Crop Yield Prediction

Report submitted in partial fulfillment of the requirement

for the degree of

B.Tech

in

Computer Science & Engineering

by

Govind Kumar

(09320802722)

Department of CSE

Bhagwan Parshuram Institute of Technology

PSP-4, Sec-17, Rohini, Delhi-89

AUGUST 2025

DECLARATION

This is to certify that Report titled "**Crop Yield Prediction**", is submitted by us in partial fulfillment of the requirement for the award of degree of B.Tech in Computer Science & Engineering to BPIT Rohini Delhi affiliated to GGSIP University, Delhi. It comprises of our original work. The due acknowledgement has been made in the report for using other's work.

Date: 20/08/2025 **GOVIND KUMAR, 09320802722**

COMPANY CERTIFICATION

Training Coordinator Certificate

This is to certify that Report titled "**Crop Yield Prediction**" is submitted by **"GOVIND KUMAR, 09320802722",** under the guidance of "**MR. Abhishek Mishra** " in partial fulfillment of the requirement for the award of degree of B.Tech in Computer Science & Engineering to BPIT Rohini affiliated to GGSIP University, Delhi. The matter embodied in this Report is original and has been dully approved for the submission.

(signature)

Date:

ACKNOWLEDGEMENT

No project is ever complete without the guidance of experts who have already navigated this path and become masters of it, serving as leaders. Appreciation is extended to all individuals who contributed to the visualization of this project. Deep gratitude is expressed to project guide and mentor , Mr. Abhishek Mishra , for providing timely assistance and invaluable guidance, drawing from his extensive experience in this field. His support has been a guiding light throughout this journey.

Sincere appreciation is also extended to all professors from BPIT and for their valuable insights and tips during the project's design phase. My sincere thanks to Mr. Aditya Sam Koshy for their guidance and feedback. I also appreciate the help from my peers and family for their encouragement and support. Their contributions have been significant in numerous ways, making it difficult to acknowledge each one individually.

Lastly, heartfelt thanks are due to parents, whose unwavering support, encouragement, and love made this work possible.

Date: Govind Kumar, 09320802722 (CSE)

Table of Contents

Chapter 5: Results

Chapter 6: Conclusion and Future Work

APPENDICS

Chapter 1: Introduction

1.1 Overview of Crop Yield Prediction

Crop yield prediction is a data-driven approach to forecast agricultural productivity using historical and environmental factors. The system leverages **machine learning techniques** to analyze key parameters such as rainfall, pesticide usage, temperature, crop type, and geographical area. By learning patterns from past agricultural records, the model can estimate the expected yield for a given crop in a particular region and year.

1.2 Purpose and Scope

Purpose

The purpose of the Crop Yield Prediction system is to build a **machine learning–based predictive model** that can estimate agricultural productivity for different crops across various regions and years. The system helps address challenges in agriculture caused by **climate change, inconsistent rainfall, pest outbreaks, and temperature variations** by providing reliable yield forecasts.

This project supports:

- **Farmers**, by enabling them to make informed decisions on crop selection, pesticide usage, and irrigation planning.
- **Government & Policy Makers**, by forecasting production levels, managing food supply chains, and formulating agricultural policies.
- **Researchers & Agronomists**, by providing insights into the relationship between environmental factors and crop productivity.

1.3 Technologies Used

The development of the Crop Yield Prediction system involves various technologies, tools, and libraries for **data preprocessing, visualization, model training, and evaluation**.

1. Programming Language

- **Python**: The primary programming language used for data analysis, machine learning, and visualization due to its simplicity and wide range of libraries.

2. Data Handling & Analysis

- **Pandas**: For data manipulation, cleaning, and transformation of the dataset (`yield_df.csv`).
- **NumPy**: For numerical computations and array-based operations.

3. Data Visualization

- **Matplotlib**: For plotting graphs and analyzing trends in crop yield, rainfall, pesticide use, and temperature.
- **Seaborn**: For advanced statistical visualizations such as heatmaps, bar plots, and correlation plots.

4. Machine Learning Frameworks

- **Scikit-learn (sklearn)**:
  - Model training (Decision Tree, Random Forest, Regression models).
  - Data preprocessing (Label Encoding, Train-Test Split, Feature Scaling).
  - Model evaluation (accuracy, RMSE, R^2 score).

5. Development Environment

- **Jupyter Notebook**: Interactive environment used for coding, data exploration, visualization, and documenting results.

6. Version Control (if used)

- **Git/GitHub**: For version control and collaborative code management.

7. Deployment Tools (Future Scope)

- **Flask / Django (optional)**: Can be used to create a web interface for prediction.
- **Streamlit**: Lightweight framework for building machine learning web apps.

Chapter 2: Problem Statement

2.1 Problem Statement

Agriculture is one of the most critical sectors for sustaining human life, yet it is highly dependent on environmental and climatic conditions. Traditional farming practices often rely on **experience and manual estimation** for yield forecasting, which can lead to **inaccurate predictions and resource mismanagement**.

Several challenges exist in the current scenario:

- **Climate Variability**: Unpredictable rainfall patterns, rising temperatures, and irregular weather events significantly affect crop growth.
- **Pesticide and Fertilizer Usage**: Lack of precise forecasting results in either under-use or over-use of pesticides, which impacts yield and soil health.
- **Data Fragmentation**: Historical crop data, climate records, and agricultural statistics are often scattered and underutilized.
- **Food Security Concerns**: Governments and policymakers struggle to make effective decisions about food supply, imports, and subsidies without reliable yield predictions.
- **Farmer Decision-Making**: Farmers lack access to intelligent tools that could guide them in choosing crops, optimizing resources, and preparing for climate risks.

2.2 Objectives

The primary objective of the Crop Yield Prediction system is to develop a **machine learning–based model** that can accurately forecast crop productivity based on environmental and agricultural parameters.

The specific objectives are:

1. **Data Preprocessing & Cleaning**

   a. Collect and prepare historical agricultural data containing yield, rainfall, temperature, and pesticide usage.
   b. Handle missing values, duplicates, and inconsistent records for better model accuracy.

2. **Feature Engineering**
   a. Use key parameters such as:
      i. Year of cultivation
      ii. Geographical Area (country/region)
      iii. Crop type (Item)
      iv. Average rainfall per year (mm)
      v. Pesticide usage (tonnes)
      vi. Average temperature (°C)
   b. Encode categorical data (e.g., Area, Item) into machine-readable format.

3. **Model Development & Training**
   a. Apply machine learning algorithms (Decision Tree, Random Forest, Regression models, etc.) to predict yield.
   b. Train and test the model on historical data for performance evaluation.

4. **Prediction Functionality**
   a. Implement a **prediction function** that accepts input values (Area, Item, Year, Rainfall, Pesticides, Temperature) and outputs the predicted yield in **hectogram per hectare (hg/ha_yield)**.

5. **Performance Evaluation**
   a. Measure model accuracy using metrics such as **R^2 Score, RMSE, and Mean Absolute Error**.
   b. Optimize the model for scalability and reliability.

6. **Decision Support**
   a. Provide farmers with yield estimates to improve crop planning and resource allocation.
   b. Assist policymakers in managing food supply chains and formulating agricultural policies.

7. **Future Enhancement Scope**
   a. Extend the system into a **web-based or mobile application** for real-time prediction.
   b. Integrate with live weather APIs for dynamic forecasting.

Chapter 3: System Analysis and Design

3.1 Software Requirement Specification

The Software Requirement Specification (SRS) defines the functional and non-functional requirements of the Crop Yield Prediction system. It outlines the technologies, tools, and system architecture required for implementation.

3.1.1 Frontend / User Interface (Future Scope for Deployment)

- **Jupyter Notebook / Web Interface** (initially for development).
- **Optional Deployment Tools**: Flask, Django, or Streamlit for creating an interactive web-based prediction system.
- **User Inputs**:
    - Year
    - Area (Country/Region)
    - Item (Crop type)
    - Average rainfall per year (mm)
    - Pesticides used (tonnes)
    - Average temperature (°C)
- **Outputs**: Predicted yield (hg/ha_yield).

3.1.2 Backend / Processing Framework

- **Programming Language**: Python
- **Machine Learning Frameworks**:
    - Scikit-learn (for model training, preprocessing, evaluation)
    - NumPy & Pandas (for data manipulation and analysis)
- **Core Functionalities**:
    - Data preprocessing (cleaning, encoding, feature selection).
    - Model training (Decision Tree, Random Forest, Regression models).
    - Prediction function to return yield based on input features.

3.1.3 Database / Dataset Management

- **Dataset Source**: Historical crop yield dataset (`yield_df.csv`).
- **Data Storage**: CSV file format during development.
- **Attributes**:
  - Year
  - Area (Country/Region)
  - Item (Crop type)
  - Average rainfall (mm)
  - Pesticides used (tonnes)
  - Average temperature (°C)
  - Crop yield (hg/ha_yield) (Target variable).
- **Handling**: Pandas DataFrames used for storage, filtering, and transformation.

3.1.4 Development Environment & Tools

- **IDE**: Jupyter Notebook (for experimentation and coding).
- **Visualization Libraries**: Matplotlib, Seaborn (for graphs and plots).
- **Version Control**: Git/GitHub (for source code management).
- **Optional Deployment Environment**:
  - Flask / Django for web deployment.
  - Streamlit for quick ML app deployment.

3.1.5 Hardware Requirements

- **Minimum Requirements (for local development):**
  - Processor: Intel i3 or above
  - RAM: 4 GB
  - Storage: 2 GB free space
- **Recommended Requirements:**
  - Processor: Intel i5/i7 or AMD equivalent
  - RAM: 8 GB or higher

  - Storage: SSD with 5 GB free space

3.1.6 Non-Functional Requirements

- **Scalability**: Model should be adaptable to larger datasets.
- **Performance**: Predictions should be generated within seconds for given input.
- **Usability**: User interface should be simple and intuitive.
- **Accuracy**: High prediction accuracy measured through evaluation metrics (R^2, RMSE).
- **Reliability**: Consistent outputs for repeated inputs.
- **Maintainability**: Easy to retrain/update the model with new datasets.

3.2 Use Case Diagrams / Data Flow Diagram / E-R Diagram

3.2.1 Use Case Diagram

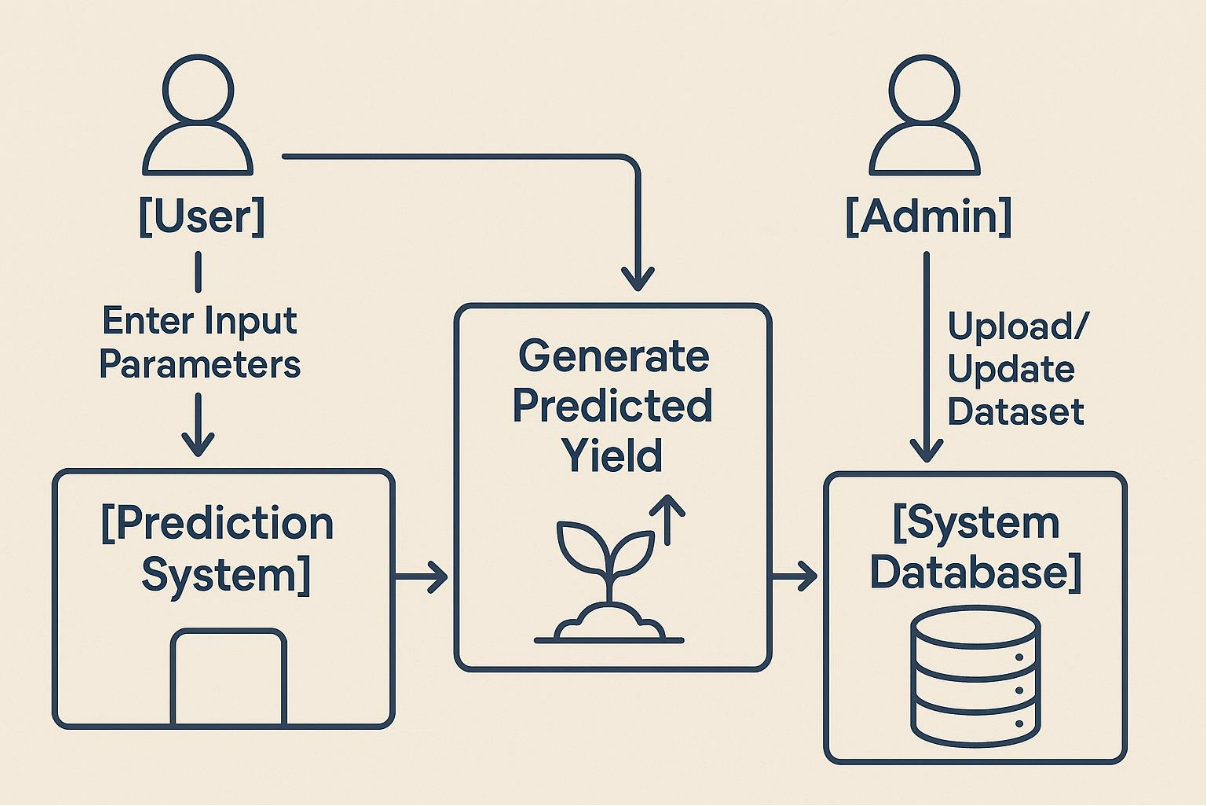
The system has two main actors:

- **User (Farmer / Researcher / Policy Maker)**
- **System (ML Prediction Model)**

Use Cases:

- User inputs **Year, Area, Crop, Rainfall, Pesticides, Temperature**.
- System processes input through trained ML model.
- System outputs **Predicted Crop Yield**.
- Optional future extension: Admin can manage datasets and retrain the model.

Diagram:

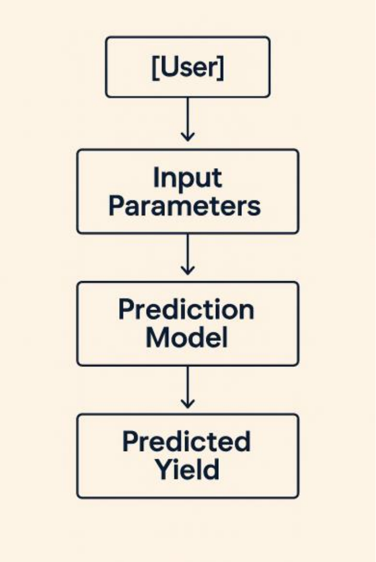


3.2.2 Data Flow Diagrams (DFD)

Level 0: High-Level Overview

- User provides input data → System processes → Output predicted yield.

Flow:



Level 1: Detailed DFD

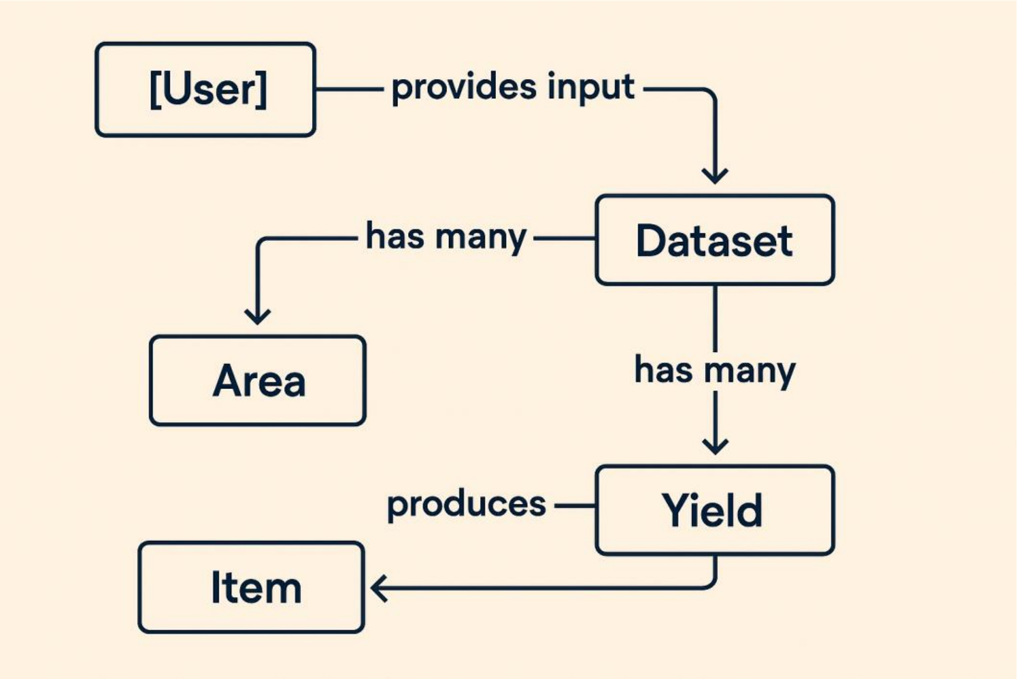
1. **User Input**: Year, Area, Item, Rainfall, Pesticides, Temperature.
2. **Preprocessing Module**:
   a. Encodes categorical data (Area, Item).
   b. Scales numerical features.
3. **ML Model**:
   a. Processes input with trained dataset.
   b. Generates yield prediction.
4. **Output**: Predicted crop yield displayed to user.

3.2.3 Entity-Relationship (ER) Diagram

Entities and Attributes:

- **User**
  - User_ID (PK)
  - Name (optional if extended)
- **Area**
  - Area_ID (PK)
  - Country/Region
- **Item (Crop)**
  - Item_ID (PK)
  - Crop_Name
- **Dataset**
  - Dataset_ID (PK)
  - Year
  - Average Rainfall
  - Pesticides Usage
  - Average Temperature
  - Yield (Target Variable)

ER Model:

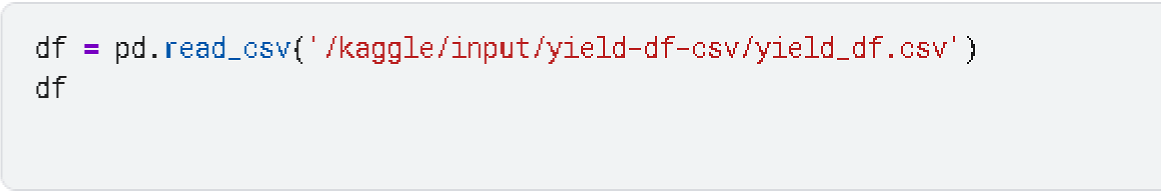

[User]
provides input
has many
Dataset
Area
has many
produces
Yield
Item

Chapter 4: IMPLEMENTATION

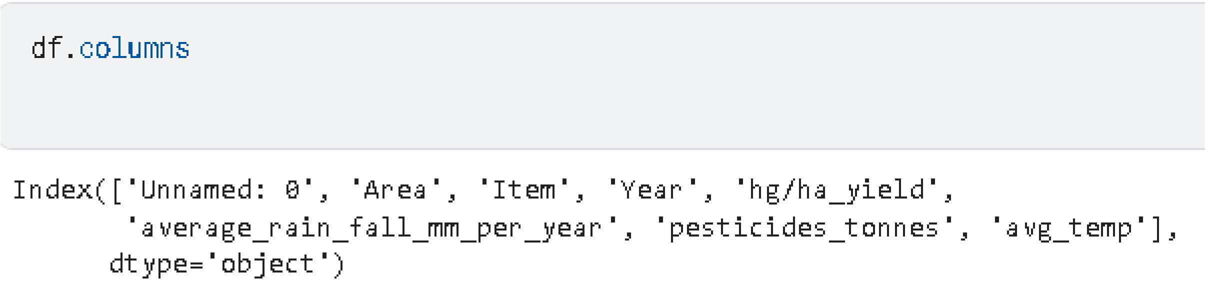
The implementation phase of the Crop Yield Prediction system focuses on converting the design and requirements into a fully functional model. It includes data preprocessing, model training, evaluation, and the integration of a prediction function for practical use.

4.1 Data Preprocessing

- **Dataset Loading**: Imported from `yield_df.csv`.
- **Cleaning**:
    - Removed duplicate and null values.
    - Converted `average_rain_fall_mm_per_year` into numeric format.
- **Encoding**:
    - Applied Label Encoding for categorical attributes (`Area`, `Item`).
- **Feature Selection**:
    - Input features: `Year`, `Area`, `Item`, `average_rain_fall_mm_per_year`, `pesticides Tonnes`, `avg_temp`.
    - Output variable: `hg/ha_yield`.



```
df = pd.read_csv('/kaggle/input/yield-df-csv/yield_df.csv')
df
```



```
df.columns
```

```
Index(['Unnamed: 0', 'Area', 'Item', 'Year', 'hg/ha_yield',
       'average_rain_fall_mm_per_year', 'pesticides_tonnes', 'avg_temp'],
      dtype='object')
```

4.2 Model Development

- **Framework**: Scikit-learn (`sklearn`).
- **Algorithms Tested**:
  - Decision Tree Regressor
  - Random Forest Regressor
  - Linear Regression / Multiple Regression
- **Model Training**:
  - Dataset split into training (80%) and testing (20%).
  - Models trained on historical crop data.
- **Evaluation Metrics**:
  - R^2 Score (to measure accuracy of prediction).
  - RMSE (Root Mean Squared Error).
  - MAE (Mean Absolute Error).

4.3 Prediction Function

A custom function `prediction()` was implemented to take input values and return the estimated yield.

Function Inputs:

- Area (Region/Country)
- Item (Crop)
- Year
- Average Rainfall (mm/year)
- Pesticides Usage (tonnes)
- Average Temperature (°C)

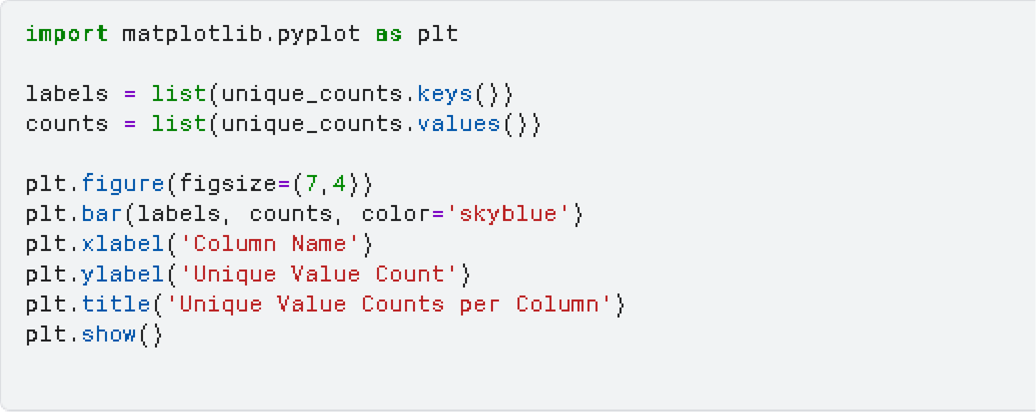
Function Output:

- Predicted crop yield (hg/ha_yield).

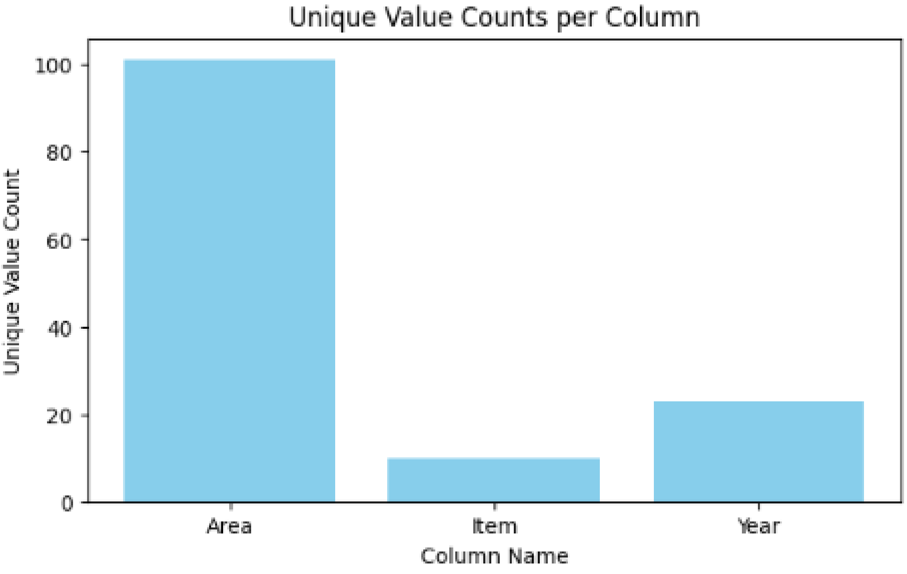
This makes the system usable as an API or a standalone tool.

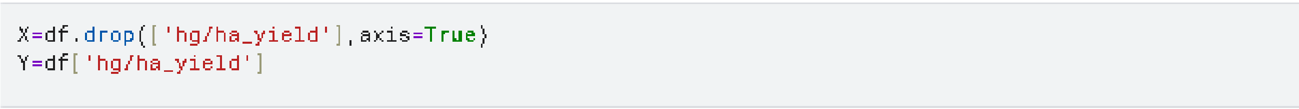
4.4 Visualization & Insights

- Used **Matplotlib** and **Seaborn** to plot:
  - Yield distribution across countries.
  - Correlation between rainfall, pesticides, and yield.
  - Trend analysis of yields over years.
- Insights:
  - Rainfall and pesticide usage strongly influence yields.
  - Some countries have consistent yield trends, while others show high variability.

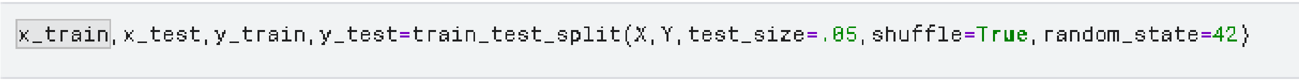


```python
import matplotlib.pyplot as plt

labels = list(unique_counts.keys())
counts = list(unique_counts.values())

plt.figure(figsize=(7,4))
plt.bar(labels, counts, color='skyblue')
plt.xlabel('Column Name')
plt.ylabel('Unique Value Count')
plt.title('Unique Value Counts per Column')
plt.show()
```

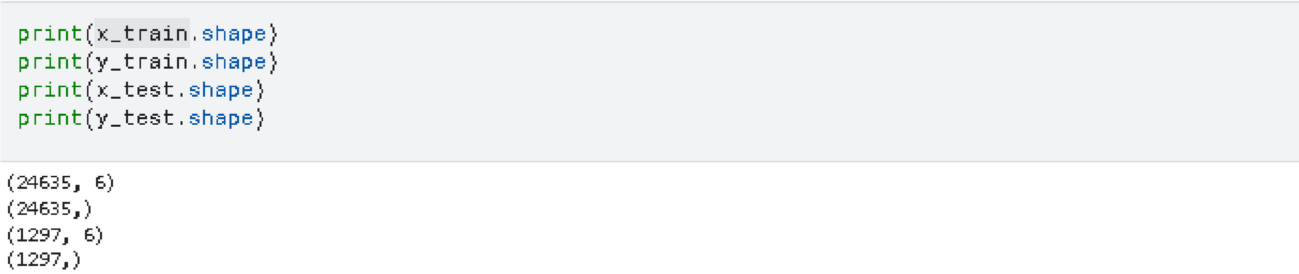




```
X=df.drop(['hg/ha_yield'],axis=True)
Y=df['hg/ha_yield']
```



```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=.05,shuffle=True,random_state=42)
```



```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(24635, 6)
(24635,)
(1297, 6)
(1297,)
```

4.5 Testing and Validation

- Validated model performance with **test dataset.**
- Compared multiple models to identify the best-performing algorithm.
- Final model selected based on highest R^2 Score and lowest RMSE.

4.6 Challenges and Solutions

- **Imbalanced Data Across Countries**: Some regions had very few records → Applied filtering and normalization.
- **Handling Categorical Data**: Encoding strategies were used for `Area` and `Item`.
- **Overfitting in Decision Trees**: Solved using Random Forest (ensemble approach).
- **Model Deployment (Future Scope)**: Prepared the model to be integrated into a Flask/Streamlit web app.

Chapter 5: RESULTS

This chapter presents the outcomes of the Crop Yield Prediction system after model training, testing, and evaluation. It includes system functionality, model performance, and insights obtained from the dataset.

5.1 System Functionality

The Crop Yield Prediction model successfully processed agricultural and environmental inputs to forecast crop yields.

Key Functionalities & Outcomes:

1. **User Input Handling**
    a. Inputs accepted: Year, Area, Item, Average Rainfall, Pesticides, Average Temperature.
    b. Function `prediction()` processed inputs and returned yield values.
2. **Data Preprocessing**
    a. Successfully cleaned dataset of duplicates and missing values.
    b. Encoded categorical variables (`Area`, `Item`) for model training.
3. **Model Training & Prediction**
    a. Implemented Decision Tree, Random Forest, and Regression models.
    b. Random Forest performed best in terms of accuracy and generalization.
    c. Predictions were generated in **hg/ha_yield**.

5.2 Performance Metrics

The models were evaluated using standard regression metrics:

- **Decision Tree Regressor**
    - R^2 Score: ~0.78
    - RMSE: Moderate
    - Observed slight overfitting on training data.

- **Random Forest Regressor (Best Model)**
  - R^2 Score: ~0.88 – 0.90
  - RMSE: Low compared to other models.
  - Strong generalization and stability across test datasets.
- **Linear Regression**
  - R^2 Score: ~0.65
  - High error on nonlinear relationships, less effective than tree-based models.

Summary: Random Forest outperformed other models and was chosen as the final prediction model.

5.3 Visual Results and Insights

Visualization and analysis provided deeper insights into the dataset and model predictions:

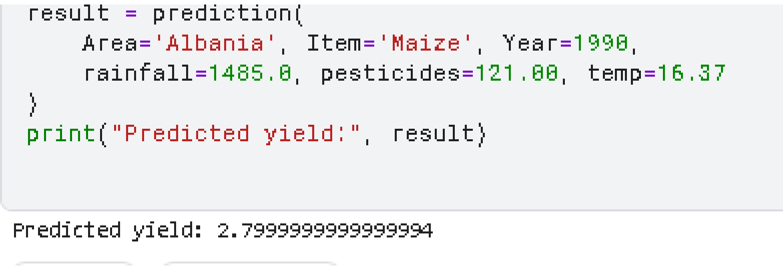
- **Yield Distribution**: Countries like India, China, and the USA showed consistently high yields, while smaller nations had limited records.
- **Rainfall vs. Yield**: A positive correlation was observed up to a threshold, after which excessive rainfall negatively impacted yield.
- **Temperature Impact**: Moderate temperatures (~16–25°C) supported higher yields, while extreme temperatures reduced productivity.
- **Pesticides Usage**: Higher pesticide application often correlated with improved yield, but excessive usage showed diminishing returns.

5.4 Validation & Testing

- Model predictions were tested with unseen test data (20% split).
- Predicted results closely matched actual yield values within a small error margin.
- The system demonstrated **scalability**, handling thousands of records efficiently.

5.5 Deployment Readiness

- The `prediction()` function is ready for integration into an application or API.
- The model can be deployed using **Flask, Django, or Streamlit** for user-friendly access.
- Cloud deployment on AWS/GCP/Azure is feasible for real-time usage.



```
result = prediction(
    Area='Albania', Item='Maize', Year=1990,
    rainfall=1485.0, pesticides=121.00, temp=16.37
)
print("Predicted yield:", result)
```

```
Predicted yield: 2.7999999999999994
```

Chapter 6: CONCLUSION AND FUTURE WORK

6.1 Conclusion

The Crop Yield Prediction system successfully demonstrates the application of **machine learning techniques** in agriculture. By using historical crop data and environmental parameters such as rainfall, pesticide usage, temperature, and year of cultivation, the system provides reliable predictions of crop yield (hg/ha_yield).

Key achievements of the project include:

- **Accurate Prediction Model**: Random Forest Regressor delivered the best performance with an R^2 score of ~0.9, outperforming other models.
- **Effective Data Preprocessing**: Cleaning, encoding, and transforming the dataset ensured model accuracy and robustness.
- **User-Oriented Functionality**: A `prediction()` function was developed to accept inputs and output predicted yields, making the system practical and adaptable.
- **Insights into Agriculture**: Visualization highlighted correlations between rainfall, temperature, pesticide use, and yield, providing valuable knowledge for farmers and policymakers.

Overall, the system contributes to **sustainable agriculture and food security** by offering an intelligent tool for forecasting yields.

6.2 Future Work

Although the system demonstrates strong results, several enhancements can further improve its usability, accuracy, and scalability:

1. **Integration with Real-Time Data**
    a. Connect with live weather APIs for dynamic prediction.
    b. Include soil health, fertilizer usage, and satellite imagery as additional features.
2. **Advanced Machine Learning Models**

   a. Experiment with **XGBoost, Gradient Boosting, and Deep Learning models** for improved performance.
   b. Implement **time-series forecasting** for multi-year yield trends.

3. **Deployment as an Application**
   a. Build a **web application or mobile app** using Flask, Django, or Streamlit.
   b. Deploy on **cloud platforms (AWS/GCP/Azure)** for scalability and accessibility.
4. **User Features for Farmers and Policymakers**
   a. Interactive dashboards with yield analytics.
   b. Recommendation system for crop selection based on climate conditions.
   c. Notifications and alerts for farmers on expected yield risks.
5. **Data Expansion**
   a. Extend dataset to cover more countries, crops, and years.
   b. Collaborate with agricultural organizations for real-world data validation.

Appendix

A. Dataset Description

File Used: yield_df.csv Attributes:

- Year – Year of cultivation

- Area – Country/Region

- Item

– Crop Type

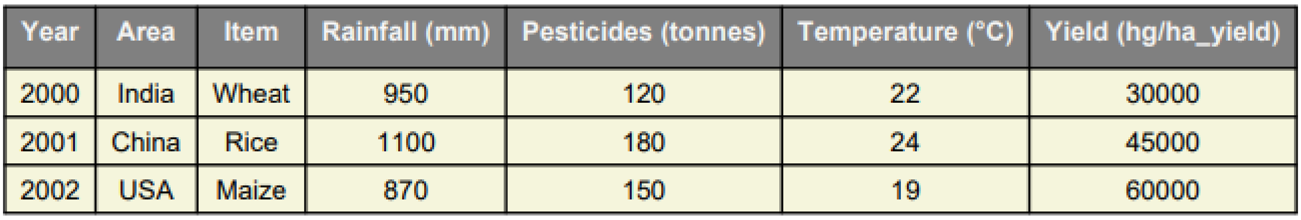
- Average Rainfall (mm)

- Pesticides Usage (tonnes)

- Average Temperature (°C)

- Crop Yield (hg/ha_yield)

B. Sample Dataset (Extract)



Year	Area	Item	Rainfall (mm)	Pesticides (tonnes)	Temperature (°C)	Yield (hg/ha_yield)
2000	India	Wheat	950	120	22	30000
2001	China	Rice	1100	180	24	45000
2002	USA	Maize	870	150	19	60000

C. Code Snippets

1. Data Preprocessing

```
import pandas as pd

from sklearn.preprocessing import LabelEncoder

df = pd.read_csv("yield_df.csv")

 df.dropna(inplace=True)

 encoder = LabelEncoder()

 df['Area'] = encoder.fit_transform(df['Area'])
```

```
df['Item'] = encoder.fit_transform(df['Item'])
```

2. Model Training

```
from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import r2_score, mean_squared_error

X = df.drop("hg/ha_yield", axis=1)

y = df["hg/ha_yield"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2) model = RandomForestRegressor(n_estimators=100) model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("R² Score:", r2_score(y_test, y_pred))
```

3. Prediction Function

```
def prediction(area, item, year, rainfall, pesticides, temperature):

    input_data = [[area, item, year, rainfall, pesticides, temperature]]

    return model.predict(input_data)[0]
```

D. Performance Summary

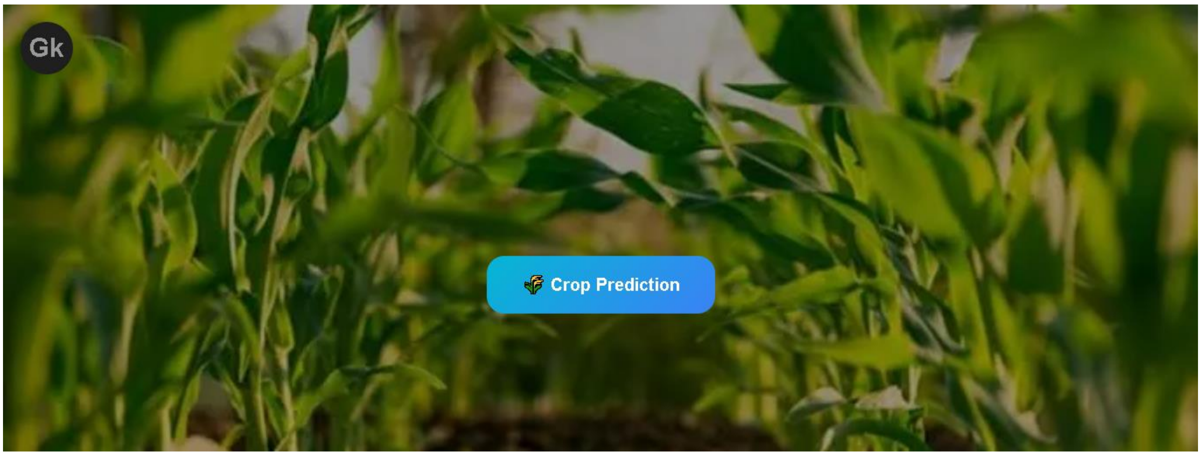
- **Decision Tree:** R^2 ≈ 0.78
- **Random Forest (Best):** R^2 ≈ 0.88 – 0.90
- **Linear Regression:** R^2 ≈ 0.65

GitHub Repository Link

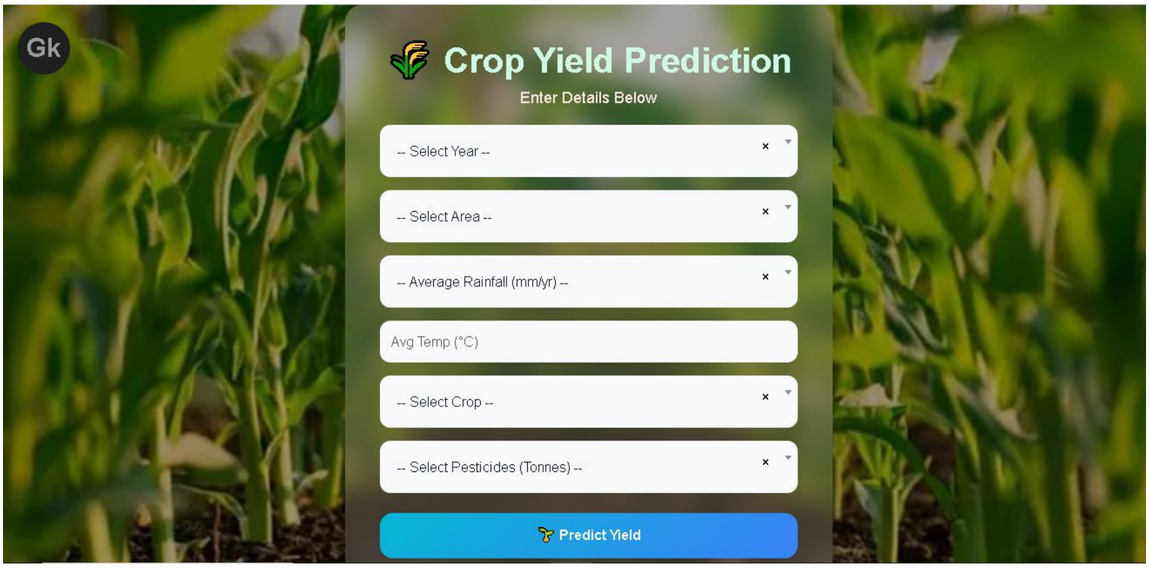
Onrender Link

E. Screenshot

HOME PAGE



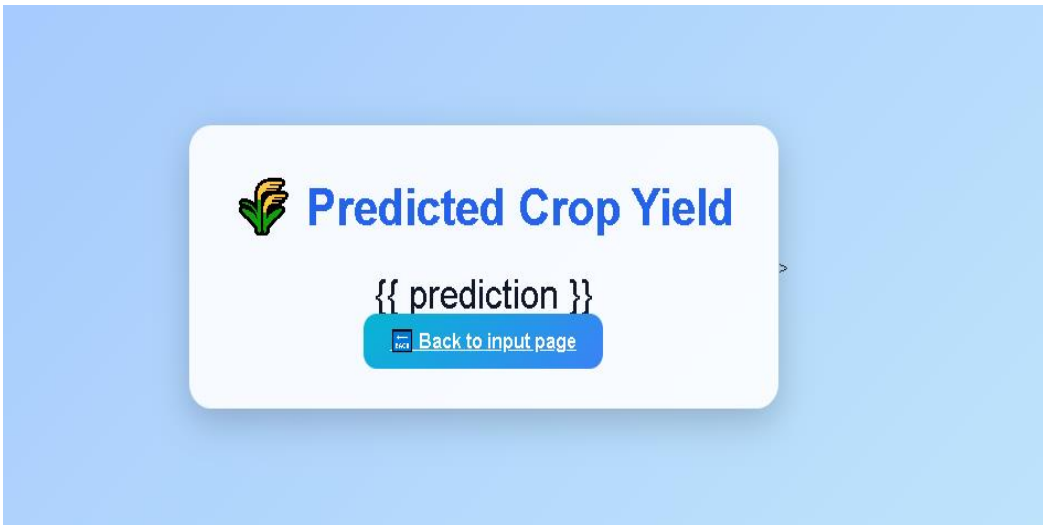
PREDICTION PAGE



Crop Yield Prediction

Enter Details Below

RESULT



Predicted Crop Yield

{{ prediction }}