** A**

**Assesment Report**

on

**“Crop Recommendation System”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**Artificial Intelligence**

By

Shaurya Pratap Singh(202401100300229)

Tushar Sharma(202401100300266)

Yatharth(202401100300287)

Shashank(202401100300225

**Under the supervision of**

“Mr. Abhishek Shukla”

**KIET Group of Institutions, Ghaziabad**

Affiliated to

**Dr. A.P.J. Abdul Kalam Technical University, Lucknow**

(Formerly UPTU)

**May, 2025**

**Problem Statement**

**Problem Overview:**

Farmers often face difficulty choosing the right crop to cultivate due to changing soil and climate conditions. Wrong crop choices can lead to low productivity and financial losses.

**Objective:**

To develop a **machine learning-based system** that recommends the most suitable crop based on real-time and historical data. The system should help farmers make **data-driven decisions** for better yield and profitability.

**Input Features:**

The model will take inputs like:

* **Soil Nutrients**: Nitrogen (N), Phosphorus (P), Potassium (K)
* **Environmental Conditions**: Temperature, Humidity, Rainfall
* **Soil pH**

**Output:**

The system will predict and recommend the **best crop** to grow under the given conditions (e.g., Rice, Wheat, Sugarcane, etc.).

**METHODOLOGY**

1. Dataset Collection:

* The dataset used in this project is typically sourced from agricultural research datasets available online (e.g., Kaggle, open government repositories).
* The dataset is provided as a .zip file and contains a .csv file with labeled crop data.

1. Data Loading and Extraction:

* The dataset ZIP file is uploaded and extracted using Python's zipfile module.
* Pandas is used to load the CSV data into a DataFrame for analysis and model development.

1. Data Exploration and Understanding:

* Dataset shape, column names, and missing values are analyzed using DataFrame functions.
* Descriptive statistics (mean, standard deviation, etc.) are reviewed using df.describe().
* This step ensures we understand the distribution and range of values in the dataset.

1. Data Visualization:

* A heatmap of feature correlations is generated using seaborn’s heatmap() function.
* Non-numeric columns like the target label (e.g., 'label' or 'crop') are excluded from correlation computation.
* This helps identify which features are strongly related and may influence model accuracy.

1. Data Preprocessing:

* Features (X) include numeric inputs like N, P, K, temperature, humidity, pH, and rainfall.
* Target variable (y) is the crop label (a categorical string value).
* The dataset is split into training and testing sets (typically 80/20) using train\_test\_split from sklearn.

1. Model Selection and Training:

* A Random Forest Classifier is selected for its robustness, accuracy, and ability to handle multi-class classification problems.
* The model is trained using the training set (X\_train, y\_train).

1. Model Evaluation:

* Predictions are made on the test set using the trained model.
* Accuracy and classification report (precision, recall, F1-score) are generated using sklearn’s metrics module.
* This helps assess how well the model can generalize to unseen data.

1. Making Predictions:

* A sample input (a list of values representing N, P, K, temperature, humidity, pH, and rainfall) is passed to the model.
* The model returns the name of the most suitable crop for those conditions.

1. Model Persistence (Optional):

* The trained model is saved to a file using joblib, allowing reuse without retraining.
* This step is useful for deploying the model in a web or mobile app.

**CODE**

**1. Upload & Extract Dataset ZIP**

**# Import necessary libraries**

**from google.colab import files**

**import zipfile**

**import os**

**# Upload ZIP file containing the dataset**

**uploaded = files.upload()**

**# Extract the ZIP file**

**for fn in uploaded.keys():**

**if fn.endswith(".zip"):**

**with zipfile.ZipFile(fn, 'r') as zip\_ref:**

**zip\_ref.extractall("crop\_dataset") # Extract to a folder**

**print("Files extracted to 'crop\_dataset'")**

**2. Load Dataset**

**import pandas as pd**

**# Get the CSV file name from the extracted folder**

**data\_path = "crop\_dataset"**

**csv\_files = [f for f in os.listdir(data\_path) if f.endswith('.csv')]**

**csv\_path = os.path.join(data\_path, csv\_files[0])**

**# Load the dataset using pandas**

**df = pd.read\_csv(csv\_path)**

**# Display first few rows**

**df.head()**

**3. Explore Dataset**

**# Print basic info about the dataset**

**print("Dataset shape:", df.shape)**

**# Show column names**

**print("\nColumns:\n", df.columns)**

**# Check for missing values**

**print("\nMissing values:\n", df.isnull().sum())**

**# Summary statistics**

**df.describe()**

**4. Visualize Feature Correlation**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**# Drop non-numeric column (e.g., target variable 'label')**

**numeric\_df = df.drop(columns=['label']) # Replace 'label' if your target column has a different name**

**# Create correlation heatmap**

**plt.figure(figsize=(10, 6))**

**sns.heatmap(numeric\_df.corr(), annot=True, cmap='YlGnBu')**

**plt.title("Feature Correlation Heatmap")**

**plt.show()**

**5. Preprocess & Split Dataset**

**from sklearn.model\_selection import train\_test\_split**

**# Separate features and target variable**

**X = df.drop('label', axis=1) # Features**

**y = df['label'] # Target (crop name)**

**# Split data into training and testing sets (80% train, 20% test)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**6. Train Classifier (Random Forest)**

**from sklearn.ensemble import RandomForestClassifier**

**# Create and train the Random Forest Classifier**

**model = RandomForestClassifier(n\_estimators=100, random\_state=42)**

**model.fit(X\_train, y\_train)**

**7. Evaluate the Model**

**from sklearn.metrics import accuracy\_score, classification\_report**

**# Predict on the test set**

**y\_pred = model.predict(X\_test)**

**# Evaluate accuracy and performance**

**print("Model Accuracy:", accuracy\_score(y\_test, y\_pred))**

**print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))**

**8. Predict for New Sample Input**

**# Predict best crop for a new input sample**

**# Format: [N, P, K, temperature, humidity, ph, rainfall]**

**sample = [[90, 42, 43, 20.5, 82.0, 6.5, 200.0]]**

**predicted\_crop = model.predict(sample)**

**print("Recommended Crop:", predicted\_crop[0])**

**9. Save the Trained Model (Optional)**

**import joblib**

**# Save the model to a file**

**joblib.dump(model, 'crop\_recommendation\_model.pkl')**

**print("Model saved as 'crop\_recommendation\_model.pkl'")**

**OUTPUT / RESULT**

A black screen with white text

AI-generated content may be incorrect.

**REFERENCES / CREDITS**

**1. Ian Goodfellow et al. – Deep Learning (MIT Press, 2016)  
2. Andrew Ng – Deep Learning Specialization (Coursera)  
3. GeeksforGeeks – CNN and AI concept articles   
4. Analytics Vidhya – Beginner-friendly ML and CNN tutorials  
5. TensorFlow & PyTorch official documentation  
6. Yann LeCun et al. – Foundational paper on CNNs (1998)**purposes. It includes anonymized customer behavior attributes such as browsing activity, purchase frequency, and spending patterns.