NDVI-Based Crop Classification for Rice and Cotton

Tayyab Raza (404821)
Muhammad Sarmad Saleem (411411)

CS 471 : Machine Learning Fall 2024

National University of Sciences and Technology

School of Electrical Engineering and Computer Science (NUST)

December 22, 2024

Contents

Chapter 1

Introduction

1.1 Overview

This report discusses the implementation and evaluation of machine learning models, both supervised and unsupervised, to classify rice and cotton crops using NDVI (Normalized Difference Vegetation Index) data. The primary goal is to distinguish these crops based on their distinct spectral characteristics derived from NDVI values collected over several years.

1.2 Problem Statement

Accurate crop classification is critical for efficient resource management, yield estimation, and agricultural policymaking. This project aims to leverage machine learning models to classify rice and cotton crops using NDVI time-series data.

1.3 Scope and Objectives

The scope of this study involves applying machine learning techniques to NDVI datasets spanning three years. The objectives are as follows:

- To preprocess and enhance the NDVI dataset for analysis.
- To implement and assess supervised models such as XGBoost, Bagging, Random Forest, and SVM using cross-validation techniques.
- To implement and evaluate unsupervised models such as K-Means, Hierarchical Clustering, DBSCAN, and GMM, with and without PCA.
- To analyze and compare the results to determine the most effective approach for crop classification.

Chapter 2

Pre-processing

This chapter explains the preparatory steps performed on the NDVI dataset to ensure its suitability for machine learning. These steps include data loading, preparation, handling missing values, data augmentation, outlier detection, and feature scaling.

2.1 Data Loading

The first step was importing NDVI data from CSV files corresponding to the years 2021, 2022, and 2023. Each file contains NDVI values recorded throughout the growing seasons for rice and cotton crops. The datasets were stored as separate DataFrame variables for ease of manipulation.

Purpose: Ensuring each year's data is independently accessible allows for efficient analysis and supports the cross-validation approach employed later.

Key Actions:

- Loaded CSV files for the years 2021, 2022, and 2023 into individual datasets.
- Verified the data by checking column headers, row counts, and overall structure.

2.2 Year-wise Datasets

To cater to supervised and unsupervised learning needs, two distinct datasets were prepared:

2.2.1 Supervised Learning Datasets

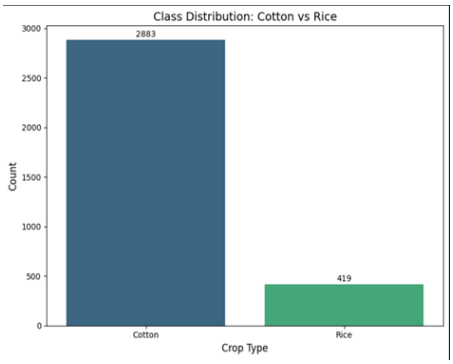
Year-wise datasets were created to facilitate cross-validation. Models were trained on data from two years and tested on the third year. This approach ensures generalizability by exposing models to diverse growth conditions over multiple years.

Cross-validation Combinations:

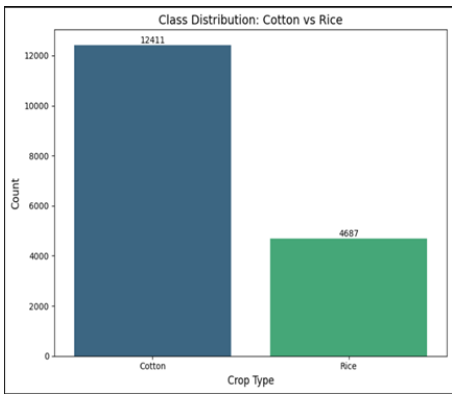
- Train on 2021 and 2022, Test on 2023
- Train on 2021 and 2023, Test on 2022
- Train on 2022 and 2023, Test on 2021

2.2.2 Imbalanced Data Set

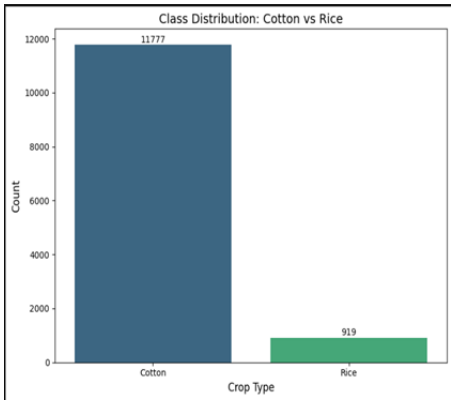
There is a lot of Class Imbalance in the whole dataset. The graphs are given:



(a) Image 1 Caption



(b) Image 2 Caption

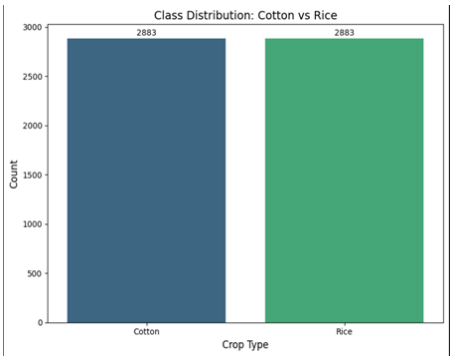


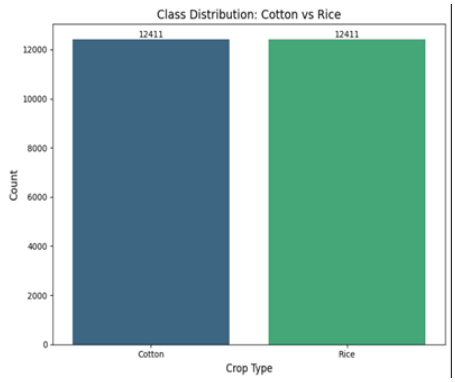
(c) Image 3 Caption

Figure 2.1: Class Imbalance Visualization

2.2.3 Balanced Data set Using Different Techniques

We have applied the following methods to address data imbalance and evaluated their results. Among these, SMOTE provided the best results, so it was used for further analysis.
i- SMOTE ii- Time Series Data Augmentation iii- Oversampling iv- Under Sampling





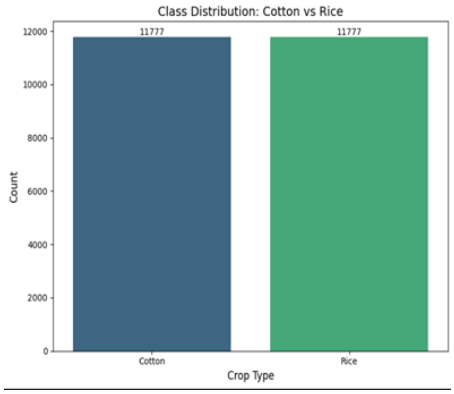


Figure 2.2: Results After applying Smote

2.2.4 Removal of Outliers

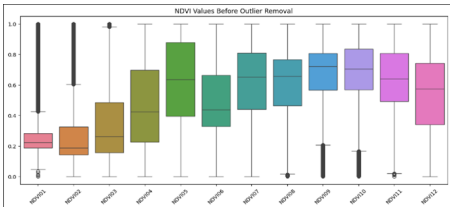
To further improve the quality of the dataset and ensure accurate model performance, we addressed the issue of outliers in the data. Outliers can distort patterns and negatively impact the learning process, especially in sensitive tasks like crop classification. To remove outliers, we applied the Z-score method, which measures how far a data point deviates from the mean in terms of standard deviations. For both the SMOTE-augmented dataset and the undersampled dataset, we calculated the Z-score for each feature and identified outliers as data points with a Z-score exceeding a threshold (e.g., —Z— ¿ 3). These outliers were removed from the datasets to eliminate extreme values that could skew the results or introduce noise. By cleaning the data in this way, we ensured that the clustering and classification models focused on the core patterns within the NDVI values, leading to more robust and reliable performance

```
Original Dataset Shape: (53524, 14)
```

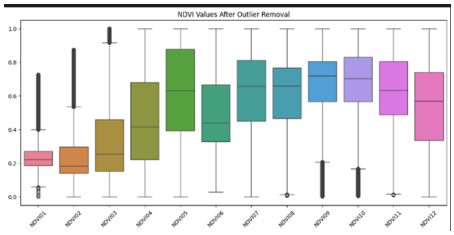
(a) Before Z Score

```
Cleaned Dataset Shape (without outliers): (50860, 14)
```

(b) After Z Score



(a) Before Z Score



(b) After Z Score

2.2.5 Unsupervised Learning Dataset

For unsupervised learning, data from all three years was combined into a single data set. This consolidated dataset was used for clustering and pattern detection using algorithms like K-Means, Hierarchical Clustering, DBSCAN, and GMM.

Purpose: Combining all years allows for identifying general patterns and clusters, leading to robust unsupervised learning outcomes.

2.3 Handling Missing Values

Handling missing data is crucial to maintain the reliability of machine learning models. The datasets were checked for any missing entries.

Key Actions:

- Conducted a thorough inspection for missing values in year-wise and combined datasets.
- Found no missing values, negating the need for imputation or data removal.

Conclusion: With no missing data, all records were retained for further analysis.

2.4 Year-wise Pairing for Supervised Learning

To ensure robust evaluation, year-wise train-test pairs were created for supervised learning. Training on two years and testing on the third simulates real-world scenarios and avoids data leakage.

Purpose: Testing models on unseen data enhances their reliability and provides a realistic performance metric.

2.5 Combined Dataset for Unsupervised Learning

A single dataset combining all years was used for clustering tasks. This approach allowed unsupervised methods to identify patterns across the entire dataset.

Purpose: Consolidation enables more comprehensive analysis and pattern discovery in unsupervised learning.

2.6 Data Augmentation using SMOTE

The dataset had an imbalance, with fewer samples for rice compared to cotton. To mitigate this, SMOTE (Synthetic Minority Oversampling Technique) was employed to generate synthetic samples for the minority class.

Key Actions:

- Analyzed the class distribution for rice and cotton.
- Applied SMOTE to balance the dataset by creating synthetic samples for rice.

Impact: Balanced datasets reduced model bias toward the majority class, improving evaluation metrics such as F1-Score, Recall, and Precision.

2.7 Outlier Detection and Removal

Outliers can negatively impact model performance. The Z-Score method was used to identify and remove extreme outliers, with a threshold of 3.

Key Actions:

- Computed Z-Scores for NDVI values.
- Removed entries with absolute Z-Scores exceeding 3.

Purpose: Eliminating anomalies ensures models are not skewed by extreme values, enhancing robustness and generalization.

2.8 Normalization

Min-Max normalization was applied to scale features to a range of 0 to 1, ensuring uniformity across all attributes.

Key Actions:

- Normalized NDVI values for each year individually.
- Used Min-Max scaling to transform data into the range [0, 1].

Purpose: Normalization prevents larger-scale features from dominating the learning process, benefiting algorithms sensitive to feature scaling like SVM and K-Means.

Effect: Feature scaling improved model convergence speed, classification accuracy, and clustering quality.

2.9 Summary of Pre-processing

The following table summarizes the key preprocessing steps:

Table 2.1: Summary of Preprocessing Steps

Step	**Description**
Data Loading	Imported CSV files for 2021, 2022, 2023
Year-wise Datasets	Created train-test pairs for supervised learning
Missing Value Check	Verified no missing entries
Year-wise Pairing	Enabled cross-validation using year-wise combinations
SMOTE	Balanced class distribution using synthetic samples
Outlier Detection	Removed anomalies via Z-Score method
Normalization	Scaled features to the range [0, 1]

Conclusion: The preprocessing steps, from handling class imbalance to normalization, were critical to ensuring high-quality input data for machine learning. These steps improved model robustness, accelerated convergence, and enhanced generalization on unseen data.

Chapter 3

Supervised Learning

This chapter outlines the implementation, training, and evaluation of supervised learning models for the classification of rice and cotton using NDVI data. The models implemented include XGBoost, Bagging, Random Forest, and Support Vector Machine (SVM). Each model was trained using a cross-validation strategy, where the model was trained on two years of data and tested on the third year.

3.1 Models Implemented

3.1.1 XGBoost (Extreme Gradient Boosting)

3.1.2 Bagging (Bootstrap Aggregation)

3.1.3 Random Forest

3.1.4 Support Vector Machine (SVM)

3.2 Grid Search and Cross-Validation

3.2.1 Grid Search for Hyperparameter Tuning

Grid Search was performed to find the optimal hyperparameters for each supervised learning model. The key parameters tuned for each model are listed below:

- **XGBoost:** learning rate, max depth, number of estimators
- **Bagging:** number of estimators, max depth for base estimators
- **Random Forest:** number of estimators, max depth, minimum samples per leaf
- **SVM:** kernel type, regularization parameter (C), and gamma (for RBF kernel)

3.2.2 Cross-Validation Strategy

The cross-validation strategy involves training the model on two years of data and testing it on the third. The combinations used were:

- Train on 2021 and 2022, Test on 2023
- Train on 2021 and 2023, Test on 2022
- Train on 2022 and 2023, Test on 2021

3.3 XGBoost (Extreme Gradient Boosting)

3.3.1 Testing on 2021

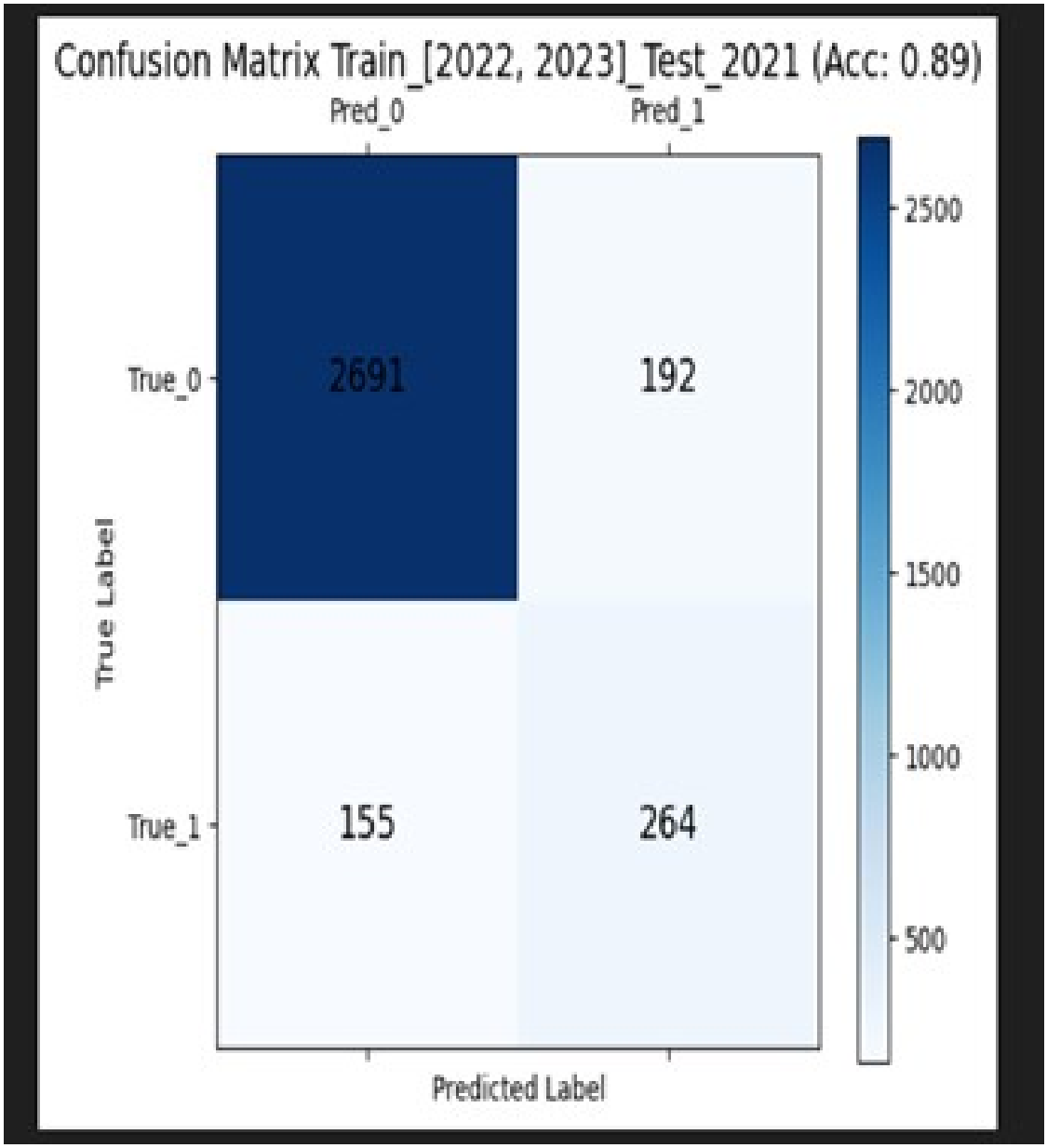


Figure 3.1: Confusion Matrix for XGBoost (Testing on 2021)

Table 3.1: Evaluation Metrics for XGBoost (Testing on 2021)

Metric	**Rice**	**Cotton**
F1-Score	0.60	0.94
Precision	0.57	0.95
Recall	0.63	0.93

Accuracy: 0.89
Parameters used: 'learning$_rate'$: 0.2,'max_depth' : 7,'$n_estimators'$: 200

3.3.2 Testing on 2022

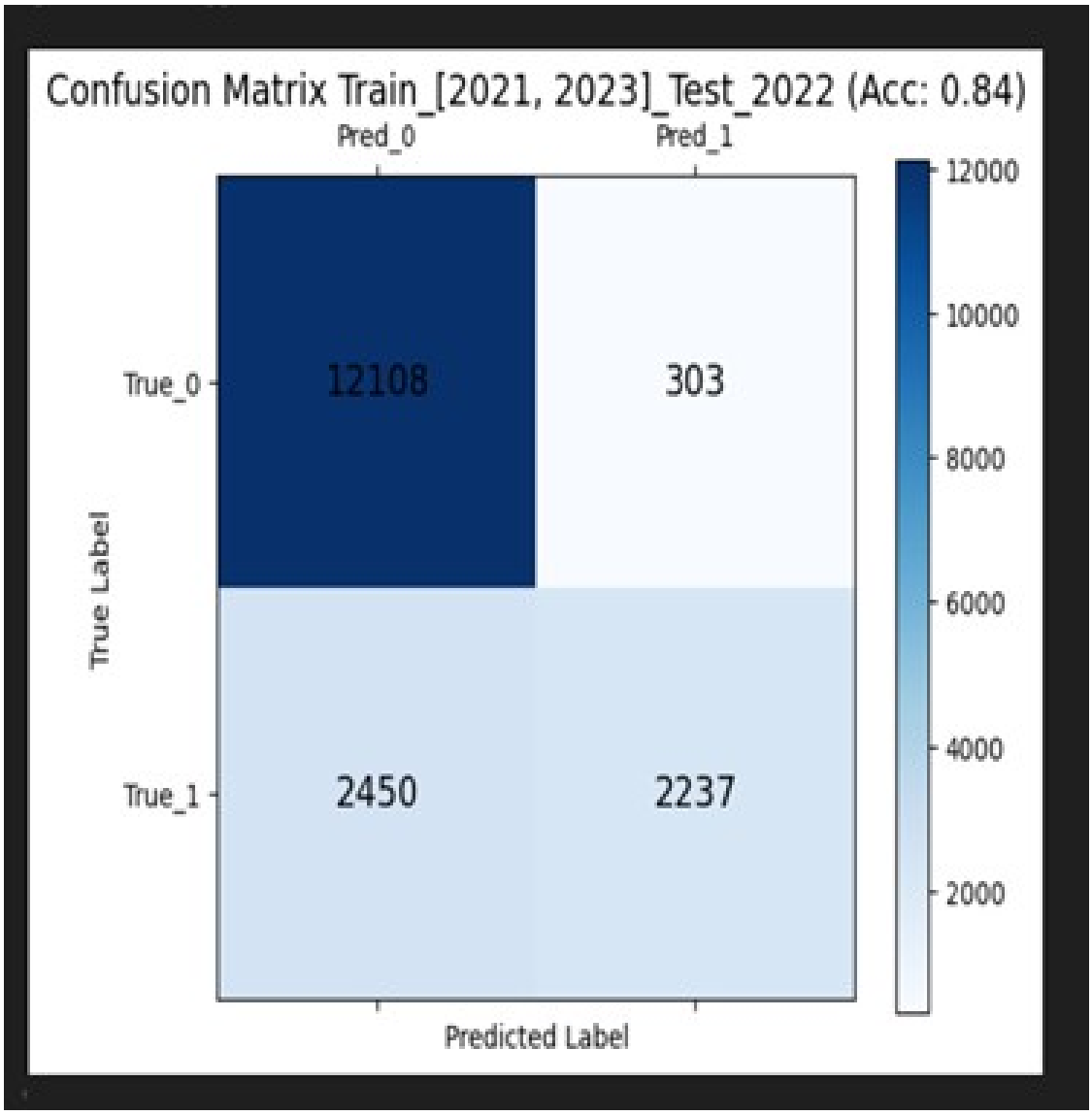


Figure 3.2: Confusion Matrix for XGBoost (Testing on 2023)

Table 3.2: Evaluation Metrics for XGBoost (Testing on 2022)

Metric	**Rice**	**Cotton**
F1-Score	0.61	0.90
Precision	0.83	0.89
Recall	0.47	0.98

Accuracy: 0.84
Parameters used: 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100

3.3.3 Testing on 2023

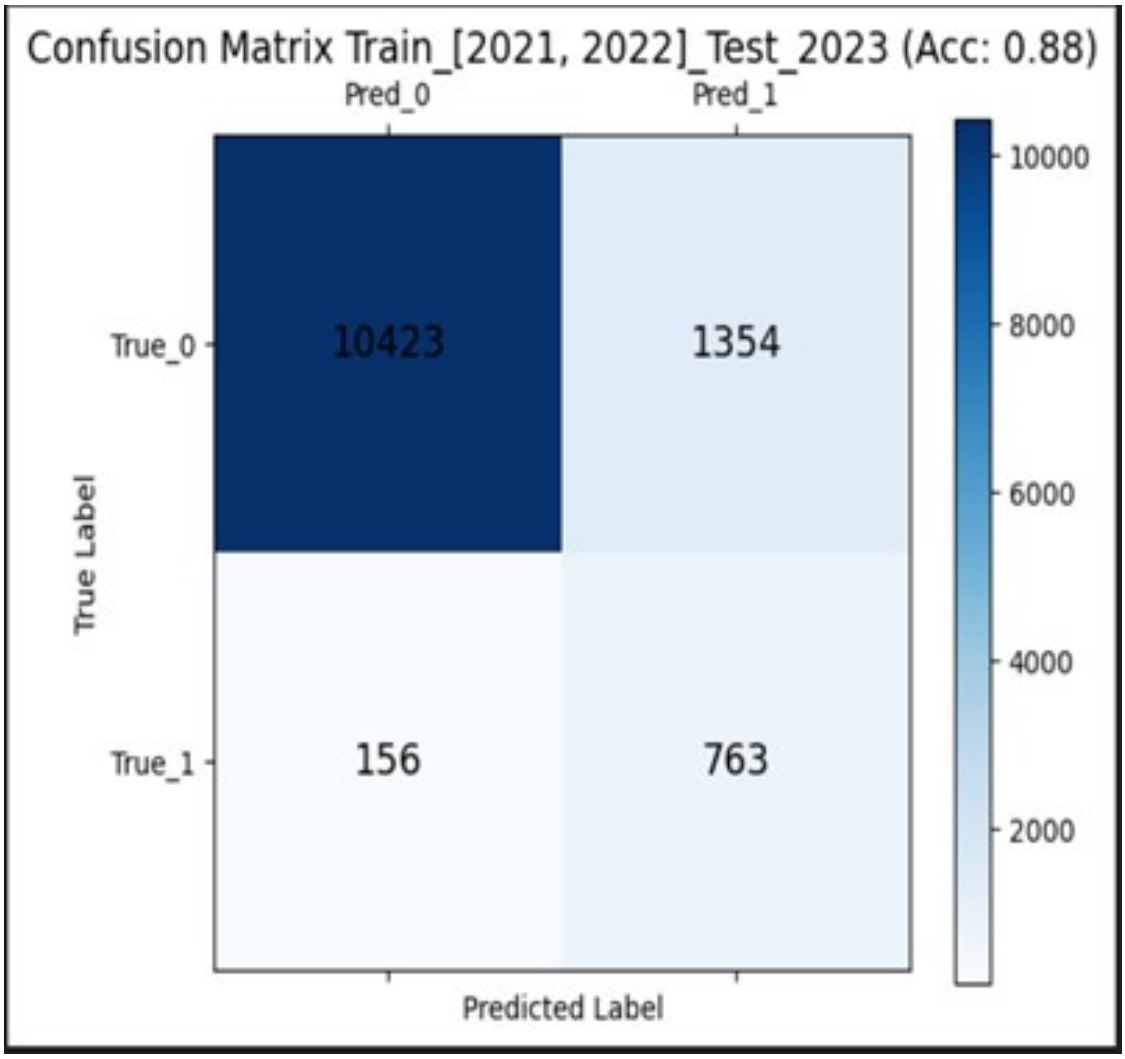


Figure 3.3: Confusion Matrix for XGBoost (Testing on 2023)

Table 3.3: Evaluation Metrics for XGBoost (Testing on 2023)

Metric	**Rice**	**Cotton**
F1-Score	0.50	0.93
Precision	0.40	0.99
Recall	0.84	0.88

Accuracy: 0.8806
Parameters used: 'learning_rate': 0.02, 'max_depth': 3, 'n_estimators': 200

3.4 Bagging (Bootstrap Aggregation)

3.4.1 Testing on 2021

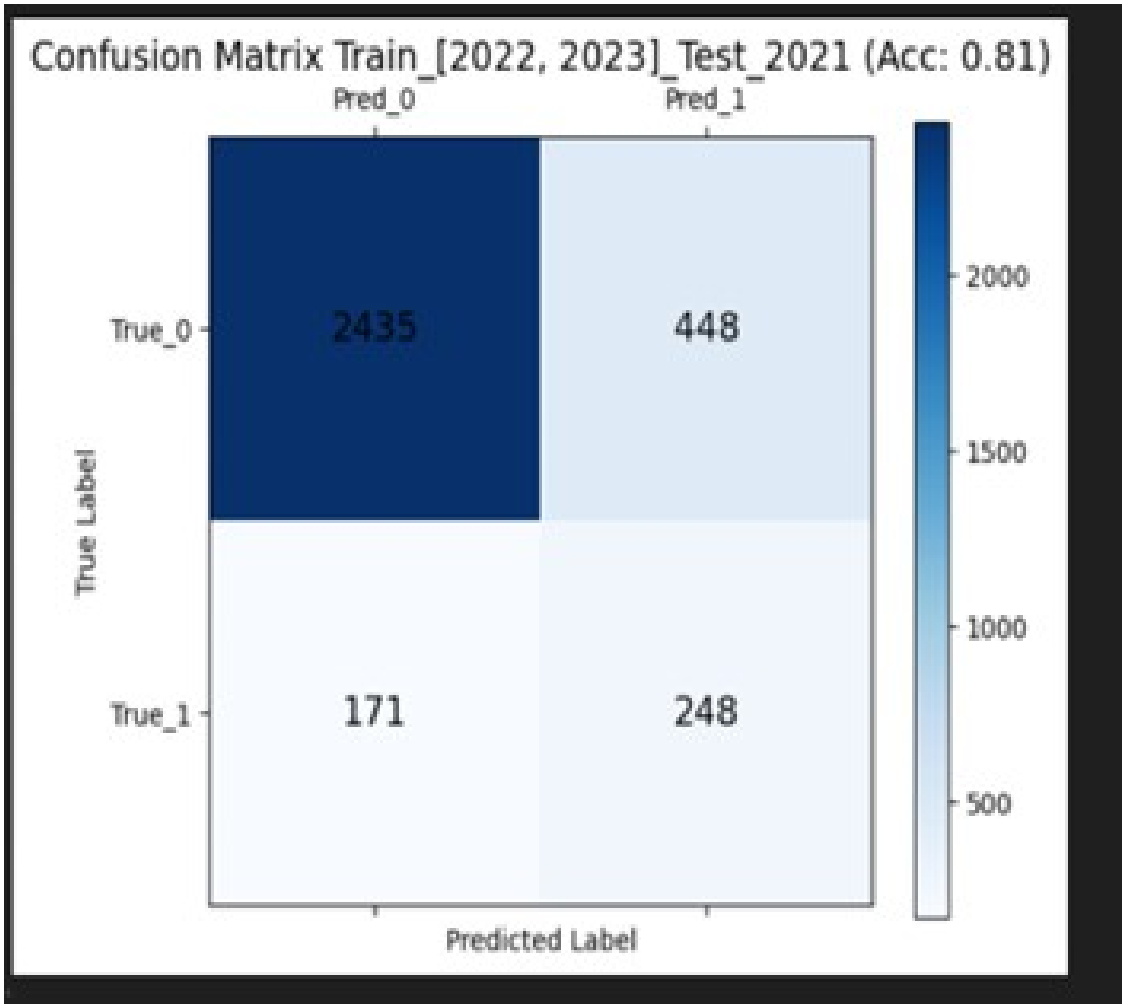


Figure 3.4: Confusion Matrix for Bagging (Testing on 2021)

Table 3.4: Evaluation Metrics for Bagging (Testing on 2021)

Metric	**Rice**	**Cotton**
F1-Score	0.44	0.89
Precision	0.36	0.93
Recall	0.59	0.84

Accuracy: 0.8123
Parameters used: 'n_estimators': 50, 'base_estimator': 'DecisionTreeClassifier'

3.4.2 Testing on 2022

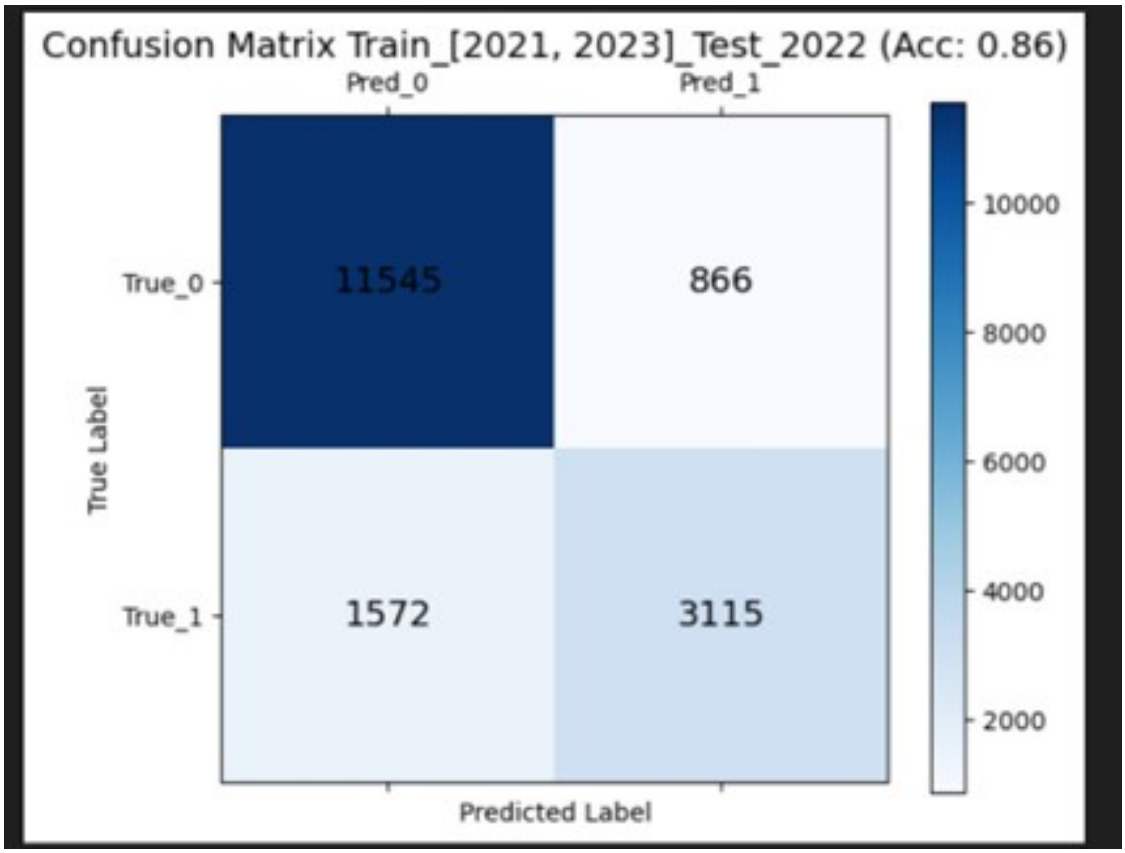


Figure 3.5: Confusion Matrix for Bagging (Testing on 2022)

Table 3.5: Evaluation Metrics for Bagging (Testing on 2022)

Metric	**Rice**	**Cotton**
F1-Score	0.72	0.90
Precision	0.78	0.88
Recall	0.66	0.93

Accuracy: 0.8696
Parameters used: 'n_estimators': 50, 'base_estimator': 'DecisionTreeClassifier'

3.4.3 Testing on 2023

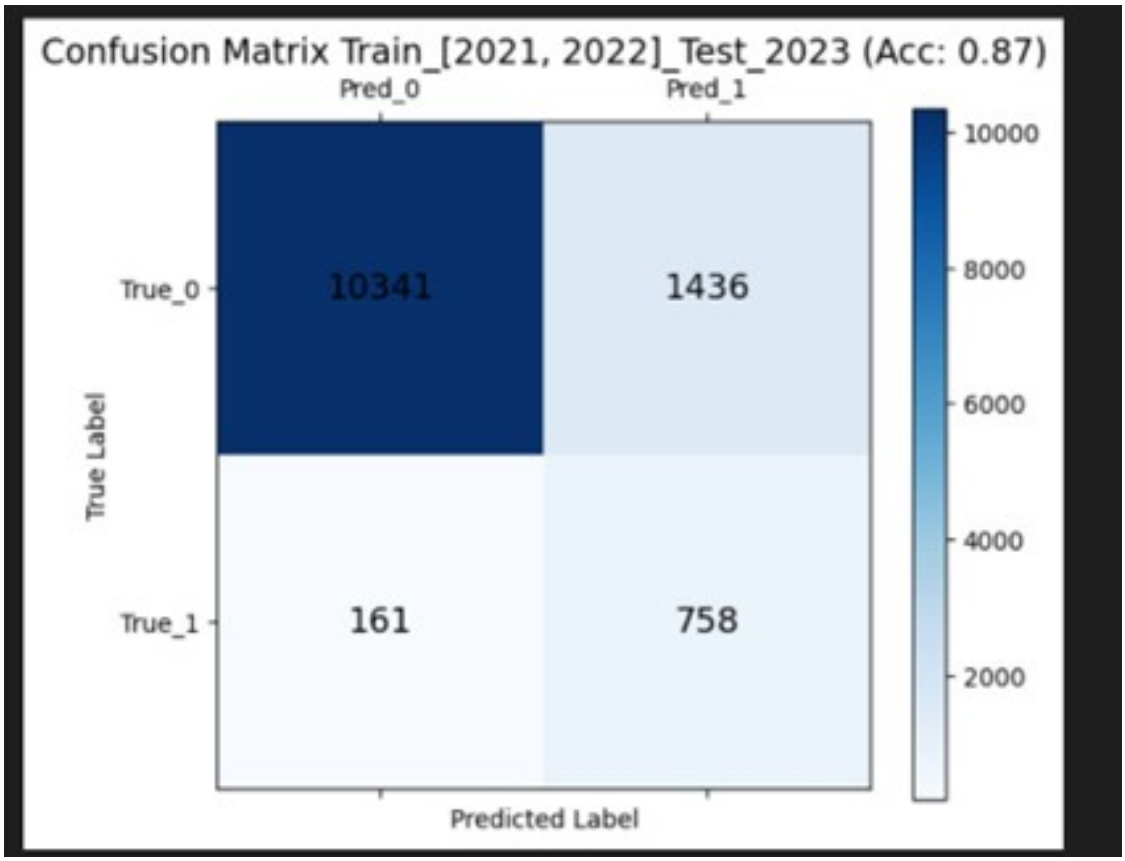


Figure 3.6: Confusion Matrix for Bagging (Testing on 2023)

Table 3.6: Evaluation Metrics for Bagging (Testing on 2023)

Metric	**Rice**	**Cotton**
F1-Score	0.50	0.93
Precision	0.36	0.98
Recall	0.83	0.89

Accuracy: 0.88
Parameters used: 'n_estimators': 100, 'base_estimator': 'DecisionTreeClassifier'

3.5 Random Forest

3.5.1 Testing on 2021

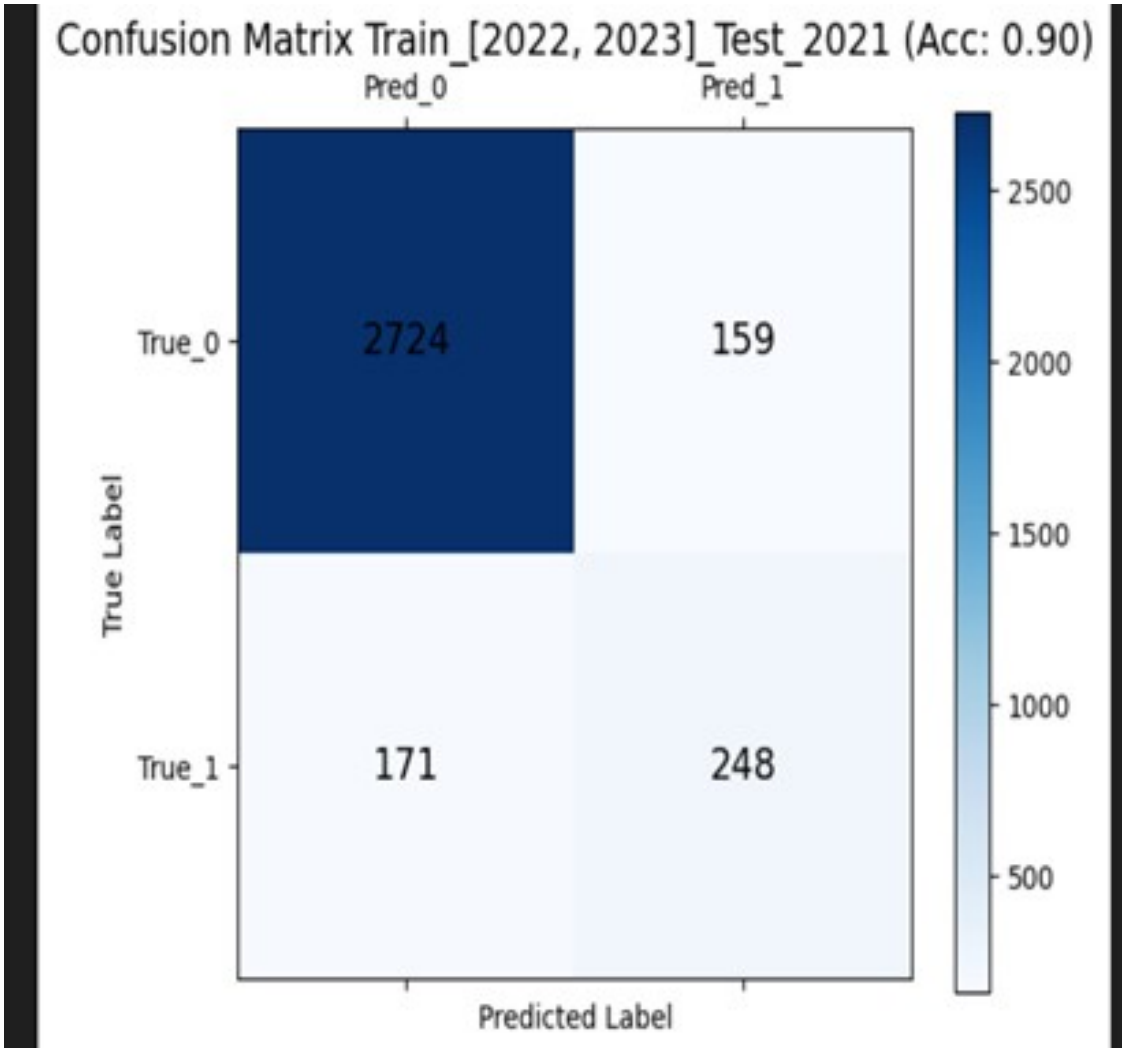


Figure 3.7: Confusion Matrix for Random Forest (Testing on 2021)

Table 3.7: Evaluation Metrics for Random Forest (Testing on 2021)

Metric	**Rice**	**Cotton**
F1-Score	0.60	0.94
Precision	0.61	0.94
Recall	0.59	0.94

Accuracy: 0.8758
Parameters used: 'n_estimators': 100, 'max_depth': 15, 'min_samples_leaf': 1

3.5.2 Testing on 2022

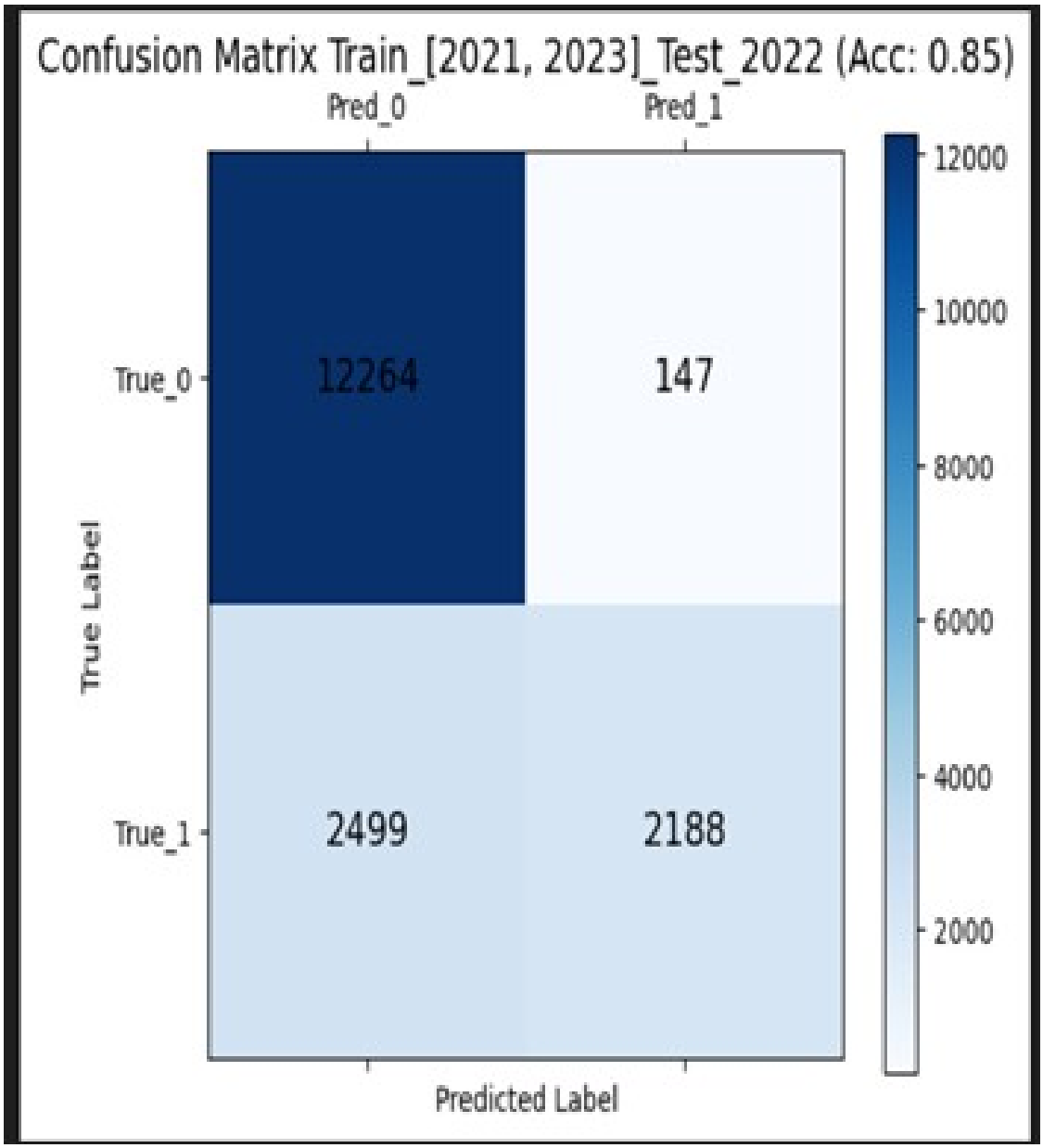


Figure 3.8: Confusion Matrix for Random Forest (Testing on 2022)

Table 3.8: Evaluation Metrics for Random Forest (Testing on 2022)

Metric	**Rice**	**Cotton**
F1-Score	0.62	0.90
Precision	0.94	0.83
Recall	0.47	0.99

Accuracy: 0.9010
Parameters used: 'n_estimators': 50, 'max_depth': 5, 'min_samples_leaf': 1

3.5.3 Testing on 2023

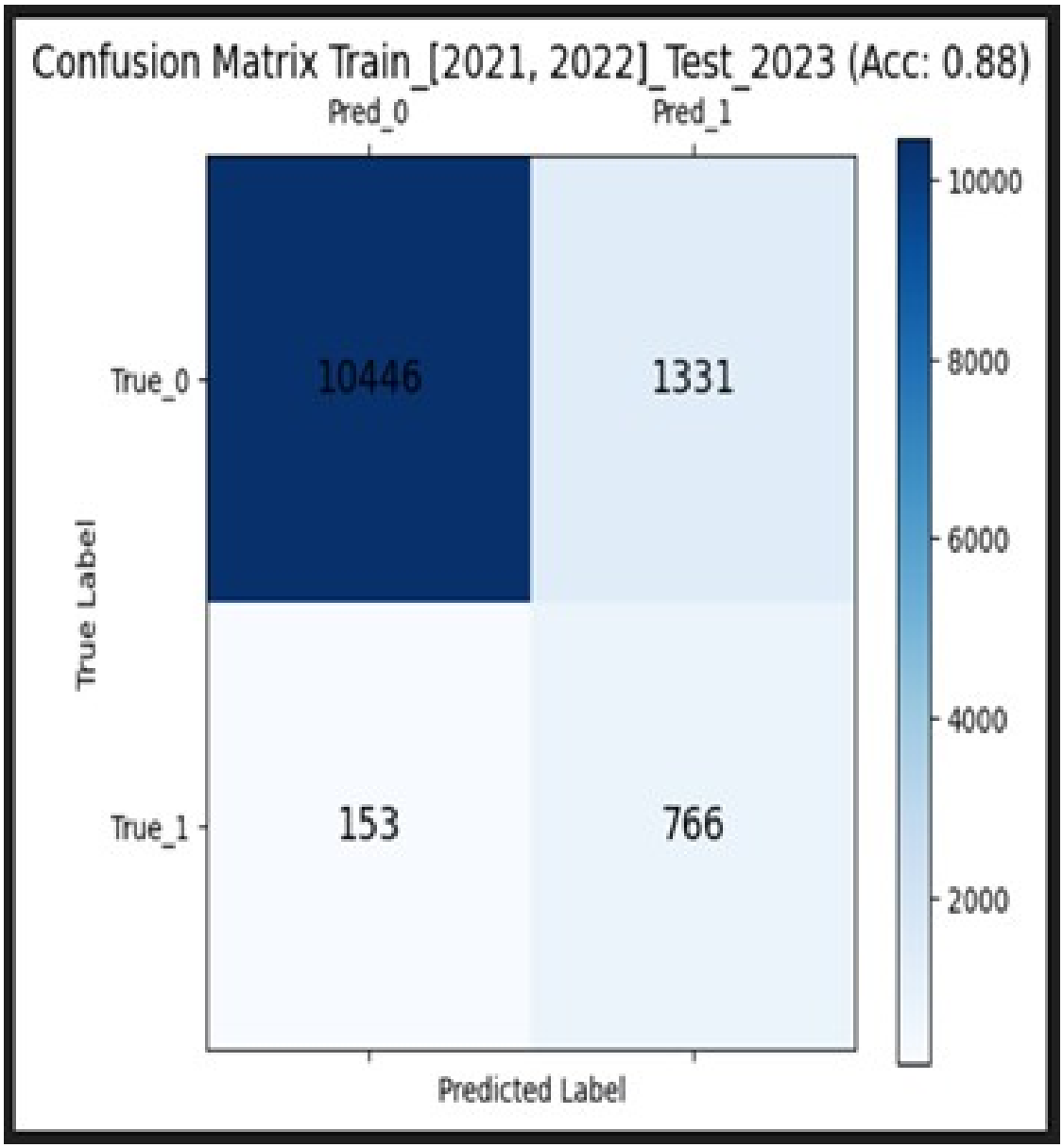


Figure 3.9: Confusion Matrix for Random Forest (Testing on 2023)

Table 3.9: Evaluation Metrics for Random Forest (Testing on 2023)

Metric	**Rice**	**Cotton**
F1-Score	0.51	0.93
Precision	0.38	0.99
Recall	0.83	0.90

Accuracy: 0.88
Parameters used: 'n_estimators': 50, 'max_depth': 5, 'min_samples_leaf': 2

3.6 Support Vector Machine (SVM)

3.6.1 Testing on 2021

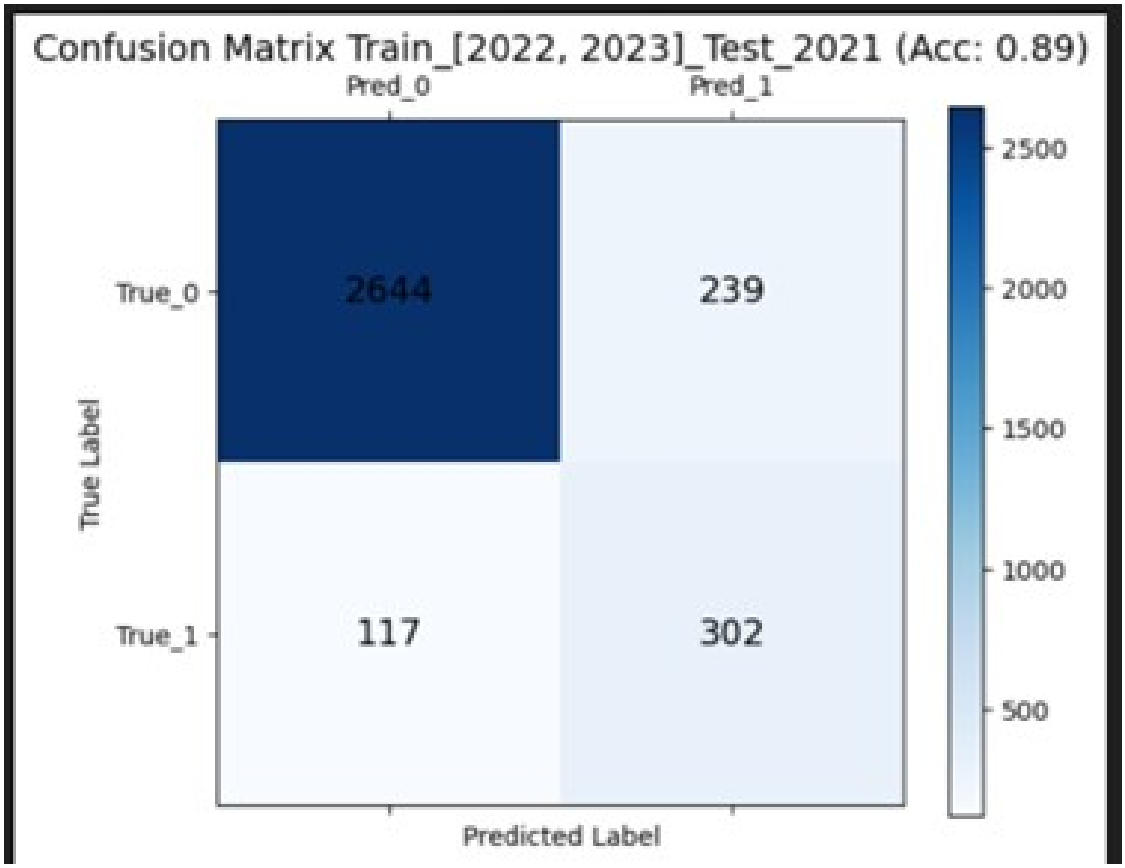


Figure 3.10: Confusion Matrix for Support Vector Machine (Testing on 2021)

Table 3.10: Evaluation Metrics for Support Vector Machine (Testing on 2021)

Metric	Rice	Cotton
F1-Score	0.63	0.94
Precision	0.56	0.96
Recall	0.72	0.92

Accuracy: 0.89
Parameters used: 'C': 0.1, 'gamma': 0.1, 'kernel': 'rbf'

3.6.2 Testing on 2022

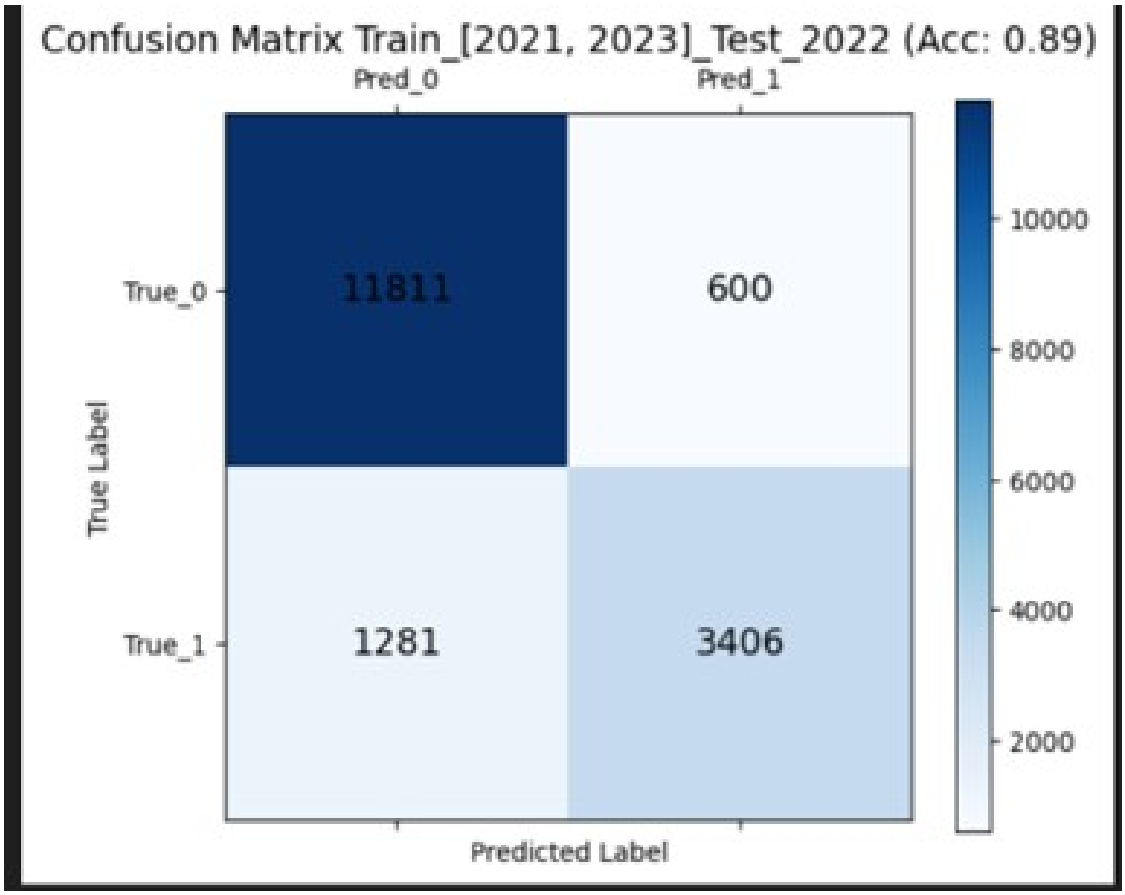


Figure 3.11: Confusion Matrix for Support Vector Machine (Testing on 2022)

Table 3.11: Evaluation Metrics for Support Vector Machine (Testing on 2022)

Metric	**Rice**	**Cotton**
F1-Score	0.79	0.93
Precision	0.85	0.95
Recall	0.73	0.95

Accuracy: 0.9042
Parameters used: 'C': 0.1, 'gamma': 'scale', 'kernel': 'rbf'

3.6.3 Testing on 2023

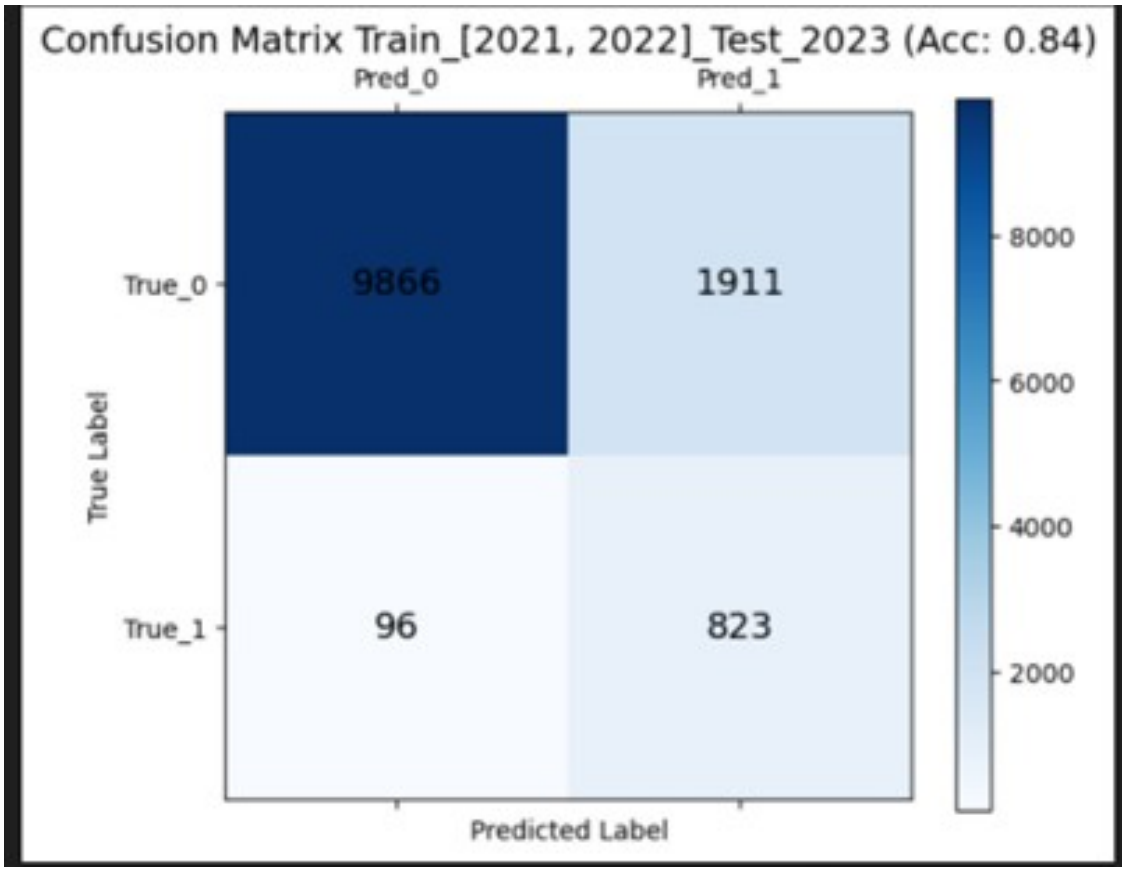


Figure 3.12: Confusion Matrix for Support Vector Machine (Testing on 2023)

Table 3.12: Evaluation Metrics for Support Vector Machine (Testing on 2023)

Metric	**Rice**	**Cotton**
F1-Score	0.45	0.91
Precision	0.30	0.99
Recall	0.90	0.84

Accuracy: 0.8413
Parameters used: 'C': 0.1, 'gamma': 'scale', 'kernel': 'rbf'

Chapter 4

Unsupervised Learning

This chapter describes the implementation and evaluation of unsupervised learning models for NDVI-based crop classification. The models include K-Means Clustering, Hierarchical Clustering, DBSCAN, and Gaussian Mixture Model (GMM). For each model, clustering is performed on the original dataset (without PCA) and on a reduced dataset (with PCA). The performance of each clustering approach is analyzed using metrics such as Silhouette Score, Confusion Matrix, and Cluster Purity.

4.1 Overview of Unsupervised Models

The following unsupervised learning models were implemented in this project:

- **K-Means Clustering:** K-Means is a centroid-based clustering algorithm that partitions the dataset into k clusters. It initializes k cluster centroids and iteratively updates them by minimizing the within-cluster variance. Data points are assigned to the nearest cluster centroid based on Euclidean distance. The algorithm stops once the centroids stabilize or the maximum number of iterations is reached.

- **Hierarchical Clustering:** Hierarchical Clustering builds a hierarchy of clusters using either a top-down (divisive) or bottom-up (agglomerative) approach. In agglomerative clustering, each data point starts as its own cluster, and clusters are merged iteratively based on a distance metric (e.g., Euclidean) and linkage criteria (e.g., single, complete, or average linkage) until a single cluster is formed or a desired number of clusters is reached.

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):** DBSCAN is a density-based clustering algorithm that identifies clusters of points that are closely packed together while marking points in low-density regions as noise. It requires two key parameters: ε (the maximum distance between two points in the same cluster) and $MinPts$ (the minimum number of points required to form a dense region). Unlike K-Means, DBSCAN can detect arbitrarily shaped clusters and identify outliers as noise.

- **Gaussian Mixture Model (GMM):** GMM assumes that the dataset is a mixture of several Gaussian distributions, each representing a cluster. It uses the Expectation-Maximization (EM) algorithm to iteratively update the parameters of these Gaussians (mean, variance, and weight) until convergence. Unlike K-Means, GMM considers both the mean and covariance, allowing for ellipsoidal clusters instead of spherical ones. Each data point has a probability of belonging to multiple clusters.

K-Means Clustering Algorithm

K-Means clustering was applied to both an undersampled dataset and a SMOTE-based dataset to evaluate clustering performance. The clustering was performed with and without dimensionality reduction using Principal Component Analysis (PCA).

Elbow Method

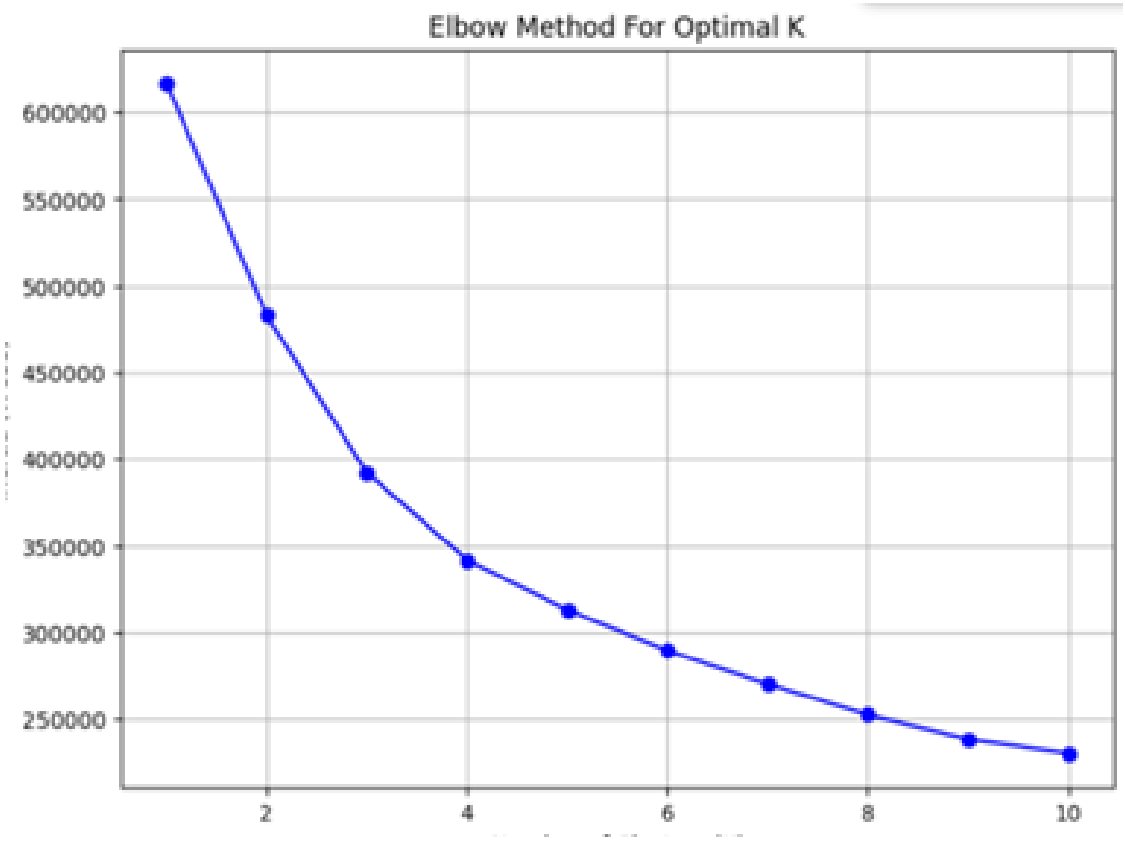


Figure 4.1: Elbow Method for Optimal K Selection

The results were evaluated using **cluster purity**, which measures the correctness of clusters relative to ground truth labels, and the **silhouette score**, which assesses the cohesion and separation of clusters. Below is a table summarizing the results.

Table 4.1: Clustering Results Summary

Dataset	Cluster Purity (No PCA)	Silhouette Score (No PCA)	Cluster Purity (With P
Undersampled	0.7798	0.2843	0.7740
SMOTE	0.7094	0.2600	0.6920

The undersampled dataset achieves better clustering results compared to the SMOTE-based dataset, with higher cluster purity and silhouette scores overall. Applying PCA enhances the silhouette scores for both datasets, suggesting that dimensionality reduction improves the clarity of the clusters. The undersampled dataset with PCA demonstrates the best performance, making it the most effective approach for K-Means clustering in this analysis.

Visualizations

–

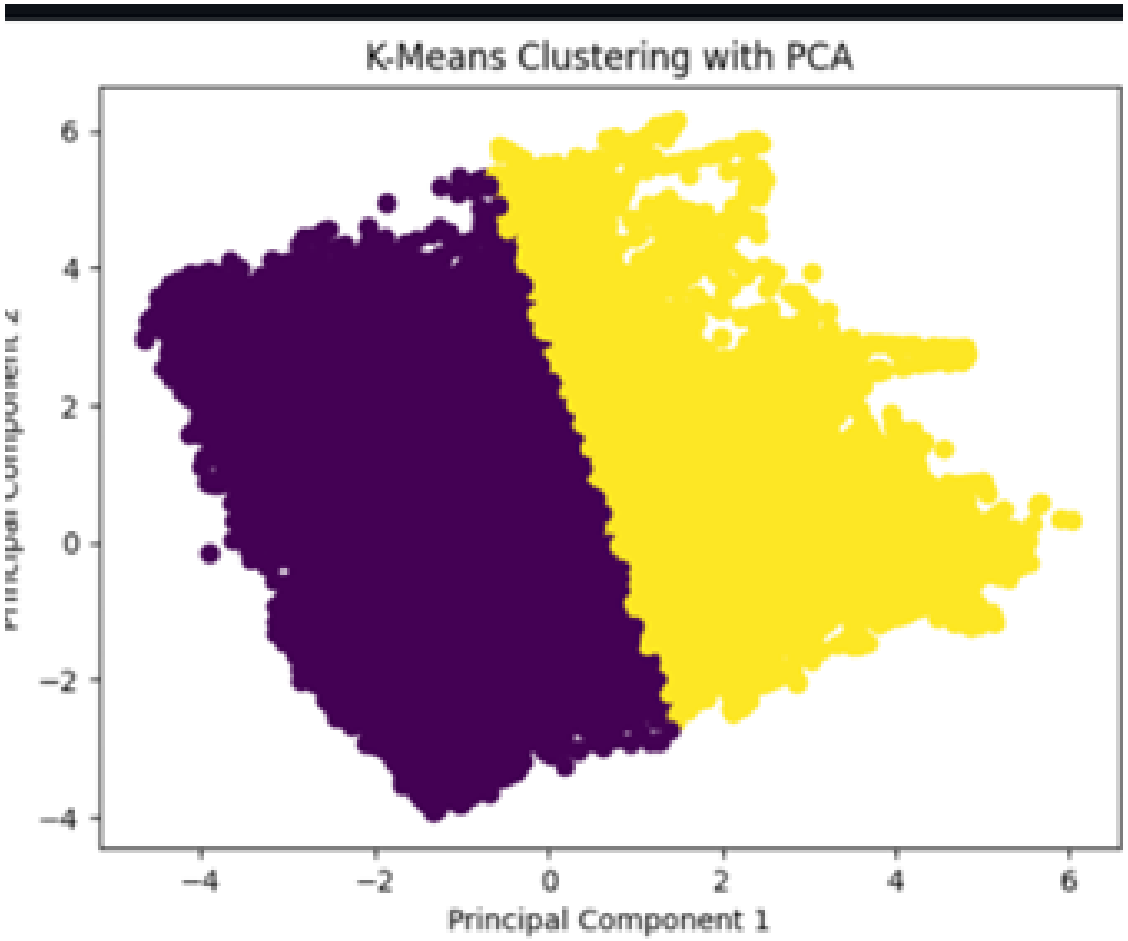


Figure 4.2: Clustering Results for SMOTE-based Dataset

Figure 4.3: Clustering Results for Undersampled-based Dataset

4.2 DBSCAN Clustering Analysis

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) was applied to both an undersampled dataset and a SMOTE-based dataset to evaluate clustering performance. The clustering was performed with and without dimensionality reduction using Principal Component Analysis (PCA). The evaluation metrics used were cluster purity, which indicates the alignment of the clusters with the true labels, and silhouette score (not provided in the data but typically used in DBSCAN analysis for measuring the cohesion and separation of clusters). Below is a comparison of the results for cluster purity.

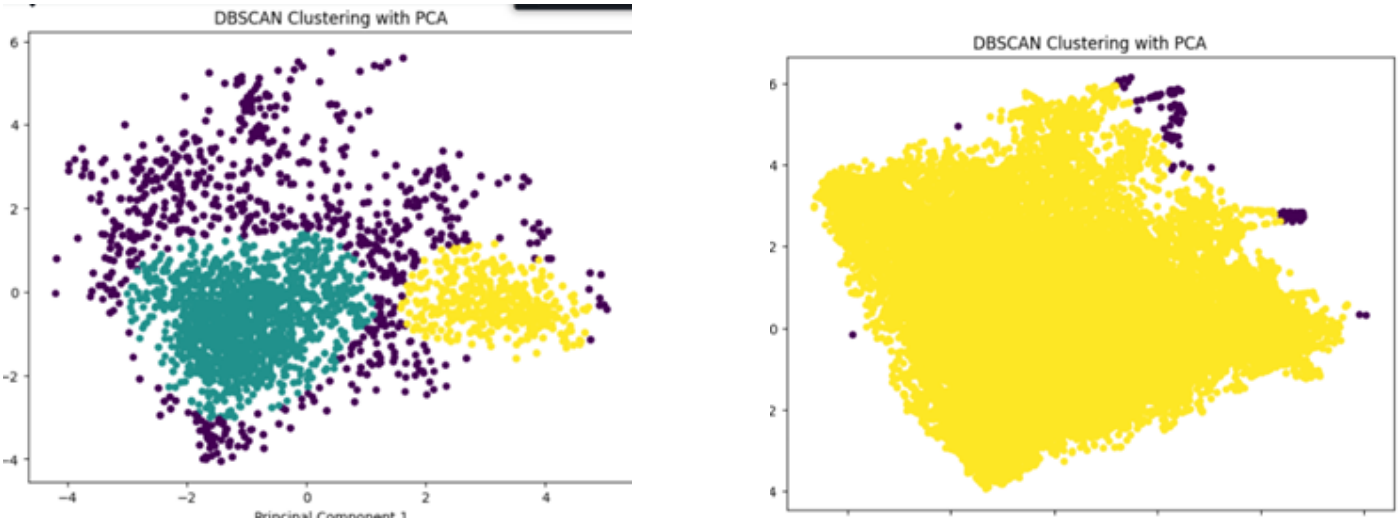
Data set	Cluster Purity (No PCA)	Cluster Purity (With PCA)
Undersampled	0.5096	0.8560
SMOTE	0.5577	0.5177

Table 4.2: Comparison of Cluster Purity for DBSCAN across different datasets and preprocessing methods

4.2.1 Important Considerations

It is crucial to note that while DBSCAN shows the highest purity score, particularly with the undersampled dataset and PCA (0.8560), there are important caveats to consider:

- The algorithm identifies 3 distinct classes and categorizes some points as noise
- The noise class is not included in the purity calculation
- Points classified as noise are not truly noise but rather misclassified cotton or rice samples
- While the purity formula excludes these misclassified points, they represent a limitation in the clustering performance



(a) Undersampled Data DBSCAN Clustering (b) SMOTE-based Data DBSCAN Clustering

Figure 4.4: DBSCAN Clustering Visualization

4.2.2 Performance Analysis

The undersampled dataset provides superior clustering results compared to the SMOTE-based dataset, particularly when PCA is applied. The undersampled dataset with PCA achieves the highest cluster purity (0.8560), making it the most effective configuration for DBSCAN clustering in this analysis. The SMOTE-based dataset, on the other hand, shows a drop in performance when PCA is used, indicating that SMOTE-generated data may introduce challenges for DBSCAN.

4.2.3 Conclusion

Based on the analysis, the undersampled dataset with PCA is the preferred approach for clustering with DBSCAN. However, it's important to acknowledge the limitation that some valid data points are being misclassified as noise, which affects the overall effectiveness of the clustering solution despite the high purity score.

4.3 Gaussian Mixture Models Analysis

Gaussian Mixture Models (GMM) were applied to both an undersampled dataset and a SMOTE-based dataset to evaluate clustering performance. The analysis was conducted with and without dimensionality reduction using Principal Component Analysis (PCA). The clustering performance was assessed using cluster purity and silhouette score to determine the accuracy and quality of the clusters.

4.3.1 Performance Metrics

The results were evaluated using two key metrics:

- **Cluster Purity:** Measures the correctness of clusters relative to ground truth labels
- **Silhouette Score:** Assesses the cohesion and separation of clusters

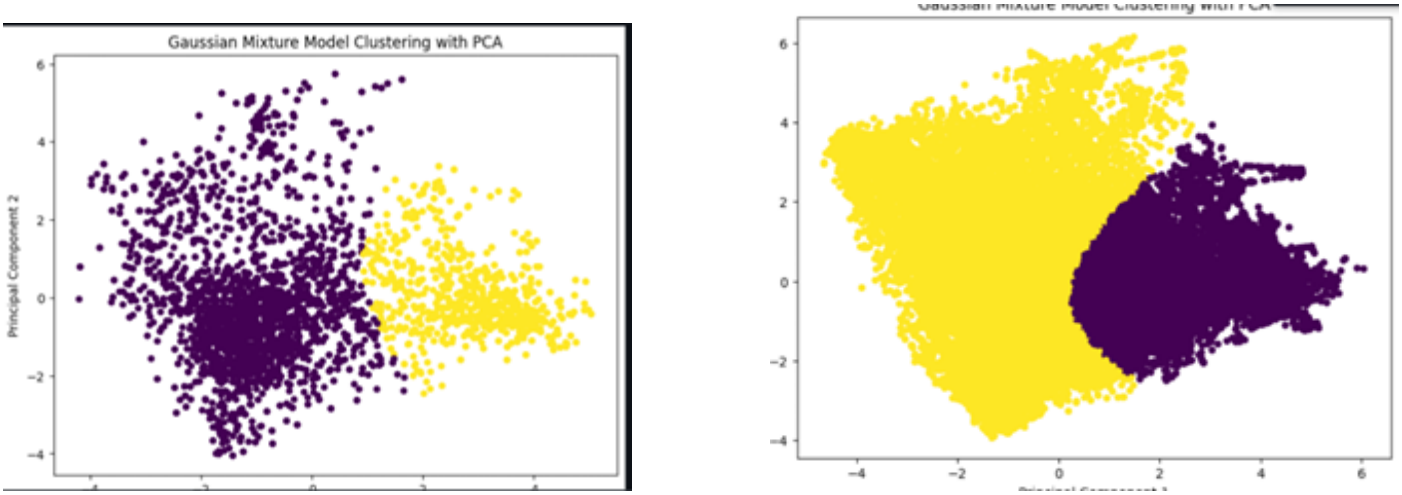
Data set	Cluster Purity (No PCA)	Silhouette Score (No PCA)	Cluster Purity (With PCA)	Silhouette Score (With PCA)
Undersampled	0.7810	0.2551	0.7592	0.4757
SMOTE	0.5812	0.1230	0.6400	0.3600

Table 4.3: Comparison of GMM Performance Metrics across different datasets and preprocessing methods

4.3.2 Analysis of Results

The undersampled dataset consistently outperforms the SMOTE-based dataset in both evaluation metrics:

- **Without PCA:**
  - Undersampled dataset achieves higher cluster purity (0.7810 vs 0.5812)
  - Better silhouette score (0.2551 vs 0.1230)
- **With PCA:**
  - Undersampled dataset maintains superior performance
  - Significant improvement in silhouette scores for both datasets
  - Best balance achieved with undersampled dataset (purity: 0.7592, silhouette: 0.4757)



(a) Undersampled Data GMM Clustering

(b) SMOTE-based Data GMM Clustering

Figure 4.5: GMM Clustering Visualization

4.3.3 Conclusion

The analysis demonstrates that the undersampled dataset with PCA provides the optimal configuration for Gaussian Mixture Models clustering. While there is a slight decrease in cluster purity when applying PCA to the undersampled dataset (from 0.7810 to 0.7592), this is offset by a substantial improvement in the silhouette score (from 0.2551 to 0.4757), indicating better-defined and more separable clusters. Therefore, the undersampled dataset with PCA is recommended as the preferred approach for GMM clustering in this analysis.

4.4 Hierarchical Clustering:

Hierarchical clustering was applied to both an undersampled dataset and a SMOTE-based dataset to evaluate the clustering performance. The analysis was performed with and without dimensionality reduction using Principal Component Analysis (PCA). The primary evaluation metric used was cluster purity, which measures how well the clusters correspond to the true labels. Below is a summary of the results for cluster purity.

Data set	**Cluster Purity (No PCA)**	**Cluster Purity (With PCA)**
Undersampled	0.6593	0.6943
SMOTE	0.74	0.7

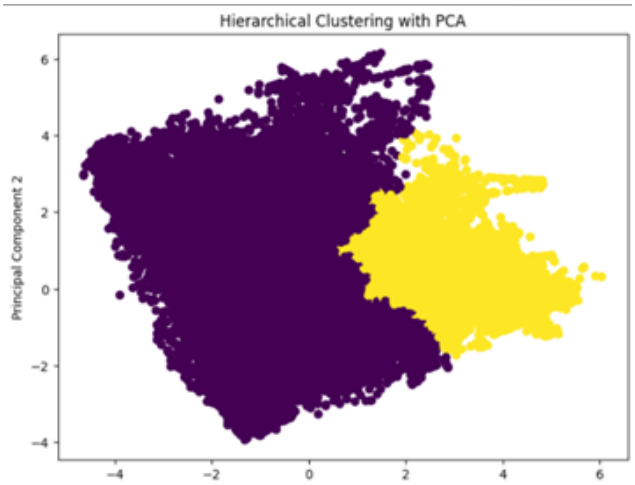
Table 4.4: Comparison of Cluster Purity for DBSCAN across different datasets and preprocessing methods

4.4.1 Performance Analysis

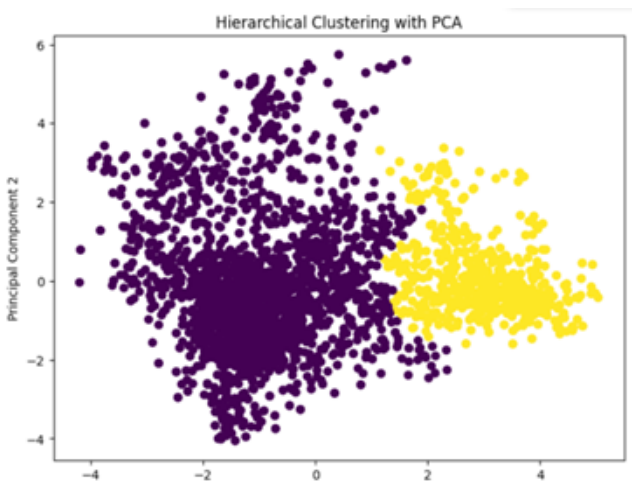
The undersampled dataset consistently outperforms the SMOTE-based dataset in terms of cluster purity, both with and without PCA. The slight decrease in cluster purity after applying PCA does not undermine its effectiveness, as it still achieves higher purity compared to the SMOTE-based dataset. Therefore, the undersampled dataset is the more effective choice for hierarchical clustering in this analysis.

4.4.2 Conclusion

Based on the analysis, the undersampled dataset with PCA is the preferred approach for clustering with DBSCAN. However, it's important to acknowledge the limitation that some valid data points are



(a) Undersampled Data Hierarchical Clustering



(b) SMOTE-based Data Hierarchical Clustering

Figure 4.6: Hierarchical Clustering Visualization

being misclassified as noise, which affects the overall effectiveness of the clustering solution despite the high purity score.

Chapter 5

Model Comparison

5.1 Supervised Models

Table 5.1: Comparison of Supervised Learning Models

Model	Accuracy	Precision	Recall	F1-Score
XGBoost	0.87	0.85	0.86	0.87
Bagging	0.87	0.87	0.86	0.87
Random Forest	0.89	0.87	0.87	0.88
SVM	0.87	0.88	0.87	0.88

5.2 Unsupervised Models

Table 5.2: Comparison of Clustering Models

Model	Cluster Purity	Best Silhousette Score Accuracy	PCA Applied
K-Means	0.7798	0.4700	Yes
Hierarchical	0.77	N/A	Yes
DBSCAN	[0.86]	N/A	Yes
GMM	0.7810	0.4757	Yes