Intelligent Crop Yield Prediction for Precision Farming

Kashyap Gorasiya kashyap2413103@iitgoa.ac.in
School of Mathematics and Computer Science, Indian Institute of Technology Goa,
Farmagudi, Ponda, Goa 403401, India

Abstract

This project explores the use of machine learning for crop yield prediction aimed at enhancing precision farming practices. Leveraging key environmental and soil parameters such as nitrogen (N), phosphorus (P), potassium (K), pH, temperature, humidity, and rainfall, we developed models to recommend optimal crops and forecast yield. The dataset includes 22 crop types across varied agricultural conditions. We implemented and compared K-Nearest Neighbors (KNN), Decision Trees, and Support Vector Machines (SVM) for both classification and regression. Results show that SVM provides the highest classification accuracy, while KNN demonstrates superior performance in yield prediction.

1 Introduction

Traditional farming methods often rely on manual observation and generalized strategies, which overlook variability in factors like soil nutrients, pH, and climate conditions. This can lead to poor yield predictions and inefficient resource use.

To address these limitations, this project applies machine learning to crop recommendation and yield estimation using key parameters such as nitrogen (N), phosphorus (P), potassium (K), pH, temperature, humidity, and rainfall. We evaluate algorithms like K-Nearest Neighbors (KNN), Decision Trees, and Support Vector Machines (SVM) on the Crop Recommendation Dataset, demonstrating improved accuracy and efficiency in agricultural decision-making.

2 Dataset Overview

- **Dataset**: Crop Recommendation Dataset (Kaggle)
- **Size**: 2200 rows, 7 features (N, P, K, temperature, humidity, pH, rainfall) + 1 label (crop)
- **Classes**: 22 distinct crop types

Classes:

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

Table 1: Dataset Feature Overview

Feature	Description	Category
N	Nitrogen content in soil	Soil Quality
P	Phosphorus content	Soil Quality
K	Potassium content	Soil Quality
Temperature	Ambient temperature (Celsius)	Weather
Humidity	Humidity percentage	Weather
pH	Soil acidity	Location
Rainfall	Rainfall (mm)	Location

3 Data Augmentation & Handling Missing Values

Handling missing data is an important step in preparing datasets for machine learning models. In this project, missing values were dealt with by applying **class-wise mean imputation**. Instead of using a global average, the missing entries for each feature were filled using the mean value calculated within the same crop class. This method helped maintain the internal consistency of each class and reduced the risk of introducing unwanted bias.

The imputation was carried out using the following formula:

$$\mu_{\text{class}}(f) = \frac{1}{n} \sum_{i=1}^{n} f_i$$

where:

- n is the number of samples available for the crop class.
- f represents a specific feature such as Nitrogen (N), Phosphorus (P), Potassium (K), Temperature, Humidity, pH, or Rainfall.

Table 2: Sample Class-wise Mean Feature Values

Crop	N	P	K	Temp (°C)	Humidity (%)	pH	Rainfall (mm)
Apple	20.80	134.22	199.89	22.63	92.33	5.93	80.36
Banana	100.23	82.01	50.05	27.38	—	—	—
Blackgram	40.02	67.47	19.24	29.97	—	—	29.97
Chickpea	40.09	67.79	—	79.92	—	—	79.92
Coconut	21.98	—	—	16.93	—	—	16.93

By filling missing values carefully, the dataset became cleaner and more reliable, laying a solid foundation for effective model training and evaluation.

4 Feature Correlation

Correlation measures how two variables move in relation to each other. The Pearson Correlation Coefficient quantifies the linear relationship between two variables, calculated by:

$$\rho(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \tag{1}$$

where:

- $\text{cov}(X, Y)$ is the covariance of X and Y.
- σ_X and σ_Y are the standard deviations of X and Y.

From the analysis, we observed the following:

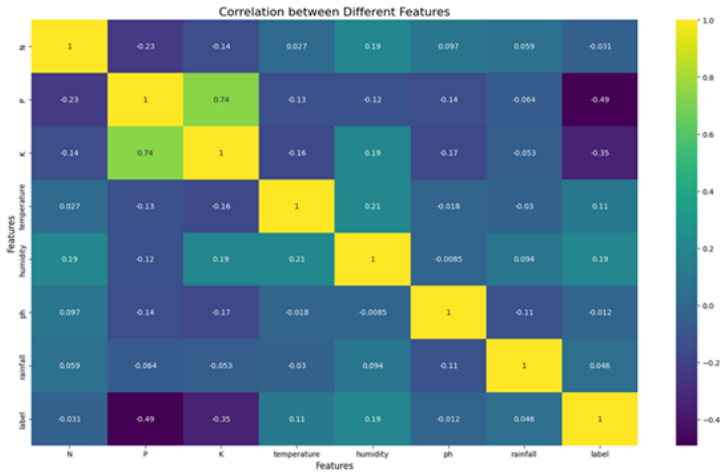

Figure 1: Correlation between different features

- **Phosphorus (P) and Potassium (K)**: Strong positive correlation of 0.74; as phosphorus increases, potassium content also tends to rise.
- **Phosphorus (P) and Crop Label**: Moderate negative correlation of -0.49; higher phosphorus content may be linked with the absence of certain crops.
- **Other Features**: Most other correlations were weak or negligible.

These findings highlight the significant role of soil properties like phosphorus and potassium in crop selection, while also showing that some relationships between features are weaker or inconclusive.

5 Dimensionality Reduction and Visualization: t-SNE

t-SNE (t-Distributed Stochastic Neighbor Embedding) reduces high-dimensional data to 2D or 3D while preserving pairwise distances. It helps visualize complex datasets and reveals patterns.

In this case, t-SNE visualizes how different crop types cluster based on features like Nitrogen, Phosphorus, and Temperature.

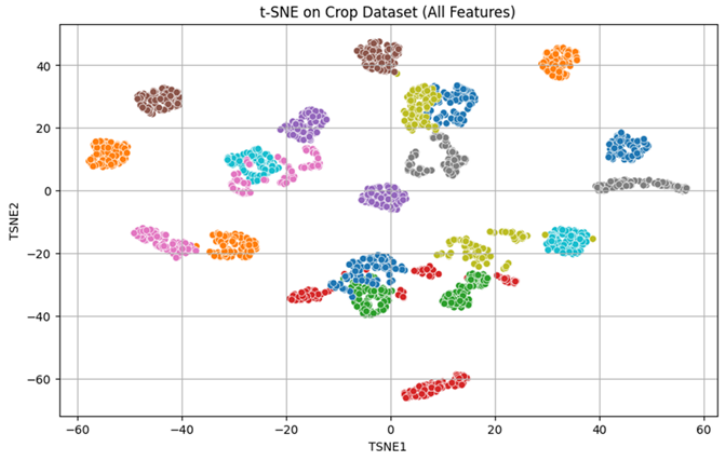

Figure 2: t-SNE visualization of crop types based on features

6 Machine Learning Algorithms

The goal of using machine learning algorithms in this project is to predict suitable crops or estimate yields based on soil and weather conditions.

Algorithm	Task Type	Application	Strengths
KNN	**Classification**	Recommend crop based on similar conditions	Simple, intuitive, no training phase
	Regression	Estimate yield based on nearest neighbors	Good for local patterns, non-parametric
Decision Tree	**Classification**	Decision rule-based crop recommendation	Easy to interpret, handles non-linear separation
	Regression	Yield prediction with decision rules	Fast inference, handles missing values
Support Vector	**Classification**	Finds optimal boundary for crop classification	Works well in high-dimensional space, robust to outliers
	Regression (SVR)	Margin-based model for precise yield prediction	High accuracy with proper kernel/parameter tuning

Table 3: Machine Learning Algorithms for Crop Recommendation and Yield Prediction

Summary

- KNN is great for detecting local patterns—similar soil/weather → similar crops or yields.
- Decision Trees help understand feature impact—identifying which factors are most influential.
- Support Vector (SVM/SVR): Effective for complex, non-linear relationships—strong performance with tuning.
- Used both for classification (crop type) and regression (yield estimation).

7 Train-Test Split

The dataset consists of 2200 rows, which were split into training and testing subsets as follows:

- **Training Set**: 70% (1540 rows)
- **Testing Set**: 30% (660 rows)

This split was achieved using the `train_test_split` function:

```
x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.3,
```

The 30% testing set allows for evaluating model performance on unseen data.

8 Machine Learning Algorithms

This section presents an overview of the machine learning algorithms used for crop recommendation and yield prediction, detailing the steps involved, relevant formulas, and the key hyperparameters.

8.1 K-Nearest Neighbors (KNN)

8.1.1 Overview

- A supervised learning algorithm used for classification.
- Classifies based on similarity to nearby points.
- Uses distance functions to find the nearest neighbors:
  - Euclidean Distance formula:
$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$
  - Manhattan Distance formula:
$$d = |x_1 - x_2| + |y_1 - y_2|$$
- The algorithm is represented with a red point (test sample), blue points (training points), and a green point (potential new point).

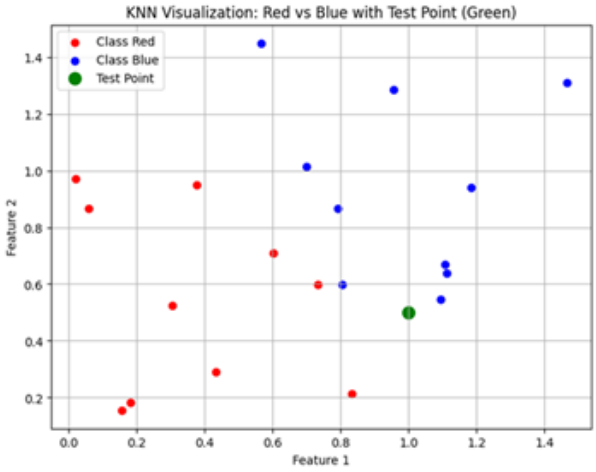

Figure 3: K-Nearest Neighbors: Red point (test sample), blue points (training points), and green point (potential new point).

8.1.2 Algorithm Steps

1. Input a new data point (test sample).
2. Calculate distance to all points in the training set.
3. Sort the distances in ascending order.
4. Select top k nearest neighbors.
5. Classify the new point based on majority vote from neighbors.

8.1.3 Hyperparameter: k

- Small k (e.g., 1–3):
  - Sensitive to noise and outliers.
- Large k (e.g., 10+):
  - More stable, but may ignore local structure.
- Choose $k < \sqrt{n}$ where n is the total number of data points.
- Use cross-validation to find the optimal k.

8.1.4 Elbow Point

- The elbow point is a method used to determine the optimal value of k.
- Plot the error rate against different values of k and choose the point where the error starts to decrease less significantly.

8.1.5 Results (Confusion Matrix)

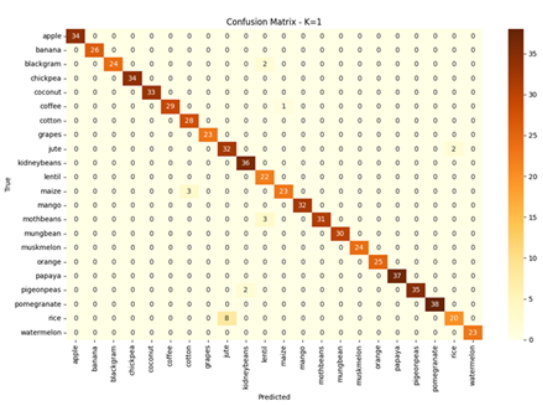

Figure 4: k = 1 Accuracy Score: 0.9681

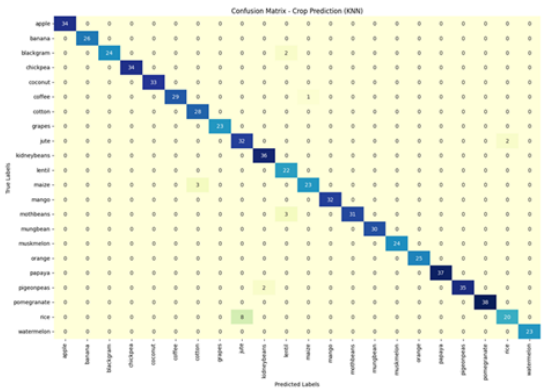

Figure 5: best_k = 3 Accuracy Score: 0.9712

8.1.6 Cross-Validation

Conclusion: The optimal value of k is found to be $k = 3$, as it provides the highest accuracy and lowest error rate. The accuracy vs. k plot shows significant improvement until $k = 3$, after which performance stabilizes. Similarly, the error rate decreases sharply up to $k = 3$, then plateaus, indicating that larger values of k do not substantially improve performance. Thus, $k = 3$ represents the "elbow point" where the balance between bias and variance is most effective.

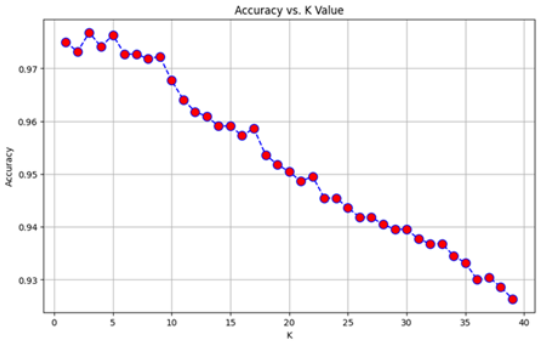

Figure 6: Accuracy vs. k

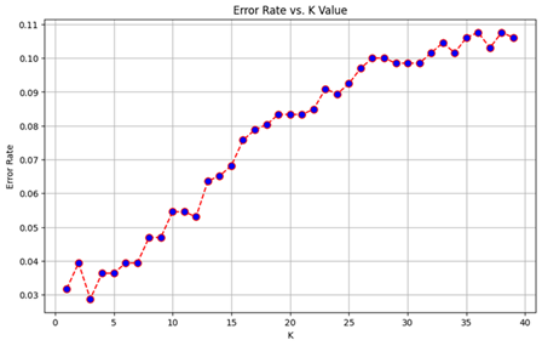

Figure 7: Error Rate vs. k

8.1.7 KNN Regression Results

Actual vs Predicted Crop Labels

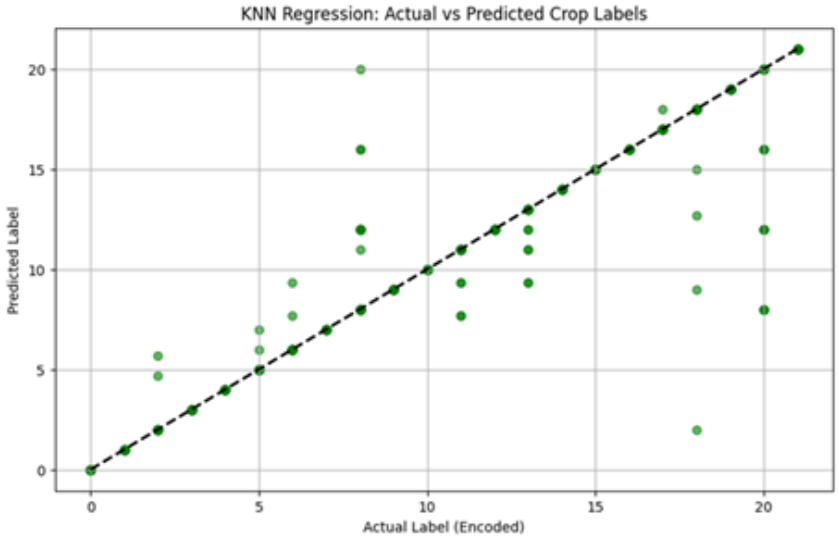

Figure 8: Actual vs Predicted Crop Labels (KNN)

Performance Metrics:

- R^2 Score: 0.9343
- Mean Squared Error (MSE): 2.6554

The KNN regression model shows strong performance with an R^2 score of 0.9343, indicating a high level of explained variance, and a Mean Squared Error (MSE) of 2.6554, reflecting the average squared difference between actual and predicted crop labels.

8.2 Decision Tree

8.2.1 Overview

A Decision Tree is a supervised learning algorithm used for both classification and regression tasks. It models decisions through a tree-like structure, where:

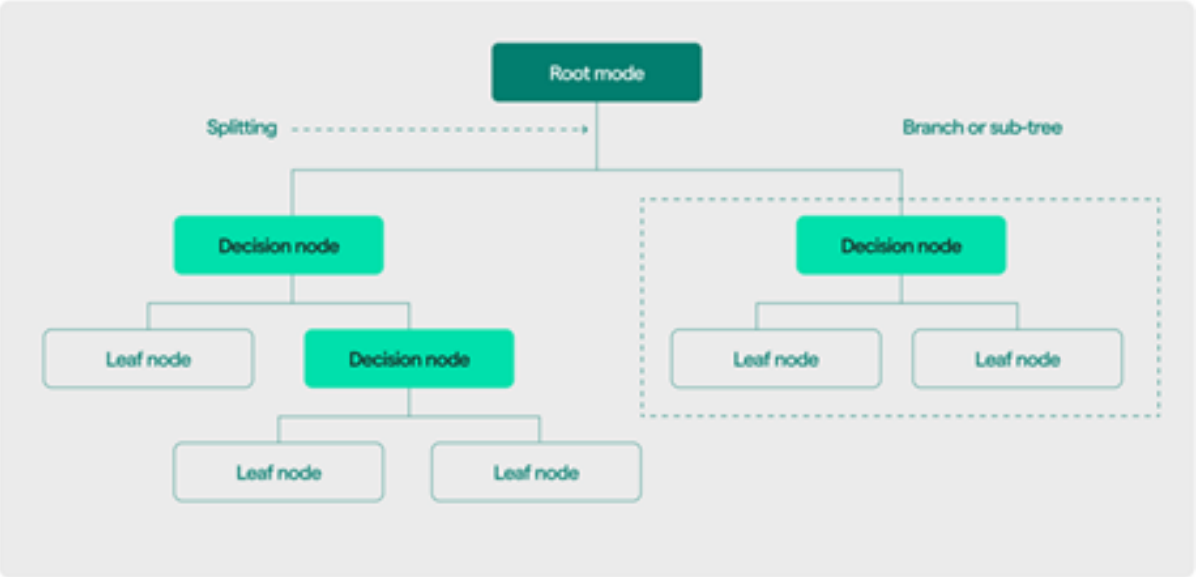

Figure 9: Decision Tree Structure

- **Internal nodes:** Represent tests on features.
- **Branches:** Represent outcomes of tests.
- **Leaf nodes:** Represent final predictions or class labels.

The tree is constructed by recursively splitting the data based on feature values, aiming to improve predictions at each node.

8.2.2 How to Select a Split?

The quality of a split is evaluated using impurity measures like **Entropy** and **Gini Index**:

- **Entropy (Disorder):**

$$\text{Entropy}(S) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

  where p_i is the probability of class i in the dataset S.

- **Gini Index (Impurity):**

$$\text{Gini}(S) = 1 - \sum_{i=1}^{k} p_i^2$$

  where p_i is the probability of class i in the dataset S.

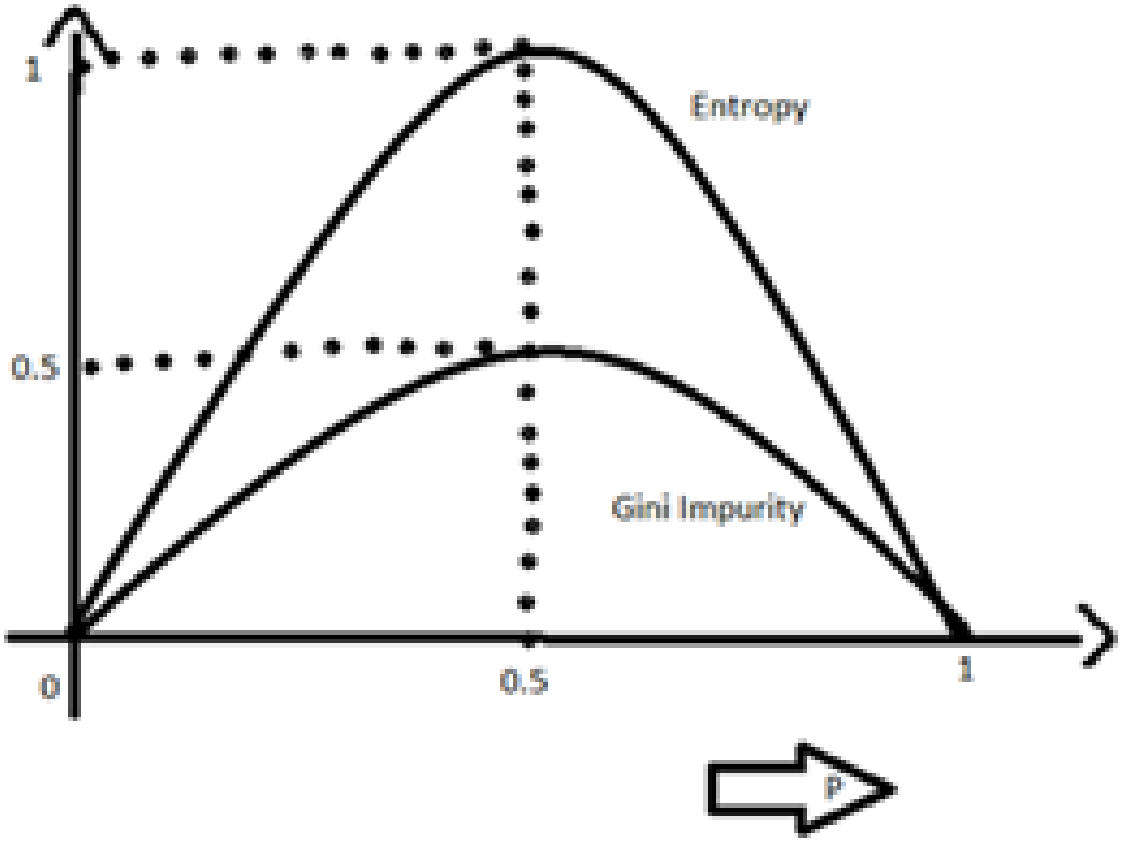

Figure 10: Decision Tree Structure

The best split is the one that minimizes impurity (i.e., maximizes information gain). The formula for Information Gain based on entropy is:

$$\text{Information Gain} = \text{Entropy(Parent)} - \sum_{k} \left(\frac{|S_k|}{|S|} \times \text{Entropy}(S_k) \right)$$

where:

- S is the original dataset,
- S_k are the subsets formed after the split,
- $|S_k|$ is the size of subset k,
- $|S|$ is the size of the parent dataset.

8.2.3 Overfitting & Solutions

Overfitting: Decision trees can overfit the data by growing too deep and capturing noise instead of general patterns. This leads to poor generalization on unseen data.

Pruning: There are two main approaches to pruning decision trees:

- **Post-pruning:** This involves pruning the tree after it is fully grown, removing nodes that do not improve the model's performance.

- **Pre-pruning:** This stops the tree from growing beyond a certain depth or splitting when the data is insufficient.

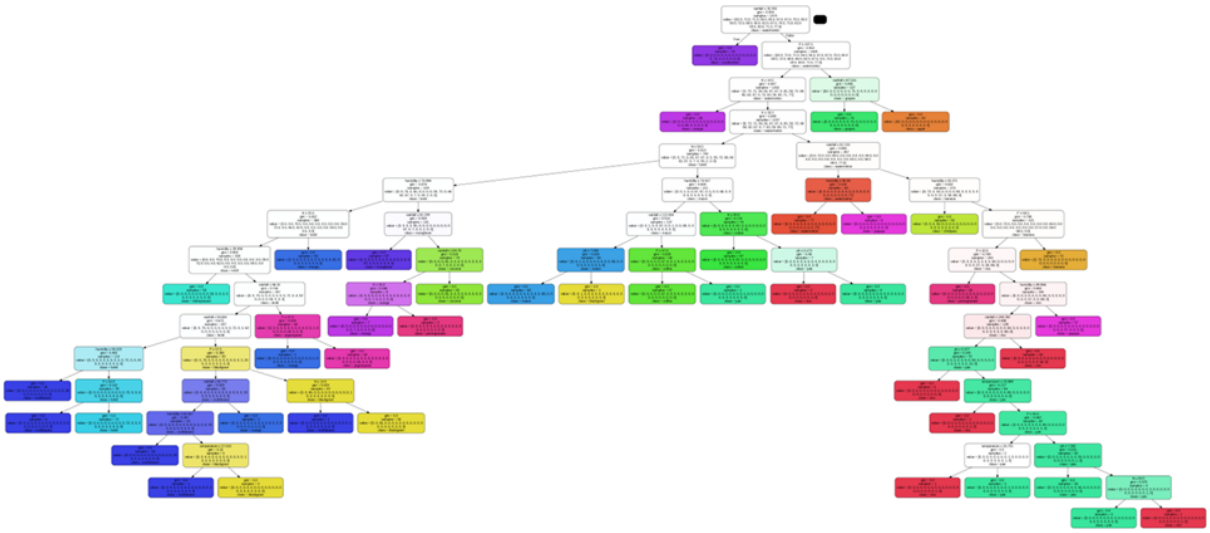

Figure 11: Complete Decision Tree with Overfitting

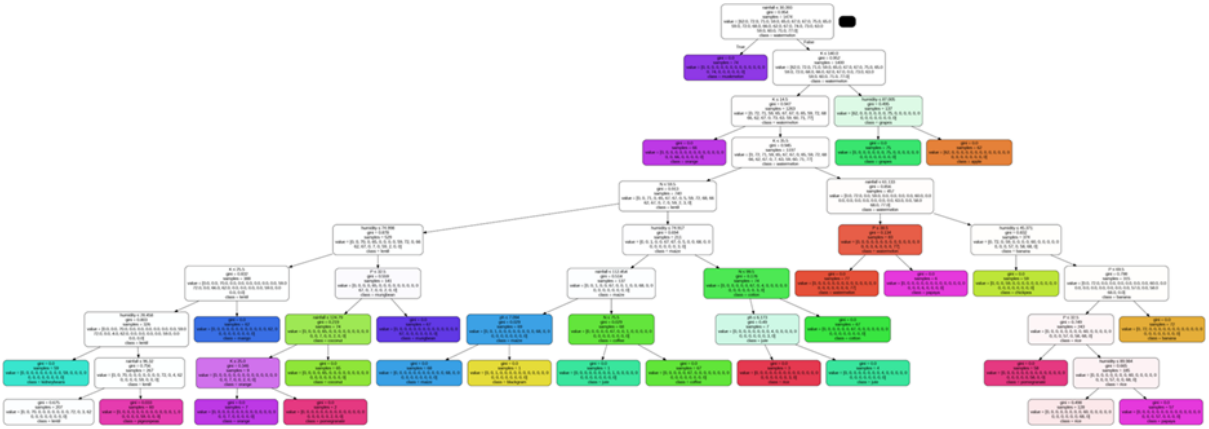

Figure 12: Post-Pruning Decision Tree max depth=9

Hyperparameters for Overfitting:

- **Max Depth:** Limits the maximum depth of the tree.
- **Min Samples Split:** Defines the minimum number of samples required to split a node.
- **Min Samples Leaf:** Defines the minimum number of samples required to be at a leaf node.

8.2.4 Results

8.2.5 Decision Tree Classification

Best Parameters: Using cross-validation, the best parameters were selected, including pre-pruning with a maximum depth of 8 to avoid overfitting during training.

Figure 13: Pre-Pruning Decision Tree max depth=8

- **Best Parameters Found:**

  {'criterion' : 'log_loss', 'max_depth' : 8, 'max_features' : 'sqrt', 'splitter' : 'best'}

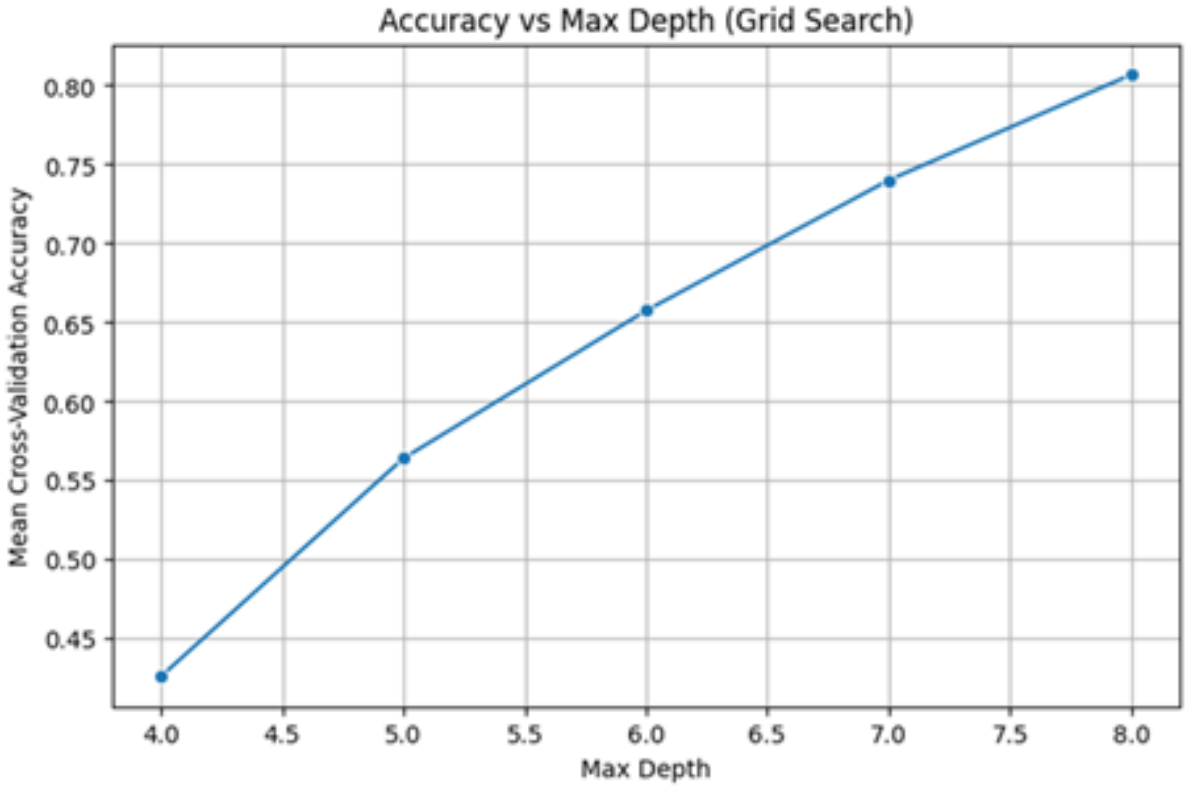

Figure 14: Decision Tree: Cross-validation (Accuracy vs. Max Depth)

8.2.6 Decision Tree Confusion Matrix Comparison

The following figures show the confusion matrices for the pre-pruned and post-pruned decision trees:

Pre-pruning Max Depth=8 (Accuracy: 0.9366) vs Post-pruning Max Depth=9 (Accuracy: 0.85):

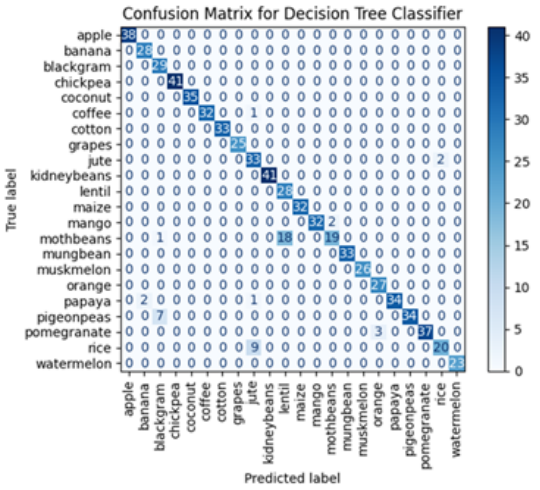

Figure 15: Confusion Matrix: Pre-pruning Max Depth=8

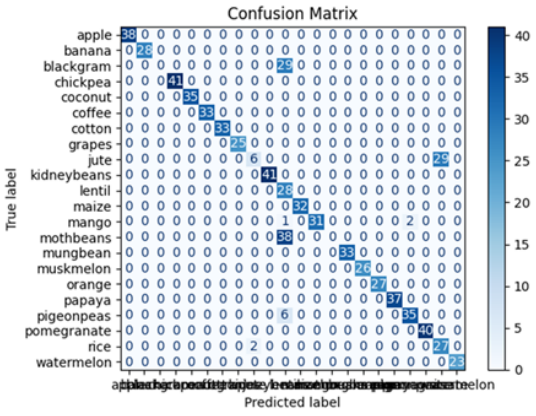

Figure 16: Confusion Matrix: Post-pruning Max Depth=9

8.2.7 Decision Tree Regression

- A supervised algorithm for predicting continuous values (e.g., crop yield).
- Splits the data based on input features to create regions with low target variance.
- At each leaf node, prediction = mean of target values in that node.

How It Works:

1. Start from the root (whole dataset).
2. At each node, find the best split to reduce variance in the target variable.
3. Repeat recursively to grow the tree.

Variance at a Node:

$$\text{Var}(S) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

where:

- y_i = target value of the i-th sample
- $\bar{y}$ = mean of target values in the node
- n = number of samples in the node (also used as weight of each child node)

Variance Reduction:

$$w_i = \frac{n_i}{n}$$

where:

- n_i = number of samples in child node i
- n = total number of samples in parent node

- $\text{Var}(\text{Child}_i)$ = variance of child node i

$$\text{Variance Reduction} = \text{Var}(\text{Parent}) - \sum_i w_i \times \text{Var}(\text{Child}_i)$$

Best Split Selection:

$$\text{Best Split} = \underset{\text{all possible splits}}{\arg\max} \quad \text{Variance Reduction}$$

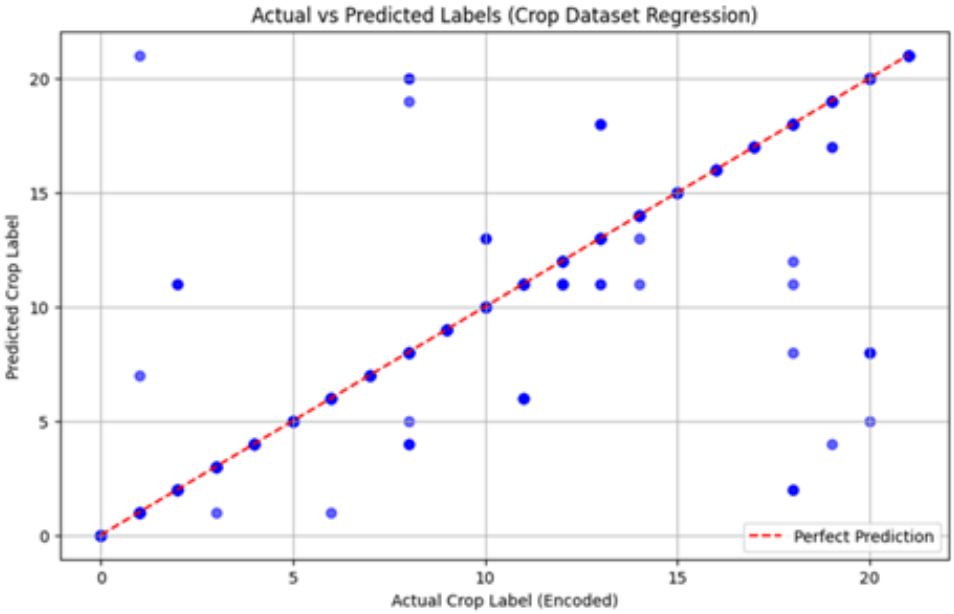

Figure 17: Decision Tree Regression Structure

8.3 Support Vector

8.3.1 Support Vector Classification

Support Vector Classification (SVC) is a supervised classification algorithm that finds the optimal hyperplane to separate data classes with the maximum margin. It supports both linear and non-linear classification (via the kernel trick).

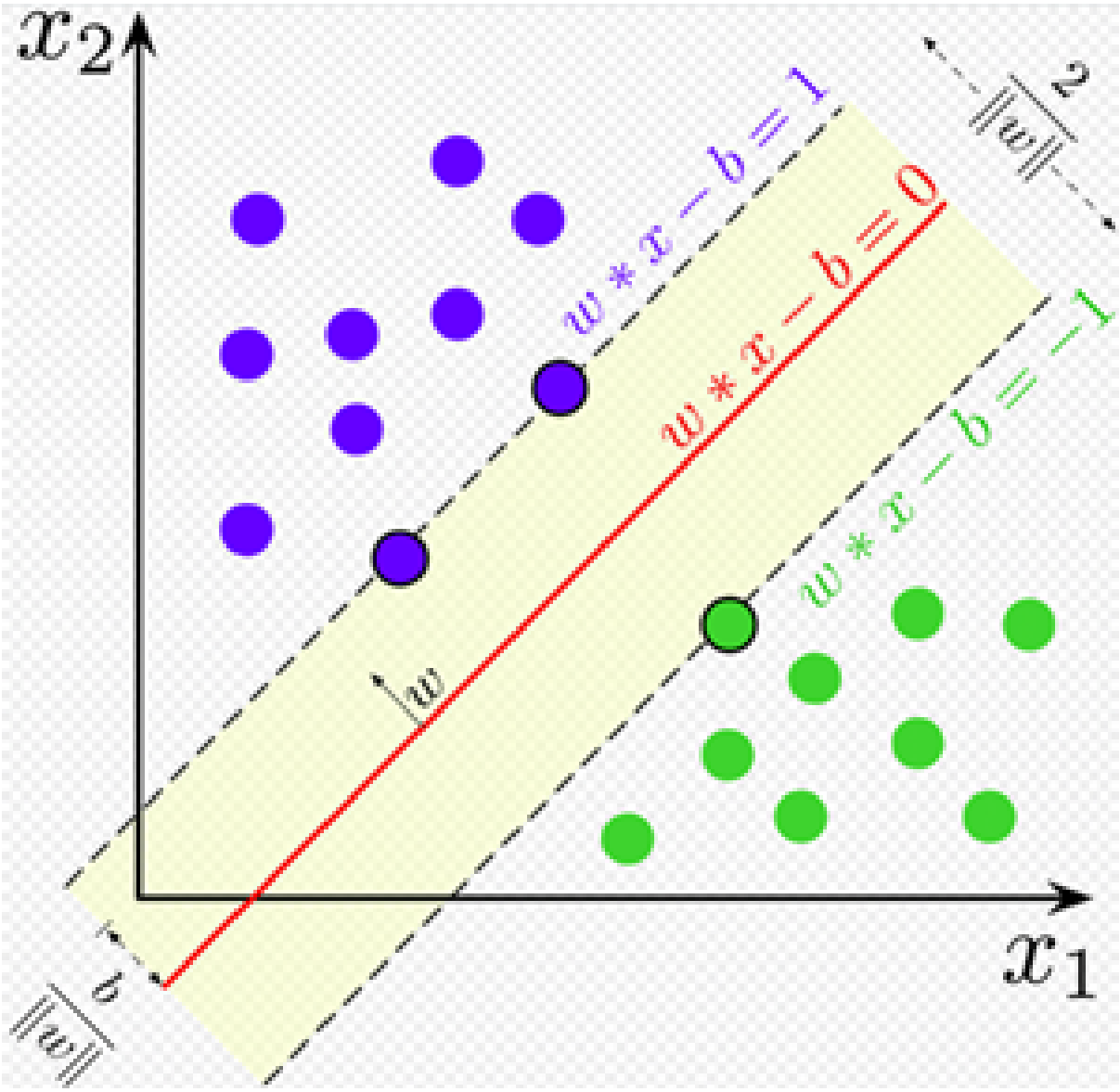

Figure 18: Support Vector Classification: Hyperplane with Maximum Margin

The objective of Support Vector Classification is to maximize the margin between two classes. The optimization goal is:

$$\text{Maximize Margin} \quad \Leftrightarrow \quad \text{Minimize } \frac{1}{2}\|w\|^2$$

Where:

- $y_i \in \{-1, +1\}$ is the true label, - x_i is the feature vector, - w is the weight vector, - b is the bias term.

Best Parameters: `C: 10, gamma: scale, kernel: rbf`
Results:

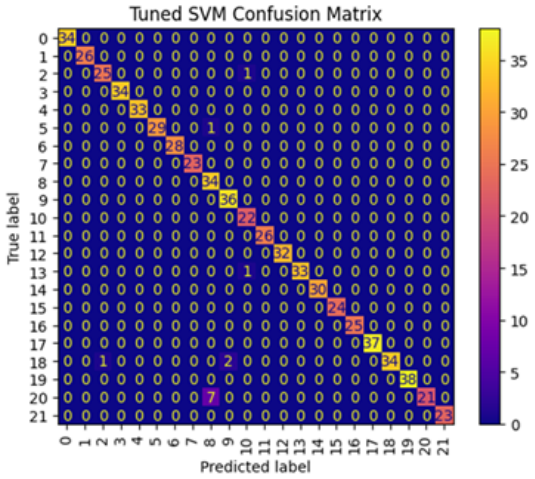

Figure 19: Accuracy Score: 0.9803

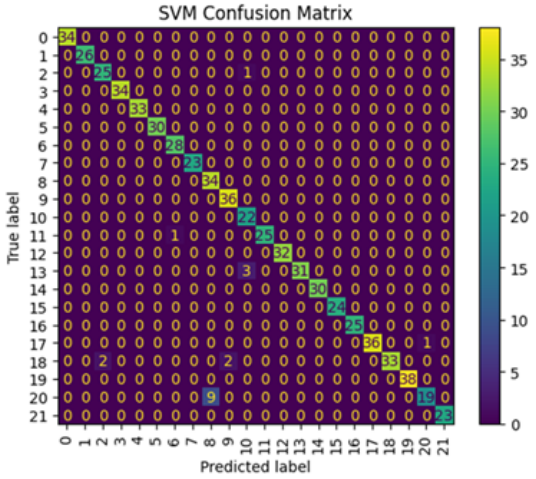

Figure 20: Accuracy Score: 0.9712

8.3.2 Support Vector Regression

Support Vector Regression (SVR) is used for predicting continuous outputs by fitting a function within a margin ϵ from the true values. Unlike classification, SVR focuses on regression tasks.

The objective in SVR is to keep the model as flat as possible while minimizing the weights. This is done by maximizing the margin, which is similar to classification.

$$\text{Maximize Margin} \quad \Leftrightarrow \quad \text{Minimize } \frac{1}{2}\|w\|^2$$

8.3.3 Objective and Constraints for SVR

Objective:

$$\text{Minimize:} \quad \frac{1}{2}\|w\|^2$$

Constraints:

$$|y_i - (w^T x_i + b)| \leq \epsilon \quad \text{for all } i$$
$$y_i - (w^T x_i + b) \leq \epsilon + \xi_i \quad \text{for all } i$$
$$(w^T x_i + b) - y_i \leq \epsilon + \xi_i^* \quad \text{for all } i$$
$$\xi_i, \xi_i^* \geq 0$$

Regularization:

$$\text{Minimize:} \quad \frac{1}{2}\|w\|^2 + C\sum_i (\xi_i + \xi_i^*)$$

Where:
- ϵ is the tolerance margin, - ξ_i, ξ_i^* are the slack variables (errors beyond ϵ), - C is the regularization parameter.

Results:

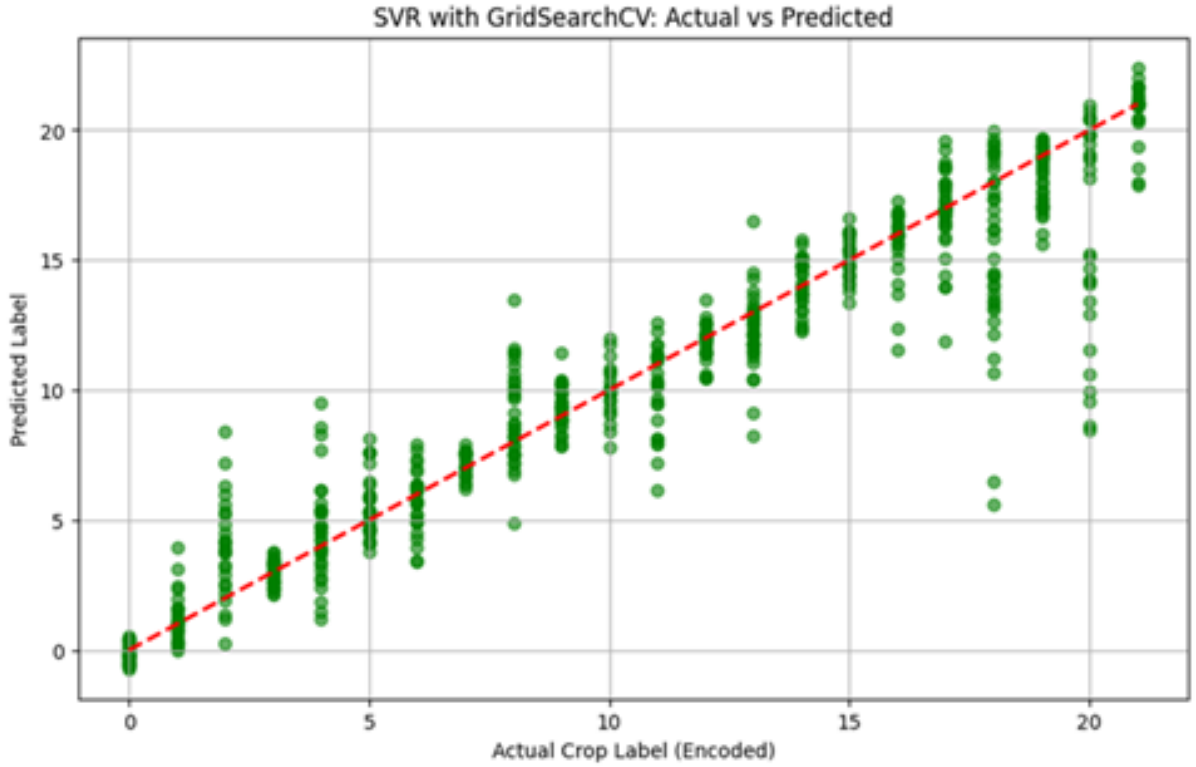

Figure 21: Support Vector Regression: Predicted vs True Values

Performance Metrics: - R^2 Score: 0.8988 - Mean Squared Error (MSE): 4.0918

9 Predict Labels: test50.csv

The dataset `test50.csv` contains several features, including N, P, K, temperature ,humidity, ph, rainfall with some missing values. The steps involved are as follows:

1. Filling Missing Values: We first handle the missing values in the dataset. Several imputation methods can be Class-wise Feature Means.

2. Prediction Using Classification Algorithms: Once the missing values are filled, we apply three classification algorithms to predict the labels for the dataset.

N	P	K	temperature	humidity	ph	rainfall
135.225	21.90391	108.2182	22.255591	66.38508	4.631889	264.6474
124.2138	19.28092	25.02405	15.42698	95.8879	9.829598	102.9343
47.09306	109.2855	142.1804	28.2724	78.98857	5.77959	298.3498
107.2684	7.640329	120.6689	41.153255	88.39695	7.53867	44.77721
40.1617	129.3661		9.1643906	30.07124	9.200731	208.9162
96.23455	107.1161	171.9446	26.782171	50.61415	8.793912	55.49008
121.6445	87.57683	29.01266	31.436731	82.19354	4.062357	293.3981
86.75497	94.87949	106.7264	24.805208	34.16507	6.663124	54.50357
41.00509	17.60328	22.96365	9.8803498	77.4472		80.64829
46.4787	7.46267	143.6062	16.989264	82.87719	6.606696	218.6624
104.0513	17.62106		40.998747	64.5186	3.742532	139.8785
38.02581			30.386221	16.83398	5.030952	261.3976
85.68839	126.3525		27.872203	88.57107	4.721368	41.21712
133.7115	73.98994	113.396		15.32035	6.734751	105.4392
10.1192	100.9658	151.9367	36.839521	32.3834	5.005432	85.21655
106.1618	95.02864	41.79687		28.13905		174.9878
126.4804	124.7721	58.91236		35.84702	3.532603	257.5849

Figure 22: Dataset with filled missing values from `test50.csv`

3. Prediction Results:

After applying the classification algorithms, we compare the predicted labels across all three models. The results are summarized as follows:

- **Number of rows where all three predictions match:** 9
- **Number of rows where any two predictions match:** 33

N	P	K	temperature	humidity	ph	rainfall	knn_pred	knn_conf	tree_pred	tree_conf	svm_pred	svm_conf
135.225	21.90391	108.2182	22.25559	66.38508	4.631889	264.6474	rice	1	pomegranate	1	rice	0.305958
124.2138	19.28092	25.02405	15.42698	95.8879	9.829598	102.9343	cotton	1	pomegranate	1	mothbeans	0.145109
47.09306	109.2855	142.1804	28.2724	78.98857	5.77959	298.3498	rice	0.666667	grapes	1	papaya	0.179357
107.2684	7.640329	120.6689	41.15326	88.39695	7.53867	44.77721	papaya	1	muskmelon	1	papaya	0.234835
40.1617	129.3661	48.14909	9.164391	30.07124	9.200731	208.9162	chickpea	1	chickpea	1	mothbeans	0.135047
96.23455	107.1161	171.9446	26.78217	50.61415	8.793912	55.49008	chickpea	1	grapes	1	mothbeans	0.154357
121.6445	87.57683	29.01266	31.43673	82.19354	4.062357	293.3981	rice	1	rice	1	rice	0.185558
86.75497	94.87949	106.7264	24.80521	34.16507	6.663124	54.50357	chickpea	1	banana	0.7	chickpea	0.237091
41.00509	17.60328	22.96365	9.88035	77.4472	6.46948	80.64829	orange	1	mungbean	1	orange	0.467035
46.4787	7.46267	143.6062	16.98926	82.87719	6.606696	218.6624	rice	0.666667	grapes	1	papaya	0.228102
104.0513	17.62106	48.14909	40.99875	64.5186	3.742532	139.8785	mango	1	muskmelon	1	pigeonpeas	0.15246
38.02581	53.36273	48.14909	30.38622	16.83398	5.030952	261.3976	pigeonpeas	1	grapes	1	pigeonpeas	0.477947
85.68839	126.3525	48.14909	27.8722	88.57107	4.721368	41.21712	banana	1	banana	1	banana	0.222943
133.7115	73.98994	113.396	25.61624	15.32035	6.734751	105.4392	chickpea	0.666667	banana	0.7	chickpea	0.148744
10.1192	100.9658	151.9367	36.83952	32.3834	5.005432	85.21655	grapes	1	grapes	1	grapes	0.228147
106.1618	95.02864	41.79687	25.61624	28.13905	6.46948	174.9878	coffee	1	jute	1	pigeonpeas	0.201805
126.4804	124.7721	58.91236	25.61624	35.84702	3.532603	257.5849	rice	0.666667	rice	1	pigeonpeas	0.139842

Figure 23: Predicted labels from `test50_pred.csv`

10 Model Comparison – Classification

Classification Task: Crop Recommendation Dataset: Crop Recommendation Dataset (Multi-class)

Models Evaluated:

- KNN Classifier (k=3)
- Decision Tree (criterion='log_loss', max_depth=8, max_features='sqrt')
- SVM Classifier (C=10, kernel='rbf', gamma='scale')

Accuracy Results:

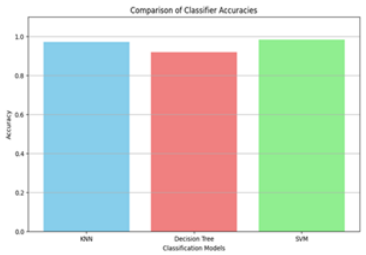

Figure 24: Model Comparison for Crop Recommendation Task

Model	Accuracy
SVM	98.03%
KNN	97.12%
Decision Tree	93.66%

Table 4: Model Comparison – Accuracy Results

Conclusion: SVM showed the best performance in terms of classification accuracy and generalization on the test data.

11 Model Comparison – Regression

Regression Task: Crop Prediction Dataset: Crop Prediction Dataset

Models Evaluated:

- KNN Regressor (k=3)
- Decision Tree Regressor (criterion='absolute_error', max_depth=12, max_features='log2')
- SVR (C=10, ϵ=0.1, kernel='rbf')

Evaluation Metrics:

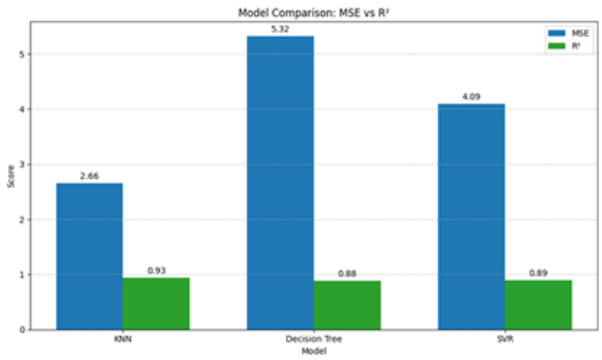

Figure 25: Regression Model Comparison for Crop Prediction Task

Model	MSE	R^2 Score
KNN	2.6554	0.9343
Decision Tree	5.3182	0.8786
SVR	4.0918	0.8941

Table 5: Model Comparison – Regression Results

KNN Regressor achieved the lowest Mean Squared Error (MSE) and the highest R^2 Score, making it best suited for this regression task.

12 Conclusion

- **SVM** is the best model for classification (high accuracy and generalization).
- **KNN** is the most effective for regression (lowest error, best fit).

13 References

- Crop Recommendation Dataset - Kaggle
- Scikit-learn Documentation
- Matplotlib Gallery
- Seaborn Tutorial

Key Metrics and Formulas

R^2 **Score:** The R^2 score measures how well the model explains the variance in the data. A score close to 1 means the model fits the data well.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Mean Squared Error (MSE): MSE calculates the average squared difference between actual and predicted values. Lower values indicate better performance.

$$MSE = \frac{1}{n}\sum(y_i - \hat{y}_i)^2$$

Classification Metrics

Accuracy: Accuracy is the ratio of correct predictions to total predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: Precision measures how many of the predicted positives were actually positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall: Recall measures how many of the actual positives were correctly identified.

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1 Score: F1 Score balances precision and recall, giving a single measure of performance.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Terms

- **True Positive (TP):** Correct positive prediction.
- **True Negative (TN):** Correct negative prediction.
- **False Positive (FP):** Incorrect positive prediction.
- **False Negative (FN):** Incorrect negative prediction.