A PROJECT REPORT

On

CROP RECOMMENDATION SYSTEM

Submitted by (MCA,4th Sem)

SIDDHARTHA KHAWAS (10871022006)

SUBHAM CHAKRABORTY (10871022007)

SUMIT KUMAR CHOUBEY(10871022006)

SHIVENDU SHIVAM (10871022051)

Under the Guidance of

Dr. Arnab Chakraborty

Asansol Engineering College, Asansol

Affiliated to MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

(MAY,2024)

Contents

ACKNOWLEDGEMENT

We extend our heartfelt gratitude to our project guide, **Dr. Arnab Chakraborty**, for his invaluable guidance and unwavering support throughout our endeavor on the 'Crop Recommendation System.' His expert advice and constant encouragement were pivotal in helping us navigate the challenges of this project. We are immensely thankful for his mentorship and the wealth of knowledge he shared with us.

Additionally, we would like to express our sincere appreciation to all the team members who dedicated their time, effort, and expertise to the success of this project. Each member's unique contributions played a crucial role in bringing our vision to fruition. Together, we overcame obstacles, collaborated effectively, and achieved our goals. Thank you for your unwavering commitment and teamwork

SIDDHARTHA KHAWAS

SUBHAM CHAKRABORTY

SUMIT KUMAR CHOUBEY

SHIVENDU SHIVAM

Project Objective

1. **Increase Yield**: Help farmers select crops that are likely to thrive in their specific area, boosting their overall harvest.
2. **Save Resources**: Recommend crops that require minimal water, fertilizer, and pesticides, reducing input costs and environmental impact.
3. **Maximize Profit:** Guide farmers towards crops with higher market demand or better prices, increasing their potential income.
4. **Promote Sustainability:** Encourage the cultivation of crops that maintain soil health and biodiversity, supporting long-term farming practices.
5. **Easy to Use:** Develop a user-friendly interface that farmers can easily navigate, making the system accessible to all levels of technical expertise.
6. **Data Security:** Protect farmers' data privacy and confidentiality, adhering to strict security measures to safeguard sensitive information.
7. **Scalability:** Design the system to accommodate future growth and increasing demand, ensuring it remains effective and efficient as more users adopt it.
8. **Continuous Improvement:** Continuously refine the recommendation algorithms based on user feedback and new data, ensuring relevance and accuracy.

Project Scope

1. **Data Collection:** Gather relevant data such as soil type, climate, historical weather patterns, crop characteristics, and market prices from reliable sources.

2. **Data Preprocessing:** Clean and preprocess the collected data to remove noise, handle missing values, and standardize formats for consistency.

3. **Feature Selection**: Identify key features that influence crop growth such as Nitrogen, Phosphorus, Potassium, Humidity, ph ,Rainfall ,Label, temperature, and crop rotation history.

4. **Model Development**: Build machine learning models to predict crop suitability and based on the selected features. Common models include decision trees, random forests, Support Vector Machine, Logistic regression.

5. **Model Training:** Train the models using historical data to learn patterns and relationships between input features and crop outcomes.

6. **Model Evaluation:** Evaluate the performance of the trained models using metrics such as accuracy, precision, recall, and F1-score to ensure reliability and effectiveness.

7. **Testing and Validation:** Conduct thorough testing and validation to ensure the system functions correctly under different scenarios and provides accurate recommendations.

Software Requirements:-

Operating System: - WINDOWS 7/8/10/11
Languages: - Python, Html, Css
Software Tools: Scikit-learn library for machine learning algorithms, Flask for web application development.

Hardware Requirements:-

Microprocessor: - Intel® Core™ i3 processors (Or equivalent)
Ram:- 4 GB
Hard Drive: - 250 GB

Data Description

Source of the data: Kaggle. The given dataset is a shortened version of the original dataset in Kaggle.

Data Description: The given train dataset has 2200 rows × 8 columns

Columns	**Attribute Name**	**Type**	**Description**	**Target Attribute**
Nitrogen	Nitrogen	Non categorical	Nitrogen is an essential macronutrient for plant function and is a key component of amino acids, which form the building blocks of plant proteins and enzymes.	No
Phosphorus	Phosphorus	Non categorical	Phosphorus (P) is essential for plant growth and is required in large amounts by plants.	No
Potassium	Potassium	Non categorical	Potassium is associated with the movement of water, nutrients and carbohydrates in plant tissue.	No
Temperature	Temperature	Non categorical	Temperature in agriculture refers to the measurement and control of temperature in agricultural settings.	No
Humidity	Humidity	Non categorical	It is the ratio of actual water vapour content to the saturated water vapour content at a given temperature and pressure expressed in percentage (%).	No
Ph	Ph	Non categorical	Soil pH or soil reaction is an indication of the acidity or alkalinity of soil and is measured in pH units.	No
Rainfall	Rainfall	Non categorical	The primary source of water for agricultural production, for large parts of the world, is rainfall or precipitation.	No
Label	Label	Categorical	The target variable which indicates the type of crop that is recommended based on the given environmental factors.	Yes

Number Of statistics of the given dataset

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.182727	52.975455	47.643182	25.363830	70.926162	6.424093	102.457727
std	37.077482	33.241552	50.630203	5.661025	23.041266	0.947815	55.638149
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	20.750000	27.000000	20.000000	22.629467	59.791831	5.955956	63.866827
50%	37.000000	51.000000	31.000000	25.549620	80.381232	6.421510	94.354745
75%	84.000000	68.000000	48.000000	28.546521	89.892282	6.922373	122.747122
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

- Count: The total number of non-missing values in each column.
- Mean: The average value of the column.
- Std (Standard Deviation): Measures the amount of variation or dispersion of the data.
- Min: The smallest value in the column.
- 25% (First Quartile): The value below which 25% of the data falls.
- 50% (Median): The middle value of the data.
- 75% (Third Quartile): The value below which 75% of the data falls.
- Max: The largest value in the column.

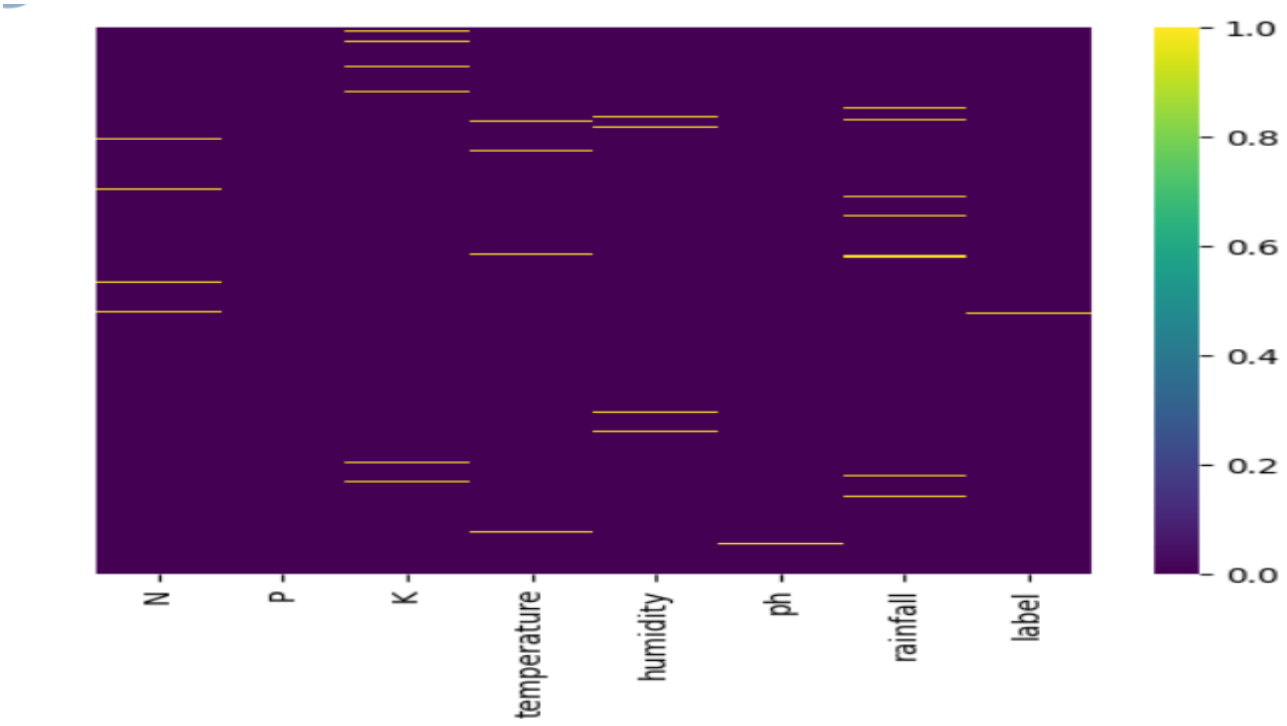
Data Pre-processing

We searched for null values in our dataset and gives you a Series where each entry represents the number of missing values in the corresponding column of the crop DataFrame. This information is useful for identifying and handling missing data in your dataset as shown in the following table:-

Column Name	**Count of Null Value**
Nitrogen	20
Phosphorus	19
Potassium	22
Temperature	26
Humidity	18
Ph	17
Rainfall	21
Label	21

Count of Null values

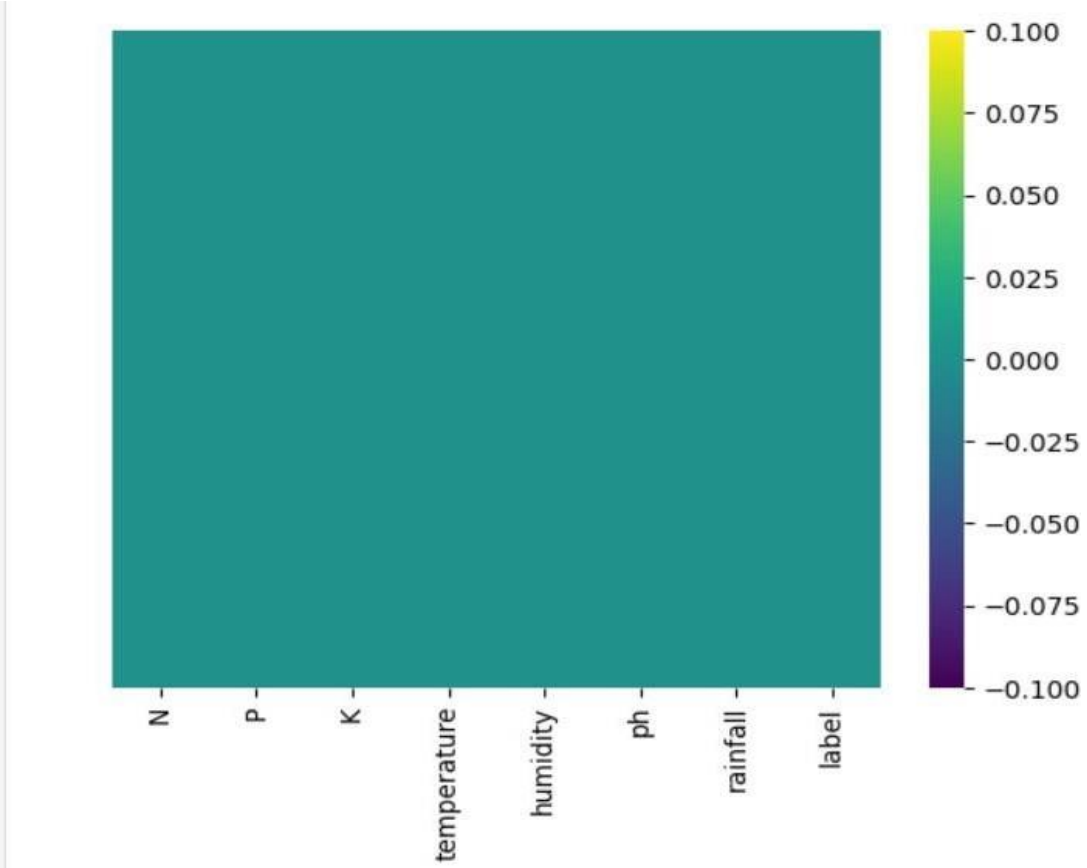
To visualise the null values, we made a heatmap plot using seaborn library function heatmap. The heatmap plot is given below:



To remove the null values, we had the following Rules:

1. **Identify Null Values:** Start by identifying where null values exist in your dataset. These null values could be represented as "NaN" (Not a Number) or other placeholders, depending on the software or tool you're using.
2. **Remove Rows with Null Values**: If you choose to remove rows containing null values, simply delete those rows from your dataset.
3. **Remove Columns with Null Values**: If certain columns have a large number of null values or are not critical for your analysis, you can remove those columns entirely from your dataset.
4. **Validate:** After removing or imputing null values, validate your dataset to ensure that the handling process was successful and that the resulting data is suitable for your analysis or modeling tasks.

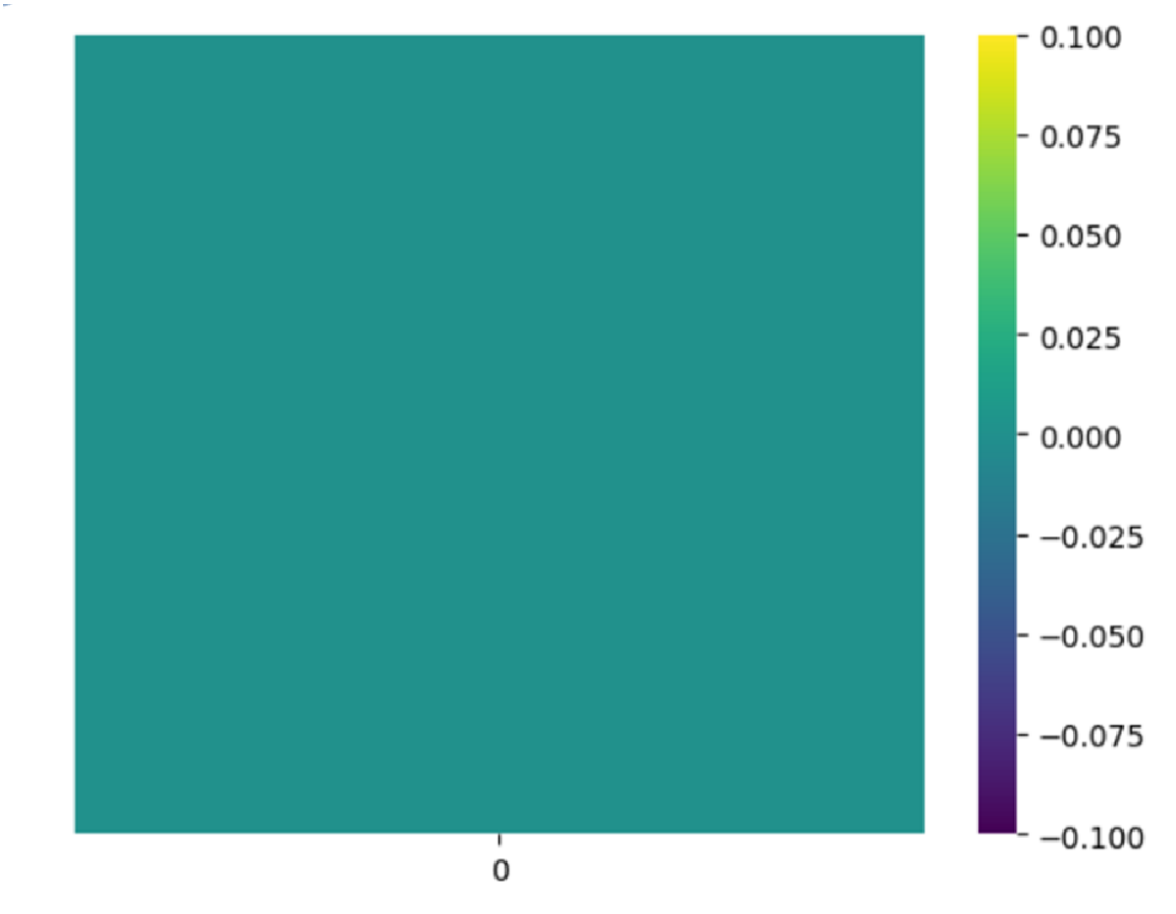
: ***Heatmap implying absence of null values***



To remove the duplicate values, we had the following Rules:

1. **Identify Duplicate Rows:** Begin by identifying duplicate rows in your dataset. Duplicate rows are rows that have identical values across all columns.
2. **Choose Columns for Comparison:** Decide which columns you want to use to identify duplicates.
3. **Remove Duplicate Rows:** Once you've identified duplicate rows, remove them from your dataset
4. **Validate:** After removing duplicate rows, validate your dataset to ensure that the process was successful and that the resulting data is clean and free of duplicates.

Heatmap implying absence of Duplicate values



Outliers

An outlier is a data point that significantly deviates from the rest of the data. It can be either much higher or much lower than the other data points, and its presence can have a significant impact on the results of machine learning algorithms.

Types of Outliers:-

There are **three** main types of outliers:

- **Global outliers:** Global outliers are isolated data points that are far away from the main body of the data. They are often easy to identify and remove.

- **Contextual outliers**: Contextual outliers are data points that are unusual in a specific context but may not be outliers in a different context. They are often more difficult to identify and may require additional information or domain knowledge to determine their significance.

- **Collective outliers:** In a given set of data, when a group of data points deviates from the rest of the data set is called collective outliers. Here, the particular set of data objects may not be outliers, but when you consider the data objects as a whole, they may behave as outliers.

Outlier Detection Methods:-

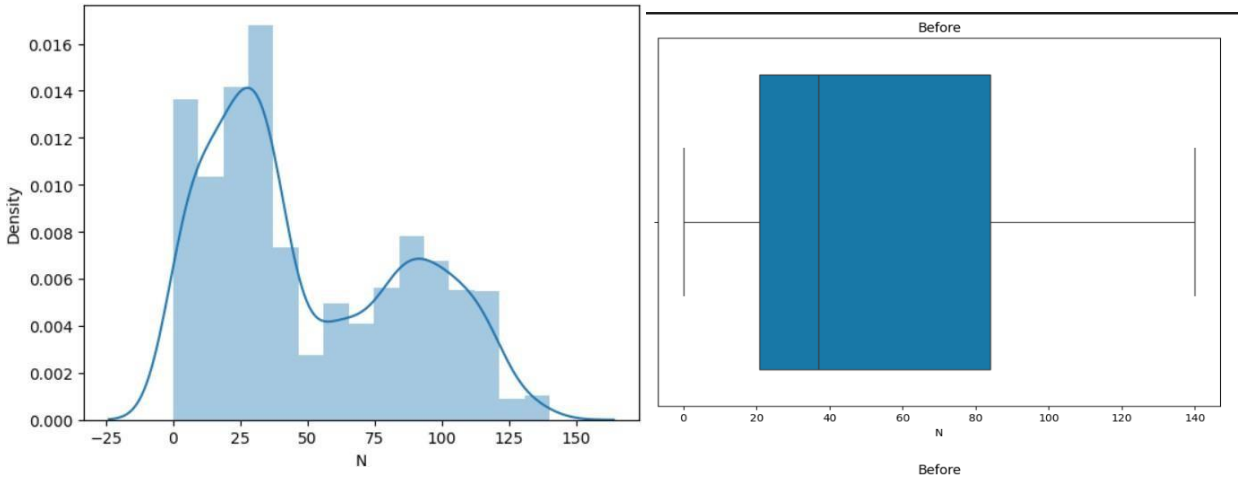
Statistical Methods:

- **Z-Score**: This method calculates the standard deviation of the data points and identifies outliers as those with Z-scores exceeding a certain threshold (typically 3 or -3).

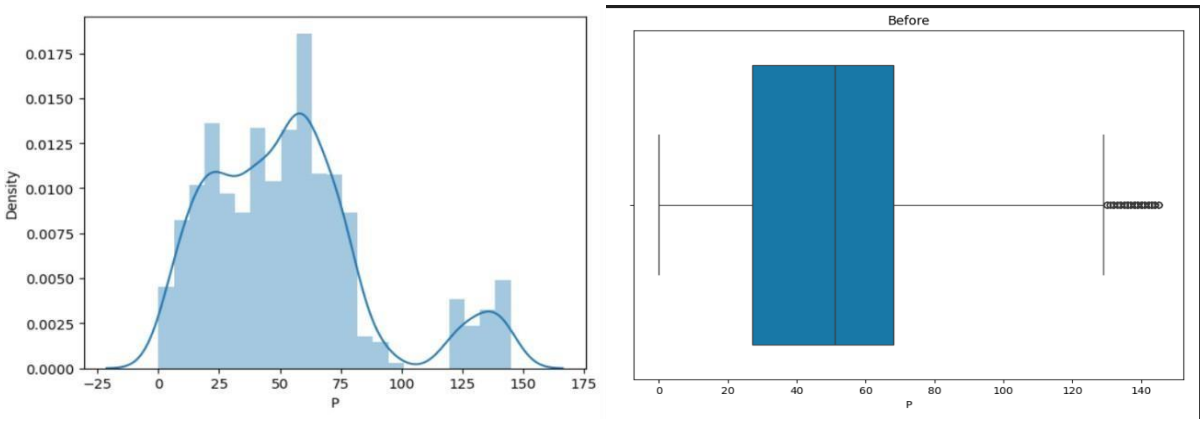
- **Interquartile Range (IQR):** IQR identifies outliers as data points falling outside the range defined by Q1-k*(Q3-Q1) and Q3+k*(Q3-Q1), where Q1 and Q3 are the first and third quartiles, and k is a factor (typically 1.5).

We plot distribution graph and boxplot to visualise the outliers. The plots are given below:

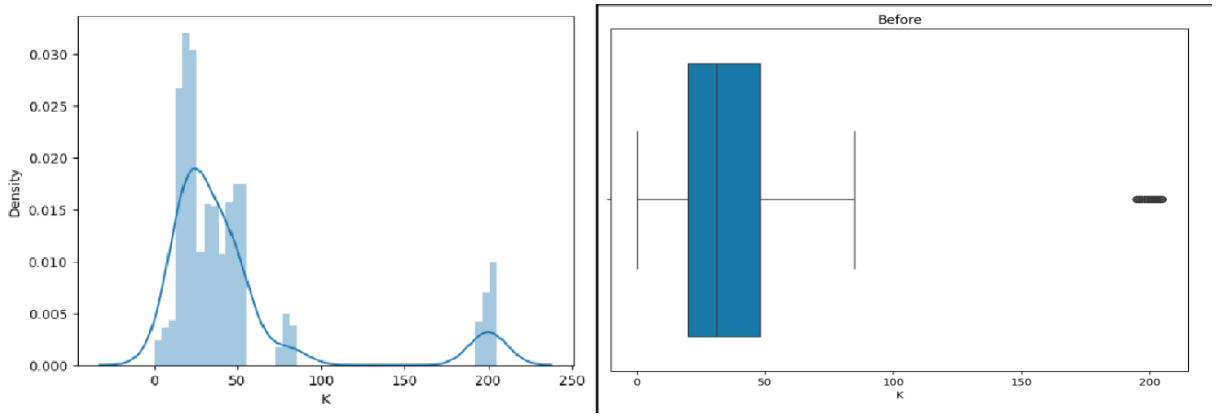
Distribution plot and boxplot of Nitrogen possible outliers.



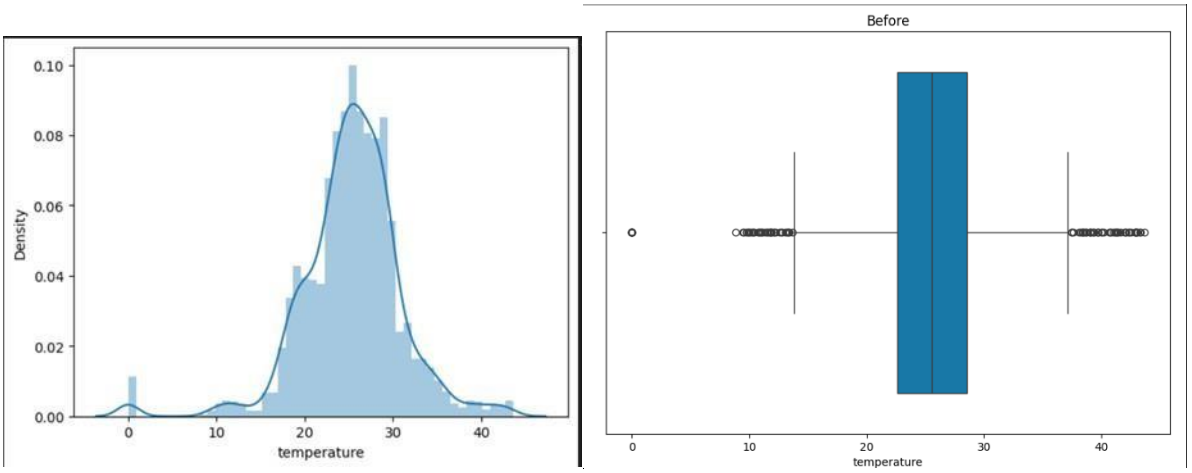
Distribution plot and boxplot of Phosphorus possible outliers.



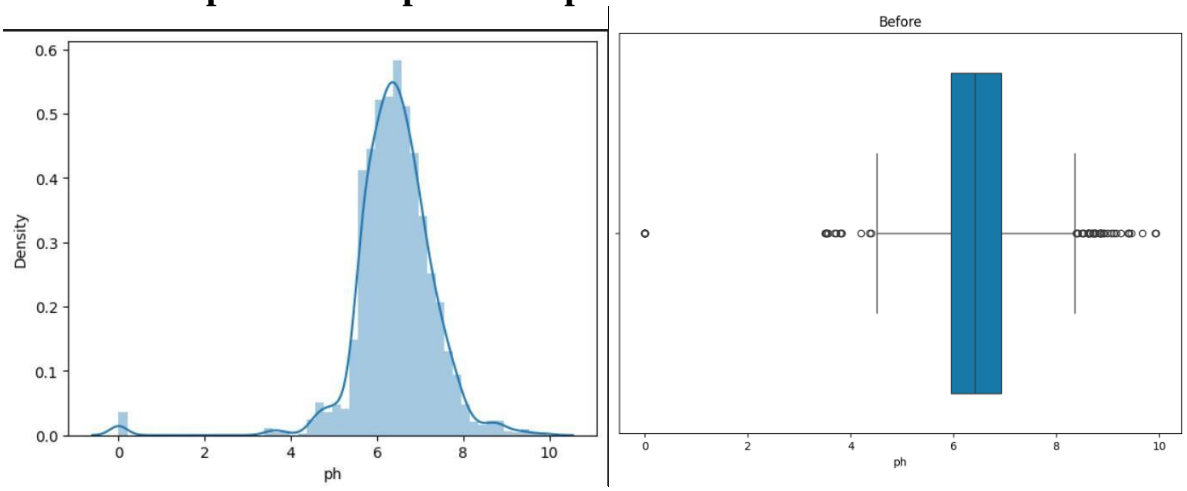
Distribution plot and boxplot of Potassium possible outliers.



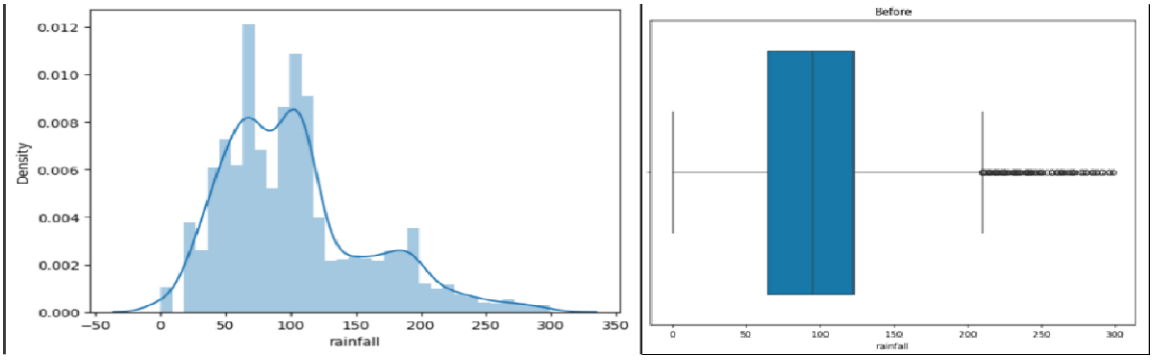
Distribution plot and boxplot of Temperature possible outliers.



Distribution plot and boxplot of Ph possible outliers.

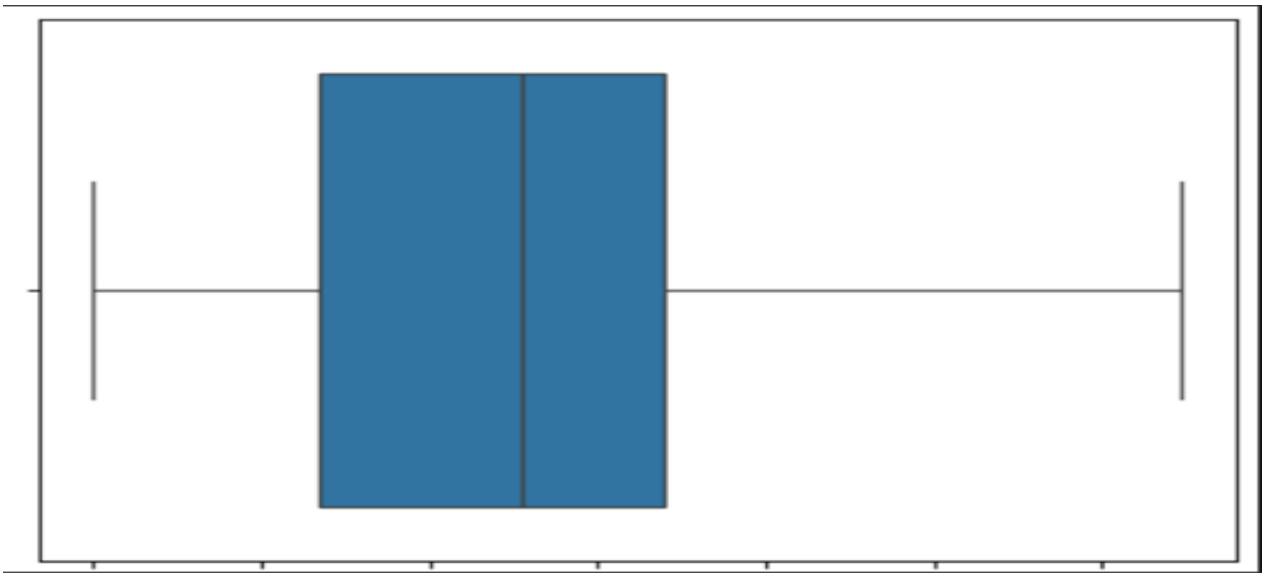


Distribution plot and boxplot of Rainfall possible outliers.

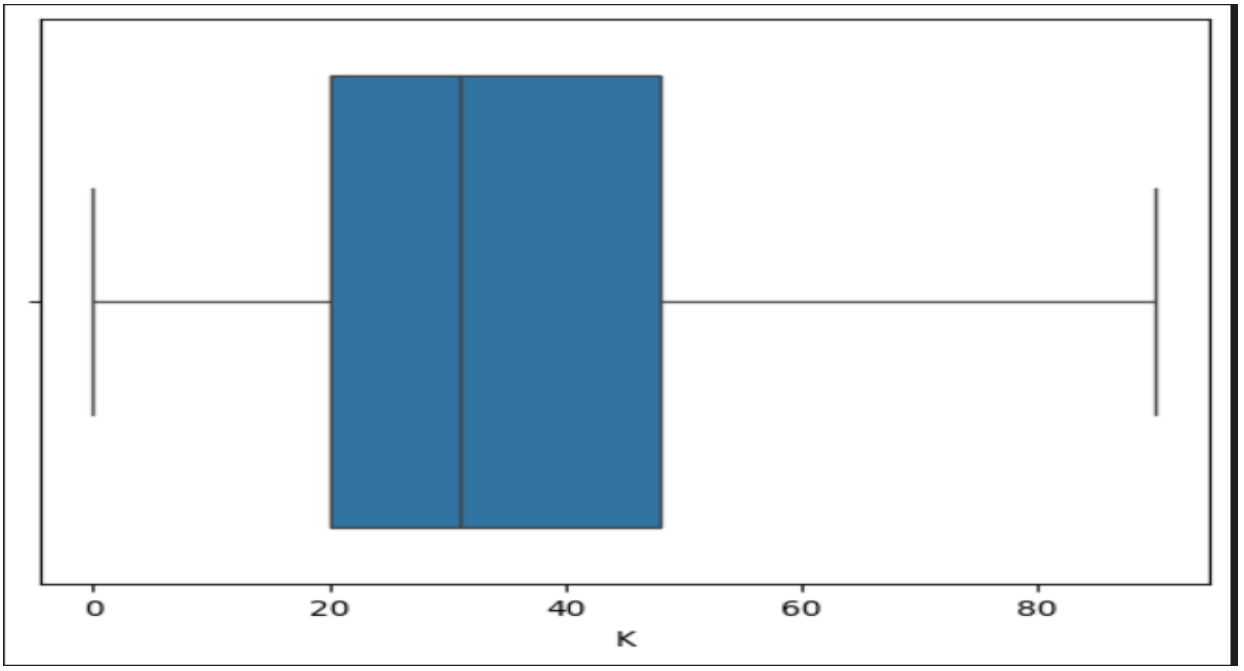


From the distribution plots and boxplots shown below, Now we have cleaned the outliers.

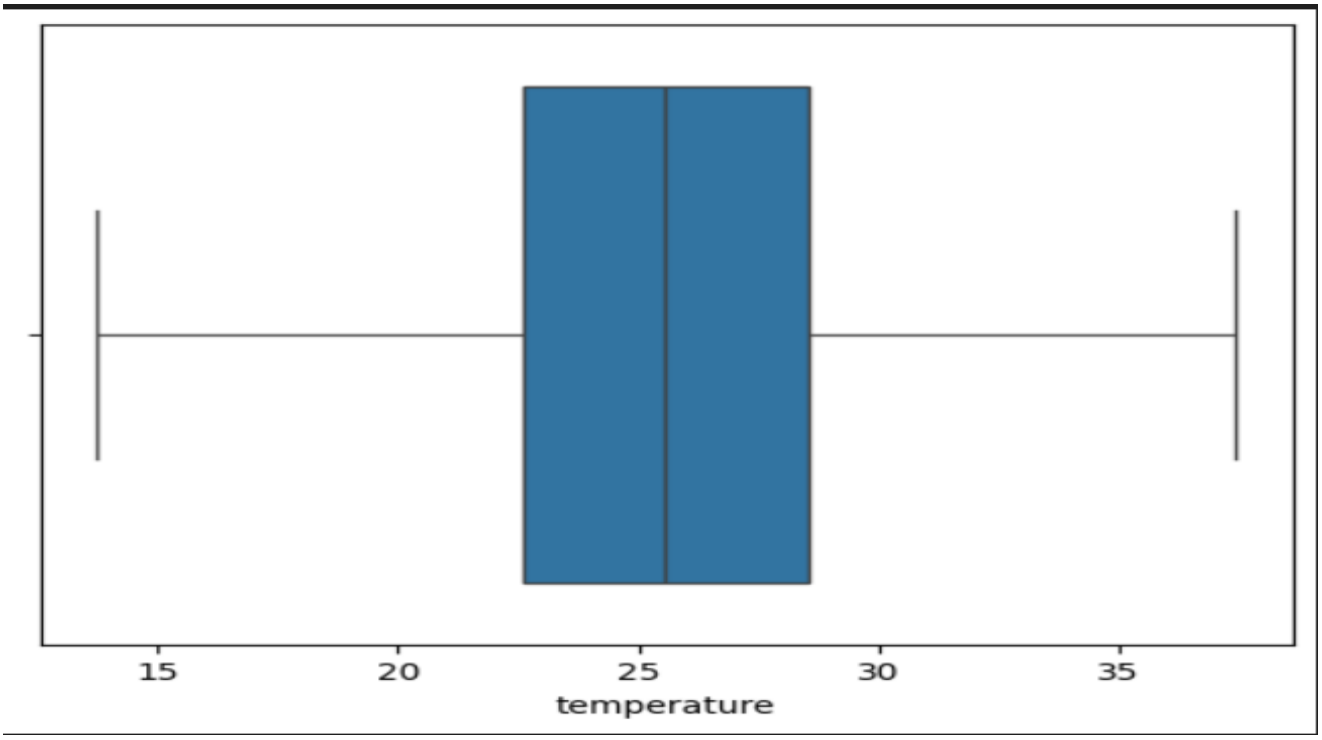
Now we have cleaned the outliers of Phosphorus



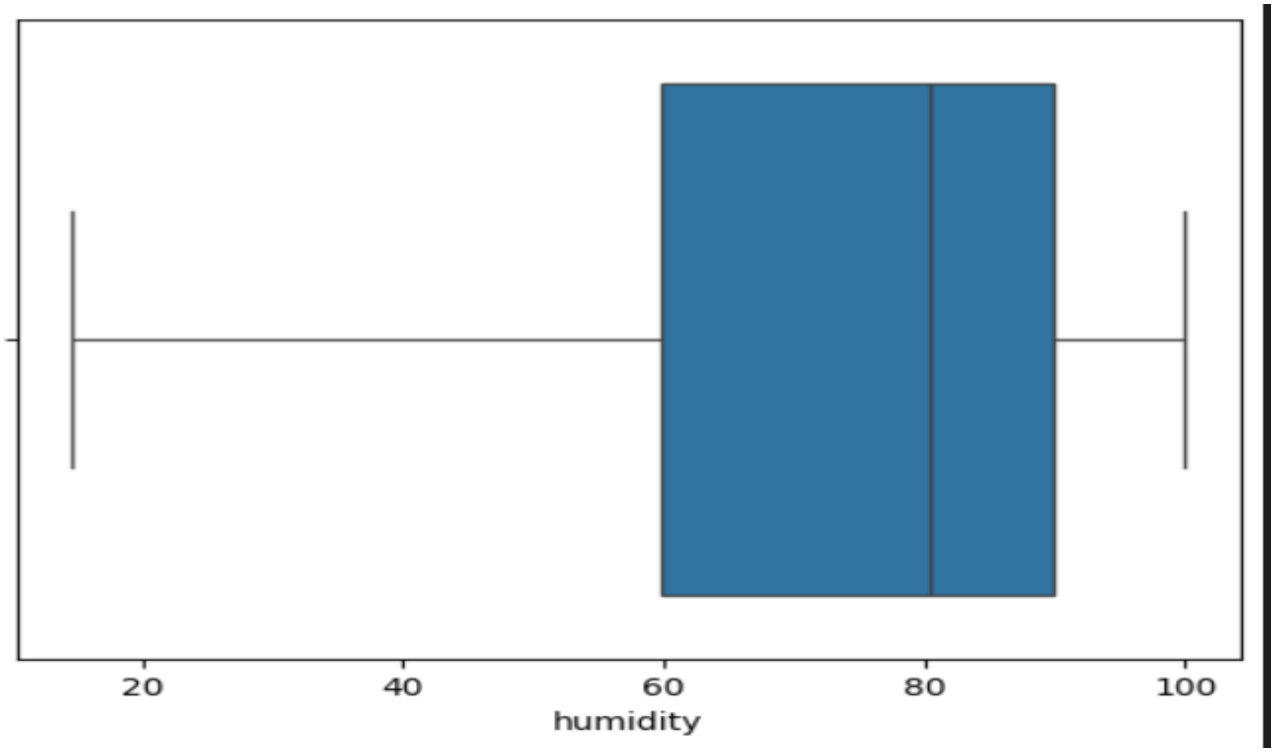
Now we have cleaned the outliers of Potassium



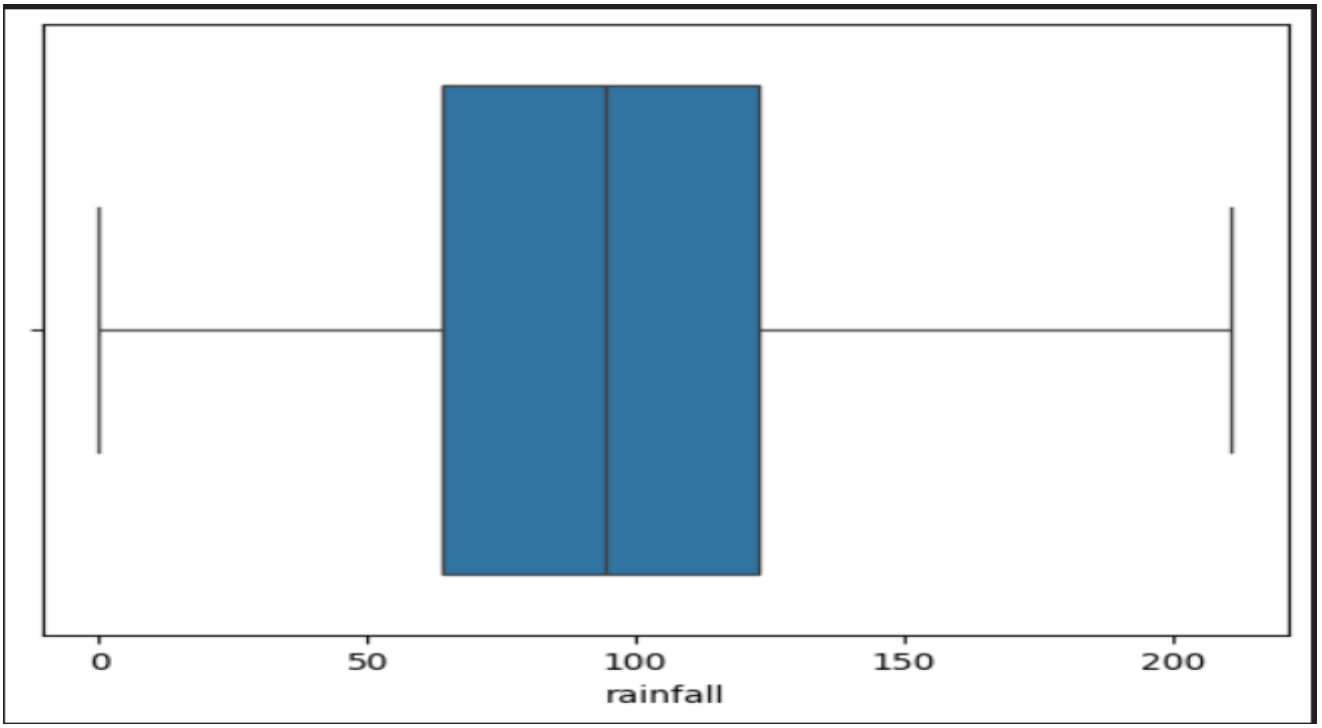
Now we have cleaned the outliers of Temperature



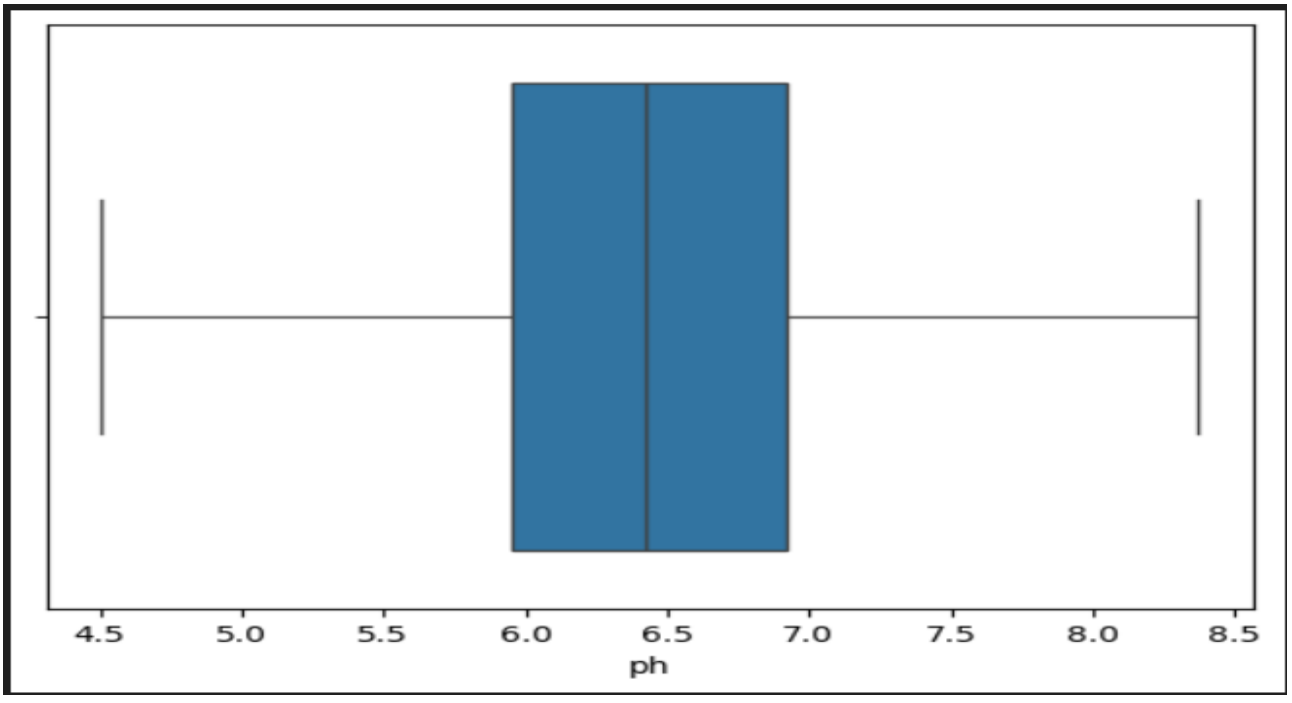
Now we have cleaned the outliers of Humidity



Now we have cleaned the outliers of Rainfall



Now we have cleaned the outliers of Ph



Model Building

- Splitting data for training and testing purpose

  We split the given train dataset into two parts for training and testing purpose. The split ratio we used is 0.75 which indicates we used 75% data for training purpose and 25% data for testing purpose. We will be using the same split ratio for all the models trained.

- We use the four models of the training and testing purpose. So the models are
  1. Random Forest
  2. Decision Tree
  3. Support Vector Machine (SVM)
  4. Logistic Regression

Random Forest

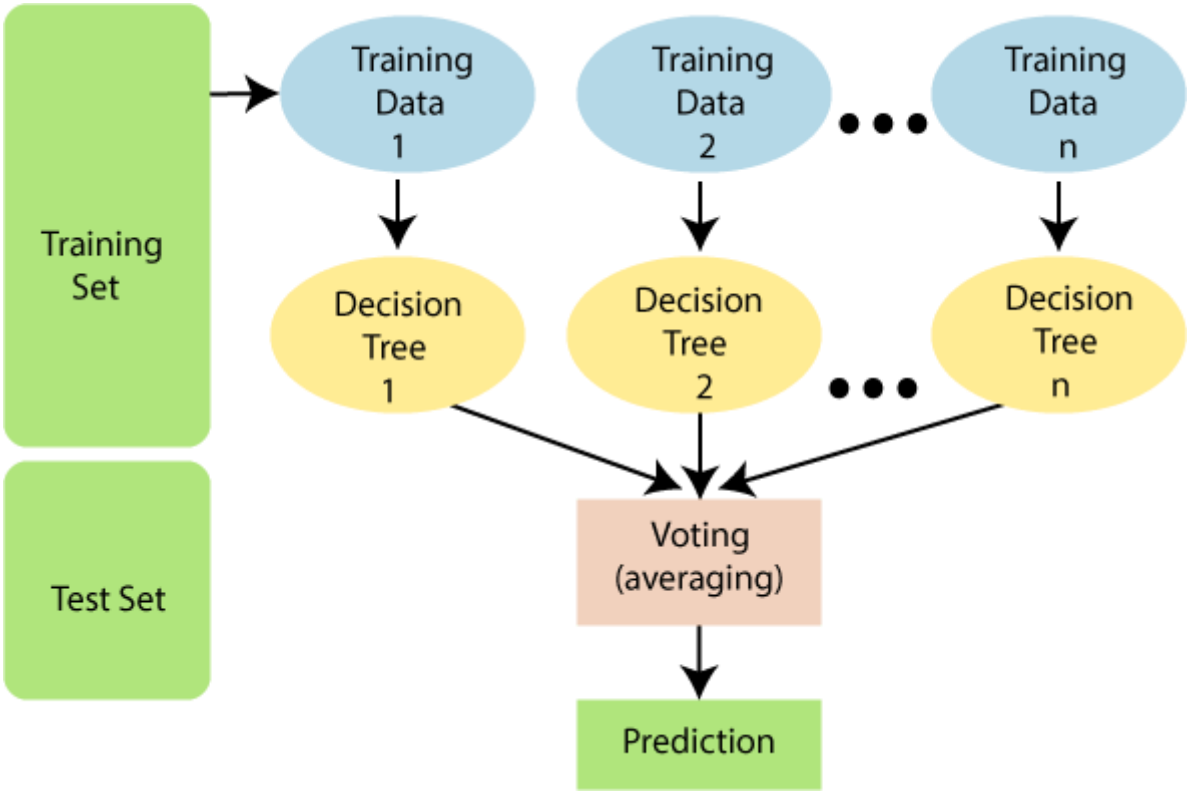
Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."

Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

- The below diagram explains the working of the Random Forest algorithm:



- **Assumptions for Random Forest**

  - There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
  - The predictions from each tree must have very low correlations.

- **Why use Random Forest?**

  - It takes less training time as compared to other algorithms.
  - It predicts output with high accuracy, even for the large dataset it runs efficiently.
  - It can also maintain accuracy when a large proportion of data is missing.

- **How does Random Forest algorithm work?**

  - Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

- The Working process can be explained in the below steps and diagram:

  Step-1: Select random K data points from the training set.

  Step-2: Build the decision trees associated with the selected data points (Subsets).

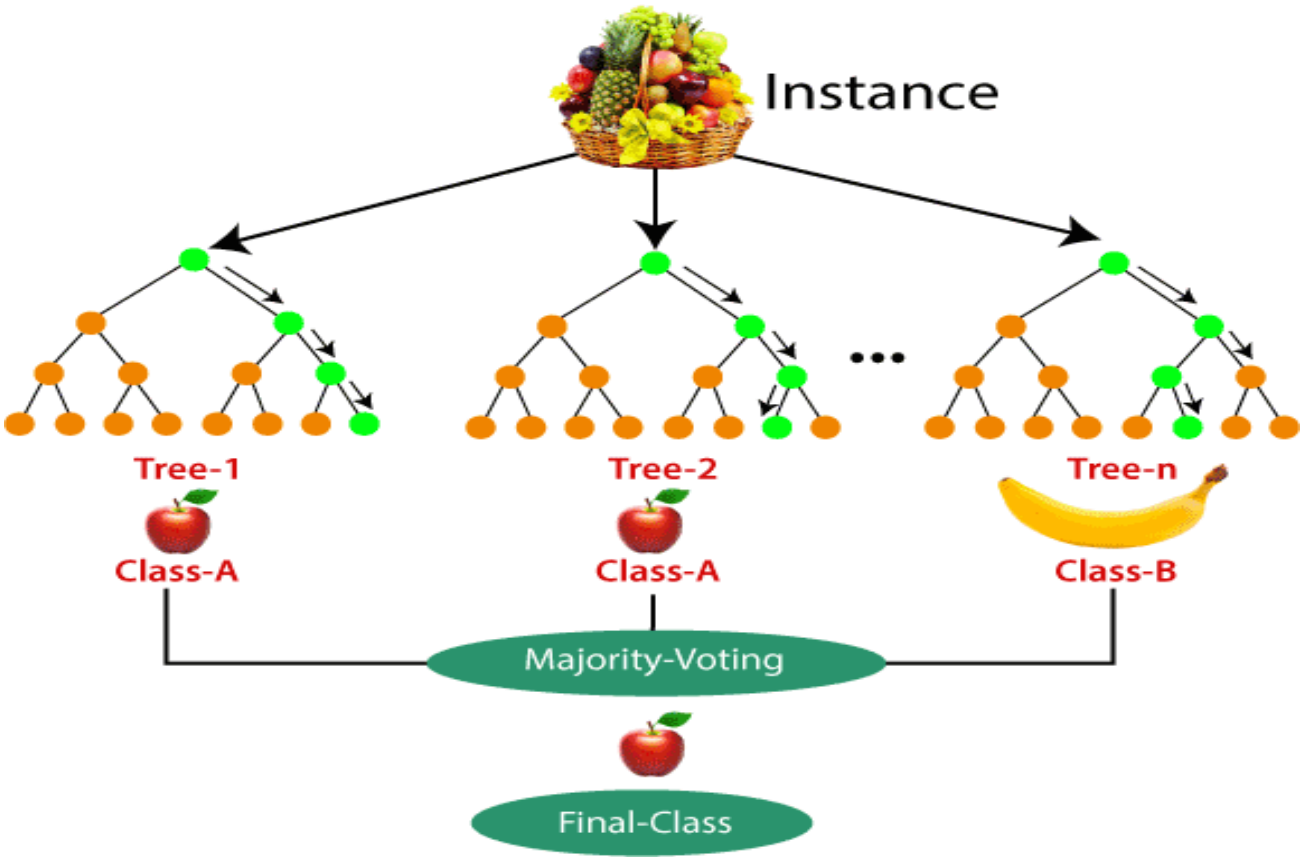
  Step-3: Choose the number N for decision trees that you want to build.

  Step-4: Repeat Step 1 & 2.

  Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

- The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



- **Applications of Random Forest**

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

- **Advantages of Random Forest**

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

- **Disadvantages of Random Forest**

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

- **Random Forest Test Accuracy**

Random Forest Testing Accuracy	**0.97**

- **Classification Report for Testing Data:**

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	29
banana	0.96	1.00	0.98	25
blackgram	1.00	1.00	1.00	19
chickpea	1.00	1.00	1.00	24
coconut	0.96	1.00	0.98	23
coffee	1.00	0.95	0.98	22

cotton	0.97	1.00	0.98	30
	precision	**recall**	**f1-score**	**support**
grapes	1.00	1.00	1.00	31
jute	0.87	1.00	0.93	26
kidneybeans	1.00	0.97	0.98	29
lentil	0.94	0.85	0.89	20
maize	1.00	1.00	1.00	20
mango	1.00	1.00	1.00	28
mothbeans	0.91	1.00	0.95	21
mungbean	0.97	1.00	0.98	32
muskmelon	1.00	0.86	0.93	22
orange	1.00	1.00	1.00	23
papaya	1.00	0.96	0.98	23
pigeonpeas	0.97	0.97	0.91	30
pomegranate	0.96	1.00	0.98	24
rice	1.00	0.83	0.91	24
watermelon	0.96	1.00	0.98	25

- **Macro average and weighted average**

Macro average	**0.98**	**0.97**	**0.97**	**550**
Weighted average	**0.98**	**0.97**	**0.97**	**550**

- **Precision**
    - Precision represents the proportion of true positive predictions among all positive predictions.

  Mathematical formula,

  Precision = TP/ (TP + FP)

- **Recall**
    - Represents the proportion of true positive predictions among all actual positive instances.

Mathematical formula,

Recall = (TP)/ (TP + FN)

- **F1-score**
    - F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics.

Mathematical formula,

F1-score = 2 * (precision * recall)/ (precision + recall)

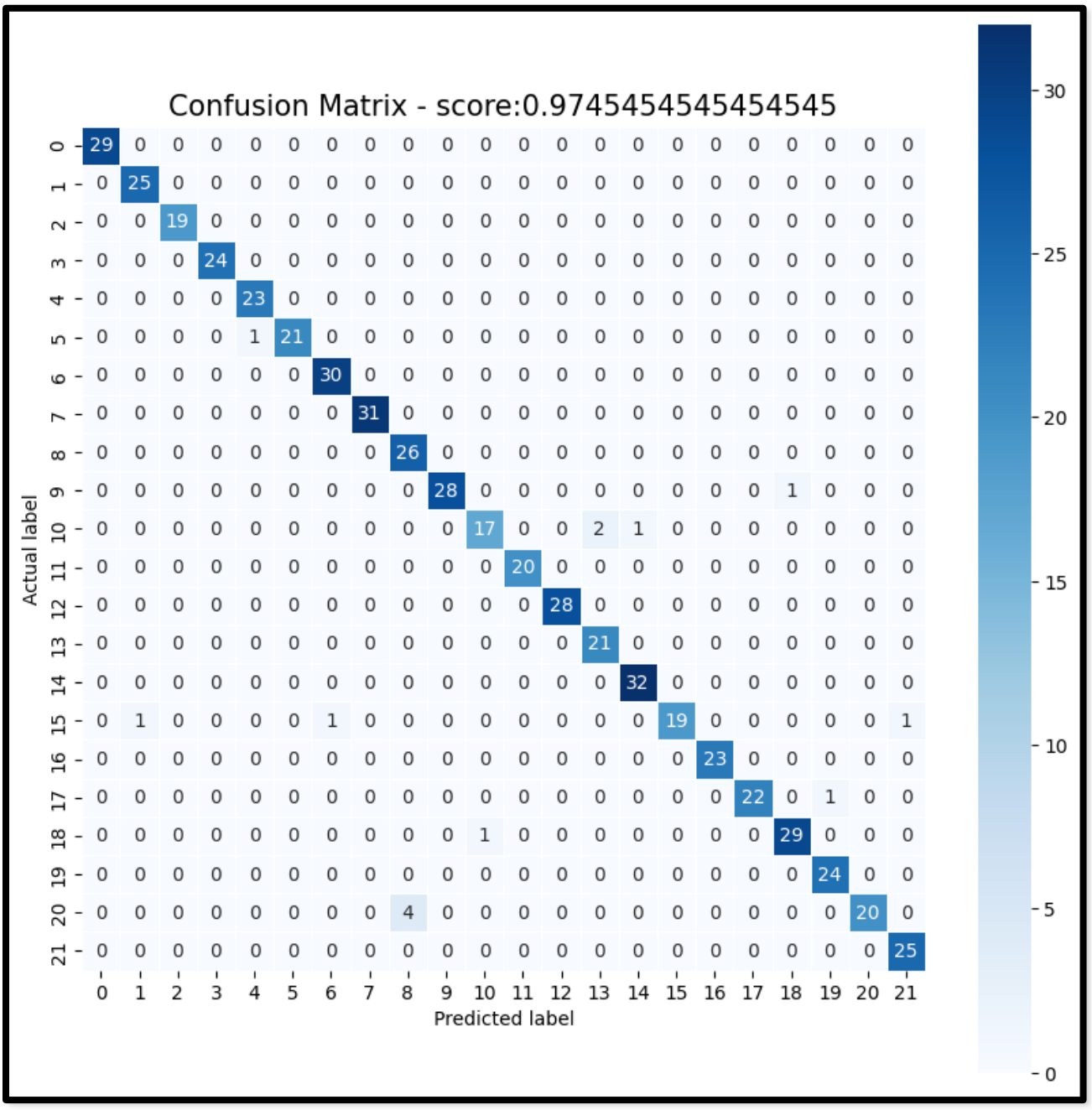
- **Support**
    - Support indicates the number of actual occurrences of each class in the testing dataset.

- **Confusion Matrix**
    - It is a much better way to evaluate the classifier

n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

- **True Negative:** Model has given prediction No, and the real or actual value was also No.
- **True Positive:** The model has predicted yes, and the actual value was also true.
- **False Negative:** The model has predicted no, but the actual value was yes, it is also called as Type-II error.

- **False Positive:** The model has predicted Yes, but the actual value was No. It is also called a Type-I error.
- **<u>Confusion Matrix of our model Random Forest</u>**



Decision Tree

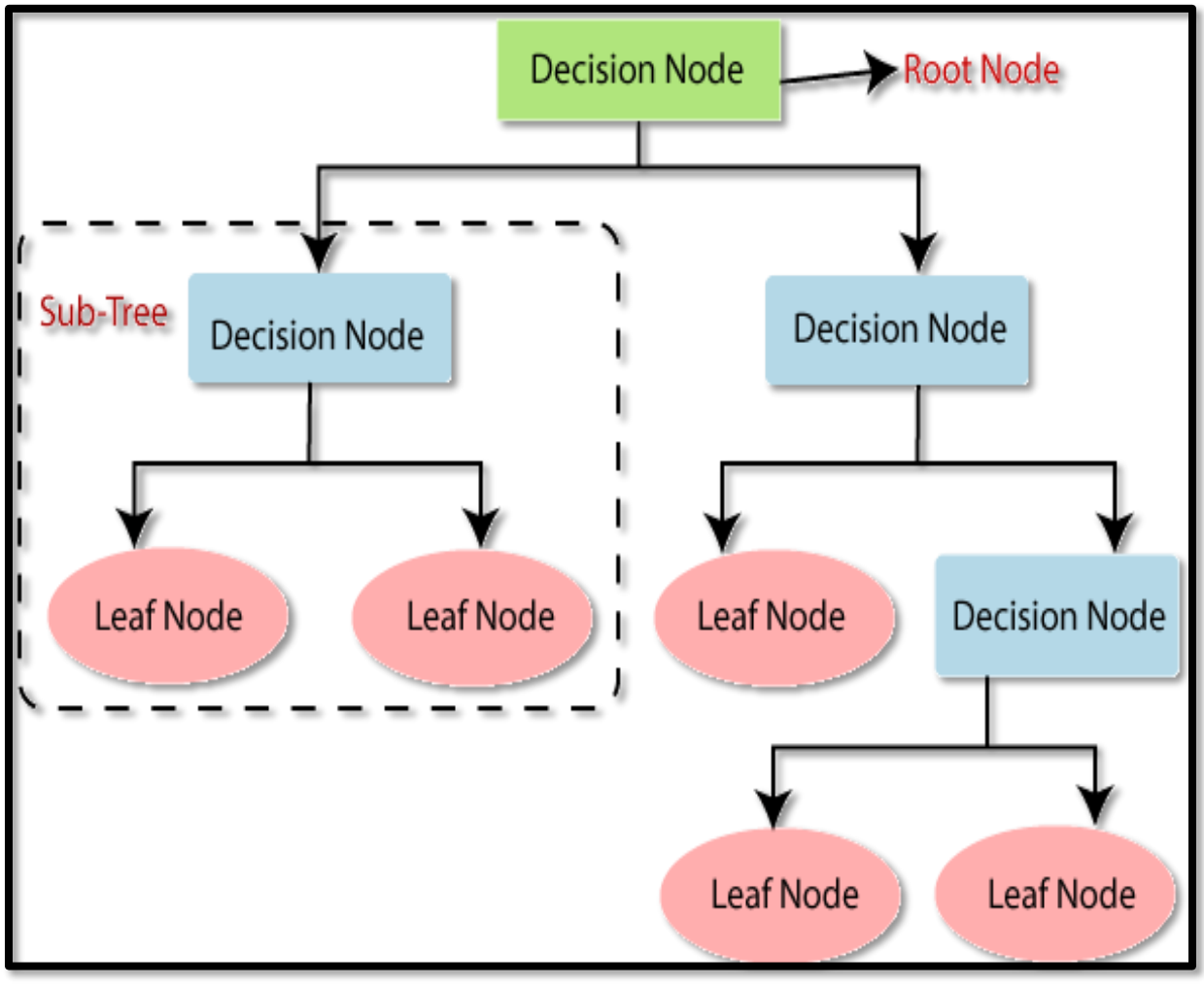
Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

- Below diagram explains the general structure of a decision tree:



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

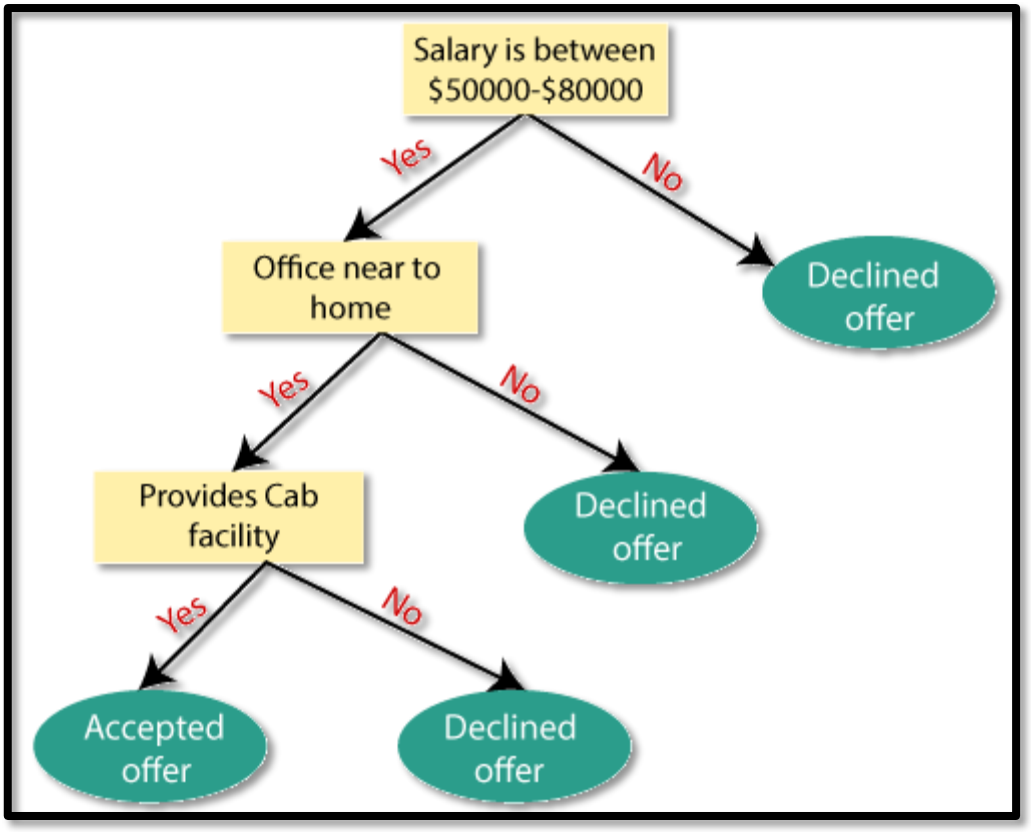
Decision Tree Terminologies

- **Root Node**: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node**: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree**: A tree formed by splitting the tree.
- **Pruning**: Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node**: The root node of the tree is called the parent node, and other nodes are called the child nodes.

Decision Tree algorithm

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using Attribute Selection Measure (ASM).
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in the step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- Information Gain
- Gini Index

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.

- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

1. Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

- S= Total number of samples
- P(yes)= probability of yes
- P(no)= probability of no

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

Gini Index= 1- $\sum_j P_j^2$

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
- For more class labels, the computational complexity of the decision tree may increase.

Decision Tree Test Accuracy

Decision Tree Testing Accuracy	**0.90**

Classification Report for Testing Data:

	precision	recall	f1-score	support
apple	1.00	0.97	0.98	29
banana	0.96	0.96	0.96	25
blackgram	0.68	0.79	0.73	19
chickpea	1.00	1.00	1.00	24
coconut	0.92	1.00	0.96	23
coffee	1.00	0.95	0.98	22
cotton	0.91	1.00	0.95	30
grapes	1.00	1.00	1.00	31
jute	0.88	0.27	0.41	26
kidneybeans	0.96	0.79	0.87	29
lentil	0.88	0.75	0.81	20
maize	0.86	0.95	0.90	20
mango	1.00	0.86	0.92	28
mothbeans	0.77	0.95	0.85	21
mungbean	0.97	1.00	0.98	32
muskmelon	1.00	0.86	0.93	22
orange	1.00	1.00	1.00	23
papaya	1.00	0.91	0.95	23
pigeonpeas	0.88	0.93	0.90	30
pomegranate	0.96	1.00	0.98	24
rice	0.54	0.92	0.68	24
watermelon	0.96	1.00	0.98	25

- **Macro average and weighted average**

Macro average	**0.91**	**0.90**	**0.90**	**550**
Weighted average	**0.92**	**0.91**	**0.90**	**550**

- **Precision**

    - Precision represents the proportion of true positive predictions among all positive predictions.

  Mathematical formula,

  Precision = TP/ (TP + FP)

- **Recall**

    - Represents the proportion of true positive predictions among all actual positive instances.

  Mathematical formula,

  Recall = (TP)/ (TP + FN)

- **F1-score**

    - F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics.

  Mathematical formula,

  F1-score = 2 * (precision * recall)/ (precision + recall)

- **Support**

    - Support indicates the number of actual occurrences of each class in the testing dataset.

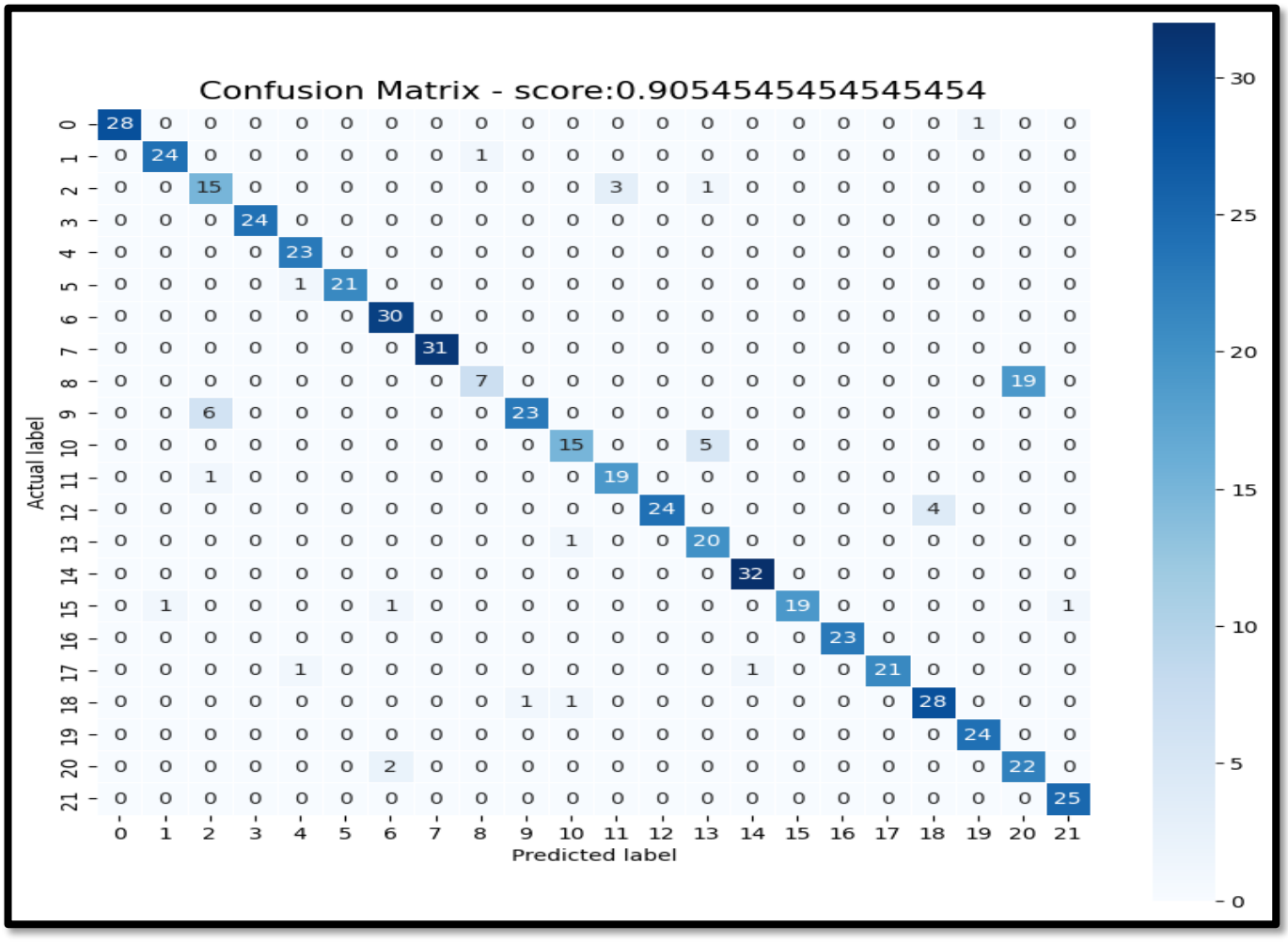
- **Confusion Matrix**

  - It is a much better way to evaluate the classifier

n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

- **True Negative:** Model has given prediction No, and the real or actual value was also No.
- **True Positive:** The model has predicted yes, and the actual value was also true.
- **False Negative:** The model has predicted no, but the actual value was yes, it is also called as Type-II error.
- **False Positive:** The model has predicted Yes, but the actual value was No. It is also called a Type-I error.

Confusion Matrix of our model Decision Tree:



Support Vector Machine(SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

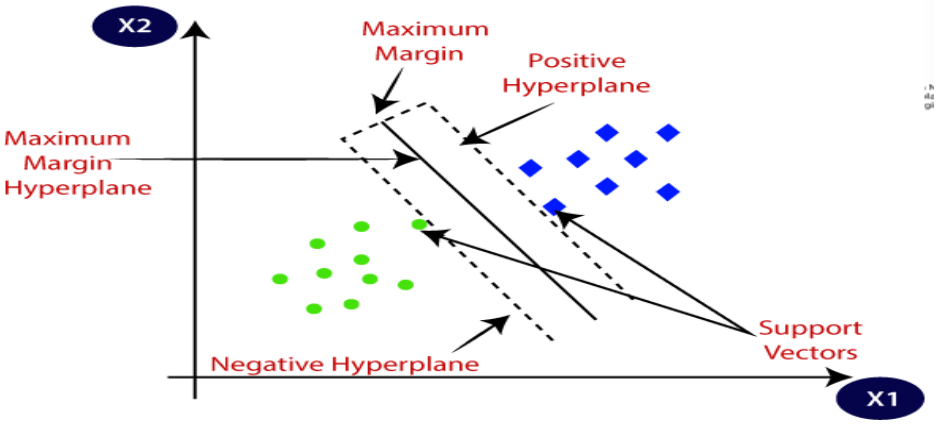
Types of Support Vector Machine:

Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:

Linear SVM: Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable. This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane that maximizes the margin between the classes is the decision boundary.

Non-Linear SVM: Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space, where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.

- *Below diagram explains the general structure of a Support Vector Machine.*



Why Use Support Vector Machine ?

SVM is a supervised machine learning algorithm which can be used for classification or regression problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs. Simply put, it does some extremely complex data transformations, then figures out how to separate your data based on the labels or outputs you've defined.
Support Vector Machines (SVMs) are utilized in machine learning for several compelling reasons:

- ***Effective for High-dimensional Data:*** SVMs perform well in scenarios where the number of features (dimensions) is much higher than the number of samples. This makes them suitable for tasks like text classification, image recognition, and gene expression analysis.

- ***Memory Efficiency:*** SVMs only use a subset of training data points (support vectors) in decision-making, which makes them memory efficient, particularly for large datasets. This also contributes to faster prediction times compared to some other algorithms.

Support Vector Machine Terminology

- **Hyperplane:** Hyperplane is the decision boundary that is used to separate the data points of different classes in a feature space. In the case of linear classifications, it will be a linear equation i.e. $wx+b = 0$.

- **Support Vectors**: Support vectors are the closest data points to the hyperplane, which makes a critical role in deciding the hyperplane and margin.

- **Margin:** Margin is the distance between the support vector and hyperplane. The main objective of the support vector machine algorithm is to maximize the margin.

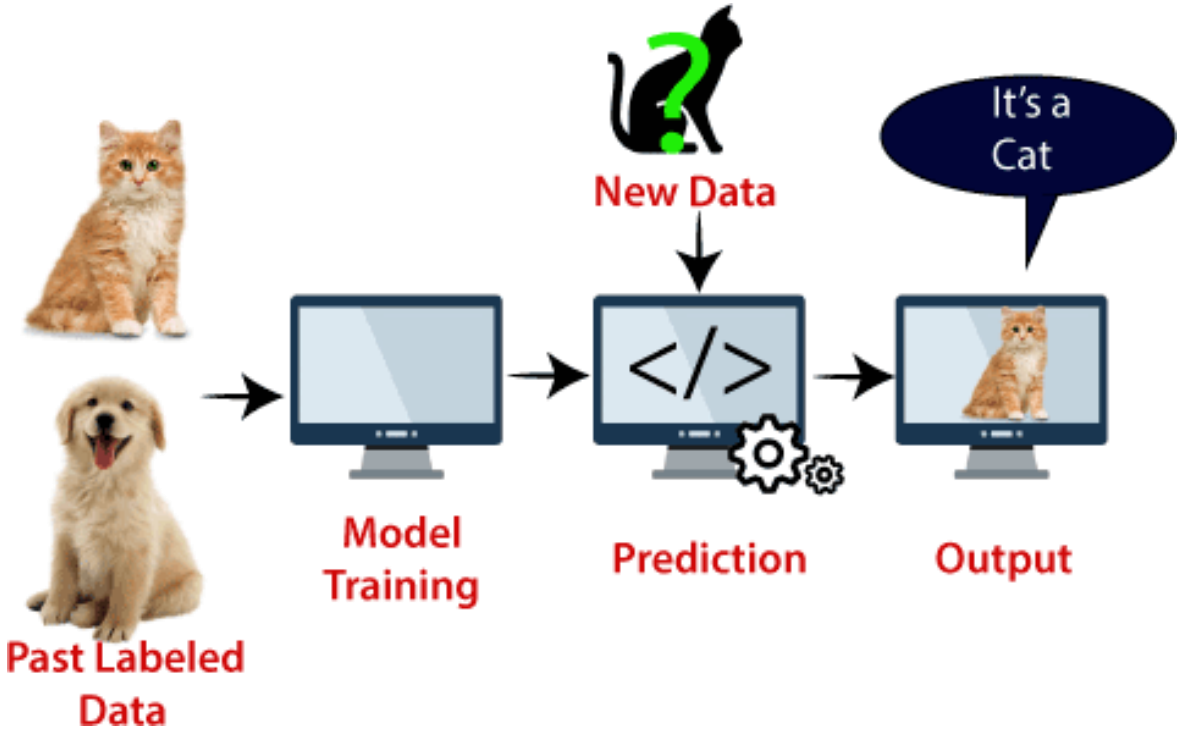
- **Kernel:** Kernel is the mathematical function, which is used in SVM to map the original input data points into high-dimensional feature spaces, so, that the hyperplane can be easily found out even if the data points are not linearly separable in the original input space.

- **Hard Margin:** The maximum-margin hyperplane or the hard margin hyperplane is a hyperplane that properly separates the data points of different categories without any misclassifications.

- **Soft Margin:** When the data is not perfectly separable or contains outliers, SVM permits a soft margin technique.

Example Of Support Vector Machine

SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



Advantages of Support Vector Machine:

- SVM works relatively well when there is a clear margin of separation between classes.
- SVM is more effective in high dimensional spaces.
- SVM is effective in cases where the number of dimensions is greater than the number of samples.
- SVM is relatively memory efficient

Disadvantages of Support Vector Machine:

- SVM algorithm is not suitable for large data sets.
- SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
- In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
- As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.

Support Vector Machine Testing Accuracy

Support Vector Machine Testing Accuracy	0.93

Classification Report for Testing Data:

	precision	recall	f1-score	support
apple	1.00	0.90	0.95	29
banana	0.92	0.92	0.92	25
blackgram	1.00	1.00	1.00	19
chickpea	0.96	1.00	0.98	24
coconut	0.82	1.00	0.90	23
coffee	0.91	0.95	0.93	22
cotton	0.96	0.90	0.93	30
grapes	1.00	0.97	0.98	31
jute	0.86	0.96	0.91	26
kidneybeans	0.97	1.00	0.98	29
lentil	0.86	0.90	0.88	20
maize	0.90	0.95	0.93	20
mango	1.00	0.93	0.96	28
mothbeans	0.95	0.90	0.93	21
mungbean	1.00	1.00	1.00	32
muskmelon	0.90	0.82	0.86	22
orange	1.00	1.00	1.00	23
papaya	0.71	0.96	0.81	23
pigeonpeas	1.00	0.87	0.93	30
pomegranate	1.00	0.88	0.93	24
rice	0.95	0.75	0.84	24
watermelon	0.93	1.00	0.96	25

Macro average and weighted average

Macro average	0.94	0.93	0.93	550
Weighted average	0.94	0.93	0.94	550

- **Precision**
    - Precision represents the proportion of true positive predictions among all positive predictions.

  Mathematical formula,

  Precision = TP/ (TP + FP)

- **Recall**
    - Represents the proportion of true positive predictions among all actual positive instances.

  Mathematical formula,

  Recall = (TP)/ (TP + FN)

- **F1-score**
    - F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics.

  Mathematical formula,

  F1-score = 2 * (precision * recall)/ (precision + recall)

- **Support**
    - Support indicates the number of actual occurrences of each class in the testing dataset.

- **Confusion Matrix**

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing**.** Some features of Confusion matrix are given below:

- For the 2 prediction classes of classifiers, the matrix is of 2*2 table, for 3 classes, it is 3*3 table, and so on.
- The matrix is divided into two dimensions, that are predicted values and actual values along with the total number of predictions.
- Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.

- It is a much better way to evaluate the classifier

n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

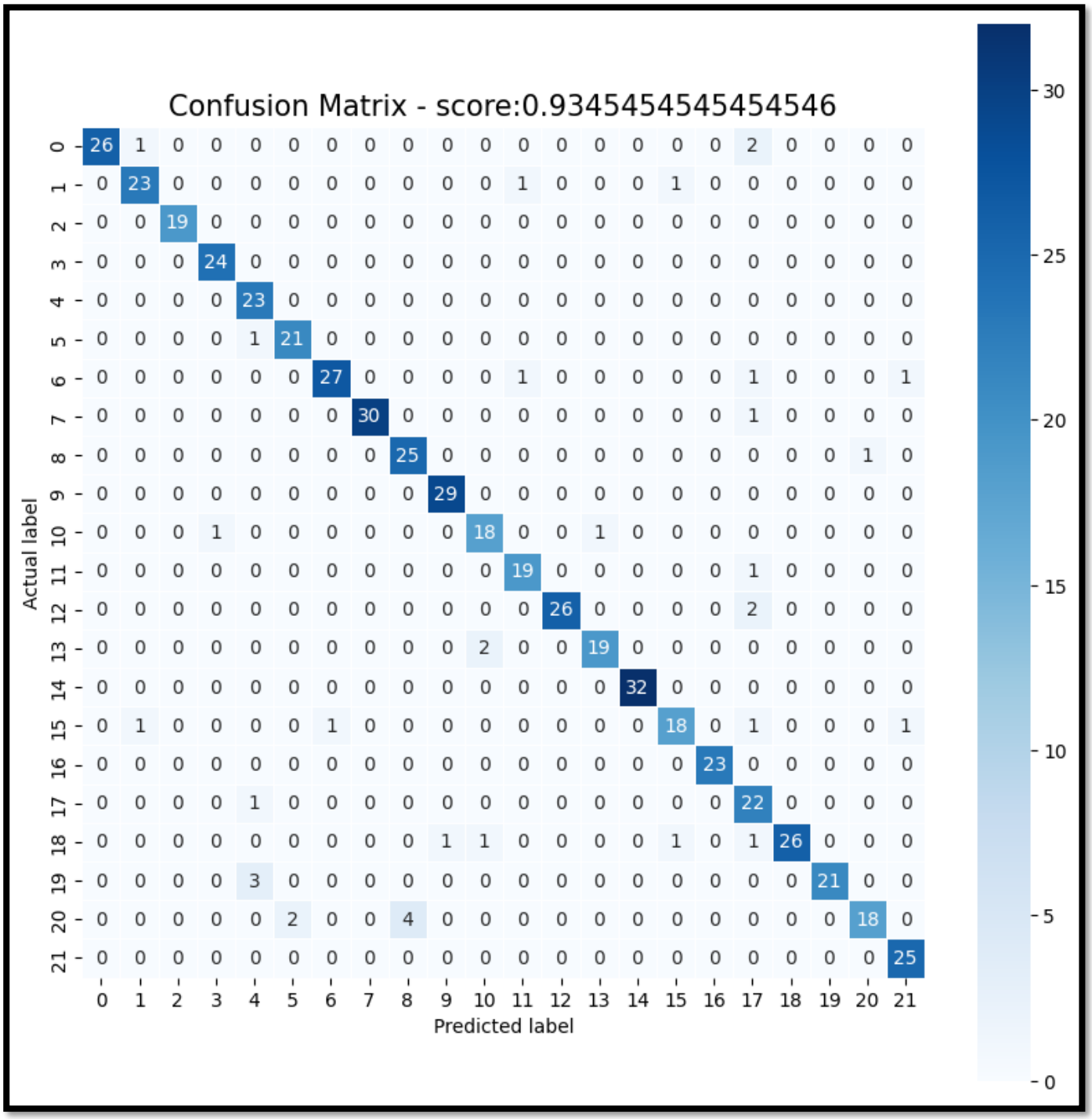
- **True Negative:** Model has given prediction No, and the real or actual value was also No.

- **True Positive:** The model has predicted yes, and the actual value was also true.

- **False Negative:** The model has predicted no, but the actual value was yes, it is also called as Type-II error.

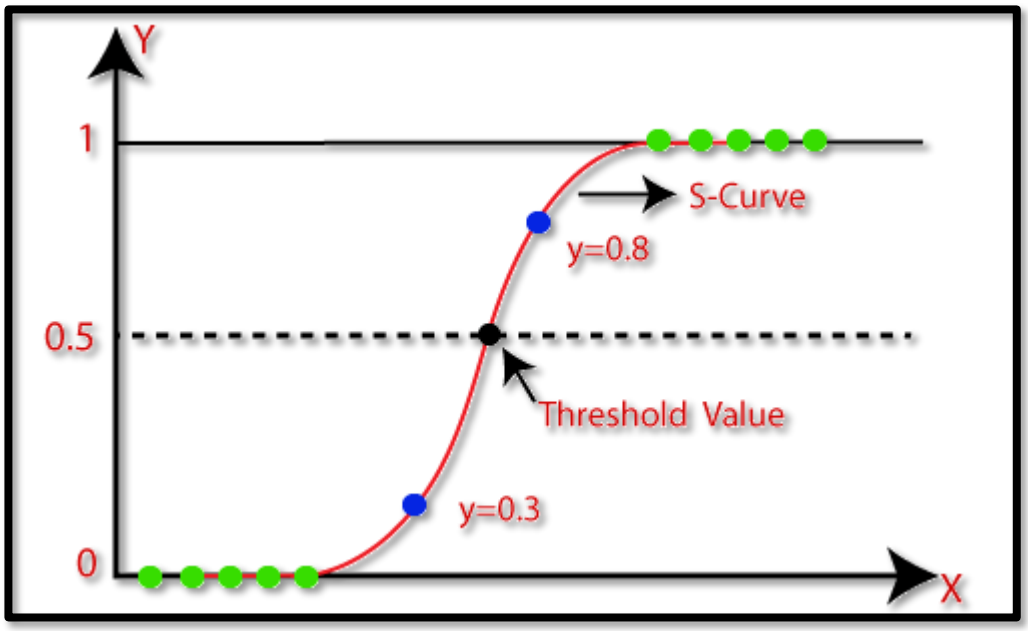
- **False Positive**: The model has predicted Yes, but the actual value was No. It is also called a Type-I error.

Confusion Matrix of our model Support Vector Machine:



Logistic Regression

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}$$; 0 for y= 0, and infinity for y=1

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

- Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High"

Steps in Logistic Regression: To implement the Logistic Regression using Python, we will use the same steps as we have done in previous topics of Regression. Below are the steps:

- Data Pre-processing step
- Fitting Logistic Regression to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

Advantages:

Interpretability: Logistic regression coefficients represent the relationship between the independent variables and the probability of a particular outcome. This makes it easy to interpret the impact of each predictor on the response variable.

Efficiency with Small Datasets: Logistic regression can perform well even with relatively small datasets. It doesn't require a large number of observations to achieve stable results, making it suitable for scenarios where data is limited.

Disadvantages:

Assumption of Linearity: Logistic regression assumes a linear relationship between the independent variables and the log-odds of the dependent variable. If this assumption is violated, the model's predictions may be inaccurate.

Limited to Linear Decision Boundaries: Logistic regression models are inherently linear classifiers, which means they can only capture linear decision boundaries between classes. In scenarios where the decision boundary is highly nonlinear, logistic regression may not perform well without feature engineering or transformation.

Logistic regression Test Accuracy

Logistic Regression Testing Accuracy	**0.92**

Classification Report for Testing Data

	precision	recall	f1-score	support
apple	1.00	0.90	0.95	29
banana	0.92	0.92	0.92	25
blackgram	0.83	1.00	0.90	19
chickpea	0.96	1.00	0.98	24
coconut	0.88	1.00	0.94	23
coffee	0.84	0.95	0.89	22
cotton	0.89	0.83	0.86	30
grapes	1.00	1.00	1.00	31
jute	0.89	0.92	0.91	26
kidneybeans	0.88	1.00	0.94	29
lentil	0.89	0.85	0.87	20
maize	0.87	1.00	0.93	20
mango	1.00	1.00	1.00	28
mothbeans	0.90	0.86	0.88	21
mungbean	1.00	1.00	1.00	32
muskmelon	0.78	0.82	0.80	22
orange	1.00	1.00	1.00	23
papaya	0.95	0.91	0.93	23
pigeonpeas	1.00	0.80	0.89	30
pomegranate	0.96	1.00	0.98	24
rice	0.95	0.83	0.89	24
watermelon	0.95	0.80	0.87	25

- **Macro average and weighted average**

Macro average	**0.93**	**0.93**	**0.92**	**550**
Weighted average	**0.93**	**0.93**	**0.93**	**550**

- **Precision**

  - Precision represents the proportion of true positive predictions among all positive predictions.

  Mathematical formula,

  Precision = TP/ (TP + FP)

- **Recall**

  - Represents the proportion of true positive predictions among all actual positive instances.

  Mathematical formula,

  Recall = (TP)/ (TP + FN)

- **F1-score**

  - F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics.

  Mathematical formula,

  F1-score = 2 * (precision * recall)/ (precision + recall)

- **Support**

  - Support indicates the number of actual occurrences of each class in the testing dataset.

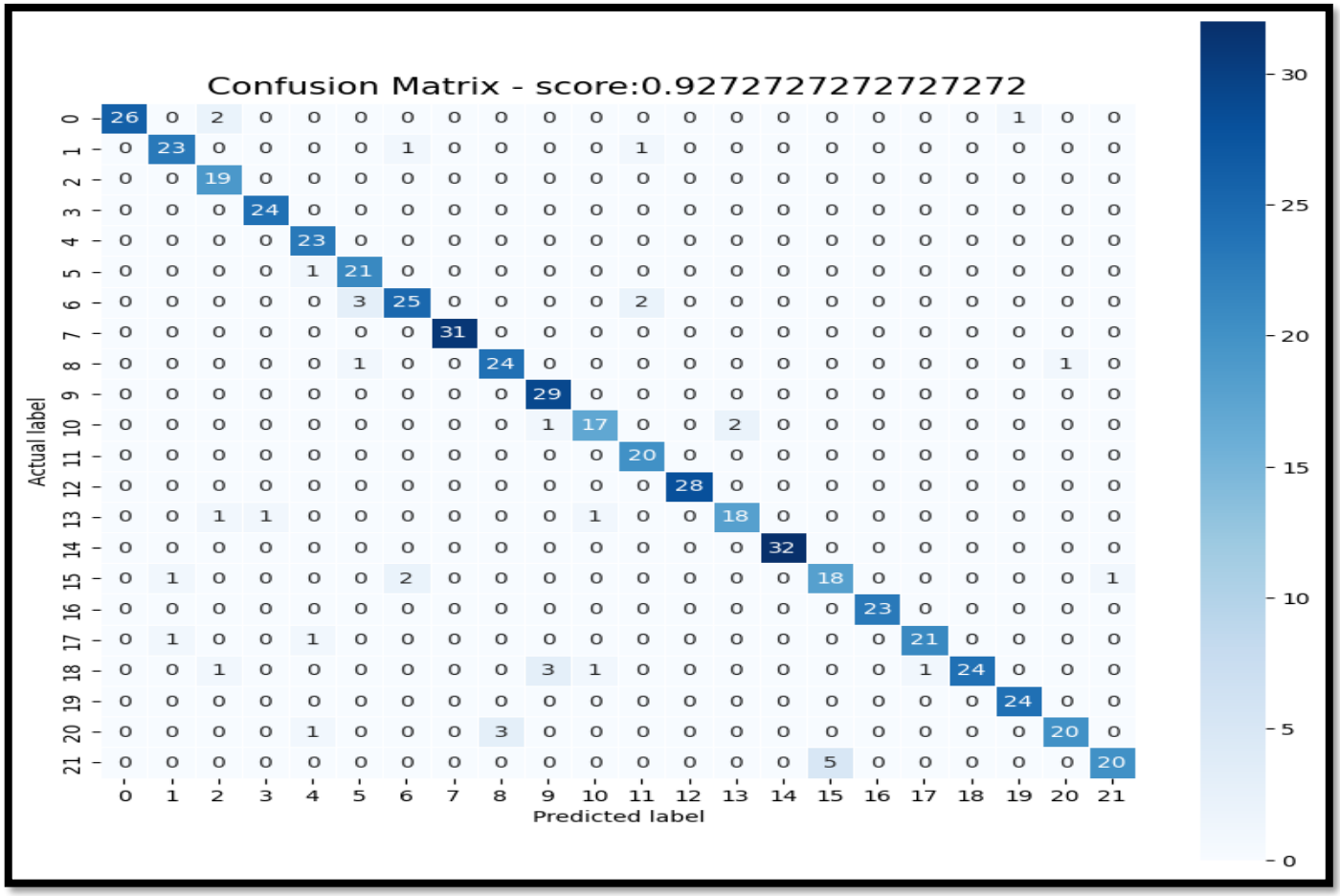
- **Confusion Matrix**

  - It is a much better way to evaluate the classifier

n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

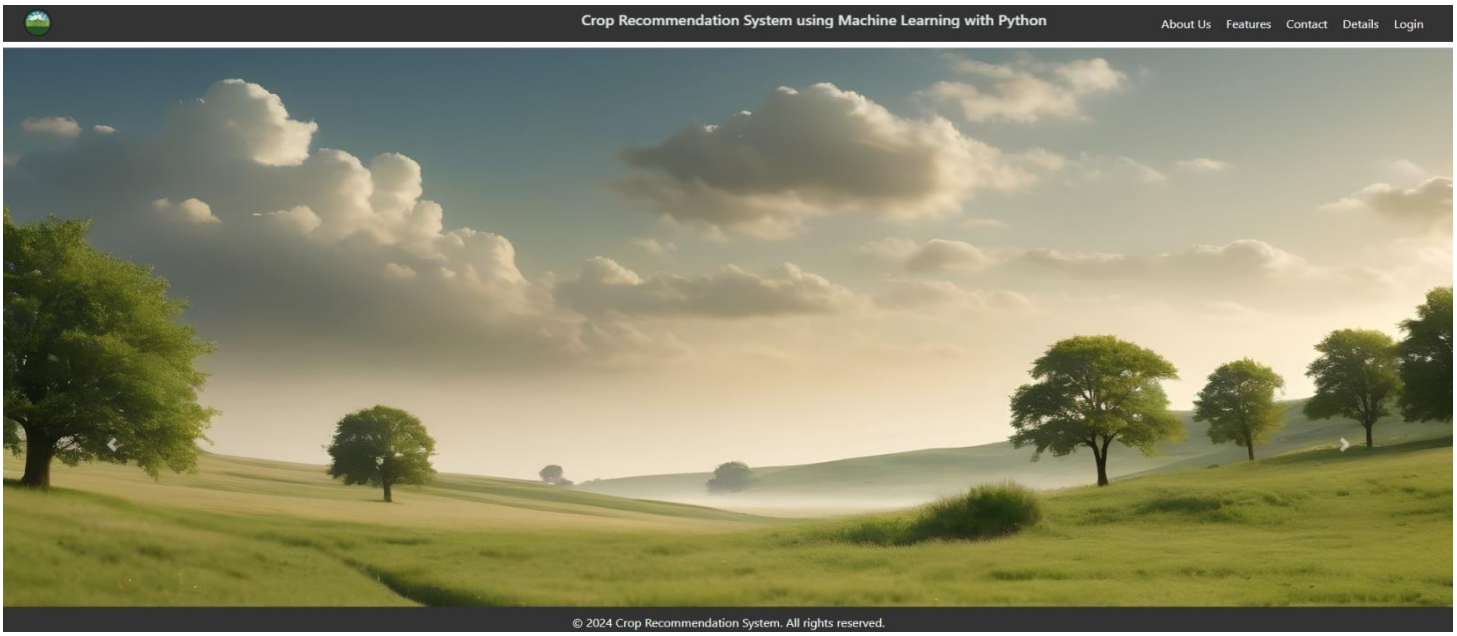
- **True Negative:** Model has given prediction No, and the real or actual value was also No.
- **True Positive:** The model has predicted yes, and the actual value was also true.
- **False Negative:** The model has predicted no, but the actual value was yes, it is also called as Type-II error.
- **False Positive:** The model has predicted Yes, but the actual value was No. It is also called a Type-I error.

Confusion Matrix of our model Logistic Regression



Gui Version of Crop Recommendation System

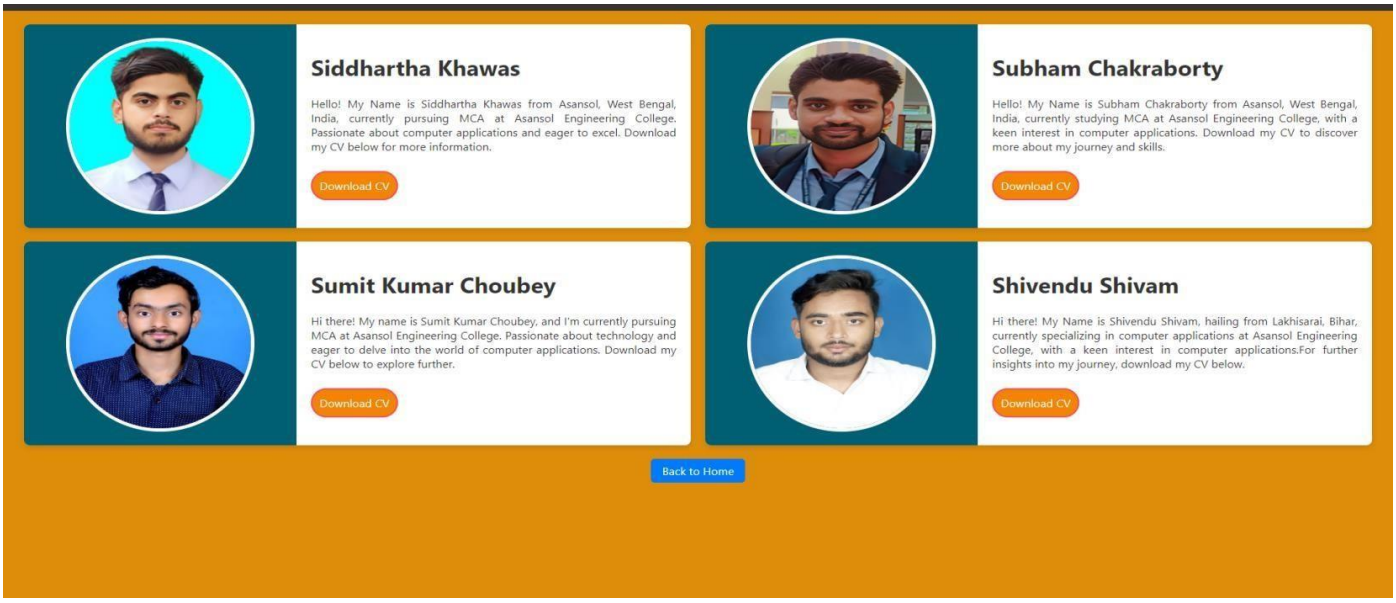
A crop recommendation system utilizes various data points such as nitrogen, phosphorus, potassium, temperature, humidity, ph value and rainfall to suggest suitable crops for farmers to cultivate. These systems often employ machine learning algorithms to analyze data and provide personalized recommendations, helping farmers make informed decisions to maximize yield and profit while minimizing risks.



In this Gui version of crop recommendation system there are some section about our project. Here you can know about these section:-

- About us

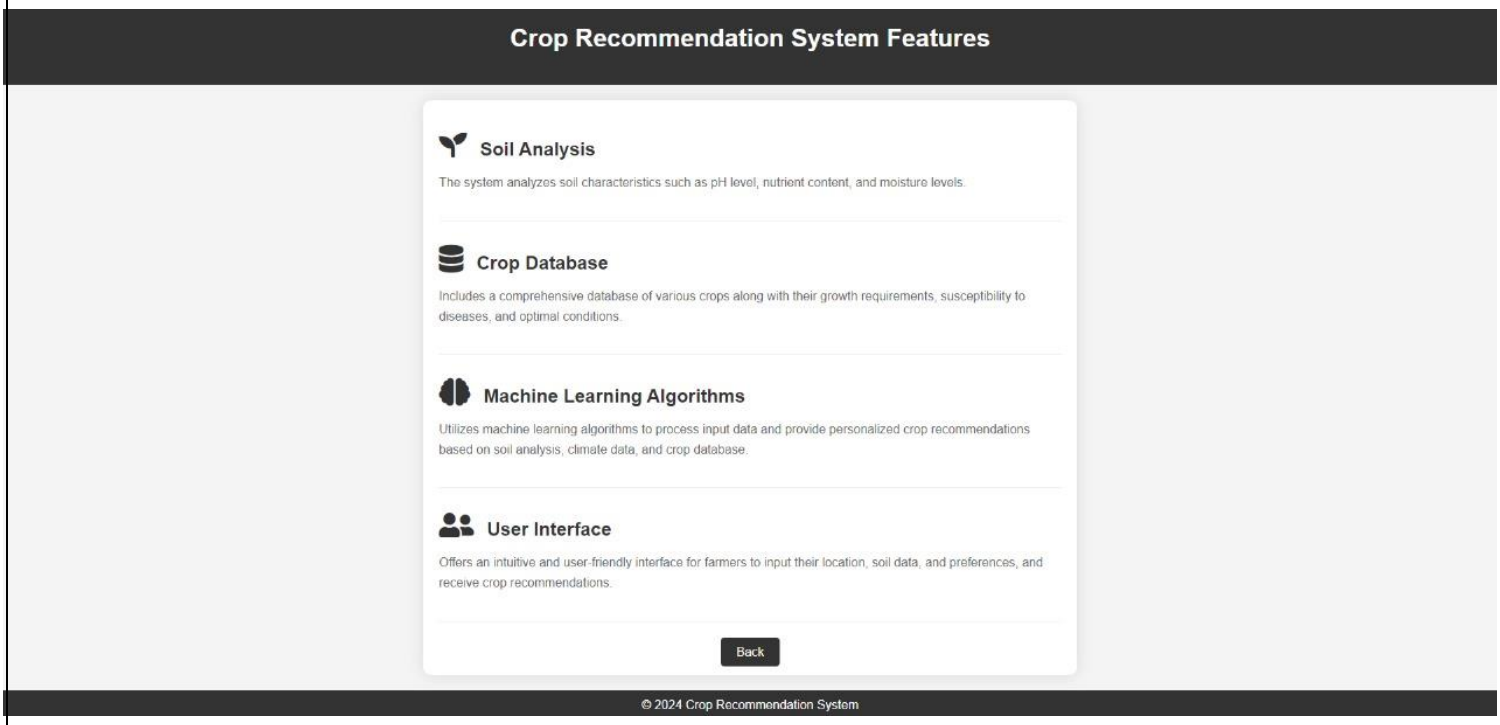
  Our project's "About us" section introduces our talented team of developers, showcasing their diverse skills and experiences. From seasoned professionals to innovative newcomers, our team is united by a passion for creating impactful solutions through technology.



- Features

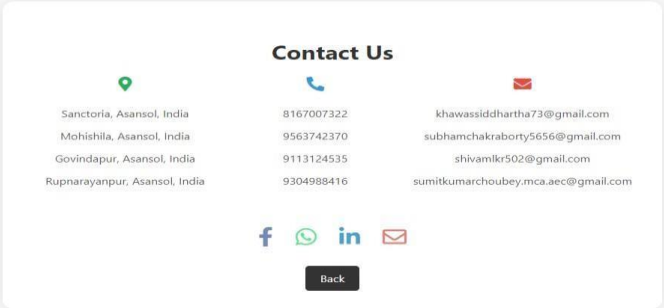
  In the features section of our projects, we highlight the key functionalities that set our solutions apart. From intuitive user interfaces to powerful backend algorithms, each feature is meticulously designed to enhance user experience and deliver tangible value. Some features about our project are given below:-

  - Soil Analysis
  - Crop Database
  - Machine learning Algorithm
  - User Interface



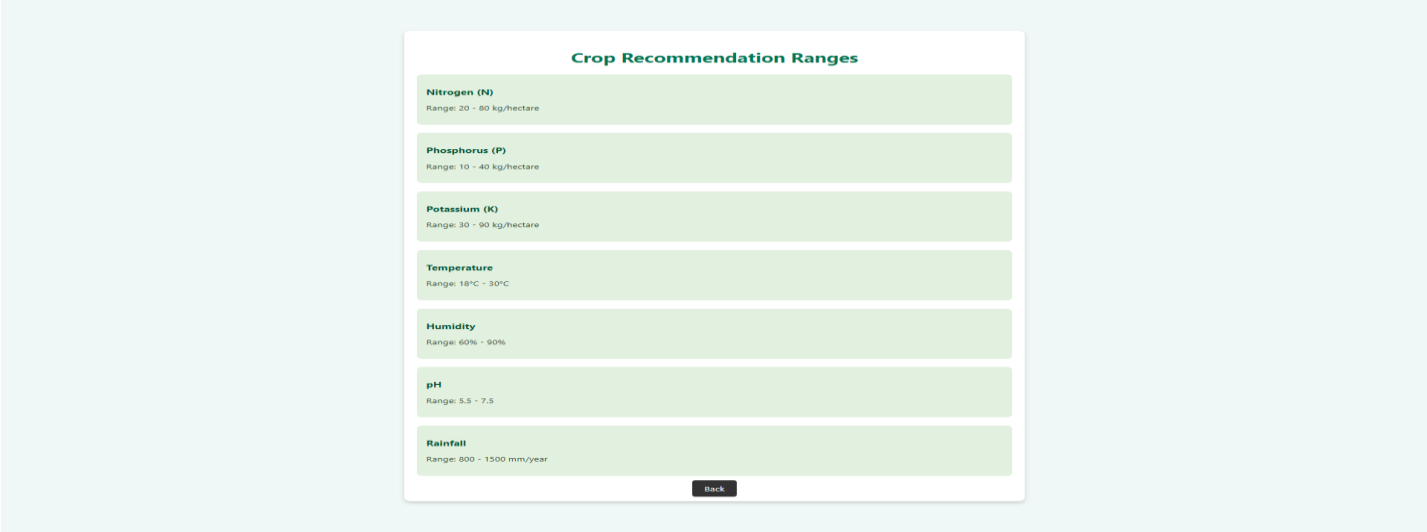
- Contact

  The contact section provides essential information for reaching out, including email addresses and phone numbers, ensuring easy communication between users and project developers.



- Details

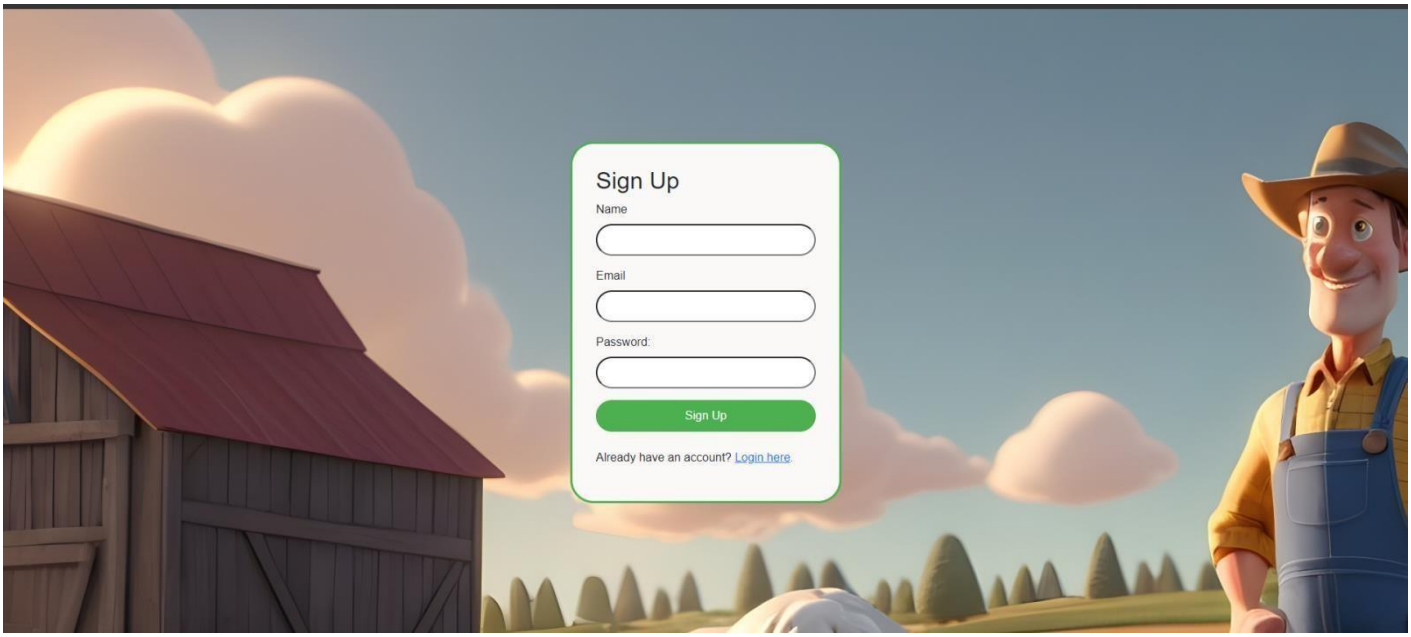
  The details section in our crop recommendation system provides essential parameters such as nitrogen, phosphorus, and potassium ranges, along with environmental factors like temperature, humidity, pH value, and rainfall ranges. These ranges serve as crucial guidelines for farmers to optimize their crop cultivation practices effectively.



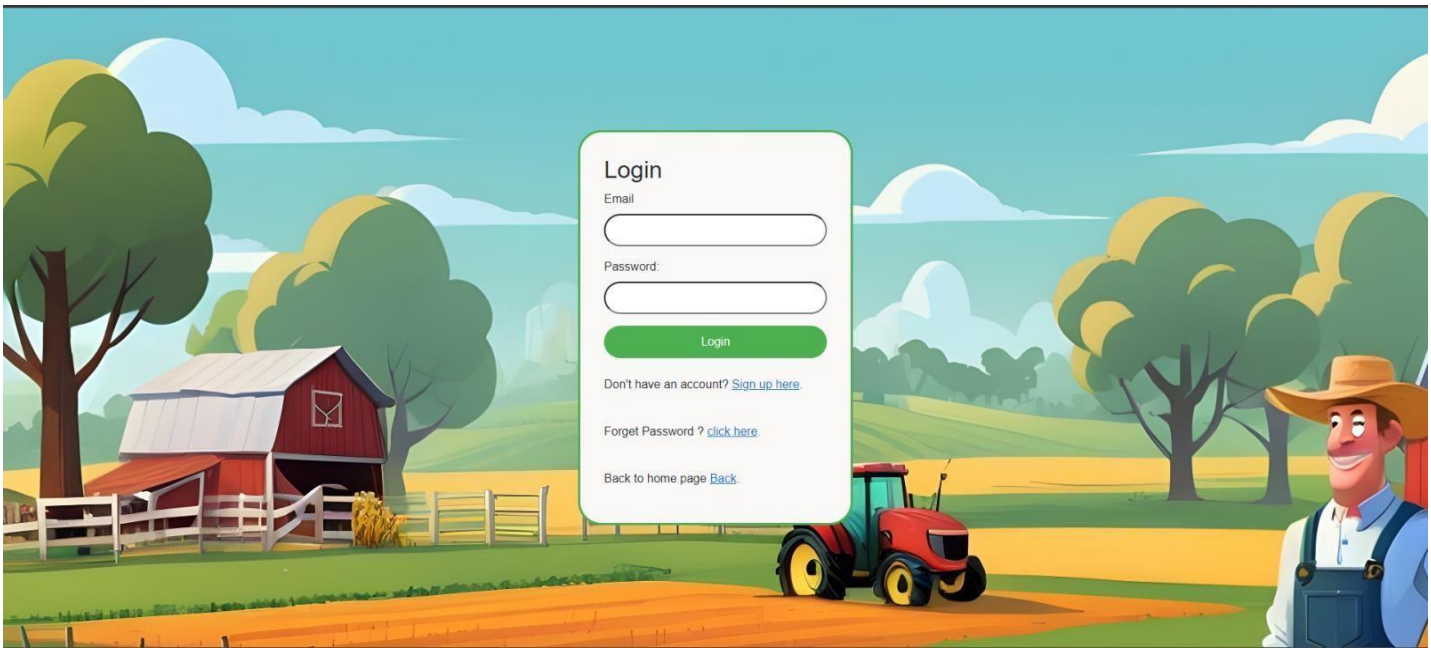
- Login

  In this login page we can login and fill necessary details and know which crop is suitable for that place. There are some key point about login page:-

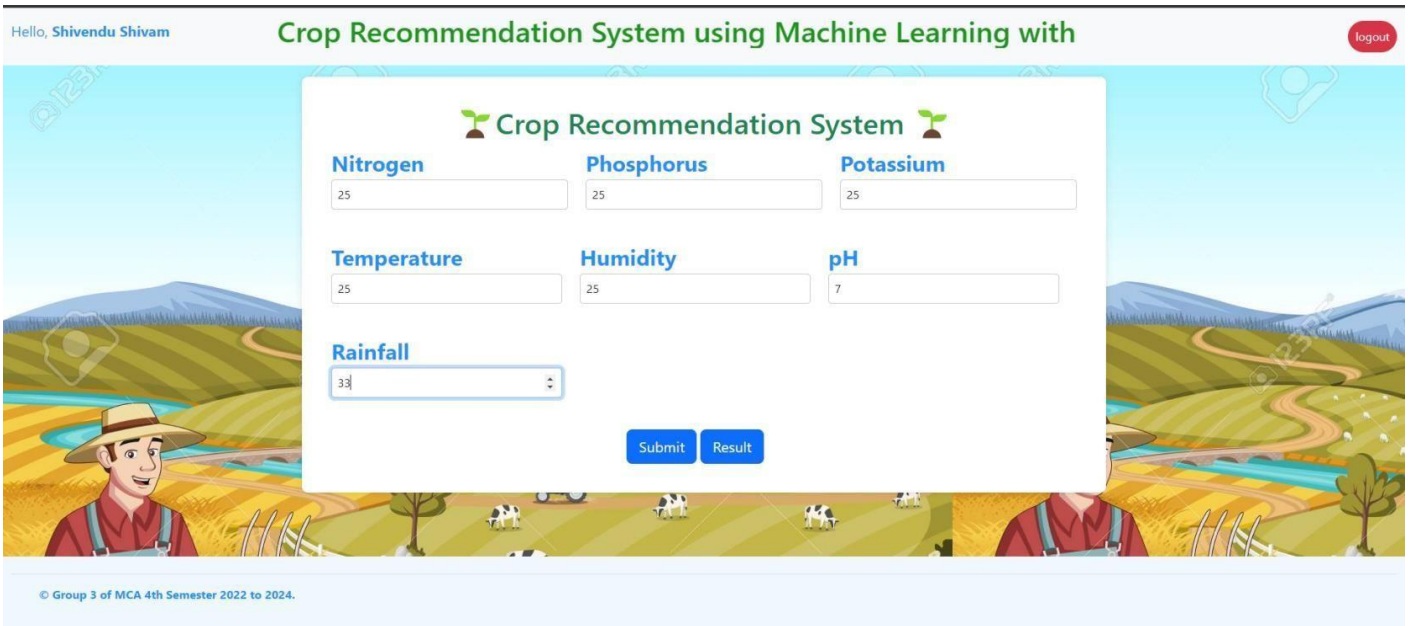
  - Before login you have to sign up with the help of sign up button which are given in the login page.
  - When you click in the sign up button a sign up page will open where you have to fill all required details which is present in sign up page.

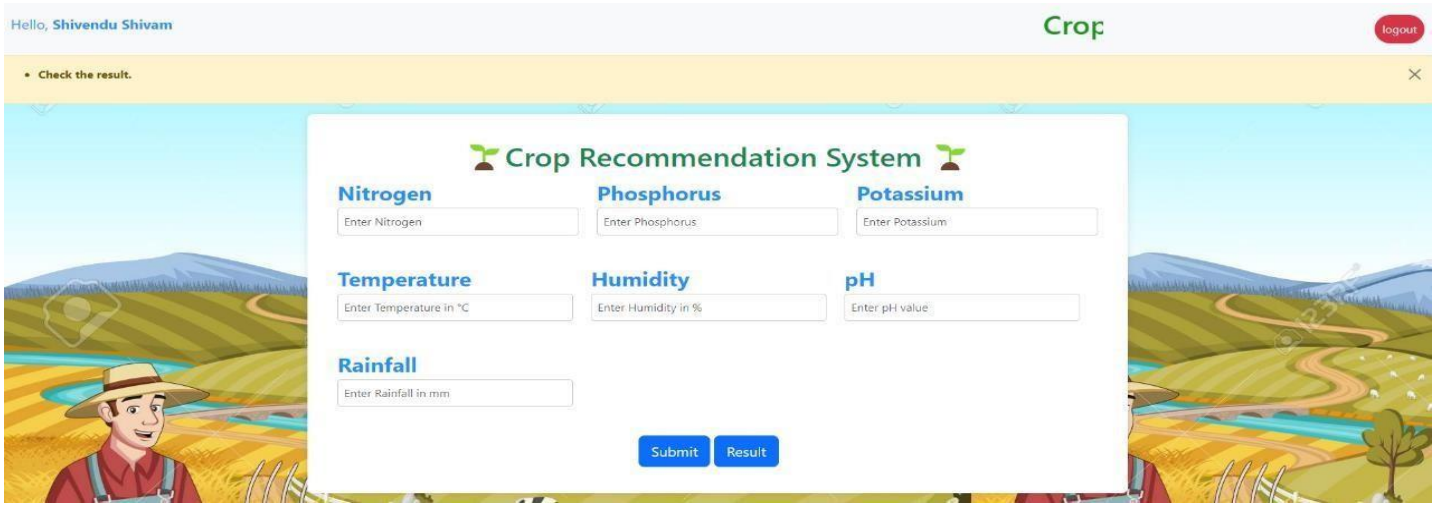


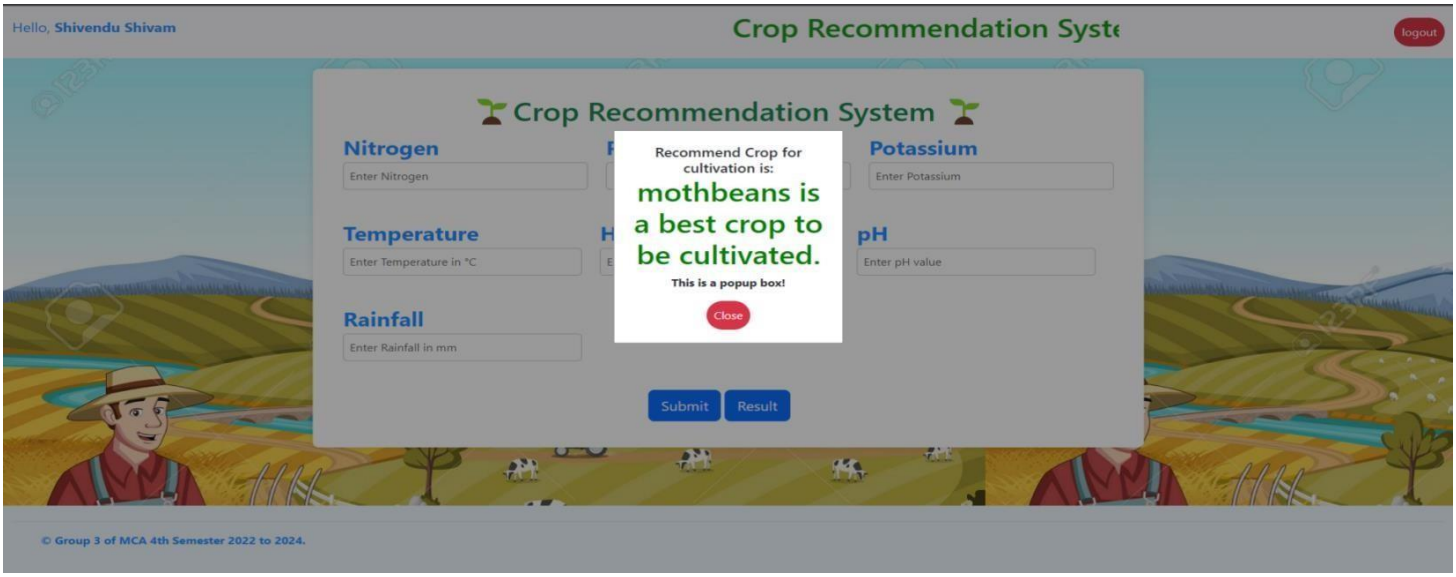
- After sign up you will get your login id and password and with the help of that login id and password you can login in that portal.



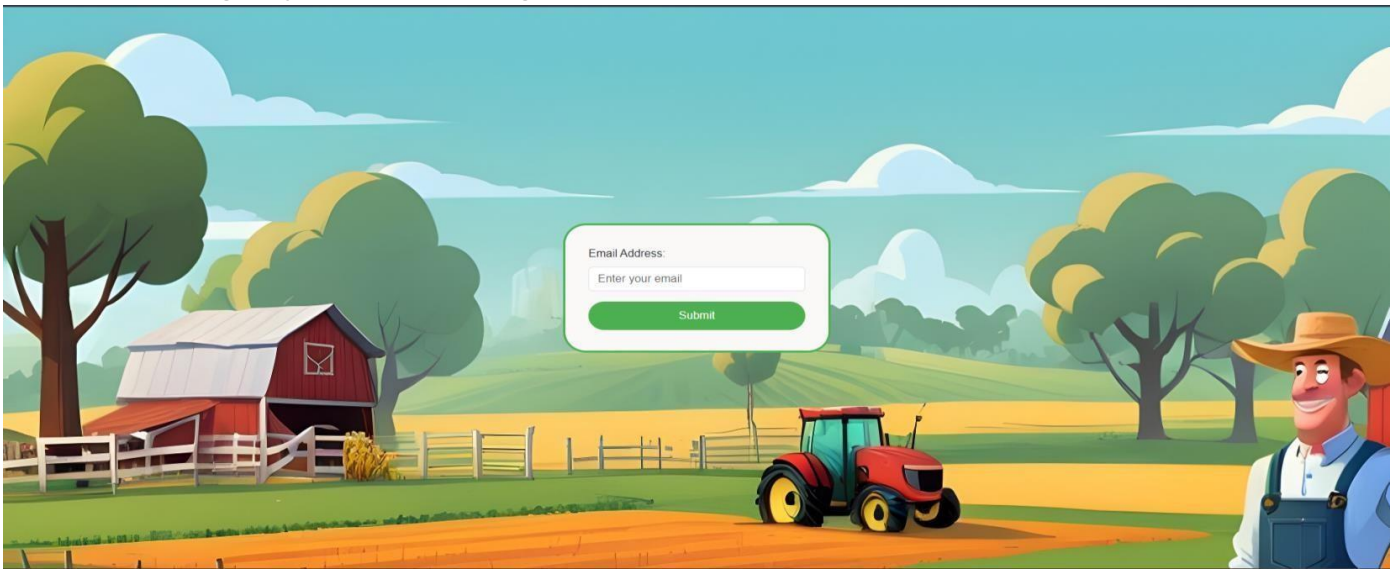
- After login a page will open in front of you where you have to fill all the data. After filling all that data you have to click submit button. After submit all the data a message will show "Check the result". After showing that message you have to click result button then a popup message will show with your crop name.

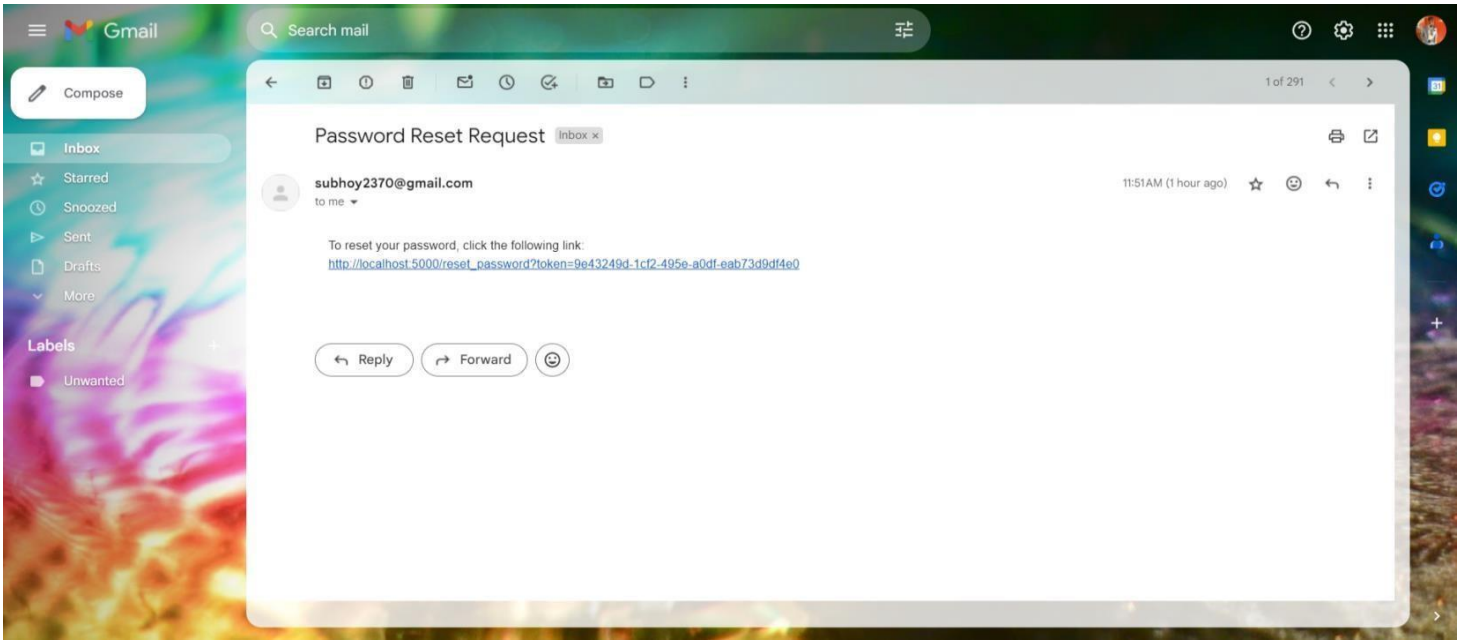


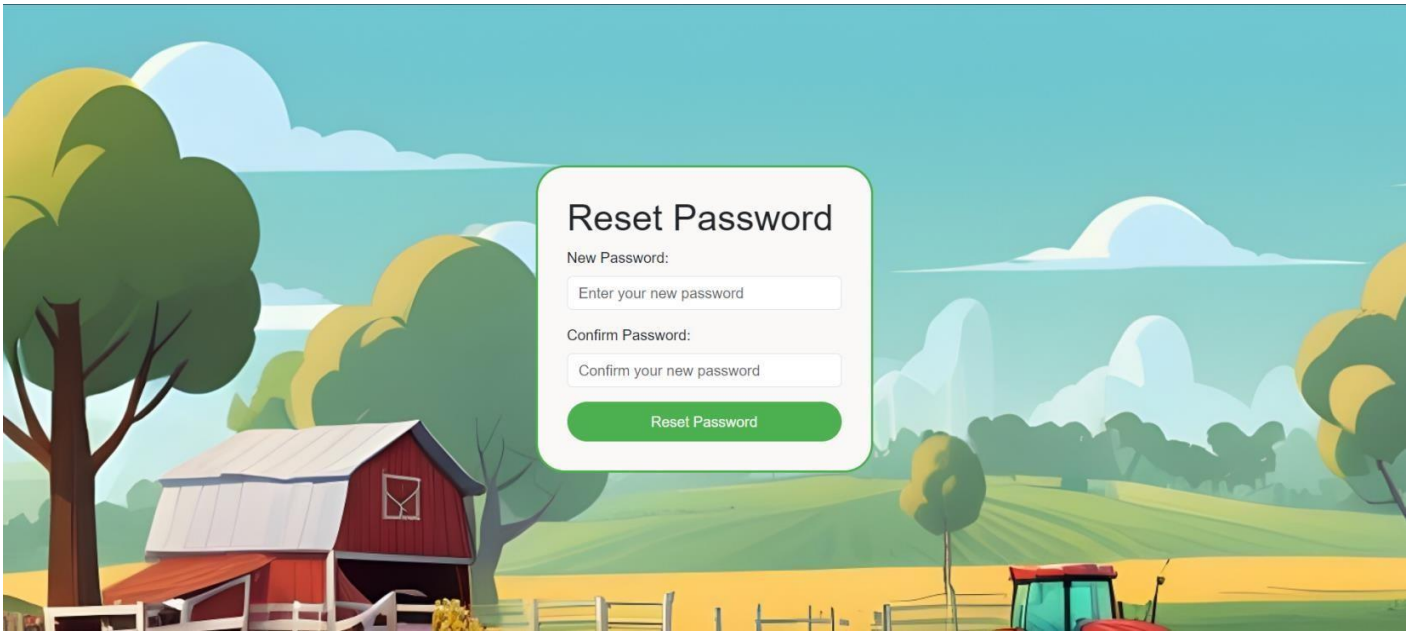




- In case you forgot your password and you are unable to login then you can use our forget password feature. In this feature you have to click forget password option after click that option a page will open where you have to fill your registered email account and the click submit button. After that a link will sent in to your mail with the help of that link you can reset the password and again you are able to login.




Gmail
Search mail
Compose
Inbox
Starred
Snoozed
Sent
Drafts
More
Labels
Unwanted
1 of 291
Password Reset Request
Inbox ×
subhoy2370@gmail.com
to me
11:51AM (1 hour ago)
To reset your password, click the following link:
http://localhost:5000/reset_password?token=9e43249d-1cf2-495e-a0df-eab73d9df4e0
Reply
Forward


Reset Password
New Password:
Enter your new password
Confirm Password:
Confirm your new password
Reset Password

Code

By combining these imports, you're setting up your Python environment for data analysis, visualization, and machine learning tasks while also configuring it to handle warnings in a specific way.

```
from __future__ import print_function

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import classification_report

from sklearn import metrics from

sklearn import tree import warnings

warnings.filterwarnings('ignore')
```

This code allows the user to upload a file from their local system to a Google Colab notebook environment, and the uploaded file(s) information is stored in the uploaded variable for further processing within the notebook.

```
from google.colab import drive

drive.mount('/content/gdrive')
```

Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).

This code reads the contents of a CSV file named 'modified_crop.csv' using Pandas and stores it in a DataFrame named crop for further processing and analysis in Python.

```
crop = pd.read_csv('/content/gdrive/My Drive/modified_crop.csv') crop
```

This code is use for quickly check the size of your DataFrame, which is especially useful when dealing with large datasets or when you need to understand the structure of the data you're working with.

For example, if you run crop.shape and it returns (100, 5), it means that the DataFrame crop has 100 rows and 5 columns. and the number of elements is 500.

```
crop.shape
```

(2550, 8)

By using crop.info(), you can quickly get an overview of the DataFrame, including its size, data types, and missing values, which is useful for initial data exploration and understanding the dataset's characteristics.

```
crop.info()
```

<class 'pandas.core.frame.DataFrame'> RangeIndex:
2550 entries, 0 to 2549

```
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   N             2530 non-null   float64
 1   P             2531 non-null   float64
 2   K             2528 non-null   float64
 3   temperature   2524 non-null   float64
 4   humidity      2532 non-null   float64
 5   ph            2533 non-null   float64
 6   rainfall      2529 non-null   float64
 7   label         2529 non-null   object 
dtypes: float64(7), object(1) memory
usage: 159.5+ KB
```

The purpose of using head() is to quickly inspect the structure and content of the DataFrame. It's often used as an initial step in data analysis to get a sense of what the data looks like before performing further operations. By examining the first few rows, you can check the column names, data types, and example values in the DataFrame.

```
crop.head()
```

The purpose of using tail() is to quickly inspect the end of the DataFrame. It's often used to check for patterns or trends in the data, especially if the data is ordered chronologically or by some other criteria. By examining the last few rows, you can see the most recent data entries and verify that the DataFrame has been properly loaded or processed.

```
crop.tail()
```

crop.isnull().sum() gives you a Series where each entry represents the number of missing values in the corresponding column of the crop DataFrame. This information is useful for identifying and handling missing data in your dataset.

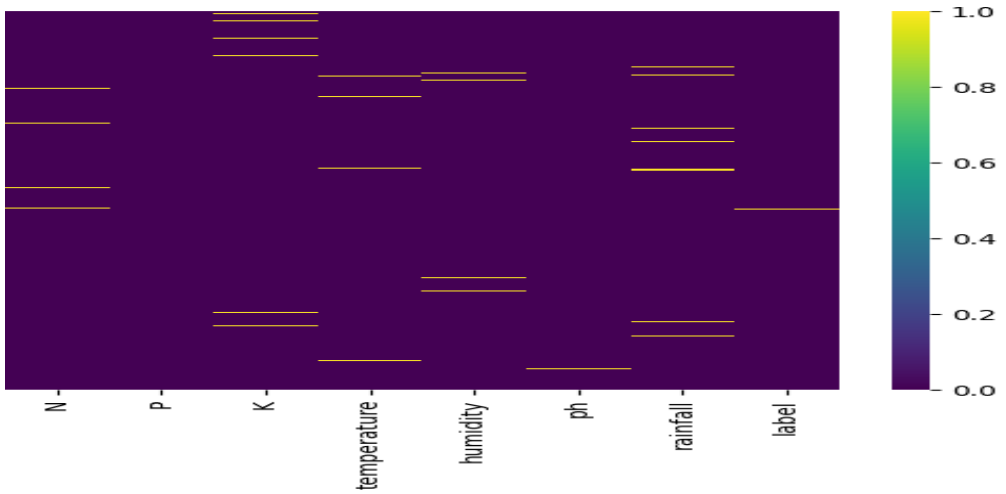
```
crop.isnull().sum()
```

```
N              20
P              19
K              22
temperature    26
humidity       18
ph             17
rainfall       21
label          21
dtype: int64
```

image.png

```
sns.heatmap(crop.isnull(),yticklabels=False,cbar=True,cmap='viridis')
```

<Axes: >



crop.isnull().sum().sum() gives you the total number of missing values in the entire DataFrame crop, summing up the counts of missing values across all columns. This information is valuable for understanding the extent of missing data in your dataset

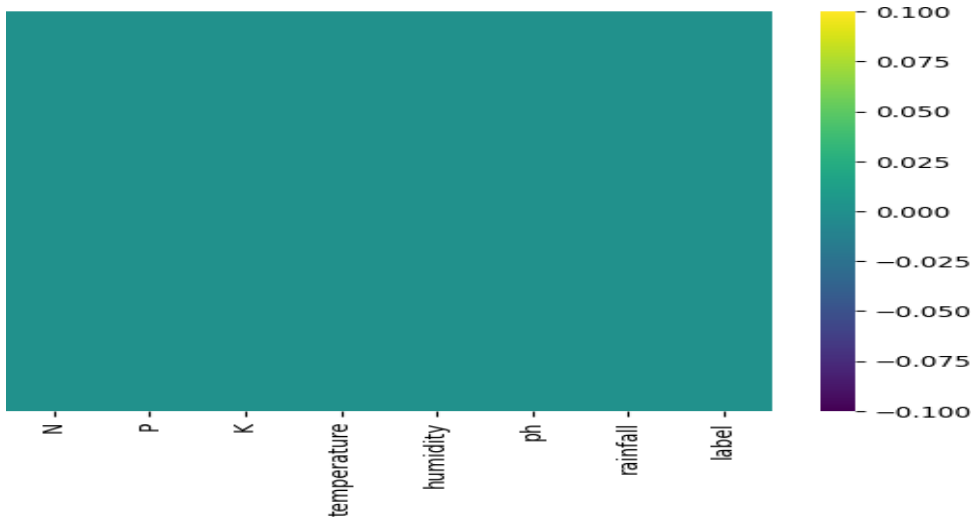
crop.isnull().sum().sum() 164

crop.duplicated().sum() gives you the total number of duplicated rows in the DataFrame crop. This information is useful for identifying and handling duplicate entries in your dataset.

crop.duplicated().sum()

350

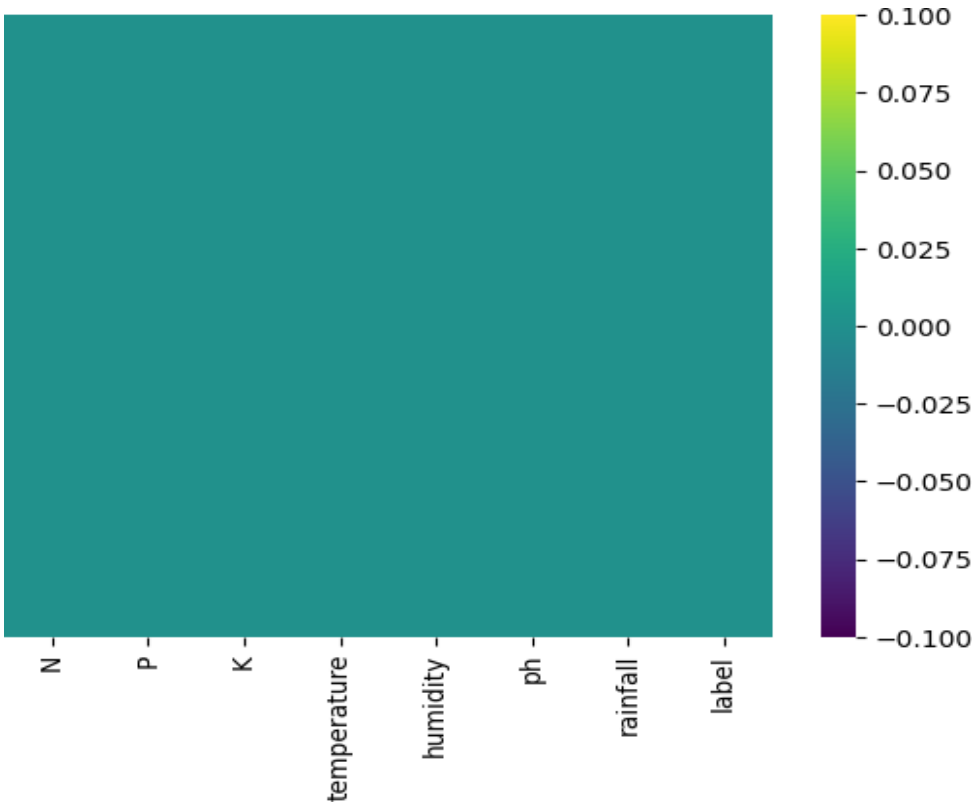
```
crop1 = crop
crop1.N.fillna(0, inplace=True)

crop1.P.fillna(0, inplace=True)

crop1.K.fillna(0, inplace=True)

crop1.temperature.fillna(0, inplace=True)

crop1.humidity.fillna(0, inplace=True)

crop1.ph.fillna(0, inplace=True)

crop1.rainfall.fillna(0, inplace=True)

crop1.label.fillna(crop1.label.mode()[0], inplace=True)

sns.heatmap(crop1.isnull(),yticklabels=False,cbar=True,cmap='viridis')

crop1.describe(include="all")
```



```
crop1.isnull().sum()
```

```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64 sns.heatmap(crop.isnull(),yticklabels=False,cbar=True,cmap='viridis')
```

<Axes: >



the DataFrame crop2 contains the data from crop1 with duplicate rows removed. This operation ensures that each row in crop2 is unique, based on all columns by default.

```
crop2 = crop1.drop_duplicates()
```

```
crop2.duplicated().sum()
```

0

returns a Series where each entry represents whether there are any missing values in the corresponding column of the DataFrame crop2. If the entry is True, it means that the column contains at least one missing value; otherwise, it's False.

```
crop2.isnull().any()
```

```
N              False
P              False
K              False
temperature    False
humidity       False
ph             False
rainfall       False
label          False
dtype:  bool
```

```
crop2.shape
```

```
(2200, 8)
```

The output of crop2.describe() will be a DataFrame where each row represents a summary statistic, and each column represents a numerical column in the original DataFrame crop2. This summary statistics can provide insights into the distribution and spread of values in the dataset, helping with data exploration and analysis.

```
crop2.describe()
```

Count: The total number of non-missing values in each column.

- Mean: The average value of the column.
- Std (Standard Deviation): Measures the amount of variation or dispersion of the data.
- Min: The smallest value in the column.
- 25% (First Quartile): The value below which 25% of the data falls.
- 50% (Median): The middle value of the data.
- 75% (Third Quartile): The value below which 75% of the data falls.
- Max: The largest value in the column.

The corr() method is commonly used to identify relationships between variables in a dataset. High positive or negative correlation coefficients can indicate strong relationships between variables, while a correlation coefficient close to 0 suggests little to no relationship. These correlation coefficients can be further analyzed and interpreted to gain insights into the dataset.

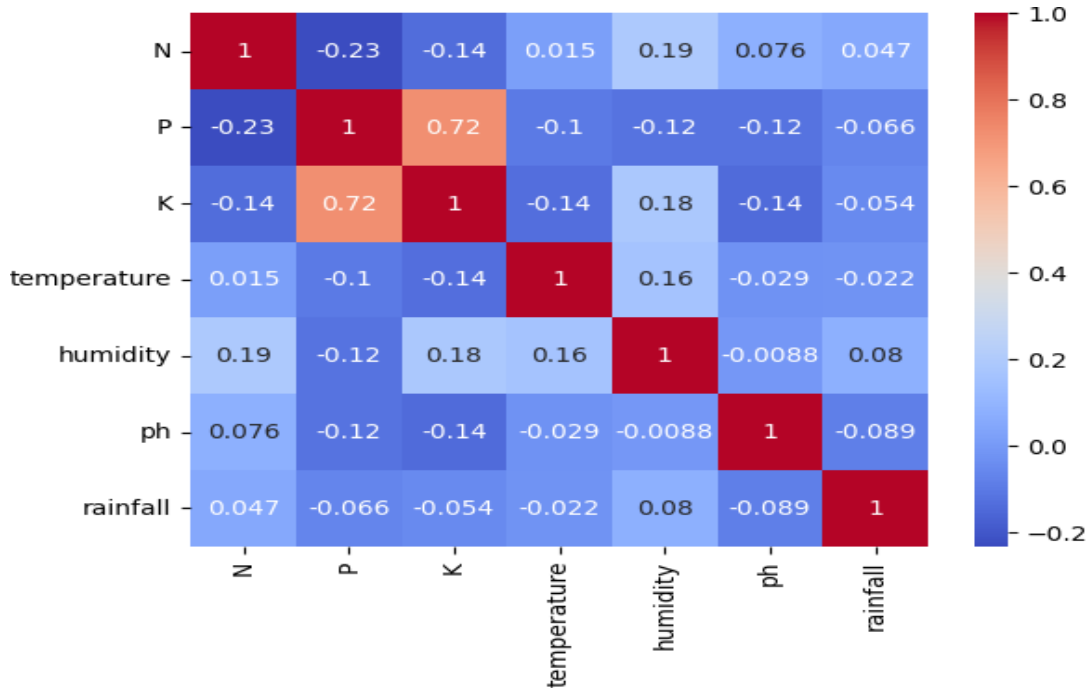
```
crop2_numeric = crop2.select_dtypes(include=[np.number]) corr =
crop2_numeric.corr()
corr
```

- This correlation matrix represents the correlation coefficients between pairs of numeric variables in the dataset.
- The value of 1.000 along the diagonal represents the correlation of each variable with itself, which is always perfect (perfect positive correlation).
- Off-diagonal values represent the correlation between pairs of variables. Each cell contains the correlation coefficient, which ranges from -1 to 1.
- A value of 1 indicates a perfect positive correlation, meaning that as one variable increases, the other also increases proportionally.
- A value of -1 indicates a perfect negative correlation, meaning that as one variable increases, the other decreases proportionally.
- A value close to 0 indicates little to no linear correlation between the variables.
- For example, the correlation coefficient between 'N' and 'P' is approximately -0.233, indicating a moderate negative correlation between these two variables.
- Similarly, the correlation coefficient between 'K' and 'humidity' is approximately 0.182, indicating a moderate positive correlation between these two variables.

By visualizing the correlation matrix as a heatmap, you can easily identify patterns and relationships between variables in the dataset. Positive correlations will appear in warm colors, negative correlations in cool colors, and no correlation in neutral colors. The annotations provide the exact correlation coefficients, making it easier to interpret the heatmap.

```
sns.heatmap(corr,annot=True,cbar=True,  cmap='coolwarm')
```

<Axes: >



Comments:

- The sns.heatmap()function from the Seaborn library is used to create a heatmap.
- The corrparameter specifies the correlation matrix that we want to visualize.
- Setting annot=Trueadds annotations to each cell of the heatmap, displaying the correlation coefficients.
- The cbar=Trueparameter adds a color bar to the side of the heatmap, indicating the scale of the colors.
- The cmap='coolwarm'parameter sets the color map to 'coolwarm', which ranges from cool (blue) to warm (red), representing negative and positive correlations respectively.
- This visualization helps in understanding the strength and direction of the linear relationships between pairs of variables in the dataset, making it easier to identify patterns and dependencies.

The .size method in Python is used to get the number of elements in an object, such as a list, tuple, set, or dictionary. However, it seems you've added parentheses to the method, which is not correct for this purpose.

```
crop2.size
```

17600

- This line of code calculates the total number of elements in the DataFrame crop2.
- The size attribute of a DataFrame returns the total number of elements, which is equal to the number of rows multiplied by the number of columns.
- The result represents the total number of data points or cells in the DataFrame crop2, including both numeric and categorical values.

The .columns attribute is used to retrieve the column labels of a DataFrame in pandas.

```
crop2.columns
```

Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')

- This line of code retrieves the column names of the DataFrame crop2.
- The columns attribute of a DataFrame returns an Index object containing the column names.
- The result represents the names of all columns in the DataFrame crop2.

The code crop2['label'].unique() is used to get the unique values in the 'label' column of the DataFrame crop2.

```
crop2['label'].unique()
```

```
array(['rice', 'muskmelon', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas', 'mothbeans', 'mungbean',
       'blackgram', 'lentil', 'pomegranate', 'banana', 'mango', 'grapes', 'watermelon', 'apple', 'orange',
       'papaya', 'coconut', 'cotton', 'jute', 'coffee'], dtype=object)
```

Comments:

- This line of code accesses the 'label' column of the DataFrame crop2.
- The .unique()method is then applied to this column, which returns an array of unique values present in the 'label' column.
- The result represents an array containing all unique values found in the 'label' column of the DataFrame crop2.
- This operation is useful for understanding the different categories or classes present in the 'label' column, providing insights into the diversity of crops or labels present in the dataset.

The dtypes attribute in pandas DataFrame is used to get the data types of each column in the DataFrame.

```
crop2.dtypes
```

```
N              float64
P              float64
K              float64
temperature    float64
humidity       float64
ph             float64
rainfall       float64
label           object
dtype: object
```

```
crop2
```

Comments:

- This line of code returns the data types of each column in the DataFrame crop2.
- The dtypesattribute of a DataFrame provides information about the data type of each column.
- The result is a Series where the index represents column names, and the values represent the corresponding data types of each column in crop2.
- Understanding the data types is important for data manipulation and analysis, as it informs about the nature of the data stored in each column (e.g., integer, float, object, etc.), helping in appropriate data handling and processing.
- The output of crop2['label'].value_counts() will be a Series where each unique label in the 'label' column of crop2 is listed along with the count of occurrences of that label in the dataset. This information is useful for understanding the distribution of different labels in the dataset and can be valuable for various analytical purposes.

```
crop2['label'].value_counts()
```

```
label
muskmelon     117
mango         100
jute          100
chickpea      100
kidneybeans   100
coconut       100
mothbeans     100
papaya        100
blackgram     100
apple         100
grapes        100
rice           99
banana         99
pomegranate    99
lentil         99
pigeonpeas     99
watermelon     98
orange         98
mungbean       98
cotton         98
```

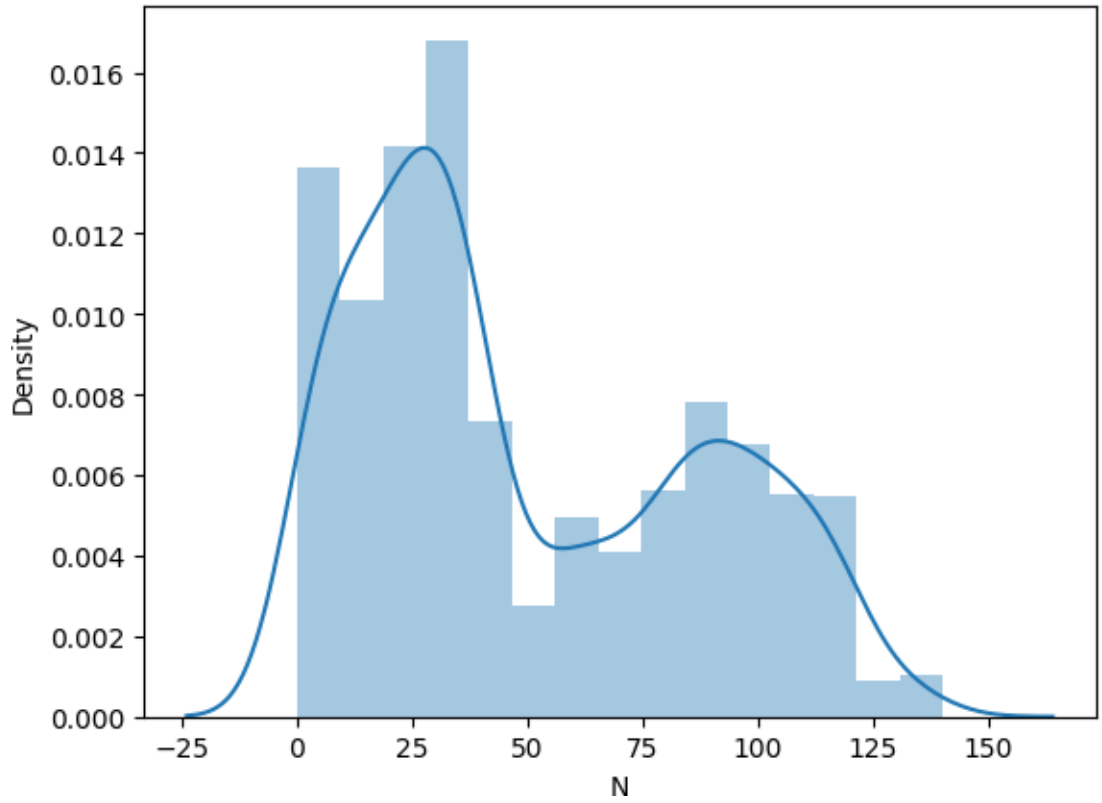
maize 98
coffee 98
Name: count, dtype: int64

Comments:

- This line of code accesses the 'label' column of the DataFrame crop2.
- The .value_counts()method is then applied to this column, which returns a Series containing the count of each unique value in the 'label' column.
- The index of the resulting Series represents the unique values found in the 'label' column, and the corresponding values represent the frequency of each unique value.
- This operation is useful for understanding the distribution of different categories or classes in the 'label' column, providing insights into the frequency of occurrence of each crop label in the dataset.

By executing this code, you'll generate a distribution plot showing the distribution of values in the 'N' column of the DataFrame crop2. This visualization helps in understanding the distribution of values and identifying any patterns or outliers present in the data.

sns.distplot(crop2['N']) plt.show()

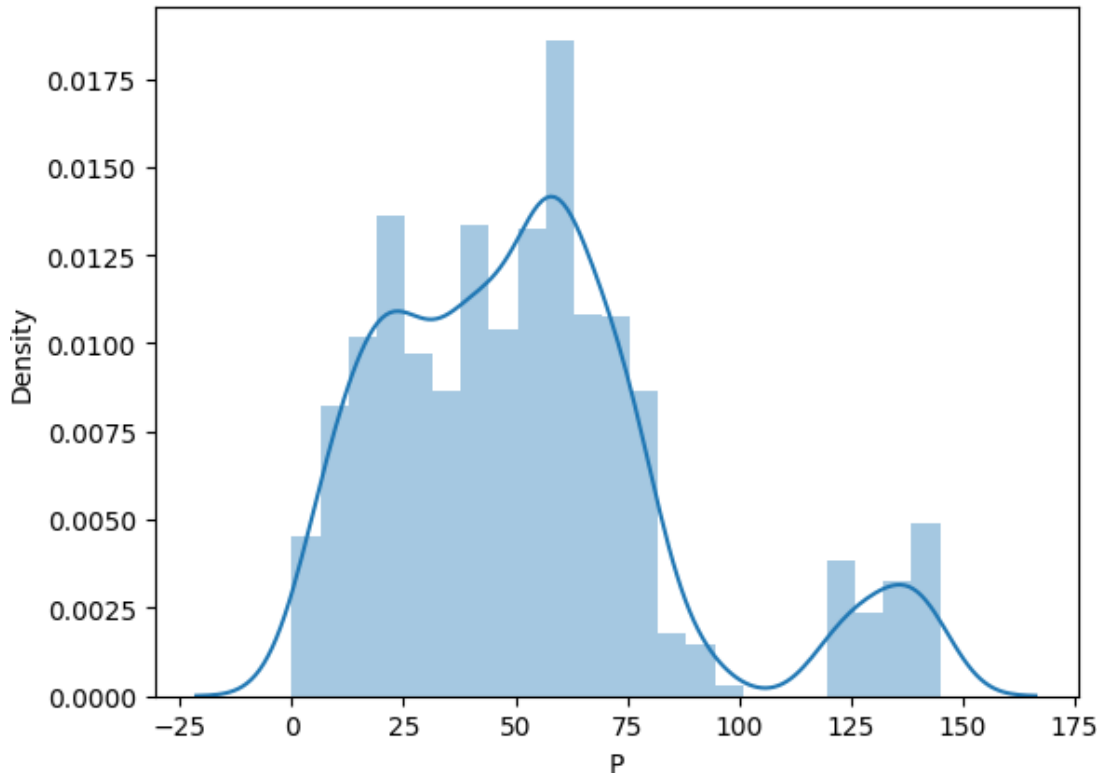


Comments:

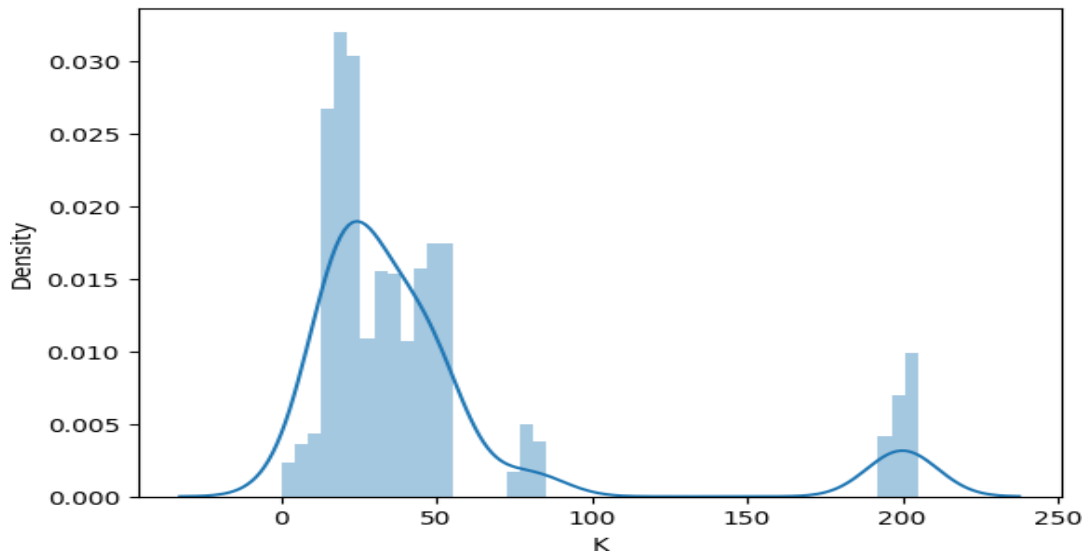
- The sns.distplot()function from the Seaborn library is used to create a distribution plot.
- The column 'N' from the DataFrame crop2is passed as the data to be plotted.
- This plot combines a histogram and a kernel density estimation (KDE) plot to visualize the distribution of values in the 'N' column.
- The plt.show()function is used to display the plot.
- This visualization provides insights into the distribution of values in the 'N' column, including measures of central tendency and spread.

By executing this code, you'll generate a distribution plot showing the distribution of values in the 'P' column of the DataFrame crop2. This visualization helps in understanding the distribution of values and identifying any patterns or outliers present in the data.

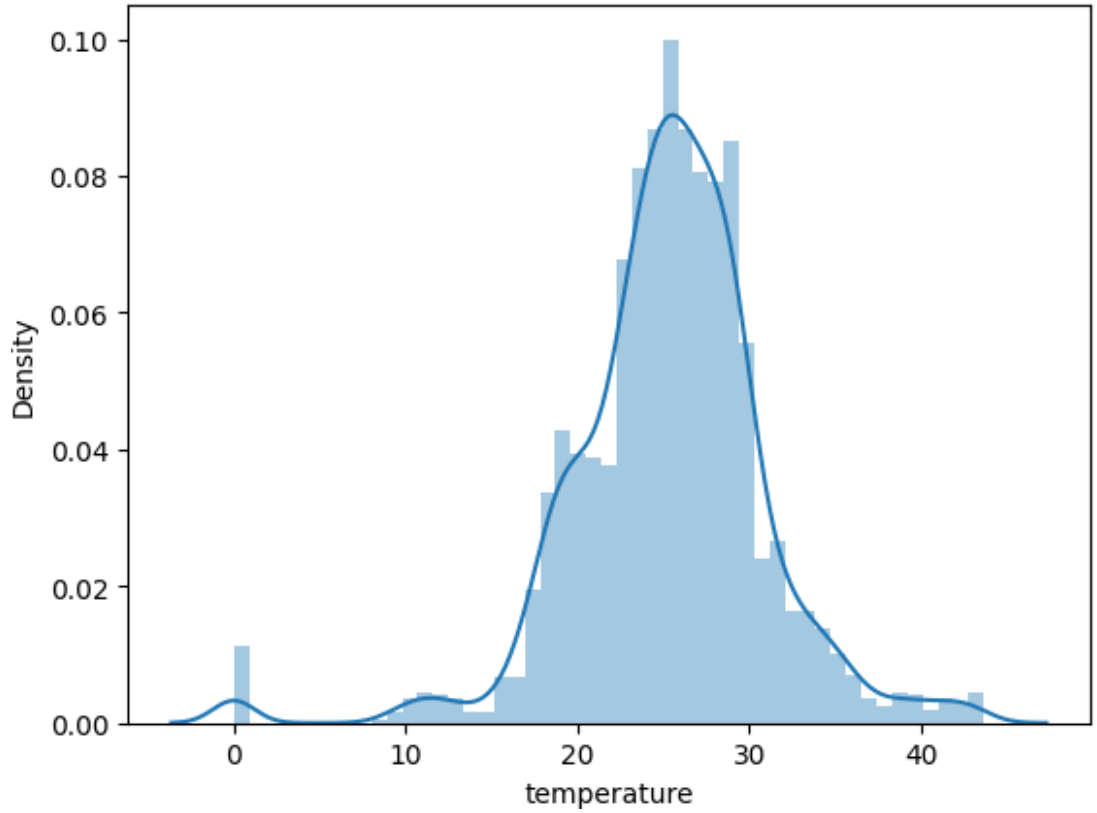
```
sns.distplot(crop2['P']) plt.show()
```



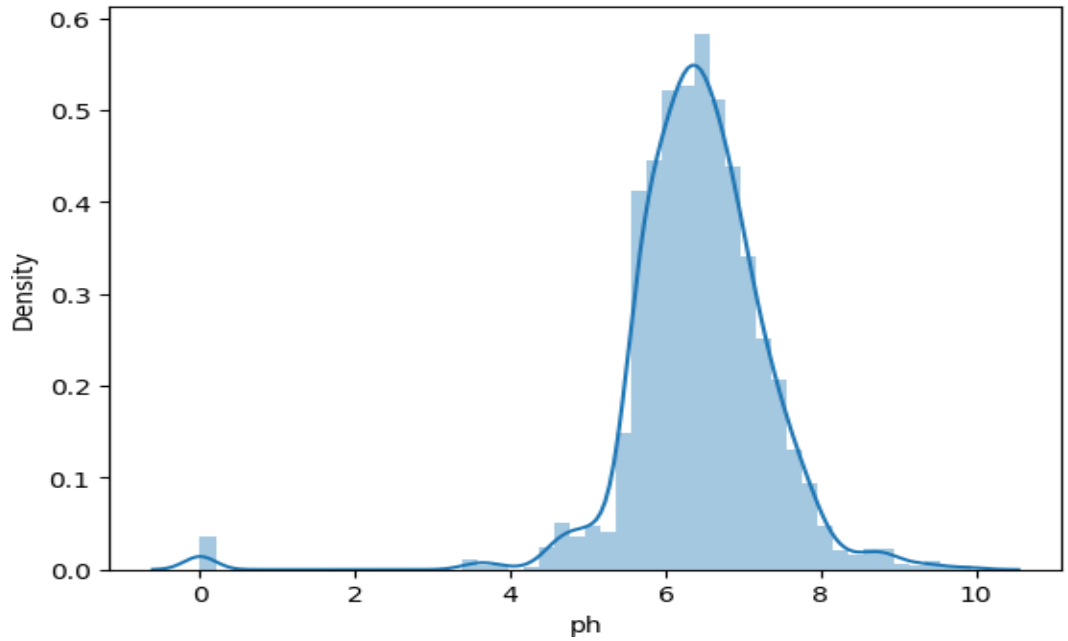

```
sns.distplot(crop2['K']) plt.show()
```



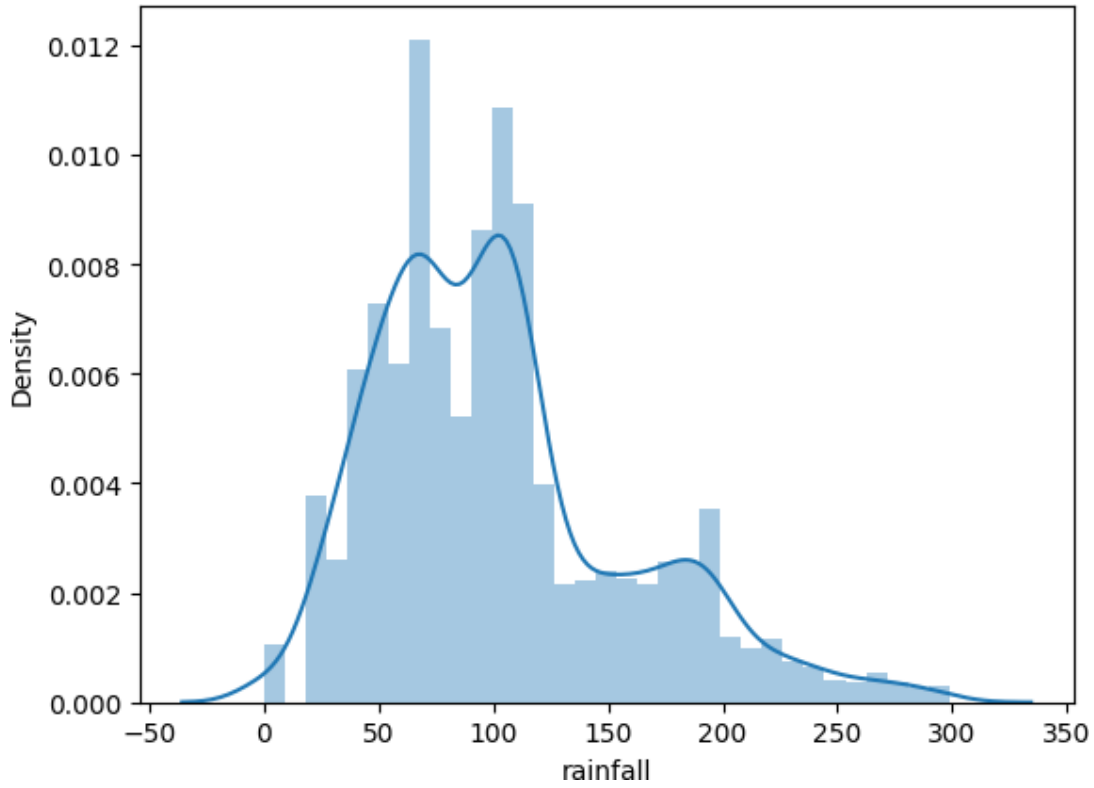
```
sns.distplot(crop2['temperature']) plt.show()
```



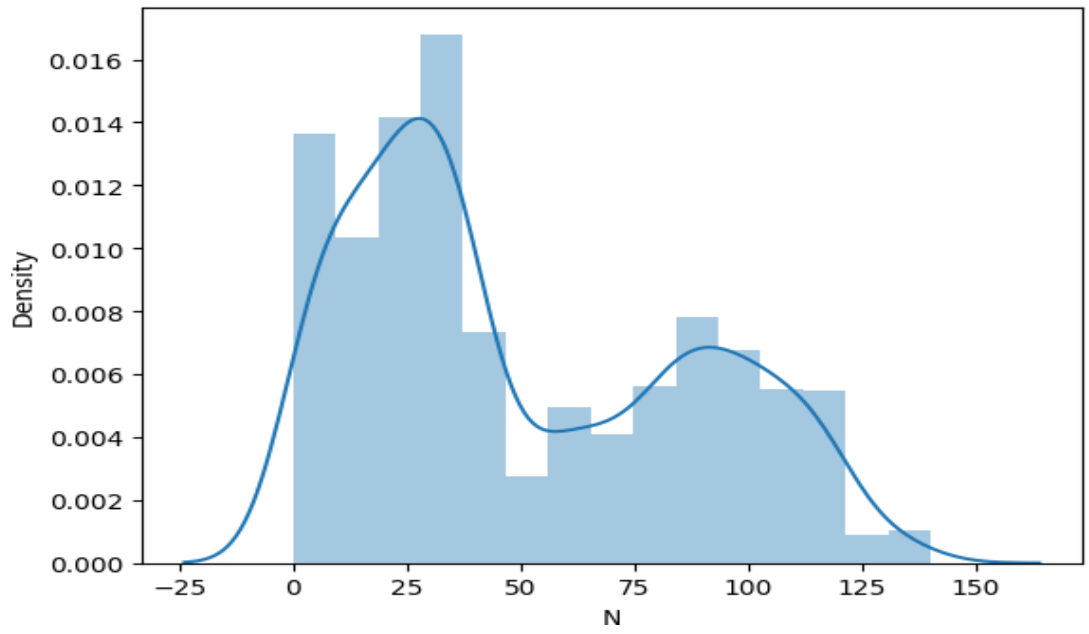

```
sns.distplot(crop2['ph']) plt.show()
```



```
sns.distplot(crop2['rainfall']) plt.show()
```

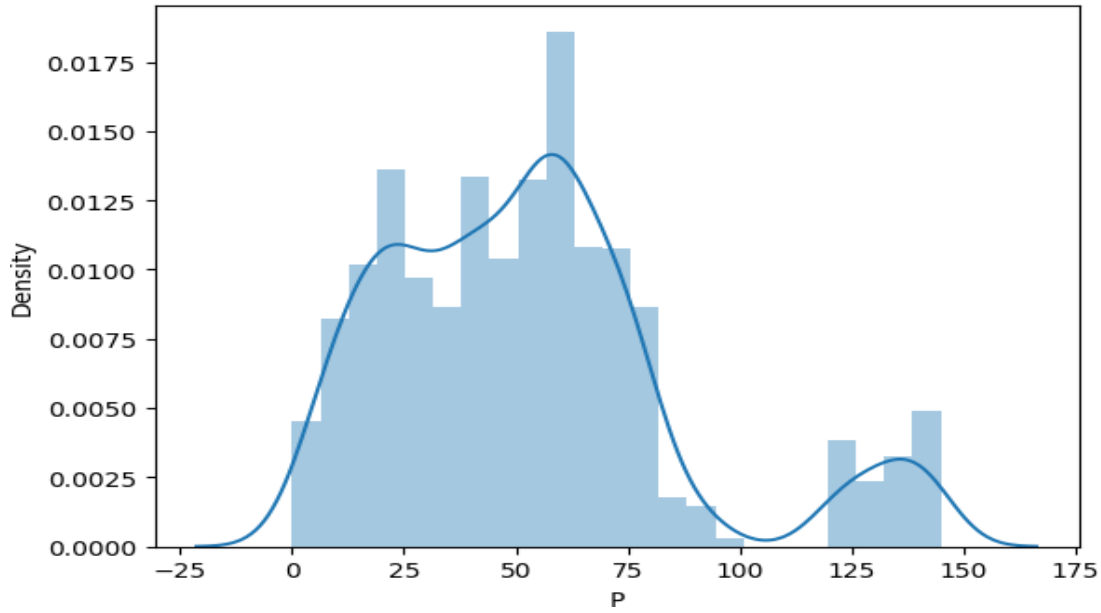



```
sns.distplot(crop2['N']) plt.show()
```

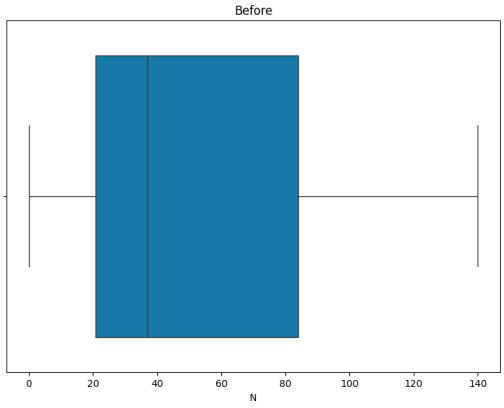


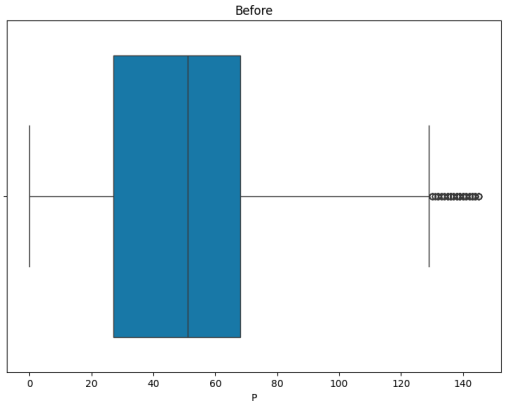
```
sns.distplot(crop2['P'])
```

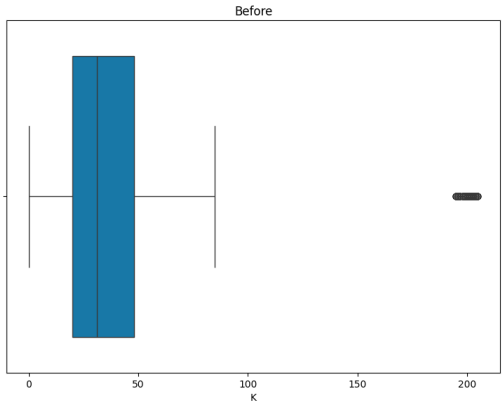
```
<Axes: xlabel='P', ylabel='Density'>
```

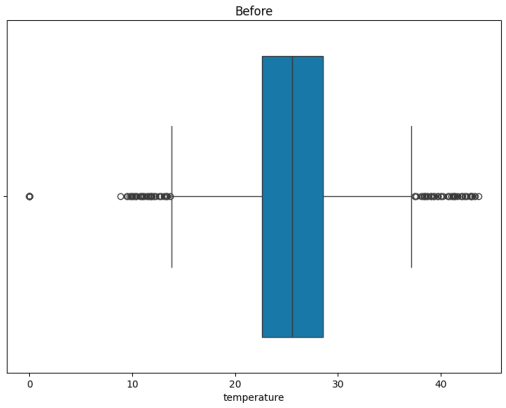


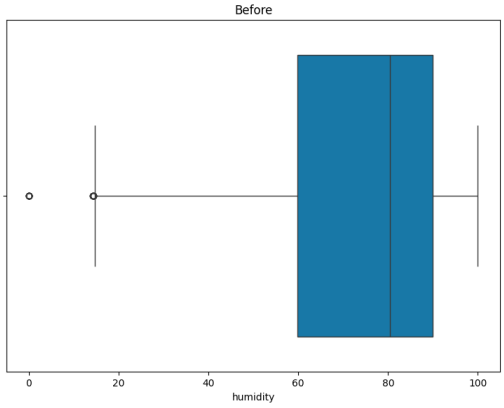

```
f, axes = plt.subplots(4, 2, figsize=(20, 30))

sns.boxplot(x="N", data=crop2, palette="winter", ax=axes[0][0]).set_title("Before") plt.grid()

sns.boxplot(x="P", data=crop2, palette="winter", ax=axes[0][1]).set_title("Before") plt.grid()

sns.boxplot(x="K", data=crop2, palette="winter", ax=axes[1][0]).set_title("Before") plt.grid()

sns.boxplot(x="temperature", data=crop2, palette="winter", ax=axes[1][1]).set_title("Before") plt.grid()

sns.boxplot(x="humidity", data=crop2, palette="winter", ax=axes[2][0]).set_title("Before") plt.grid()

sns.boxplot(x="ph", data=crop2, palette="winter", ax=axes[2][1]).set_title("Before") plt.grid()

sns.boxplot(x="rainfall", data=crop1, palette="winter", ax=axes[3][0]).set_title("Before") plt.grid()
```

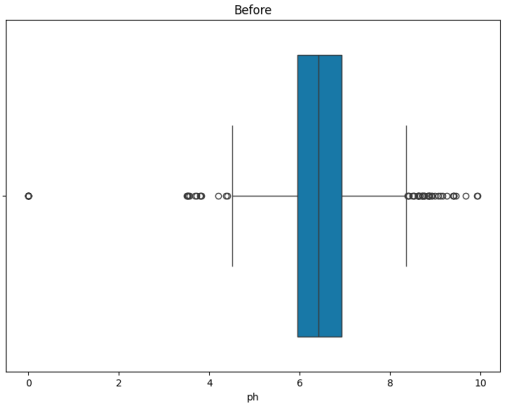

Before
0
20
40
60
80
100
120
140
N

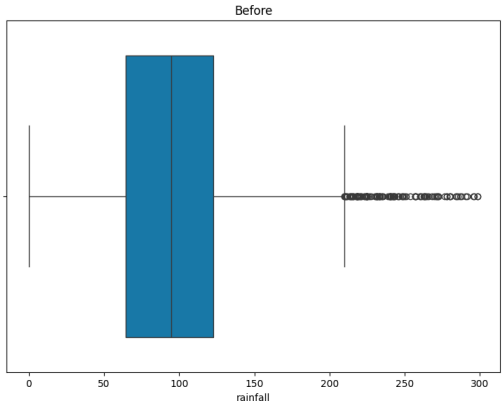

Before
0
20
40
60
80
100
120
140
P

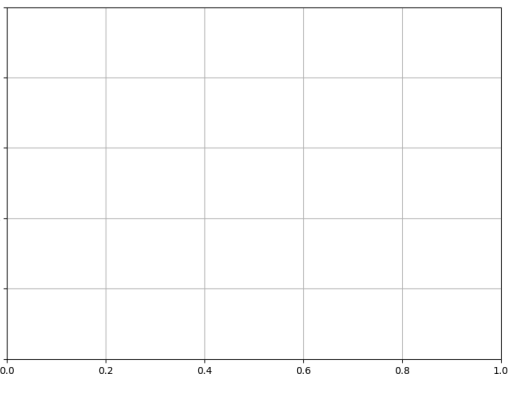

Before
0
50
100
150
200
K


Before
0
10
20
30
40
temperature


Before
0
20
40
60
80
100
humidity

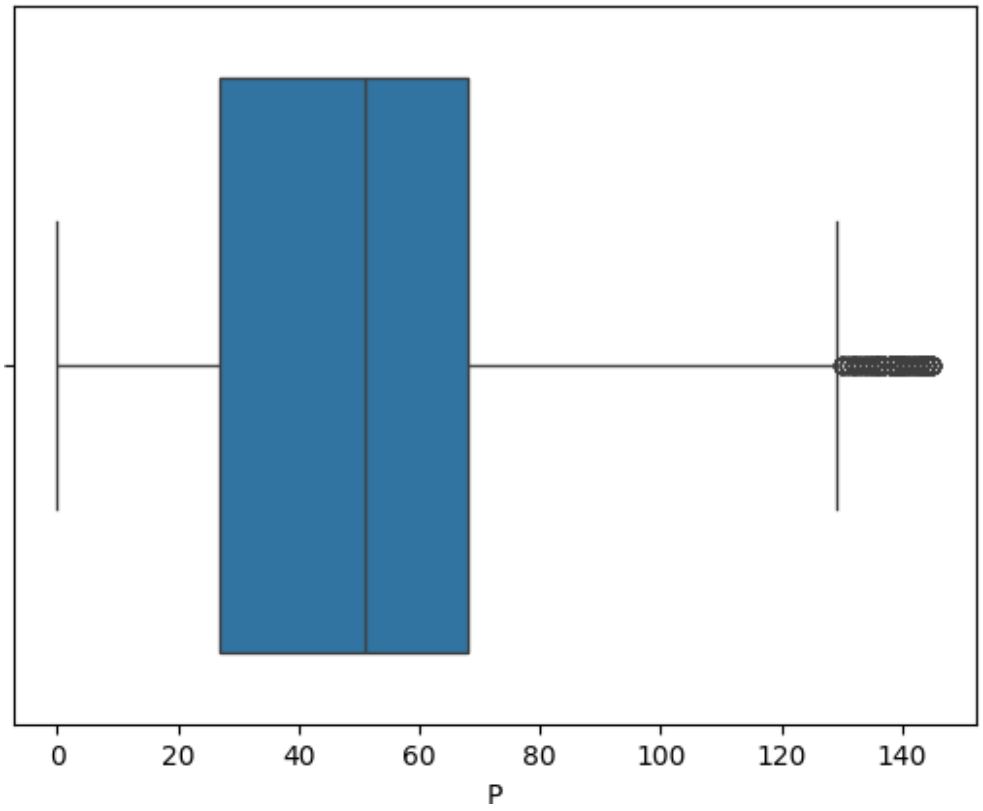

Before
0
2
4
6
8
10
ph


Before
0
50
100
150
200
250
300
rainfall


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

```
sns.boxplot(x=crop2['P'])
```

```
<Axes: xlabel='P'>
```




```
crop3 = crop2.copy() crop3
len(crop3) 2200
```

Comments:

- crop2.copy() creates a deep copy of the DataFrame crop2, ensuring that any changes made to crop3 do not affect crop2.
- crop3 is then displayed, likely to inspect the content or structure of the copied DataFrame.
- len(crop3) calculates the length of crop3, which represents the number of rows in the DataFrame. This can provide information about the size of the dataset.

```
Q1 = crop3['P'].quantile(0.25) Q3 =
crop3['P'].quantile(0.75)
IQR = Q3 - Q1


print('Q1=',Q1,', Q3=', Q3,', IQR=' ,IQR)



upper_limit = Q3 + (1.5 * IQR)
lower_limit = Q1 - (1.5 * IQR)


print("lower_limit=",lower_limit)
print("upper_limit",upper_limit)
```

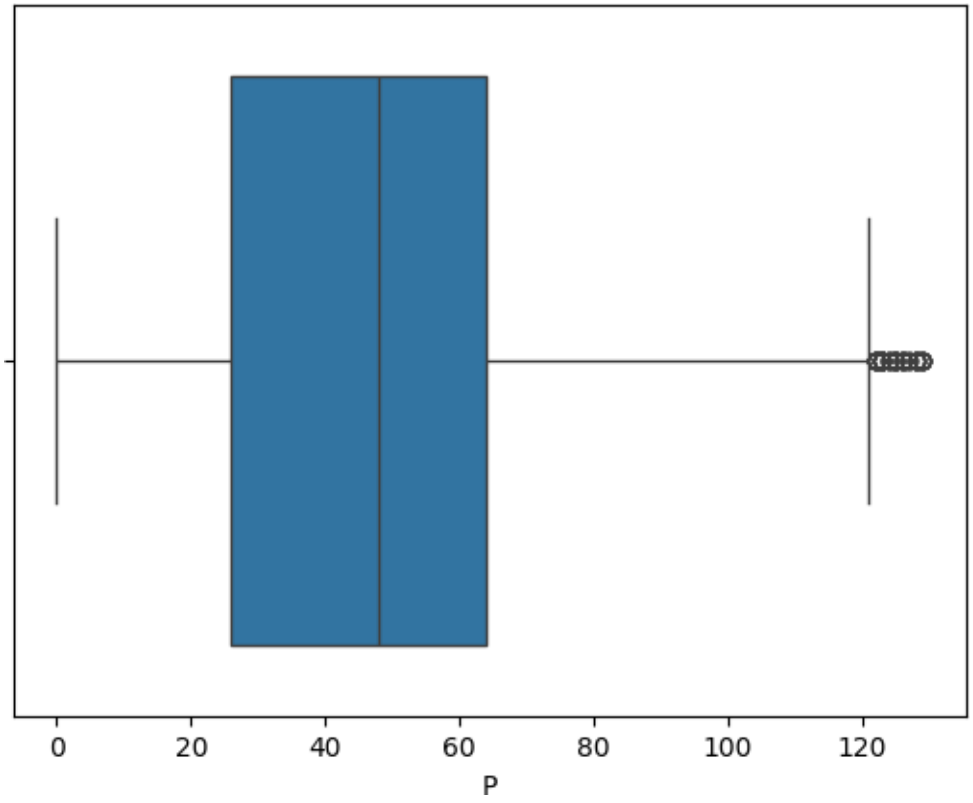
```
crop3.loc[(crop3['P'] > upper_limit) | (crop3['P'] < lower_limit)]


new_df = crop3.loc[(crop3['P'] < upper_limit) & (crop3['P'] > lower_limit)]


print("before removing outliers:",len(crop3)) print("after
removing outliers:",len(new_df)) print("outliers:",len(crop3)-
len(new_df))
```

```
Q1= 27.0 , Q3= 68.0 , IQR= 41.0
lower_limit= -34.5
upper_limit 129.5
before removing outliers: 2200 after
removing outliers: 2069 outliers: 131
```

- Comments:
- Step 1 calculates the first quartile (Q1), third quartile (Q3), and interquartile range (IQR) for the 'P' column, which are essential for identifying outliers.
- Step 2 defines the upper and lower limits for what is considered an outlier based on the calculated quartiles and IQR.
- Outliers are then identified in the 'P' column based on the defined upper and lower limits.
- Trimming the data involves creating a new DataFrame (new_df) that contains only the rows where the 'P' column values are within the defined limits, effectively removing outliers.
- The number of rows before and after removing outliers is displayed, along with the number of outliers removed.

```
sns.boxplot(x=new_df['P'])
```

```
<Axes: xlabel='P'>
```

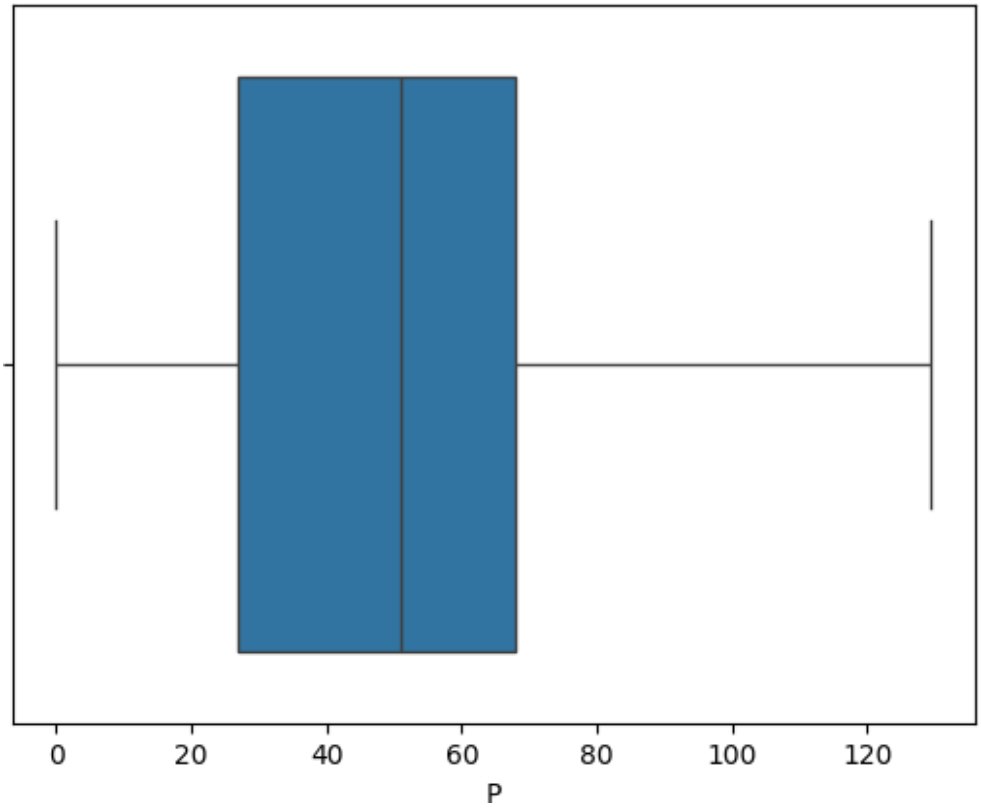


```
new_df = crop3.copy()

new_df.loc[(new_df['P'] > upper_limit), 'P'] = upper_limit

new_df.loc[(new_df['P'] < lower_limit), 'P'] = lower_limit
```

- Comments:
- The first line creates a copy of the DataFrame crop3 and assigns it to a new DataFrame new_df. This ensures that any modifications made to new_df do not affect crop3.
- The second line caps the values in the 'P' column of new_df that exceed the upper limit by setting them to the upper limit.
- The third line caps the values in the 'P' column of new_df that are below the lower limit by setting them to the lower limit.

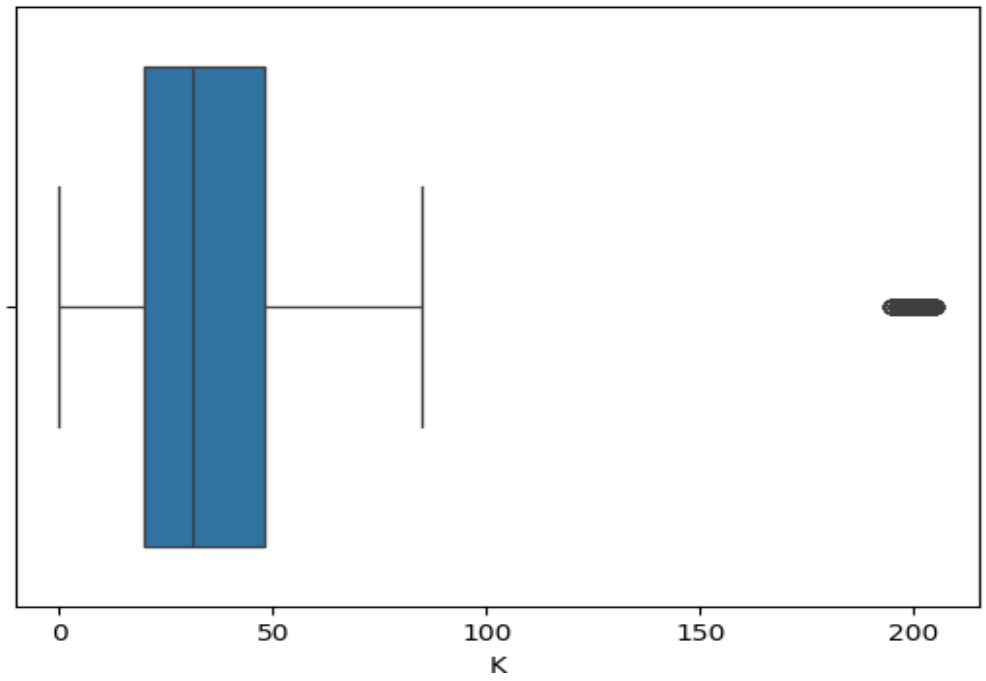
```
sns.boxplot(x=new_df['P'])
```

<Axes: xlabel='P'>




```
sns.boxplot(x=crop2['K'])
```

<Axes: xlabel='K'>




```
Q1 = crop3['K'].quantile(0.25) Q3 =
crop3['K'].quantile(0.75)
IQR = Q3 - Q1


print('Q1=',Q1,', Q3=', Q3,', IQR=' ,IQR)



upper_limit = Q3 + (1.5 * IQR)
lower_limit = Q1 - (1.5 * IQR)


print("lower_limit=",lower_limit)
print("upper_limit",upper_limit)


crop3.loc[(crop3['K'] > upper_limit) | (crop3['K'] < lower_limit)]


new_df = crop3.loc[(crop3['K'] < upper_limit) & (crop3['K'] > lower_limit)]


print("before removing outliers:",len(crop3)) print("after
removing outliers:",len(new_df)) print("outliers:",len(crop3)-
len(new_df))
```

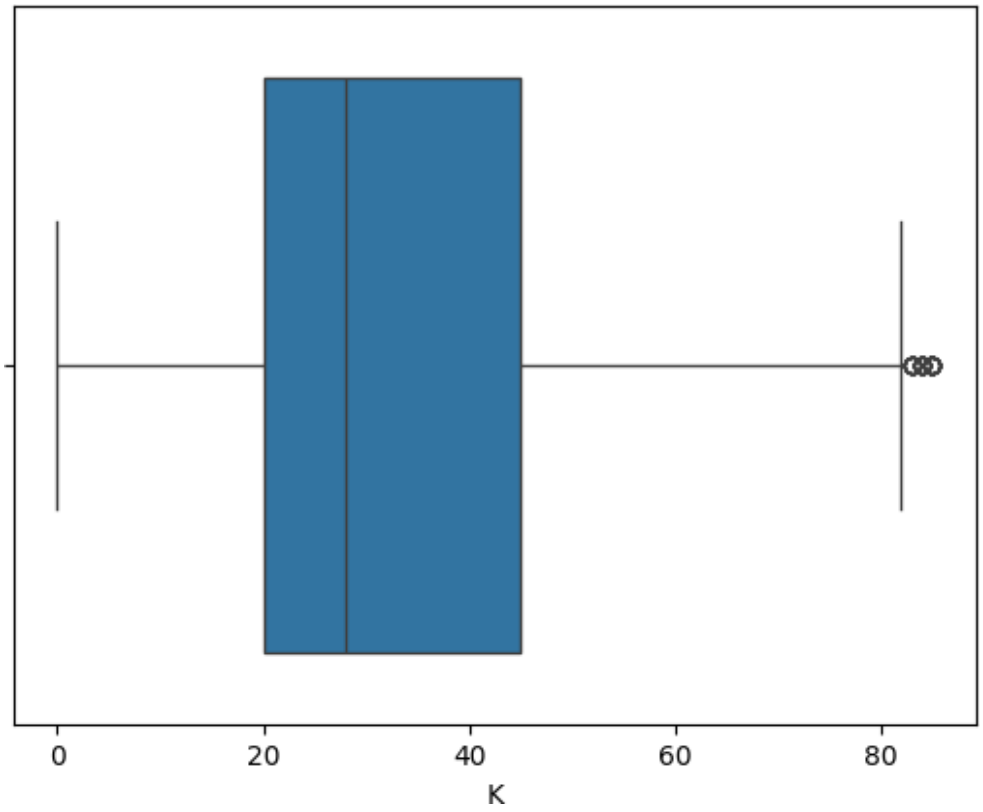
```
Q1= 20.0 , Q3= 48.0 , IQR= 28.0
lower_limit= -22.0
upper_limit 90.0
before removing outliers: 2200 after
removing outliers: 2002 outliers: 198
```

- Comments:
- Step 1 calculates the first quartile (Q1), third quartile (Q3), and interquartile range (IQR) for the 'N' column, which are essential for identifying outliers.

- Step 2 defines the upper and lower limits for what is considered an outlier based on the calculated quartiles and IQR.
- Outliers are then identified in the 'N' column based on the defined upper and lower limits.
- Trimming the data involves creating a new DataFrame (new_df) that contains only the rows where the 'N' column values are within the defined limits, effectively removing outliers.
- The number of rows before and after removing outliers is displayed, along with the number of outliers removed.

```
sns.boxplot(x=new_df['K'])
```

```
<Axes: xlabel='K'>
```




```
new_df = crop3.copy()
```

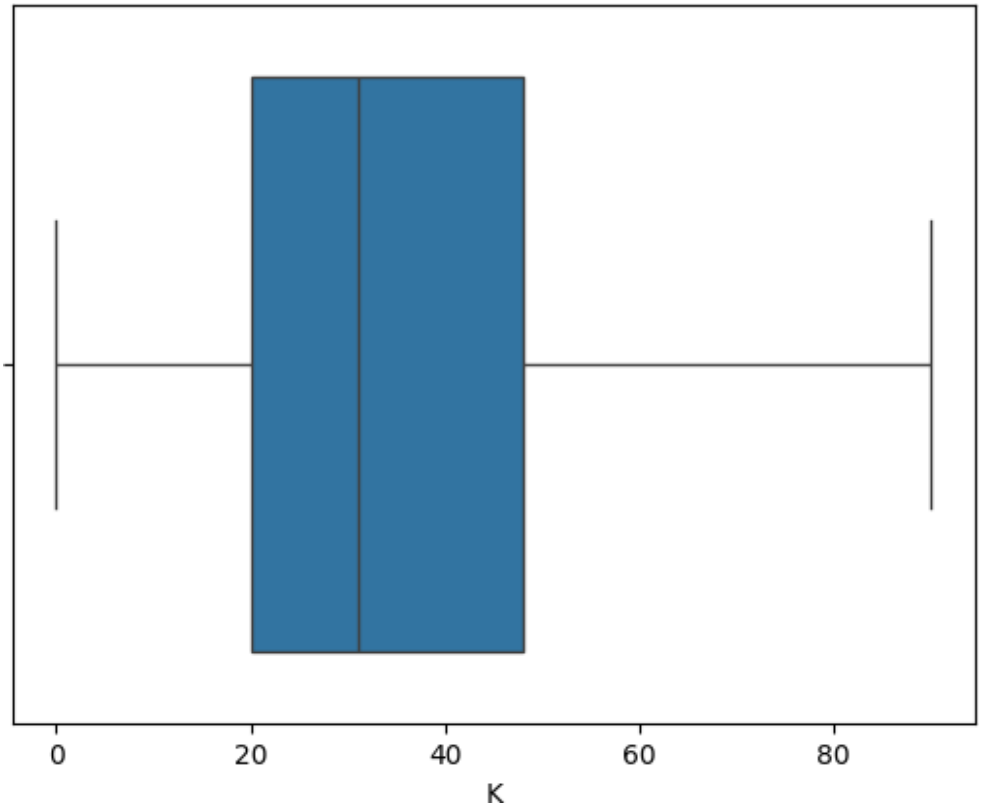
```
new_df.loc[(new_df['K'] > upper_limit), 'K'] = upper_limit
```

```
new_df.loc[(new_df['K'] < lower_limit), 'K'] = lower_limit
```

- Comments:
- The first line creates a copy of the DataFrame crop3 and assigns it to a new DataFrame new_df. This ensures that any modifications made to new_df do not affect crop3.
- The second line caps the values in the 'K' column of new_df that exceed the upper limit by setting them to the upper limit.
- The third line caps the values in the 'K' column of new_df that are below the lower limit by setting them to the lower limit.

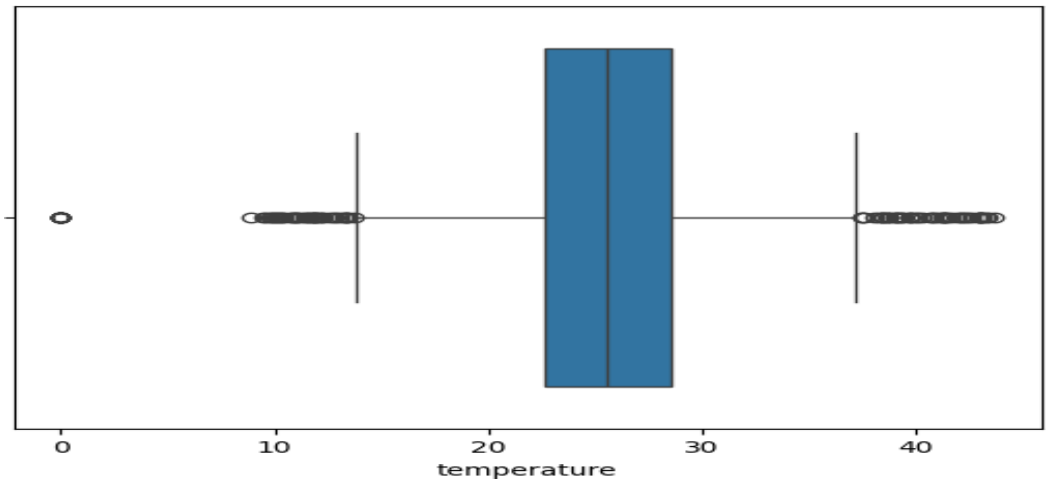
```
sns.boxplot(x=new_df['K'])
```

<Axes: xlabel='K'>




```
sns.boxplot(x=crop2['temperature'])
```

<Axes: xlabel='temperature'>




```
Q1 = crop3['temperature'].quantile(0.25) Q3 =
crop3['temperature'].quantile(0.75)
IQR = Q3 - Q1
```

```
print('Q1=',Q1,', Q3=', Q3,', IQR=' ,IQR)
```

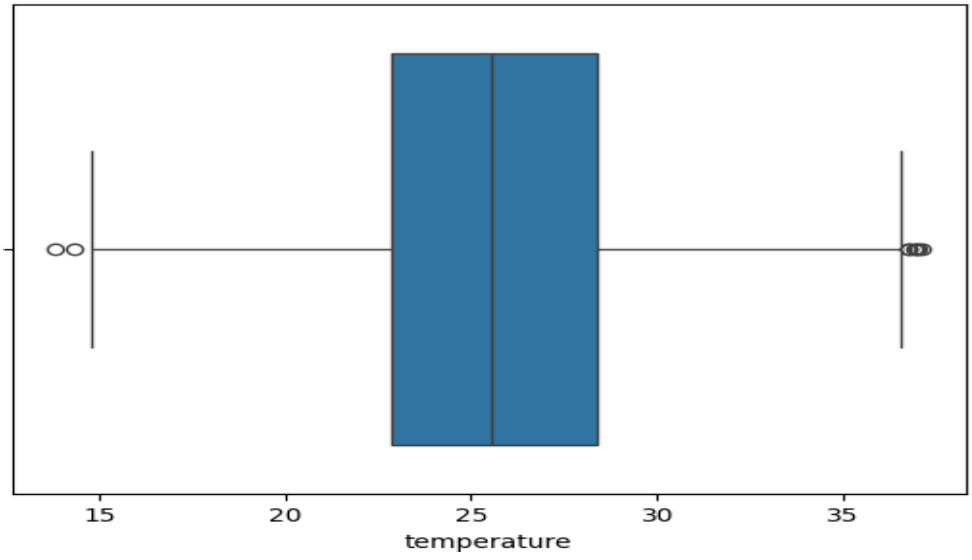
```
upper_limit = Q3 + (1.5 * IQR)
lower_limit = Q1 - (1.5 * IQR)


print("lower_limit=",lower_limit)
print("upper_limit",upper_limit)


crop3.loc[(crop3['temperature'] > upper_limit) | (crop3['temperature'] < lower_limit)]


new_df = crop3.loc[(crop3['temperature'] < upper_limit) & (crop3['temperature'] > lower_limit)]


print("before removing outliers:",len(crop3)) print("after
removing outliers:",len(new_df)) print("outliers:",len(crop3)-
len(new_df))
```

```
Q1= 22.629466635 , Q3= 28.546521374999998 , IQR= 5.9170547399999975
lower_limit= 13.753884525000004
upper_limit 37.422103484999994 before
removing outliers: 2200 after removing
outliers: 2093 outliers: 107
```

- Comments:
- Step 1 calculates the first quartile (Q1), third quartile (Q3), and interquartile range (IQR) for the 'temperature' column, which are essential for identifying outliers.
- Step 2 defines the upper and lower limits for what is considered an outlier based on the calculated quartiles and IQR.
- Outliers are then identified in the 'temperature' column based on the defined upper and lower limits.
- Trimming the data involves creating a new DataFrame (new_df) that contains only the rows where the 'temperature' column values are within the defined limits, effectively removing outliers.
- The number of rows before and after removing outliers is displayed, along with the number of outliers removed.

```
sns.boxplot(x=new_df['temperature'])
```

```
<Axes: xlabel='temperature'>
```



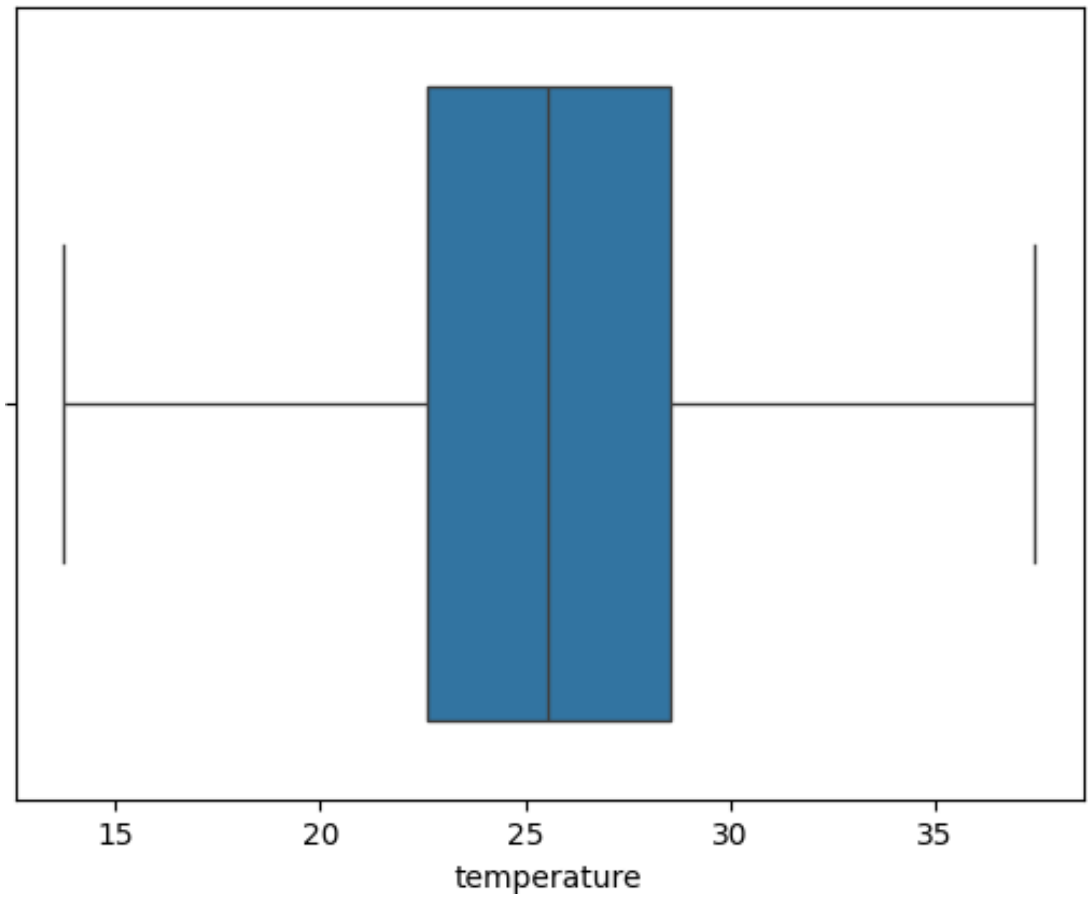
```
new_df = crop3.copy()

new_df.loc[(new_df['temperature'] > upper_limit), 'temperature'] = upper_limit

new_df.loc[(new_df['temperature'] < lower_limit), 'temperature'] = lower_limit
```

Comments:

- The first line creates a copy of the DataFrame crop3 and assigns it to a new DataFrame new_df. This ensures that any modifications made to new_df do not affect crop3.
- The second line caps the values in the 'P' column of new_df that exceed the upper limit by setting them to the upper limit.
- The third line caps the values in the 'P' column of new_df that are below the lower limit by setting them to the lower limit.

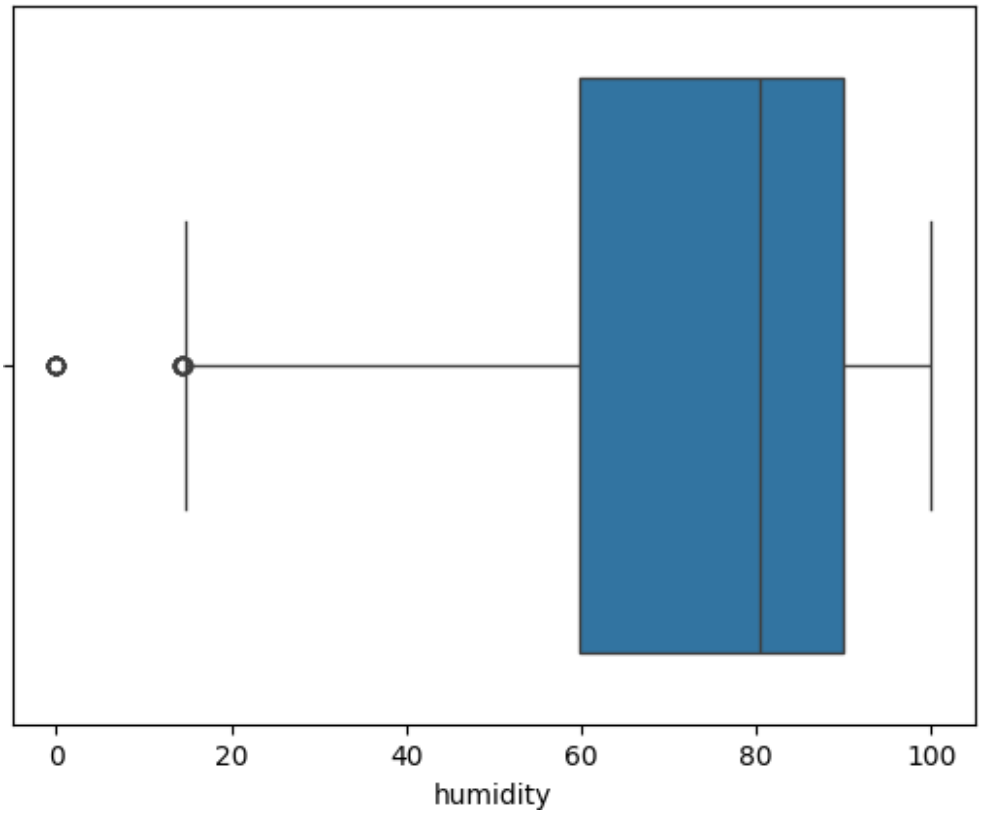
```
sns.boxplot(x=new_df['temperature'])
```

```
<Axes: xlabel='temperature'>
```



```
sns.boxplot(x=crop3['humidity'])
```

```
<Axes: xlabel='humidity'>
```

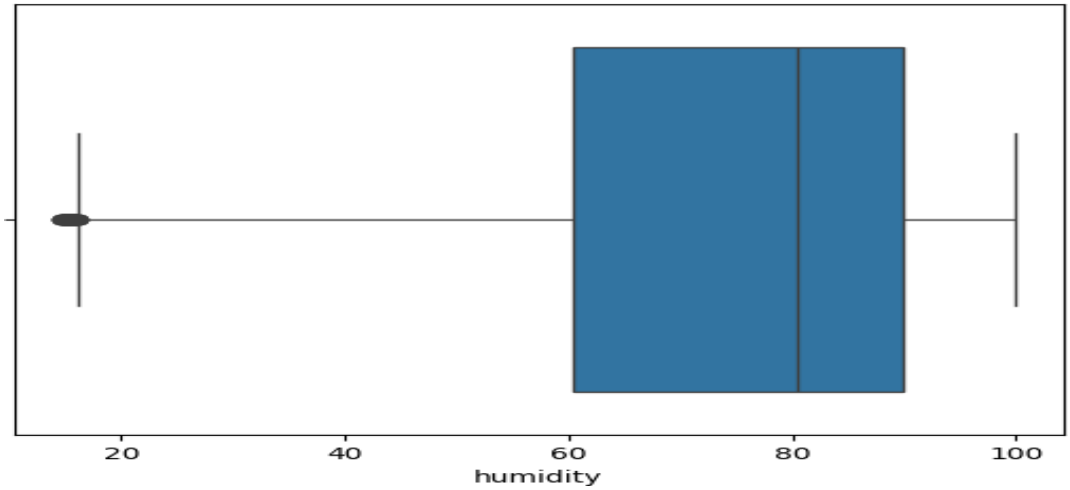



```
Q1 = crop3['humidity'].quantile(0.25) Q3 =
crop3['humidity'].quantile(0.75)
IQR = Q3 - Q1

print('Q1=',Q1,', Q3=', Q3,', IQR=' ,IQR)

upper_limit = Q3 + (1.5 * IQR)
lower_limit = Q1 - (1.5 * IQR)


print("lower_limit=",lower_limit)
print("upper_limit",upper_limit)

crop3.loc[(crop3['humidity'] > upper_limit) | (crop3['humidity'] < lower_limit)]


new_df = crop3.loc[(crop3['humidity'] < upper_limit) & (crop3['humidity'] > lower_limit)]


print("before removing outliers:",len(crop3)) print("after
removing outliers:",len(new_df)) print("outliers:",len(crop3)-
len(new_df))

Q1= 59.79183075 , Q3= 89.892282115 , IQR= 30.100451364999998
lower_limit= 14.641153702500006
upper_limit 135.04295916249998 before
removing outliers: 2200 after removing
outliers: 2175 outliers: 25
```

- Comments:
- Step 1 calculates the first quartile (Q1), third quartile (Q3), and interquartile range (IQR) for the 'humidity' column, which are essential for identifying outliers.
- Step 2 defines the upper and lower limits for what is considered an outlier based on the calculated quartiles and IQR.
- Outliers are then identified in the 'humidity' column based on the defined upper and lower limits.
- Trimming the data involves creating a new DataFrame (new_df) that contains only the rows where the 'humidity' column values are within the defined limits, effectively removing outliers.
- The number of rows before and after removing outliers is displayed, along with the number of outliers removed.

```
sns.boxplot(x=new_df['humidity'])
```

```
<Axes: xlabel='humidity'>
```




```
new_df = crop3.copy()
```

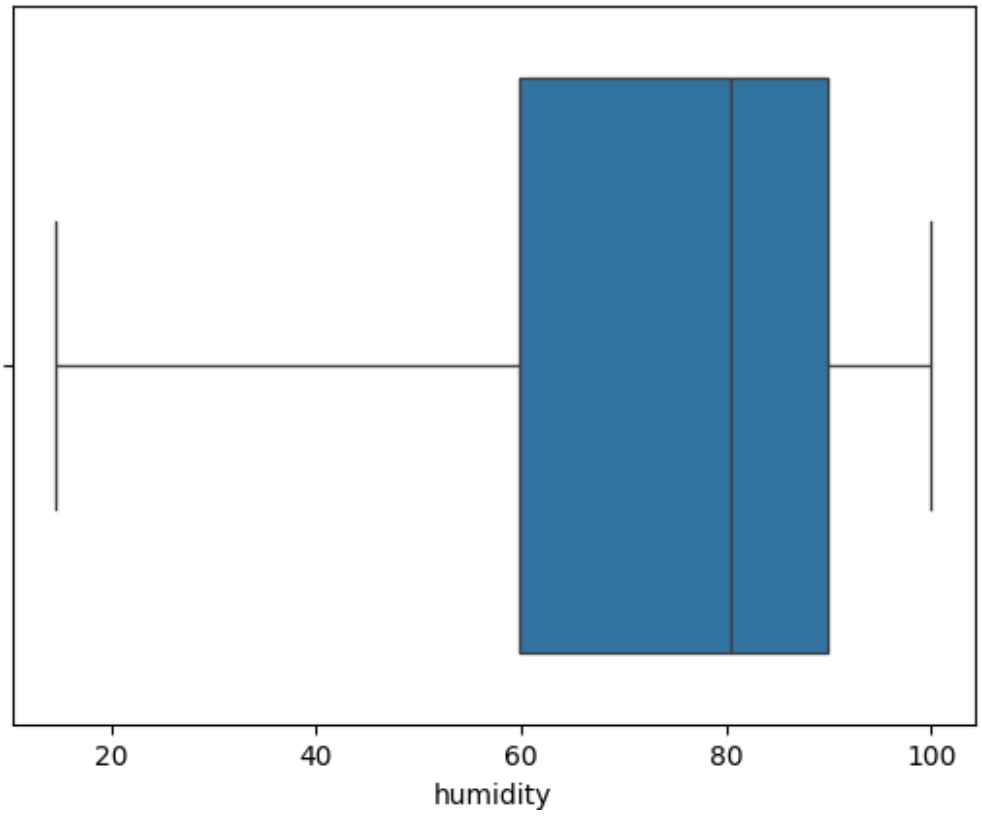
```
new_df.loc[(new_df['humidity'] > upper_limit), 'humidity'] = upper_limit
```

```
new_df.loc[(new_df['humidity'] < lower_limit), 'humidity'] = lower_limit
```

- Comments:
- The first line creates a copy of the DataFrame crop3 and assigns it to a new DataFrame new_df. This ensures that any modifications made to new_df do not affect crop3.
- The second line caps the values in the 'humidity' column of new_df that exceed the upper limit by setting them to the upper limit.
- The third line caps the values in the 'humidity' column of new_df that are below the lower limit by setting them to the lower limit.

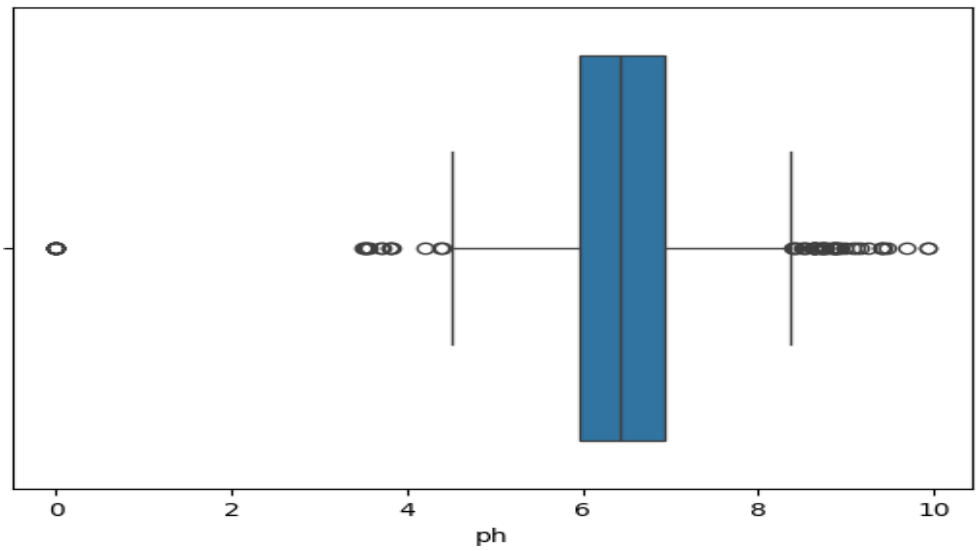
```
sns.boxplot(x=new_df['humidity'])
```

```
<Axes: xlabel='humidity'>
```




```
sns.boxplot(x=crop3['ph'])
```

```
<Axes: xlabel='ph'>
```



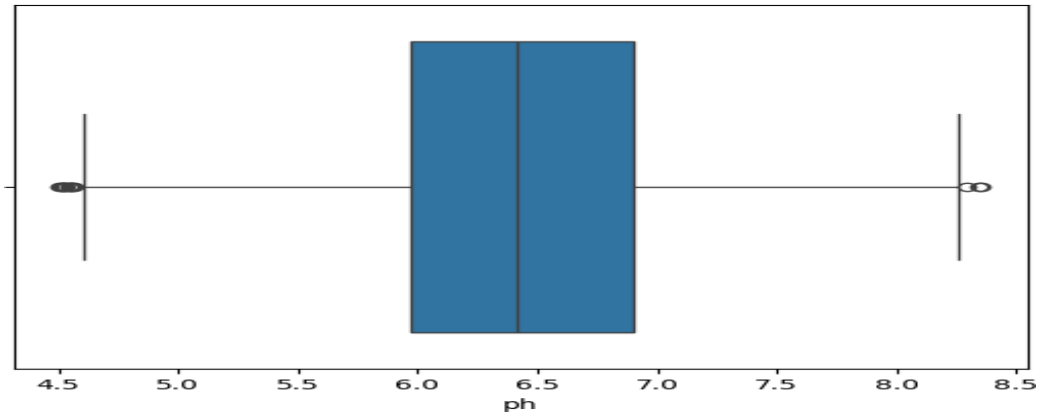
```
Q1 = crop3['ph'].quantile(0.25) Q3 =
crop3['ph'].quantile(0.75)
IQR = Q3 - Q1


print('Q1=',Q1,', Q3=', Q3,', IQR=' ,IQR)

upper_limit = Q3 + (1.5 * IQR)
lower_limit = Q1 - (1.5 * IQR)


print("lower_limit=",lower_limit)
print("upper_limit",upper_limit)


crop3.loc[(crop3['ph'] > upper_limit) | (crop3['ph'] < lower_limit)]


new_df = crop3.loc[(crop3['ph'] < upper_limit) & (crop3['ph'] > lower_limit)]


print("before removing outliers:",len(crop3)) print("after
removing outliers:",len(new_df)) print("outliers:",len(crop3)-
len(new_df))
```

```
Q1= 5.955956037 , Q3= 6.92237275125 , IQR= 0.9664167142500002
lower_limit= 4.506330965625
upper_limit 8.371997822625001 before
removing outliers: 2200 after removing
outliers: 2133 outliers: 67
```

- Comments:
- Step 1 calculates the first quartile (Q1), third quartile (Q3), and interquartile range (IQR) for the 'ph' column, which are essential for identifying outliers.
- Step 2 defines the upper and lower limits for what is considered an outlier based on the calculated quartiles and IQR.
- Outliers are then identified in the 'ph' column based on the defined upper and lower limits.
- Trimming the data involves creating a new DataFrame (new_df) that contains only the rows where the 'ph' column values are within the defined limits, effectively removing outliers.
- The number of rows before and after removing outliers is displayed, along with the number of outliers removed.

```
sns.boxplot(x=new_df['ph'])
```

```
<Axes: xlabel='ph'>
```

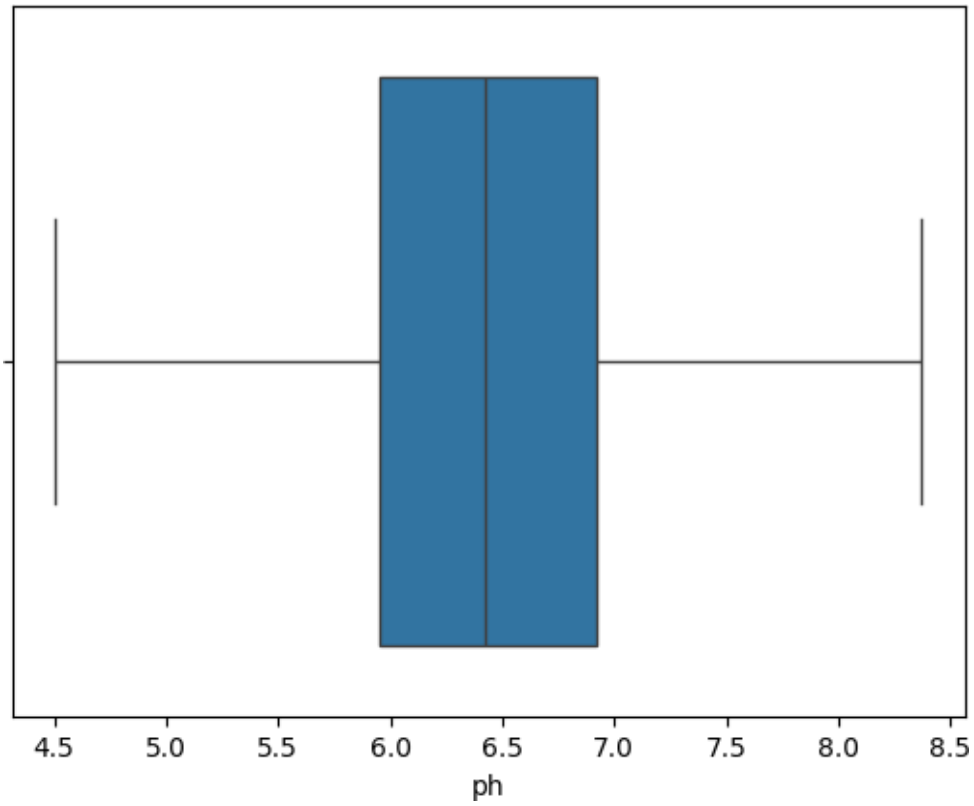


```
new_df = crop3.copy()


new_df.loc[(new_df['ph'] > upper_limit), 'ph'] = upper_limit


new_df.loc[(new_df['ph'] < lower_limit), 'ph'] = lower_limit
```

- Comments:
- The first line creates a copy of the DataFrame crop3 and assigns it to a new DataFrame new_df. This ensures that any modifications made to new_df do not affect crop3.
- The second line caps the values in the 'ph' column of new_df that exceed the upper limit by setting them to the upper limit.
- The third line caps the values in the 'ph' column of new_df that are below the lower limit by setting them to the lower limit.

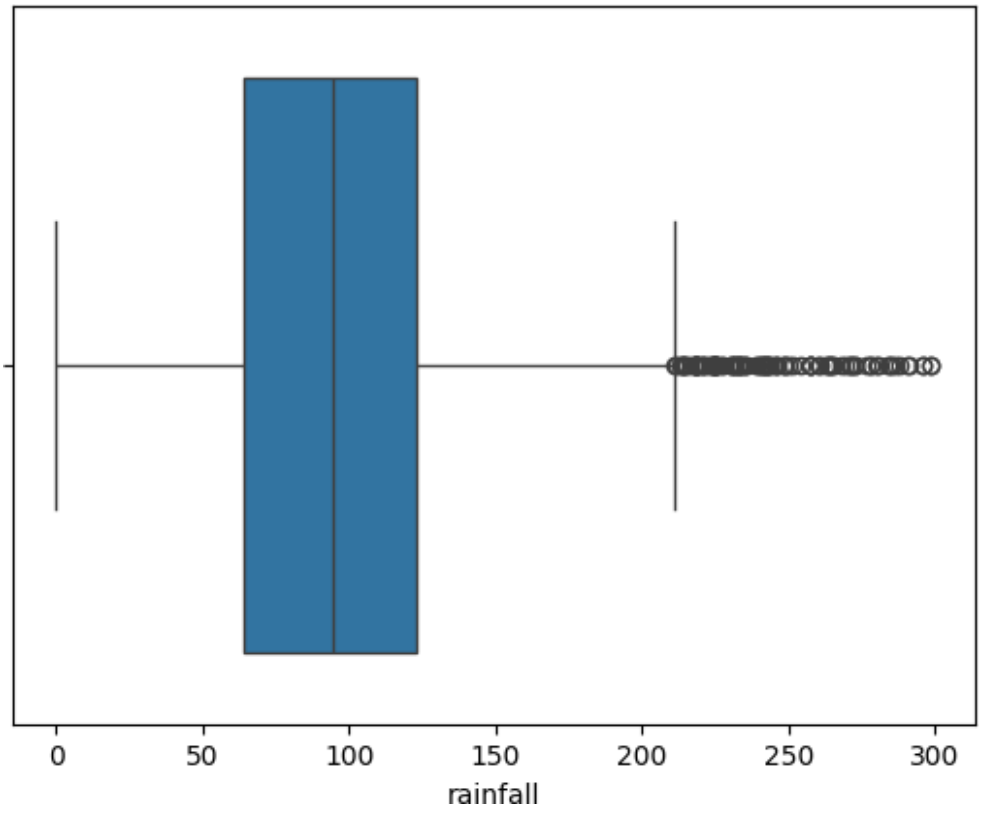
```
sns.boxplot(x=new_df['ph'])
```

<Axes: xlabel='ph'>



```
sns.boxplot(x=crop3['rainfall'])
```

```
<Axes: xlabel='rainfall'>
```



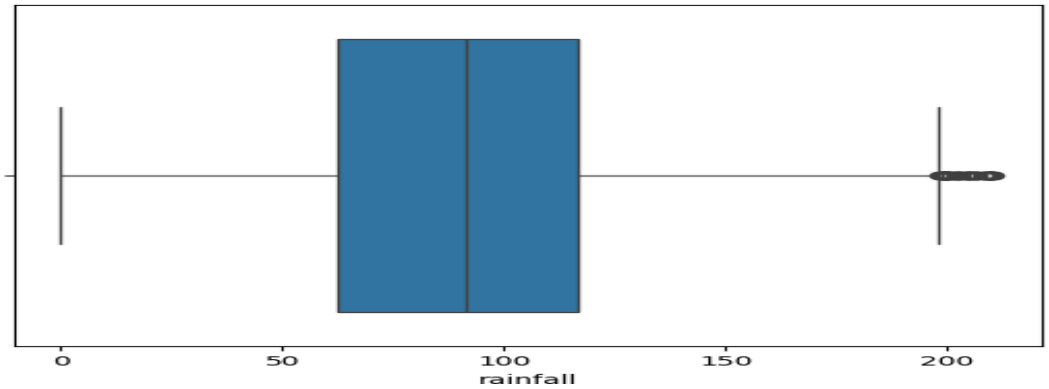

```
Q1 = crop3['rainfall'].quantile(0.25) Q3 =
crop3['rainfall'].quantile(0.75)
IQR = Q3 - Q1


print('Q1=',Q1,', Q3=', Q3,', IQR=' ,IQR)

upper_limit = Q3 + (1.5 * IQR)
lower_limit = Q1 - (1.5 * IQR)


print("lower_limit=",lower_limit)
print("upper_limit",upper_limit)


crop3.loc[(crop3['rainfall'] > upper_limit) | (crop3['rainfall'] < lower_limit)]


new_df = crop3.loc[(crop3['rainfall'] < upper_limit) & (crop3['rainfall'] > lower_limit)]


print("before removing outliers:",len(crop3)) print("after
removing outliers:",len(new_df)) print("outliers:",len(crop3)-
len(new_df))

Q1= 63.86682675 , Q3= 122.747122125 , IQR= 58.880295375
lower_limit= -24.453616312500003
upper_limit 211.0675651875 before
removing outliers: 2200
```

after removing outliers: 2096 outliers:
104

- Comments:
- Step 1 calculates the first quartile (Q1), third quartile (Q3), and interquartile range (IQR) for the 'rainfall' column, which are essential for identifying outliers.
- Step 2 defines the upper and lower limits for what is considered an outlier based on the calculated quartiles and IQR.
- Outliers are then identified in the 'rainfall' column based on the defined upper and lower limits.
- Trimming the data involves creating a new DataFrame (new_df) that contains only the rows where the 'rainfall' column values are within the defined limits, effectively removing outliers.
- The number of rows before and after removing outliers is displayed, along with the number of outliers removed.

```
sns.boxplot(x=new_df['rainfall'])
```

<Axes: xlabel='rainfall'>

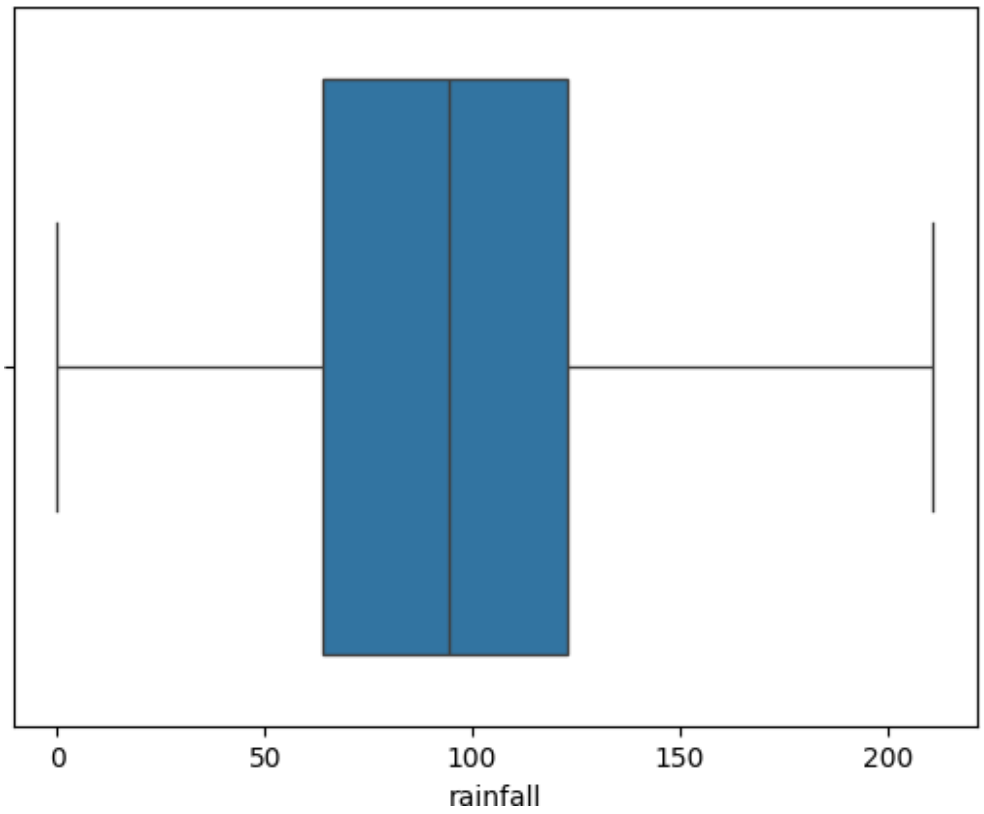



```
new_df = crop3.copy()


new_df.loc[(new_df['rainfall'] > upper_limit), 'rainfall'] = upper_limit


new_df.loc[(new_df['rainfall'] < lower_limit), 'rainfall'] = lower_limit
```

- Comments:
- The first line creates a copy of the DataFrame crop3 and assigns it to a new DataFrame new_df. This ensures that any modifications made to new_df do not affect crop3.
- The second line caps the values in the 'rainfall' column of new_df that exceed the upper limit by setting them to the upper limit.
- The third line caps the values in the 'rainfall' column of new_df that are below the lower limit by setting them to the lower limit.

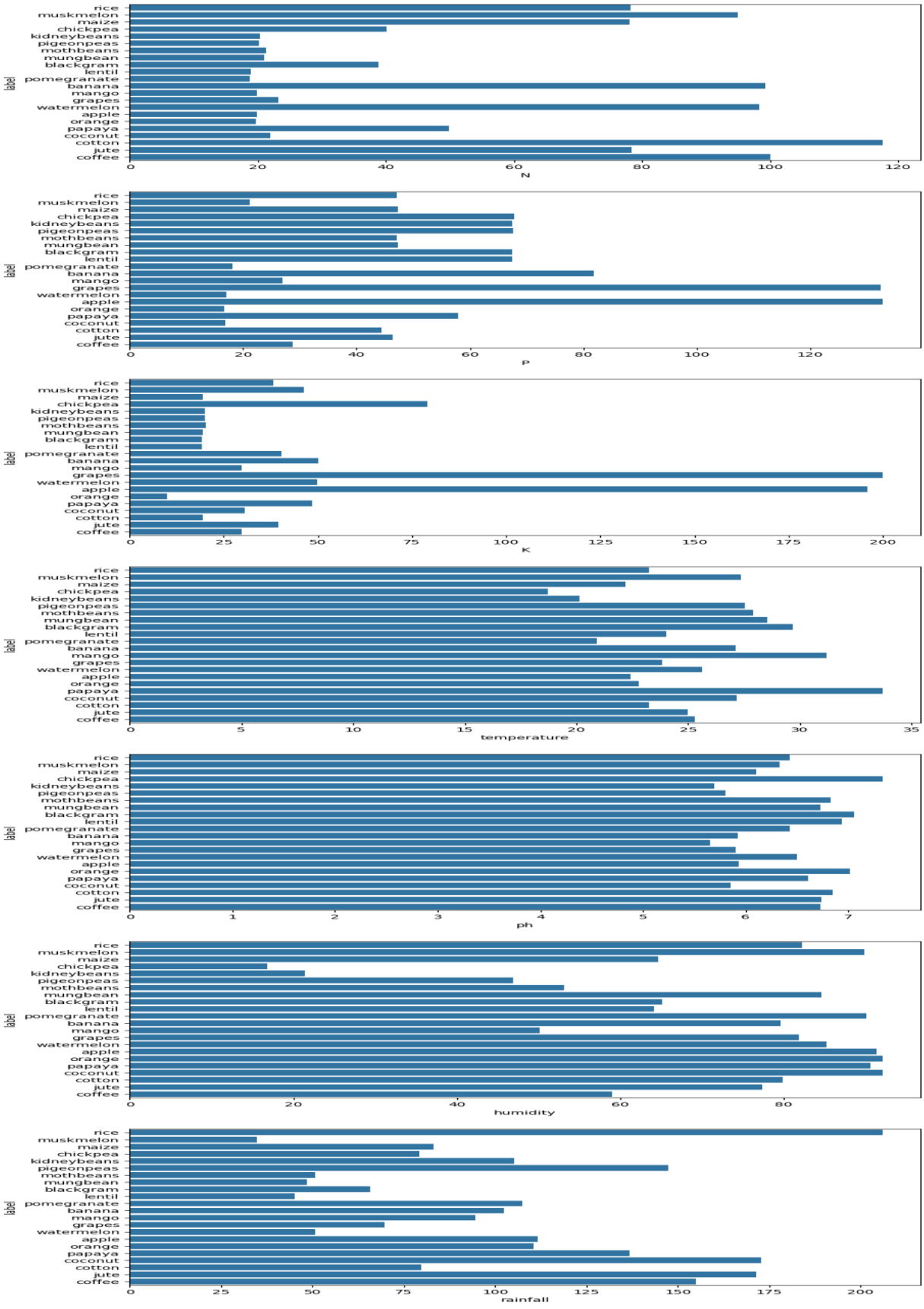
```
sns.boxplot(x=new_df['rainfall'])
```

<Axes: xlabel='rainfall'>




```
f, axes = plt.subplots(7, 1, figsize=(10, 40))

sns.barplot(x=new_df.N, y=crop2.label, ax=axes.flat[0], errorbar=None)

sns.barplot(x=new_df.P, y=crop2.label, ax=axes.flat[1], errorbar=None)

sns.barplot(x=new_df.K, y=crop2.label, ax=axes.flat[2], errorbar=None)


sns.barplot(x=new_df.temperature, y=crop2.label, ax=axes.flat[3], errorbar=None)

sns.barplot(x=new_df.ph, y=crop2.label, ax=axes.flat[4], errorbar=None)

sns.barplot(x=new_df.humidity, y=crop2.label, ax=axes.flat[5], errorbar=None)

sns.barplot(x=new_df.rainfall, y=crop2.label, ax=axes.flat[6], errorbar=None)
```

```
<Axes: xlabel='rainfall', ylabel='label'>
```


label
rice
muskmelon
maize
chickpea
kidneybeans
pigeonpeas
mothbeans
mungbean
blackgram
lentil
pomegranate
banana
mango
grapes
watermelon
apple
orange
papaya
coconut
cotton
jute
coffee
0
20
40
60
80
100
120
N
P
K
temperature
ph
humidity
rainfall

this code prepares your data by selecting relevant features and the target variable for use in a machine learning model. The features are the input variables used to make predictions, while the target variable is the output variable you want to predict.

```
features = new_df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']] target = new_df['label']

labels = new_df['label']

new_df2 = features.fillna(features.mean())

print("Features:",features.head(),sep="\n") print("\n")
print("Target:",target.head(),sep="\n") print("\n")
print("Shape of features:",features.shape) print("Shape of
target:",target.shape)
```

Features:

	N	P	K	temperature	humidity	ph	rainfall
0	90.0	42.0	43.0	20.879744	82.002744	6.502985	202.935536
1	85.0	58.0	41.0	21.770462	80.319644	7.038096	211.067565
2	60.0	55.0	44.0	23.004459	82.320763	7.840207	211.067565
3	74.0	35.0	40.0	26.491096	80.158363	6.980401	211.067565
4	78.0	42.0	42.0	20.130175	81.604873	7.628473	211.067565

Target:

```
0 rice
1 rice
2 rice
3 rice
4 rice
Name: label, dtype: object
```

Shape of features: (2200, 7) Shape of
target: (2200,)

By initializing these lists, you're setting up a structure to collect and organize the results of model evaluation, likely for comparison or further analysis. As you evaluate different models, you can append their accuracy scores to the acc list and their names to the model list, allowing you to track and analyze the performance of each model.

```
acc = [] model =
[]

from sklearn.model_selection import train_test_split

Xtrain, Xtest, Ytrain, Ytest = train_test_split(features,target,test_size = 0.25,random_state =51) print("Shape of Xtrain:",Xtrain.shape)
print("Shape of Ytrain:",Ytrain.shape) print("Shape of
Xtest:",Xtest.shape) print("Shape of
Ytest:",Ytest.shape)
```

Shape of Xtrain: (1650, 7) Shape of
Ytrain: (1650,) Shape of Xtest:
(550, 7) Shape of Ytest: (550,)

```
from sklearn.preprocessing import StandardScaler scaler =

StandardScaler()

scaler.fit(Xtrain) means =
scaler.mean_

print("Means of each feature after scaling:\n", means)
```

```
scaling_factors = scaler.scale_

print("Scaling factors (standard deviation) of each feature after scaling:\n", scaling_factors) Means of each feature after scaling:
 [ 50.2430303          52.45939394  46.86909091  25.38206679  70.96831023
    6.435959       100.40170904]
Scaling factors (standard deviation) of each feature after scaling:
 [36.87189572 32.75441936 49.47330914  5.5393124  22.90832563  0.92998743
 52.35222465]
Xtrain_df = pd.DataFrame(Xtrain)

Xtrain_df.describe().round(2)

Xtrain_sc = scaler.transform(Xtrain) Xtest_sc =
scaler.transform(Xtest)
print("Xtrain_sc:\n",Xtrain_sc) print("\n")
print("Xtest_sc:\n",Xtest_sc)
```

```
Xtrain_sc:
 [[-1.36263757 -1.02152304 -0.28033481 ...  1.05942573 -6.92047955
    1.99263638]
 [-0.71173531 -0.96046257 -0.09841854 ...  0.70977892  0.60544407
    0.17270105]
 [-0.68461439  0.19968622 -0.56331568 ...  0.55519597  0.20502513
   -1.07827074]
 ...
 [-0.03371213  0.07756529  0.08349773 ...  1.00126441  0.25873444
    0.66411205]
 [-0.87446088  0.07756529 -0.48246401 ...  -0.6398653  0.59114849
    0.69578174]
 [ 1.78338999 -0.89940211 -0.44203817 ... -0.81622567  0.46641619
    0.74429246]]


Xtest_sc:
 [[-0.92870273  0.84082107 -0.54310276 ... -0.62399234  0.97137004
    0.41283755]
 [ 0.4002227          0.32180714  0.06328481 ...  0.88864388  0.09516294
   -0.40360641]
 [ 1.05112495 -1.32682535  0.00264605 ...  0.86441874  0.24992141
   -1.3686037 ]
 ...
 [-0.60325161  0.44392807 -0.52288984 ... -0.31593097 -0.19888954
   -1.02723441]
 [-0.84733995 -0.80781142 -0.36118649 ...  1.21505283 -0.82195704
    1.2639306 ]
 [ 1.07824588 -0.0140254  -0.44203817 ... -0.07003799  0.41573416
    0.05389861]]
```

```
Xtrain_sc = pd.DataFrame(Xtrain_sc, columns=['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']) Xtest_sc = pd.DataFrame(Xtest_sc,
columns=['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall'])
```

- Xtrain_sc = pd.DataFrame(Xtrain_sc, columns=['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']): This line converts the scaled training data Xtrain_sc (which was previously a NumPy array) into a Pandas DataFrame. It specifies column names for the DataFrame using the columns parameter. Each column name corresponds to a feature in the dataset.
- Converting Scaled Testing Data to DataFrame:

- Xtest_sc = pd.DataFrame(Xtest_sc, columns=['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']): Similarly, this line converts the scaled testing data Xtest_sc (also a NumPy array) into a Pandas DataFrame. It assigns the same column names as in the training data DataFrame.

- Comments:
- Converting the scaled data back into Pandas DataFrames allows for easier manipulation and analysis of the data, leveraging Pandas' functionality. Assigning column names ensures that the DataFrame retains the same structure as the original dataset, facilitating further processing or analysis downstream.

```
Xtrain_sc.head()

Xtrain_sc.describe().round(2)

from sklearn.preprocessing import MinMaxScaler
mmc= MinMaxScaler() mmc.fit(Xtrain)
```

```
MinMaxScaler()
```

```
Xtrain_mmc = mmc.transform(Xtrain) Xtest_mmc
= mmc.transform(Xtest)
print("Xtrain_mmc:",Xtrain_mmc) print("\n")
print("Xtest_mmc:",Xtest_mmc)
```

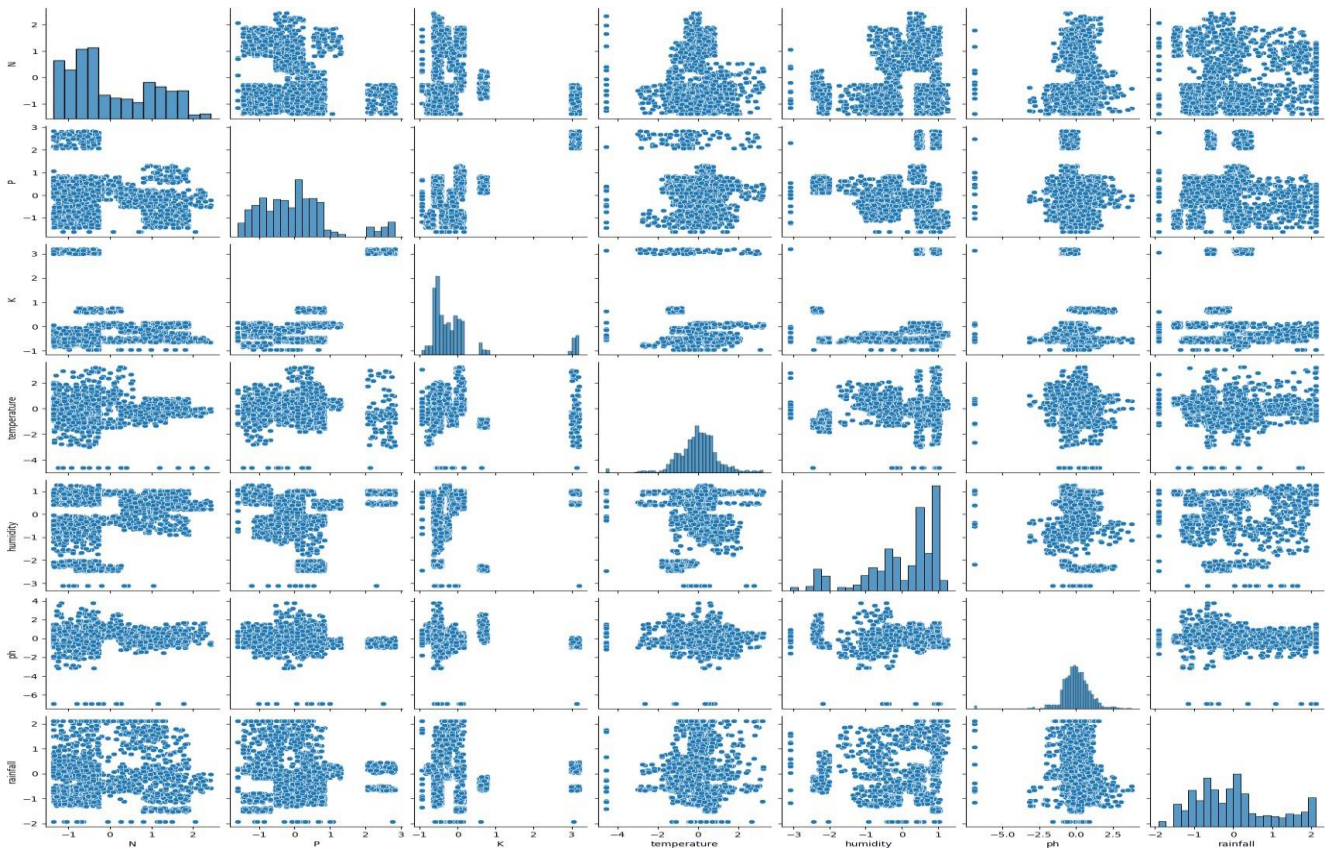
```
Xtrain_mmc: [[0.         0.13103448 0.16097561 ... 0.95255244 0.         0.96992949]
 [0.17142857 0.14482759 0.20487805 ... 0.87243969 0.70447413 0.51852113]
 [0.17857143 0.40689655 0.09268293 ... 0.8370209  0.66699238 0.20823587]
 ...
 [0.35       0.37931034 0.24878049 ... 0.93922624 0.67201991 0.64040845]
 [0.12857143 0.37931034 0.11219512 ... 0.56320275 0.70313597 0.64826365]
 [0.82857143 0.15862069 0.12195122 ... 0.52279422 0.69146024 0.66029603]]


Xtest_mmc: [[0.11428571 0.55172414 0.09756098 ... 0.56683964 0.73872712 0.57808348]
 [0.46428571 0.43448276 0.24390244 ... 0.91342208 0.65670858 0.37557649]
 [0.63571429 0.06206897 0.22926829 ... 0.9078715  0.67119495 0.13622302]
 ...
 [0.2        0.46206897 0.10243902 ... 0.63742413 0.62918341 0.22089468]
 [0.13571429 0.17931034 0.14146341 ... 0.98821046 0.57086034 0.78918468]
 [0.64285714 0.35862069 0.12195122 ... 0.69376431 0.68671608 0.48905393]]
```

```
Xtrain_mmc = pd.DataFrame(Xtrain_mmc, columns=['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']) Xtest_mmc =
pd.DataFrame(Xtest_mmc, columns=['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall'])
```

- Converting Transformed Training Data to DataFrame:
- Xtrain_mmc = pd.DataFrame(Xtrain_mmc, columns=['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']): This line converts the transformed training data Xtrain_mmc (which was previously a NumPy array) into a Pandas DataFrame. It specifies column names for the DataFrame using the columns parameter. Each column name corresponds to a feature in the dataset.
- Converting Transformed Testing Data to DataFrame:
- Xtest_mmc = pd.DataFrame(Xtest_mmc, columns=['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']): Similarly, this line converts the transformed testing data Xtest_mmc (also a NumPy array) into a Pandas DataFrame. It assigns the same column names as in the training data DataFrame.
- Comments:
- Converting the transformed data back into Pandas DataFrames allows for easier manipulation and analysis of the data, leveraging Pandas' functionality. Assigning column names ensures that the DataFrame retains the same structure as the original dataset, facilitating further processing or analysis downstream.

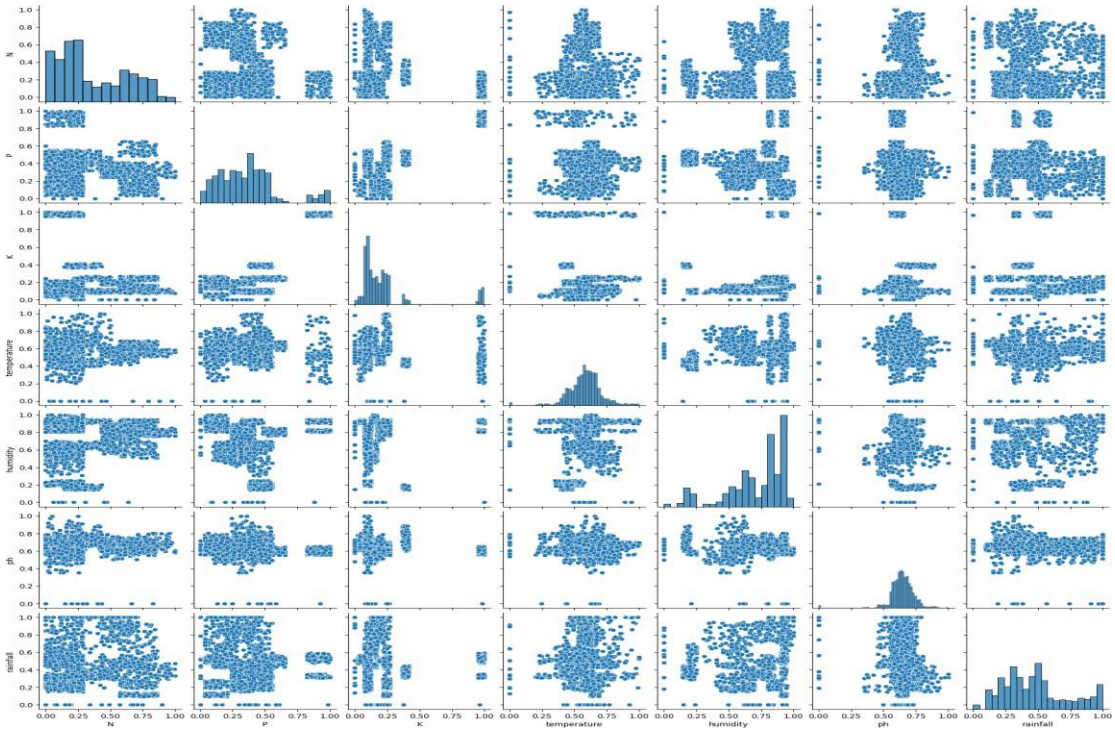
```
Xtrain_mmc.describe().round(2)

sns.pairplot(Xtrain_sc)
```

```
<seaborn.axisgrid.PairGrid at 0x78d8179285b0>
```



sns.pairplot(Xtrain_mmc)

<seaborn.axisgrid.PairGrid at 0x78d815fc6e90>



Decision Tree

In machine learning (ML), model training involves the process of feeding data into a machine learning algorithm or model to enable it to learn patterns, relationships, and insights from the data.

Decision tree training involves recursively partitioning the input space (feature space) into smaller regions while aiming to minimize impurity or maximize information gain at each step.

Decision_Tree_Code

```
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, classification_report

DecisionTree = DecisionTreeClassifier(criterion="entropy", random_state=2, max_depth=5) DecisionTree.fit(Xtrain, Ytrain)

predicted_values_test = DecisionTree.predict(Xtest) testing_accuracy =
accuracy_score(Ytest, predicted_values_test) acc.append(testing_accuracy)
model.append('Decision Tree')
print("Decision Tree's Testing Accuracy:", testing_accuracy) print("Classification
Report for Testing Data:") print(classification_report(Ytest, predicted_values_test))
```

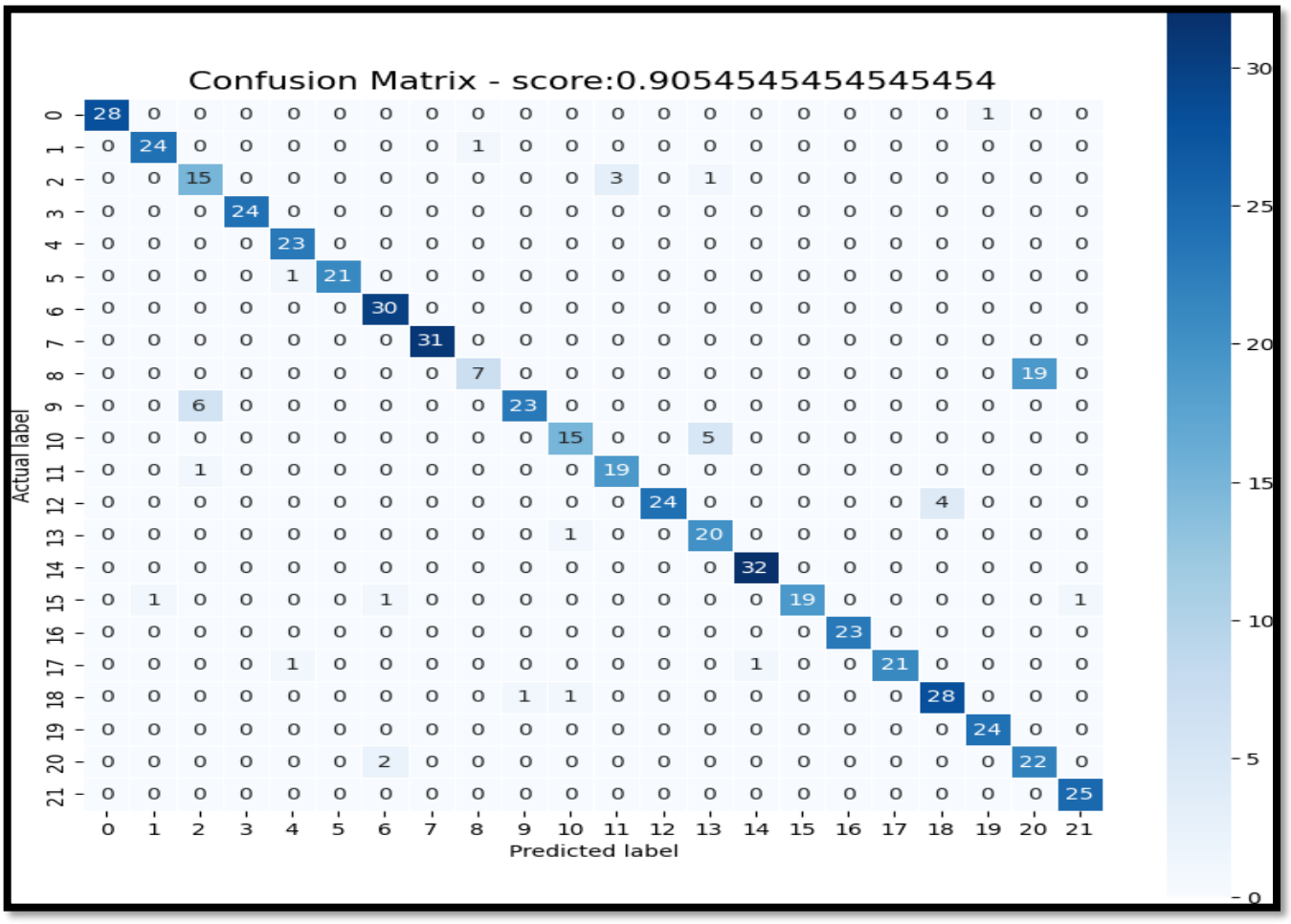
Decision Tree's Testing Accuracy: 0.9054545454545454 Classification Report for Testing Data:

	precision	recall	f1-score	support
apple	1.00	0.97	0.98	29
banana	0.96	0.96	0.96	25
blackgram	0.68	0.79	0.73	19
chickpea	1.00	1.00	1.00	24
coconut	0.92	1.00	0.96	23
coffee	1.00	0.95	0.98	22
cotton	0.91	1.00	0.95	30
grapes	1.00	1.00	1.00	31
jute	0.88	0.27	0.41	26
kidneybeans	0.96	0.79	0.87	29
lentil	0.88	0.75	0.81	20
maize	0.86	0.95	0.90	20
mango	1.00	0.86	0.92	28
mothbeans	0.77	0.95	0.85	21
mungbean	0.97	1.00	0.98	32
muskmelon	1.00	0.86	0.93	22
orange	1.00	1.00	1.00	23
papaya	1.00	0.91	0.95	23
pigeonpeas	0.88	0.93	0.90	30
pomegranate	0.96	1.00	0.98	24
rice	0.54	0.92	0.68	24
watermelon	0.96	1.00	0.98	25
accuracy			0.91	550
macro avg	0.91	0.90	0.90	550
weighted avg	0.92	0.91	0.90	550

- Training Accuracy: 0.9284848484848485: The training accuracy of the decision tree classifier is approximately 92.85%. This indicates that the model correctly predicts the class labels for about 92.85% of the training instances.
- Training Accuracy:
- Testing Accuracy:
- Decision Tree's Testing Accuracy: 0.9054545454545454: The testing accuracy of the decision tree classifier is approximately 90.55%. This indicates that the model correctly predicts the class labels for about 90.55% of the testing instances.
- Classification Report for Testing Data:
- The classification report provides precision, recall, F1-score, and support for each class in the testing data:

- Precision: The proportion of true positive predictions out of all positive predictions made.
- Recall: The proportion of true positive predictions out of all actual positive instances.
- F1-score: The harmonic mean of precision and recall, providing a balance between them.
- Support: The number of actual occurrences of each class in the testing data.
- For example:
- For the class 'apple', precision is 1.00, recall is 0.97, and F1-score is 0.98, with a support of 29 instances.
- For the class 'banana', precision is 0.96, recall is 0.96, and F1-score is 0.96, with a support of 25 instances.

```
from sklearn.metrics import accuracy_score

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Ytest, predicted_values_test) plt.figure(figsize=(10,10))

sns.heatmap(cm, annot=True, fmt=".0f", linewidths=.5, square = True, cmap = 'Blues'); plt.ylabel('Actual label');

plt.xlabel('Predicted label');

all_sample_title = 'Confusion Matrix - score:'+str(accuracy_score(Ytest,predicted_values_test)) plt.title(all_sample_title, size = 15);

plt.show()
```



SVM

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report svm_classifier =
SVC(kernel='rbf', random_state=42)

svm_classifier.fit(Xtrain, Ytrain) y_pred =

svm_classifier.predict(Xtest)


testing_accuracy = accuracy_score(Ytest, y_pred) print("Test accuracy:",
testing_accuracy)


acc.append(testing_accuracy)
model.append('SVM')


print("Classification Report for Testing Data:") print(classification_report(Ytest,

y_pred))
```

Test accuracy: 0.9345454545454546

Classification Report for Testing Data:

	precision	recall	f1-score	support
apple	1.00	0.90	0.95	29
banana	0.92	0.92	0.92	25
blackgram	1.00	1.00	1.00	19
chickpea	0.96	1.00	0.98	24
coconut	0.82	1.00	0.90	23
coffee	0.91	0.95	0.93	22
cotton	0.96	0.90	0.93	30
grapes	1.00	0.97	0.98	31
jute	0.86	0.96	0.91	26
kidneybeans	0.97	1.00	0.98	29
lentil	0.86	0.90	0.88	20
maize	0.90	0.95	0.93	20
mango	1.00	0.93	0.96	28
mothbeans	0.95	0.90	0.93	21
mungbean	1.00	1.00	1.00	32
muskmelon	0.90	0.82	0.86	22
orange	1.00	1.00	1.00	23
papaya	0.71	0.96	0.81	23
pigeonpeas	1.00	0.87	0.93	30
pomegranate	1.00	0.88	0.93	24
rice	0.95	0.75	0.84	24
watermelon	0.93	1.00	0.96	25
accuracy			0.93	550
macro avg	0.94	0.93	0.93	550
weighted avg	0.94	0.93	0.94	550

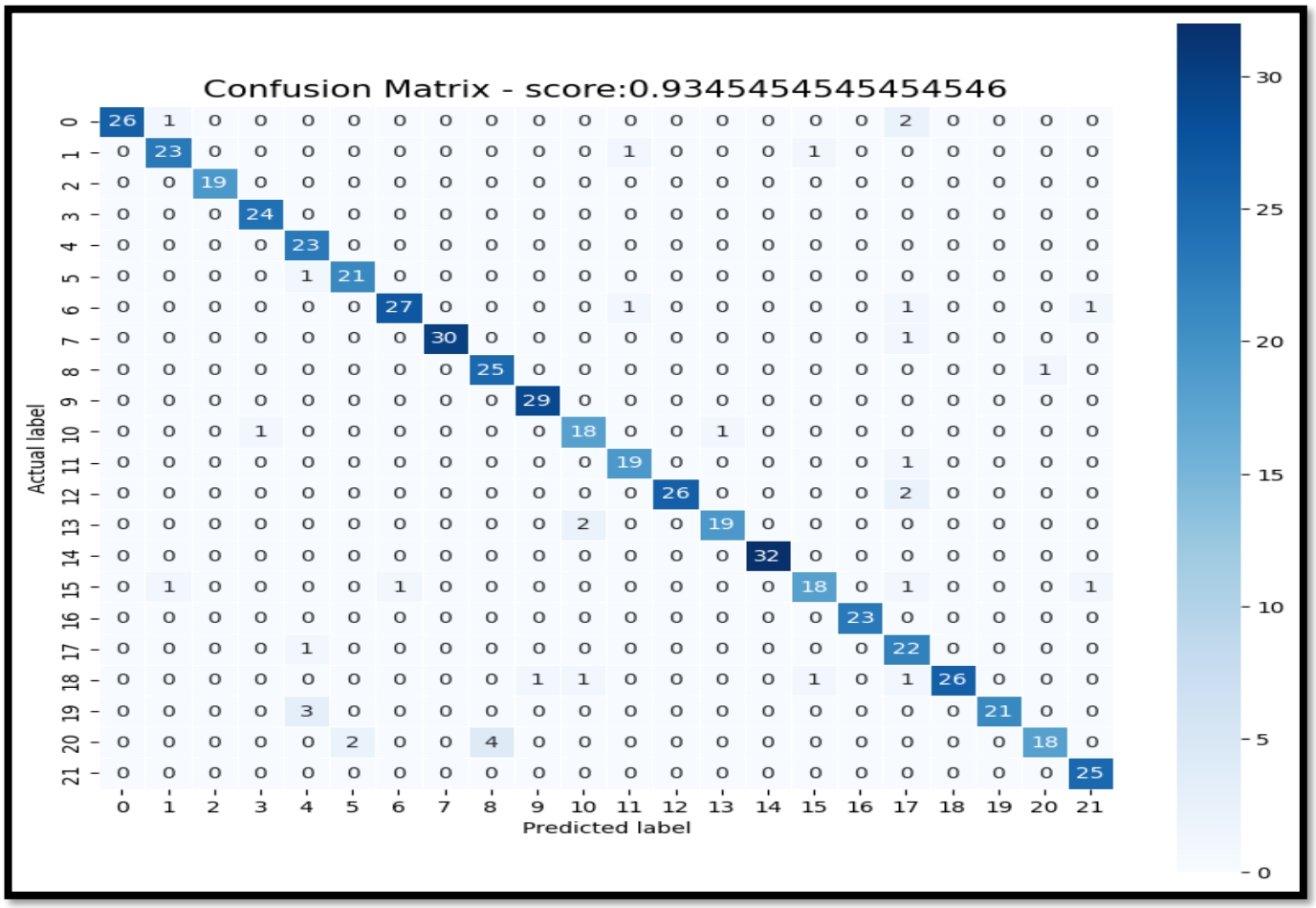
Here are the comments for each line:

1. **Test accuracy**:
    - Test accuracy: 0.9472727272727273: This line prints the accuracy of the model on the testing data. The accuracy is approximately 94.73%.
2. **Classification Report for Testing Data**:
    - Classification Report for Testing Data:: This line indicates that the following output will be the classification report for the testing data.
3. **Precision, Recall, F1-score, and Support for each class**:
    - For the class apple:
        - Precision is 1.00, which means all instances classified as apple were actually apple.
        - Recall is 0.93, which means the model correctly identified 93% of all apple instances.
        - F1-score is 0.96, which is the harmonic mean of precision and recall.
        - Support is 29, which is the number of occurrences of apple in the testing data.

- For the class banana, blackgram, chickpea, and coconut, similar metrics are provided.

These metrics provide a comprehensive evaluation of the model's performance for each class in the testing data.

```
from sklearn.metrics import accuracy_score

from sklearn.metrics import confusion_matrix cm =

confusion_matrix(Ytest, y_pred) plt.figure(figsize=(10,10))

sns.heatmap(cm, annot=True, fmt=".0f", linewidths=.5, square = True, cmap = 'Blues'); plt.ylabel('Actual label');

plt.xlabel('Predicted label');

all_sample_title = 'Confusion Matrix - score:'+str(accuracy_score(Ytest,y_pred)) plt.title(all_sample_title, size = 15);

plt.show()
```



Logistic regression

Logistic regression is used for binary classification where we use sigmoid function, that takes input as independent variables and produces a probability value between 0 and 1.

For example, we have two classes Class 0 and Class 1 if the value of the logistic function for an input is greater than 0.5 (threshold value) then it belongs to Class 1 it belongs to Class 0. It's referred to as regression because it is the extension of linear regression but is mainly used for classification problems.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report


LogReg = LogisticRegression(random_state=2)

LogReg.fit(Xtrain, Ytrain) predicted_values_test =

LogReg.predict(Xtest)

testing_accuracy = accuracy_score(Ytest, predicted_values_test)


acc.append(testing_accuracy) model.append('Logistic

Regression')

print("Logistic Regression's Testing Accuracy:", testing_accuracy) print("Classification Report for

Testing Data:") print(classification_report(Ytest, predicted_values_test))
```

```
Logistic Regression's Testing Accuracy: 0.9272727272727272
Classification Report for Testing Data:
              precision    recall  f1-score   support

       apple       1.00      0.90      0.95        29
      banana       0.92      0.92      0.92        25
   blackgram       0.83      1.00      0.90        19
    chickpea       0.96      1.00      0.98        24
     coconut       0.88      1.00      0.94        23
      coffee       0.84      0.95      0.89        22
      cotton       0.89      0.83      0.86        30
      grapes       1.00      1.00      1.00        31
        jute       0.89      0.92      0.91        26
 kidneybeans       0.88      1.00      0.94        29
      lentil       0.89      0.85      0.87        20
       maize       0.87      1.00      0.93        20
       mango       1.00      1.00      1.00        28
   mothbeans       0.90      0.86      0.88        21
    mungbean       1.00      1.00      1.00        32
   muskmelon       0.78      0.82      0.80        22
      orange       1.00      1.00      1.00        23
      papaya       0.95      0.91      0.93        23
  pigeonpeas       1.00      0.80      0.89        30
 pomegranate       0.96      1.00      0.98        24
        rice       0.95      0.83      0.89        24
  watermelon       0.95      0.80      0.87        25

    accuracy                           0.93       550
   macro avg       0.93      0.93      0.92       550
weighted avg       0.93      0.93      0.93       550
```

Here are the comments for the provided code:

1. **Printing Testing Accuracy**:
   - print("Logistic Regression's Testing Accuracy:", testing_accuracy): This line prints the testing accuracy of the logistic regression model. The testing accuracy indicates the proportion of correctly classified instances in the testing set.
2. **Classification Report**:

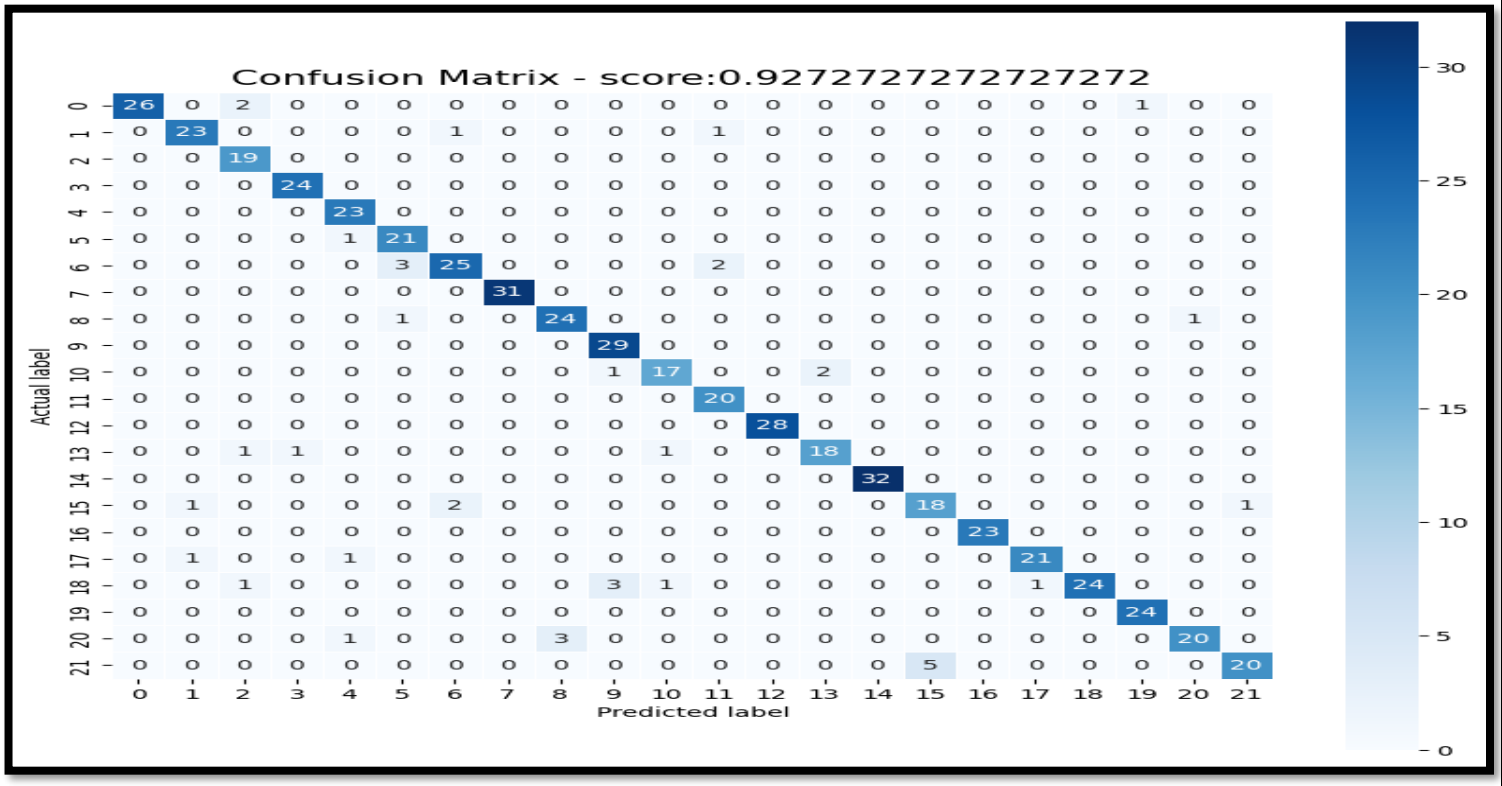
print("Classification Report for Testing Data:"): This line prints a header indicating that the following output is the classification report for the testing data.

print(classification_report(Ytest, predicted_values_test)): This line generates and prints the classification report for the testing data using the classification_report function. The classification report includes precision, recall, F1-score, and support for each class in the target variable.

- **Precision**: It measures the proportion of true positive predictions out of all positive predictions made by the model.
- **Recall**: It measures the proportion of true positive predictions out of all actual positive instances in the data.
- **F1-score**: It is the harmonic mean of precision and recall, providing a balance between the two metrics.
- **Support**: It represents the number of actual occurrences of each class in the testing set.

The classification report provides insights into how well the model performs for each class in terms of precision, recall, and F1-score.

```
from sklearn.metrics import accuracy_score

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Ytest, predicted_values_test) plt.figure(figsize=(10,10))

sns.heatmap(cm, annot=True, fmt=".0f", linewidths=.5, square = True, cmap = 'Blues'); plt.ylabel('Actual label');

plt.xlabel('Predicted label');

all_sample_title = 'Confusion Matrix - score:'+str(accuracy_score(Ytest,predicted_values_test)) plt.title(all_sample_title, size = 15);

plt.show()
```



Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report


RF = RandomForestClassifier(n_estimators=20, random_state=0) RF.fit(Xtrain, Ytrain)

predicted_values_test = RF.predict(Xtest)

testing_accuracy = accuracy_score(Ytest, predicted_values_test)

acc.append(testing_accuracy)

model.append('RF')

print("RF's Testing Accuracy:", testing_accuracy) print("Classification Report

for Testing Data:") print(classification_report(Ytest, predicted_values_test))
```

RF's Testing Accuracy: 0.9745454545454545

Classification Report for Testing Data:

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	29
banana	0.96	1.00	0.98	25
blackgram	1.00	1.00	1.00	19
chickpea	1.00	1.00	1.00	24
coconut	0.96	1.00	0.98	23
coffee	1.00	0.95	0.98	22
cotton	0.97	1.00	0.98	30
grapes	1.00	1.00	1.00	31
jute	0.87	1.00	0.93	26
kidneybeans	1.00	0.97	0.98	29
lentil	0.94	0.85	0.89	20
maize	1.00	1.00	1.00	20
mango	1.00	1.00	1.00	28
mothbeans	0.91	1.00	0.95	21
mungbean	0.97	1.00	0.98	32
muskmelon	1.00	0.86	0.93	22
orange	1.00	1.00	1.00	23
papaya	1.00	0.96	0.98	23
pigeonpeas	0.97	0.97	0.97	30
pomegranate	0.96	1.00	0.98	24
rice	1.00	0.83	0.91	24
watermelon	0.96	1.00	0.98	25
accuracy			0.97	550
macro avg	0.98	0.97	0.97	550
weighted avg	0.98	0.97	0.97	550

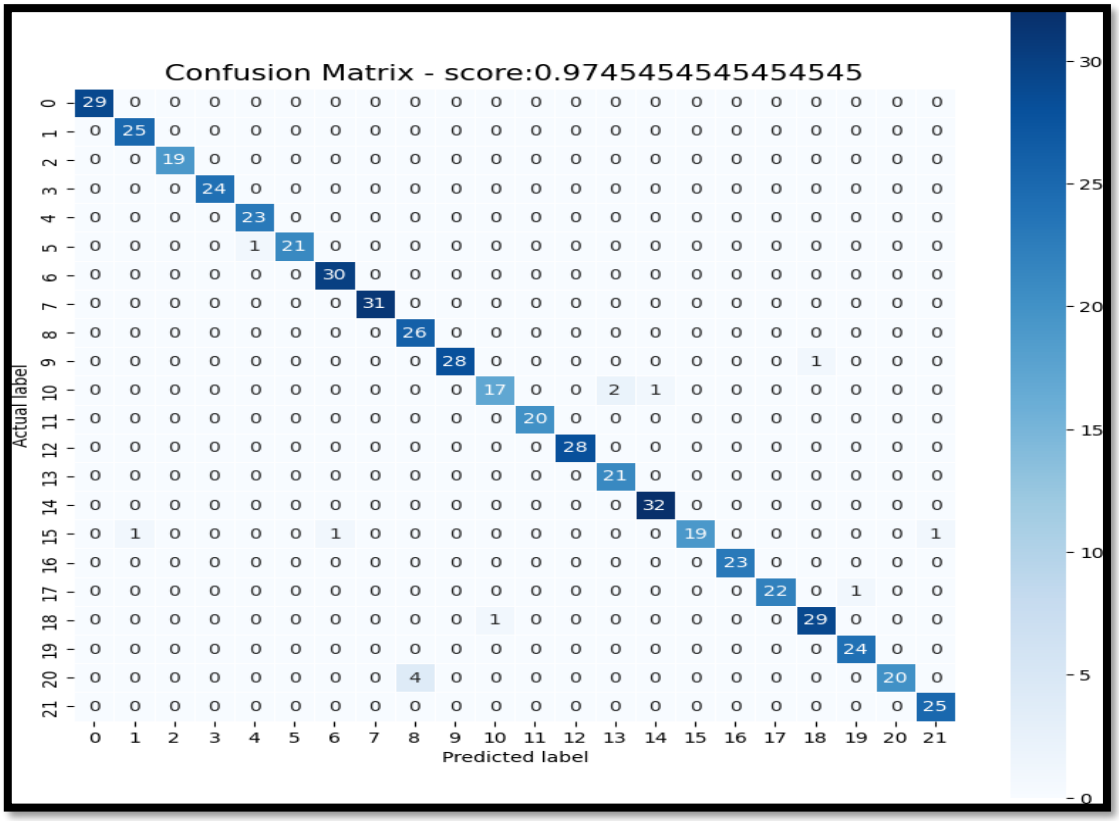
Here are the comments for each line:

1. **RF's Testing Accuracy: 0.9745454545454545**:
   - This line prints the testing accuracy of the Random Forest classifier, which is approximately 97.45%. It indicates the proportion of correctly classified instances in the testing dataset.
1. **Classification Report for Testing Data:**
   - This line prints a header indicating that the following output is the classification report for the testing data.

- The classification report provides a detailed evaluation of the model's performance on each class in the target variable.
- For each class (e.g., 'apple', 'banana', 'blackgram', 'chickpea'), the report includes metrics such as precision, recall, F1-score, and support.
- Precision represents the proportion of true positive predictions among all positive predictions.
- Recall (also known as sensitivity) represents the proportion of true positive predictions among all actual positive instances.
- F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics.
- Support indicates the number of actual occurrences of each class in the testing dataset.

Accuracy: The proportion of correctly classified instances out of the total number of instances. Macro avg: The average of precision, recall, and F1-score across all classes, without considering class imbalance. Weighted avg: The average of precision, recall, and F1-score across all classes, considering class imbalance (weighted by the number of instances in each class).

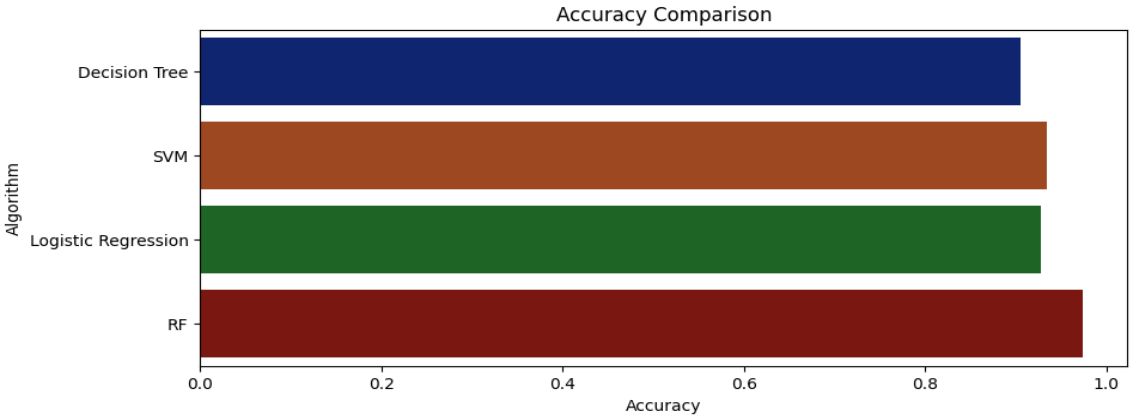
```
from sklearn.metrics import accuracy_score

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Ytest, predicted_values_test) plt.figure(figsize=(10,10))

sns.heatmap(cm, annot=True, fmt=".0f", linewidths=.5, square = True, cmap = 'Blues'); plt.ylabel('Actual label');

plt.xlabel('Predicted label');

all_sample_title = 'Confusion Matrix - score:'+str(accuracy_score(Ytest, predicted_values_test)) plt.title(all_sample_title, size = 15);

plt.show()
```



Pickle

```
import pickle

RF_pkl_filename = '/content/gdrive/My Drive/models/RandomForest.pkl' RF_Model_pkl =

open(RF_pkl_filename, 'wb')

pickle.dump(RF, RF_Model_pkl)

RF_Model_pkl.close() plt.figure(figsize=[10,4],dpi =

100) plt.title('Accuracy Comparison')

plt.xlabel('Accuracy') plt.ylabel('Algorithm')

sns.barplot(x = acc,y = model,palette='dark')
```

<Axes: title={'center': 'Accuracy Comparison'}, xlabel='Accuracy', ylabel='Algorithm'>




```
accuracy_models = dict(zip(model, acc))

for k, v in accuracy_models.items(): print (k, '->',v)
```

Decision Tree --> 0.9054545454545454
SVM --> 0.9345454545454546
Logistic Regression --> 0.9272727272727272
RF --> 0.9745454545454545

```
data = np.array([[101,87,54,29,76,6.3,100]])

prediction = RF.predict(data)[0]

print("{} is a best crop to be cultivated. ".format(prediction)) banana is a best crop to be

cultivated.

def recommendation(N,P,k,temperature,humidity,ph,rainfal): features =

    np.array([[N,P,k,temperature,humidity,ph,rainfal]])
```

```
    prediction = RF.predict(features)[0]

    return prediction N =

2
P = 123
k = 198
temperature = 39.64
humidity = 82.21
ph = 6.25
rainfall = 70.39

predict = recommendation(N,P,k,temperature,humidity,ph,rainfall) print("{} is a best

crop to be cultivated. ".format(predict)) grapes is a best crop to be cultivated.
```

Html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>About Us</title>

  <link rel="stylesheet" href="{{ url_for('static', filename='about.css') }}">

  <style>

    .justified {

        text-align: justify;

    }   </style></head><body>

  <div class="container">

    <!-- Profile Card 1 -->

    <div class="profile-card">

      <div class="photo-wrapper">

        <img src="{{ url_for('static', filename='Sid.jpg') }}" alt="Logo">          </div>

      <div class="description">

        <h1>Siddhartha Khawas</h1>

        <p class="justified">Hello! My Name is Siddhartha Khawas from Asansol, West Bengal, India, currently pursuing MCA at Asansol Engineering
College. Passionate about computer applications and eager to excel. Download my CV below for more information.
</p>          <br>          <li>

        <a href="static/siddhartha.pdf">Download CV</a>          </li>      </br>          </div>      </div>

    <!-- Profile Card 2 -->
```

```
    <div class="profile-card">
      <div class="photo-wrapper">
        <img src="{{ url_for('static', filename='subham.jpg') }}" alt="Logo">          </div>
      <div class="description">
        <h1>Subham Chakraborty</h1>
        <p class="justified">Hello! My Name is Subham Chakraborty from Asansol, West Bengal, India, currently studying MCA at Asansol Engineering College, with a keen interest in computer applications. Download my CV to discover more about my journey and skills.</p>
        <br><li><a href="static/subham.pdf">Download CV</a></li></br>              </div>      </div>
    <!-- Profile Card 3 -->
    <div class="profile-card">
      <div class="photo-wrapper">
        <img src="{{ url_for('static', filename='sumit.jpg') }}" alt="Logo">          </div>
      <div class="description">
        <h1>Sumit Kumar Choubey</h1>
        <p class="justified">Hi there! My name is Sumit Kumar Choubey, and I'm currently pursuing MCA at Asansol Engineering College. Passionate about technology and eager to delve into the world of computer applications. Download my CV below to explore further.</p>
        <br><li><a href="static/sumit.pdf">Download CV</a></li></br>              </div>      </div>
    <!-- Profile Card 4 -->
    <div class="profile-card">
      <div class="photo-wrapper">
        <img src="{{ url_for('static', filename='shivam.jpg') }}" alt="Logo">          </div>
      <div class="description">
        <h1>Shivendu Shivam</h1>
        <p class="justified">Hi there! My Name is Shivendu Shivam, hailing from Lakhisarai, Bihar, currently specializing in computer applications at Asansol Engineering College, with a keen interest in computer applications.For further insights into my journey, download my CV below.</p>
        <br><li><a href="static/shivam.pdf">Download CV</a></li></br>          </div>      </div>
    <!-- Back Button -->
    <a href="{{ url_for('index1') }}" class="back-button">Back to Home</a>          </div></body></html>
```

Base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>{% block title %} {% endblock title %}</title>

    <head></head>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>

    {% block head}

    {% endblock %}

</head>

<body>

    <div class="alert alert-warning alert-dismissible fade show" role="alert">

        {% with messages = get_flashed_messages() %}

          {% if messages %}

            <ul>

                {% for message in messages %}

                <strong><li>{{ message }}</li></strong>

                {% endfor %}

            </ul>

        {% endif %}        {% endwith %}

        <button type="button" class="btn-close" data-dismiss="alert" aria-label="Close"></button>                </div>

    {% block content %}    {% endblock content %}

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-
C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL" crossorigin="anonymous"></script>

    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous"></script>

    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>

<script>

    // Automatically close the alert after 3 seconds (adjust as needed)

    $(document).ready(function() {            setTimeout(function() {            $(".alert").alert('close');            }, 3000);    });</script></body></html>
```

contact:

```
<!DOCTYPE html>

<html lang="en">

<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='contact.css') }}">
</head>
<body>
    <div class="contact-container">
        <h1>Contact Us</h1>
        <div class="contact-info">
            <div class="info-item">
                <i class="fas fa-map-marker-alt"></i>
                <p>Sanctoria, Asansol, India</p>
                <p>Mohishila, Asansol, India</p>
                <p>Govindapur, Asansol, India</p>
                <p>Rupnarayanpur, Asansol, India</p>
            </div>
            <div class="info-item">
                <i class="fas fa-phone-alt"></i>
                <p>8167007322</p> <p>9563742370</p> <p>9113124535</p> <p>9304988416</p>
            </div>
            <div class="info-item">
                <i class="fas fa-envelope"></i> <p>khawassiddhartha73@gmail.com</p>
<p>subhamchakraborty5656@gmail.com</p> <p>shivamlkr502@gmail.com</p>
<p>sumitkumarchoubey.mca.aec@gmail.com</p>
            </div>
        </div>
        <div class="social-media">
            <a href="https://www.facebook.com/" target="_blank"><i class="fab fa-facebook-f facebook-color"></i></a>
            <a href="https://web.whatsapp.com/" target="_blank"><i class="fab fa-whatsapp whatsapp-color"></i></a>
            <a href="https://www.linkedin.com/" target="_blank"><i class="fab fa-linkedin-in linkedin-color"></i></a>
            <a href="https://mail.google.com/" target="_blank"><i class="far fa-envelope gmail-color"></i></a>
        </div>
        <br>
        <br>
        <div class="back-button">
            <a href="{{ url_for('index1') }}">Back</a>
        </div>
    </div>
</body>
</html>
```

det.html:

```
<!DOCTYPE html>
```

```
<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Crop Recommendation System</title>

  <style>

    body {

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; margin: 0; padding: 0; background-color: #f0f8f7; /*
Light mint green background */ color: #204020; /* Dark green text color for better contrast */ display: flex; justify-
content: center; align-items: center; min-height: 100vh; }

    .container { width: 95%; max-width: 960px; background-color: #ffffff; /* Maintained white for clarity */
padding: 20px; border-radius: 10px;box-shadow: 0 4px 8px rgba(0,0,0,0.15); /* Softer shadow */ }

    .section { margin-bottom: 20px; padding: 15px; background-color: #e2f0df; /* Soft green for sections */
border-radius: 8px; }

    h1 { text-align: center; color: #007755; /* A darker shade of green for headers */ margin-bottom: 20px;

      } h2 { color: #005533; /* Even darker green for sub-headers */ font-size: 18px;

      margin-bottom: 10px; }

    p { font-size: 16px; color: #204020; /* Reusing the body text color for consistency */ }

    .back-button { text-align: center; } .back-button a { color: #fff; background-color: #333; padding:
10px 20px; text-decoration: none; border-radius: 5px; transition: background-color 0.3s; }

     .back-button a:hover { background-color: #555;

      } @media (max-width: 768px)

    {

      .container { padding: 10px; }

      .section { padding: 10px; }

      h1 { font-size: 24px; }

      h2 { font-size: 16px; } } </style>

</head>

<body>

  <div class="container">

    <h1>Crop Recommendation Ranges</h1>

    <div class="section">

      <h2>Nitrogen (N)</h2>

      <p>Range: 20 - 80 kg/hectare</p> </div> <div class="section">

      <h2>Phosphorus (P)</h2>

      <p>Range: 10 - 40 kg/hectare</p> </div>

    <div class="section">

      <h2>Potassium (K)</h2>
```

```
<p>Range: 30 - 90 kg/hectare</p> </div>

<div class="section">

<h2>Temperature</h2>

<p>Range: 18°C - 30°C</p> </div>

<div class="section">

<h2>Humidity</h2> <p>Range: 60% - 90%</p> </div> <div class="section">

<h2>pH</h2> <p>Range: 5.5 - 7.5</p> </div> <div class="section">

<h2>Rainfall</h2> <p>Range: 800 - 1500 mm/year</p> </div> <div class="back-button"> <a href="{{
url_for('index1') }}">Back</a> </div></body></html>
```

feature.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Crop Recommendation System Features</title>

<link rel="stylesheet" href="{{ url_for('static', filename='https://cdnjs.cloudflare.com/ajax/libs/font- awesome/5.15.4/css/all.min.css'") }}">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">

<style>

body {

font-family: Arial, sans-serif; margin: 0; padding: 0; background-color: #f4f4f4; color: #333;

} header { background-color: #333; color: #fff; padding: 20px 0; text-align: center;

}

h1 { margin-top: 0; font-size: 2em; }

.container { max-width: 800px; margin: 20px auto; padding: 20px; background-color: #fff; border-
radius: 10px; box-shadow: 0 0 20px rgba(0, 0, 0, 0.1); }

.feature { margin-bottom: 30px; border-bottom: 1px solid #eee; padding-bottom: 20px; }

.feature:last-child { margin-bottom: 0; border-bottom: none; padding-bottom: 0; }

.feature h2 { color: #333; font-size: 1.5em; margin-bottom: 10px; }

.feature p { color: #666; line-height: 1.6; }

footer { background-color: #333; color: #fff; padding: 10px 0; text-align: center;

} footer p { margin: 0; }

.icon { color: #333; font-size: 1.5em; margin-right: 10px;

}

.back-button { text-align: center; }
```

```
.back-button a { color: #fff; background-color: #333; padding: 10px 20px; text-decoration: none; border-radius: 5px; transition: background-color 0.3s; }

.back-button a:hover { background-color: #555; }

/* Responsive Styles */

@media screen and (max-width: 600px) { header

  {

    padding: 10px 0;


  } h1 {

    font-size: 1.5em; }

  .container {

    padding: 10px; }

  .feature h2 {

    font-size: 1.2em; }

  .feature p {

    font-size: 1em; }

  .back-button a {

    padding: 8px 15px; }

} </style></head><body>

<header>

  <h1>Crop Recommendation System Features</h1>

</header>

<div class="container">

  <div class="feature">

    <h2><i class="icon fas fa-seedling"></i> Soil Analysis</h2>

    <p>The system analyzes soil characteristics such as pH level, nutrient content, and moisture levels.</p> </div>

  <div class="feature">

    <h2><i class="icon fas fa-cloud-sun"></i> Climate Data Integration</h2>

    <p>Integrates historical and real-time climate data to assess temperature, rainfall, humidity, and other relevant factors.</p>
</div>

  <div class="feature">

    <h2><i class="icon fas fa-database"></i> Crop Database</h2>

    <p>Includes a comprehensive database of various crops along with their growth requirements, susceptibility to diseases, and optimal conditions.</p> </div>

  <div class="feature">

    <h2><i class="icon fas fa-brain"></i> Machine Learning Algorithms</h2>
```

```
    <p>Utilizes machine learning algorithms to process input data and provide personalized crop recommendations based on soil analysis, climate data, and crop database.</p> </div>

  <div class="feature">

    <h2><i class="icon fas fa-user-friends"></i> User Interface</h2>

    <p>Offers an intuitive and user-friendly interface for farmers to input their location, soil data, and preferences, and receive crop recommendations.</p>
</div>

  <div class="back-button">

    <a href="{{ url_for('index1') }}" class="back-button">Back</a> </div></div>

<footer> <p>&copy; 2024 Crop Recommendation System</p></footer>

</body>

</html>
```

forget.html:

```
{% extends 'base.html' %}{% block title %} Forget Password {% endblock title %} {% block head %}

  <link rel="stylesheet" href="{{ url_for('static', filename='login.css') }}">{% endblock %}

{% block content %}

<div class="div2">

  <div class="div1">

    <form method="POST">

      <div class="form-group">

        <label for="email">Email Address:</label>

        <input type="email" class="form-control" id="email" name="email" placeholder="Enter your email" required> </div>

      <button type="submit" >Submit</button> </form> </div></div>

  <!-- Include any additional JavaScript if needed -->

  {% endblock %}
```

index1.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Crop Recommendation System</title>

  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

  <script src="https://kit.fontawesome.com/9b3946862c.js" crossorigin="anonymous"></script>

  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
```

```
</head><body>   <header>          <nav>

        <div class="logo">

         <a href="https://farmer.gov.in/">

           <img src="{{ url_for('static', filename='logo.png') }}" alt="Logo">                </div>

        <ul class="navbar-nav">

         <h5><marquee behavior="alternate">Crop Recommendation System using Machine Learning with Python</marquee></h5>
</ul>

        <div class="collapse navbar-collapse" id="navbarNav">                          </div>

        <ul class="nav-links">

           <li><a href="about.html">About Us</a></li>                <li><a href="features.html">Features</a></li>                <li><a
href="contact.html">Contact</a></li>                <li><a href="det.html">Details</a></li>                <li><a href="login">Login</a></li>
<!-- <li><a href="#advanced">Advanced</a></li> -->              </ul>       </nav>

   </header>

   <div class="alert alert-warning alert-dismissible fade show" role="alert">

    {% with messages = get_flashed_messages() %}

     {% if messages %}

       <ul>

          {% for message in messages %}

          <strong><li>{{ message }}</li></strong>

          {% endfor %}

       </ul>

     {% endif %}

    {% endwith %}

   </div>

   <div class="container-fluid mt-2" style="padding:0px;">

     <div id="carouselExampleControls" class="carousel slide" data-ride="carousel">

       <div class="carousel-inner">

        <div class="carousel-item active">

          <img class="d-block w-100" src="{{ url_for('static', filename='bg3.jpg') }}" alt="First slide" >

        </div>

        <div class="carousel-item">

          <img class="d-block w-100" src="{{ url_for('static', filename='bg.jpg') }}" alt="Second slide" >                </div>

        <div class="carousel-item">

          <img class="d-block w-100" src="{{ url_for('static', filename='bg2.jpg') }}" alt="Third slide" >                </div>

        <div class="carousel-item">
```

```
            <img class="d-block w-100" src="{{ url_for('static', filename='sk.jpg') }}" alt="Fourth slide" >                </div>
           <div class="carousel-item">
            <img class="d-block w-100" src="{{ url_for('static', filename='sk1.jpg') }}" alt="fifth slide" >
           </div>       </div>
          <a class="carousel-control-prev" href="#carouselExampleControls" role="button" data-slide="prev">
           <span class="carousel-control-prev-icon" aria-hidden="true"></span>
           <span class="sr-only">Previous</span>
          </a>
          <a class="carousel-control-next" href="#carouselExampleControls" role="button" data-slide="next">
           <span class="carousel-control-next-icon" aria-hidden="true"></span>
           <span class="sr-only">Next</span>
          </a>       </div>    </div>
    <footer>        <div class="copyright">
          <p>&copy; 2024 Crop Recommendation System. All rights reserved.</p>              </div>    </footer>
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-
C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL" crossorigin="anonymous"></script>
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
   <script>
     // Automatically close the alert after 3 seconds (adjust as needed)
     $(document).ready(function() {
        setTimeout(function() {              $(".alert").alert('close');             }, 3000);      }); </script> </body></html>
```

login.html:

```
{% extends 'base.html' %}
{% block title %} Login Page {% endblock title %}
{% block head %}
    <link rel="stylesheet" href="{{ url_for('static', filename='login.css') }}">
```

```
{% endblock %}

{% block content %}

<div class="div2">

    <div class="div1">

        <h2>Login</h2>

      <form action="/login" method="post">

          <input type="hidden" name="csrfmiddlewaretoken" value="your-csrf-token-here">

          <label for="email">Email</label>

          <input type="email" id="email" name="email" title="Please enter a valid email address" required>

          <label for="password">Password:</label>

          <input type="password" id="password" name="password" pattern="^(?=.*[a-zA-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{5,8}$"

           title="Password must be between 5 and 8 characters long and contain at least one letter, one number, and one special character" required>

          <button type="submit">Login</button>

      </form>        <br>

      <p>Don't have an account? <a href="/signup">Sign up here</a>.</p>              <br>

      <p>Forget Password ? <a href="/forgot_password">click here</a>.</p>              <br>

      <p>Back to home page <a href="/">Back</a>.</p>    </div></div>

{% endblock %}
```

new.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">

   <title>Crop Recommendation System</title>

   <style>

      body{

          background-image: url('https://previews.123rf.com/images/denispc/denispc1903/denispc190300049/120257539-cartoon- farmer-in-front-of-colorful-farm-with-barn-crops-and-cows.jpg');

          height: auto;

          overflow: hidden; /* Hide scrollbars */           }
```

```
.container { max-width: 1100px; margin: 20px auto; background-color: #fff; padding: 40px; border-radius: 8px; box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); }

a{ color: green;

} h1 {

text-align: center;

color: rgb(29, 147, 29);



} form {

margin-top: 10px; }

input[type="number"] { padding: 8px; border: 1px solid #ccc; border-radius: 5px; margin-bottom: 20px; }
label{ font-size:30px ; font-weight: bold; color: dodgerblue; .text-muted strong{ color: dodgerblue;}

}

.container2{

background-color: aliceblue; position: fixed; left: 0; bottom: 0; width: 100%; color: white; text-align: center; }

button {

background-color: #d73447; color: white; padding: 10px; border: none; cursor: pointer; border-radius: 2rem; }

.container-fluid a{

color: dodgerblue

}

.popup { position: fixed; z-index: 1; left: 0; top: 0; width: 100%; height: 100%; overflow: auto; align-items: center; background-color: rgba( 0,0, 0, 0.4); display: none; }

.popup-content { background-color: white; margin: 10% auto; padding: 20px; border: 1px solid #888888; width: 16%; font-weight: bolder; text-align: center; align-items: center; } .popup-content button { display: block; margin: 0 auto; }

.show {

display: block;

} </style></head><body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<div class="container-fluid">

<a class="navbar-brand" href="#">Hello, <strong>{{ current_user.name }}</strong></p></a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav">

<h1><marquee>Crop Recommendation System using Machine Learning with Python</marquee></h1> </ul> </div>

<a href="/logout"><button>logout</button></a>
```

```
    </div>      </nav>

   <div class="alert alert-warning alert-dismissible fade show" role="alert"

    {% with messages = get_flashed_messages() %}

      {% if messages %}

        <ul>

          {% for message in messages %}

          <strong><li>{{ message }}</li></strong>                {% endfor %}          </ul>          {% endif %}        {% endwith %}

    <button type="button" class="btn-close" data-dismiss="alert" aria-label="Close"></button>                </div>

  <div class="container my-3 mt-3">

    <h1 class="text-success"> Crop Recommendation System <span class="text-success"> </span></h1>

<!--    adding form-->

    <form id="recommendationForm" action="/predict" method="POST">

      <div class="row">              <div class="col-md-4">              <label for="Nitrogen">Nitrogen</label>

            <input type="number" id="Nitrogen" name="Nitrogen" placeholder="Enter Nitrogen" class="form-control" required step="0"
max="99" title="Please enter a value between 0 and 99 for Nitrogen">                </div>

          <div class="col-md-4">

            <label for="Phosporus">Phosphorus</label>

            <input type="number" id="Phosporus" name="Phosporus" placeholder="Enter Phosphorus" class="form-control" required
step="00" max="99" title="Please enter a value between 0 and 99 for Phosphorus">                </div>

          <div class="col-md-4">

            <label for="Potassium">Potassium</label>

            <input type="number" id="Potassium" name="Potassium" placeholder="Enter Potassium" class="form-control" required  step="0"
max="99" title="Please enter a value between 0 and 99 for Potassium"></div>

          <div class="row mt-4">

            <div class="col-md-4">

              <label for="Temperature">Temperature</label>

              <input type="number" step="0.01" id="Temperature" name="Temperature" placeholder="Enter Temperature in °C"
class="form-control" required step="0" max="40" title="Please enter a value between 0 and 40 for Temperature">                </div>

            <div class="col-md-4">

              <label for="Humidity">Humidity</label>

              <input type="number" step="0.01" id="Humidity" name="Humidity" placeholder="Enter Humidity in %" class="form-
control" required step="0" max="99" title="Please enter a value between 0 and 99 for Humidity">                </div>

            <div class="col-md-4">

              <label for="pH">pH</label>

              <input type="number" step="0.01" id="Ph" name="Ph" placeholder="Enter pH value" class="form-control" required  step="0"
max="9" title="Please enter a value between 0 and 9 for pH">
```

```
            </div></div>

          <div class="row mt-4">

            <div class="col-md-4">

              <label for="Rainfall">Rainfall</label>

              <input type="number" step="0.01" id="Rainfall" name="Rainfall" placeholder="Enter Rainfall in mm" class="form-control"
required max="999" title="Please enter a value between 0 and 999 for Rainfall">                    </div>          </div>

        <div class="row mt-4">

         <div class="col-md-12 text-center">

           <button type="submit" class="btn btn-primary btn-lg">Submit</button>

           <button type="button" class="btn btn-primary btn-lg" id="myButton">Result</button>

         </div>          </div>          </form>

      {% if result %}

      <div id="myPopup" class="popup">

        <div class="popup-content">

        <h5 class="card-title">Recommend Crop for cultivation is:</h5>          <h1 style="color: green">          {{ result }}

      </h1>

      <p>This is a popup box!</p>

   <button id="closePopup">

      Close

   </button></div>

</div>{% endif %}

 <div class="container2">

   <footer class="d-flex flex-wrap justify-content-between align-items-center py-3 my-4 border-top">

     <div class="col-md-4 d-flex align-items-center">

       <a href="/" class="mb-3 me-2 mb-md-0 text-muted text-decoration-none lh-1">

         <svg class="bi" width="30" height="24"><use xlink:href="#bootstrap"/></svg>          </a>

       <span class="text-muted"><strong>&copy; Group 3 of MCA 4th Semester 2022 to 2024.</strong></span>

     </div>   </footer>  </div>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-
C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL" crossorigin="anonymous"></script>

  <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous"></script>

  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>

  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script> <script>
```

```
    // Automatically close the alert after 3 seconds (adjust as needed)
    $(document).ready(function() {          setTimeout(function() {             $(".alert").alert('close');          }, 3000);    });
</script>
<script>
    myButton.addEventListener( "click",          function () {
        myPopup.classList.add("show")          }    );
    closePopup.addEventListener("click",
      function () {
        myPopup.classList.remove(
          "show"       );       }  );
    window.addEventListener(          "click",       function (event) {
        if (event.target == myPopup) {
          myPopup.classList.remove(
            "show"          }       }   );
</script></body></html>
reset_password.html:
{% extends 'base.html' %}
{% block title %} Reset Password {% endblock title %}
{% block head %}
    <link rel="stylesheet" href="{{ url_for('static', filename='login.css') }}">
{% endblock %}{% block content %}
<div class="div2">
    <div class="div1">
      <h1>Reset Password</h1>
      <form method="POST" action="/reset_password">
        <input type="hidden" name="token" value="{{ request.args.get('token') }}">
        <div class="form-group">
          <label for="password">New Password:</label>
          <input type="password" class="form-control" id="password" name="password" placeholder="Enter your new password"
          pattern="^(?=.*[a-zA-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{5,8}$" title="Password must be between 5 and 8
characters long and contain at least one letter, one number, and one special character" required>
        </div>
        <div class="form-group">
          <label for="confirm_password">Confirm Password:</label>
```

```
            <input type="password" class="form-control" id="confirm_password" name="confirm_password" placeholder="Confirm your  new password"

            pattern="^(?=.*[a-zA-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{5,8}$" title="Password must be between 5 and 8 characters long and contain at least one letter, one number, and one special character" required>                                </div>

          <button type="submit">Reset Password</button>              </form>    </div></div>

        <!-- Include any additional JavaScript if needed -->

    {% endblock %}
```

signup.html:

```
{% extends 'base.html' %}

  {% block title %} Signup Page {% endblock title %}

{% block head %}

<link rel="stylesheet" href="{{ url_for('static', filename='signup.css') }}">

{% endblock %}

{% block content %}

<div class="div2">

   <div class="div1">

      <h2>Sign Up</h2>

      <form action="/signup" method="post">

         <input type="hidden" name="csrfmiddlewaretoken" value="your-csrf-token-here"

         <label for="Name">Name</label>

         <input type="text" id="name" name="name" title="Please enter the name" required>

         <label for="email">Email</label>

         <input type="email" id="email" name="email" title="Please enter a valid email address" required

         <label for="password">Password:</label>

         <input type="password" id="password" name="password" pattern="^(?=.*[a-zA-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{5,8}$"

          title="Password must be between 5 and 8 characters long and contain at least one letter, one number, and one special character"  required>

         <button type="submit">Sign Up</button>

      </form>        <br>        <p>Already have an account? <a href="/login">Login here</a>.</p> </div></div>{% endblock %}
```

CSS

About.css

```
body, html {    margin: 0;   padding: 0;   box-sizing: border-box;   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background-color: #de8d0a;   color: #333;}
```

```
.container {   display: flex;   flex-wrap: wrap; /* Allows cards to wrap */   justify-content: center;   align-items: flex-start;   gap: 20px; /*  Adds space between cards */   padding: 20px;}

.profile-card {   background: white;   box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);   display: flex;   flex-direction: row;   width: calc(50% -  20px); /* Adjust width for spacing */   overflow: hidden;   border-radius: 8px;   transition: box-shadow 0.3s ease-in-out;}

.profile-card:hover {    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);}

.photo-wrapper {    flex: 1 1 40%;   display: flex;   justify-content: center;   align-items: center;   padding: 20px;   background: #005f73;}.photo-wrapper img {    width: 250px;   /* max-width: 200px;  */   height: 250px;   border-radius: 50%;   border: 5px solid  #ffffff; transition: transform 0.3s ease-in-out;}

.photo-wrapper img:hover {    transform: scale(1.05);}

.description {    flex: 1 1 60%; padding: 20px;}

.description li {    list-style: none;    }

.description li a {    color: white;/* position: relative; */   padding: 9px 10px;   background: #f58404;   border: 2px solid #fd4766;  transition: all 0.4s ease-in-out;   border-radius: 500px;   text-decoration: none;   }@media (max-width: 768px) {

   .profile-card {        flex-direction: column;         width: auto; /* Full width on small screens */    }

   .description {        text-align: center;    }}

/* Styling for the back button */

.back-button {   display: inline-block;   padding: 7px 15px;   background-color: #007bff;   color: #fff;   text-decoration: none;   border  radius: 5px;   transition: background-color 0.3s ease;}

.back-button:hover {    background-color: #0056b3;}  Contact.css

body {

  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; margin: 0;  padding: 0;  background-color: #f9f9f9;}

.contact-container  {

  max-width: 800px; margin: 50px auto;  background-color: #ffffff;  padding: 40px;  border-radius: 15px;  box-shadow: 0 0 20px rgba(0,  0,0, 0.1);}

.contact-container h1 { margin-bottom: 20px;  color: #333333;  text-align: center;}

.contact-info  {

  display: flex;  flex-wrap: wrap; /* Allow flex items to wrap to the next line */ justify-content: space-between;}

.info-item { flex-basis: calc(33.33% - 20px);text-align: center;  margin-bottom: 20px;}

.info-item i {  font-size: 24px;  margin-bottom: 10px; color: #2c3e50; /* Dark blue */}

.info-item p { color: #555555;}

/* Change the color of location, phone, and email icons */

.info-item .fas.fa-map-marker-alt {  color:

  #27ae60; /* Green */}

.info-item .fas.fa-phone-alt { color: #3498db; /* Blue */}

.info-item .fas.fa-envelope {color: #e74c3c; /* Red */}

.social-media { display: flex;  justify-content: center; margin-top: 30px;}
```

```
.social-media a {  margin: 0 15px;  opacity: 0.7;  transition: opacity 0.3s ease;}

.social-media a:hover {  opacity: 1;}

/* Increase the font size of social media icons */

.social-media i {  font-size: 32px;}

/* Restore original colors for social media icons */

.facebook-color {color: #3b5998; /* Facebook blue */}

.whatsapp-color { color: #25D366; /* WhatsApp green */}

.linkedin-color { color: #0077b5; /* LinkedIn blue */}

.gmail-color {color: #D44638; /* Gmail red */}

.back-button { text-align: center;}

.back-button a { color: #fff; background-color: #333; padding: 10px 20px; text-decoration: none; border-radius: 5px; transition: background-color 0.3s;}.back-button a:hover {  background-color: #555;}

/* Responsive Styles */

@media screen and (max-width: 600px) {

 header {  padding: 10px 0;  }

 h1 {    font-size: 1.5em;  }

 .container {      padding: 10px;  }

 .feature h2 {      font-size: 1.2em;  }

 .feature p {      font-size: 1em; }

 .back-button a {      padding: 8px 15px; }}
```

Fg.css

```
.login-container{

   border-style: solid;   border-color: #4CAF50;   border-radius: 2rem;   padding: 1rem;   width: 300px;   justify-content: center;   align-  items: center;}

label {    margin-bottom: 8px;}

input {    padding: 8px;    margin-bottom: 16px;    border-radius: 2rem;}

button {    background-color: #4CAF50;    color: white;    padding: 10px;    border: none;    cursor: pointer;    border-radius: 2rem;}

.div4{   font-family: Arial, sans-serif; margin: 0;  padding: 0;   display: flex;   justify-content: center;   align-items: center;   height:  100vh;}

.card {

   border-radius: 2rem;   padding: 1rem;}
```

login.css

```
.div2 {   font-family: Arial, sans-serif;   margin: 0;   padding: 0;   display: flex;   justify-content: center; align-items: center;   height:  100vh;}

form {   width: 300px;   display: flex;   flex-direction: column;}

label {  margin-bottom: 8px;}
```

```
input {  padding: 8px;  margin-bottom: 16px;  border-radius: 2rem;}

button {  background-color: #4CAF50;  color: white; padding: 10px;  border: none;  cursor: pointer;  border-radius: 2rem;}

.div1{  border-style: solid;  border-color: #4CAF50;  border-radius: 2rem;  padding: 2rem;  background-color:rgb(250, 249, 248);}  body{

background-image: url('bg.jpg');  overflow: hidden; /* Hide scrollbars */}
```

signup.css

```
.div2 {  font-family: Arial, sans-serif;  margin: 0;  padding: 0;  display: flex;  justify-content: center;  align-items: center;  height:  100vh;  }

form {  width: 300px;  display: flex;  flex-direction: column;}

label {  margin-bottom: 8px;}

input {  padding: 8px;  margin-bottom: 16px;  border-radius: 2rem;}

button {  background-color: #4CAF50;  color: white;  padding: 10px;  border: none;  cursor: pointer;  border-radius: 2rem;}

.div1{  border-style: solid;  border-color: #4CAF50;  border-radius: 2rem;  padding: 2rem;  background-color:rgb(250, 249, 248);
}

body{  background-image: url('bg3.jpg');  overflow: hidden; /* Hide scrollbars */}
```

style.css

```
/* Reset default margin and padding */

body, ul, footer {  margin: 0;  padding: 0;  overflow: hidden; /* Hide scrollbars */}

body{  height: 1px;}

/* Navbar styles */

header {  background-color: #333;  padding: 10px 0; /* Adjust padding */}

nav {  display: flex;  justify-content: space-between;  align-items: center;  padding: 0 20px;  color: #fff;  height: 30px; /* Adjust height */  }

.nav-links {  list-style: none;  display: flex;}.navbar-nav{  width: 150%;  color: rgb(231, 239, 236);  }

.nav-links li {  margin-right: 20px;}

.nav-links li a {  color: #fff;  text-decoration: none;}

/* Logo styles */

.logo img {  height: 55px; /* Adjust logo size */}

/* Main content styles */

main {  padding: 0;}

/* Image slider styles */

.container-fluid {  width: 100vw; /* Make the image container cover the entire viewport width */  height: 89vh; /* Make the image  container
cover the entire viewport height */  overflow: hidden;}

.container-fluid img {  width: 100%;  height: 100%;  object-fit: cover; /* Cover the entire container, maintaining aspect ratio */}  footer

{  background-color: #333;  color: #fff;  padding: 10px 0;  text-align: center;}

footer p {  margin: 0;}
```

App.py

```
from flask import Flask, request, render_template,redirect,url_for,flash
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user, current_user  from
flask_mysqldb import MySQL
from flask_bcrypt import Bcrypt
import uuid  # For generating unique tokens  import
smtplib  # For sending emails
from email.mime.text import MIMEText  # For composing email content
from email.mime.multipart import MIMEMultipart  # For composing email content
import numpy as np
import pandas
import sklearn
import pickle
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)  model =
pickle.load(open('models/RandomForest.pkl','rb'))
app = Flask(__name__)
app.secret_key = "4949skjdjfdjfjfkfkksd"
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = '1234'
app.config['MYSQL_DB'] = 'flask'
mysql = MySQL(app)
bcrypt = Bcrypt(app)
login_manager = LoginManager()
login_manager.init_app(app)  #User
Loader funcation
@login_manager.user_loader
def load_user(email):
   return User.get(email)
class User(UserMixin):
   def __init__(self, user_id,name, email):
      self.id = user_id
      self.name = name
```

```
        self.email = email
    @staticmethod
    def get(user_id):
        cursor = mysql.connection.cursor()
        cursor.execute('select name, email from users where id = %s',(user_id,))  result
        = cursor.fetchone()
        cursor.close()
        if result:
            return User(user_id, result[0],result[1])  @app.route('/') #
THis is decoreater its use for create a url.  def index1():
    return render_template('index1.html')
@app.route('/about.html')
def about():
    return render_template('about.html')
@app.route('/features.html')
def features():
    return render_template('features.html')
@app.route('/contact.html')
def contact():
    return render_template('contact.html')
@app.route('/det.html')
def det():
    return render_template('det.html')
@app.route('/login', methods = ['GET','POST'])  def
login():
    if request.method == 'POST':
        email = request.form['email']  password
        = request.form['password']  cursor =
        mysql.connection.cursor()
        cursor.execute('select id,name,email, password from users where email = %s', (email,))  user_data
        = cursor.fetchone()
        cursor.close()
```

```
        if user_data and bcrypt.check_password_hash(user_data[3], password):
            user = User(user_data[0],user_data[1],user_data[2])
            login_user(user)
            flash('login successfully insert the data and predict the crop', 'success')  return
            render_template('new.html')
        else:
            flash('Invalid email or password. Please try again.', 'error')  return
    render_template('login.html')
@app.route('/signup', methods = ['GET','POST'])
def signup():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        password = request.form['password']
        # Check if email already exists in the database
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT COUNT(*) FROM users WHERE email = %s', (email,))
        count = cursor.fetchone()[0]
        cursor.close()
        if count > 0:
            flash('Email already taken. Please choose a different email.', 'error')
            return redirect(url_for('signup'))
        hashed_password = bcrypt.generate_password_hash(password).decode('utf-8')
        cursor = mysql.connection.cursor()
        cursor.execute('insert into users (name,email,password) values(%s,%s,%s)',(name,email,hashed_password))  mysql.connection.commit()
        cursor.close()
        flash('Account created successfully. You can now login.', 'success')  return
        redirect(url_for('login'))
    return render_template('signup.html')  #
@app.route('/')
# def index():
#     return render_template('index.html')
```

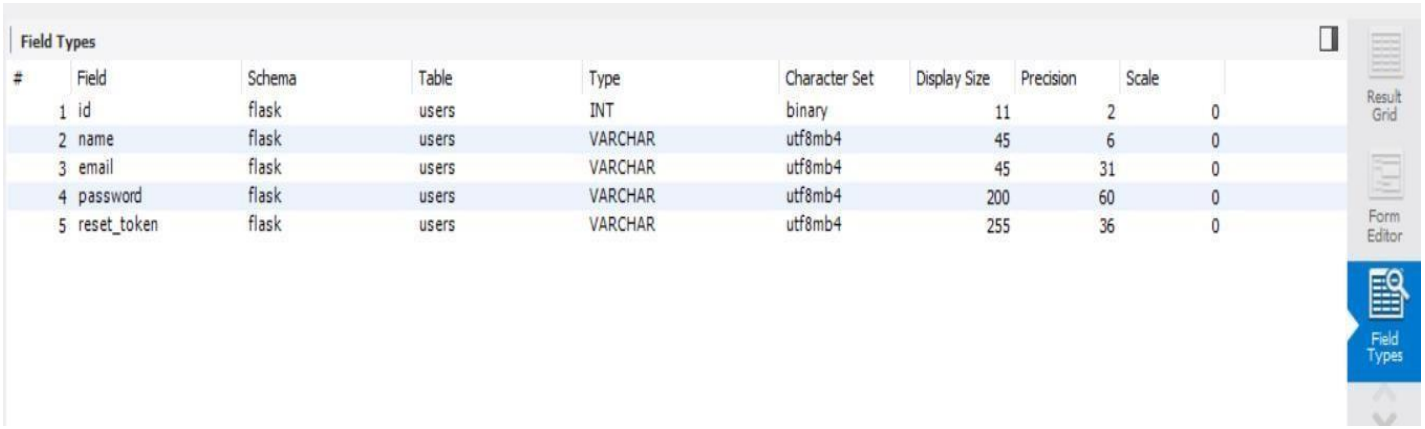
```
@app.route("/predict",methods=['POST'])
@login_required
def predict():
    N = request.form['Nitrogen']  P
    = request.form['Phosporus']  k =
    request.form['Potassium']
    temperature = request.form['Temperature']
    humidity = request.form['Humidity']
    ph = request.form['Ph']
    rainfall = request.form['Rainfall']
    feature_list = [N,P,k,temperature,humidity,ph,rainfall]
    features = np.array([[feature_list]]).reshape(1, -1)
    prediction = model.predict(features)[0]  print(prediction)
    crop_list = ["rice", "maize", "jute", "cotton", "coconut", "papaya", "orange",
          "apple", "muskmelon", "watermelon", "grapes", "mango", "banana",
          "pomegranate", "lentil", "blackgram", "mungbean", "mothbeans",  "pigeonpeas",
          "kidneybeans", "chickpea", "coffee"]
    if prediction in crop_list:
       result = "{} is a best crop to be cultivated. ".format(prediction)  else:
       result = "Sorry, we could not determine the best crop to be cultivated with the provided data."  #
    return render_template('crop.html',result = result)
    flash('Check the result.', 'succes')
    return render_template('new.html',result = result)
@app.route('/logout')
@login_required
def logout():
    logout_user()
    flash('logout successful see you soon.', 'success')
    return redirect(url_for('index1'))
@app.route('/forgot_password', methods=['GET', 'POST'])
def forgot_password():
```

```
    if request.method == 'POST':
        email = request.form['email']
        # Check if the email exists in the database  cursor
        = mysql.connection.cursor()
        cursor.execute('SELECT id FROM users WHERE email = %s', (email,))
        user_id = cursor.fetchone()
        cursor.close()
        if user_id:
            # Generate a unique token for password reset  token
            = str(uuid.uuid4())
            # Save the token in the database
            cursor = mysql.connection.cursor()
            cursor.execute('UPDATE users SET reset_token = %s WHERE id = %s', (token, user_id[0]))
            mysql.connection.commit()
            cursor.close()
            # Send email with password reset link
            send_reset_email(email, token)
            flash('Password reset instructions have been sent to your email.', 'success')  else:
            flash('Email address not found. Please try again.', 'error')
    return render_template('forgot_password.html')
def send_reset_email(email, token):
    # Email sending logic goes here  #
    Example using SMTP  smtp_server
    = 'smtp.gmail.com'  smtp_port =
    587
    sender_email  =  'subhoy2370@gmail.com'
    sender_password  =  'zytp  ymro  kshi  kyxx'
    receiver_email = email
    message = MIMEMultipart()
    message['From'] = sender_email
    message['To'] = receiver_email
    message['Subject'] = 'Password Reset Request'
```

```
    body = f"""
    To reset your password, click the following link:
    http://localhost:5000/reset_password?token={token}  message.attach(MIMEText(body,
    'plain'))
    with smtplib.SMTP(smtp_server, smtp_port) as server:
        server.starttls()
        server.login(sender_email, sender_password)
        server.send_message(message)
@app.route('/reset_password', methods=['GET', 'POST'])
def reset_password():
    if request.method == 'POST':
        # Extract the token and new password from the form submission
        token = request.form['token']
        new_password = request.form['password']
        confirm_password = request.form['confirm_password']
        # Validate the new password and confirm password match  if
        new_password != confirm_password:
            flash('Passwords do not match. Please try again.', 'error')
            return render_template('login.html')
        # Verify the token and update the user's password in the database  else:
          cursor = mysql.connection.cursor()
          cursor.execute('SELECT id FROM users WHERE reset_token = %s', (token,))
          user = cursor.fetchone()
          cursor.close()
        if user:
            # Hash the new password
            hashed_password = bcrypt.generate_password_hash(new_password).decode('utf-8')  #
            Update the user's password and clear the reset token
            cursor = mysql.connection.cursor()
            cursor.execute('UPDATE users SET password = %s, reset_token = NULL WHERE id = %s', (hashed_password, user[0]))  mysql.connection.commit()
            cursor.close()
```

```
            flash("Password reset successful. You can now login with your new password.", "success")  return
            redirect(url_for('login'))
        else:
            flash("Invalid or expired token. Please request a new password reset.", "error")  return
            redirect(url_for('forgot_password'))
        # TODO: Verify the token and update the user's password in the database  #
        Implement your logic to verify the token and update the user's password
    # If the request method is GET, render the reset_password.html template  #
    Pass the token value from the URL query parameters to the template  token =
    request.args.get('token')
    return render_template('reset_password.html', token=token)
@app.after_request
def add_header(response):
    response.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
    response.headers["Pragma"] = "no-cache"
    response.headers["Expires"] = "0"
    return response
@login_manager.unauthorized_handler
def unauthorized():
    flash('You must be logged in to access this page.', 'error')
    return redirect(url_for('login'))
if __name__ == "__main__":
        app.run(debug=True)
```

Database Mysql



#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	id	flask	users	INT	binary	11	2	0
2	name	flask	users	VARCHAR	utf8mb4	45	6	0
3	email	flask	users	VARCHAR	utf8mb4	45	31	0
4	password	flask	users	VARCHAR	utf8mb4	200	60	0
5	reset_token	flask	users	VARCHAR	utf8mb4	255	36	0

Future Scope

1. Precision Agriculture Integration: Crop recommendation systems will seamlessly integrate with precision agriculture technologies, optimizing resource allocation and enhancing yield efficiency.

2. Climate Adaptation: These systems will evolve to provide real-time recommendations tailored to changing climatic conditions, aiding farmers in adapting their crop choices to mitigate climate risks.

3. Data Fusion: Advanced algorithms will fuse multi-source data including satellite imagery, weather forecasts, and soil sensors to provide comprehensive recommendations for optimal crop selection.

4. AI-driven Insights: Future systems will leverage artificial intelligence to analyze historical data, farmer preferences, and market trends, delivering personalized insights for sustainable farming practices.

5. Global Expansion: Crop recommendation systems will extend their reach globally, supporting farmers in emerging economies with tailored advice to improve agricultural productivity and food security.

References

Website: https://www.kaggle.com/

Website: https://www.javatpoint.com/machine-learning

Website: https://www.geeksforgeeks.org/crop-recommendation-system-using-tensorflow/

Website: https://www.researchgate.net/publication/346627389_Crop_Recommendation_System

Certificate

This is to certify that Mr. Siddhartha Khawas of Asansol Engineering College, registration number: 221080510052, has successfully completed a project on “Crop Recommendation System” using Machine Learning with Python under the guidance of Dr. Arnab Chakraborty.

Dr. Arnab Chakraborty

(Asansol Engineering College)

Certificate

This is to certify that Mr. Subham Chakraborty of Asansol Engineering College, registration number: 221080510055, has successfully completed a project on “Crop Recommendation System” using Machine Learning with Python under the guidance of Dr. Arnab Chakraborty.

Dr. Arnab Chakraborty

(Asansol Engineering College)

Certificate

This is to certify that Mr. Sumit Kumar Choubey of Asansol Engineering College, registration number: 221080510059, has successfully completed a project on “Crop Recommendation System” using Machine Learning with Python under the guidance of Dr. Arnab Chakraborty.

Dr. Arnab Chakraborty

(Asansol Engineering College)

Certificate

This is to certify that Mr. Shivendu Shivam of Asansol Engineering College, registration number: 221080510048, has successfully completed a project on “Crop Recommendation System” using Machine Learning with Python under the guidance of Dr. Arnab Chakraborty.

Dr. Arnab Chakraborty

(Asansol Engineering College)