National University of Sciences and Technology School of Electrical Engineering and Computer Science

CS 471 Machine Learning Fall 2024

NDVI-Based Crop Classification for Rice and Cotton Using Machine Learning Project Report

Instructor: Professor Dr Moazam Fraz

Submitted By:

Name	CMS
Muhammad Sarmad Saleem	411411
Tayyab Raza	404821

Github Link:

https://github.com/sarmadsaleem333/ML-Semester-Project-Crop-Classifier

Table of Contents

1. **Supervised Machine Learning**
2. **Data Loading**
    - Load Dataset from Code

3. **Data Preprocessing**
   3.1 Null Values
   3.2 Class Imbalanced
   3.3 Class Balanced
   3.4 Combined Dataset
   3.5 Removal of Outliers
   3.6 Scaling
4. **Feature Selection**
5. **Models**
   i. Random Forest
   ii. XG-Boost
   iii. Bagging
   iv. SVM
6. **Unsupervised Learning**
7. **Unsupervised Algorithms on Both Dataframes**
   - K-Means Clustering Algorithm
   - Gaussian Mixture Models
   - DB Scan Algorithm
   - Hiererical Algorithm

Supervised Learning

Supervised learning is a type of machine learning where the model is trained on labeled data, using input-output pairs to learn a mapping from inputs to desired outputs.

Data Loading:

The data loading and preprocessing process involved systematically organizing and extracting NDVI values for the years 2021, 2022, and 2023. The dataset was stored in a directory named Dataset, with subfolders for each year containing CSV files, where filenames encoded crop type and year information. A custom function, load_and_process_data, was implemented to automate the loading process. This function read all CSV files within a specified year's folder, extracted the crop type from the filenames, and appended it as a new column (CropType). Additionally, a Year column was added to distinguish data across the years. The function iterated over the files, concatenating them into a single Pandas dataframe for each year. This approach allowed us to create processed datasets for 2021, 2022, and 2023, ensuring that each included NDVI values along with metadata for crop type and year. This automated and structured process ensured consistency and prepared the dataset for further analysis and machine learning tasks.

Cotton 3 items

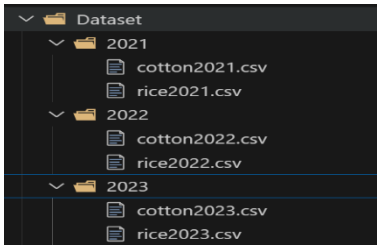
Name	Last modified	File size
cotton2021.csv	Oct 23, 2024	404 KB
cotton2022.csv	Oct 23, 2024	2 MB
cotton2023.csv	Oct 23, 2024	2 MB

Rice 3 items

Name	Last modified	File size
rice2021.csv	Oct 23, 2024	59 KB
rice2022.csv	Oct 23, 2024	651 KB
rice2023.csv	Oct 23, 2024	128 KB

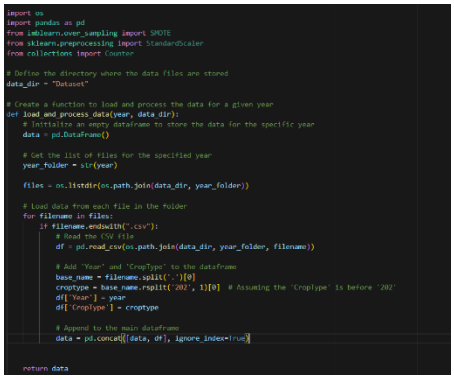
Changed Some Format For Better Understanding:

Now we Have a Main folder and inside main folder 3 subfolders name 2021,2022,2023 and inside each folder rice and cotton csv files.



Load Dataset from code :

In this code we have to give just Dataset folder name and then load that folders.



```
import os
import pandas as pd
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler
from collections import Counter

# Define the directory where the data files are stored
data_dir = "Dataset"

# Create a function to load and process the data for a given year
def load_and_process_data(year, data_dir):
    # Initialize an empty dataframe to store the data for the specific year
    data = pd.DataFrame()

    # Get the list of files for the specified year
    year_folder = str(year)

    files = os.listdir(os.path.join(data_dir, year_folder))

    # Load data from each file in the folder
    for filename in files:
        if filename.endswith(".csv"):
            # Read the CSV file
            df = pd.read_csv(os.path.join(data_dir, year_folder, filename))

            # Add 'Year' and 'CropType' to the dataframe
            base_name = filename.split('.')[0]
            croptype = base_name.rsplit('_202', 1)[0]  # Assuming the 'Croptype' is before '202'
            df['Year'] = year
            df['Croptype'] = croptype

            # Append to the main dataframe
            data = pd.concat([data, df], ignore_index=True)


    return data
```

Null Values:

There are no Null values in the dataset.

Class Imbalanced:

The dataset used in this project was highly imbalanced, with a significantly larger number of samples for cotton compared to rice across all years. This imbalance posed a challenge for machine learning models, as it could lead to a bias toward the majority class (cotton), resulting in poor generalization and reduced performance on the minority class (rice). To

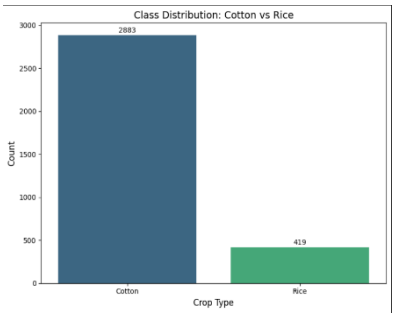
address this issue, we employed a combination of oversampling and undersampling techniques to balance the class distributions.

Oversampling was implemented to increase the number of samples in the minority class, rice, using methods such as Random Oversampling and SMOTE (Synthetic Minority Oversampling Technique). While Random Oversampling works by duplicating existing samples in the minority class, SMOTE generates synthetic samples by interpolating between existing data points, thereby preserving the diversity of the dataset and reducing the risk of overfitting.

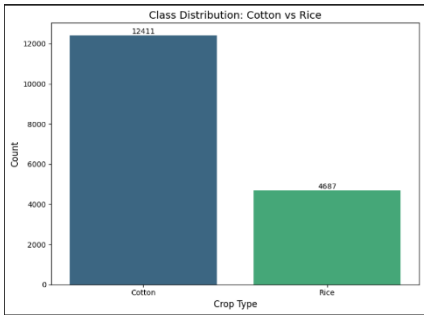
In addition to oversampling, we also applied undersampling, which balances the dataset by reducing the number of samples in the majority class (cotton). By aligning the size of the majority class with the minority class, this approach ensures a more balanced representation without introducing synthetic data.

For our analysis, we created two balanced datasets: one using SMOTE for oversampling and another using undersampling. These approaches ensured that the machine learning models could effectively learn patterns from both classes without being biased by the initial class imbalance, ultimately improving classification performance and robustness.

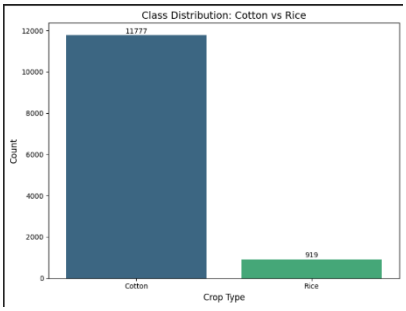
There is a lot of Class Imbalance in the whole dataset. Given are the Graphs:



(2021)



(2022)



(2023)

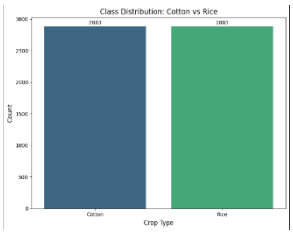
Class Balanced:

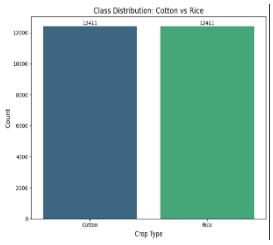
We have applied Following things for Data Imbalance removal and checked results on all but Smote gives the best results, So I continued Work.

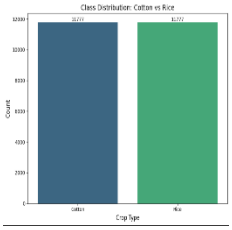
- Smote
- Random oversampling
- Time series augmentation

Smote:

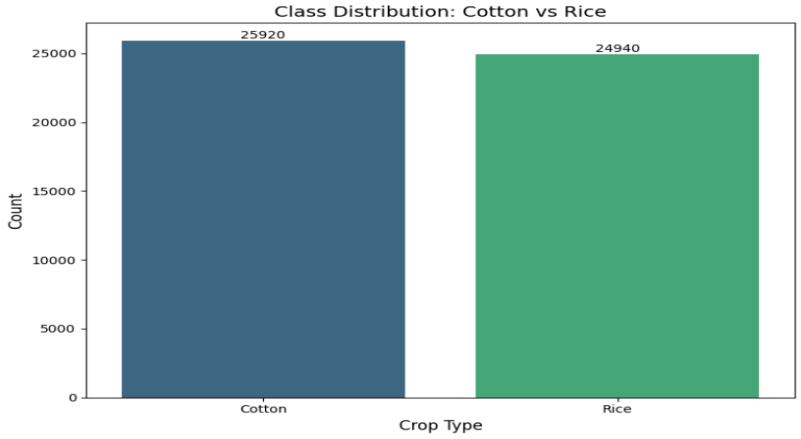
2021 2022 2023







Remove Duplicates for improving Results:



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.

Here the Outliers are disturbing our results and our final results were not good so we applied multiple Outlier removal techniques but Z-Score with threshold =3

Gives best results.

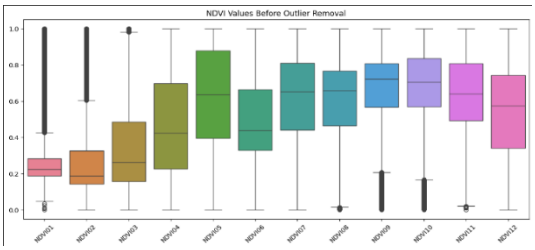
i-Before Z-Score Shape:

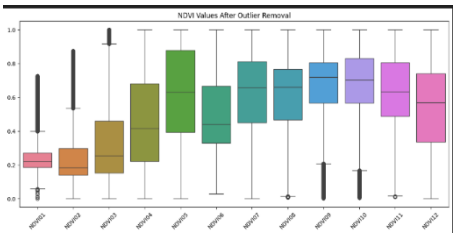
(53524,14)

ii-After Z-Score Shape:

(50860,14)

Visual Representation:





Scaling:

In our analysis, we applied standard scaling to normalize the numerical features of the dataset, ensuring that all features had a mean of 0 and a standard deviation of 1. This step is crucial for improving the performance of machine learning algorithms, especially those that

are sensitive to feature magnitude, such as K-Means clustering and Principal Component Analysis (PCA).

We implemented a flexible scaling function that supports multiple scaling techniques, but selected standard scaling as it is particularly effective when the data follows a normal distribution. The function excluded categorical and target columns (e.g., CropType_rice and Year) to prevent altering non-numerical or classification-related data. Using the StandardScaler from the sklearn.preprocessing library, we standardized the numerical columns by subtracting the mean and dividing by the standard deviation for each feature.

Standard scaling ensured that all features contributed equally to the model training process, reducing the dominance of features with larger numerical ranges and improving the model's convergence and accuracy.

Scaling is Important step in data-preprocessing .As I have applied all the scaling for checking results like

- Standard Scaling
- Mini max
- Robust
- Quantile

But Minmax is giving best results as our data set is almost scaled so minmax improved my results.

Feature Selection:

As in Correleation matrix almost all features are taking part in determining the results so it is not necessary to remove any Column .

Models:

- Random Forest
- XGboost
- Bagging
- SVM

Random Forest:

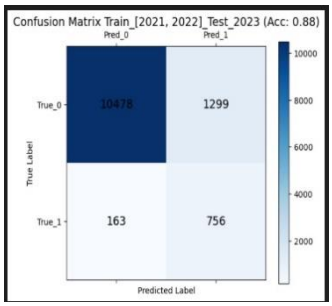
Random Forest:

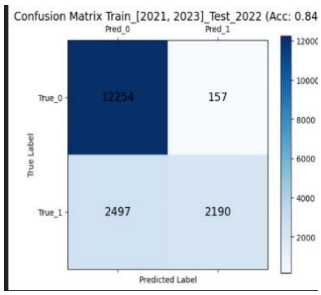
```
--- Training Model 1: Train Years [2021, 2022] → Test Year 2023 ---
Training RandomForestClassifier...

Accuracy: 0.8848
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.89      0.93    11777
           1       0.37      0.82      0.51      919

    accuracy                           0.88    12696
   macro avg       0.68      0.86      0.72    12696
weighted avg       0.94      0.88      0.90    12696
```

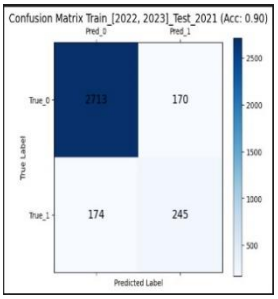
```
--- Training Model 2: Train Years [2021, 2023] → Test Year 2022 ---
Training RandomForestClassifier...

Accuracy: 0.8448
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.99      0.90    12411
           1       0.93      0.47      0.62     4687

    accuracy                           0.84    17098
   macro avg       0.88      0.73      0.76    17098
weighted avg       0.86      0.84      0.83    17098
```

```
--- Training Model 3: Train Years [2022, 2023] → Test Year 2021 ---
Training RandomForestClassifier...

Accuracy: 0.8958
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.94      0.94      2883
           1       0.59      0.58      0.59       419

    accuracy                           0.90      3302
   macro avg       0.77      0.76      0.76      3302
weighted avg       0.90      0.90      0.90      3302
```

Confusion matrix:







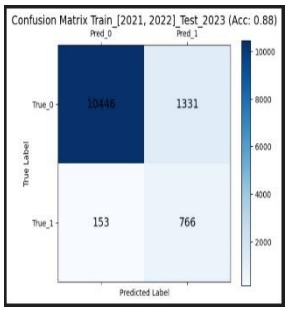
Random Forest(with Grid Search:

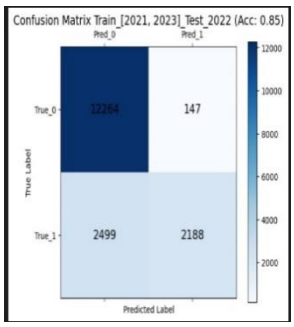
```
--- Training Model 1: Train Years [2021, 2022] → Test Year 2023 ---
Training RandomForestClassifier with GridSearchCV...
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best hyperparameters found: {'max_depth': 50, 'min_samples_leaf': 1, 'n_estimators': 300}

Accuracy: 0.8831
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.86      0.93     11777
           1       0.37      0.83      0.51       919

    accuracy                           0.88     12696
   macro avg       0.68      0.86      0.72     12696
weighted avg       0.94      0.88      0.90     12696
```

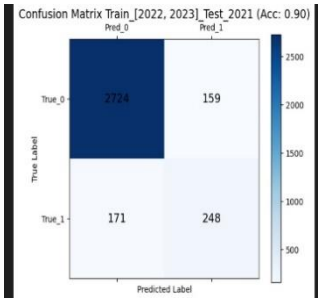
```
--- Training Model 2: Train Years [2021, 2023] → Test Year 2022 ---
Training RandomForestClassifier with GridSearchCV...
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best hyperparameters found: {'max_depth': 50, 'min_samples_leaf': 1, 'n_estimators': 300}

Accuracy: 0.8452
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.98      0.90     12411
           1       0.64      0.17      0.02      4687

    accuracy                           0.85     17098
   macro avg       0.88      0.73      0.76     17098
weighted avg       0.86      0.85      0.83     17098
```

```
--- Training Model 3: Train Years [2022, 2023] → Test Year 2021 ---
Training RandomForestClassifier with GridSearchCV...
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best hyperparameters found: {'max_depth': 50, 'min_samples_leaf': 1, 'n_estimators': 300}

Accuracy: 0.9001
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.94      0.94      2883
           1       0.61      0.59      0.60       419

    accuracy                           0.90      3302
   macro avg       0.78      0.77      0.77      3302
weighted avg       0.90      0.90      0.90      3302
```

Confusion matrix:







XG-Boost

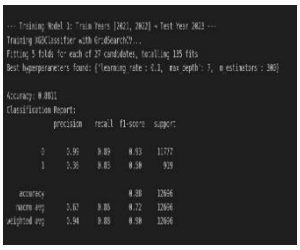
Results:

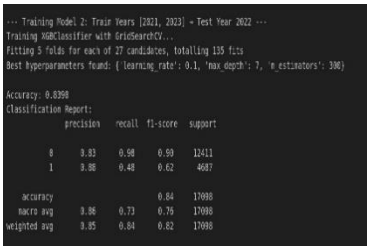
```
--- Training Model 1: Train Years [2021, 2022] → Test Year 2023 ---
Training XGBClassifier...

Accuracy: 0.8886
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.88      0.93     11777
           1       0.36      0.84      0.50       919

    accuracy                           0.88     12696
   macro avg       0.67      0.86      0.72     12696
weighted avg       0.94      0.88      0.90     12696
```

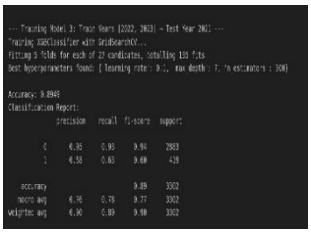
```
--- Training Model 2: Train Years [2021, 2023] → Test Year 2022 ---
Training XGBClassifier...

Accuracy: 0.8389
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.98      0.90     12411
           1       0.89      0.47      0.61      4687

    accuracy                           0.84     17098
   macro avg       0.86      0.72      0.75     17098
weighted avg       0.85      0.84      0.82     17098
```

```
--- Training Model 3: Train Years [2022, 2023] → Test Year 2021 ---
Training XGBClassifier...

Accuracy: 0.8919
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.93      0.94      2883
           1       0.57      0.63      0.60       419

    accuracy                           0.89      3302
   macro avg       0.76      0.78      0.77      3302
weighted avg       0.90      0.89      0.89      3302
```

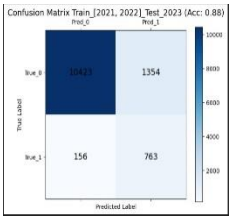
XGBoost (with Grid Search):

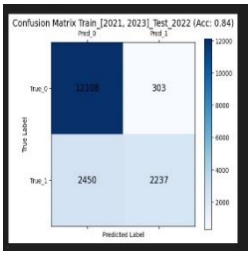


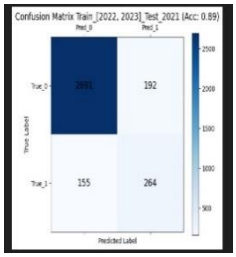




Confusion matrix:

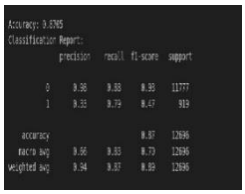


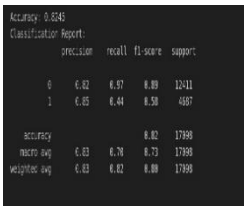


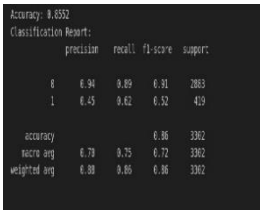


Bagging

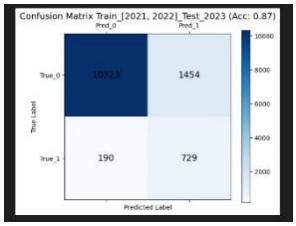
Results:

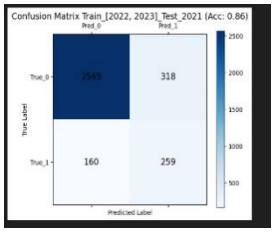


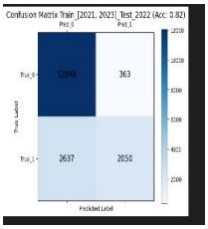




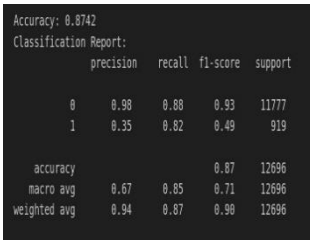
Confusion matrix:

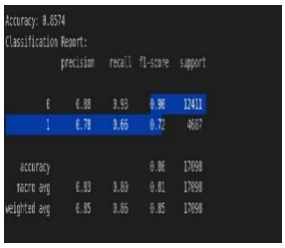


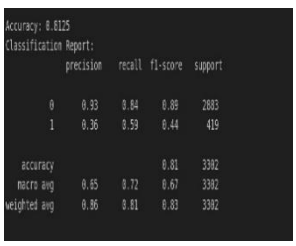




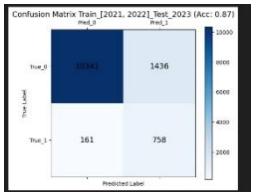
Bagging(with grid Search)

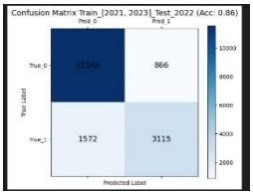


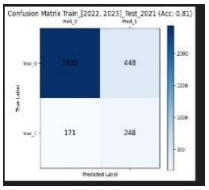




Confusion matrix:

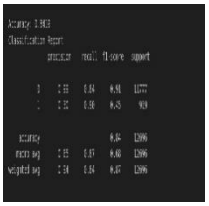


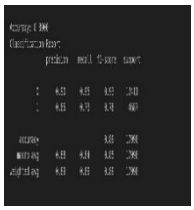


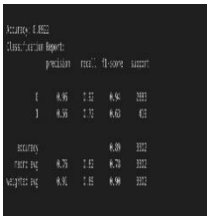


SVM

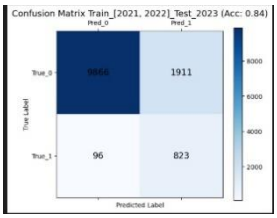
Results

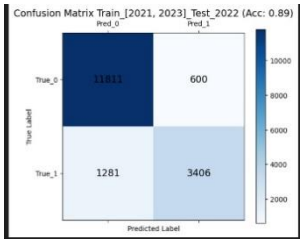


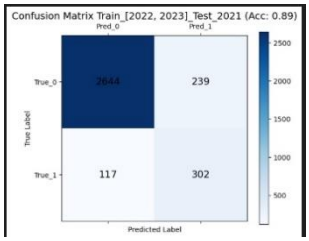




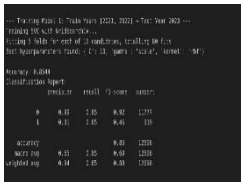
Confusion matrix:

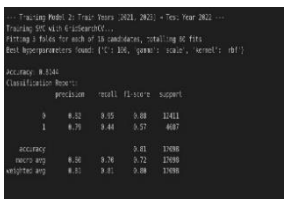


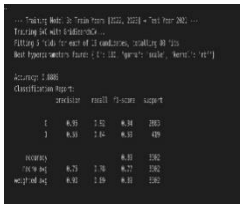




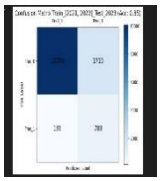
SVM(with Grid Search)

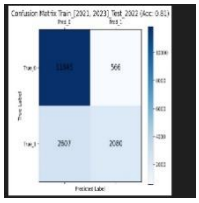


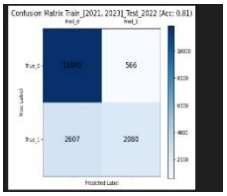




Confusion matrix:







Best Model:

Here the best model according to above results is SVM and it has performed well and there results shown in table and confusion matrix above.

Unsupervised Learning

Unsupervised learning is a type of machine learning where the model is trained on data without labeled outcomes, aiming to identify inherent patterns or structures within the data. It is commonly used for clustering, dimensionality reduction, and anomaly detection.

Data Loading :

Same as above loaded in supervised learning

Data Imbalance Issue:

Same techniques as we used in supervised paradigm.

For our analysis, we created two balanced datasets: one using SMOTE for oversampling and another using undersampling. These approaches ensured that the machine learning models could effectively learn patterns from both classes without being biased by the initial class imbalance, ultimately improving classification performance and robustness.

Outliers Removal:

Same as we used in supervised paradigm

Scaling:

In unsupervised we applied Standard rather then Min max because we need to apply PCA here and PCA requires data to be scaled in standard scaling.In our analysis, we applied **standard scaling** to normalize the numerical features of the dataset, ensuring that all features had a mean of 0 and a standard deviation of 1. This step is crucial for improving the performance of machine learning algorithms, especially those that are sensitive to feature magnitude, such as K-Means clustering and Principal Component Analysis (PCA).

We implemented a flexible scaling function that supports multiple scaling techniques, but selected **standard scaling** as it is particularly effective when the data follows a normal

distribution. The function excluded categorical and target columns (e.g., CropType_rice and Year) to prevent altering non-numerical or classification-related data. Using the StandardScaler from the sklearn.preprocessing library, we standardized the numerical columns by subtracting the mean and dividing by the standard deviation for each feature.

Standard scaling ensured that all features contributed equally to the model training process, reducing the dominance of features with larger numerical ranges and improving the model's convergence and accuracy.

Unsupervised Algoritms on Both dataframes

We applied unsupervised algorithms on both dataframes Oversampled (SMOTE) and undersampled with PCA and Non PCA.

Cluster Purity Formula:

Formula:

$$\text{Purity} = \frac{1}{N}\sum_{i=1}^{k}\max_{j}|C_i \cap T_j|$$

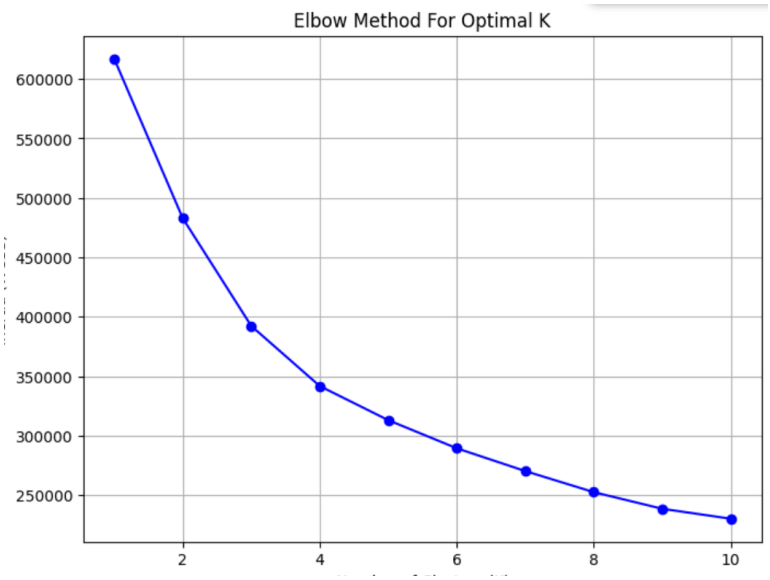
Where:

- N = total number of data points.
- k = number of clusters.
- C_i = the set of data points in cluster i.
- T_j = the set of data points from class j that are assigned to cluster i.
- $|C_i \cap T_j|$ = the number of points from c ↓ j in cluster i.

K Mean 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).

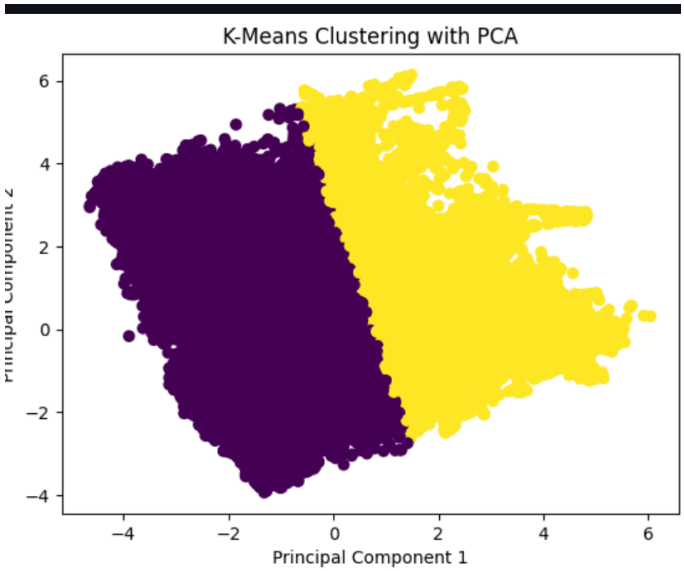
Elbow method:



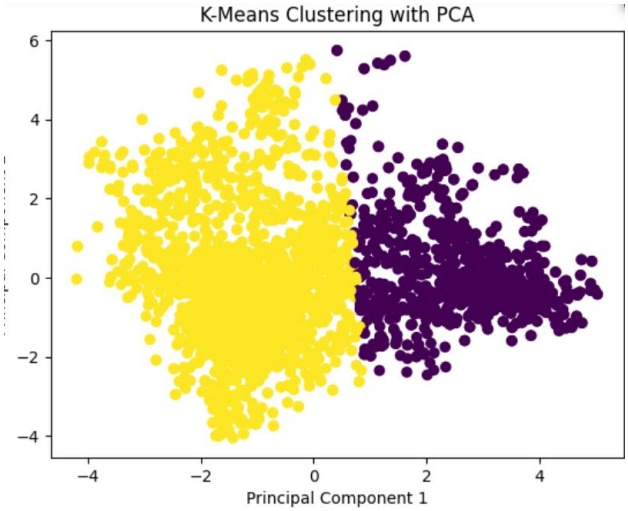
The results were evaluated using **cluster purity**, which measures the correctness of clusters relative to ground truth labels, and the **silhouette score**, which assesses the cohesion and separation of clusters. Below is a table summarizing the results.

Data set	Cluster Purity(No PCA)	Silhouette Score (No PCA	Cluster Purity (With PCA)	Silhouette Score (With PCA)
Undersampled	0.7798	0.2843	0.7740	0.470
SMOTE	0.7094	0.2600	0.6920	0. .405

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.



(Smote based data)



(Undersampled)

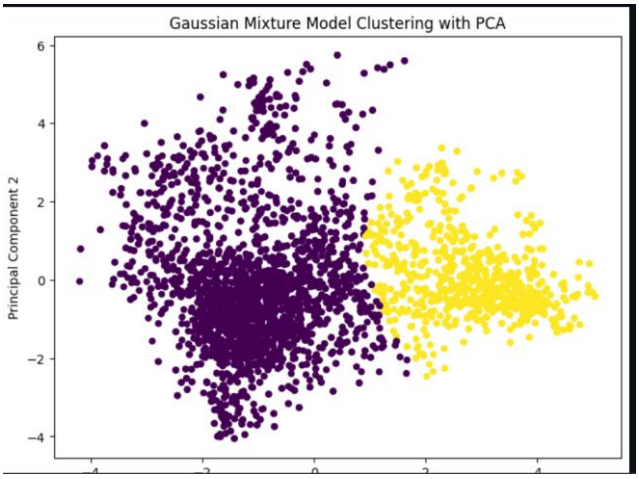
Gaussian Mixture Models:

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. Below are the results summarized in a table.

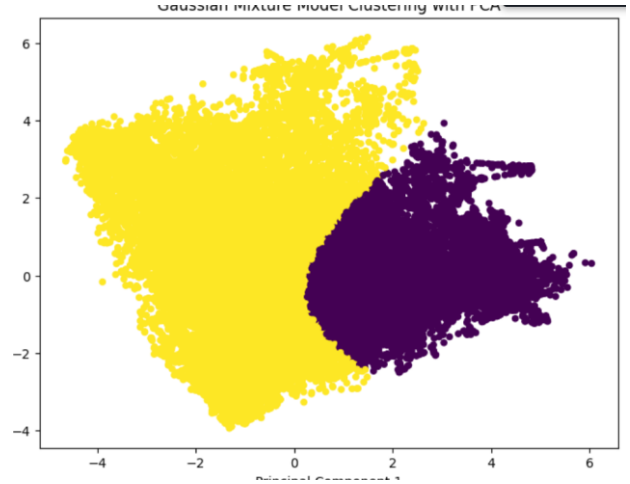
The results were evaluated using **cluster purity**, which measures the correctness of clusters relative to ground truth labels, and the **silhouette score**, which assesses the cohesion and separation of clusters. Below is a table summarizing the results.

Data set	Cluster Purity(No PCA)	Silhouette Score (No PCA	Cluster Purity (With PCA)	Silhouette Score (With PCA)
Undersampled	0.781	0.2551	0.7592	0.4757
SMOTE	0.5812	0.1230	0.6400	0.3600

The undersampled dataset consistently outperforms the SMOTE-based dataset in both cluster purity and silhouette scores. Applying PCA improves the silhouette scores for both datasets, indicating that dimensionality reduction helps in forming clearer clusters. However, the **undersampled dataset with PCA** achieves the best balance between cluster purity (0.7592) and silhouette score (0.4757), making it the preferred approach for clustering using Gaussian Mixture Models.



(Under sampled Data)



(Smote based data)

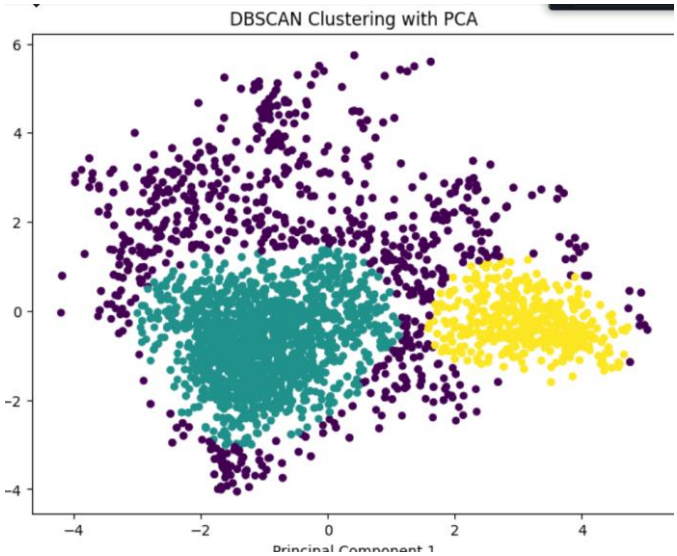
DB Scan algorithm:

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.

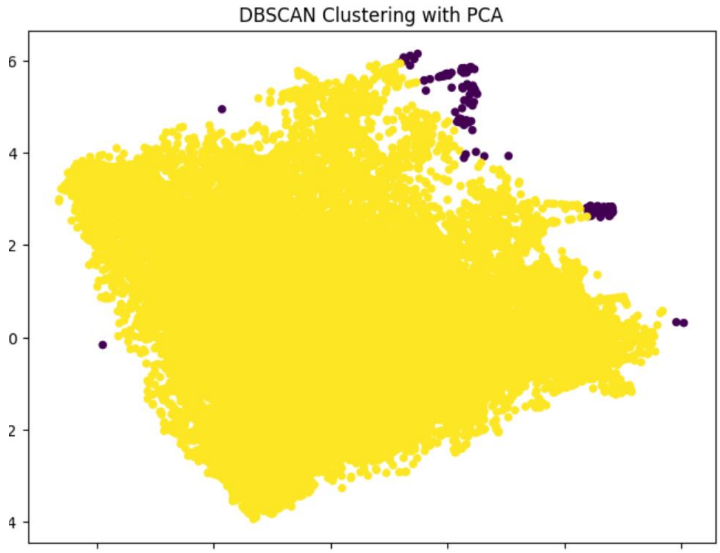
Data set	Cluster Purity(No PCA)	Cluster Purity (With PCA)
Undersampled	0.5096	0.8560
SMOTE	0.5577	0.5177

Because even DBSCAN has highest purity but it identifies 3 classes and one as noise and that noise class is not part of purity calculation The **DBSCAN** algorithm may create more clusters than you have true classes (e.g., 3 clusters instead of 2), but the purity calculation only considers the **valid clusters** that contain actual data points, so that invalid clusters in our case is not noise but they are cotton or rice even though formula ignores them but they are misclassified as noise

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. Therefore, the undersampled dataset with PCA is the preferred approach or clustering with DBSCAN.



(Under sampled Data)



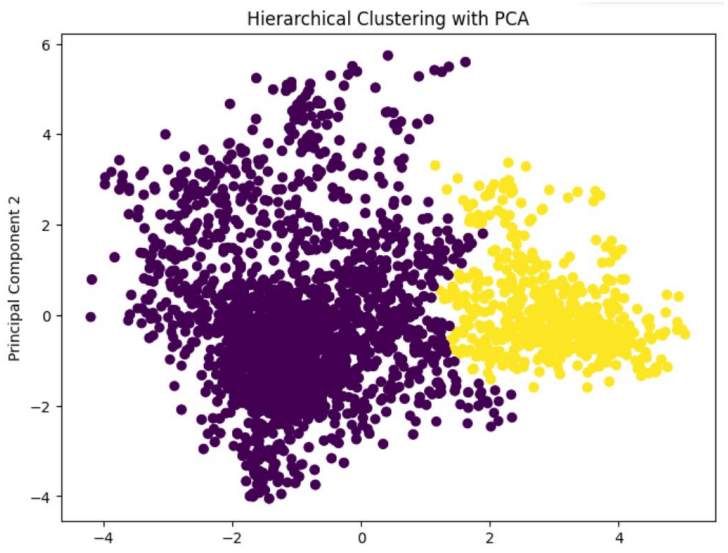
(Smote based data)

Hierarchical Clustering:

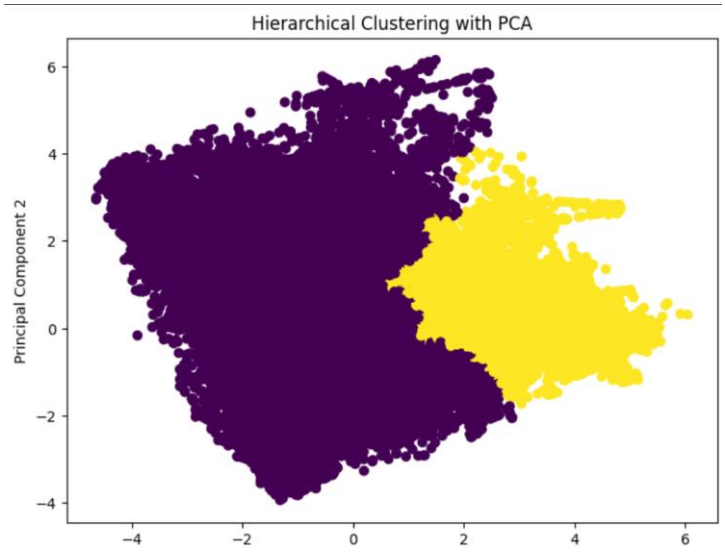
Hierarchical clustering was applied to both an undersampled dataset and a SMOTE-based dataset to evaluate the clustering performance. The analysis was performed with and without dimensionality reduction using Principal Component Analysis (PCA). The primary evaluation metric used was **cluster purity**, 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	0.6943	0.6593
SMOTE	0.7	0.74

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.



(Smote based data)



(Under sampled Data)

Total Model Comparison: K-Means, Gaussian, DBSCAN, and Hierarchical Clustering:

By seeing all above tables

- **Best Overall Performance:** The **Undersampled** dataset consistently outperforms the **SMOTE** dataset across all algorithms, with PCA applied in most cases yielding the best results.
- **K-Means & Gaussian Mixture Models** show high performance with **cluster purity** values close to 0.78, with PCA further improving clustering cohesion.
- **DBSCAN** delivers an exceptional **cluster purity** of **0.8560** with the **undersampled dataset** and PCA, marking it as the top performer in terms of cluster purity but since we can not say it final model because originally we have 2 clusters and it is making some times 3 and sometimes only 1 .
- **Hierarchical Clustering** also performs well with the **undersampled dataset**, showing strong results with **cluster purity = 0.7700**.

Best Model:

According to results we can give the DBSCAN highest model performer but in terms of analytics if we see the original labelled mapping K means and Gaussian are regarded as best performer in this regard.

Because even DBSCAN has highest purity but it identifies 3 classes and one as noise and that noise class is not part of purity calculation The **DBSCAN** algorithm may create more clusters than you have true classes (e.g., 3 clusters instead of 2), but the purity calculation only considers the **valid clusters** that contain actual data points, so that invalid clusters in our case is not noise but they are cotton or rice even though formula ignores them but they are misclassified as noise

Conclusion:

In this project, we applied both **supervised** and **unsupervised** learning techniques to classify agricultural crop types—rice and cotton—based on NDVI data. Through the use of **Supervised Learning**, particularly **Support Vector Machines (SVM)**, we observed strong performance, as SVM effectively handled the labeled data, providing accurate predictions for crop classification. The **unsupervised** learning methods, including **K-Means**, **Gaussian Mixture Models (GMM)**, **DBSCAN**, and **Hierarchical Clustering**, offered interesting insights, although the results showed mixed performance due to the challenges of working with imbalanced data and lack of labeled classes in clustering.

In unsupervised learning, the accuracy varied across different clustering algorithms and methods, such as **SMOTE** and **undersampling**, with some models like DBSCAN showing promising cluster purity, especially with dimensionality reduction using **PCA**. However, as expected, the clustering results were not as straightforward as supervised methods since the models had to infer patterns without the help of labeled data. Visualizations and graphs generated throughout the project helped in understanding the performance of each model and provided clear comparisons of their clustering capabilities.

This project has been a valuable learning experience, deepening our understanding of machine learning techniques, including challenges in dealing with imbalanced datasets, choosing appropriate models, and evaluating clustering results. The insights gained from both **supervised** and **unsupervised** methods provide a strong foundation for further exploration in machine learning, especially in agricultural applications like crop classification.