Indian Institute of Technology, Dharwad

CS209 : Artificial Intelligence
And
CS214 : Artificial Intelligence Laboratory
Project Report : **Team-3**

Course Instructor:
Dr. Dileep A.D.

Mentor Name:
Anupama Bidargaddi

Submitted by:
1. Saipushkar Nagaraj
2. Utkarsh Raj
3. Sunny Raj
4. Nakirekanti Viraj

Abstract

This project focuses on an unsupervised approach to optimizing key agronomic factors that influence soybean growth to achieve maximal yield, discussing the methodology, implementation and results. Rather than relying on labeled data, we apply clustering techniques to categorize soyabean growth patterns into distinct levels based on a variety of phenotypic and genotypic variables.
The insights obtained from the model can be utilized for efficient and effective soyabean cultivation.

Contents

Chapter 1

Introduction

1.1 Overview

Artificial Intelligence (CS209) Course Project by Team-3, a hybrid approach combining supervised and unsupervised learning models to extract from an **Advanced Soybean Agriculture** dataset to optimize and detect optimal growth conditions.

Link to Repository : Repository

1.2 Introduction

Soybean [1] plays a crucial role in global agriculture, offering high-protein value and serving as a key crop in both food and industrial sectors. Enhancing soybean yield is of significant interest, but direct yield prediction is often hindered by the variability of environmental conditions and limited availability of labeled data. In such scenarios, unsupervised learning provides a powerful alternative to unravel hidden patterns within the data.
In this study, we employ various unsupervised learning techniques for clustering soyabean growth records into different levels based on a variety of environmental, genotypic and phenotypic variables.
These growth levels are further used to analyze the most influential input factors contributing to maximal yield. This approach emphazises on data-driven decision making without the need for explicit yield variables, making it broadly applicable in precision farming systems.

1.3 Plan of Action

The plan of action taken to achieve the above described objectives are:

- Employ Clustering Algorithms on the dataset to obtain growth levels dependent on the input parameters of phenotypes and genotypes.
  - Clustering algorithms such as KMeans and KMedoid are employed to cluster the data to obtain distinct growth levels.
  - Clustering is performed on different versions on the same data - input parameters *C, G, and S*, selective output parameters and the entirety of data.

  - Best performing clusters are evaluated by using various metrics like **Inertia Score**, **Silhoutte Score**, **Davies Bouldin Score**, **Calinski Harabasz Score**.

- Employ Classification Algorithms to predict the growth level of a given set of phenotypes and genotypes.
  - Classification Algorithms such as **XGBoost**, **SVM**, **RandomForest** are employed to build a classifier that can analyze given set of inputs and predict the yield for said inputs.
  - Best hyper-parameters are determined using **GridSearchCV**. The models so formed are evaluated using metrics like **Accuracy Score Precision Score**, **Recall Score**, **F1-Score** and **Cross-Validation Score**.

- Analyze the data from the above models and extract meaningful insights regarding the optimization of parameters to maximise the yield.
  - Analysis Methods like **SHAP score** are used to establish

1.4 Dataset Description

This dataset consists of 55,450 rows and 13 columns, capturing essential agricultural parameters related to soybean plants. The dataset includes the following key attributes:

- **Plant Height** – Measures the growth of the soybean plant.
- **Number of Pods** – The count of soybean pods per plant.
- **Biological Weight** – The total biomass of the plant.
- **Chlorophyll Content** – Indicates the plant's photosynthetic efficiency.
- **Protein Percentage** – The percentage of protein in the soybean seeds.
- **Seed Yield** – The total soybean seed production.
- **Relative Water Content** – The water retention capacity of leaves.

A key feature of this dataset is the **Parameters** column, which encodes experimental conditions affecting soybean growth. The parameters include:

- **G (Genotype)**: Six different soybean genotypes.
- **C (Salicylic Acid)**: Three levels – 250 mg, 450 mg, and a control level.
- **S (Water Stress)**: Two levels –
  - Water stress at 5% of field capacity.
  - Water stress at 70% of field capacity.

Chapter 2

Methodology

2.1 Data Cleaning & Preprocessing

2.1.1 Handling Null values:

The dataset does not contain null values in any of the fields and thus no preprocessing with regards to null values was performed

2.1.2 Handling Duplicate Values:

Among the **55,450** rows, it was reported that **55,342** records were duplicates. A fresh modified dataset was thus made upon removing the duplicate records.

```
data_original
```

	Parameters	Random	Plant Height (PH)	Number of Pods (NP)	Biological Weight (BW)	Sugars (Su)	Relative Water Content in Leaves (RWCL)	ChlorophyllA663	Chlorophyllb649	Protein Percentage (PPE)	Weight of 300 Seeds (W3S)	Leaf Area Index (LAI)	Seed Yield per Unit Area (SYUA)	Number of Seeds per Pod (NSP)	Protein Content (PCO)
0	C1S1G5	R1	50.5	130.3	111	0.433	0.732	1.4	3.1	33.2	33.6	0.08	5567.4	1.86	0.82
1	C2S3G4	R1	44.5	132.0	80	0.334	0.674	7.1	2.2	38.5	34.4	0.09	2245.5	1.87	0.13
2	C2S1G6	R1	52.2	150.0	83	0.490	0.677	1.8	1.3	33.6	35.2	0.07	4326.7	2.04	0.15
3	C1S1G1	R1	50.5	140.8	66	0.163	0.745	1.1	3.0	33.5	52.3	0.09	6214.5	2.20	0.14
4	C1S2G6	R2	49.2	175.6	73	0.795	0.725	7.5	2.1	39.6	31.2	0.10	3897.8	1.88	0.60
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
55445	C1S1G2	R1	52.4	136.0	223	0.523	0.713	1.9	3.3	38.2	38.1	0.07	5216.2	2.06	0.50
55446	C1S1G6	R1	45.7	145.0	111	0.466	0.756	1.4	2.2	37.7	28.8	0.09	3211.2	1.84	0.44
55447	C1S2G2	R1	50.0	166.3	85	0.523	0.676	8.5	1.3	36.5	35.2	0.08	4358.8	2.20	0.58
55448	C1S2G3	R3	44.0	209.0	132	0.663	0.875	9.0	1.3	39.7	31.2	0.06	4472.7	2.60	0.32
55449	C2S3G3	R1	50.0	127.0	124	0.322	0.612	9.9	2.1	35.4	33.3	0.04	5123.6	1.86	0.40

55450 rows × 15 columns

(a) Original DataFrame

```
data_original.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55450 entries, 0 to 55449
Data columns (total 15 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   Parameters                             55450 non-null  object
 1   Random                                 55450 non-null  object
 2   Plant Height (PH)                      55450 non-null  float64
 3   Number of Pods (NP)                    55450 non-null  float64
 4   Biological Weight (BW)                 55450 non-null  int64
 5   Sugars (Su)                            55450 non-null  float64
 6   Relative Water Content in Leaves (RWCL)  55450 non-null  float64
 7   ChlorophyllA663                        55450 non-null  float64
 8   Chlorophyllb649                        55450 non-null  float64
 9   Protein Percentage (PPE)               55450 non-null  float64
 10  Weight of 300 Seeds (W3S)              55450 non-null  float64
 11  Leaf Area Index (LAI)                  55450 non-null  float64
 12  Seed Yield per Unit Area (SYUA)        55450 non-null  float64
 13  Number of Seeds per Pod (NSP)          55450 non-null  float64
 14  Protein Content (PCO)                  55450 non-null  float64
dtypes: float64(12), int64(1), object(2)
memory usage: 6.3+ MB
```

(b) DataFrame Info

```
data_original.duplicated().sum()
```

```
np.int64(55342)
```

(c) Duplicated Data Values

Figure 2.1.1: Overview of Original Dataset and Its Analysis

2.1.3 Eliminating redundant fields:

Redundant fields such as ***Parameters*** (Upon which feature engineering is performed) and ***Random*** were eliminated to reduce redundancy.

2.1.4 Standardization of fields:

All Attributes were standardised to eliminate any form of bias when employing clustering or classification algorithms on this data.

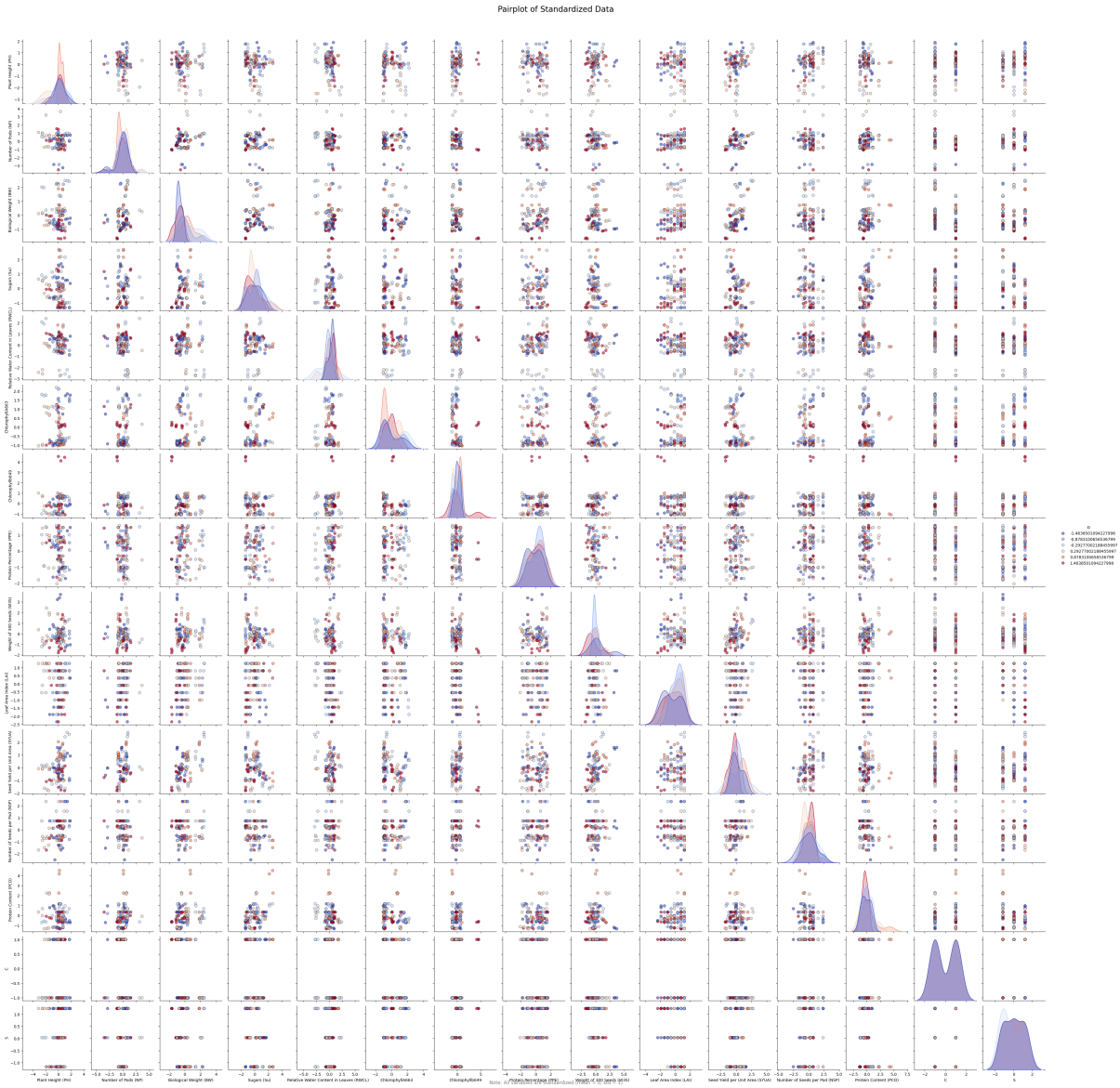


Figure 2.1.2: Standardized dataset

2.1.5 Handling Outliers:

As can be seen from 2.2.2, outliers exist in the data, but due to the following reasons, no preprocessing techniques were applied to tackle them:

- Owing to the presence of very less data points (2.1.1c), removing these outliers was an expensive choice as the number of data points left behind would be far too less to perform analysis.
- Owing to the presence of very less data points (2.1.1c), imputing these outliers with standard tendencies or applying regression techniques would incurr loss of relationship of these variables with other parameters in the dataset.

2.2 Statistical Inferences

A correlation heatmap was generated to visualize the strength of linear relationships between numeric variables. Strong positive or negative correlations highlight variables that move together or in opposite directions. Identifying highly correlated features is crucial, as they can impact model performance due to multicollinearity.

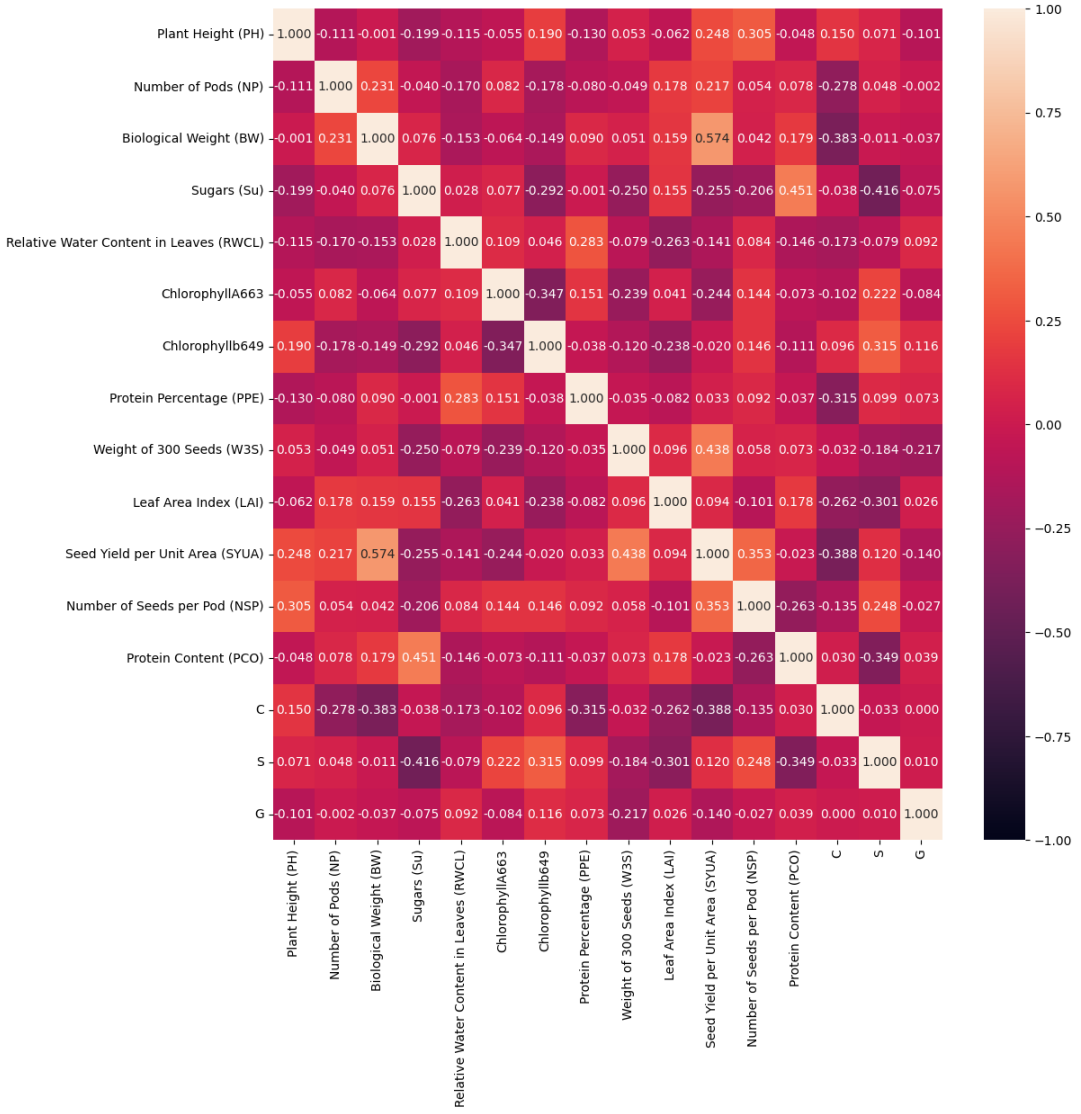


Figure 2.2.1: Correlation HeatMap

Boxplots were utilized to detect the presence of outliers.

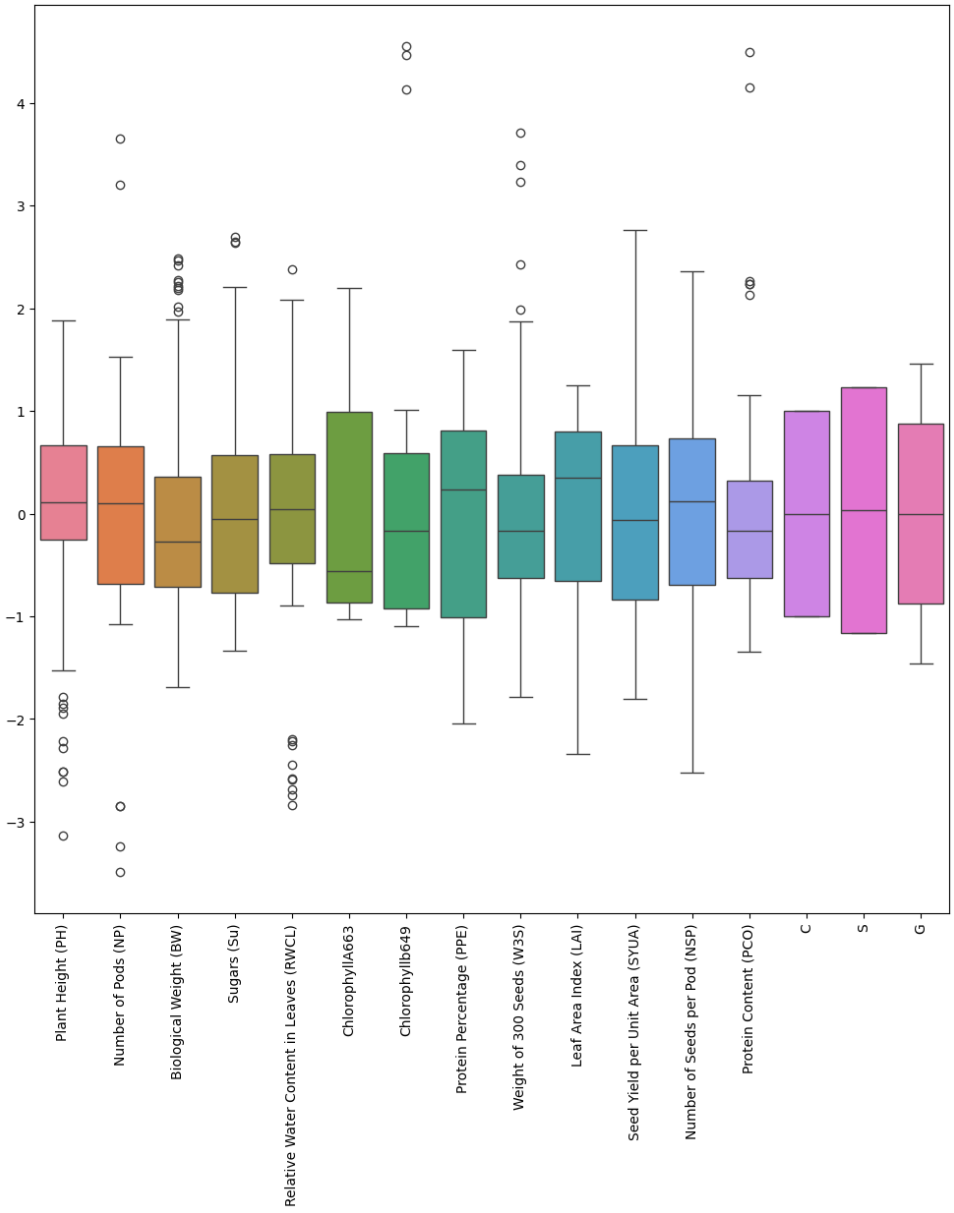


Figure 2.2.2: Box Plot of Numerical Attributes

A pairplot was created to analyze pairwise relationships between all numerical variables. The diagonal elements show distributions, while the off-diagonal scatter plots reveal trends, clusters, or potential outliers.

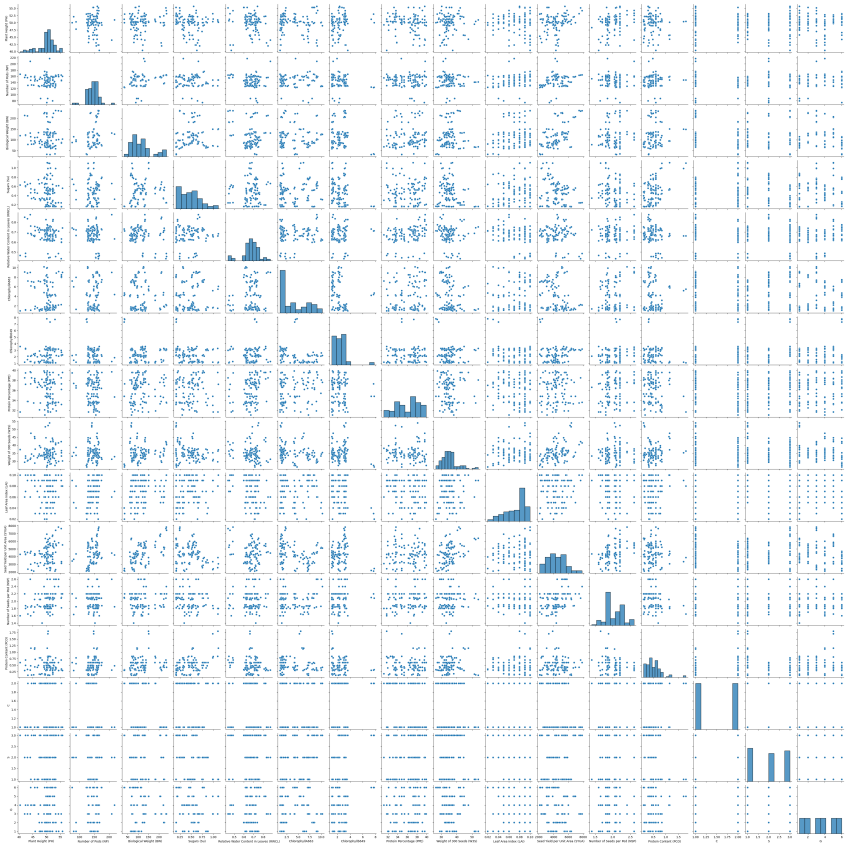


Figure 2.2.3: Pair Plot before Standardizing

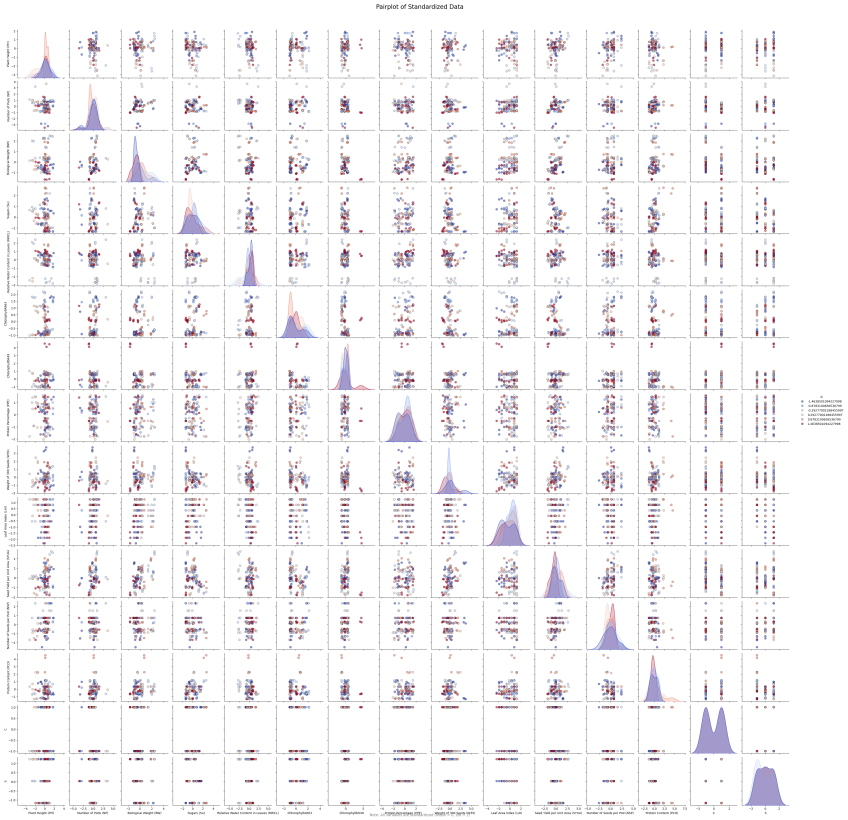


Figure 2.2.4: Pair Plot after standardizing

The distribution of each numerical feature was plotted to understand its underlying shape. Most features were found to follow (normal/skewed) distributions as the plot was made on the Normalized dataset, guiding further preprocessing steps.

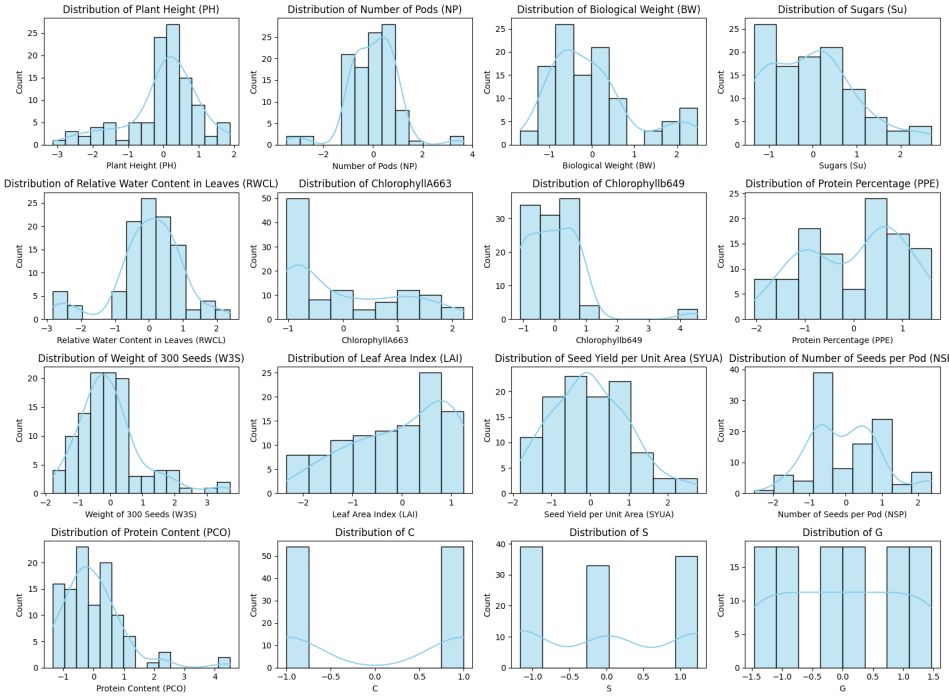
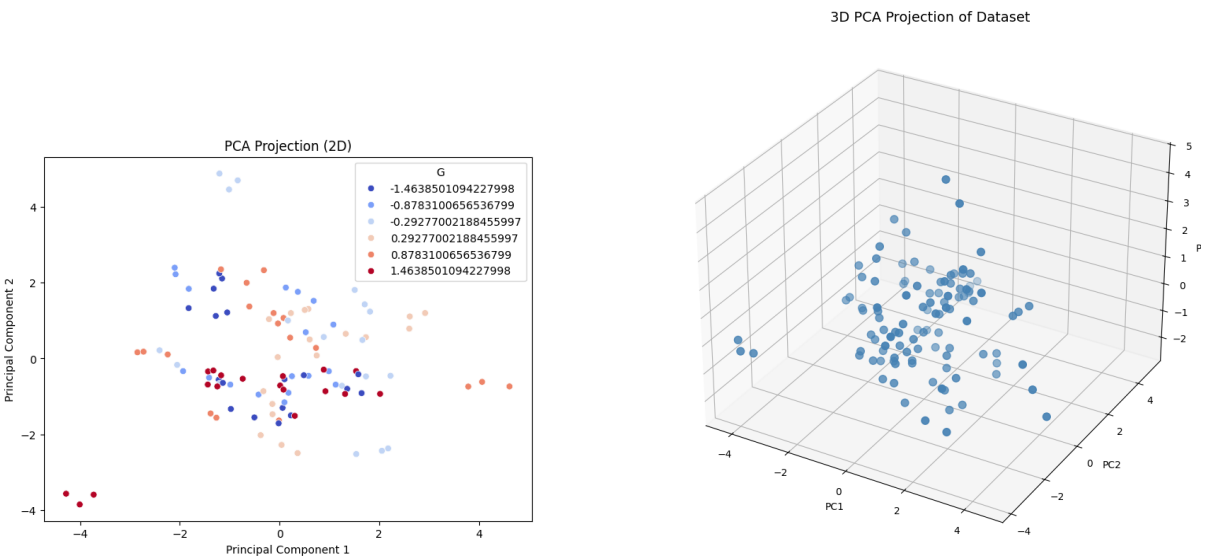


Figure 2.2.5: Distribution of all Numerical Features



(a) 2-D projection of the entire dataset

(b) 3D-Projection of the entire dataset

Figure 2.2.6: Projection of the dataset onto lower dimensions for easy visualization

2.3 Feature Engineering

2.3.1 Ordinal-Encoding of the Parameters Attribute:

The ***Parameters*** Attribute that had the parameters - **C**,**G** and **S** were extracted and encoded into seperate fields, namely **C**,**G** and **S**.

2.3.2 One-Hot-Encoding of the Parameters Attribute:

Once Ordinal Encoding on the ***Parameters*** attribute is performed, One-Hot-Encoding is performed to obtain seperate fields for each class a given *Parameter*

2.3.3 Mutual Information Regression

To extract meaningful insights certain parameters were chosen to be **optimized** in-order for a higher yield output. They are

- SYUA - Seed Yield per Unit Area
- Number of Seeds per Pod
- Sugars (SU)
- Protein Content (PCO)

Mutual Information Regression plots were made to determine the effect and impact of other parameters on these ***target*** variables.

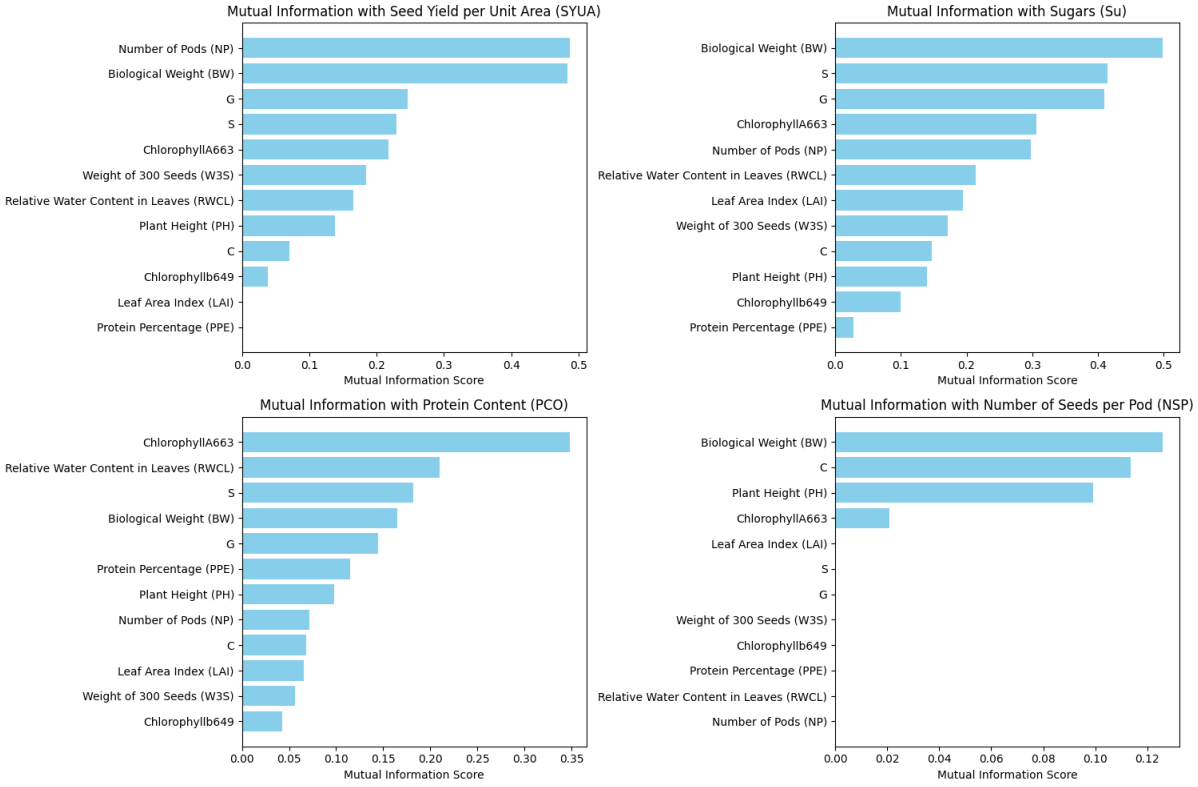


Figure 2.3.1: Mutual Information Regression Plot

2.4 Clustering

2.4.1 Algorithms used

Clustering was performed in different phases by employing **KMeans** [2] and **KMedoid** [3] Clustering techniques. Other sophisticated clustering methods such as **DBSCAN**, **Agglomerative Clustering** were avoided due to the sparsity of data leading to poor performance from these algorithms.

2.5 Implementation

2.5.1 Techniques employed for visualization:

TSNE (T-distributed Stochastic Neighbor Embedding)

t-SNE (t-Distributed Stochastic Neighbor Embedding) [4] is a technique to reduce the dimensions of data, mainly for visualization, while keeping similar points close together.

PCA (Principle Component Analysis)

A versatile feature engineering tool with many applications, one of which is to reduce the dimensions of data with minimal loss. [5]

2.5.2 Phase-1: KMeans Clustering was applied on all parameters

KMeans Clustering techniques on the data set that was reduced to lower dimensions using **PCA** (Principle Component Analysis) and **TSNE** (T-distributed Stochastic Neighbor Embedding).
The effective number of clusters was identified using **Elbow-method** upon plotting **Inertia** against the number of clusters.

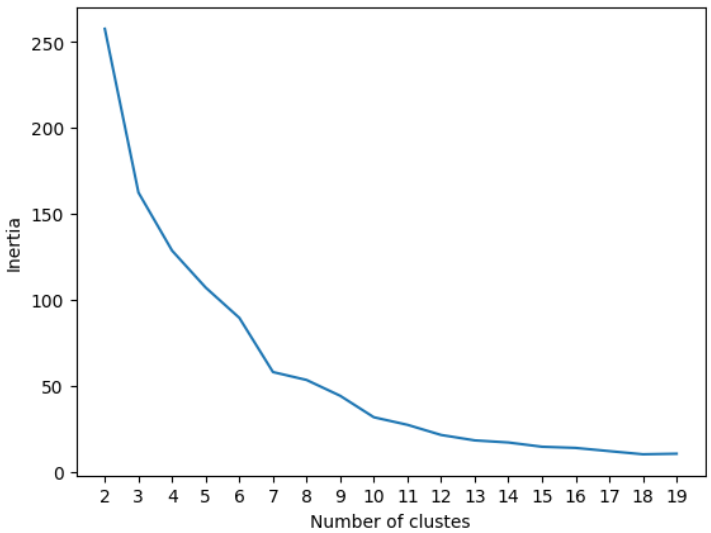
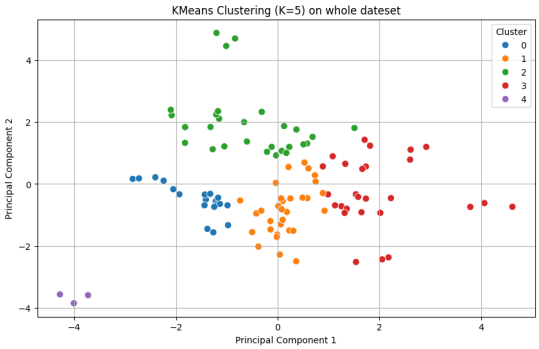
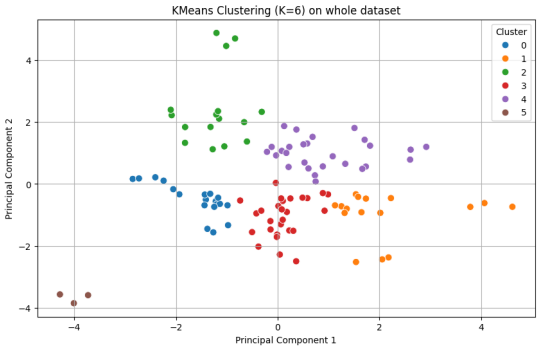


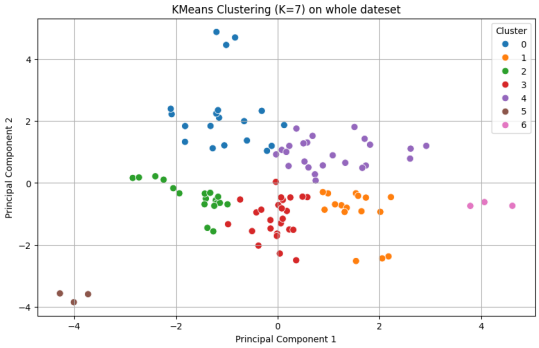
Figure 2.5.1: Elbow Curve Method Analysis for optimal Clusters



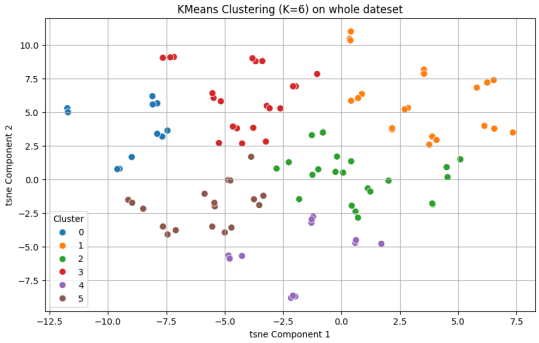
(a) KMeans - 5 Clusters (PCA)



(b) KMeans - 6 Clusters (PCA)



(c) KMeans - 7 Clusters (PCA)



(d) t-SNE - 6 Clusters

Figure 2.5.2: Cluster visualizations using PCA and t-SNE

2.5.3 Phase-2: KMeans Clustering on All Output Parameters

Kmeans and KMedoid Clustering Techniques were applied on all ***Ouput parameters*** in order to test the hypothesis *"Distinct Growth Levels can be established by clustering all output parameters"*.
We have defined **output parameters** as those that are measured after the soyabean crop has been harvested. From the dataset, we observe that these parameters are namely,

'Plant Height (PH)', 'Number of Pods (NP)', 'Biological Weight (BW)', 'Sugars (Su)', 'Relative Water Content in Leaves (RWCL)', 'ChlorophyllA663', 'Chlorophyllb649', 'Protein Percentage (PPE)', 'Weight of 300 Seeds (W3S)', 'Leaf Area Index (LAI)', 'Seed Yield per Unit Area (SYUA)', 'Number of Seeds per Pod (NSP)', 'Protein Content (PCO)' .

The effective number of clusters was determined by using **Elbow Method** upon plotting inertia values against number of clusters.

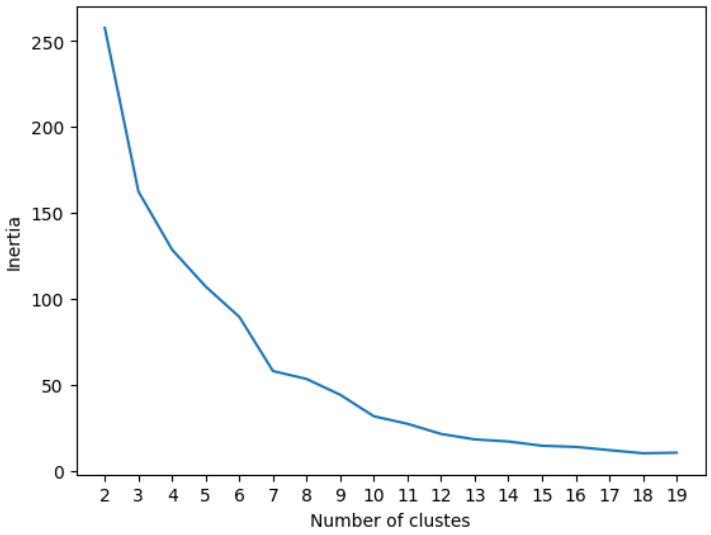
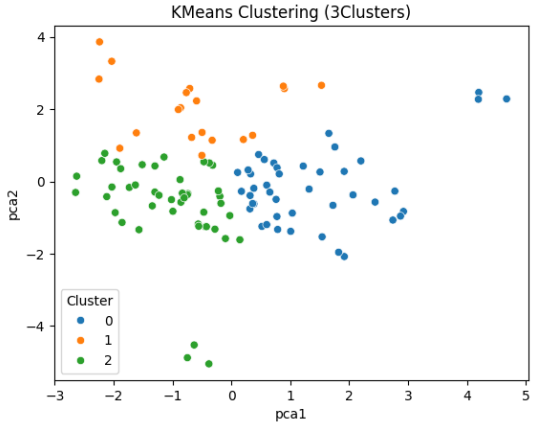
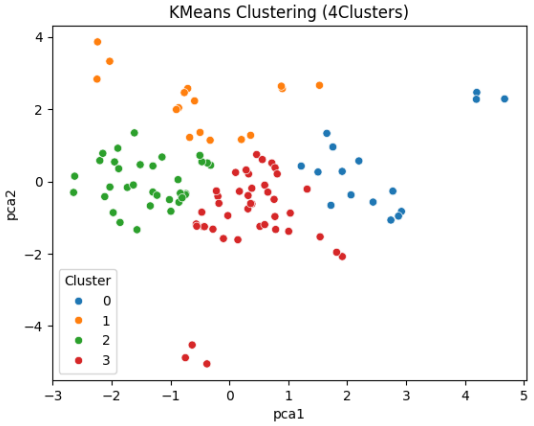


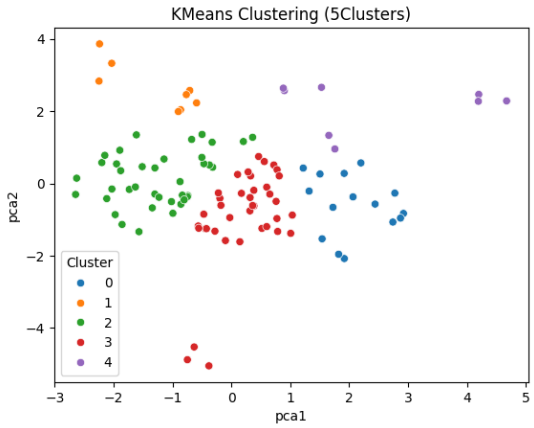
Figure 2.5.3: Elbow Curve Method Analysis for optimal Clusters



(a) KMeans - 3 Clusters (PCA) on output parameters



(b) KMeans - 4 Clusters (PCA) on output parameters



(c) KMeans - 5 Clusters (PCA) on output parameters

(d) KMeans - 6 Clusters (PCA) on output parameters

Figure 2.5.4: Cluster visualizations using PCA

2.5.4 Phase-3: Applying Gaussian Mixture Model Clustering on PCA data

Gaussian Mixture Model is a probabilistic model that works under certain assumptions:

- Data is generated from a mixture of several Gaussian Normal Distributions, each with its own mean and variance.
- A data point belongs to each cluster with a certain probability.

Model Assumption

Given K clusters, the probability density function for a data point $\mathbf{x}$ is:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

where:

- π_k is the mixing coefficient for cluster k ($\sum_{k=1}^{K} \pi_k = 1$),
- $\boldsymbol{\mu}_k$ is the mean of cluster k,
- $\boldsymbol{\Sigma}_k$ is the covariance matrix of cluster k,
- $\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the multivariate normal distribution.

Training Using Expectation-Maximization (EM) Algorithm

GMMs are typically trained using the Expectation-Maximization (EM) algorithm, which consists of two main steps:

- **E-Step (Expectation Step):** Compute the responsibilities $\gamma(z_{ik})$ indicating the probability that point i belongs to cluster k:

$$\gamma(z_{ik}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

- **M-Step (Maximization Step):** Update the parameters using the responsibilities:

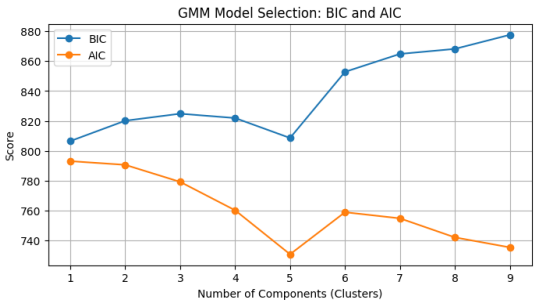
$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik}) \mathbf{x}_i$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik})(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T$$

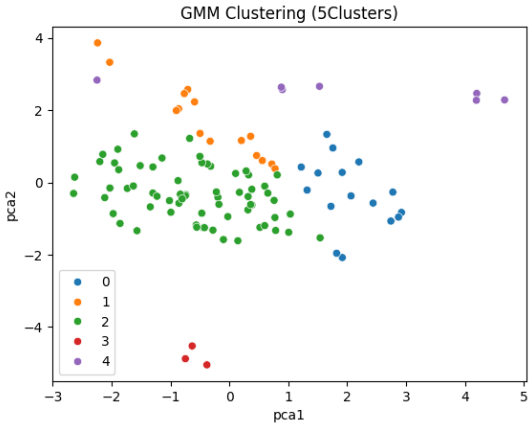
$$\pi_k = \frac{N_k}{N}$$

where $N_k = \sum_{i=1}^{N} \gamma(z_{ik})$.

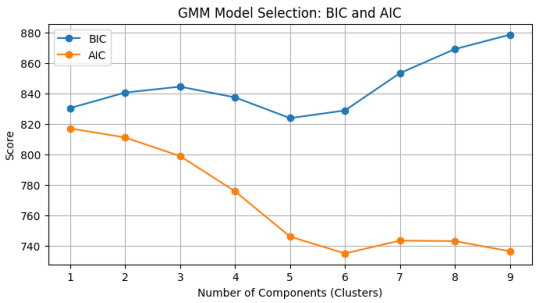
The **Gaussian Mixture Model** was trained on PCA-2 data, and the best clustering model was chosen using metrics like **BIC** and **AIC**.



(a) Optimal Cluster using GMM on Output Parameters



(b) Optimal Clustering for 5 Cluster Components



(c) Optimal Cluster using GMM on All Parameters

(d) Optimal Clustering for 6 Cluster Components

Figure 2.5.5: Cluster visualizations using PCA

2.5.5 Phase-4: Applying K-Means and K-Medoids Clustering with $k = 2$

Given the small dataset size (only 108 unique samples), applying clustering with more than two clusters resulted in fewer samples per group, making it impractical for reliable training and analysis. Therefore, $k = 2$ was selected as a balanced trade-off to allow downstream supervised learning and explainability.

Why $k = 2$?

- The dataset contains a limited number of unique samples.
- Higher values of k would result in very small cluster sizes, which are unsuitable for robust model training.

- Visualizations of production metrics like *Seed Yield per Unit Area (SYUA)* and *Protein Percentage (PPE)* showed clear separation when $k = 2$ was used.

To support the choice of $k = 2$ for both clustering methods (K-Means and K-Medoids), we analyze:

- Silhouette scores for various k values.
- Scatter plots of the clusters in reduced dimensions.

K-Means ($k = 2$)

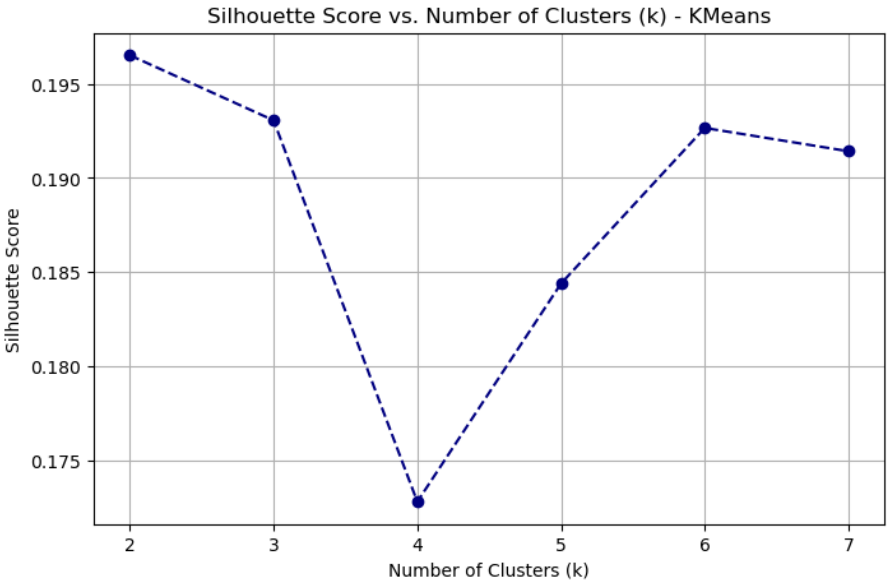


Figure 2.5.6: Silhouette Score vs k for K-Means Clustering

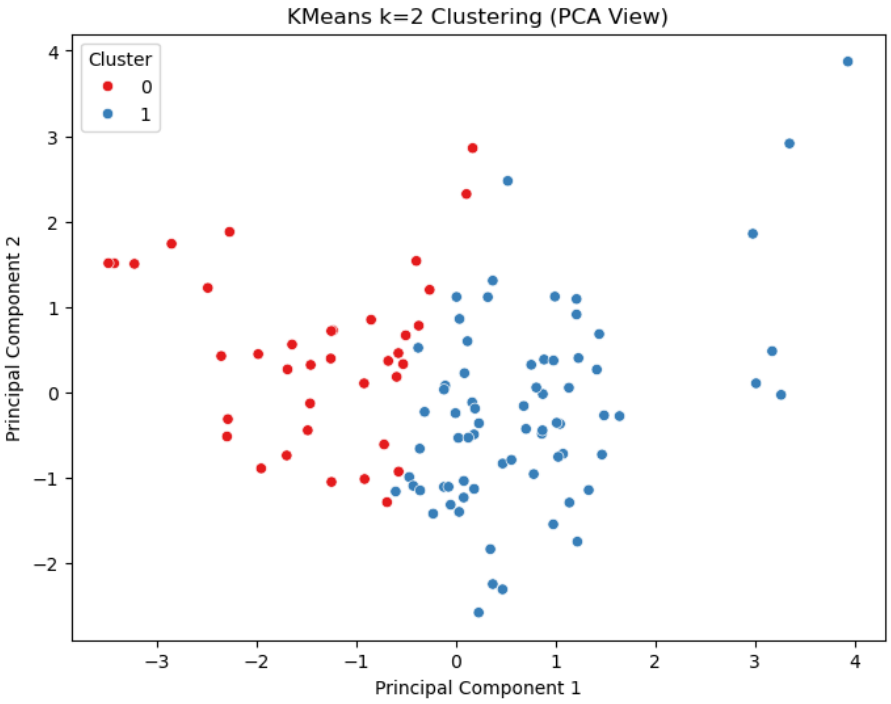


Figure 2.5.7: Cluster Scatter Plot for K-Means with $k = 2$

K-Medoids ($k = 2$)

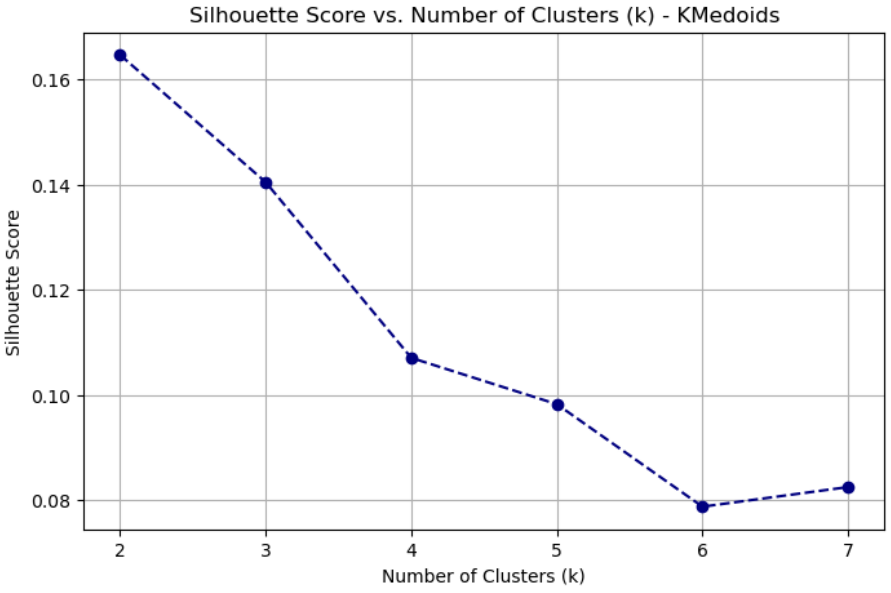


Figure 2.5.8: Silhouette Score vs k for K-Medoids Clustering

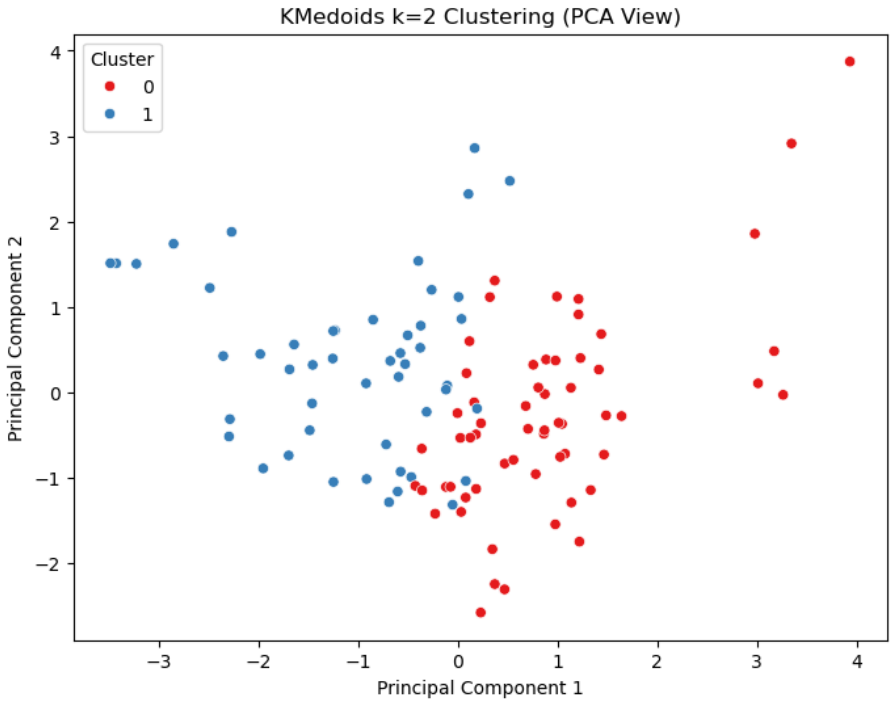


Figure 2.5.9: Cluster Scatter Plot for K-Medoids with $k = 2$

2.6 Evaluation

Various different metrics were used in order to evaluate the effectiveness of the clusters like

- Inertia Values using **Elbow Method** depicted by J in equation 2.1

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \|x_n - \mu_k\|^2 \tag{2.1}$$

  z_{nk} is 1 if $x_n \in$ Cluster k, else 0

- Silouhette Score: [6] For a single data point i, Silouhette score is measured as:

$$s(i) = \frac{b(i) - a(i)}{\mathbf{max}(a(i), b(i))} \tag{2.2}$$

  Where $\mathbf{a(i)}$ is the average distance from the point iii to all other points within the same cluster. This represents the "tightness" or intra-cluster cohesion, $\mathbf{b(i)}$ is the smallest average distance from point iii to all points in any other cluster. This captures the inter-cluster separation.

- Davies Bouldin Score: [7]

$$\text{DB} = \frac{1}{N} \sum_{i=1}^{N} \max_{j \neq i} \left(\frac{S_i + S_j}{M_{ij}} \right) \tag{2.3}$$

  where,

$$S_i = \frac{1}{|C_i|} \sum_{x \in C_i} \|x - \mu_i\|_2 \quad \text{and} \quad M_{ij} = \|\mu_i - \mu_j\|_2$$

with:

- N = Number of clusters
- S_i = Average intra-cluster distance (how compact cluster i is)
- μ_i = Centroid of cluster i
- M_{ij} = Distance between centroids of clusters i and j

The Davies-Bouldin score measures the average "worst-case" similarity between each cluster and its most similar one, based on the ratio of intra-cluster distances to inter-cluster distances.

A **lower Davies-Bouldin score** indicates better clustering: clusters are compact and well-separated.

- Calinski Harabasz Score: [8]

$$\text{CH} = \frac{\text{Tr}(B_k)}{\text{Tr}(W_k)} \times \frac{N-k}{k-1} \tag{2.4}$$

where,

- N = Total number of data points
- k = Number of clusters
- B_k = Between-cluster dispersion matrix
- W_k = Within-cluster dispersion matrix

The terms are defined as:

$$\text{Tr}(B_k) = \sum_{i=1}^{k} n_i \|\mu_i - \mu\|_2^2 \quad \text{and} \quad \text{Tr}(W_k) = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|_2^2$$

where:

- n_i = Number of points in cluster i
- μ_i = Centroid of cluster i
- μ = Overall mean of the data

The **Calinski-Harabasz Score** measures how well-separated and dense the clusters are.

A **higher Calinski-Harabasz score** indicates better-defined clusters.

- Bayesian Information Criterion (BIC): [9] The BIC similarly balances model fit and complexity, but penalizes model complexity more strongly. It is defined as:

$$\text{BIC} = k \ln(n) - 2 \ln(L) \tag{2.5}$$

where:

- n is the number of data points,

- k is the number of parameters,
- L is the maximum likelihood.

A lower BIC value indicates a better model.

- Akaike Information Criterion (AIC): [10] The AIC balances model fit and complexity by penalizing the number of parameters in the model. It is given by:

$$\text{AIC} = 2k - 2\ln(L) \tag{2.6}$$

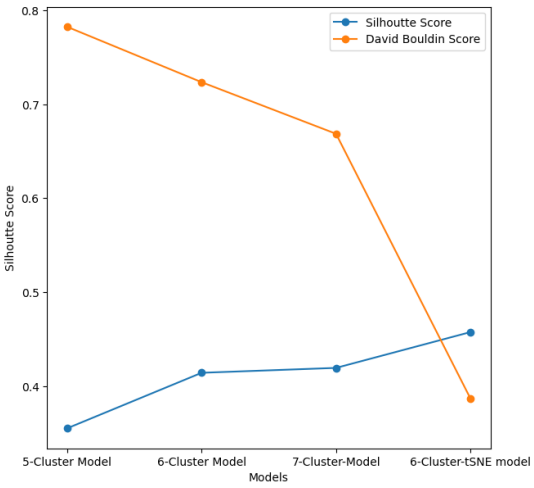
where:

- k is the number of parameters in the model,
- L is the maximum value of the likelihood function for the model.

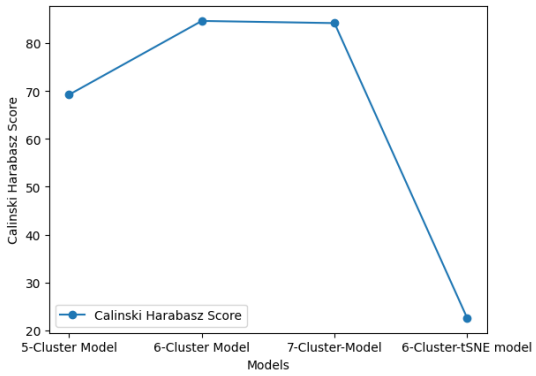
A lower AIC value indicates a better model.

2.6.1 Phase 1 Results:

Visualizing the metrics corresponding to each of the clusters helps filter out clusters that are not efficient, this way the **KMeans-6 Cluster Model** on all parameters was found to perform the best.



(a) Silouhtette & Bouldin Scores

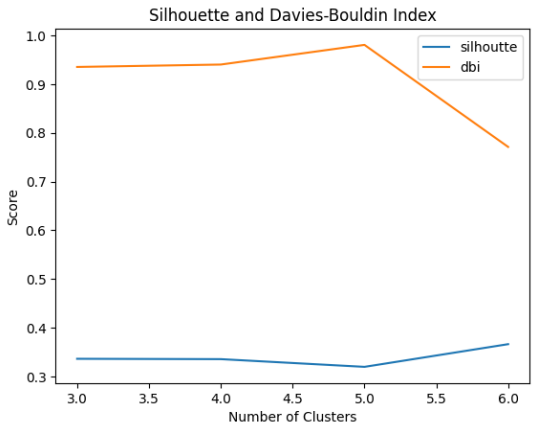


(b) Calinski's Score

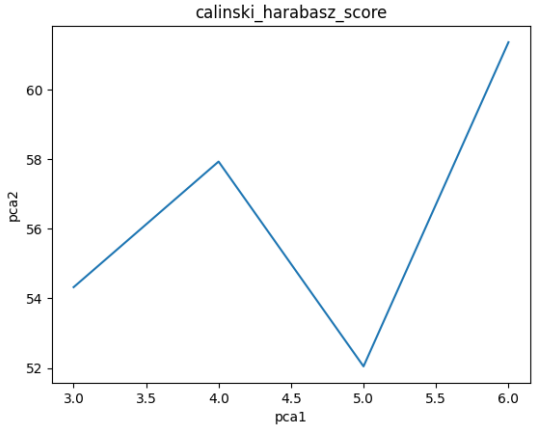
Figure 2.6.1: Result evaluation of PCA-2 on all parameters

2.6.2 Phase 2 Results:

Visualizing the metrics corresponding to each of the clusters helps filter out clusters that are not efficient, this way the **KMeans-6 Cluster Model** on all output parameters was found to perform the best.



(a) Silouhtette & Bouldin Scores



(b) Calinski's Score

Figure 2.6.2: Result evaluation of PCA-2 on output parameters

2.6.3 Phase 3 Results:

In the case of the **Gaussian Mixture Model**, Bayesia Information Criterion (BIC) and Akaike Information Criterion (AIC) were employed to identify the most optimal clusters. Lower values of both AIC and BIC indicate a better model.

- In the case where only output parameters were considered **refer:2.5.5a**, optimal number of Clusters turned out to be **5** because of a low BIC and AIC score.
- In the case where all parameters were considered, **refer:2.5.5c**, optimal number of Clusters turned out to be **6** because of a low BIC and AIC score.

2.6.4 Why K-Means and K-Medoids with *k=2* Were Chosen

For clustering the soybean samples based on phenotype and genotype features, we selected $k = 2$ for both **K-Means** and **K-Medoids** due to the following reasons:

- **Interpretability:** Boxplots of selected output parameters across clusters showed clear and meaningful separation between the two classes. This provided a biologically interpretable split into "high production" and "low production" categories.
- **Dataset Size:** The dataset consisted of only 108 unique observations. Using more than 2 clusters would lead to fewer samples per cluster, making it impractical for downstream classification and analysis tasks.
- **Robustness:** K-Medoids, being less sensitive to outliers compared to K-Means, was especially useful when dealing with real-world agricultural data where minor noise or measurement error is common.

2.6.5 Visual Comparison of Cluster Quality for K-Means and K-Medoids (k = 2)

To visually support the choice of $k = 2$ for both K-Means and K-Medoids, we present the boxplots for key production traits. The following figures demonstrate clear separation between clusters for both clustering methods.

K-Means ($k = 2$)

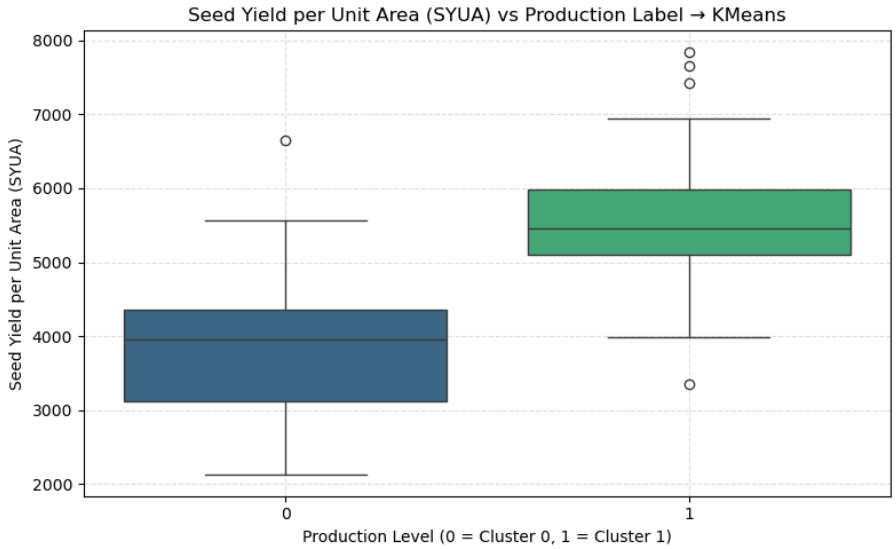


Figure 2.6.3: Boxplot of Seed Yield per Unit Area (SYUA) across K-Means Clusters ($k = 2$)

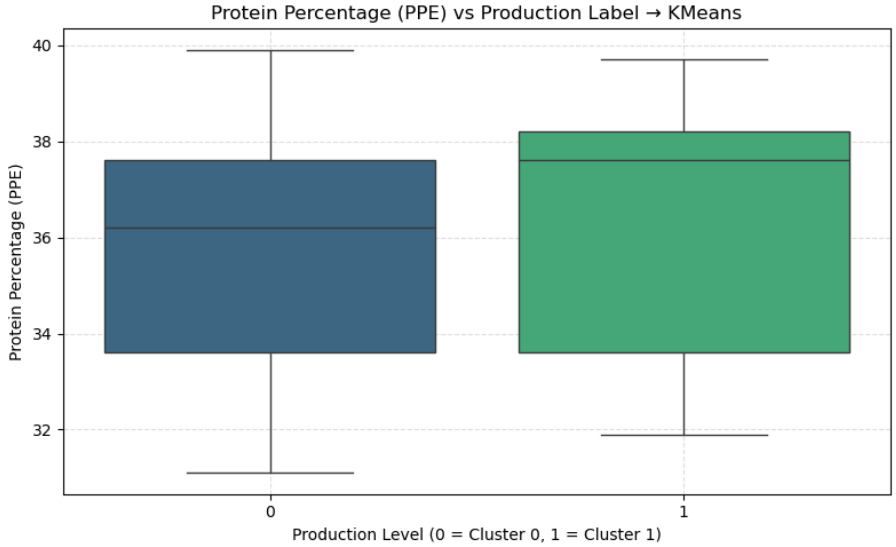


Figure 2.6.4: Boxplot of Protein Percentage (PPE) across K-Means Clusters ($k = 2$)

K-Medoids ($k = 2$)

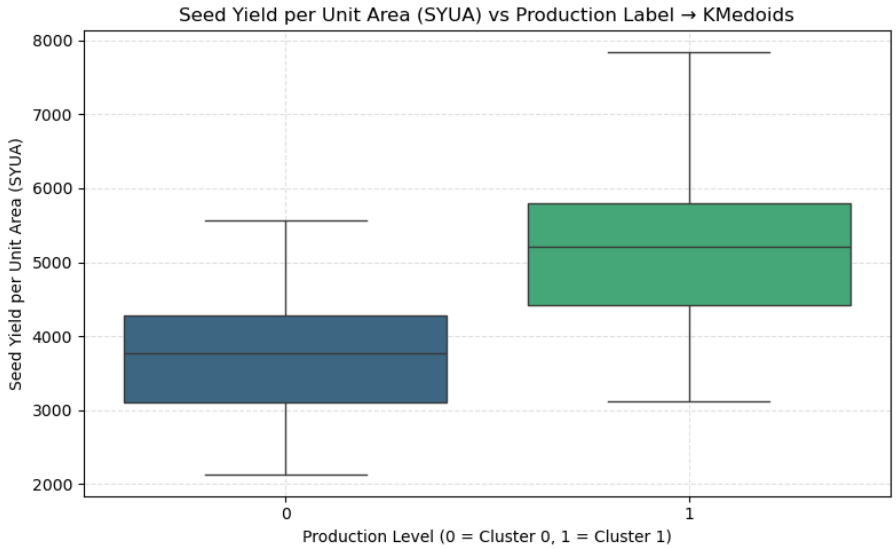


Figure 2.6.5: Boxplot of Seed Yield per Unit Area (SYUA) across K-Medoids Clusters ($k = 2$)

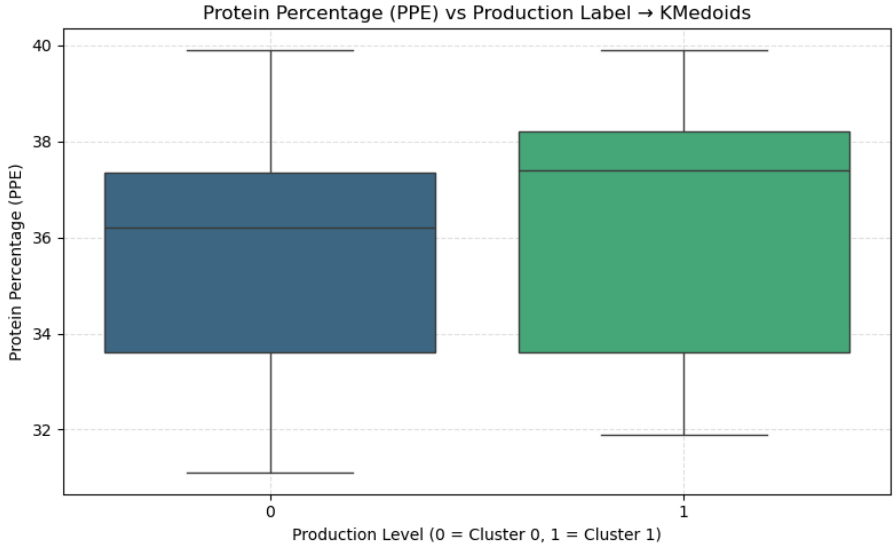


Figure 2.6.6: Boxplot of Protein Percentage (PPE) across K-Medoids Clusters ($k = 2$)

2.7 Classification Algorithms

The following classification algorithms were used to implement classification algorithms to classify the input data into distinct growth levels.

- **Random Forest Classifier**
- **SVM**
- **XGBoost Classifier**
- **Neural Networks**

Upon performing **SHAP** and **Statistical Analysis** of the different clusters that were made **ref:??**, we observed the **2-Cluster Model** that was obtained by applying KMeans and KMedoid Algorithms on select output parameters had the most effective results in terms of obtaining distinct growth levels from the clusters.

2.8 Model Training and Testing

2.8.1 Preparing Data for classification task

- Owing to the small number of data points, we have split the dataset into **Train** and **Test** sets, with an **80-20** split without an explicit validation set.
- **GridSearchCV** algorithm was used to find the most optimal hyper-parameters to be used in the classification algorithm. **GridSearchCV** performs implicit validation checks to determine optimal parameters hence eliminating the need to have an explicit validation set.

2.8.2 Training the Algorithm

- The classification algorithms mentioned above (2.7) were trained on 3 different sets of training data, namely
  - Training data consisting of only the input parameters, i.e, **C**,**G** and **S** parameters.
  - Training data consisting of only the Output Parameters (2.5.3)
  - Training data consisting of both of the above sets of variables.
  - The above steps were repeated for both KMeans(2) Clustered Data as well as KMedoid(2) Clustered Data.

2.8.3 Random Forest Algorithm

Mathematical Background

Random Forest is an ensemble learning method used for classification and regression. It builds multiple decision trees and combines their outputs to improve prediction accuracy and control overfitting.

Let the training dataset be:

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}, \quad \text{where } x_i \in \mathbb{R}^d$$

For each decision tree T_k:

- A bootstrap sample $\mathcal{D}_k$ is drawn from $\mathcal{D}$.
- At each split in the tree, a random subset of features $\mathcal{F}_k \subset \{1, 2, ..., d\}$ is considered.
- The best split is chosen using a splitting criterion such as:
  - **Gini Impurity**: $G = 1 - \sum_{i=1}^{C} p_i^2$
  - **Entropy**: $H = -\sum_{i=1}^{C} p_i \log p_i$

The final prediction for a test sample x is obtained via majority voting:

$$\hat{\mathbf{y}} = \text{mode}\left(T_1(x), T_2(x), ..., T_K(x)\right)$$

Hyperparameters used to train the models and their Roles

- **n_estimators**: Number of trees in the forest.
- **criterion**: Metric used to evaluate split quality; either `gini` or `entropy`.
- **max_depth**: Maximum depth of each tree to prevent overfitting.
- **min_samples_split**: Minimum number of samples required to split a node.
- **min_samples_leaf**: Minimum number of samples required at a leaf node.

2.8.4 Support Vector Machine (SVM)

Support Vector Machines (SVMs) are powerful supervised learning algorithms used for classification tasks. SVM works by finding the hyperplane that best separates data points from different classes with the maximum margin.

Mathematical Formulation

For a linearly separable dataset, SVM solves the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to: } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i \end{aligned} \tag{2.7}$$

Here:

- $\mathbf{w}$ is the normal vector to the hyperplane,
- b is the bias term,
- $y_i \in \{-1, 1\}$ are the class labels,
- $\mathbf{x}_i$ are the input data points.

For non-linearly separable data, SVM introduces slack variables ξ_i and uses kernel functions:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{2.8}$$
$$\text{subject to: } y_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b) \geq 1-\xi_i, \quad \xi_i \geq 0$$

Where $\phi(\cdot)$ is the transformation function applied through a kernel.

Hyperparameters Used

- **kernel**: Specifies the kernel type to be used:
  - `linear`: linear hyperplane
  - `poly`: polynomial kernel $K(x, x') = (x \cdot x' + c)^d$
  - `rbf`: Radial Basis Function (Gaussian) $K(x, x') = \exp(-\gamma\|x - x'\|^2)$
  - `sigmoid`: Sigmoid kernel function
- **C**: Regularization parameter that controls the trade-off between achieving a low error on the training data and minimizing the norm of the weights.
- **degree**: Degree of the polynomial kernel function (used only for `poly`).
- **gamma**: Defines the influence of a single training example in `rbf`, `poly`, and `sigmoid`.

Implementation Description

In this project, SVM was trained using different kernel functions. A 5-fold Stratified Cross-Validation approach was adopted to evaluate performance metrics. For the polynomial kernel, a hyperparameter tuning process was done using GridSearchCV with:

- degree = 2
- C = 0.1
- gamma = auto

2.8.5 XGBoost Classifier

XGBoost (Extreme Gradient Boosting) is an optimized implementation of the gradient boosting algorithm, which is widely used for classification and regression tasks due to its high performance and scalability.

Mathematical Formulation

Given a dataset with n instances and m features, the prediction at iteration t is:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i), \quad f_t \in \mathcal{F}$$

where f_t is a decision tree from the functional space $\mathcal{F}$.
The objective function to minimize is:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k)$$

Here, l is a differentiable convex loss function (e.g., logistic loss), and $\Omega(f_k)$ is the regularization term given by:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

where:

- T is the number of leaves in the tree
- w_j are the weights of the leaves
- γ and λ are regularization parameters

XGBoost uses second-order Taylor approximation to efficiently optimize the objective.

Hyperparameter Descriptions

- **n_estimators**: Number of boosting rounds (trees).
- **learning_rate**: Step size shrinkage to prevent overfitting.
- **max_depth**: Maximum depth of each tree.
- **min_child_weight**: Minimum sum of instance weights in a child node.
- **gamma**: Minimum loss reduction required to make a further partition.
- **subsample**: Fraction of samples to be used for fitting the trees.
- **colsample_bytree**: Fraction of features to be used for each tree.
- **reg_alpha**: L1 regularization term.
- **reg_lambda**: L2 regularization term.
- **scale_pos_weight**: Controls balance of positive/negative classes.
- **booster**: Type of booster to use, usually 'gbtree'.
- **eval_metric**: Evaluation metric to be optimized.

Model Configurations

Model 1: KMeans (Input Parameters)

```
colsample_bytree=1, gamma=0, learning_rate=0.1, max_depth=3,
n_estimators=150, reg_alpha=0, reg_lambda=1, subsample=1
```

Model 2: KMeans (Output Parameters)

```
colsample_bytree=0.9, gamma=0, learning_rate=0.6, max_depth=4,
n_estimators=180, reg_alpha=0.05, reg_lambda=3, subsample=0.9
```

Model 3: KMeans (All Parameters)

```
learning_rate=0.2, max_depth=4, n_estimators=190,
eval_metric='error', booster='gbtree', min_child_weight=1,
colsample_bytree=0.8, subsample=0.9, gamma=0, scale_pos_weight=1
```

Model 4: KMedoid (Input Parameters)

```
learning_rate=0.05, max_depth=2, n_estimators=150
```

Model 5: KMedoid (Output Parameters)

```
learning_rate=0.05, max_depth=2, n_estimators=150
```

Model 6: KMedoid (All Parameters - Fully Tuned)

```
seed=100, n_estimators=100, learning_rate=0.075, gamma=0.0,
colsample_bytree=0.7, reg_alpha=0, reg_lambda=0.005,
max_depth=6, min_child_weight=1, subsample=0.6
```

This model was tuned in multiple stages:

- *Stage 1*: max_depth and min_child_weight
- *Stage 2*: gamma
- *Stage 3*: subsample and colsample_bytree
- *Stage 4*: reg_alpha and reg_lambda
- *Stage 5*: learning_rate

Each stage involved cross-validation to select the best performing parameter.

2.8.6 Neural Networks

Introduction

A **Neural Network (NN)** is a supervised machine learning model inspired by the structure of the human brain. It consists of layers of interconnected *neurons* (nodes), where each neuron applies a transformation to the input data. Neural networks are particularly useful for modeling complex, non-linear relationships.

Mathematical Formulation

Let $\mathbf{x} \in \mathbb{R}^n$ be the input vector, and let the neural network contain L layers. Each layer l computes:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{w_o}^{(l)}$$
$$\mathbf{a}^{(l)} = \phi^{(l)}(\mathbf{z}^{(l)})$$

where:

- $\mathbf{a}^{(0)} = \mathbf{x}$ is the input,
- $\mathbf{W}^{(l)}$ is the weight matrix of layer l,
- $\mathbf{w_o}^{(l)}$ is the bias vector of layer l,
- $\phi^{(l)}$ is the activation function (e.g., ReLU, tanh),
- $\mathbf{a}^{(l)}$ is the output (activation) of layer l.

The output of the final layer is used to compute a loss $\mathcal{L}(\hat{y}, y)$ between the predicted $\hat{y}$ and true label y, commonly using cross-entropy for classification tasks.

Training Objective

Training aims to minimize the empirical risk:

$$\min_{\{\mathbf{W}^{(l)}, \mathbf{w_o}^{(l)}\}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}_i), y_i)$$

where $f(\mathbf{x}_i)$ is the NN's prediction for input $\mathbf{x}_i$, and N is the number of training samples. Optimization is typically performed using variants of stochastic gradient descent (SGD).

Activation Functions Used

In this experiment, the following activation functions were tested:

- `identity`: $\phi(x) = x$
- `logistic` (sigmoid): $\phi(x) = \frac{1}{1+e^{-x}}$
- `tanh`: $\phi(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
- `relu`: $\phi(x) = \max(0, x)$

Hyperparameters Used

The `MLPClassifier` was configured with the following key hyperparameters:

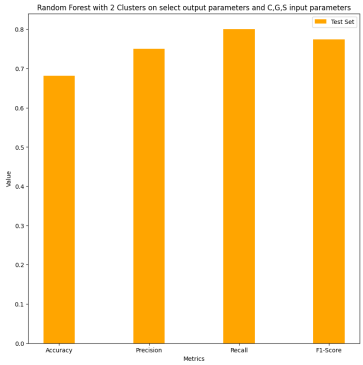
- `hidden_layer_sizes=(100,)`: One hidden layer with 100 neurons.
- `activation`: One of `'identity'`, `'logistic'`, `'tanh'`, or `'relu'`.
- `max_iter=1000`: Maximum number of iterations for training.
- `random_state=42`: For reproducibility.

Cross-Validation and Evaluation

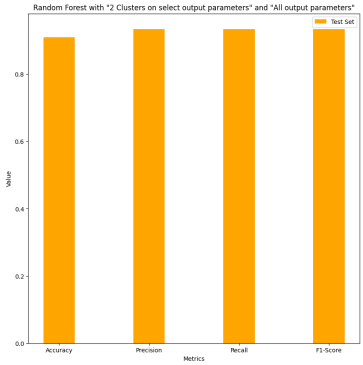
- **Stratified 5-Fold Cross-Validation** was used to evaluate performance while preserving class distribution.
- Metrics collected per fold: Accuracy, Precision, Recall, and F1 Score.

2.9 Testing Algorithms & Performance Metrics

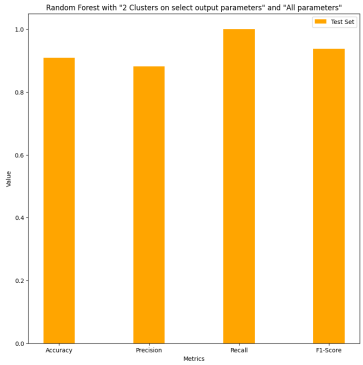
2.9.1 Random Forest Algorithm



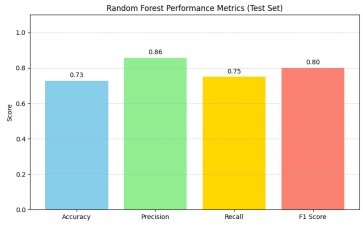
(a) Random Forest on CGS Kmeans



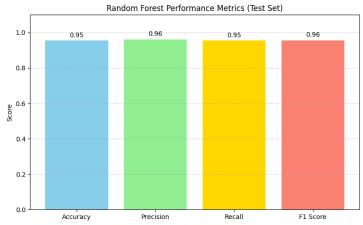
(b) Random Forest on output Kmeans



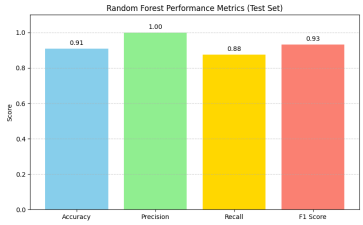
(c) Random Forest on all parameters using Kmeans



(d) Random Forest on CGS parameters using Kmedoid



(e) Random Forest on output Kmedoid



(f) Random Forest on all parameters using Kmedoid

Figure 2.9.1: Evaluation for Random Forest Classifier

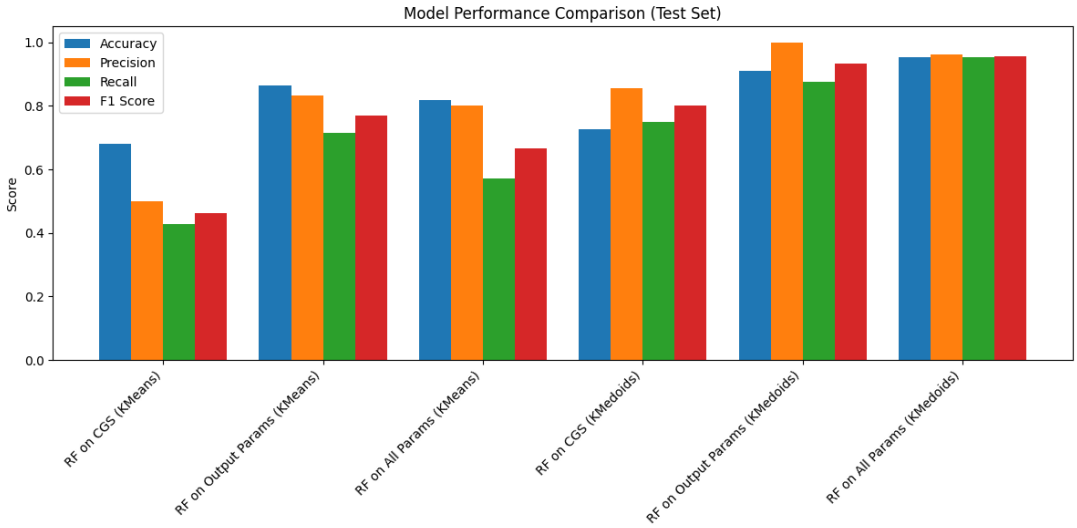


Figure 2.9.2: Evaluation Metrics for Random Forest Algorithm

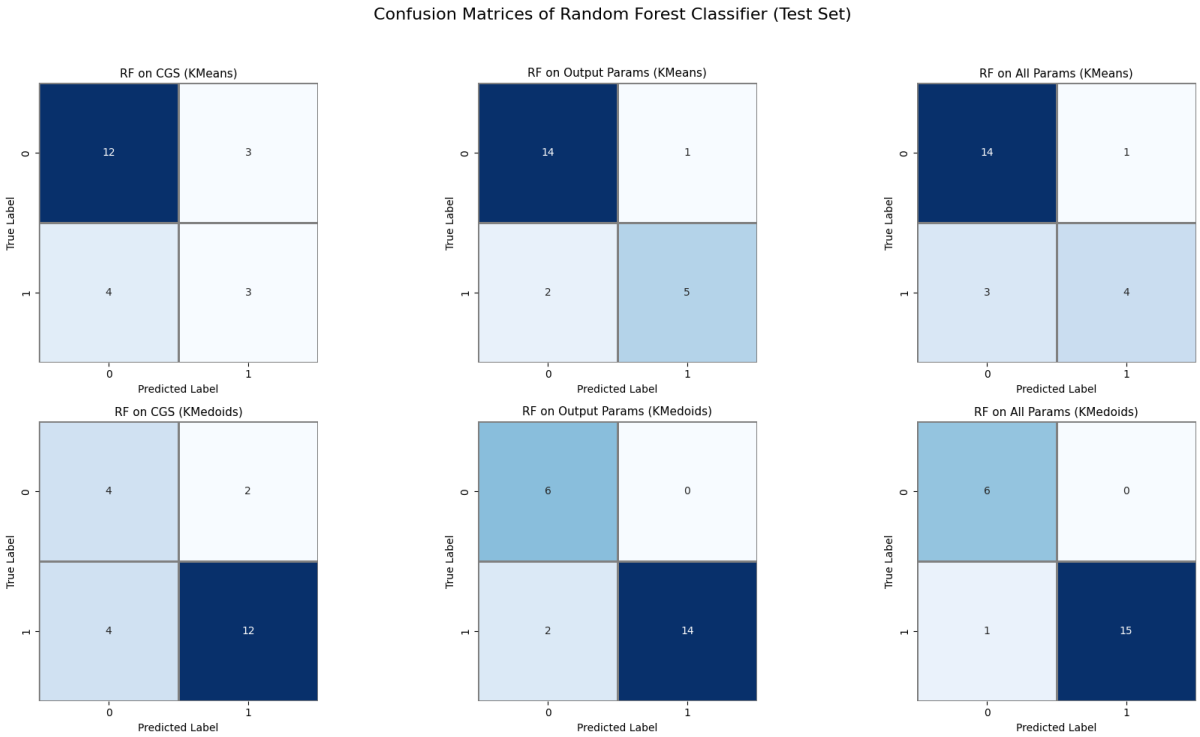


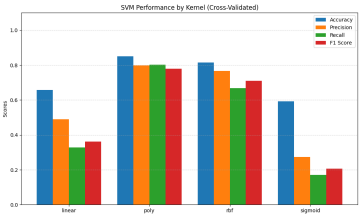
Figure 2.9.3: Confusion Matrix Plot of all Results from Random Forest Algorithm

Model	Accuracy	Precision	Recall	F1 Score
RF on CGS (KMeans)	0.6818	0.5000	0.4286	0.4615
RF on Output Params (KMeans)	0.8636	0.8333	0.7143	0.7692
RF on All Params (KMeans)	0.8182	0.8000	0.5714	0.6667
RF on CGS (KMedoids)	0.7273	0.8571	0.7500	0.8000
RF on Output Params (KMedoids)	0.9091	1.0000	0.8750	0.9333
RF on All Params (KMedoids)	**0.9545**	**0.9610**	**0.9545**	**0.9556**

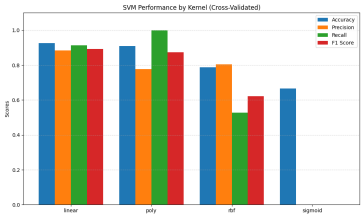
Table 2.9.1: Evaluation Metrics for Random Forest Classifier on KMeans and KMedoids

Observation: The best overall performance was observed for the Random Forest classifier trained on **all parameters clustered using KMedoids**, achieving a test accuracy of 95.45% and an F1-score of 0.9556.

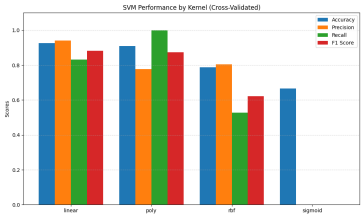
2.9.2 Support Vector Machine (SVM) Algorithm



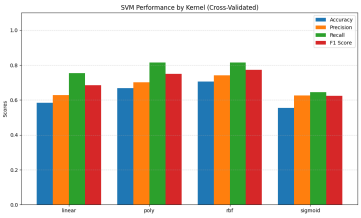
(a) SVM on CGS Kmeans



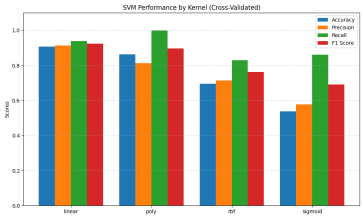
(b) SVM on output Kmeans



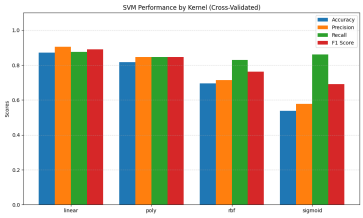
(c) SVM all params Kmeans



(d) SVM on CGS parameters using Kmedoid



(e) SVM on output Kmedoid



(f) SVM all params Kmedoid

Figure 2.9.4: Evaluation for Support Vector Machine Classifier

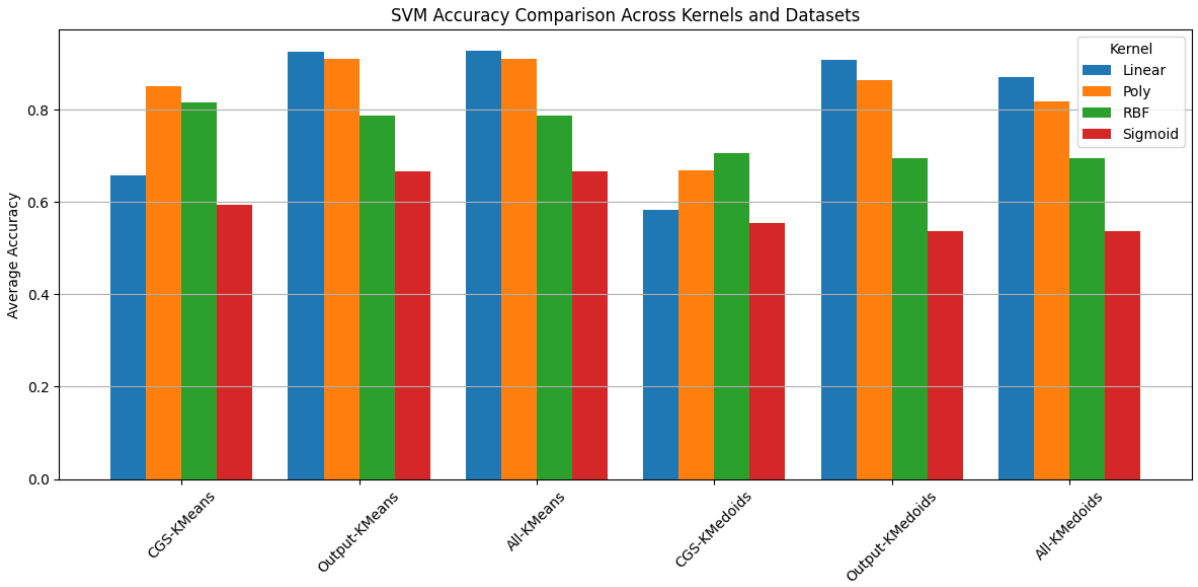


Figure 2.9.5: SVM accuracy plot across all models

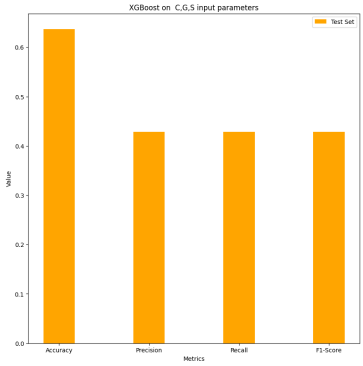
Dataset	**Linear**	**Polynomial**	**RBF**	**Sigmoid**
CGS (KMeans)	0.6571	0.8511	0.8147	0.5935
Output Params (KMeans)	0.9255	0.9091	0.7870	0.6667
All Params (KMeans)	**0.9264**	0.9091	0.7870	0.6667
CGS (KMedoids)	0.5835	0.6684	0.7052	0.5541
Output Params (KMedoids)	0.9082	0.8636	0.6957	0.5377
All Params (KMedoids)	0.8714	0.8182	0.6957	0.5377

Table 2.9.2: Average Accuracy of SVM Classifiers Across Different Kernels and Clustering Strategies.

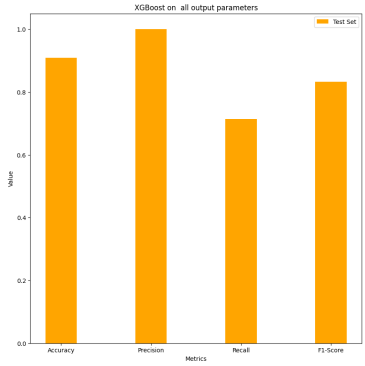
Observation: Based on the average accuracy scores, the best performing model is the **Support Vector Machine with a linear kernel**, applied to the **KMeans-clustered dataset using all parameters**, achieving an average accuracy of **0.9264**. This model outperformed all others across both clustering techniques and kernel types.

.

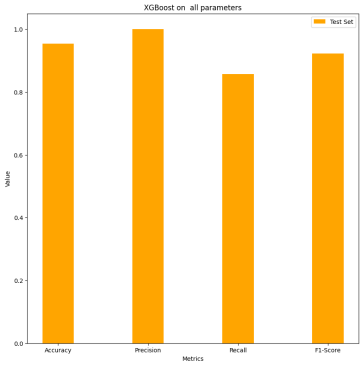
2.9.3 XGBoost Classifier



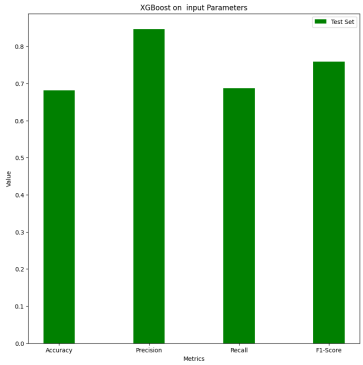
(a) XGBoost on CGS Kmeans



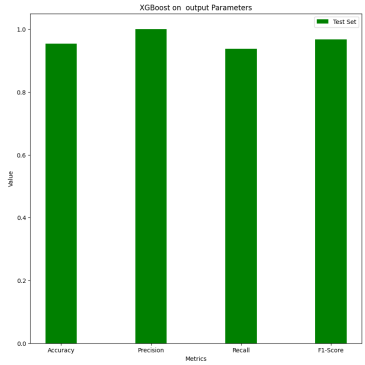
(b) XGBoost on output Kmeans



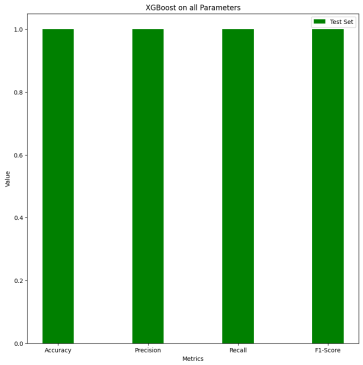
(c) XGBoost all params Kmeans



(d) XGBoost on CGS parameters using Kmedoid



(e) XGBoost on output Kmedoid



(f) XGBoost all params Kmedoid

Figure 2.9.6: Evaluation for XGBoost Classifier

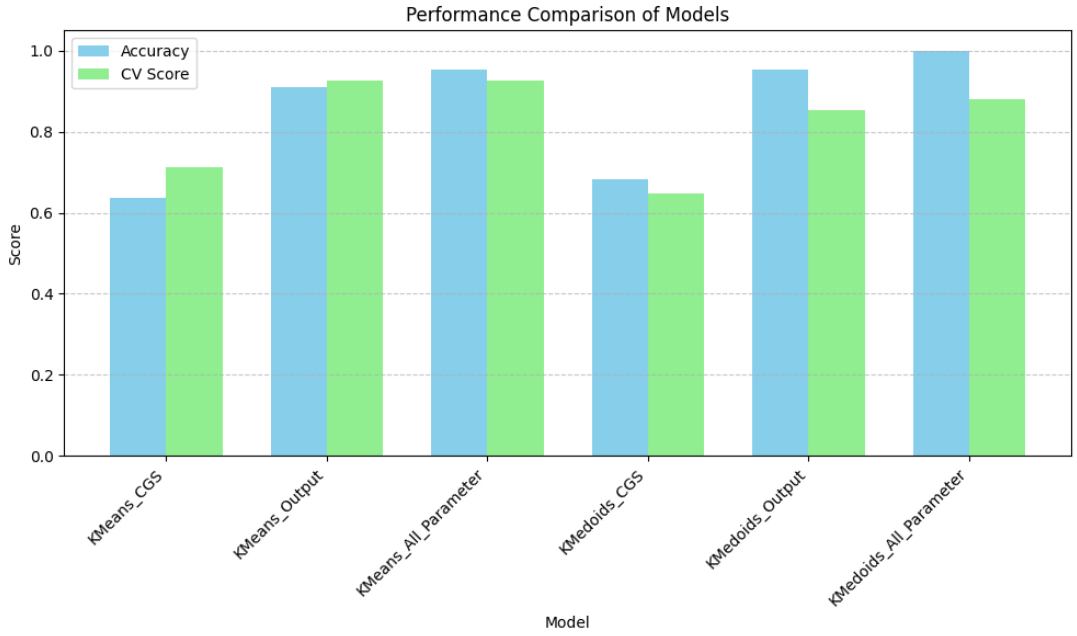


Figure 2.9.7: XGBoost Evaluation of Cross Val Score and Accuracy over all models

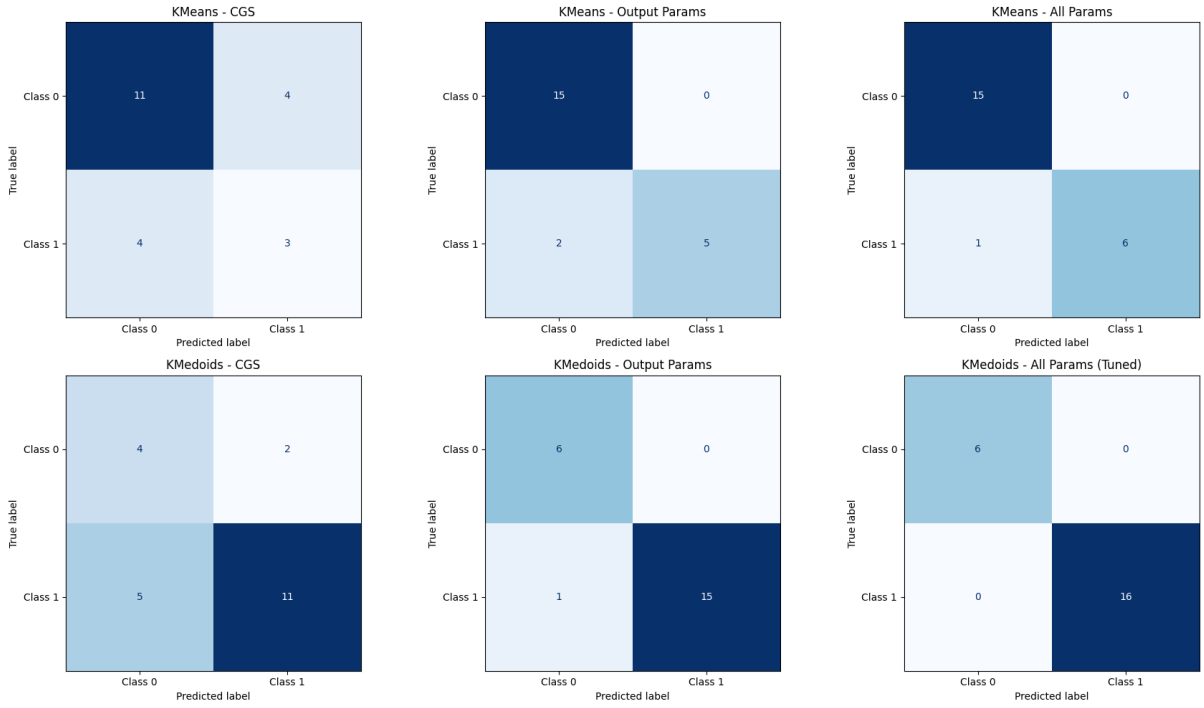
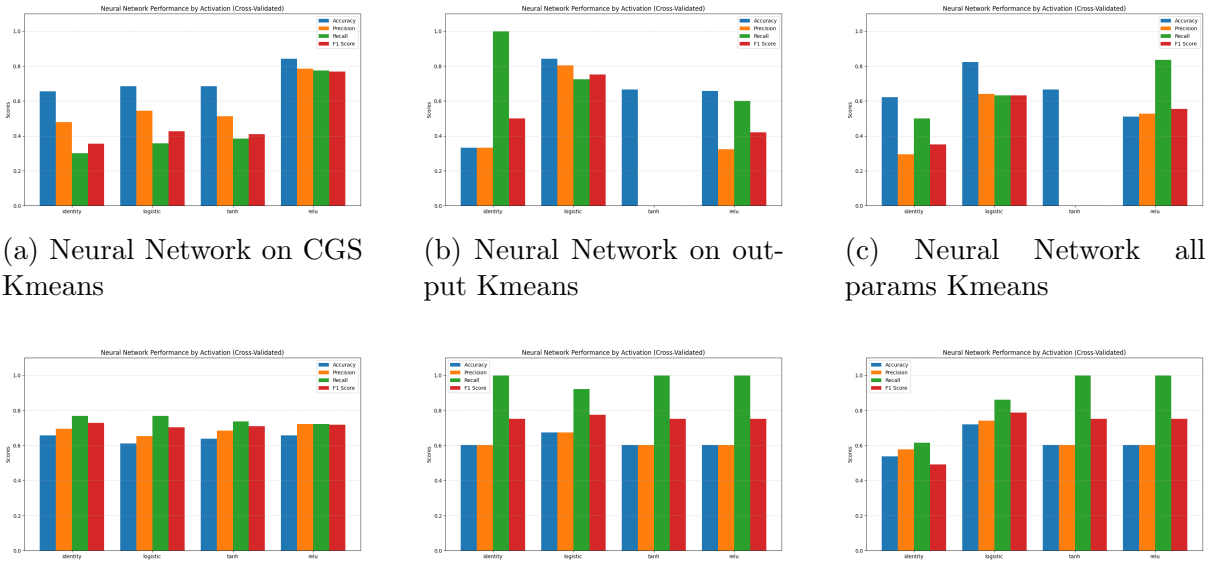


Figure 2.9.8: XGBoost Evaluation by Confusion Matrix over all models

Dataset	Accuracy	Precision	Recall	F1 Score	CV Score
KMeans - CGS	0.6364	0.4286	0.4286	0.4286	0.7121
KMeans - Output Params	0.9091	1.0000	0.7143	0.8333	0.9264
KMeans - All Params	0.9545	1.0000	0.8571	0.9231	0.9264
KMedoids - CGS	0.6818	0.8462	0.6875	0.7586	0.6481
KMedoids - Output Params	0.9545	1.0000	0.9375	0.9677	0.8519
KMedoids - All Params	1.0000	1.0000	1.0000	1.0000	0.8792

Table 2.9.3: XGBoost Performance Metrics Across Datasets and Parameter Sets

2.9.4 Neural Networks



(a) Neural Network on CGS Kmeans

(b) Neural Network on output Kmeans

(c) Neural Network all params Kmeans

(d) Neural Network on CGS parameters using Kmedoid

(e) Neural Network on output Kmedoid

(f) Neural Network all params Kmedoid

Figure 2.9.9: Evaluation for Neural Networks Classifier

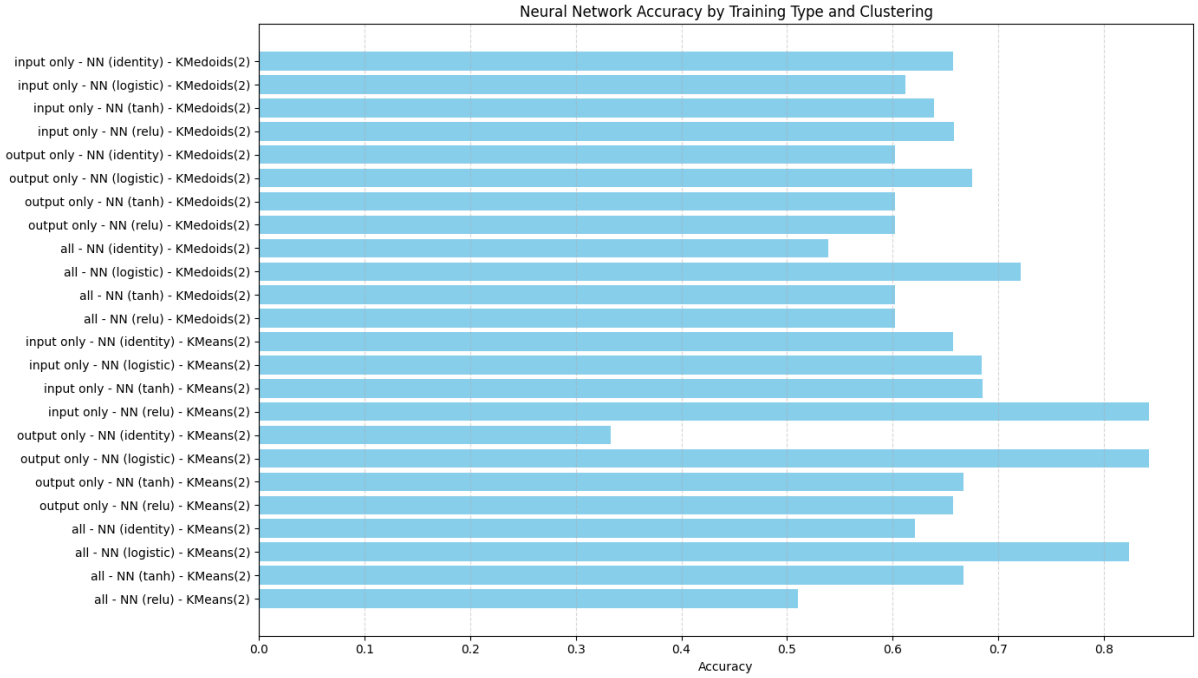


Figure 2.9.10: Neural Networks Evaluation across all models

Table 2.9.4: Neural Network Model Performance with Different Activations and Clustering Methods

Training Type	Clustering	Model (Activation)	Accuracy
Input only	KMedoids(2)	NN (identity)	0.6576
Input only	KMedoids(2)	NN (logistic)	0.6117
Input only	KMedoids(2)	NN (tanh)	0.6390
Input only	KMedoids(2)	NN (relu)	0.6584
Output only	KMedoids(2)	NN (identity)	0.6022
Output only	KMedoids(2)	NN (logistic)	0.6749
Output only	KMedoids(2)	NN (tanh)	0.6022
Output only	KMedoids(2)	NN (relu)	0.6022
All	KMedoids(2)	NN (identity)	0.5390
All	KMedoids(2)	NN (logistic)	0.7216
All	KMedoids(2)	NN (tanh)	0.6022
All	KMedoids(2)	NN (relu)	0.6022
Input only	KMeans(2)	NN (identity)	0.6563
Input only	KMeans(2)	NN (logistic)	0.6840
Input only	KMeans(2)	NN (tanh)	0.6848
Input only	KMeans(2)	NN (relu)	0.8424
Output only	KMeans(2)	NN (identity)	0.3333
Output only	KMeans(2)	NN (logistic)	0.8424
Output only	KMeans(2)	NN (tanh)	0.6667
Output only	KMeans(2)	NN (relu)	0.6571
All	KMeans(2)	NN (identity)	0.6212
All	KMeans(2)	NN (logistic)	0.8242
All	KMeans(2)	NN (tanh)	0.6667
All	KMeans(2)	NN (relu)	0.5104

Observation:

- **Best Performance:** The highest accuracy (0.8424) is achieved by the neural network using the `ReLU` activation function, trained on `input parameters` with `KMeans(2)` clustering.
- **Clustering Impact:**
  - `KMeans(2)` clustering generally results in better model performance compared to `KMedoids(2)`, especially for `ReLU` and `logistic` activations.
  - For example, `NN (logistic)` on all parameters achieves an accuracy of **0.8242** with KMeans, compared to **0.7216** with KMedoids.
- **Activation Function Comparison:**
  - `ReLU` and `logistic` activations consistently outperform `identity` and `tanh` activations.
  - The `identity` activation shows poor performance, particularly for output-only training with KMeans (**0.3333** accuracy).

Chapter 3

Results

3.1 Clustering Results

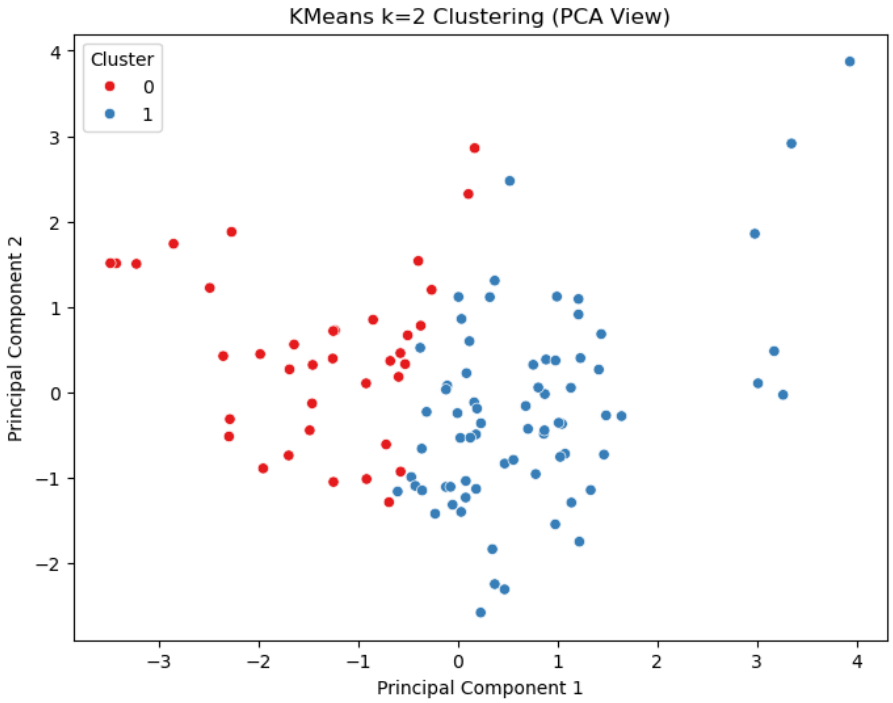


Figure 3.1.1: Cluster Scatter Plot for K-Means with $k = 2$

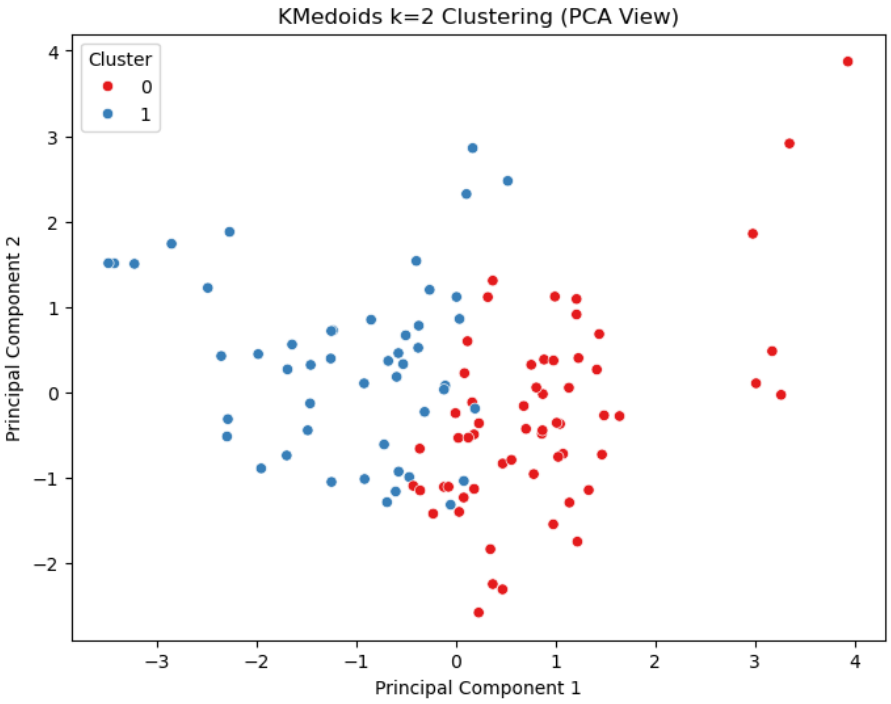


Figure 3.1.2: Cluster Scatter Plot for K-Medoids with $k = 2$

3.2 Classification Results

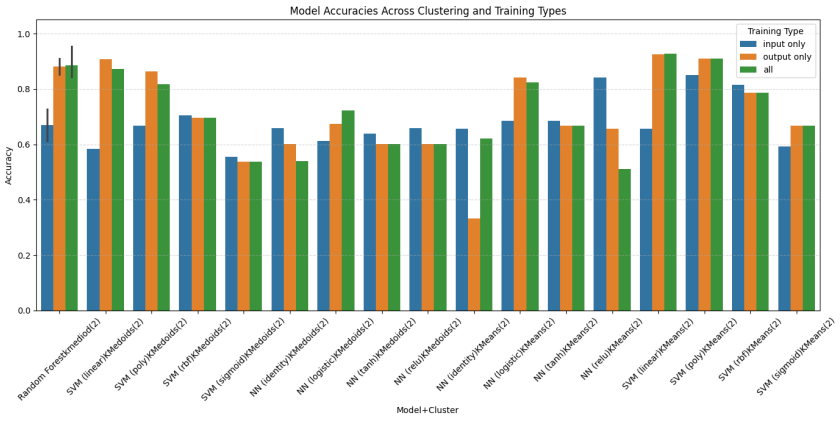


Figure 3.2.1: Classification Model Evaluation

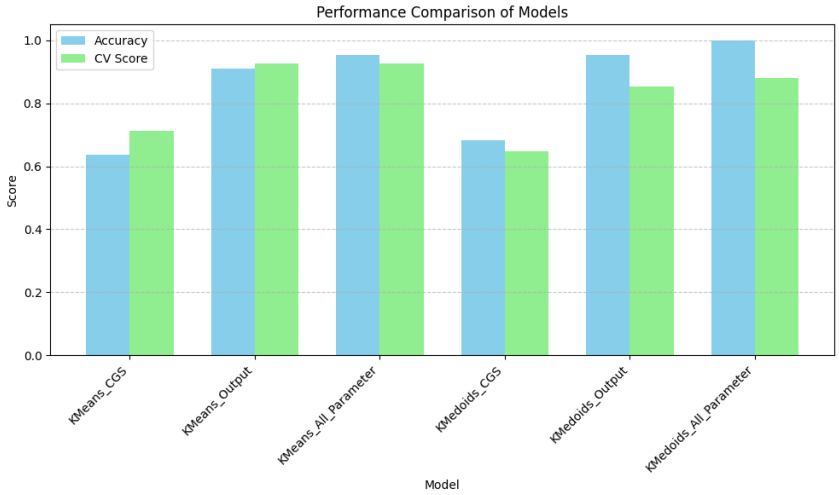


Figure 3.2.2: XGBoost - Model Evaluation

Based on the comprehensive evaluation of various models trained on different parameter subsets (input only, output only, all parameters) and clustered using **KMeans(2)** and **KMedoids(2)**, we draw the following conclusion:
Best Model:

- **Model:** XGBoost
- **Clustering:** KMeans(2)
- **Training Parameters:** All Parameters
- **Accuracy:** 0.9545
- **Precision:** 1.0
- **Recall:** 0.8571
- **F1 Score:** 0.9231
- **Cross-Validation Score:** 0.9264

This configuration demonstrates the highest balance between performance metrics and cross-validation stability, making it the most robust classifier for soybean classification.

Models with similar Results

1. **XGBoost + All Parameters + KMedoid(2) with extensive hyper-parameter tuning**
    - Accuracy : 1.0
    - Cross Validation Score : 0.8792
2. **Random Forest + All Parameters + KMedoids(2)**
    - Accuracy: 0.9545
    - Precision/Recall/F1: 1.0

- Cross-Validation Score: 0.8792

3. **SVM (Linear) + All Parameters + KMeans(2)**
   - Accuracy: 0.9264

Observations

- Training on **All Parameters** yields the best results compared to training on input/output parameters alone.
- **XGBoost** consistently outperforms other models when hyperparameter tuning is applied.
- **KMeans(2)** clustering slightly edges out KMedoids(2) in top-performing combinations.
- **Neural Networks** show inconsistent performance and are not preferred.
- **SVM with Linear Kernel** also delivers competitive results, especially when used with all or output parameters.

Recommendation

For optimal soybean classification performance, we recommend using:

XGBoost trained on **All Parameters** with **KMeans(2)** clustering

3.3 Analysis to Get Optimal Conditions for Maximizing Soybean Productivity

This section focuses on identifying conditions that maximize soybean productivity using two complementary approaches:

- **(1) Cluster-based analysis:** Models were trained after applying K-Means clustering ($k = 2$) to separate high vs low productivity groups.
- **(2) Regression-based analysis:** Without using clustering, we trained predictive models using a composite yield metric derived from biological indicators.

3.3.1 Composite Yield Metric Definition

To quantify soybean productivity while considering both yield and nutritional quality, we constructed a new metric called the **Composite Yield Metric**, defined as follows:

1. Compute Weighted Yield per Unit Area (WYUA):

$$WYUA = \left(\frac{\text{Weight of 300 Seeds (W3S)}}{300}\right) \times \text{Seed Yield per Unit Area (SYUA)}$$

2. Normalize WYUA and Protein Percentage (PPE) to [0, 1] using MinMaxScaler.
3. Compute the final composite score:

$$\text{Composite\_Yield\_Metric} = \left(\frac{WYUA_{\text{scaled}} + PPE_{\text{scaled}}}{2}\right) \times 100$$

This metric ranges between 0 and 100, providing a balanced view of both yield quantity and protein content.

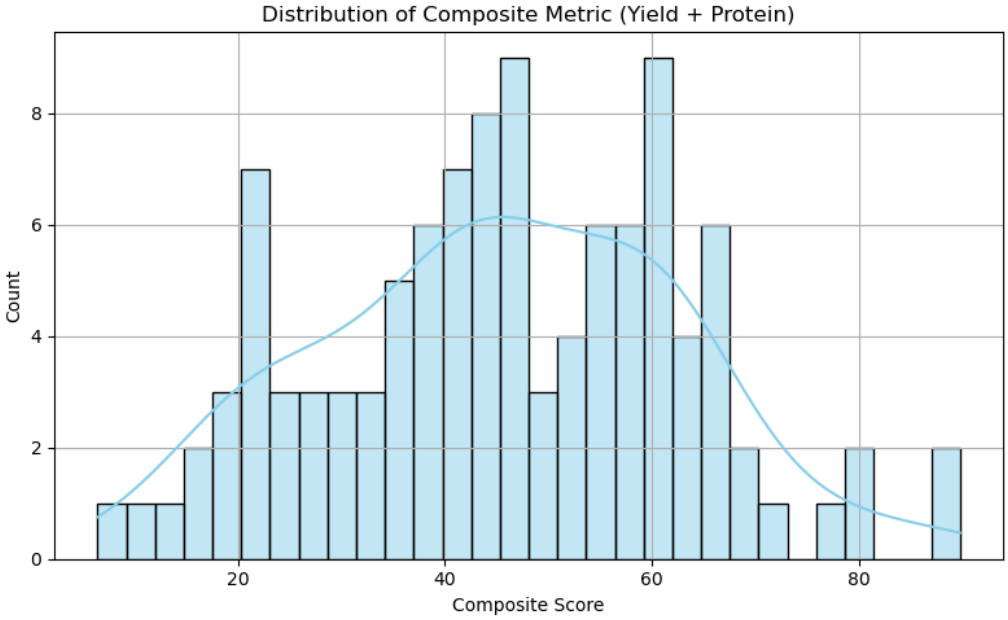


Figure 3.3.1: Distribution of the Composite Yield Metric

3.3.2 SHAP Feature Importance (Cluster-Based)

A Gradient Boosting Regressor was trained on samples labeled by K-Means clustering. SHAP values were used to identify key features impacting high production.

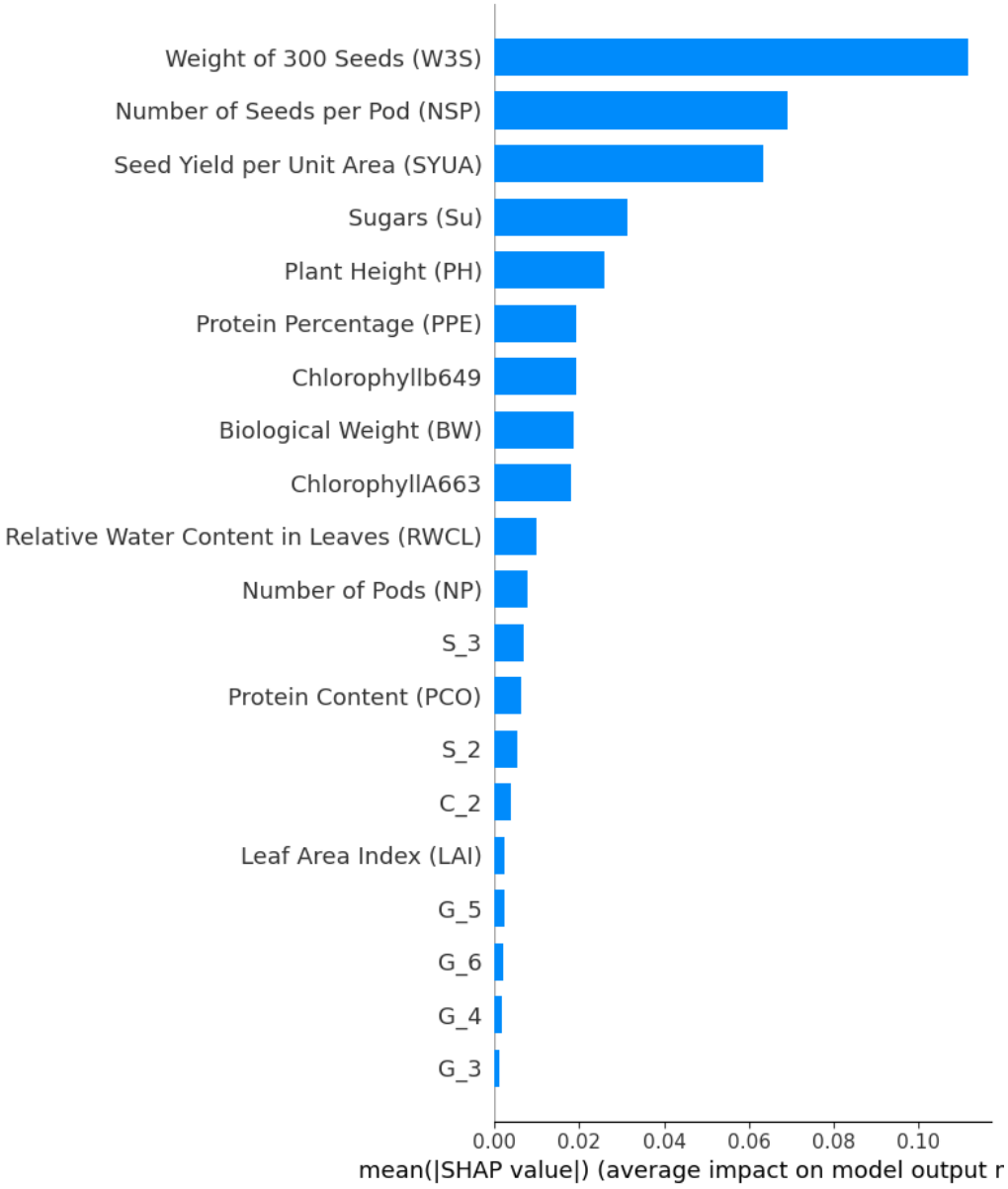


Figure 3.3.2: SHAP Summary Plot (Bar) – Cluster-based Model

Figure 3.3.3: SHAP Summary Plot (Dot) – Cluster-based Model

3.3.3 SHAP Feature Importance (Regression-Based)

A regression model trained on the composite metric identified the most influential input features.

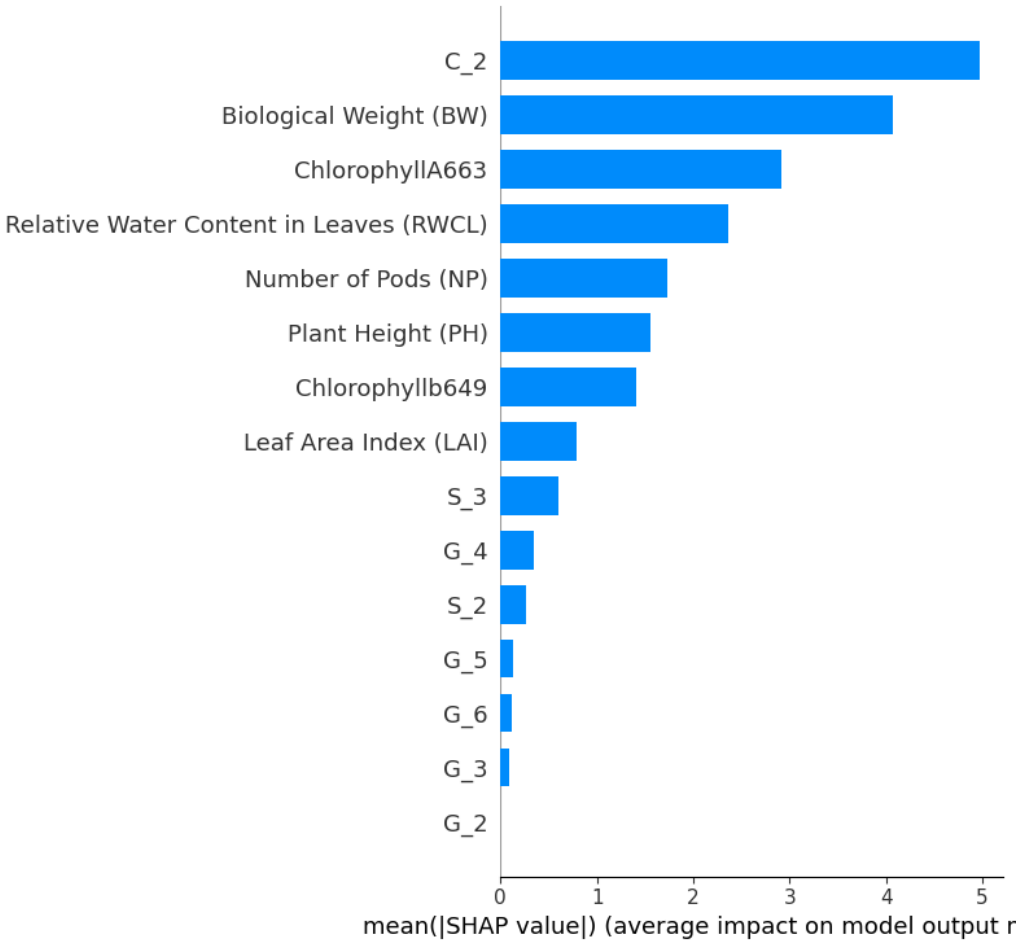


Figure 3.3.4: SHAP Summary Plot (Bar) – Composite Yield Metric (Regression)

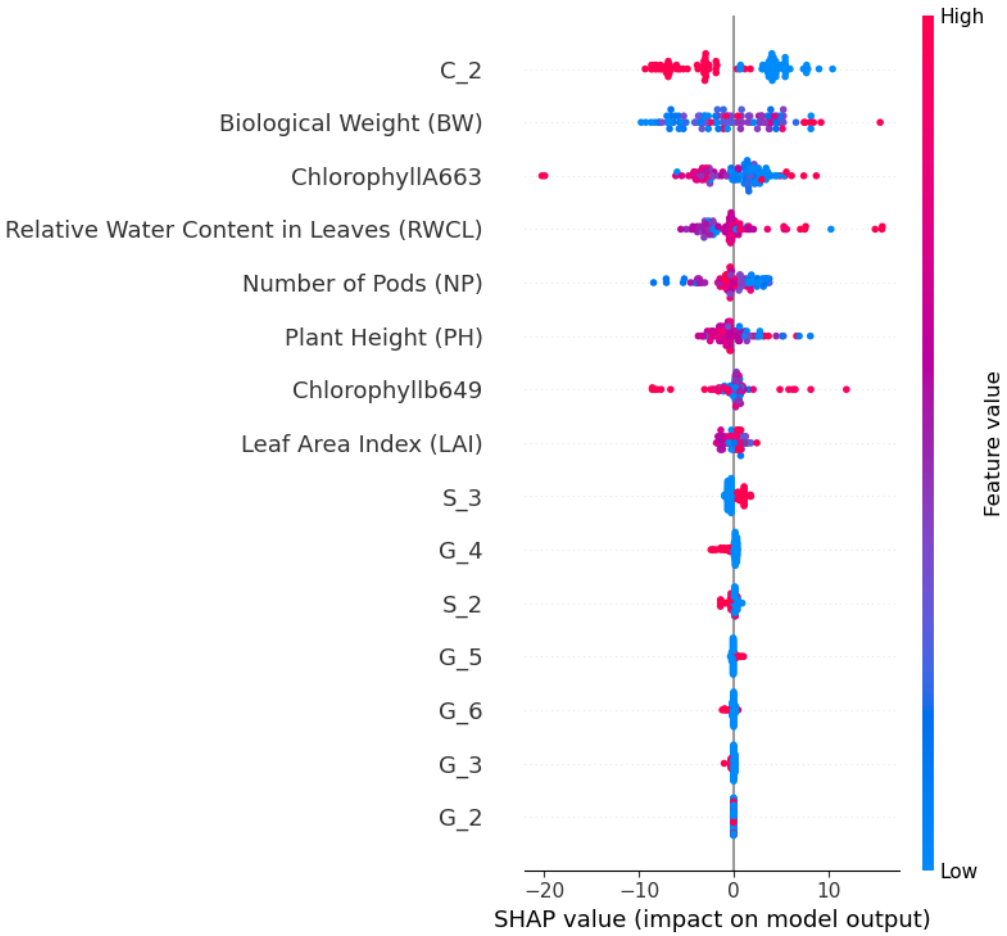


Figure 3.3.5: SHAP Summary Plot (Dot) – Composite Yield Metric (Regression)

3.3.4 Feature Importance and Ranges

Top Features (Regression Importance)

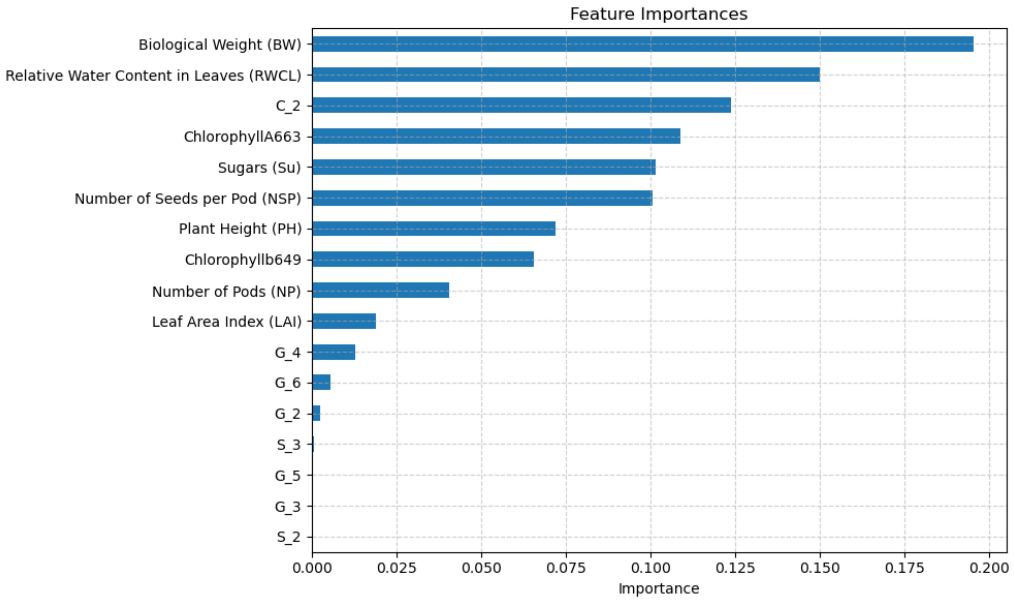


Figure 3.3.6: Feature Importance for Composite Yield Metric

Optimal Ranges Identified

Based on univariate analysis of key features, the following ranges are associated with higher productivity:

- **Biological Weight (BW):** 80–90
- **Sugars (Su):** 0.2–0.3 and 0.45–0.55
- **Plant Height (PH):** 40–49 and 54–56
- **Chlorophyllb649:** 1–3
- **Leaf Area Index (LAI):** 0.02–0.05 and 0.075–0.10

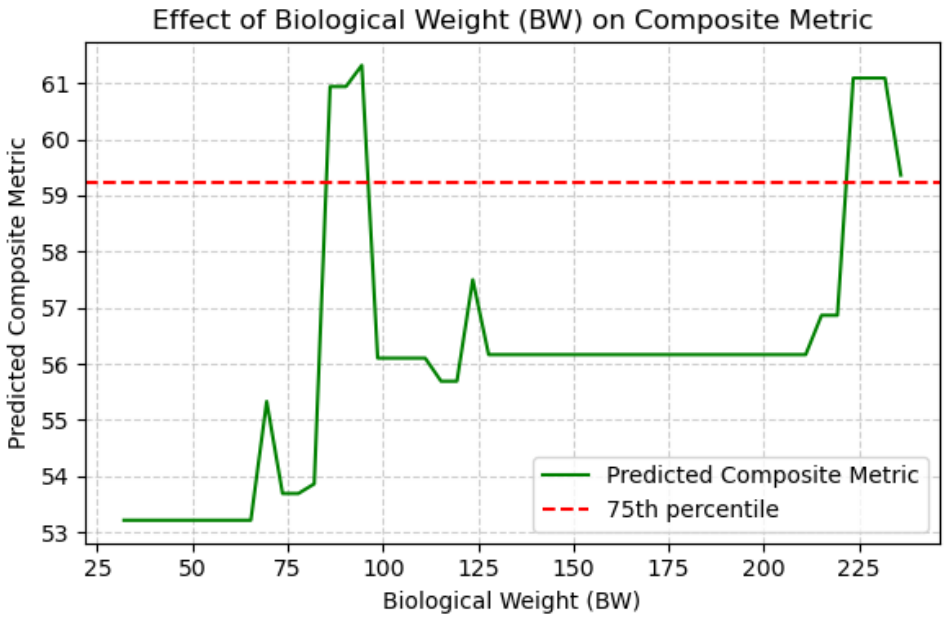


Figure 3.3.7: Effect of Biological Weight on Composite Yield

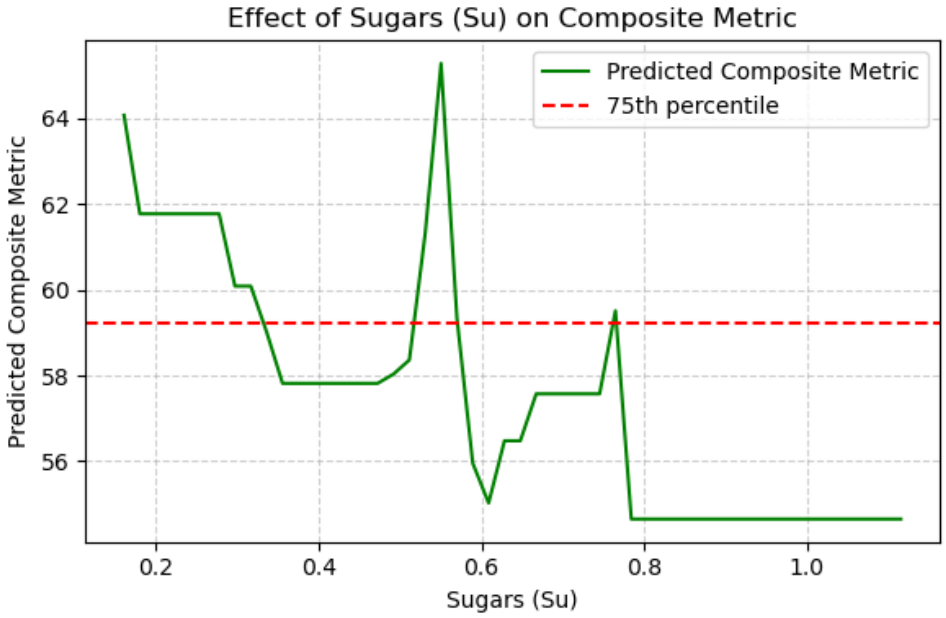


Figure 3.3.8: Effect of Sugar Content on Composite Yield

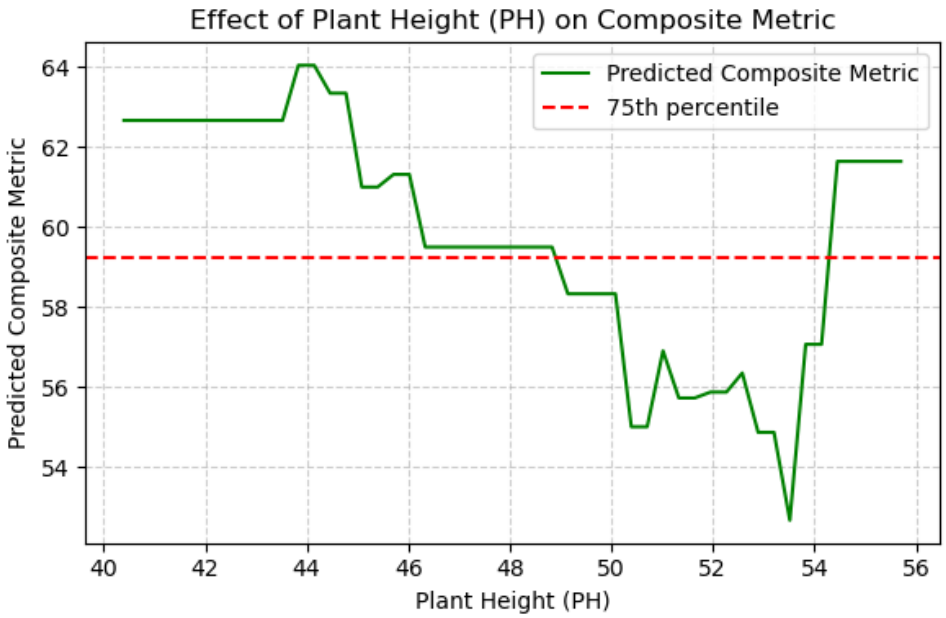


Figure 3.3.9: Effect of Plant Height on Composite Yield

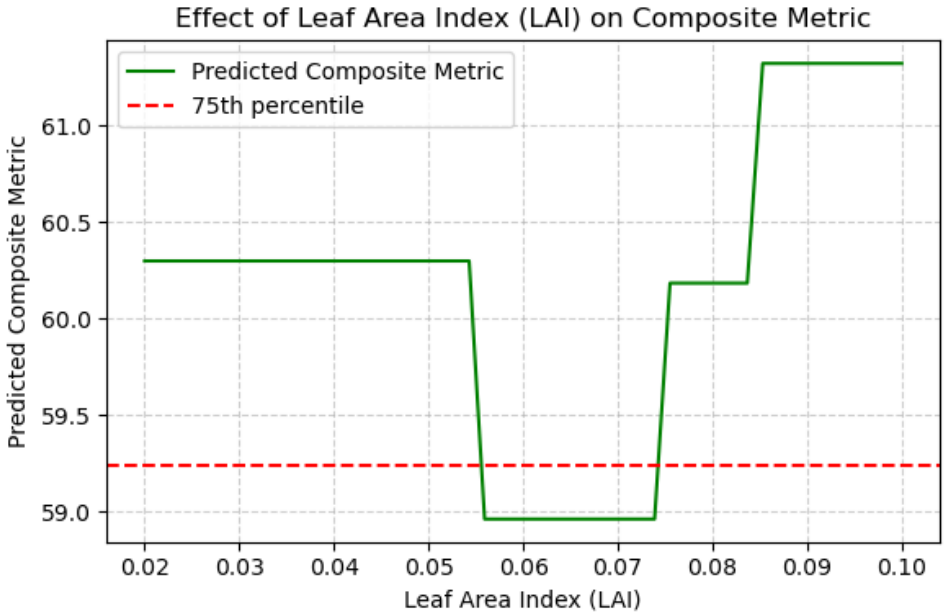


Figure 3.3.10: Effect of LAI on Composite Yield

3.3.5 SHAP Waterfall Plot for Best Sample

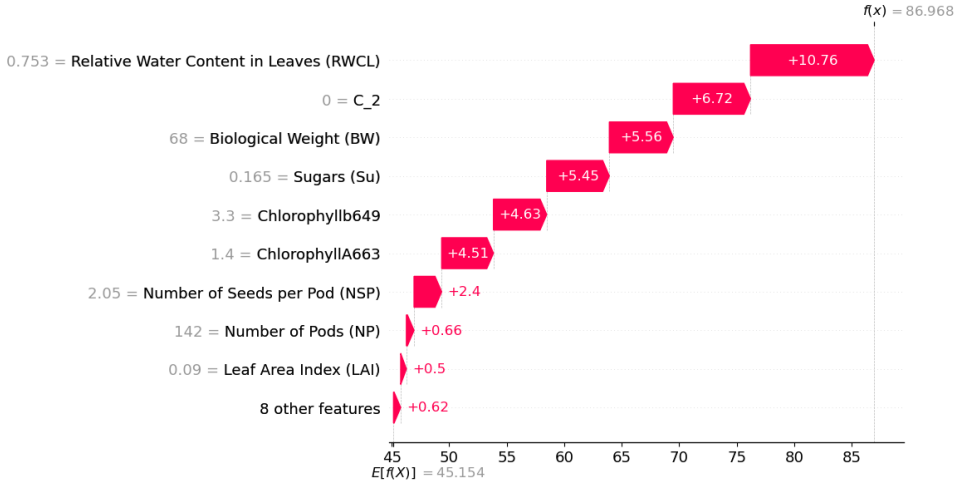


Figure 3.3.11: SHAP Waterfall Plot – Top Performing Sample

3.3.6 Interpretable Decision Tree Rules for High Productivity

To derive human-interpretable insights, a simple decision tree regressor was trained to model the Composite Yield Metric. The resulting decision rules are summarized below, showing feature thresholds that lead to high or low productivity outcomes.

Decision Tree Rules for High Yield + Protein Composite:

- **If** Biological Weight (BW) ≤ 85.5:
  - **If** C_2 ≤ 0.5:
    - **If** Chlorophyllb649 ≤ 3.05:
      - **If** BW ≤ 75.0: **Predicted Composite Score = 59.85**
      - **Else**: **Score = 44.81**
    - **Else if** Chlorophyllb649 ≤ 3.2: **Score = 68.20**
    - **Else**: **Score = 89.77**
  - **Else if** C_2 ≥ 0.5:
    - **If** ChlorophyllA663 ≤ 1.55:
      - **If** Chlorophyllb649 ≤ 3.15: **Score = 30.80**
      - **Else**: **Score = 46.00**
    - **Else if** ChlorophyllA663 ≥ 1.55:
      - **If** G_6 ≤ 0.5: **Score = 20.39**
      - **Else**: **Score = 34.36**
- **Else if** BW ≥ 85.5:
  - **If** Leaf Area Index (LAI) ≤ 0.08: **Score = 66.82**
  - **Else**: **Score = 29.19**

Figure 3.3.12: Extracted Decision Tree Rules for High Composite Yield Metric

3.3.7 Feature Impact: Experimental Conditions

To understand how genotype (G), salicylic acid (S), and water stress (C) affect productivity:

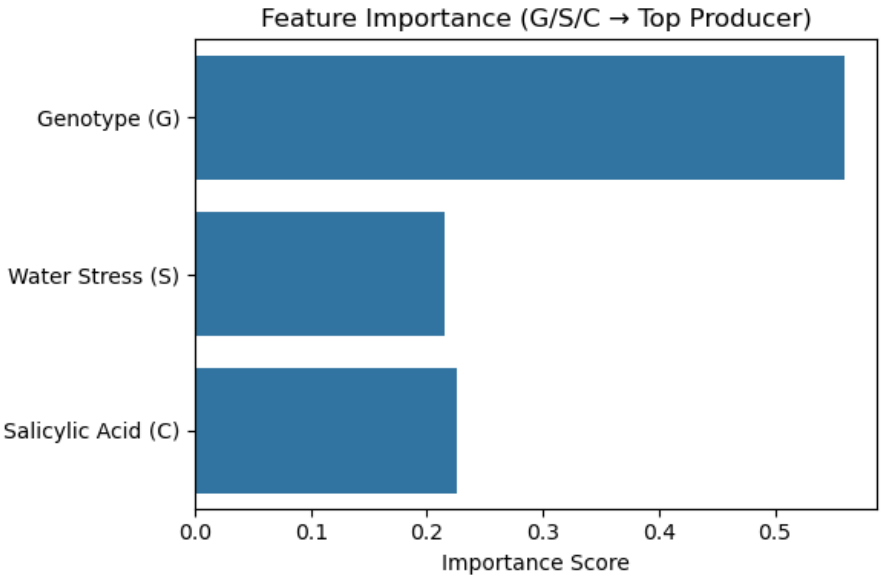


Figure 3.3.13: Feature Importance (G, S, C → High Production)

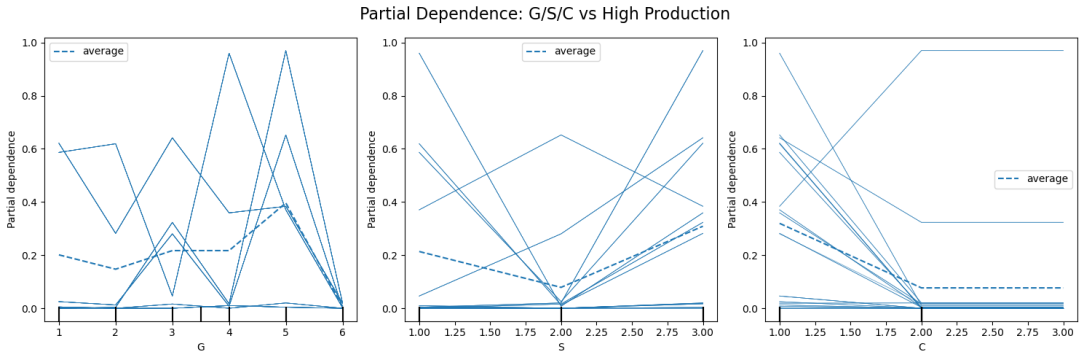


Figure 3.3.14: Partial Dependence Plot – G/S/C vs High Production Probability

3.3.8 Heatmaps: Genotype and Treatment Combinations

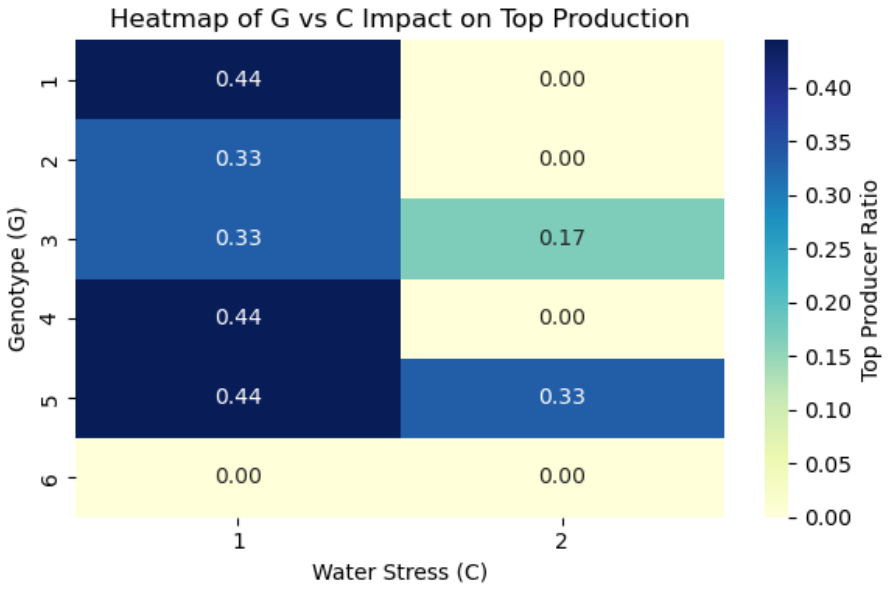


Figure 3.3.15: Heatmap of Genotype vs Salicylic Acid (S)

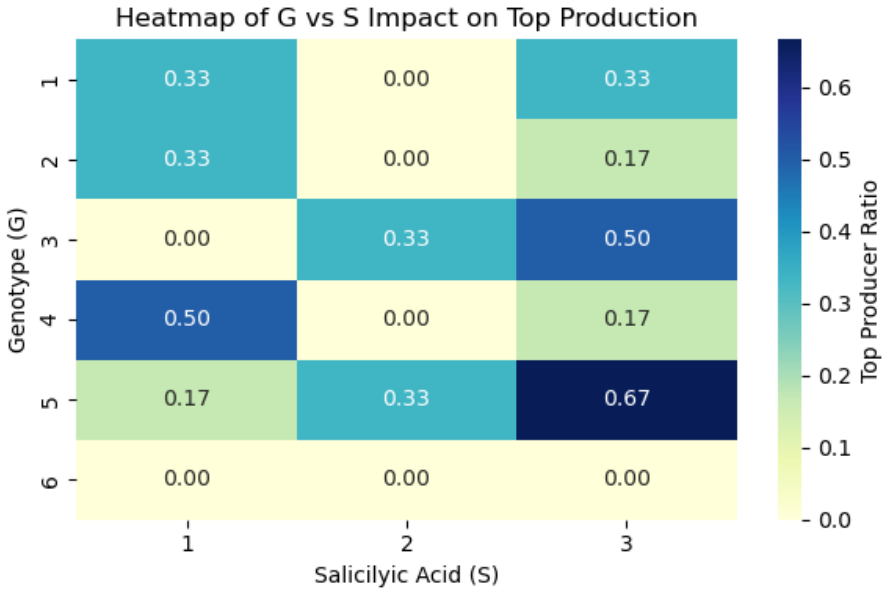


Figure 3.3.16: Heatmap of Genotype vs Water Stress (C)

Chapter 4

Conclusion and Future Scope

4.1 Conclusion

In this study, we employed a hybrid machine learning pipeline integrating unsupervised clustering (KMeans and KMedoids) with supervised classification models (Random Forest, SVM, Neural Networks, XGBoost) to predict soybean productivity under varying experimental conditions. Our evaluation focused on three subsets of input features: input parameters, output parameters, and the combination of all.

Key findings include:

- The **XGBoost classifier**, trained on **all parameters** and clustered using **KMeans(2)**, emerged as the best-performing model with an accuracy of **95.45%**, precision and F1-score of **1.0** and **0.9231**, respectively, and a cross-validation score of **92.64%**.
- Training on **all parameters** consistently outperformed models trained on input-only or output-only features, highlighting the benefit of comprehensive data inclusion.
- Among clustering techniques, **KMeans(2)** generally offered marginally better performance than KMedoids(2), particularly when combined with XGBoost and SVM (Linear).
- **Neural Networks** displayed high variability and were not as reliable as tree-based or kernel-based models for this task.
- Feature interpretation through **SHAP** and **Partial Dependence Plots (PDPs)** emphasized the importance of **Seed Yield per Unit Area**, **Protein Content**, **W3S**, and **Sugars** in driving productivity.
- Agronomic treatment factors such as **Genotype G1/G3**, **250 mg Salicylic Acid**, and **moderate water stress (70% FC)** were associated with optimal soybean performance.

Together, these insights form a robust data-driven framework to support decision-making in soybean cultivation and stress management.

4.2 Future Scope

The current work opens several avenues for further exploration:

1. **Multi-class Classification:** Extend the binary classification to multiclass outputs representing varying levels of yield or stress severity.
2. **Temporal Analysis:** Integrate time-series data to model the progression of stress effects and crop growth, enabling dynamic prediction and adaptive treatment scheduling.
3. **Explainable AI:** Further deploy advanced interpretability techniques (e.g., counterfactual analysis, LIME [11]) to understand decision boundaries and model behavior in uncertain scenarios.
4. **Transfer Learning:** Explore transfer learning frameworks to adapt the trained models to other leguminous crops or new soybean genotypes without retraining from scratch [12].
5. **Precision Agriculture Integration:** Combine this classification framework with drone or satellite-based phenotyping data for real-time in-field decision support systems [13].
6. **Model Uncertainty Quantification:** Incorporate Bayesian models or uncertainty quantification to express confidence levels in predictions, aiding cautious agronomic decision-making [14].
7. **Data Augmentation and Synthetic Generation:** Use generative models to simulate rare stress conditions or missing combinations of treatments to enhance model robustness.

This work provides a blueprint for integrating machine learning with agronomic science to optimize crop productivity, especially under complex biotic and abiotic stress conditions. Continued research in this direction holds promise for enhancing food security and sustainable agriculture practices.

Bibliography

[1] Wikipedia contributors, "Soybean — wikipedia," 2024, accessed: 2025-04-15. [Online]. Available: https://en.wikipedia.org/wiki/Soybean

[2] scikit-learn developers, "sklearn.cluster.kmeans — scikit-learn documentation," 2024, accessed: 2025-04-15. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[3] Wikipedia contributors, "K-medoids — wikipedia," 2024, accessed: 2025-04-15. [Online]. Available: https://en.wikipedia.org/wiki/K-medoids

[4] scikit-learn developers, "sklearn.manifold.tsne," 2024, accessed: 2025-04-11. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

[5] ——, "sklearn.decomposition.pca — scikit-learn documentation," 2024, accessed: 2025-04-15. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[6] H. Gültekin, "Silhouette score," 2020, accessed: 2025-04-11. [Online]. Available: https://medium.com/biased-algorithms/silhouette-score-d85235e7638b

[7] scikit-learn developers, "sklearn.metrics.davies_bouldin_score — scikit-learn documentation," 2024, accessed: 2025-04-15. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html

[8] ——, "sklearn.metrics.calinski_harabasz_score — scikit-learn documentation," 2024, accessed: 2025-04-15. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.calinski_harabasz_score.html

[9] W. Contributors, "Bayesian information criterion," https://en.wikipedia.org/wiki/Bayesian_information_criterion, 2025, accessed: 2025-04-15.

[10] ——, "Akaike information criterion," https://en.wikipedia.org/wiki/Akaike_information_criterion, 2025.

[11] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[13] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018.

[14] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," *International Conference on Machine Learning*, pp. 1050–1059, 2016.