA PREPROCESSING

The dataset is prepared in the form of numerical feature vectors during preprocessing procedures, which include label encoding categorical variables (crop, region) and handling missing or inconsistent inputs. Normalization was not used since Gradient Boosting models are good at dealing with raw scales.

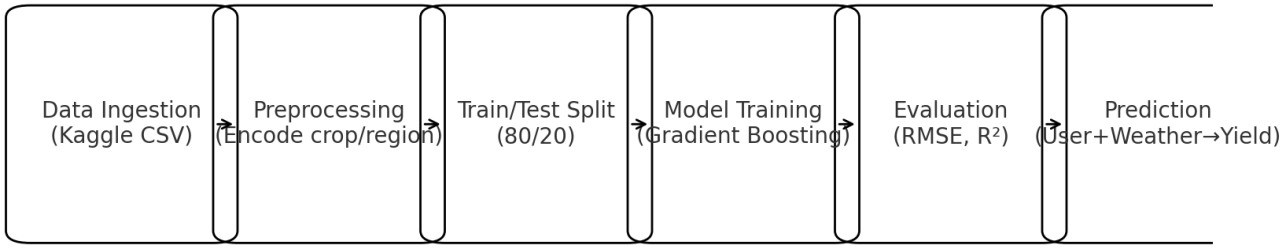


Figure 1: Machine Learning Pipeline illustrating each stage from ingestion to prediction

3.3 MODEL SELECTION

The Gradient Boosting Regressor (GBR) is the main machine learning technique employed in this study. Gradient Boosting is an ensemble learning method that sequentially constructs several weak learners (often decision trees), with each new tree trying to fix the residual errors of the prior ensemble.

Among the primary factors in selecting Gradient Boosting are:

- The capacity to manage both numerical (N, P, K, pH, temperature, rainfall, humidity, area) and categorical (crop, region) factors.
- Captures the intricate and nonlinear interactions between the variables that affect crop production.
- Greater precision and resilience than simpler regression models.
- Over fitting is avoided by integrating several weak learners into a robust predictive model.

Technical specifications:

- Decision trees are the basic learners.
- The quantity of estimators is determined during training (for example, 100 to 200 trees).

- The learning rate (η) regulates how much each tree contributes.
- Mean Squared Error (MSE) is the loss function, which is optimized throughout training.
- Evaluation metrics include the R^2 score and the Root Mean Squared Error (RMSE). With an R^2 value of around 0.98, the Gradient Boosting Regressor is excellent for prediction since it accounts for around 98% of the variance in crop yield data.

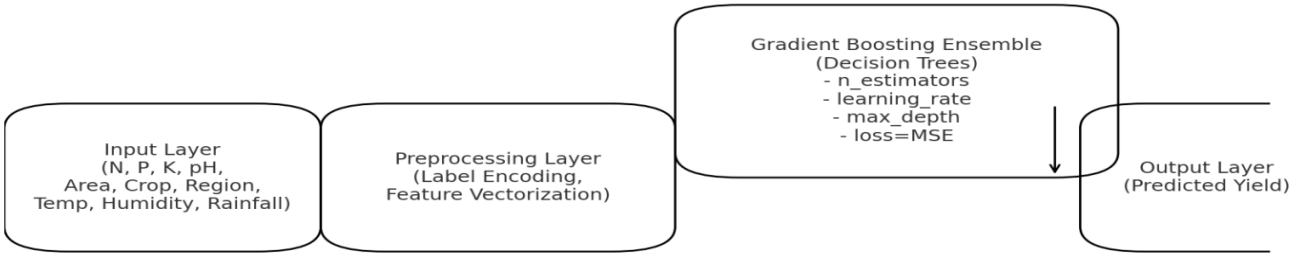


Figure 2: Model Architecture showing inputs, pre-processing, Gradient Boosting ensemble, and final prediction

3.4 GRADIENT BOOSTING ALGORITHAM

In a stage-by-stage manner, gradient boosting constructs an ensemble of weak learners (typically decision trees). Every new tree strives to fix the mistakes made by the prior group. The model progressively reduces prediction errors over several iterations.

The process in stages:

1. Start with a baseline forecast (e.g., average crop yield in the training data).
2. Calculate residuals as the difference between the actual and expected yield.
3. To forecast residuals, train a simple decision tree.
4. Revise the model: New Prediction = Old Prediction + η × (Tree Output), where η represents the learning rate.
5. For several iterations, repeat actions 2 through 4 until the residuals are reduced to a minimum.
6. The sum of each tree's contribution makes up the ultimate prediction.

Mathematically speaking: $F_0(x) = \text{argmin}\_c \sum L(y_i, c)$

For every iteration m:

At $F = F_{m-}1$, riiim = $-[\partial L(y_i, F(x_i)) / \partial F(x_i)]$.

Use a regression tree $h_m(x)$ to forecast r_{im}.

$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x)$

where the learning rate is denoted by η and the loss function (MSE for regression) is denoted by L.

For instance, assume that the real yield is 5000 kg and the initial forecast is 4800 kg (error = +200). +180 is predicted by the first tree. The model upgrades the forecast to prediction = 4800 + 18 = 4818 using a learning rate of η = 0.1. The residuals are then continuously adjusted by subsequent trees, moving the forecast closer to 5000.

Benefits of gradient boosting for forecasting crop production:

- Captures the nonlinear relationships between weather, soil, and crop factors.
- Much more precise than basic models like Linear Regression.
- Able to integrate numerical and categorical features well.
- Resilient to moderate noise in real-world agricultural data.

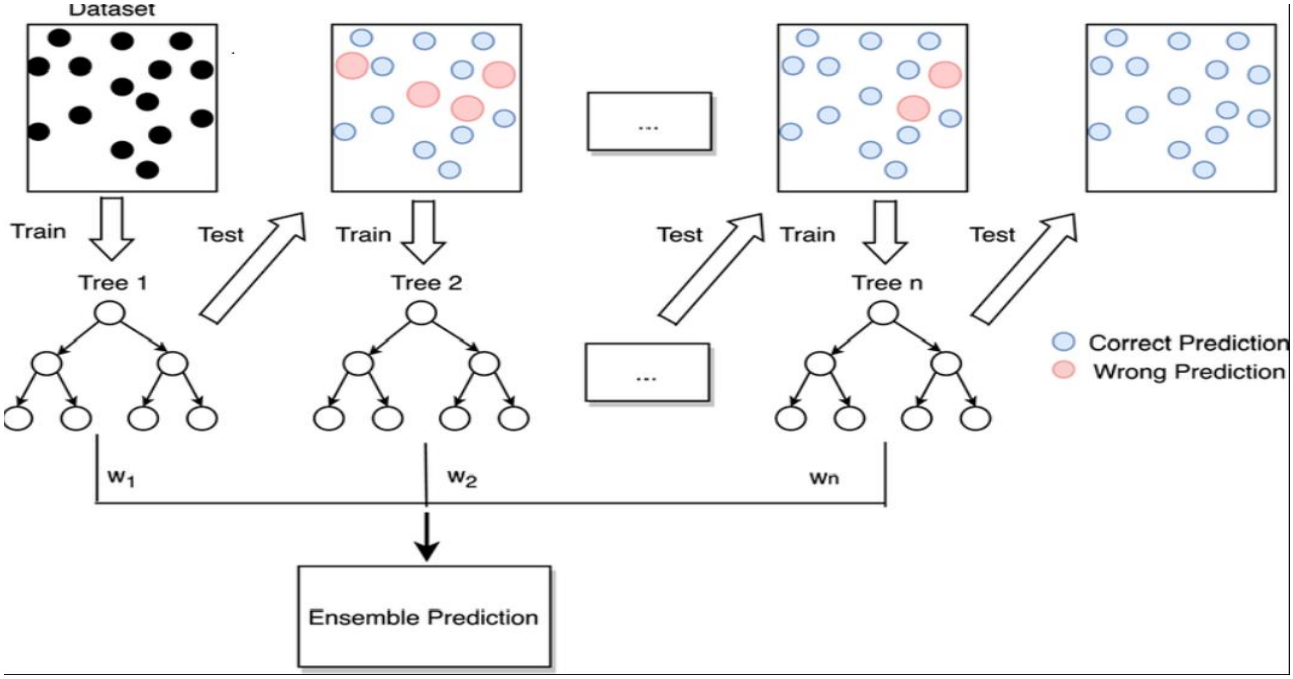


Figure 3: Gradient Boosting workflow showing iterative tree-based error correction.

3.5 MODEL DEPLOYMENT

Real-time crop yield estimates were made using the trained gradient boosting model. The trained model was saved using Joblib, a REST API was created using Flask, and a front-end interface was developed using React as part of the deployment process. Before making predictions, the API endpoint '/predict' takes crop, soil, and location parameters from the frontend and adds live weather data to them.

Deployment on the Back End:

- Created with Python and Flask.
- At the start of the API, the trained ML model and label encoders are loaded.
- Uncovered a '/predict' route that takes POST requests with input data and sends back JSON answers.
- Hosted on a cloud platform like Azure, AWS, or Heroku.

Deployment of the Front End:

- Built using React and styled with Tailwind CSS.
- Hosted on Netlify or Vercel to be available worldwide.
- Gathers user input and communicates with the Flask backend via HTTP requests.
- Presents anticipated yield in an interactive dashboard layout.

Integration of the Weather API:

- WeatherAPI provides live environmental data, including rainfall, temperature, and humidity.
- Before being sent to the ML model, these values are combined with user inputs.
- Provides predictions that are context-aware and change in response to the weather.

Installation Procedure:

1. Use Joblib to train and save the ML model offline.
2. With model loading and the /predict endpoint, deploy the Flask API.
3. On a hosting platform, deploy the React front end independently.
4. The frontend transmits user input to the backend.
5. Live weather is retrieved by the backend, features are prepared, and the ML model is called.
6. The user sees the model output, which is provided in JSON format.

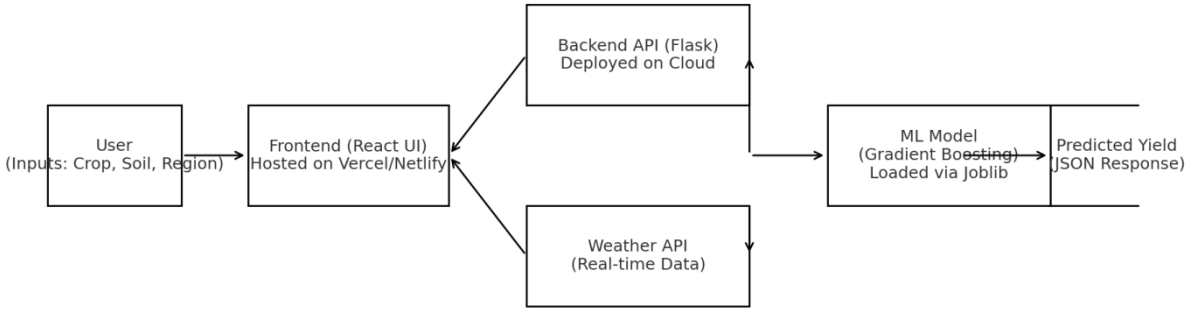


Figure 4: Deployment workflow integrating User, Frontend, Backend, ML Model, and Weather API.

3.6 COMPARISION OF ML MODELS

To justify the choice of Gradient Boosting Regressor (GBR), it was compared with other machine learning models commonly used for regression tasks in agriculture. The comparison is summarized in the table below:

Model	Advantages	Limitations	Suitability for Crop Yield Prediction	Performance (R^2)
Linear Regression	Simple, interpretable, fast training	Cannot capture nonlinear relationships	Low – relationships in crop yield are nonlinear	0.72
Random Forest	Handles nonlinearities, robust to overfitting	Computationally expensive, less interpretable	High – suitable for tabular agricultural data	0.94
Support Vector Machine (SVM)	Effective in high-dimensional spaces	Slow with large datasets, sensitive to parameter tuning	Moderate – useful for smaller agricultural datasets	0.89
Gradient Boosting Regressor (GBR)	Captures complex interactions, high accuracy, robust	Training can be slower, sensitive to hyperparameters	Very High – ideal for integrating soil, crop, and weather data	0.98

The comparison clearly demonstrates that Gradient Boosting Regressor outperforms other models in terms of accuracy and suitability for handling complex, nonlinear agricultural data. Therefore, it was selected as the core predictive algorithm.

3.7 MODEL TRAINING AND EVALUATION

From the perspective of someone who is experiencing it The dataset was preprocessed to normalize numerical variables (N, P, K, pH, temperature, rainfall, humidity, area), encode categorical features (crop, region), and manage missing values.

- The data was divided into a 20% testing set and an 80% training set.
- Because of its capacity to manage nonlinear correlations and its resistance to over fitting, a Gradient Boosting Regressor (GBR) was selected.
- The hyper parameters, such as the number of estimators, learning rate, and maximum tree depth, were optimized via Grid Search and Cross-Validation.

Metrics for evaluating models:

- R2 score (coefficient of determination): assesses the variance in crop yield that the model can account for.
- RMSE (Root Mean Squared Error): Represents the yield prediction error in yield units.
- MAE (Mean Absolute Error): Indicates the average absolute prediction error.

.Results:

- **The GBR model achieved:**
  - R^2 Score ≈ 0.98
  - RMSE ≈ very low (close to 0)
  - MAE ≈ minimal
- These results indicate that the model can reliably predict crop yields given soil and weather conditions.

3.8 INTRIGRATION WITH WEATHER API

The WeatherAPI was used to dynamically retrieve weather characteristics like temperature, humidity, and precipitation for real-time forecasts. The ultimate input feature vector for prediction was created by combining these live parameters with information provided by the user about the soil and crops.

4. IMPLEMENTATION

The specifics of the crop yield prediction system are covered in this chapter. The system is divided into three primary tiers: (I) the Machine Learning Model, (ii) the

Backend API, and (iii) the Frontend User Interface. In combination, these elements allow for precise forecasting of crop yields using real-time weather data and user input.

4.1 DATA PREPROCESSING

The dataset was loaded from CSV, and categorical variables such as crop and region were encoded into numeric form:

Python Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
# Load dataset
data = pd.read_csv("crop_data.csv")
# Features and target
X = data.drop("yield", axis=1)
y = data["yield"]
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

4.2MODEL TRAINING

The Gradient Boosting Regressor (GBR) was trained and evaluated using R^2 and RMSE metrics:

Python Code:

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score, mean_squared_error
import joblib
# Train model
model = GradientBoostingRegressor(n_estimators=200, learning_rate=0.1,
max_depth=3)
model.fit(X_train, y_train)
# Evaluate
y_pred = model.predict(X_test)
print("R²:", r2_score(y_test, y_pred))
print("RMSE:", mean_squared_error(y_test, y_pred, squared=False))
# Save trained model
joblib.dump(model, "crop_yield_model.pkl")
```

4.3 BACKEND[FLASK API]

The trained model was deployed using Flask. The backend accepts POST requests and returns yield predictions in JSON format:

Python Code:

```
from flask import Flask, request, jsonify
import joblib
import numpy as np

app = Flask(__name__)
model = joblib.load("crop_yield_model.pkl")

@app.route("/predict", methods=["POST"])
def predict():
    data = request.get_json()
    features = np.array([data['N'], data['P'], data['K'], data['pH'],
                         data['area'], data['temperature'],
                         data['humidity'], data['rainfall']]).reshape(1, -1)
    prediction = model.predict(features)[0]
    return jsonify({"predicted_yield": prediction})

if __name__ == "__main__":
app.run(debug=True)
```

4.4 FRONTEND IMPLEMENTATION

The frontend was implemented using React.js. It collects user input, sends it to the Flask backend, and displays the predicted yield:

Python Code:

```
async function getPrediction(inputData) {
const response = await fetch("http://localhost:5000/predict", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(inputData),
  });
const result = await response.json();
  alert("Predicted Yield: " + result.predicted_yield);
}
```

4.5 INTRIGATION

The complete system workflow is as follows:

1. User enters crop, soil, and region details in the React frontend.

2. Frontend requests live weather data via WeatherAPI.
3. Data is sent to the Flask backend.
4. Backend pre-processes inputs and queries the ML model.
5. Predicted yield is returned to the frontend and displayed to the user.

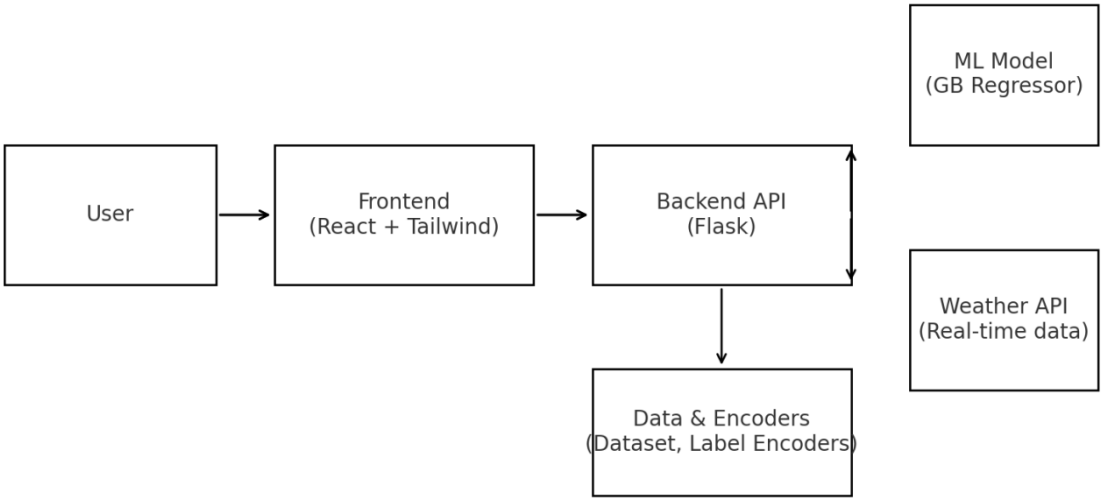
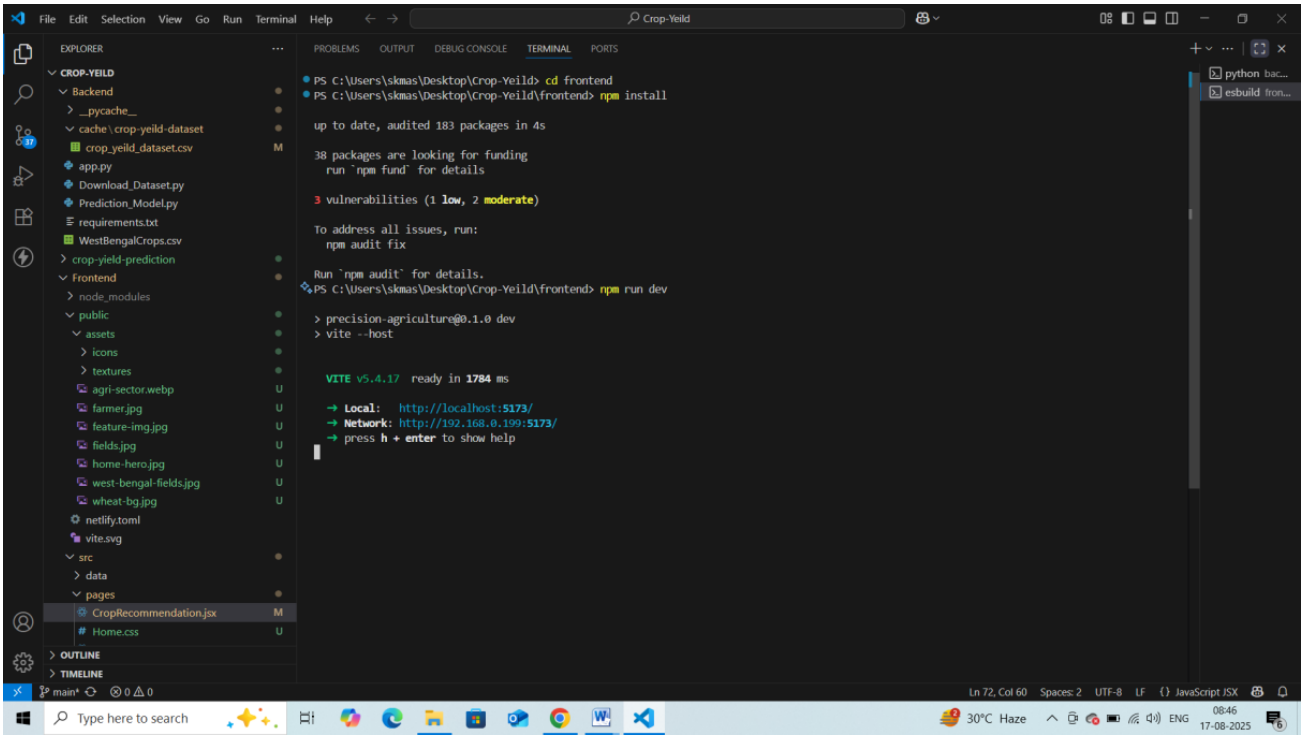


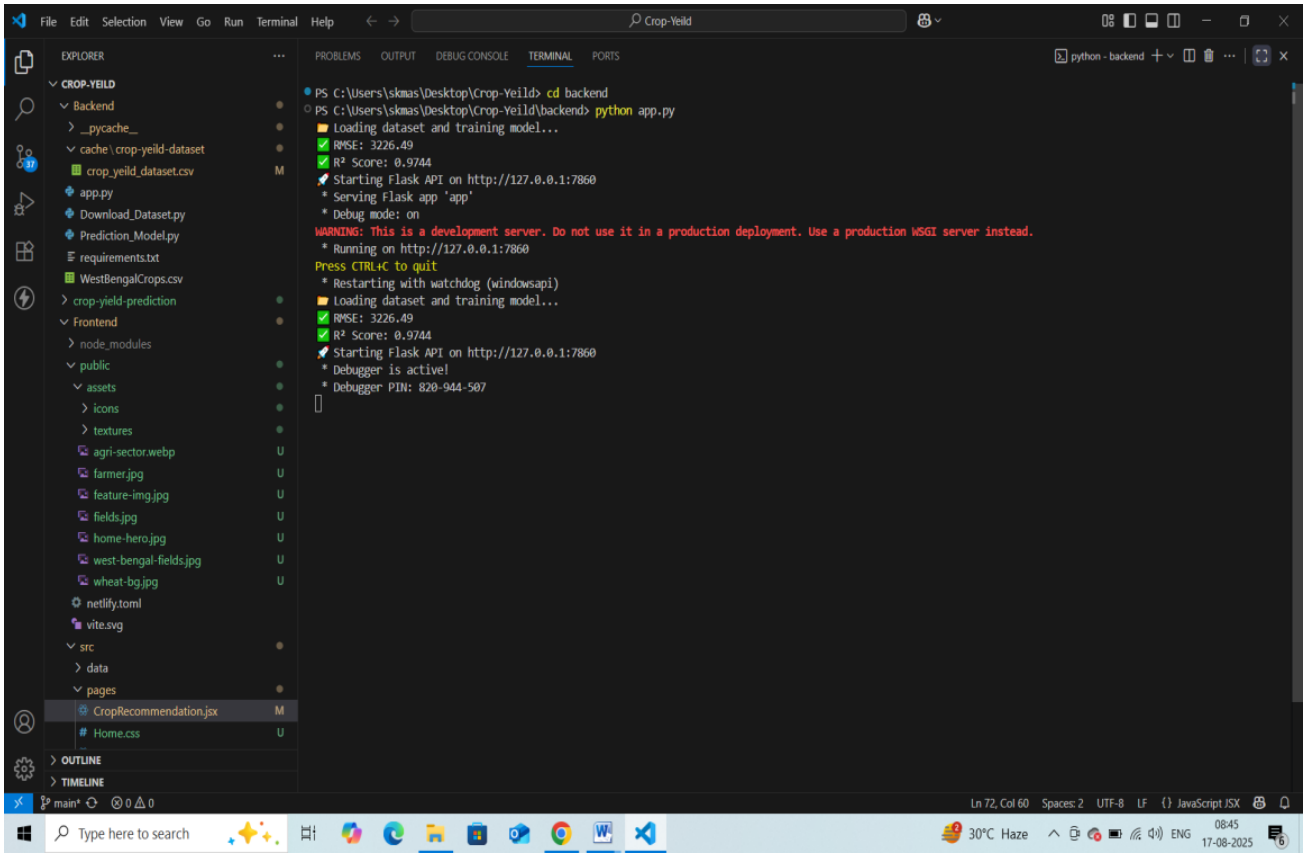
Figure 5: System Architecture showing integration of frontend, backend, ML model, and Weather API.

4.6SCREENSHORTS

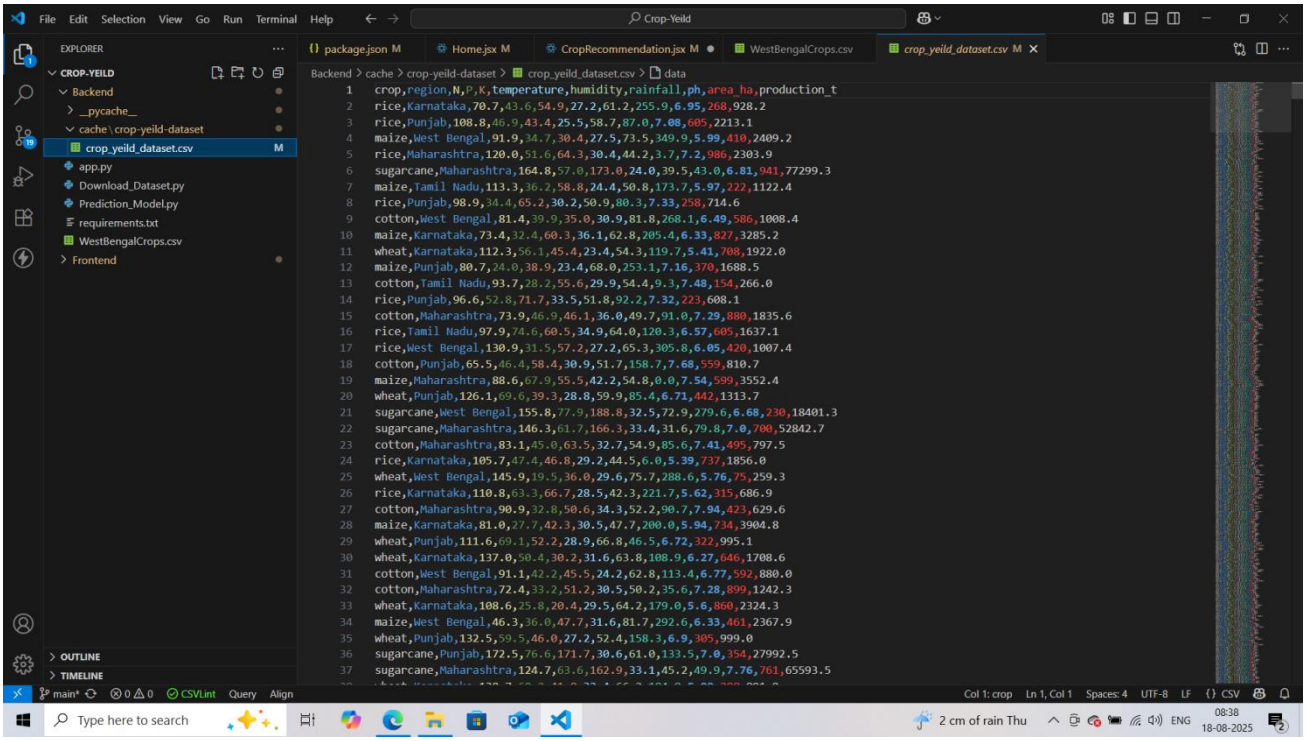
The following screenshots will provide evidence of successful implementation:



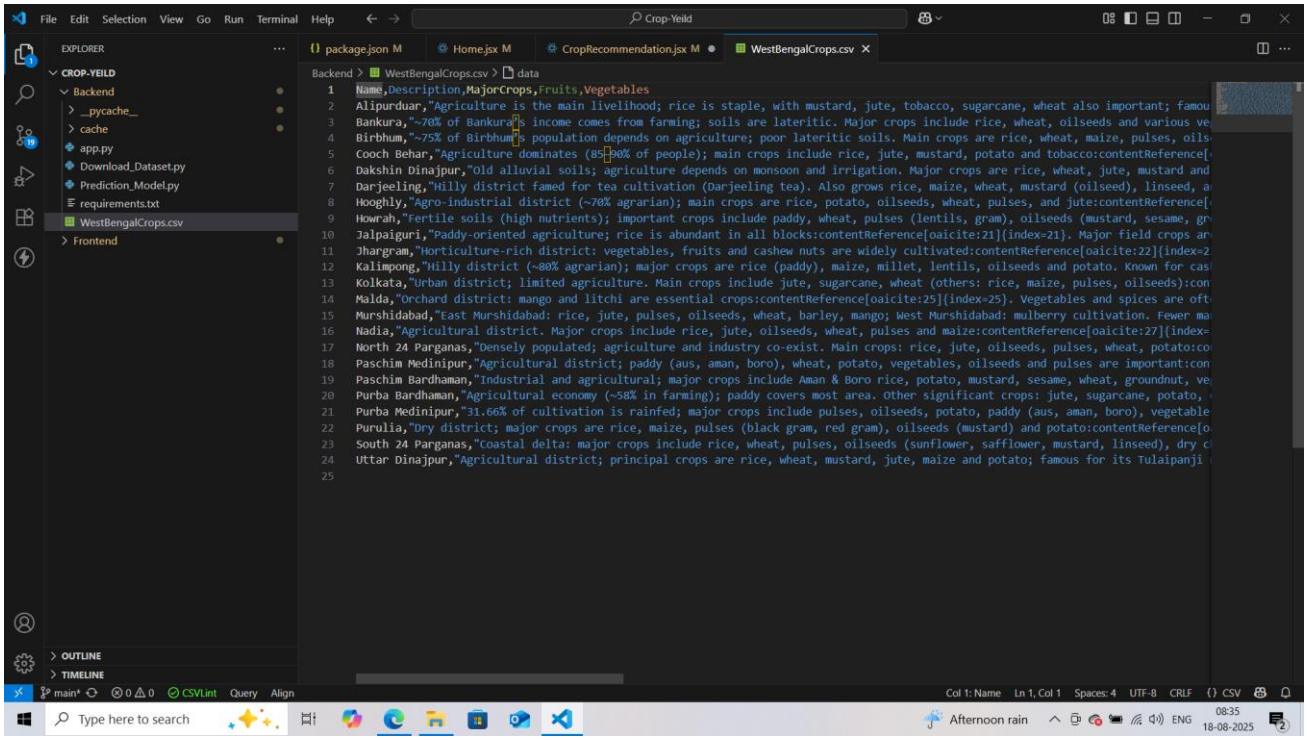
Flask API running and prediction via Postman.



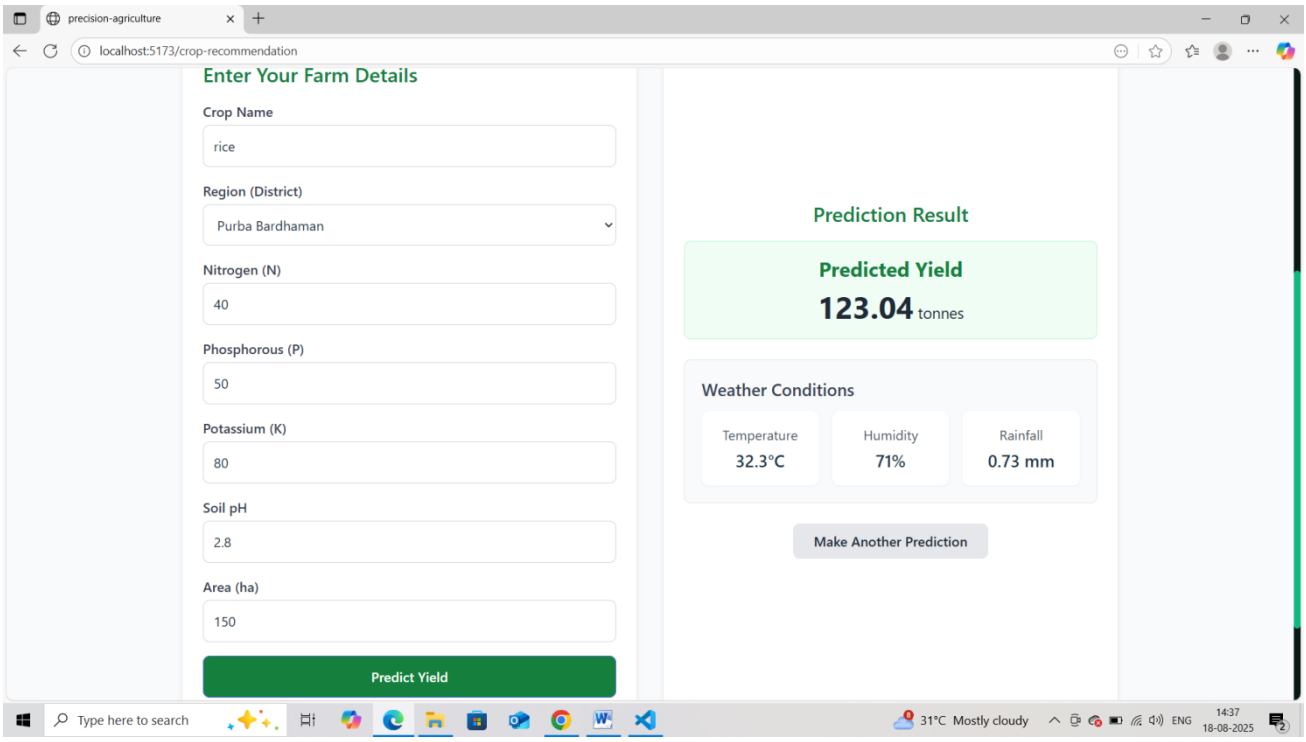
Model training output (R² and RMSE values).



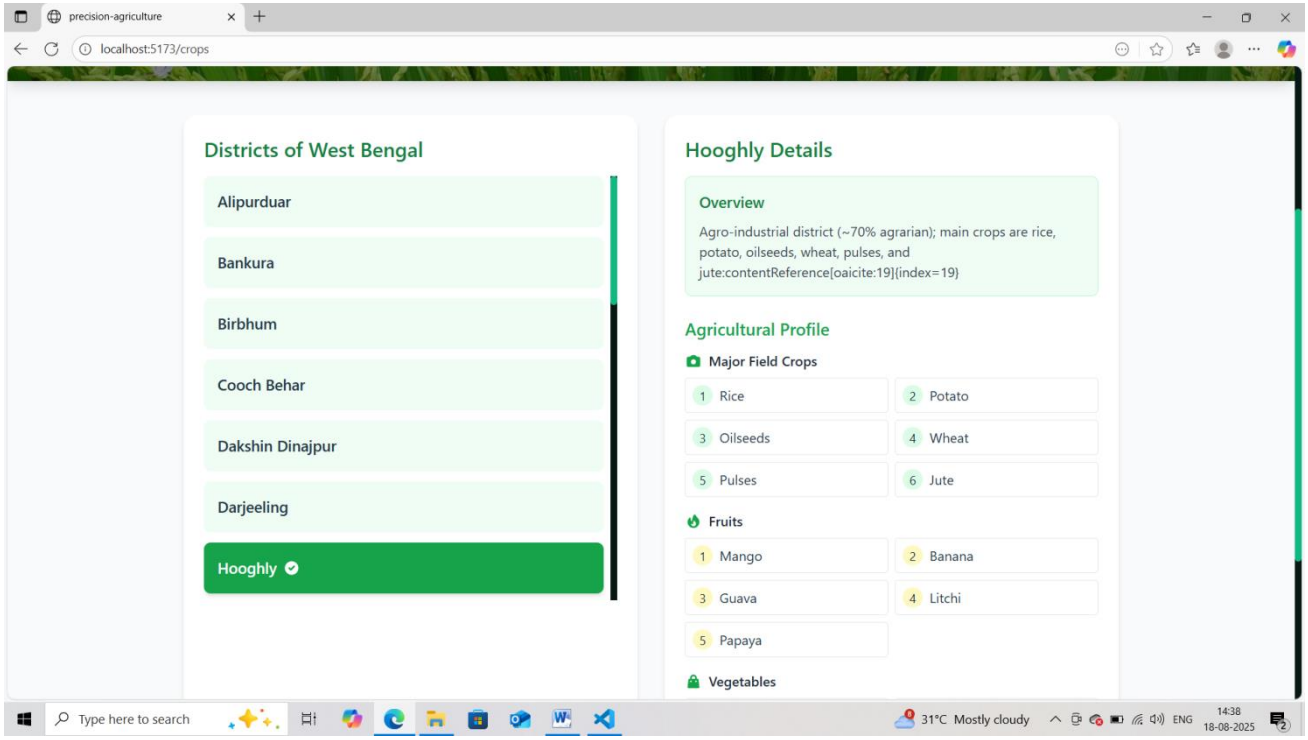
dataset view (CSV file).



dataset view (CSV file).



Crop Recommendation Based on Weather Conditions of a District.



Frontend input form and displayed prediction.

5. FUTURE SCOPE

Machine learning's capacity to forecast crop production is demonstrated by the present system. But there are still a number of avenues for future development and growth, such as:

1. Incorporation of Additional Meteorological Factors: The incorporation of characteristics like soil moisture, sunshine hours, and wind speed might increase the precision of predictions.
2. Using Modern Models: In order to identify temporal and sequential trends in agricultural data, sophisticated deep learning techniques, such as LSTM networks, may be used.
3. Mobile Application: Creating a mobile app version would improve farmers' access to the system, especially in rural regions.
4. Recommendations for Multiple Crops: In addition to predicting output, the model may be expanded to advise the best crop for a particular season and location.
5. Geospatial Data Integration: Integrating GIS (Geographical Information Systems) data and satellite imagery would enable very localized and area-specific forecasts.
6. Widespread Implementation: Government organizations or NGOs may utilize the system for large-scale food security planning and distribution at the national level.

6. LIMITATIONS AND RISK FACTORS

Although the suggested system offers useful forecasts for crop yield, there are some restrictions:

1. Dataset Size and Coverage: The dataset utilized for training has a limited number of samples and geographical coverage, which may prevent the generalization of predictions to larger areas.
2. Dependence on Weather Data: The system's accuracy is heavily reliant on real-time weather API data, which may occasionally be unreliable or unavailable.
3. A Reduced Feature Set: The model takes into account just a few select soil and weather characteristics. Additional variables like irrigation methods, fertilizer application, and pest invasions are not considered.
4. Model Assumptions: The Gradient Boosting model makes the assumption that there are consistent links between features and yields, but this may not be the case in the event of unexpected climate change or severe weather.
5. Scalability limitations: The system has been tested on small-scale datasets and might need to be improved for real-time integration of large-scale agricultural data.

RISK FACTORS

However, the efficacy of the suggested approach may be affected by some risk variables:

• **Risk of data availability**: Agricultural datasets may be restricted in size or location, which could make it difficult to generalize findings.
• **Weather Data Dependency**: Reliance on external weather APIs means predictions may fail if the service is inaccurate or unavailable.
• **Risk of Model Overfitting**: The model may overfit with smaller datasets and perform badly on data it hasn't seen before.
• **Scalability Risk**: The system may encounter difficulties in real-time, large-scale applications, even if it works well on small datasets.
• **User Adoption Risk**: Due to a lack of internet connectivity or technical literacy, farmers may have problems using the system.
• **Climate and Environmental Risk**: Extreme climate events, pest infestations, or catastrophes may affect yields in ways that the model cannot predict.

7. CONCLUSION

Using soil and weather data, this study showed how machine learning methods, particularly the Gradient Boosting Regressor (GBR), can be used to forecast crop yields. The model's ability to capture intricate, nonlinear linkages between agricultural characteristics was demonstrated by its excellent accuracy, which was measured by an R2 score of around 0.98.

The system was effectively implemented using a React frontend and a Flask backend, allowing for the real-time integration of live weather data and user inputs. Because of this practical application, farmers and officials may use the system to make educated choices on crop planning and food distribution.

The project emphasized the promise of machine learning in agriculture, but it also recognized its drawbacks, such as its dependence on weather APIs, feature simplification, and dataset size. By including bigger datasets, more sophisticated deep learning models, geographic information, and mobile-based deployment in future studies, these restrictions may be overcome, increasing access for farmers in rural areas.

This study, in conclusion, lays a solid groundwork for the use of data-driven methods in agriculture, showing that machine learning may make a major contribution to sustainable farming methods, food security, and efficient agricultural planning.

8. REFERENCES

[1] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.

[2] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5), 1189–1232.

[3] Meena, H. M., & Singh, J. P. (2020). Crop yield prediction using machine learning algorithms: A case study of India. International Journal of Computer Applications, 975, 8887.

[4] Shankar, S., & Rajesh, R. (2019). Weather-based crop yield prediction using machine learning techniques.International Journal of Innovative Technology and Exploring Engineering (IJITEE), 8(7), 1169–1173.

[5] Kaggle. (2023). Crop yield prediction dataset. Retrieved from https://www.kaggle.com/

[6] WeatherAPI. (2023). Weather data API for developers. Retrieved from https://www.weatherapi.com/

[7] ChatGPT A.I tools for coding reference https://chatgpt.com/

THANK YOU