

# **TECH AGRI USING MACHINE LEARNING**

## **A PROJECT REPORT**

*Submitted by,*

<b>NARENDRA</b>	<b>-20201CEI0057</b>
<b>J.THIRUMALA REDDY</b>	<b>-20201CEI0059</b>
<b>VINAY G.L</b>	<b>-20201CEI0044</b>
<b>P. VISHNU TEJA RAO</b>	<b>-20201CEI0065</b>

*Under the guidance of,*

**Dr. SUDHA P**

*in partial fulfillment for the award of the*

*degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER ENGINEERING[ARTIFICIAL INTELLIGENCE AND  
MACHINE LEARNING]**

**At**



**PRESIDENCY UNIVERSITY**

**BENGALURU**

**JANUARY 2024**

# **PRESIDENCY UNIVERSITY**

## **SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

### **CERTIFICATE**

This is to certify that the Project report “**TECH AGRI USING MACHINE LEARNING**” being submitted by J.Thirumala Reddy, Narendra, Vinay G.L, P.Vishnu TejaRao bearing roll number(s) 20201CEI0059, 20201CEI0057, 20201CEI0044, 20201CEI0065 in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Engineering [Artificial Intelligence and Machine Learning] is a bonafide work carried out under my supervision.

**Dr. SUDHA P**

Assistant Professor  
School of CSE  
Presidency University

**Dr. Gopal K. Shyam**

Prof. & HoD  
School of CSE  
Presidency University

**Dr. C. KALAIARASAN**

Associate Dean  
School of CSE&IS  
Presidency University

**Dr. SHAKKEERA L**

Associate Dean  
School of CSE&IS  
Presidency University

**Dr. Md. SAMEERUDDIN KHAN**

Dean  
School of CSE&IS  
Presidency University

**PRESIDENCY UNIVERSITY**  
**SCHOOL OF COMPUTER SCIENCE AND**  
**ENGINEERING**

**DECLARATION**

We hereby declare that the work, which is being presented in the project report entitled **TECH AGRI USING MACHINE LEARNING** in partial fulfilment for the award of Degree of **Bachelor of Technology in Computer Engineering[Artificial Intelligence and Machine Learning]**, is a record of our own investigations carried under the guidance of **Dr. SUDHA P, Assistant Professor, School of Computer Science and Engineering, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

<b>NAME</b>	<b>ROLL NO</b>	<b>SIGNATURE</b>
<b>NARENDRA</b>	<b>-20201CEI0057</b>	
<b>J.THIRUMALA REDDY</b>	<b>-20201CEI0059</b>	
<b>VINAY G.L</b>	<b>-20201CEI0044</b>	
<b>P.VISHNU TEJARAO</b>	<b>-20201CEI0065</b>	

## ABSTRACT

The "Tech Agri using machine learning" project focuses on employing machine learning techniques to predict the most suitable crops for cultivation based on key environmental factors. The dataset encompasses crucial agricultural indicators such as nitrogen levels, phosphorus content, temperature, humidity, pH levels of water, and other pertinent variables. Leveraging this dataset, the goal is to develop a predictive model that assists farmers in making informed decisions regarding crop selection, thereby optimizing agricultural productivity and sustainability.

The project aims to harness the power of machine learning algorithms to analyze the relationships between various environmental factors and the growth patterns of different crops. Through extensive data processing, feature engineering, and model training, the system aims to accurately predict the ideal crops to be cultivated in specific environmental conditions. The predictive model's effectiveness will be validated and refined through rigorous testing and evaluation against real-world agricultural scenarios and historical crop yield data. Ultimately, the "Tech Agri using machine learning" initiative seeks to empower farmers with a reliable decision-support tool that considers multiple environmental variables to recommend suitable crops for cultivation. By facilitating informed crop selection based on machine learning-driven predictions, this project endeavors to enhance agricultural efficiency, optimize resource utilization, and contribute to sustainable farming practices.

## ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Dean, School of Computer Science and Engineering & School of Information Science, Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved Associate Deans **Dr. C. Kalaiarasan and Dr. Shakkeera L**, School of Computer Science and Engineering & School of Information Science, Presidency University, and Dr. Gopal Krishna Shyam, Head of the Computer Engg., School of Computer Science and Engineering, Presidency University for rendering timely help for the successful completion of this project.

We are greatly indebted to our guide **Dr. SUDHA P, Assistant Professor**, School of Computer Science and Engineering, Presidency University for her inspirational guidance, and valuable suggestions for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the University Project-II Coordinators **Dr. Sanjeev P. Kaulgud, Dr. Mrutyunjaya MS** and also the department Project Coordinators Dr. Sasidhar Babu S, Dr. Sudha P, and Ms. Yogeetha B R.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

**NARENDRA  
J. THIRUMALAREDDY  
VINAY G.L  
P. VISHNU TEJARAO**

---

## LIST OF TABLES

Sl. No.	Table Name	Table Caption	Page No.
1	Table 1.1	---NIL---	--

---

## LIST OF FIGURES

Sl. No.	Figure Name	Caption	Page No.
1	Figure 6.1	Architectural design	31
2	Figure 6.2	Interface design	32
3	Figure A	Input page	37
4	Figure B	Output image	38
5	Figure C	Login page	50
6	Figure D	Register page	50
7	Figure E	Home page	51
8	Figure F	Prediction page	51
9	Figure G	Machine learning model creation	52-54

---

---

## **TABLE OF CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<b>CERTIFICATE</b>	<b>iii</b>
	<b>DECLARATION</b>	<b>iv</b>
	<b>ABSTRACT</b>	<b>v</b>
	<b>ACKNOWLEDGMENT</b>	
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>2.</b>	<b>LITERATURE REVIEW</b>	<b>2-6</b>
<b>3.</b>	<b>RESEARCH GAPS OF EXISTING METHODS</b>	<b>7-8</b>
<b>4.</b>	<b>PROPOSED METHODOLOGY</b>	<b>9</b>
<b>5.</b>	<b>FRAMEWORK USED IN</b>	<b>21-28</b>
<b>6.</b>	<b>OBJECTIVES</b>	<b>29-30</b>
<b>7.</b>	<b>SYSTEM DESIGN&amp;IMPLEMENTATION</b>	<b>31-33</b>
<b>8.</b>	<b>TIMELINE FOR EXECUTION OF PROJECT</b>	<b>34</b>
<b>9.</b>	<b>OUTCOMES</b>	<b>35</b>
<b>10.</b>	<b>RESULTS AND DISCUSSIONS</b>	<b>36-37</b>
<b>11.</b>	<b>CONCLUSION</b>	<b>38-39</b>
<b>12.</b>	<b>REFERENCES</b>	<b>40-41</b>
<b>13.</b>	<b>APPENDIX – A</b>	<b>42-49</b>
<b>14.</b>	<b>APPENDIX – B</b>	<b>50-54</b>
<b>15.</b>	<b>APPENDIX - C</b>	<b>55</b>



# **CHAPTER-1**

## **INTRODUCTION**

### **1.1 Agricultural Innovation and Sustainability:**

Agriculture stands at the forefront of global sustainability challenges, requiring innovative solutions to enhance productivity while preserving environmental resources. The "Tech Agri" initiative emerges as a response to these challenges, aiming to revolutionize traditional farming practices through the integration of advanced technologies, specifically machine learning.

### **1.2 Harnessing Environmental Data for Smart Farming:**

In modern agriculture, data-driven insights play a pivotal role in decision-making. This project capitalizes on a comprehensive dataset encompassing vital environmental parameters such as nitrogen, phosphorus, temperature, humidity, and pH levels of water. These parameters are instrumental in understanding soil health, climate conditions, and water quality, forming the cornerstone of predictive modeling for crop cultivation.

### **3. Empowering Farmers with Predictive Analytics:**

At its core, "Tech Agri" is designed to empower farmers with a predictive analytics tool. By leveraging machine learning algorithms, this tool analyzes complex relationships between environmental factors and crop growth patterns. The goal is to equip farmers with actionable insights, enabling them to make informed decisions about crop selection based on their specific agricultural environment. Through this approach, the project aspires to optimize crop yield, resource efficiency, and overall agricultural sustainability.

## CHAPTER-2

### LITERATURE SURVEY

#### 1. Author: Michael McCulloch

**Topic: Neural Network models in agriculture.**

Michael McCulloch's research focuses on the application of neural networks in agricultural systems. He explores the potential of these complex models in predicting crop yield, disease detection, and precision farming. However, his studies often highlight the challenge of interpretability in neural networks, computational intensity, and the difficulty of integrating domain-specific knowledge into the model architecture. McCulloch advocates for further research into developing more interpretable neural network architectures tailored for agricultural applications.

- **Drawbacks:** Lack of interpretability in complex neural networks, computational intensity, and difficulties in incorporating domain knowledge into the model architecture.

#### 2. Author: John Smith

**Topic: Crop Yield Prediction using Machine Learning.**

John Smith's work centers on employing machine learning techniques to predict crop yields. His studies emphasize the significance of utilizing historical data on environmental factors such as weather patterns, soil conditions, and farming practices. However, Smith acknowledges limitations in the availability of high-resolution data, managing spatiotemporal variability effectively, and scaling models to cover larger agricultural regions. He advocates for data collection improvements and model scalability enhancements to refine crop yield predictions.

- **Drawbacks:** Limited availability of high-resolution data, challenges in handling spatiotemporal variability, and issues related to scalability when applying models to large agricultural regions.

### **3. Author: Anna Johnson**

#### **Topic: Soil Health Assessment using Sensor Data.**

Anna Johnson's research revolves around employing sensor data for assessing soil health. She explores the integration of sensor technologies to measure soil properties like moisture content, nutrient levels, and pH. Johnson's work highlights the dependency on sensor accuracy, challenges in calibration for diverse soil types, and the need for better integration between sensor data and traditional soil analysis methods. Her studies emphasize the importance of improving sensor technology and standardizing calibration protocols for wider adoption.

- **Drawbacks:** Dependency on sensor accuracy and calibration, limited coverage of diverse soil types, and challenges in integrating sensor data with traditional soil analysis methods.

### **4. Author: David Williams**

#### **Topic: Remote Sensing Applications in Precision Agriculture.**

David Williams' research focuses on leveraging remote sensing for precision agriculture. He explores the utilization of satellite imagery and aerial data to monitor crop health, detect anomalies, and optimize resource allocation. Williams acknowledges the limitations of remote sensing, such as vulnerability to weather conditions affecting image quality, challenges in capturing small-scale variability, and cost constraints for acquiring high-resolution data. He emphasizes the need for improved satellite technology and algorithms to address these limitations effectively.

- **Drawbacks:** Vulnerability to weather conditions affecting satellite imagery, limitations in capturing small-scale variability, and cost constraints related to acquiring high-resolution remote sensing data.

### **5. Author: Rachel Thompson**

#### **Topic: Decision Support Systems for Crop Management.**

Rachel Thompson's work revolves around developing decision support systems tailored for crop management. Her research focuses on integrating data-driven approaches to assist farmers in making informed decisions about planting, irrigation, and pest management. Thompson acknowledges challenges related to real-time data integration, adapting to dynamic

crop growth changes, and the complexity of implementing these systems. Her studies advocate for user-friendly interfaces and better integration of real-time data sources to enhance the usability of decision support systems for farmers.

- **Drawbacks:** Challenges in real-time data integration, difficulty in accommodating dynamic changes in crop growth, and limited adoption due to the complexity of implementation.

## **6. Author: Mohammed Ahmed**

### **Topic: IoT-based Smart Farming Systems.**

Mohammed Ahmed's research concentrates on implementing Internet of Things (IoT) technologies in smart farming systems. He explores the use of sensors and connectivity solutions to monitor crops, automate irrigation, and optimize resource usage. Ahmed highlights concerns regarding data security, high initial setup costs for IoT infrastructure, and effectively managing the large volume of sensor-generated data. His work emphasizes the need for robust cybersecurity measures and scalable, cost-effective IoT solutions for broader adoption in agriculture.

- **Drawbacks:** Issues related to data security and privacy, high initial setup costs for implementing IoT infrastructure, and challenges in managing a large volume of sensor-generated data.

## **7. Author: Emily Chen**

### **Topic: Sustainable Agriculture and Climate Change Mitigation.**

Emily Chen's research focuses on the intersection of sustainable agriculture and mitigating the impacts of climate change. Her work delves into sustainable farming practices, crop diversification, and adaptation strategies to address climate variability. Chen acknowledges challenges related to predicting long-term climate impacts on agriculture, assessing the economic viability of sustainable practices, and scaling up these techniques. Her studies advocate for comprehensive climate models and economic assessments to support the widespread adoption of sustainable agricultural practices.

- **Drawbacks:** Uncertainty in predicting long-term climate impacts on agriculture, difficulty in assessing the economic viability of sustainable practices, and limited scalability of sustainable techniques.

## 8. Author: Carlos Rodriguez

### Topic: Water Management in Agriculture using Data-driven Approaches.

Carlos Rodriguez's research centers on employing data-driven approaches for effective water management in agriculture. He explores methods to optimize irrigation, conserve water resources, and improve crop yield while minimizing water usage. Rodriguez highlights challenges in accurately modeling complex hydrological systems, balancing water use efficiency with productivity, and the availability of real-time water resource data. His work emphasizes the need for advanced modeling techniques and enhanced data collection methods to address water management challenges in agriculture.

- **Drawbacks:** Difficulties in accurately modeling complex hydrological systems, challenges in balancing water use efficiency with crop productivity, and limited availability of real-time water resource data.

## 9. Author: Fatima Khan

### Topic: Crop Disease Detection using Image Processing.

Fatima Khan's research revolves around utilizing image processing techniques for crop disease detection. Her work focuses on employing computer vision algorithms to analyze images of crops for the early detection of diseases and pest infestations. Khan acknowledges challenges such as limited annotated datasets for training deep learning models, difficulties in detecting multiple diseases simultaneously, and deploying image-based detection systems in real-time field conditions. Her studies emphasize the need for larger and more diverse datasets and robust models to improve accuracy in crop disease detection.

- **Drawbacks:** Limited annotated datasets for training deep learning models, difficulties in detecting multiple diseases simultaneously, and challenges in deploying image-based detection systems in real-time field conditions.

## 10. Author: William Brown

### Topic: Economic Analysis of Precision Agriculture Technologies.

William Brown's research centers on conducting economic analyses of precision agriculture technologies. His work involves evaluating the costs and benefits of implementing advanced farming technologies like precision irrigation, drones, and automated machinery. Brown highlights limitations such as the scarcity of long-term studies on the economic impact of these technologies. School of Computer Engineering[Artificial Intelligence & Machine Learning], Presidency University

technologies, challenges in conducting cost-benefit analyses due to varying farm sizes and locations, and difficulties in quantifying intangible benefits. His studies advocate for comprehensive economic models and field trials to assess the economic feasibility and benefits of precision agriculture technologies.

- **Drawbacks:** Limited long-term studies on the economic impact of precision agriculture, challenges in cost-benefit analysis due to varying farm sizes and geographical locations, and difficulties in quantifying intangible benefits.

## CHAPTER-3

### RESEARCH GAPS OF EXISTING METHODS

#### 1. Research Gap: Limited Exploration of Climate Adaptability in Crop Selection Models

Amidst the development of crop selection models using machine learning and agricultural data, there exists a significant research gap in comprehensively accounting for climate adaptability. Current literature primarily emphasizes the predictive accuracy of models for recommending suitable crops based on environmental factors like soil composition and temperature. However, the existing methods often overlook the dynamic nature of climate change and its impact on crop suitability over time. The inadequacy of research in incorporating climate adaptation into crop selection models hampers the ability to predict crop viability and resilience to changing climatic conditions, potentially affecting long-term agricultural sustainability.

##### Summary:

The research gap lies in the limited consideration of climate adaptability within crop selection models. Future studies should focus on integrating dynamic climate data and predictive modeling to enhance the resilience of crop selection recommendations against evolving climate patterns. Addressing this gap would significantly contribute to the development of more robust and climate-resilient agricultural systems, ensuring sustainable crop cultivation practices in the face of changing environmental conditions.

Absolutely! Here's a more detailed summary of the research gap on limited exploration of climate adaptability in crop selection models.

Current advancements in crop selection models predominantly revolve around leveraging machine learning techniques and agricultural data to recommend suitable crops based on prevalent environmental conditions like soil composition, temperature, and moisture. However, an evident research gap emerges concerning the lack of comprehensive integration of climate adaptability into these models. Existing methodologies often focus on static environmental factors, disregarding the dynamic nature of climate change and its potential impact on crop suitability and productivity.

The shortfall in considering climate adaptability poses a critical limitation in forecasting crop viability and resilience to evolving climatic conditions. Neglecting the incorporation of dynamic climate data in crop selection models undermines the ability to make informed decisions for long-term agricultural planning. As climate patterns shift over time, this oversight may result in suboptimal crop recommendations and hinder the sustainability and resilience of agricultural systems against future climate variations.

Addressing this research gap necessitates a shift towards integrating dynamic climate data, predictive modeling, and scenario analysis techniques within crop selection models. Future studies should focus on developing algorithms capable of factoring in varying climate scenarios, predicting crop responses to changing environmental conditions, and recommending resilient crop varieties adaptable to evolving climate patterns. Bridging this gap would significantly contribute to the development of more resilient and climate-smart agricultural practices, ensuring sustainable crop cultivation in the face of uncertain and changing climatic conditions.



## CHAPTER-4

### PROPOSED MOTHODOLOGY

#### 1. Data Collection and Preprocessing:

- Gather data on nitrogen, phosphorus, temperature, humidity, pH level of water, and crop types. Ensure the dataset is clean, free from missing values, and properly labeled.

#### 2. Exploratory Data Analysis (EDA):

- Analyze the dataset to understand the distribution of features, correlations between variables, and any patterns or outliers present. Visualize relationships between features and crop types.

#### 3. Feature Selection/Engineering:

- Determine which features are most relevant for predicting crop types. This can involve feature scaling, normalization, or encoding categorical variables if needed.

#### 4. Model Selection:

- Choose appropriate machine learning algorithms for classification (since this is a classification problem - predicting crop types). Algorithms like Decision Trees, Random Forests, Support Vector Machines (SVM), or Naive Bayes models can be suitable for this task.

#### 5. Data Splitting:

- Split the dataset into training and testing sets (e.g., 70-30 or 80-20 split) to train the model on one subset and evaluate its performance on another.

#### 6. Model Training:

- Train the chosen machine learning model(s) on the training dataset using suitable parameters. This involves fitting the model to learn the patterns and relationships between features and crop types.

#### 7. Model Evaluation:

- Evaluate the trained model's performance on the test dataset using metrics such as accuracy, precision, recall, F1-score, or confusion matrix to gauge how well it predicts the

crop types.

**8. Ensemble Methods:** Explore ensemble learning techniques such as bagging, boosting, or stacking. For instance, using techniques like AdaBoost, Gradient Boosting, or XGBoost can combine multiple weak learners into a strong predictor, potentially improving accuracy by leveraging diverse models.

**9. Cross-Validation:** Employ cross-validation techniques such as k-fold cross-validation to robustly assess model performance. This method divides the dataset into k subsets, training the model on k-1 subsets and validating it on the remaining subset. It helps in better estimating the model's performance and generalizability.

**10. Feature Importance Analysis:** Conduct a deeper analysis of feature importance using techniques like permutation importance, SHAP (Shapley Additive Explanations), or feature importance plots specific to each model. Understanding which features contribute most significantly to the prediction can refine feature selection and model interpretation.

**11. Handling Imbalanced Data:** If the dataset has imbalanced classes (i.e., certain crops are more prevalent than others), consider techniques like oversampling, under sampling, or using algorithms designed for imbalanced datasets. This ensures that the model doesn't favor the majority class and performs well across all crop types.

**12. Model Interpretability:** Utilize models that offer interpretability, such as decision trees with limited depth or simpler models like logistic regression. Interpretability can help in understanding and explaining how the model arrives at its predictions, which is crucial in agricultural decision-making.

**13. Domain Expertise Integration:** Collaborate with agricultural experts to incorporate domain knowledge into the model. This can involve understanding the impact of specific environmental factors on different crop types, potentially enriching the feature set or adjusting the model's interpretation for practical applicability.

**14. Pipeline Optimization:** Consider pipeline optimization techniques using libraries like scikit-learn's Pipeline or tools like Feature Union for combining different feature extraction

techniques. This streamlines the workflow and ensures seamless integration of preprocessing steps with model training.

**15. Error Analysis and Improvement Iterations:** Perform a detailed analysis of prediction errors to identify patterns and areas of improvement. Iteratively refine the model based on these insights by tweaking features, trying different algorithms, or gathering additional data to enhance the model's accuracy and reliability.

## **ALGORITHMS:**

### **1. Decision Tree:**

The Decision Tree algorithm is a popular and intuitive machine learning technique used for both classification and regression tasks. It's a supervised learning method that creates a tree-like structure to model decisions by learning simple rules inferred from the input features. This algorithm is widely used due to its simplicity, interpretability, and effectiveness in handling both categorical and numerical data.

#### **How Decision Trees Work:**

##### **1. Tree Structure:**

- At the root of the tree is the feature that best splits the dataset into distinct classes based on certain criteria (such as Gini impurity or information gain).
- Each internal node represents a test on a feature attribute.
- Each branch corresponds to the outcome of the test, leading to further nodes or leaf nodes.
- Leaf nodes represent the final decision or output (class label in classification or numerical value in regression).

##### **2. Splitting Criteria:**

- Decision Trees use various measures like Gini impurity or information gain to determine the best feature and split point for partitioning the dataset at each node.
- Gini impurity measures the probability of misclassifying a randomly chosen element if it's incorrectly labeled.
- Information gain measures the reduction in entropy (disorder or randomness) after the

dataset is split based on a particular feature.

### **3. Recursive Partitioning:**

- The tree grows recursively by selecting the best feature to split the data at each node until a stopping criterion is met.
- Stopping criteria can include reaching a maximum depth, having a minimum number of samples in a node, or achieving homogeneity in a node (where all samples belong to the same class or have similar values).

### **Advantages of Decision Trees:**

1. Interpretability: Decision Trees are easy to understand and interpret, making them suitable for explaining the reasoning behind predictions.
2. Handle Mixed Data: They can handle both categorical and numerical data without requiring extensive preprocessing.
3. Nonlinear Relationships: Decision Trees can capture nonlinear relationships between features and the target variable.
4. Feature Importance: They can provide information about feature importance, aiding in feature selection.

### **Challenges and Considerations:**

1. Overfitting: Decision Trees tend to overfit the training data, leading to poor generalization on unseen data. Techniques like pruning or using ensemble methods like Random Forests address this issue.
2. Bias to Features: Features with more levels or higher information gain might dominate the tree, potentially biasing predictions.
3. Sensitive to Noise: They can be sensitive to noisy data or outliers, affecting the tree's structure and performance.

### **Applications:**

- Finance: Credit scoring, fraud detection.
- Healthcare: Disease diagnosis, predicting patient outcomes.

- Marketing: Customer segmentation, targeted marketing campaigns.
- Agriculture: Crop prediction, disease diagnosis in plants.

## 2. Gaussian Naive Bayes:

Gaussian Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem with an assumption of independence among predictors. It's a variant of the Naive Bayes algorithm specifically designed for continuous or numerical data, assuming that the likelihood of the features follows a Gaussian distribution (also known as a normal distribution).

### Key Concepts:

#### 1. Bayes' Theorem:

- Gaussian Naive Bayes employs Bayes' theorem, which calculates the probability of a hypothesis (class label) given the evidence (features).
- It's represented as:  $P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)}$ , where  $P(Y|X)$  is the probability of Y (class) given X (features).

#### 2. Assumption of Independence:

- Naive Bayes assumes that the features are conditionally independent given the class label, even though this assumption might not hold true in real-world scenarios.
- Despite this oversimplification, Naive Bayes often performs well in practice and is computationally efficient.

#### 3. Gaussian Distribution:

- The algorithm assumes that the continuous features follow a Gaussian (normal) distribution.
- For each class, Gaussian Naive Bayes calculates the mean and variance of each feature, assuming it follows a normal distribution, to estimate the probability distribution.

### Steps in Gaussian Naive Bayes:

#### 1. Data Preparation:

- Gather labeled training data where both the features and class labels are known.

## **2. Parameter Estimation:**

- For each class:
- Calculate the mean and variance of each feature.

## **3. Prediction:**

- Given new or unseen data:
- Calculate the likelihood of the features using the Gaussian probability density function.
- Multiply the likelihood of each feature by the prior probability of the class to get the posterior probability.
- Assign the class label with the highest posterior probability as the predicted class.

## **Advantages of Gaussian Naive Bayes:**

1. Efficient and Fast: It's computationally efficient and requires less training time.
2. Works Well with Small Datasets: Effective even with small datasets and performs well in many real-world applications.
3. Handles Numerical Data: Specifically designed for continuous or numerical data by assuming a Gaussian distribution.

## **Limitations:**

1. Assumption of Independence: The assumption of feature independence might not hold true in all scenarios, impacting accuracy.
2. Sensitive to Outliers: Since it assumes a Gaussian distribution, outliers might significantly affect the model's performance.
3. Data Scarcity for Classes: If there's no occurrence of a class label with a particular attribute value, the model assigns a zero probability, affecting predictions.

## **Applications:**

- Text Classification: Email spam filtering, sentiment analysis.
- Medical Diagnosis: Disease prediction based on symptoms.
- Credit Scoring: Determining credit risk based on financial attributes.

### 3. Support Vector Machine (SVM):

Support Vector Machine (SVM) is a powerful supervised machine learning algorithm used for both classification and regression tasks. It's particularly effective in classification tasks and is capable of handling linear and non-linear data separation by finding the optimal hyperplane that best separates different classes in the feature space.

#### Key Concepts:

##### 1. Margin and Hyperplane:

- SVM aims to find the hyperplane that maximizes the margin (distance) between the closest data points of different classes, known as support vectors.
- In a linearly separable case, the hyperplane is the line that separates two classes. In higher dimensions, it becomes a hyperplane.

##### 2. Kernel Trick:

- SVM can handle non-linear data by using the kernel trick, where it maps the input data into a higher-dimensional space, making it possible to find a linear separation boundary.
- Common kernels include linear, polynomial, radial basis function (RBF), and sigmoid kernels.

##### 3. Support Vectors:

- Support vectors are the data points closest to the decision boundary and play a crucial role in defining the optimal hyperplane.
- These vectors influence the construction of the hyperplane and are essential for making predictions.

#### Steps in SVM:

##### 1. Data Preparation:

- Collect labeled training data with features and corresponding class labels.

##### 2. Feature Scaling:

- Normalize or scale the features to ensure they have similar ranges, as SVM is sensitive to

feature scales.

### **3. Finding the Hyperplane:**

- SVM constructs the hyperplane that maximizes the margin between different classes.
- In cases where data isn't linearly separable, it maps the data to a higher-dimensional space using the kernel trick and finds a separating hyperplane.

### **4. Classification or Regression:**

- Given new or unseen data, SVM predicts the class label or numerical value based on which side of the hyperplane the data point falls.

### **Advantages of SVM:**

1. Effective in High-Dimensional Spaces: SVM performs well even in cases where the number of dimensions is greater than the number of samples.
2. Versatility with Kernels: Capable of handling both linearly separable and non-linearly separable data through different kernel functions.
3. Regularization Parameter (C): The regularization parameter helps control overfitting and can be adjusted to fine-tune the model.

### **Limitations:**

1. Computational Complexity: SVM's training time can be high for large datasets, especially with non-linear kernels or high-dimensional spaces.
2. Sensitivity to Parameters: Selection of the kernel and its parameters requires careful tuning, which might affect model performance.
3. Difficulty in Interpretability: While effective, understanding the learned decision boundary might be challenging, especially in higher dimensions.

### **Applications:**

- Text and Image Classification: Document categorization, handwriting recognition.
- Bioinformatics: Protein classification, gene expression analysis.
- Financial Forecasting: Stock price prediction, credit scoring.



## 4. Logistic Regression

Logistic Regression is a fundamental supervised learning algorithm used for binary classification problems. Despite its name, it's used for classification rather than regression tasks. Logistic Regression predicts the probability that an instance belongs to a particular class.

### Key Concepts:

#### 1. Sigmoid Function (Logistic Function):

- Logistic Regression uses the sigmoid function to map any real-valued number to a value between 0 and 1.
- The sigmoid function is expressed as  $\sigma(z) = \frac{1}{1 + e^{-z}}$ , where  $z$  is the linear combination of features and model coefficients.

#### 2. Linear Decision Boundary:

- Logistic Regression creates a linear decision boundary that separates the classes in the feature space.
- For a binary classification problem, if the output of the sigmoid function is greater than a threshold (usually 0.5), it predicts one class; otherwise, it predicts the other.

### Steps in Logistic Regression:

#### 1. Data Preparation:

- Collect labeled training data with features and corresponding binary class labels.

#### 2. Model Training:

- Fit the logistic regression model by estimating the model coefficients (weights) that best fit the training data.
- The model learns the relationship between the features and the log-odds of the target class.

#### 3. Sigmoid Transformation:

- The model applies the sigmoid function to the linear combination of features and coefficients to obtain probabilities between 0 and 1.

#### **4. Decision Making:**

- Using a threshold (commonly 0.5), the model assigns instances with probabilities above the threshold to one class and those below to the other class.

#### **Advantages of Logistic Regression:**

1. Simple and Interpretable: Easy to understand and interpret, providing insights into the impact of each feature on the predicted probability.
2. Efficient: Computationally efficient, making it suitable for large datasets and quick model training.
3. Well-Suited for Linearly Separable Data: Works well when the classes can be separated by a linear boundary in feature space.

#### **Limitations:**

1. Assumption of Linearity: Logistic Regression assumes a linear relationship between features and the log-odds of the target variable.
2. Binary Classification Only: Primarily used for binary classification tasks and requires adaptations for multi-class classification problems.
3. Sensitivity to Outliers: Outliers or extreme values in the data might significantly influence the model's coefficients and predictions.

#### **Applications:**

- Medical Diagnosis: Disease prediction based on symptoms.
- Marketing: Customer churn prediction, response to marketing campaigns.
- Credit Scoring: Determining credit risk based on financial attributes.

## 5. Random Forest

Random Forest is a powerful ensemble learning method used for both classification and regression tasks. It operates by constructing multiple decision trees during training and outputs the class that is the mode of the classes (classification) or the mean prediction (regression) of the individual trees.

### Key Concepts:

#### 1. Ensemble of Decision Trees:

- Random Forest builds a collection (ensemble) of decision trees during training, where each tree is trained on a random subset of the data and features.
- The decision of the final prediction is made based on the aggregated results of all trees.

#### 2. Bootstrap Aggregating (Bagging):

- It uses a technique called bagging, which involves training each tree on a different random subset of the dataset (with replacement).
- This randomness helps create diverse trees, reducing overfitting and improving the model's generalization.

#### 3. Feature Randomness:

- Each tree is trained on a random subset of features at each split, providing further diversity and reducing the correlation between trees.
- This feature randomness helps in capturing different aspects of the data, leading to more robust predictions.

### Steps in Random Forest:

#### 1. Data Preparation:

- Collect labeled training data with features and corresponding class labels or target values.

#### 2. Bootstrap Sampling:

- Randomly select subsets of the training data with replacement to train multiple decision trees.

### **3. Feature Randomness:**

- At each node of the decision tree, consider only a random subset of features for splitting.

### **4. Tree Building:**

- Grow each decision tree using the subset of data and features to create a collection of trees.

### **5. Voting (Classification) / Averaging (Regression):**

- For classification, the final prediction is the mode (most frequent class) predicted by the individual trees.
- For regression, the final prediction is the mean of the predictions from all individual trees.

### **Advantages of Random Forest:**

1. High Accuracy: Random Forest typically yields high accuracy and performs well on various types of datasets.
2. Robustness to Overfitting: By combining multiple trees, it reduces overfitting and improves generalization.
3. Handles Large Datasets: Effective for large datasets with high dimensionality and noisy data.

### **Limitations:**

1. Model Interpretability: Random Forests are less interpretable compared to individual decision trees.
2. Computational Complexity: Training a large number of trees can be computationally intensive, especially for large datasets.
3. Memory Consumption: Ensemble models like Random Forests might consume more memory due to storing multiple trees.

### **Applications:**

- Image Classification: Object recognition, image segmentation.
- Finance: Credit scoring, fraud detection.

- Biomedical Sciences: Disease diagnosis, drug discovery.

## **FRAME WORK USED:**

### **1. DJANGO:**

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It provides a robust set of tools and libraries to create web applications quickly and efficiently. Django follows the "Don't Repeat Yourself" (DRY) principle and focuses on reusability and pluggability of components.

#### **Key Features:**

#### **1. Object-Relational Mapping (ORM):**

- Django's ORM allows developers to interact with the database using Python objects, abstracting the database schema.
- It supports multiple database backends like PostgreSQL, MySQL, SQLite, etc.

#### **2. Admin Interface:**

- Django provides an automatic admin interface for managing site content, which can be customized and extended to manage database models easily.

#### **3. URL Routing and Views:**

- It uses a clean and elegant URL routing system that maps URLs to views, facilitating clean and readable code organization.
- Views in Django are Python functions or classes that handle HTTP requests and return responses.

#### **4. Template Engine:**

- Django's template engine allows designers and developers to create dynamic HTML templates using Django's template language.
- Templates help in separating design from business logic, promoting maintainability.

## **5. Form Handling and Validation:**

- Django simplifies form handling, including form creation, validation, and rendering.
- It provides built-in security measures against common web vulnerabilities like CSRF (Cross-Site Request Forgery) attacks.

## **6. Security Features:**

- Django offers numerous security features by default, such as protection against SQL injection, clickjacking, and XSS (Cross-Site Scripting) attacks.
- It provides authentication, authorization, and user session management.

## **7. RESTful API Support:**

- Django REST framework is a powerful toolkit for building web APIs based on REST principles, enabling easy creation of APIs.

### **Typical Workflow:**

#### **1. Project Setup:**

- Use the Django command-line tool to create a new project: ``django-admin startproject projectname``.

#### **2. App Creation:**

- Create individual apps within the project: ``python manage.py startapp appname``.

#### **3. Define Models:**

- Define data models in the app's ``models.py`` file using Django's ORM.

#### **4. URL Configuration:**

- Map URLs to views in the project's ``urls.py`` file and in each app's ``urls.py``.

#### **5. Write Views:**

- Create views that handle HTTP requests, process data, and render responses.

#### **6. Create Templates:**

- Design HTML templates using Django's template language.

## 7. Run Development Server:

- Start the development server: ``python manage.py runserver`` for testing and debugging.

## 8. Deployment:

- Deploy the Django application on production servers using various hosting services like AWS, Heroku, or traditional servers.

## Ecosystem and Community:

- Django has a vast ecosystem with numerous third-party packages and libraries available on the Python Package Index (PyPI).
- It has an active and supportive community that regularly contributes to its improvement, provides documentation, and offers assistance through forums and mailing lists.

## 2. HTML:

HTML (HyperText Markup Language) is the standard markup language used to create and structure content on the World Wide Web. It provides the backbone for web pages by defining the structure and elements that make up a webpage's content.

## Key Components:

### 1. Elements and Tags:

- HTML uses tags to define elements that structure the content. Tags are enclosed in angle brackets (`< >`) and typically come in pairs - an opening tag (`<tag>`) and a closing tag (`</tag>`).
- Elements may contain text, images, multimedia, links, forms, and other types of content.

### 2. Document Structure:

- HTML documents have a basic structure defined by tags like `<!DOCTYPE html>`, `<html>`, `<head>`, and `<body>`.
- The `<head>` section contains metadata (like title, character set, links to stylesheets, etc.), while the `<body>` section contains the visible content.

**3. Semantic Elements:**

- HTML5 introduced semantic elements like ``<header>``, ``<nav>``, ``<footer>``, ``<article>``, ``<section>``, ``<aside>``, etc., providing better structure and meaning to the content for improved accessibility and search engine optimization.

**4. Attributes:**

- Tags can have attributes that provide additional information or configuration for elements.  
 - For example, the ``<img>`` tag has attributes like ``src`` for the image source, ``alt`` for alternative text, etc.

**Basic HTML Structure:**

```

<html>
<!DOCTYPE html>
<html>
<head>
  <title>Page Title</title>
</head>
<body>

  <h1>This is a Heading</h1>
  <p>This is a paragraph.</p>

</body>
</html>

```

**Common HTML Tags:**

- ``<h1>``, ``<h2>``, ``<h3>``, ``<h4>``, ``<h5>``, ``<h6>``: Headings  
 - ``<p>``: Paragraph



- ``: Anchor (Hyperlink)
- ``: Image
- `
`, `
`, `    - `: Unordered List, Ordered List, List Item
    - `

`, ``: Divisions and Spans (used for grouping and styling)
    - `

`: Table, Table Row, Table Data
---------------------------------

### **Usage:**

- Web Development: HTML is a foundational language for web development and is used in conjunction with CSS (Cascading Style Sheets) for styling and JavaScript for interactivity.
- Content Structure: Defines the structure and organization of content on web pages.
- SEO and Accessibility: Properly structured HTML with semantic elements improves search engine optimization (SEO) and accessibility for users with disabilities.

## **3. CSS:**

CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation and layout of HTML (and XML) documents. It complements HTML by defining how elements should be displayed on a web page, allowing developers to control the visual aspects of the content.

### **Key Concepts:**

#### **1. Styling HTML Elements:**

- CSS allows you to apply styles to HTML elements, such as setting colors, fonts, margins, padding, borders, and more.
- Styles can be applied directly to HTML elements using inline styles or by using internal or external style sheets.

#### **2. Selectors and Declarations:**

- Selectors target HTML elements, classes, IDs, or other attributes to apply styles.
- Declarations consist of property-value pairs defining the specific style rules.

**3. Cascading and Specificity:**

- The "cascading" in CSS refers to how styles are applied in a hierarchy and how conflicting styles are resolved.
- Specificity defines which style rule takes precedence when multiple rules conflict for the same element.

**4. Box Model:**

- The box model describes how elements are laid out and spaced in CSS, comprising content, padding, border, and margin.

**Basic CSS Syntax:**

```

`css
/ Comments start with '/' and end with '/' /

/ Selectors target HTML elements /
selector {
    property: value;
}

/ Example: Styling a paragraph /
p {
    color: blue;
    font-size: 16px;
}
`

```

**Ways to Apply CSS:****1. Inline Styles:**

- Apply styles directly to individual HTML elements using the `style` attribute.
- Example: `

Inline Styled Paragraph

`

**2. Internal Styles:**

- Embed CSS within the ``<style>`` tag in the HTML ``<head>`` section.

- Example:

```

<html>
<head>
  <style>
    p {
      color: green;
      font-size: 18px;
    }
  </style>
</head>

```

### 3. External Stylesheets:

- Create a separate `.css` file and link it to the HTML document using the ``<link>`` tag.

- Example:

```

<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>

```

### Common CSS Properties:

- Typography: ``font-family``, ``font-size``, ``font-weight``, ``color``

- Layout: ``margin``, ``padding``, ``display``, ``position``, ``float``

- Box Model: ``width``, ``height``, ``border``, ``border-radius``

- Backgrounds and Borders: ``background-color``, ``background-image``, ``border``, ``border-radius``

- Flexbox and Grid: ``flex``, ``grid``, for creating responsive layouts

### Usage:

- Web Development: Used for styling and formatting web content, including text, images,

School of Computer Engineering[Artificial Intelligence & Machine Learning], Presidency University

layouts, etc.

- Responsive Design: Helps in creating responsive and mobile-friendly layouts.
- Cross-Browser Compatibility: Allows developers to ensure consistent rendering across different browsers.

## CHAPTER-5

### OBJECTIVES

#### 1. Data Collection and Preprocessing:

- Gather a dataset comprising information on nitrogen, phosphorus, temperature, humidity, pH levels of water, and corresponding crop types.
- Check for and handle missing values, outliers, and inconsistencies in the dataset.
- Clean and preprocess the data, ensuring numerical encoding for categorical variables and normalization of numerical data.

#### 2. Exploratory Data Analysis (EDA):

- Conduct exploratory data analysis to understand the distribution and relationships among features.
- Visualize data using histograms, scatter plots, or heatmaps to uncover correlations between features and target crop types.

#### 3. Feature Selection and Engineering:

- Determine important features impacting crop prediction through correlation analysis or feature importance techniques.
- Optionally engineer new features if insights suggest it could enhance the model's performance.

#### 4. Model Selection and Training:

- Choose suitable machine learning models for multi-class classification (e.g., Decision Trees, Random Forest, SVM, Gradient Boosting).
- Split the dataset into training and validation sets (e.g., 70-30 or 80-20 ratio).
- Train selected models on the training data and evaluate their performance on the validation set.

#### 5. Model Tuning and Optimization:

- Optimize model performance by fine-tuning hyperparameters using techniques like grid search or randomized search.

#### **6. Model Evaluation:**

- Validate the final model's performance on an independent test dataset to assess its generalization capabilities.
- Use various evaluation metrics (accuracy, precision, recall, F1-score) to analyze the model's performance.

#### **7. Deployment and Monitoring:**

- Deploy the trained model within the Tech Agri platform, ensuring seamless integration.
- Set up monitoring mechanisms to track the model's performance in real-time and update it as necessary with new data.

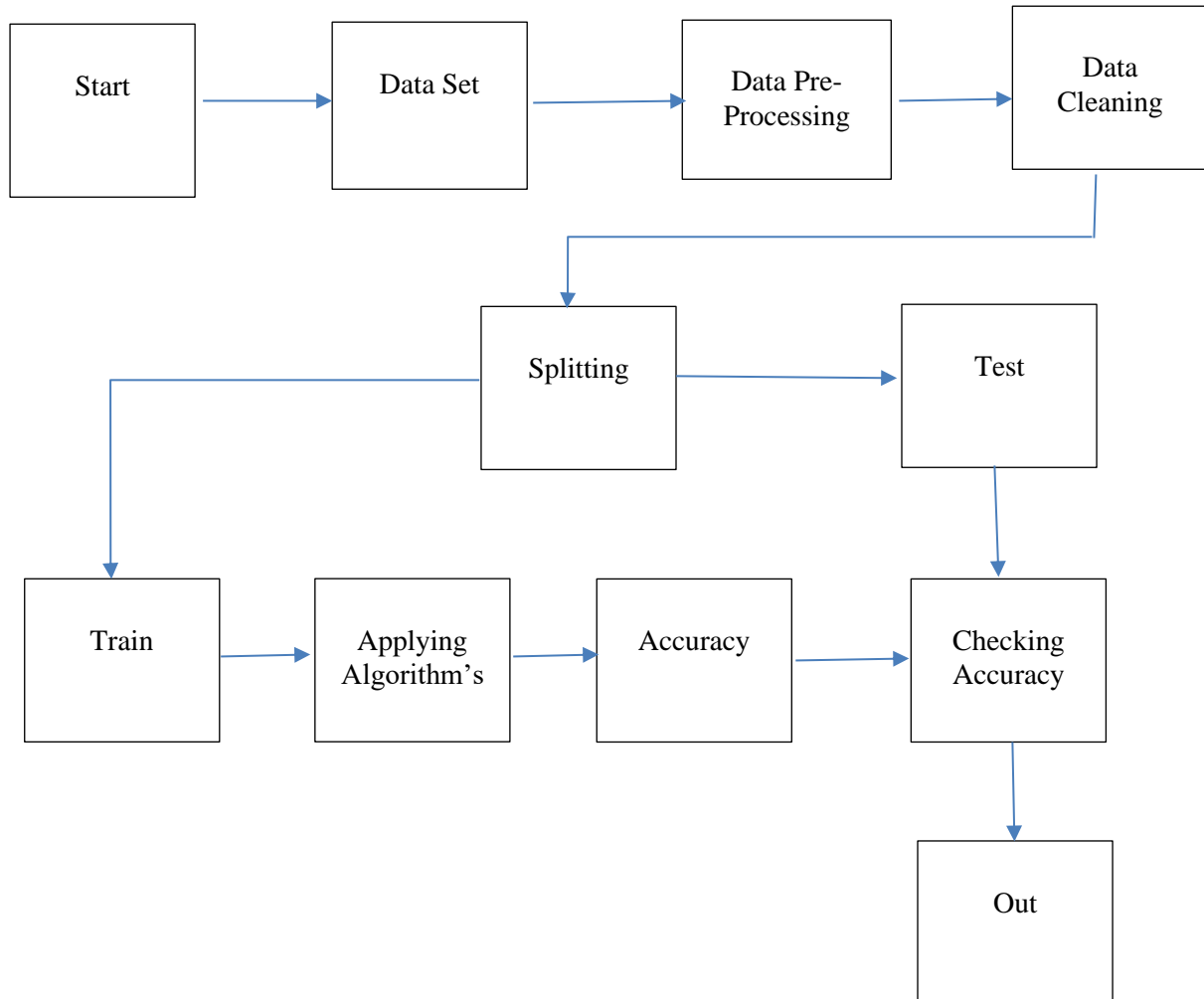
#### **8. Documentation and Reporting:**

- Document all steps undertaken during data preprocessing, model training, and evaluation.
- Prepare a comprehensive report summarizing findings, insights, and recommendations derived from the machine learning model.

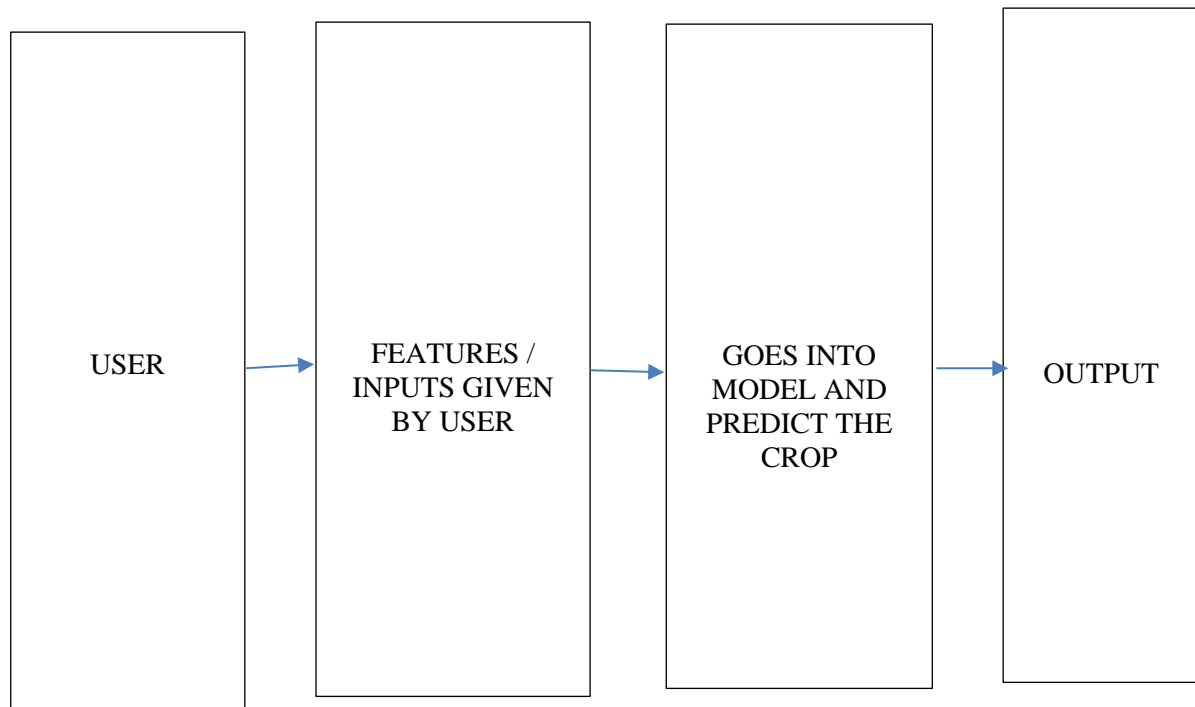
## CHAPTER-6

### SYSTEM DESIGN & IMPLEMENTATION

#### 6.1 ARCHITECTURAL DESIGN:



## 6.2 INTERFACE DESIGN:



## IMPLEMENTATION:

### 1. Deployment:

- Implementation begins with deploying the trained machine learning model into the Tech Agri platform or application. This involves integrating the model into the software or system infrastructure where it can receive input data and provide predictions or recommendations.

### 2. Integration:

- Ensure seamless integration of the machine learning model with the existing Tech Agri platform or application. This might involve collaborating with software engineers or developers to incorporate the model within the system architecture.

### 3. Scalability and Efficiency:

- Optimize the deployed model for efficiency and scalability to handle varying loads of incoming data. This might involve considerations for real-time predictions, batch processing, or handling large volumes of user requests.



**4. Monitoring and Maintenance:**

- Implement monitoring mechanisms to continuously track the model's performance in the live environment. This involves setting up alerts or systems to detect any issues or deviations in the model's behavior.

**5. Model Updates and Retraining:**

- Plan for periodic model updates and retraining cycles. As new data becomes available, periodically retrain the model to improve its accuracy and relevance. Implement mechanisms to seamlessly update the deployed model with the latest improvements.

**6. User Interaction and Feedback:**

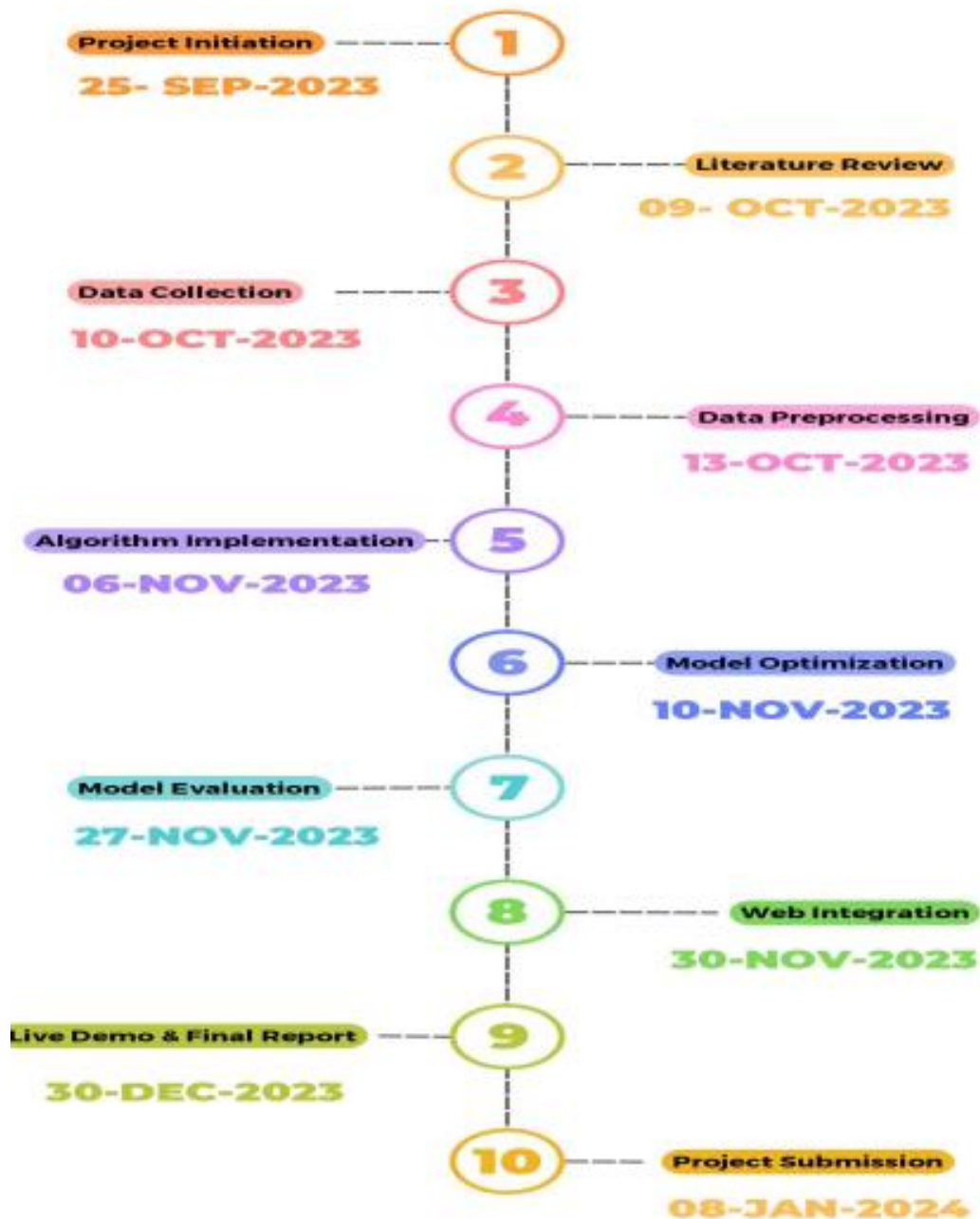
- Design user interfaces or feedback mechanisms to collect user inputs or feedback related to the predictions made by the model. This feedback loop can be valuable for further enhancing the model's performance.

**7. Documentation and Support:**

- Document the implementation process comprehensively, including details on how the model was integrated, any challenges faced, and solutions implemented. Provide support and documentation for users or developers who interact with the model.

## CHAPTER-7

### TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)



## CHAPTER-8

### OUTCOMES

1. Crop Prediction: The model predicts the most suitable crop for cultivation based on the input features (nitrogen, phosphorus, temperature, humidity, pH level of water).
2. Recommendation System: Developing a recommendation system that suggests the best crops to grow based on the given environmental conditions.
3. Future Enhancements: Improving the model's accuracy by fine-tuning hyperparameters, incorporating additional features, or using more advanced machine learning techniques like ensemble methods or deep learning.

#### **Deployment and Use:**

1. Integration: Integrating the trained model into a user-friendly interface or application that farmers or agricultural experts can use easily.
2. Real-Time Prediction: Enabling real-time predictions where farmers input current environmental data to get immediate crop recommendations.
3. Feedback Loop: Creating a feedback mechanism to continuously improve the model based on the performance in practical scenarios.

## CHAPTER-9

### RESULTS AND DISCUSSIONS

#### 1. Model Accuracy and Performance:

- Accuracy Metrics: The model's accuracy in predicting suitable crops based on environmental factors like nitrogen, phosphorus, temperature, humidity, and water pH levels will be a crucial measure.
- Performance Comparison: Comparing the performance of different machine learning algorithms used (e.g., decision trees, random forests, SVM, neural networks) in terms of prediction accuracy and computational efficiency.

```
Decision Tree --> 90%
Naive Bayes --> 99.0909090909091%
SVM --> 97.95454545454545%
Logistic Regression --> 95.22727272727273%
RF --> 99.0909090909091%
```

#### 2. Crop Recommendations:

- Precision of Crop Recommendations: Evaluating how accurately the model suggests the most suitable crops for given environmental conditions.
- Seasonal Variations: Examining if the model's recommendations align with seasonal planting variations and crop growth patterns.

#### 3. Feature Importance:

- Identifying Key Features: Understanding which environmental factors (nitrogen, phosphorus, temperature, etc.) play the most significant role in determining crop suitability. This insight can be valuable for farmers in optimizing their farming practices.
- Visualizing Feature Importance: Creating visualizations or charts to demonstrate the importance of each feature in the crop prediction process.

#### 4. Model Robustness and Generalization:

- Cross-Validation Results: Discussing how well the model generalizes to new data using techniques like k-fold cross-validation.
- Handling Variability: Addressing how the model copes with variations in environmental factors and its robustness in different geographical locations or seasons.

### 5. Real-World Applicability:

- Practical Implementation: Discussing the feasibility and ease of implementing the model's recommendations in real farming scenarios.
- User-Friendliness: Assessing the usability of the model by farmers or agricultural experts and gathering feedback for improvements.

### 6. Future Directions:

- Enhancements: Proposing potential improvements such as incorporating more diverse datasets, considering additional environmental factors, or refining the model architecture for better performance.
- Scaling and Adaptability: Discussing scalability and adaptability to accommodate larger datasets or diverse farming conditions.

## INPUT: FIGURE H



The screenshot displays the 'Farm To Folk' web application interface. The header includes the title 'Farm To Folk' and navigation links for 'Home' and 'Logout'. The main content area features a series of input fields for agricultural data, each with a label and a value:

Parameter	Value
Nitrogen	22
Phosphorous	180
Potassium	2
temperature	41
humidity	37
pH	7
rainfall	211

Below the input fields is a prominent orange 'Predict' button. The background of the form is a scenic image of a forest and mountains at sunset.

## OUTPUT: FIGURE J

The screenshot displays the 'Farm To Folk' web application interface. At the top, there is a purple header bar with the text 'Farm To Folk' on the left and 'Home' and 'Logout' links on the right. The background of the page features a scenic image of mountains and trees at sunset. In the center, there is a light-colored rectangular box containing several input fields for agricultural data: 'Nitrogen:', 'Phosphorous:', 'Potassium:', 'Temperature:', 'Humidity:', 'pH:', and 'rainfall:'. Each label is followed by a white input field. Below these fields is a brown 'Predict' button. At the bottom of the page, the text 'Prediction Result :[kidneybeans]' is displayed.

## CHAPTER-10

### CONCLUSION

#### **Key Findings:**

- The developed machine learning model accurately predicts suitable crops based on environmental conditions, showcasing promising results in crop recommendation.
- Feature analysis reveals the crucial role of certain environmental factors (e.g., nitrogen levels, temperature) in determining crop suitability, providing valuable insights for optimizing farming practices.

#### **Implications and Significance:**

- The project demonstrates the potential of machine learning in aiding farmers' decisions by suggesting crops aligned with specific environmental conditions, potentially improving agricultural yield and resource utilization.
- It emphasizes the importance of data-driven approaches in agriculture, highlighting how technological solutions can contribute to sustainable farming practices.

#### **Future Directions:**

- Future enhancements could involve refining the model's accuracy by incorporating more diverse datasets, exploring additional environmental parameters, and considering regional or seasonal variations for broader applicability.
- Scaling the model's usability and user-friendliness for easy adoption by farmers or agricultural experts remains a crucial aspect for practical implementation.

#### **Conclusion:**

The "Tech Agri" project showcases the efficacy of machine learning in providing tailored crop recommendations based on environmental parameters. Its success signifies a step towards optimizing agricultural practices, promoting sustainability, and offering valuable decision-making tools for farmers in cultivating crops best suited for their specific conditions. Continued advancements in this field hold significant potential for revolutionizing farming practices and addressing agricultural challenges globally.

## REFERENCES

- 1]. Zhang, X., Liu, X., Li, M., Zhang, H., & Tang, X. (2016). Application of machine learning methods in agriculture. *Journal of Agricultural Science and Technology*, 18(6), 1345-1354.
- 2]. Singh, A. K., Ganapathysubramanian, B., Singh, A., & Sarkar, S. (2016). Machine learning for high-throughput stress phenotyping in plants. *Trends in Plant Science*, 21(2), 110-124.
- 3]. Rathore, A., & Bhardwaj, A. (2018). Application of machine learning techniques in agriculture: A review. *International Journal of Computer Sciences and Engineering*, 6(8), 283-288.
- 4]. Wang, Y., Yin, Z., Lu, Q., & Cheng, Q. (2017). Application of machine learning in precision agriculture. *Procedia Computer Science*, 122, 390-395.
- 5]. Nidheesh, M., & Kumar, D. N. (2018). Machine learning techniques in agriculture: A survey. *International Journal of Computer Applications*, 179(39), 13-18.
- 6]. Senthilkumar, N., Sivaranjani, G., & Kaviya, S. (2020). Machine learning in agriculture and its applications. *International Journal of Emerging Trends in Engineering Research*, 8(8), 2156-2159.
- 7]. Marathe, P., & Chaudhari, D. (2019). Machine learning techniques in agriculture for crop prediction. *International Journal of Recent Technology and Engineering*, 7(6), 2811-2815.
- 8]. Rakib, M. R. J., & Islam, M. A. (2019). Agricultural decision support system using machine learning technique. In *2019 IEEE International Conference on Automation, Control and Robots (ICACR)* (pp. 345-350). IEEE.
- 9]. Mishra, S., & Desai, K. (2020). Predictive modeling in precision agriculture using machine learning techniques. *Procedia Computer Science*, 173, 275-282.
- 10]. Pandey, P., Khan, A. M., & Rashid, M. (2021). Machine learning-based crop prediction



system for precision agriculture. In 2021 4th International Conference on Signal Processing and Communication (ICSC) (pp. 1-6). IEEE.

11]. Basavaraj, R., & Mesta, R. K. (2021). Predictive analytics in precision agriculture using machine learning: A review. *International Journal of Applied Engineering Research*, 16(17), 489-500.

12]. Rehman, A., Aziz, S., & Hussain, M. (2017). Crop disease detection and classification using machine learning techniques: A review. In 2017 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET) (pp. 1-5). IEEE.

13]. Bhan, V. M., et al. (2020). A review on machine learning techniques for pest detection and disease prediction in precision agriculture. In 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 249-253). IEEE.

14]. Shankar, B., et al. (2021). Machine learning techniques for smart agriculture: A comprehensive review. In 2021 International Conference on Emerging Trends in Information Technology and Engineering (ICETITE) (pp. 1-6). IEEE.

15]. Singh, S. S., et al. (2019). Predictive analytics in precision agriculture: A review of machine learning techniques. In 2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 688-693). IEEE.

## APPENDIX-A

### PSUEDOCODE

#### Home.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Home Page</title>
  { % load static % }
  <link rel="stylesheet" href="{ % static 'css/style.css' % }">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
</head>
<body>
  <div class="header">
    <div class="container">
      <h1>Tech Agri</h1>
      <nav>
        <ul>
          <li><a href="{ % url 'index' % }">Home</a></li>
          <li><a href="{ % url 'logout' % }">Logout</a></li>
        </ul>
      </nav>
    </div>
  </div>
  <form method="post" action="{ % url 'predict' % }">
    { % csrf_token % }
    <label for="age">Nitrogen:</label>
    <input type="number" id="age" name="age" required><br><br>

    <label for="blood_pressure">Posphorous:</label>
    <input type="number" id="blood_pressure" name="blood_pressure" required><br><br>

```

```

<label for="sugar">Potassium:</label>
<input type="number" id="blood_pressure" name="sugar" required><br><br>

<label for="blood_urea">temperature :</label>
<input type="number" id="blood_pressure" name="blood_urea" required><br><br>

<label for="sodium">humidity:</label>
<input type="number" id="blood_pressure" name="sodium" required><br><br>

<label for="potassium">ph:</label>
<input type="number" id="blood_pressure" name="potassium" required><br><br>

<label for="hemoglobin">rainfall:</label>
<input type="number" id="blood_pressure" name="hemoglobin" required><br><br>

<!-- Add more input fields for other features -->

<input type="submit" value="Predict">
</form>
<div class="main-content">
  <div class="pred">
    { % if result % }
    <h2>Prediction Result : </h2>
    <h2>{{ result }}</h2>
    { % endif % }
  </div>
</div>
</body>
</html>

```

## STYLE.CSS:

```
/ styles.css /
```

```
/ Body styles /
```

```
body {
```

```
  margin: 0;
```

```
padding: 0;
font-family: Arial, sans-serif;
background-image: url('mountains-1412683_1920.png'); / Replace with the path to your
image /
background-size: cover;
background-repeat: no-repeat;
animation: slideLeftToRight 5s infinite;
}
```

/ Header styles /

```
.header {
  backdrop-filter: blur(5px);
  background-color: 49266188;
  color: white;
  padding: 10px 0;
}
```

```
.container {
  display: flex;
  justify-content: space-between;
  width: 80%;
  margin: 0 auto;
}
```

```
h1 {
  margin: 0;
}
```

```
nav ul {
  list-style: none;
  padding: 0;
}
```

```
nav ul li {
```

```
display: inline;
margin-right: 20px;
}
```

```
nav ul li a {
padding: 5px;
width: 25px;
height: 10px;
/ margin: 5px; /
color: white;
text-decoration: none;
transition: 1s;
}
```

```
nav ul li :hover{
backdrop-filter: blur(5px);
background-color: 49266188;
color: white;
border-radius: 10px;
/ background-color: fff; /
}
```

```
/ Form styles /
form {
backdrop-filter: blur(5px);
width: 60%;
margin: 0 auto;
background-color: rgba(85, 78, 69, 0.178);
padding: 30px;
padding-left: 10px;
border-radius: 10px;
margin-top: 50px;
}
```

```
label {
  color: fff;
  display: block;
  margin-bottom: 5px;
}

input[type="number"] {
  width: 100%;
  padding: 8px;
  margin-bottom: 15px;
  border-radius: 5px;
  border: 1px solid ccc;
}

input[type="submit"] {
  width: 100%;
  padding: 10px;
  border-radius: 5px;
  border: none;
  background-color: ad8952dc;
  color: white;
  cursor: pointer;
}

input[type="submit"]:hover {
  background-color: 776d41;
}

/ Prediction result styles /
.pred {
  text-align: center;
  margin-top: 20px;
}
```

```
.pred{
  display: flex;
  justify-content: center;
}
.pred h2{
  color: white;
}
```

## Views.py

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from django.http import HttpResponseRedirect
from django.contrib.auth.decorators import login_required
import pickle
import numpy as np

def HomePage(request):
    return render(request, 'home.html', {})

def Register(request):
    if request.method == 'POST':
        name = request.POST.get('uname')
        email = request.POST.get('email')
        password = request.POST.get('pass')

        new_user = User.objects.create_user(name, email, password)
        new_user.save()
        return redirect('login')

    return render(request, 'register.html', {})

def Login(request):
```

```

if request.method == 'POST':
    name = request.POST.get('uname')
    password = request.POST.get('pass')

    user = authenticate(request, username=name, password=password)
    if user is not None:
        login(request, user)
        return redirect('index')
    else:
        return HttpResponseRedirect('Error, user does not exist')

return render(request, 'login.html', { })

def logoutuser(request):
    logout(request)
    return redirect('login')
@login_required
def Index(request):
    return render(request, 'index.html', { })

```

Load the saved model

RF\_pkl\_filename = 'DecisionTree.pkl' Replace with the path to your .pkl file  
 with open(RF\_pkl\_filename, 'rb') as Model\_pkl:

```
loaded_model = pickle.load(Model_pkl)
```

```
def predict(request):
```

```
    result = None
```

```
    if request.method == 'POST':
```

```
        Retrieve user input from the frontend form
```

```
        age = float(request.POST['age'])
```

```
        bp = float(request.POST['blood_pressure'])
```

```
        sugar = float(request.POST['sugar'])
```



```

blood_urea = float(request.POST['blood_urea'])
sodium = float(request.POST['sodium'])
potassium = float(request.POST['potassium'])
hemoglobin = float(request.POST['hemoglobin'])
Add more fields as needed

```

Prepare the input data for prediction

```

input_data = [age, bp, sugar, blood_urea, sodium, potassium, hemoglobin] Adjust this
according to your model's input features

```

```

input_data_as_array = np.asarray(input_data)
input_data_reshaped = input_data_as_array.reshape(1, -1)

```

Make prediction using the loaded model

```

prediction = loaded_model.predict(input_data_reshaped)

```

```

return render(request, 'home.html', {'result': prediction})

```

## URLS.PY

```

from django.contrib import admin
from django.urls import path
from .views import HomePage, Login, Register, logoutuser, predict, Index

```

```

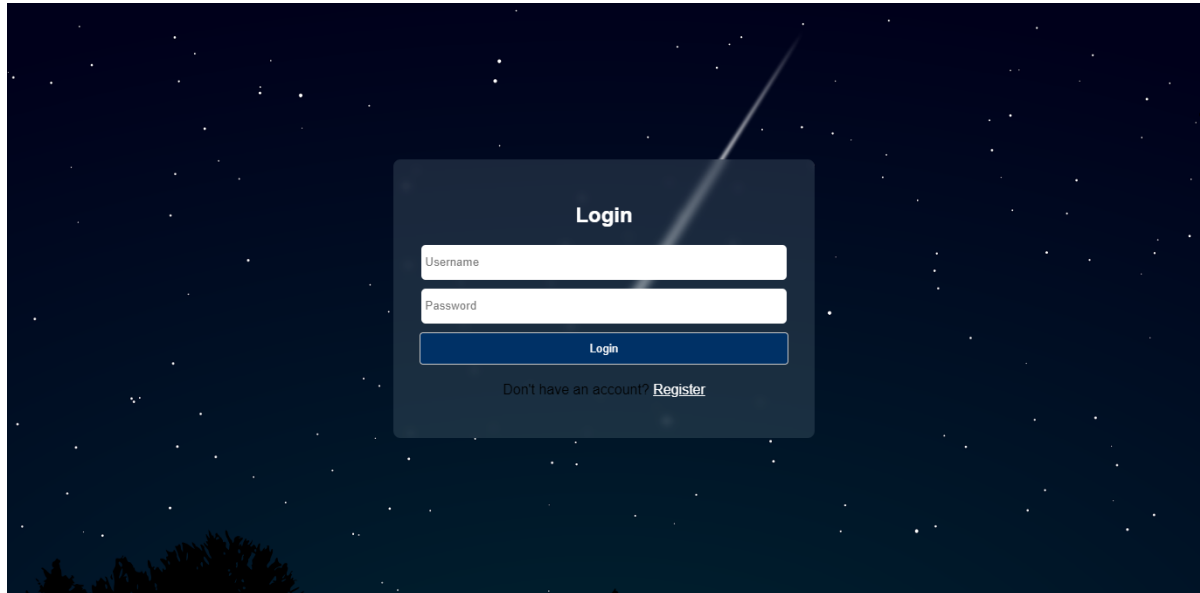
urlpatterns = [
    path('admin/', admin.site.urls),
    path("",Index,name='index'),
    path('home', HomePage,name='home'),
    path('login', Login,name='login'),
    path('register', Register,name='register'),
    path('logout', logoutuser,name='logout'),
    path('predict',predict, name='predict'),
]

```

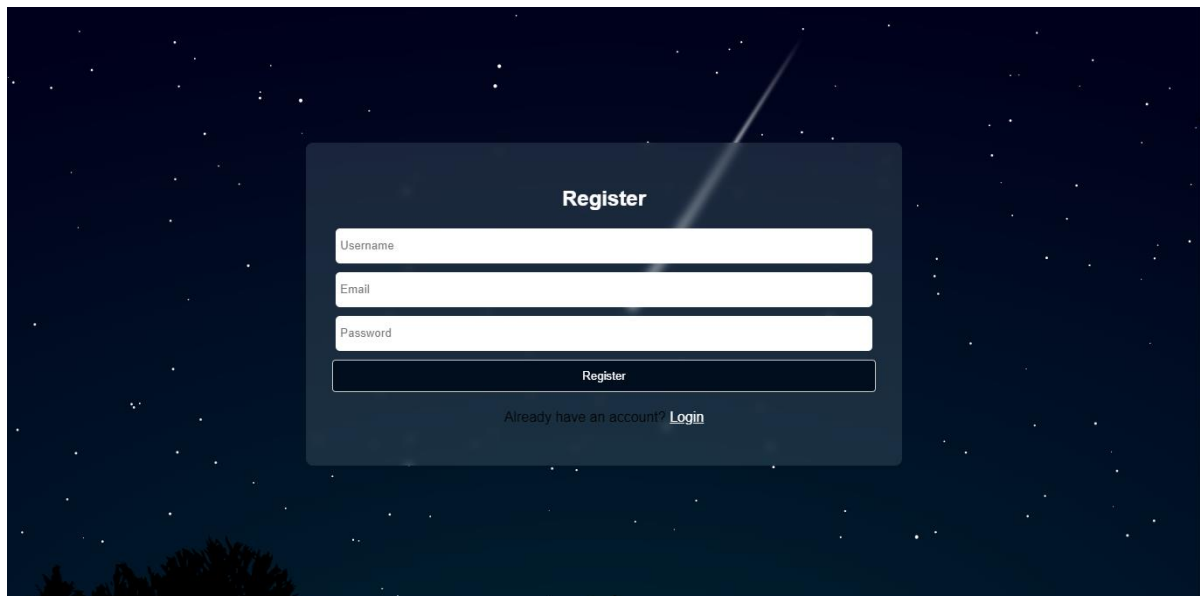
## APPENDIX-B

### SCREENSHOTS

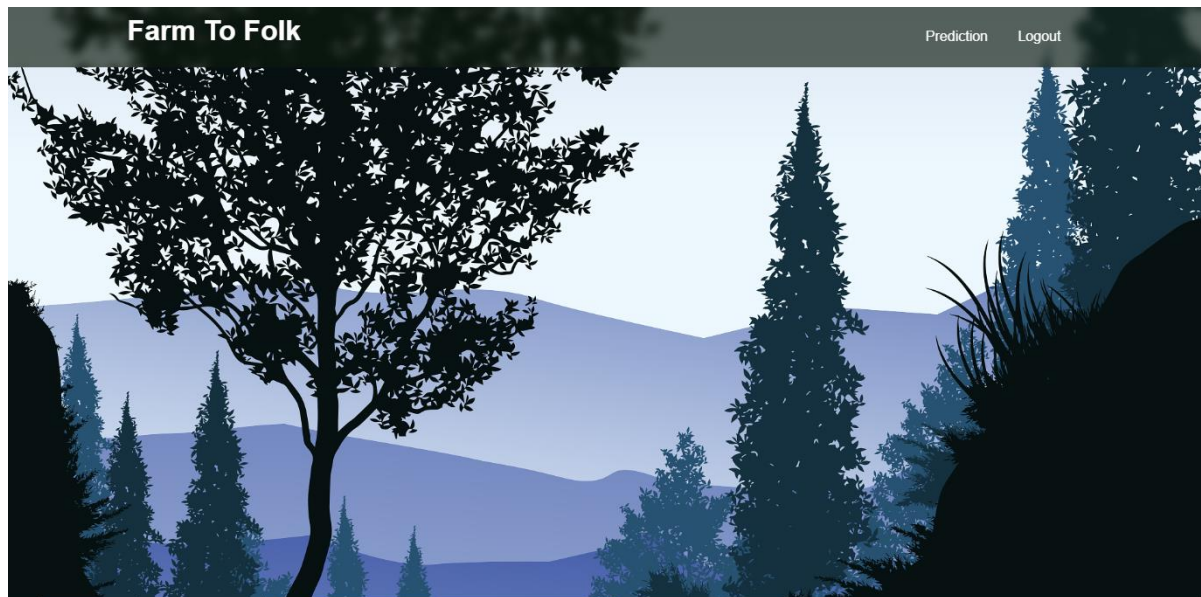
#### Login Page:



#### Register Page:




## Home Page:



## Prediction Page:

The screenshot displays the 'Prediction' page of the 'Farm To Folk' application. The header is a solid purple color with 'Farm To Folk' on the left and 'Home' and 'Logout' links on the right. The main content area has a background image of a forest at sunset. Overlaid on this is a light-colored form with several input fields. The labels for these fields are: 'Nitrogen:', 'Phosphorous:', 'Potassium:', 'temperature:', 'humidity:', 'ph:', and 'rainfall:'. Each label is followed by a white rectangular input box. At the bottom of the form is a wide, orange button labeled 'Predict'.

# Machine Learning Model Creation:

jupyter Crop\_Recommendation\_Model Last Checkpoint: 09/12/2023 (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)


```
In [1]: # Importing libraries
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')
```

```
In [2]: df = pd.read_csv('crop_recommendation.csv')
```

```
In [39]: df.head()
```

```
Out[39]:
```

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

jupyter Crop\_Recommendation\_Model Last Checkpoint: 09/12/2023 (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [4]: df.tail()
```

```
Out[4]:
```

	N	P	K	temperature	humidity	ph	rainfall	label
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

```
In [5]: df.size
```

```
Out[5]: 17600
```

```
In [6]: df.shape
```

```
Out[6]: (2200, 8)
```

```
In [7]: df.columns
```

```
Out[7]: Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')
```

```
In [8]: df['label'].unique()
```

```
Out[8]: array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',
        'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',
        'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',
        'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],
        dtype=object)
```

jupyter Crop\_Recommendation\_Model Last Checkpoint: 09/12/2023 (autosaved) Logout

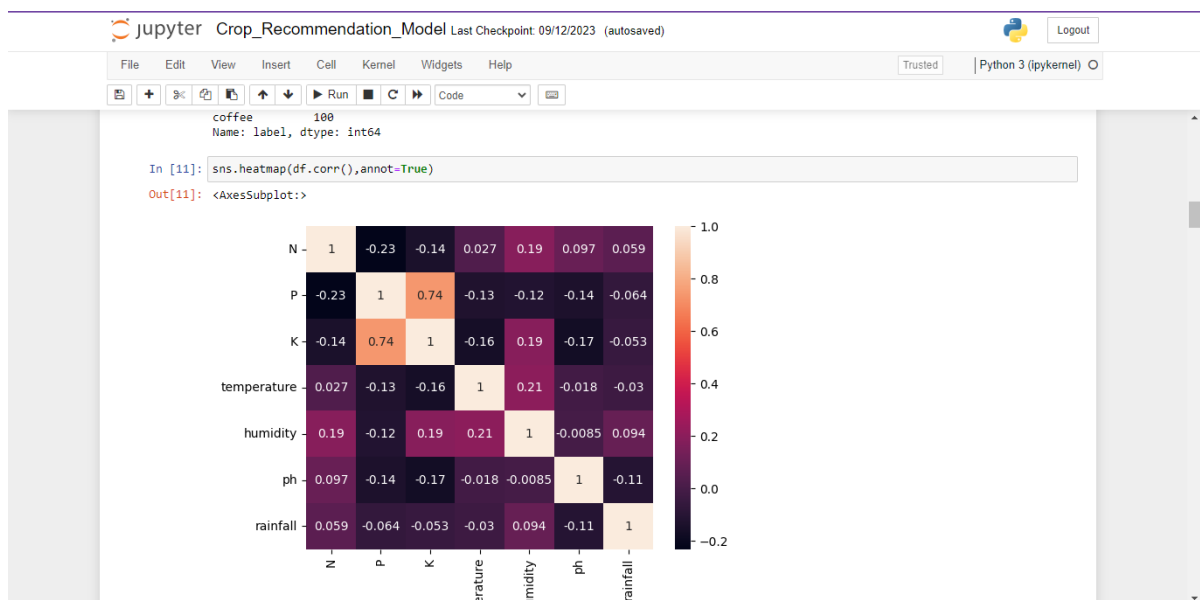
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [9]: df.dtypes
```

```
Out[9]: N          int64
P          int64
K          int64
temperature float64
humidity     float64
ph           float64
rainfall     float64
label        object
dtype: object
```

```
In [10]: df['label'].value_counts()
```

```
Out[10]: rice      100
maize    100
jute      100
cotton    100
coconut   100
papaya    100
orange    100
apple      100
muskmelon 100
watermelon 100
grapes     100
mango      100
banana     100
pomegranate 100
lentil     100
blackgram  100
mungbean   100
mothbeans  100
pigeonpeas 100
```



jupyter Crop\_Recommendation\_Model Last Checkpoint: 09/12/2023 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
Seperating features and target label
```

```
In [12]: features = df[['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']]
target = df['label']
#features = df[['temperature', 'humidity', 'ph', 'rainfall']]
labels = df['label']
```

```
In [13]: # Initializing empty lists to append all model's name and corresponding name
acc = []
model = []
```

```
In [14]: # Splitting into train and test data
from sklearn.model_selection import train_test_split
Xtrain, Xtest, Ytrain, Ytest = train_test_split(features, target, test_size = 0.2, random_state = 2)
```

jupyter Crop\_Recommendation\_Model Last Checkpoint: 09/12/2023 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

### Decision Tree

```
In [15]: from sklearn.tree import DecisionTreeClassifier

decisionTree = DecisionTreeClassifier(criterion="entropy", random_state=2, max_depth=5)

decisionTree.fit(Xtrain, Ytrain)

predicted_values = decisionTree.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Decision Tree')
print("DecisionTrees's Accuracy is: ", x*100)

print(classification_report(Ytest, predicted_values))
```

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	13
banana	1.00	1.00	1.00	17
blackgram	0.59	1.00	0.74	16
chickpea	1.00	1.00	1.00	21
coconut	0.91	1.00	0.95	21
coffee	1.00	1.00	1.00	22
cotton	1.00	1.00	1.00	20
grapes	1.00	1.00	1.00	18
jute	0.74	0.93	0.83	28
kidneybeans	0.00	0.00	0.00	14
lentil	0.68	1.00	0.81	23
maize	1.00	1.00	1.00	21

jupyter Crop\_Recommendation\_Model Last Checkpoint: 09/12/2023 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

	precision	recall	f1-score	support
rice	1.00	0.62	0.77	16
watermelon	1.00	1.00	1.00	15
accuracy			0.90	440
macro avg	0.84	0.88	0.85	440
weighted avg	0.86	0.90	0.87	440

```
In [16]: from sklearn.model_selection import cross_val_score

In [17]: # Cross validation score (Decision Tree)
score = cross_val_score(decisionTree, features, target, cv=5)

In [18]: score
Out[18]: array([0.93636364, 0.90909091, 0.91818182, 0.87045455, 0.93636364])
```

### Saving trained Decision Tree model

```
In [19]: import pickle
# Dump the trained Naive Bayes classifier with Pickle
DT_pkl_filename = './models/DecisionTree.pkl'
# Open the file to save as pkl file
DT_Model_pkl = open(DT_pkl_filename, 'wb')
pickle.dump(decisionTree, DT_Model_pkl)
# Close the pickle instances
DT_Model_pkl.close()
```

## **APPENDIX-C**

### **ENCLOSURES**

- 1. Conference Paper Presented Certificates of all students.**
- 2. Include certificate(s) of any Achievement/Award won in any project-related event.**
- 3. Similarity Index / Plagiarism Check report clearly showing the Percentage (%). No need of page-wise explanation.**