Crop Classification Using the NDVI Values

Contents

0. Required Libraries

Library/Function	Purpose
!pip install imbalanced-learn	Installs the imbalanced-learn library for handling imbalanced datasets.
!pip install xgboost	Installs the xgboost library for implementing the XGBoost algorithm, useful for classification tasks.
import numpy as np	Imports NumPy, a fundamental library for numerical computing.
import pandas as pd	Imports Pandas for data manipulation and analysis using DataFrames.
import matplotlib.pyplot as plt	Imports Matplotlib for basic plotting and visualizations.
import seaborn as sns	Imports Seaborn for advanced statistical visualizations and enhanced plots.
from sklearn import datasets	Imports datasets module from scikit-learn to access toy datasets for experimentation.
from sklearn.model_selection import train_test_split	Splits the dataset into training and testing subsets for model evaluation.
from sklearn.preprocessing import StandardScaler	Standardizes features by removing the mean and scaling to unit variance.
from sklearn.svm import SVC	Imports the Support Vector Classifier (SVC) for classification tasks.
from sklearn.metrics import accuracy_score, confusion_matrix	Imports metrics for evaluating classification models, including accuracy and confusion matrix.
from matplotlib.colors import ListedColormap	Customizes plot colors for visualizations.
import warnings	Handles warnings to suppress unnecessary outputs.

warnings.filterwarnings('ignore')	Ignores warnings during model training and evaluation to reduce clutter.
from sklearn.cluster import DBSCAN	Imports DBSCAN, a clustering algorithm used for outlier detection.
from sklearn.ensemble import IsolationForest	Imports IsolationForest, an ensemble method for detecting outliers.
from sklearn.neighbors import LocalOutlierFactor	Detects outliers by evaluating the local density of data points.
from scipy import stats	Provides statistical functions (e.g., Z-scores, p-values) for outlier detection and hypothesis testing.
from sklearn.metrics import silhouette_score	Measures the quality of clustering by evaluating similarity within clusters.
from imblearn.over_sampling import RandomOverSampler	Performs random oversampling to balance class distribution by duplicating minority class instances.
from imblearn.under_sampling import RandomUnderSampler	Performs random undersampling to balance class distribution by reducing majority class instances.
from imblearn.over_sampling import SMOTE	Uses SMOTE to generate synthetic examples for the minority class to address class imbalance.
from sklearn.metrics import precision_score, recall_score, f1_score	Imports performance metrics to evaluate the model: Precision, Recall, and F1 Score.
from sklearn.preprocessing import PowerTransformer	Applies power transformations to make data more Gaussian-like, improving model performance.
from sklearn.model_selection import GridSearchCV	Performs hyperparameter tuning through cross-validation to select the best model parameters.

from sklearn.ensemble import BaggingClassifier, RandomForestClassifier	Imports ensemble methods (Bagging and Random Forest) to combine multiple models for improved performance.
from sklearn.svm import SVC	Re-imports SVC for use as a base model for classification.
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report	Imports additional evaluation metrics to measure overall model performance.
from xgboost import XGBClassifier, plot_importance	Imports XGBoost for model training and visualization of feature importance.
sns.set()	Configures default Seaborn aesthetics for improved plot styling.
plt.style.use('ggplot')	Sets the Matplotlib style to ggplot for a predefined plot aesthetic.
from sklearn.cluster import KMeans	Implements the K-Means clustering algorithm for grouping data into clusters based on similarity.
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster	Provides tools for hierarchical clustering, dendrogram visualization, and flat clustering.
from sklearn.mixture import GaussianMixture	Fits a Gaussian Mixture Model (GMM) to the data, used for clustering or density estimation.
from sklearn.neighbors import NearestNeighbors	Finds nearest neighbors, often used for clustering or recommendation systems.
from xgboost import plot_importance	Visualizes feature importance scores from an XGBoost model for interpretability.

Link for Notebook:

https://colab.research.google.com/drive/1k4Vg5l9wrUMnZPppqftiAR38sdnhGIep?usp=sharing

1. User Defined Functions For Project

1.1 Data Loading and Inspection:

- **load_csv**: Loads CSV files into pandas DataFrames with error handling for issues like file not found or empty data.
- **inspect_data**: Provides an overview of the dataset by displaying the first few rows, data types, missing values, and summary statistics. This helps assess the dataset's quality and structure before detailed analysis.

1.2 Labeling and Classifying Data:

- **label_dataset**: Adds a label column to the dataset, categorizing data into different classes (e.g., rice or cotton). This is essential for supervised machine learning tasks like classification.

1.3 EDA

- **analyze_numerical_feature**: Performs univariate analysis on numerical features, providing summary statistics, skewness, and visualizations (histograms, KDE plots, and boxplots) to understand data distributions and outliers.
- **analyze_categorical_feature**: Analyzes categorical features, showing value counts and missing values along with visualizations (bar and pie charts) to understand the frequency distribution of categories.

- **bivariate_analysis_numerical_numerical**: Analyzes the relationship between two numerical features using scatter plots and regression lines, along with calculating the correlation coefficient to identify linear or non-linear relationships.

- **bivariate_analysis_numerical_categorical**: Investigates the relationship between a numerical feature and a categorical feature, visualizing data through bar plots, box plots, and KDE plots to explore class differences.

1.4 Time Series Analysis:

- **seasonal_time_series_rice:** Plots seasonal NDVI time series for rice across three years (2021, 2022, 2023), helping to visualize trends and variations in crop health over time.
- **seasonal_time_series_cotton**: Similar to the rice function, but for cotton, plotting seasonal NDVI trends across multiple years to assess crop growth and seasonal health.

1.5 Missing Values Checking Function

- **Function**: check_missing_values(dataframe)
- **Purpose**: Identify columns in a DataFrame with missing values.
- **Returns**: List of columns with missing values or a message indicating none were found.

1.6 Duplicated Values Checking Function

- **Function**: check_duplicates(dataframe, ignore_column='Class')
- **Purpose**: Check for duplicate rows in a DataFrame, while allowing one column to be ignored.
- **Returns**: DataFrame of duplicate rows, or a message if no duplicates are found.

1.7 Percentage of Duplicates

- **Function**: calculate_duplicate_percentage(dataframe, ignore_column='Class')
- **Purpose**: Calculate the percentage of duplicate rows in a DataFrame, ignoring a specific column.
- **Returns**: Percentage of duplicate rows.

1.8 Outlier Detection

- **Methods**:
    - **IQR**: Detect outliers using the Interquartile Range method.
    - **Z-Score**: Detect outliers using the Z-score method.
    - **DBSCAN**: Use DBSCAN to identify outliers.
    - **Isolation Forest**: Use Isolation Forest for outlier detection.
    - **LOF**: Local Outlier Factor method for detecting outliers.

- **Returns**: DataFrame containing the detected outliers, along with the percentage.

1.9 Data Balancing Functions

- **Random Oversampling**: Balances the dataset by duplicating samples in the minority class.
- **Random Undersampling**: Reduces the size of the majority class.
- **SMOTE**: Synthetic Minority Over-sampling Technique, generates synthetic data for the minority class.

1.10 Feature Scaling

- **Function**: scaling_features(data)
- **Purpose**: Applies Power Transformation for skewness correction and Standardization (Z-score normalization).
- **Returns**: Scaled dataset.

1.11 Visualization Functions

- **Function**: scatter_plot(df, target_column)
- **Purpose**: Create a scatter plot of the first two features, color-coded by the target variable.

1.12 Supervised Learning Algorithm Functions

- **XGBoost**
  - **Grid Search**: Hyperparameter tuning with cross-validation.
  - **Model Training**: Train, evaluate, and visualize the performance (accuracy, precision, recall, F1-score).
  - **Returns**: Trained model and performance metrics.

- **Bagging**
  - **Grid Search**: Hyperparameter tuning using Bagging (with DecisionTreeClassifier as the base estimator).
  - **Model Training**: Train, evaluate, and generate performance metrics.
- **Random Forest**
  - **Grid Search**: Hyperparameter optimization using Random Forest.
  - **Model Training**: Train, evaluate, and generate feature importance visualization.
- **SVM**

- **Grid Search**: Hyperparameter tuning for SVM models.
- **Model Training**: Train, evaluate, and generate performance metrics (classification report, accuracy).

1.13 Unsupervised Learning Algorithm Functions

1. Dimensionality Reduction with PCA

Function: apply_pca

- **Purpose:**
  - Reduce the dimensionality of a dataset while retaining as much variance as possible.
- **Input Parameters:**
  - data (pd.DataFrame): Input dataset (features).
  - n_components (int): Number of principal components to keep (default is 2).
- **Processing Steps:**
  - Checks if the input is a Pandas DataFrame; raises an error otherwise.
  - Applies Principal Component Analysis (PCA) to transform the data.
  - Constructs a new DataFrame containing the principal components with descriptive column names.
- **Returns:**
  - A transformed dataset as a Pandas DataFrame with n_components columns representing the principal components.

Optimal k Determination (Elbow Method)

- **Function:** find_optimal_k
- **Objective:** Determines the optimal number of clusters for K-Means clustering using the Elbow Method.
- **Process:**
  - Computes inertia (sum of squared distances) for different cluster counts (k).
  - Plots inertia values against the number of clusters (k) to visualize the "elbow point".
- **Visualization:** Generates a plot to assist in selecting the optimal k.
- **Returns:** None (visual output only).

K-Means Clustering

- **Function:** kmeans_clustering
- **Objective:** Performs K-Means clustering on the input data.

- **Process:**
  - Trains a K-Means model with the specified number of clusters (n_clusters).
  - Predicts cluster labels for the input data.
- **Returns:**
  - Trained K-Means model.
  - Cluster labels for each data point.

Hierarchical Clustering

Function: hierarchical_clustering

- **Purpose:**
  - Perform hierarchical clustering to group data points into clusters based on a chosen linkage method.
- **Input Parameters:**
  - data (np.ndarray): Dataset for clustering.
  - n_clusters (int): Desired number of clusters.
  - method (string): Linkage method to compute the hierarchical clustering (default is "ward").
- **Processing Steps:**
  - Computes a linkage matrix using the specified method.
  - Assigns cluster labels based on the maximum number of clusters specified.
- **Returns:**
  - An array of cluster labels for each data point.

Dendrogram Visualization for Hierarchical Clustering

Function: plot_dendrogram

- **Purpose:**
  - Generate and visualize dendrograms for hierarchical clustering using multiple linkage methods.
- **Input Parameters:**
  - data (np.ndarray): Dataset for dendrogram visualization.
  - method (str): Linkage method for hierarchical clustering.
  - title (str): Title for the dendrogram plot (default is 'Hierarchical Clustering Dendrogram').
  - max_depth (int): Maximum depth of the dendrogram to display.
- **Processing Steps:**
  - Computes linkage matrices using four linkage methods: single, complete, average, and ward.

    - Plots truncated dendrograms for each linkage method to provide an overview of clustering structures.
- **Returns:**
    - Displays the dendrogram plots for visual analysis of clustering.

DBSCAN Clustering

- **Find Optimal Epsilon**
    - **Description**: Determines the optimal value of epsilon (ϵ\epsilonϵ) for DBSCAN clustering using a k-distance graph.
    - **Parameters**:
        - X: Feature data for clustering.
        - k: Number of nearest neighbors to consider (default: 24).
    - **Outputs**: Visualizes the k-distance graph to aid in choosing ϵ\epsilonϵ.
    - **Returns**: None (visualization-only function).
- **DBSCAN Clustering Function**
    - **Description**: Implements DBSCAN clustering for a given dataset.
    - **Parameters**:
        - data: Input dataset.
        - eps: Maximum distance for samples to be considered neighbors.
        - min_samples: Minimum number of points required to form a dense region.
    - **Outputs**: Cluster labels for each data point.
    - **Returns**: Cluster labels as a NumPy array.
- **DBSCAN Experimentation**
    - **Description**: Performs experiments with DBSCAN clustering for various ϵ\epsilonϵ and min_samples values. Visualizes clustering results and computes evaluation metrics such as silhouette score.
    - **Parameters**:
        - X: Standardized dataset.
        - eps_range: List of ϵ\epsilonϵ values to test.
        - min_samples_range: List of min_samples values to test.
        - pca_df: DataFrame containing PCA-transformed data for visualization.
    - **Outputs**:
        - Visualization of clusters for each parameter combination.
        - Prints number of clusters, noise points, and silhouette scores.
    - **Returns**: None.

Gaussian Mixture Model (GMM) Clustering

- **GMM Clustering Function**
    - **Description**: Applies clustering using Gaussian Mixture Models to partition data into clusters based on a probabilistic model.
    - **Parameters**:
        - data: Input dataset.

        - n_clusters: Number of mixture components (clusters).
    - **Outputs**:
        - Cluster labels for each data point.
        - Probabilities for each point belonging to each cluster.
        - Log-likelihood of the fitted model.
    - **Returns**: Cluster labels as a NumPy array.

Clustering Evaluation and Visualization Functions

Clustering Evaluation: evaluate_clustering

- **Purpose:** Evaluate clustering performance using multiple metrics and visualize the results.
- **Features:**
    - Computes evaluation metrics:
        - Confusion Matrix
        - Normalized Mutual Information (NMI)
        - Cluster Purity
        - Silhouette Score (optional, if data is provided).
    - Maps predicted cluster labels to true labels based on the majority class.
    - Visualizes the confusion matrix as a heatmap.
- **Parameters:**
    - labels: Predicted cluster labels (NumPy array).
    - true_labels: Ground truth labels (NumPy array).
    - data: (Optional) Original dataset for silhouette score calculation.
- **Returns:**
    - Dictionary containing evaluation metrics.

Cluster Visualization: plot_comparison

- **Purpose:** Compare the clustering results with true labels through visualization.
- **Features:**
    - Plots two scatter plots side by side:
        - True labels.
        - Predicted cluster labels.
    - Uses the first two dimensions of the data for visualization.
- **Parameters:**
    - data: Dataset for visualization (NumPy array or DataFrame).
    - true_labels: Ground truth class labels.
    - predicted_labels: Predicted cluster labels.

Cluster Visualization with Centroids: visualize_clusters

- **Purpose:** Visualize clustering results with optional centroids.

- **Features:**
  - Plots data points with cluster labels and centroids (if provided).
  - Customizable title and axis labels.
- **Parameters:**
  - data: Dataset for visualization (NumPy array or DataFrame).
  - labels: Cluster labels.
  - title: Plot title.
  - centroids: (Optional) Centroid coordinates for the clusters.

Feature and Cluster Analysis Functions

Feature Mapping: map_features_to_indices

- **Purpose:** Map feature names to their corresponding indices for clustering.
- **Features:**
  - Converts feature name pairs into index pairs.
  - Handles missing or invalid feature names gracefully.
- **Parameters:**
  - feature_names: List of feature names.
  - feature_pairs: List of feature name pairs.
- **Returns:**
  - List of index pairs corresponding to the feature name pairs.

Cluster Visualization with Features: plot_clusters_with_features

- **Purpose:** Plot clusters using specified feature pairs with centroids.
- **Features:**
  - Supports multiple feature pairs for visualization.
  - Highlights cluster centroids.
  - Handles large datasets and customizable axis labels.
- **Parameters:**
  - X: Feature dataset (NumPy array or DataFrame).
  - pred: Cluster labels.
  - cluster_centers: Centroid coordinates for the clusters.
  - feature_pairs: List of feature index pairs to plot.
  - feature_names: Names of the features for labeling.

Cluster-Class Relationship Analysis

Cluster-Class Distribution: cluster_class_distribution

- **Purpose:** Analyze and visualize the relationship between clusters and true class labels.
- **Features:**
  - Creates bar charts showing the distribution of true labels in each cluster.
  - Provides a textual summary of the distribution for each cluster.
- **Parameters:**

    - cluster_labels: Predicted cluster labels (NumPy array).
    - true_labels: Ground truth class labels (NumPy array).
- **Visualization:**
    - Grouped bar charts for each cluster, showing the distribution of class labels.
    - Configurable plot aesthetics (e.g., labels, legends).

2. Dataset Overview

This section provides a detailed overview of the datasets used in the project. It explains the structure of the data, its contents, and an inspection of its features. The datasets include historical data for two crops—rice and cotton—from the years 2021, 2022, and 2023.

2.1 Introduction About the Project

Crop classification using satellite-derived data is an essential task in modern agriculture to monitor and assess the performance of crops over time. This project aims to classify rice and cotton crops based on their unique Normalized Difference Vegetation Index (NDVI) values collected monthly for three consecutive years (2021, 2022, and 2023). The classification task will help identify the crop type accurately and contribute to agricultural planning and decision-making.

2.2 Dataset Details

The dataset consists of NDVI data collected for two crop types—**rice** and **cotton**—spanning three years (2021, 2022, and 2023). The data is organized into six separate CSV files, divided by crop type and year. The following folders contain the data:

- **Rice Folder**:
  Contains NDVI data for rice crops across three years:
  - rice2021.csv
  - rice2022.csv
  - rice2023.csv
- **Cotton Folder**:
  Contains NDVI data for cotton crops across three years:
  - cotton2021.csv

- cotton2022.csv
- cotton2023.csv

Each dataset contains 12 monthly NDVI readings (NDVI01, NDVI02, ..., NDVI12) for individual crop instances in a given year. These features reflect the vegetation health and growth patterns of the crops.

2.3 Inspection of Datasets

To better understand the structure of the data, each dataset was inspected for its size, column names, and data types. The inspections reveal the following details:

2.3.1 Rice 2021

- **Rows**: 419
- **Columns**: 12 NDVI features (all of type float64)
- **Memory Usage**: 39.4 KB

2.3.2 Rice 2022

- **Rows**: 4,687
- **Columns**: 12 NDVI features (all of type float64)
- **Memory Usage**: 439.5 KB

2.3.3 Rice 2023

- **Rows**: 919
- **Columns**: 12 NDVI features (all of type float64)
- **Memory Usage**: 86.3 KB

2.3.4 Cotton 2021

- **Rows**: 2,883

- **Columns**: 12 NDVI features (all of type float64)
- **Memory Usage**: 270.4 KB

2.3.5 Cotton 2022

- **Rows**: 12,411
- **Columns**: 12 NDVI features (all of type float64)
- **Memory Usage**: 1.1 MB

2.3.6 Cotton 2023

- **Rows**: 11,777
- **Columns**: 12 NDVI features (all of type float64)
- **Memory Usage**: 1.1 MB

2.4 Observations from Inspection

The inspection highlights the following:

1. Each dataset is consistent in its structure, containing exactly 12 NDVI columns with no missing values.
2. There is variation in the number of records across different datasets due to the differences in crop fields monitored each year.
3. Memory usage varies significantly based on the number of records, with cotton datasets (especially from 2022 and 2023) being larger in size.

This inspection ensures the data is clean and suitable for further preprocessing steps, such as labeling and merging.

2.5 Summary

The datasets for rice and cotton crops provide comprehensive NDVI information across three years. This data will serve as the foundation for the classification task. The inspection confirmed the data's consistency, ensuring it is ready for the next step of preprocessing.

3. Data Labelling and Merging

This section focuses on preparing the dataset for analysis and modeling by assigning labels to distinguish between the two crops (rice and cotton) and merging the data across years. This step ensures that the data is organized and labeled appropriately for downstream machine learning and statistical tasks.

3.1 Labelling the Rice and Cotton Data

Labelling the data is an essential preprocessing step to distinguish between the two crop types in the dataset. A binary label was assigned to each dataset:

- **Rice Data**: Assigned a label of **1**, indicating the crop type is rice.
- **Cotton Data**: Assigned a label of **0**, indicating the crop type is cotton.

The datasets for rice (2021, 2022, and 2023) and cotton (2021, 2022, and 2023) were processed using a custom function called label_dataset(). This function appends a new column, Label, to each dataset. The added column serves as a categorical indicator for crop classification, making the datasets suitable for supervised learning tasks.

- **Rice Datasets**:
    - rice2021: Label = 1
    - rice2022: Label = 1
    - rice2023: Label = 1
- **Cotton Datasets**:
    - cotton2021: Label = 0
    - cotton2022: Label = 0
    - cotton2023: Label = 0

This consistent labeling ensures the datasets are properly organized and ready for merging.

3.2 Validating the Label Assignment

To verify that the labels were correctly assigned to each dataset, the inspect_data() function was used. This function examines the datasets to ensure that the labels are present and accurately reflect the crop type. This step ensures data integrity and helps avoid errors during model training and analysis.

Inspection outputs confirmed:

- The presence of the Label column in each dataset.
- Correct assignment of labels (1 for rice and 0 for cotton).
- Consistency in data structure and column count after labelling.

3.3 Merging the Year-Wise Data

After labelling, the datasets for rice and cotton were merged year-wise. This step consolidates the data from both crops into a single dataset for each year. The merging process was performed using the pd.concat() function, which combines the rice and cotton datasets for a given year along the row axis. This creates three combined datasets:

- **Year 1 (2021)**: Combined data from rice2021 and cotton2021.
- **Year 2 (2022)**: Combined data from rice2022 and cotton2022.
- **Year 3 (2023)**: Combined data from rice2023 and cotton2023.

The resulting datasets provide a unified structure, facilitating year-wise analysis and comparison. Each dataset retains the Label column, which allows the differentiation of crop types during analysis.

3.4 Storing the Year-Wise Data

To ensure reproducibility and efficient data handling, the merged datasets for each year were saved as CSV files. These files were stored in the project directory for future use in modeling and analysis. The file paths for the saved datasets are as follows:

- **Year 1 (2021)**: /content/drive/MyDrive/CropClassificationProject/1_data/merged_data/year1.csv
- **Year 2 (2022)**: /content/drive/MyDrive/CropClassificationProject/1_data/merged_data/year2.csv
- **Year 3 (2023)**: /content/drive/MyDrive/CropClassificationProject/1_data/merged_data/year3.csv

Saving the data ensures that the preprocessed datasets are readily available for downstream tasks without requiring repetition of the merging and labelling steps.

3.5 Shape of the Data Before and After Merging

The following table summarizes the shape (number of rows and columns) of the datasets before and after merging:

Dataset	Rows	Columns
Rice 2021	419	13
Cotton 2021	2,883	13
Year 1 (2021)	3,302	13
Rice 2022	4,687	13
Cotton 2022	12,411	13
Year 2 (2022)	17,098	13
Rice 2023	919	13
Cotton 2023	11,777	13

Year 3 (2023)	12,696	13

The merged datasets reflect the consolidation of rice and cotton data for each year, with no loss of information during the merging process.

3.6 Summary

The data labelling and merging process prepared the dataset for analysis by:

1. Assigning meaningful labels for crop classification.
2. Validating the integrity of the labelled datasets.
3. Merging rice and cotton data year-wise to create unified datasets.
4. Saving the merged datasets for streamlined use in subsequent steps.

This comprehensive preprocessing ensures the datasets are ready for exploratory data analysis, feature engineering, and machine learning.

4. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a critical step in understanding the structure, distribution, and relationships within a dataset. It helps in identifying potential issues, uncovering patterns, and informing decisions for data preprocessing and modeling. This section presents the EDA conducted for three years of combined rice and cotton data.

4.1 Exploratory Data Analysis for Year 1

The Year 1 dataset contains combined NDVI values for rice and cotton crops from 2021. This dataset is analyzed using two main approaches: **univariate analysis** and **bivariate analysis**.

4.1.1 Univariate Analysis

Univariate analysis examines individual features independently to understand their distribution, central tendencies, and potential issues such as outliers or skewness.

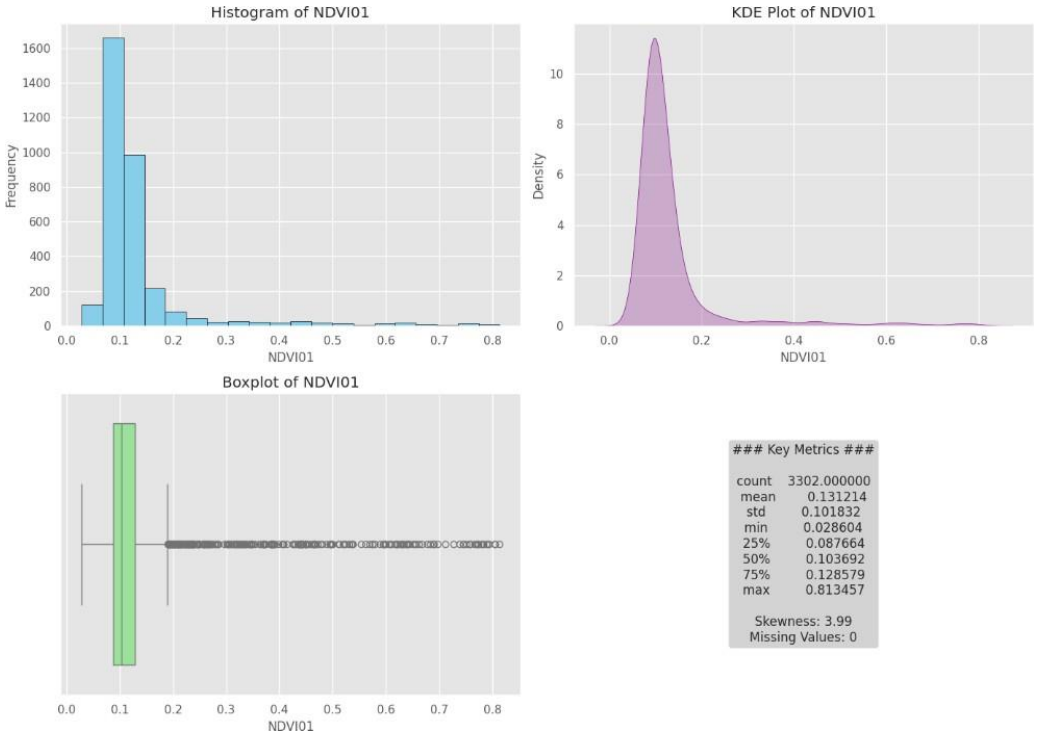
4.1.1.1 Steps of Univariate Analysis

1. **Descriptive Statistics**:
    - Summary statistics, including mean, median, mode, standard deviation, range, and quartiles, are computed for each feature.
    - These statistics provide insights into the overall distribution and variability of the data.
2. **Visualizations**:
    - Histograms, box plots, and density plots are used to visually explore the distribution of numerical features. These help identify outliers, skewness, and the spread of the data.
3. **Identifying Outliers**:
    - Outliers are detected using visualizations and descriptive statistics. The source of outliers (e.g., measurement errors or natural variation) is analyzed to decide whether to include or exclude them.
4. **Skewness**:
    - Skewness is checked to determine whether the data deviates from a normal distribution. If necessary, transformations (e.g., logarithmic or square root) are considered to address skewness.
5. **Conclusions**:
    - The findings from the univariate analysis are summarized, highlighting any potential issues that require attention before proceeding with modeling.

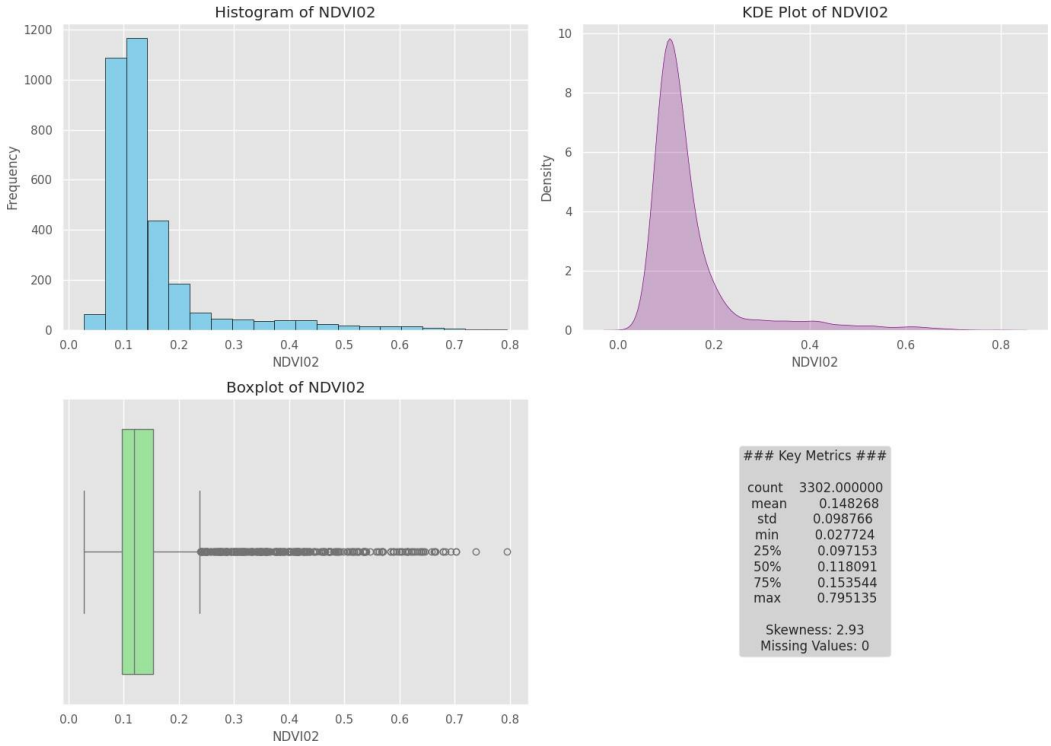
4.1.1.2 Numerical Variables

Each numerical feature (NDVI01 to NDVI12) is analyzed independently using descriptive statistics and visualizations.

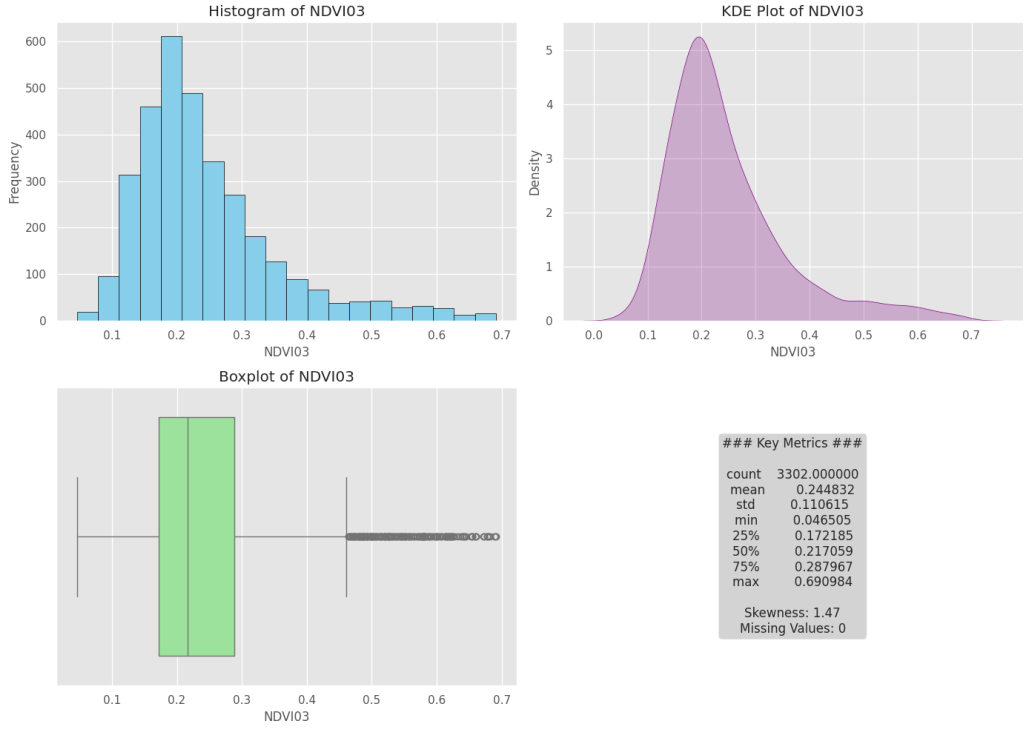
1. NDVI01



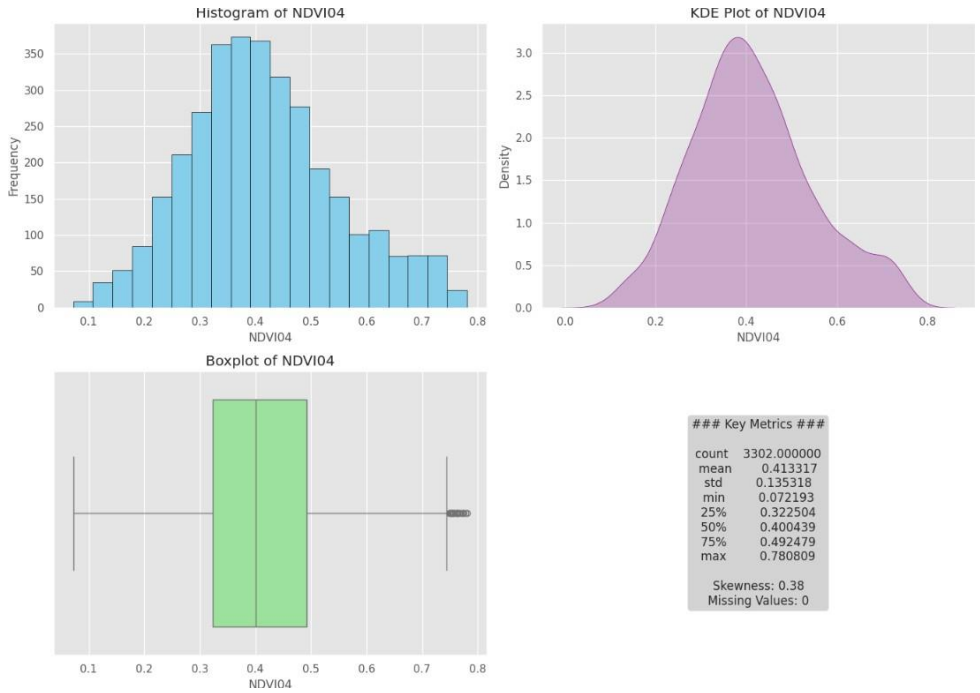
2. NDVI02



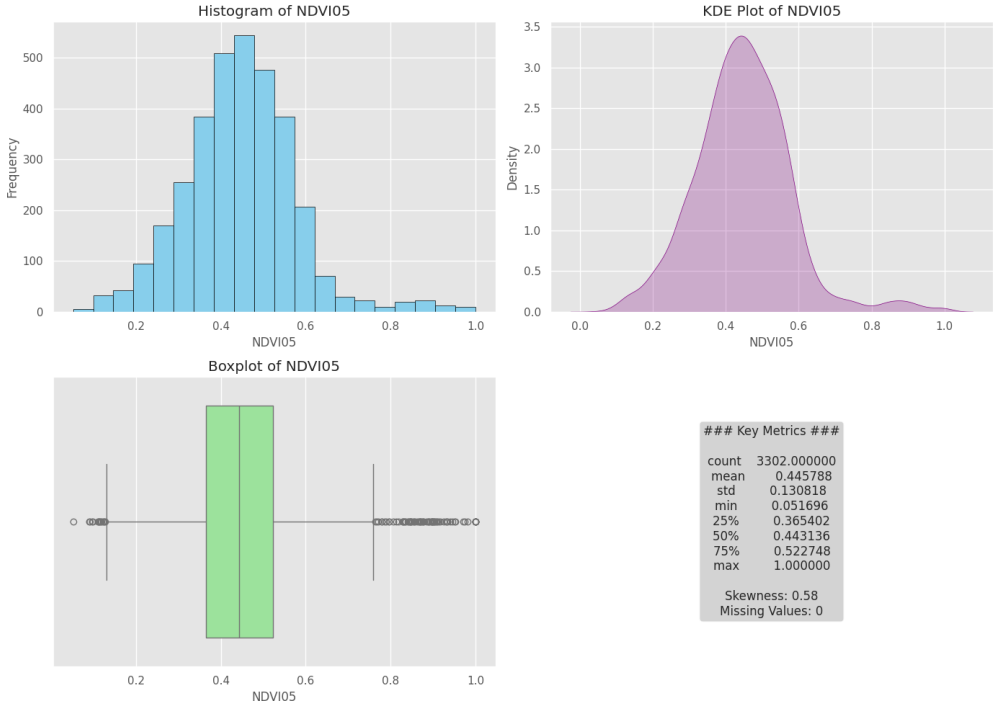
3. NDVI03



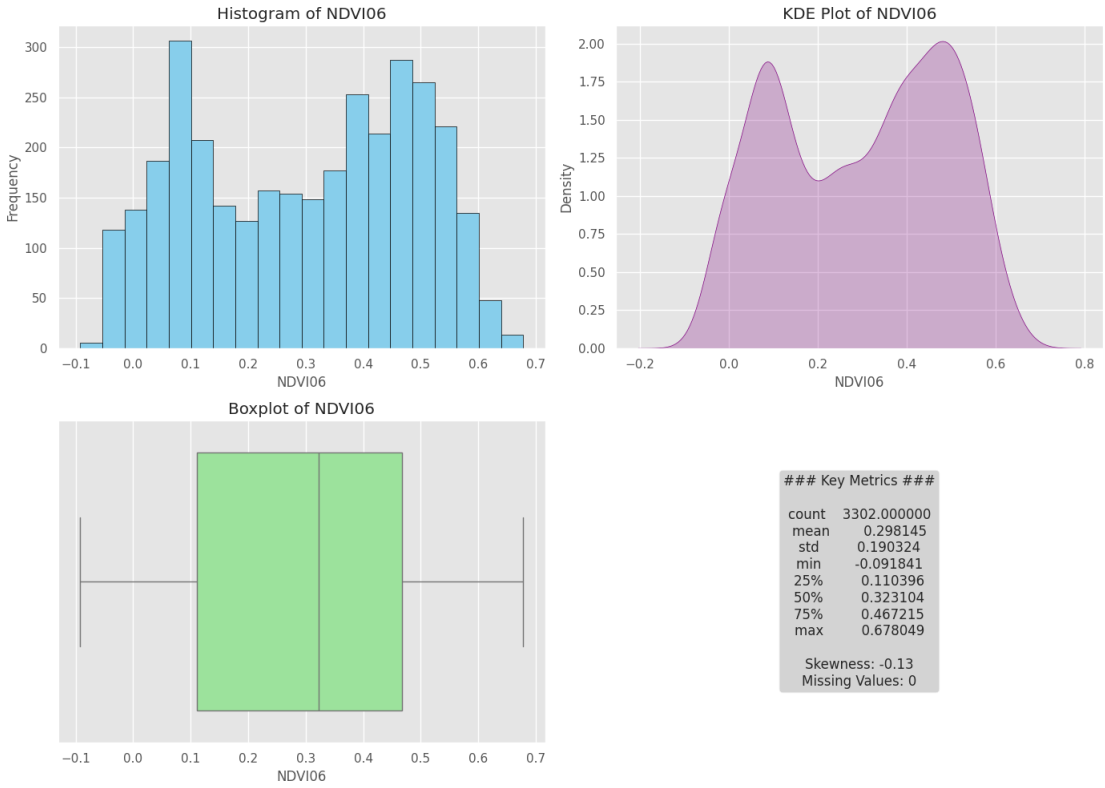
4. NDVI04



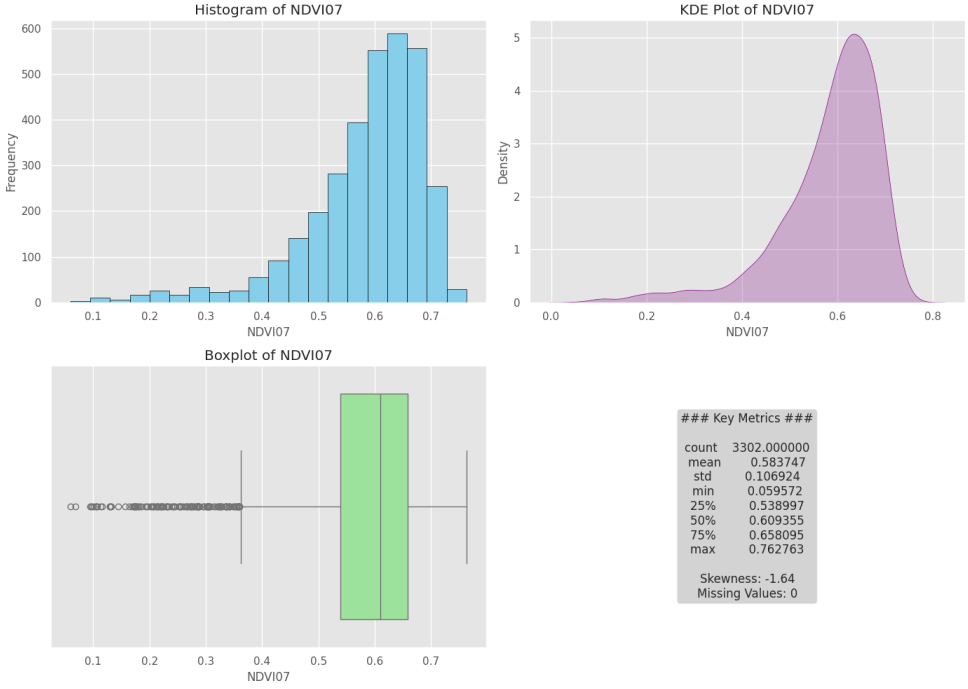
5. NDVI05



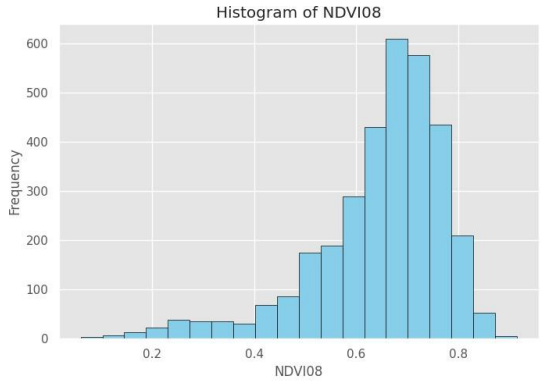
6. NDVI06

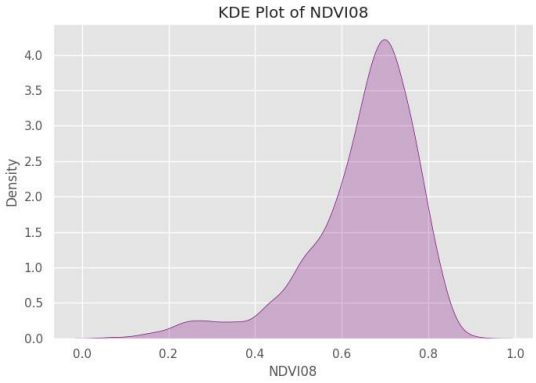


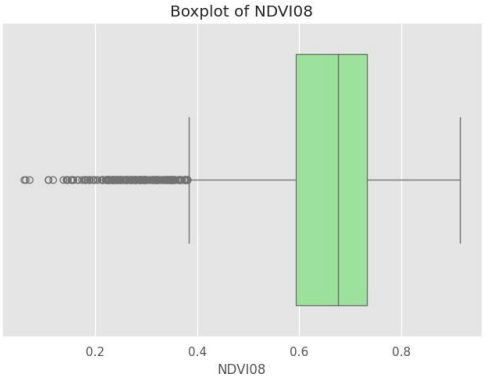
7. NDVI07



8. NDVI08





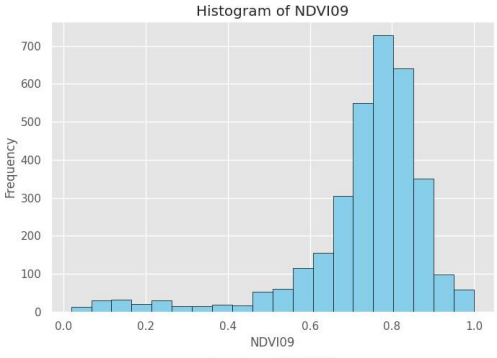


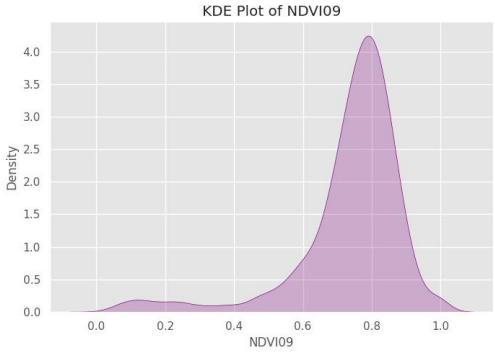
Key Metrics

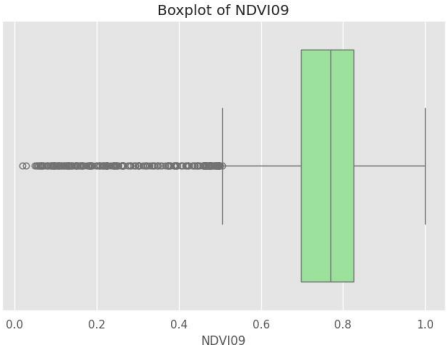
count	3302.000000
mean	0.647445
std	0.128884
min	0.060626
25%	0.592696
50%	0.675556
75%	0.732479
max	0.914129

Skewness: -1.38
Missing Values: 0

9. NDVI09





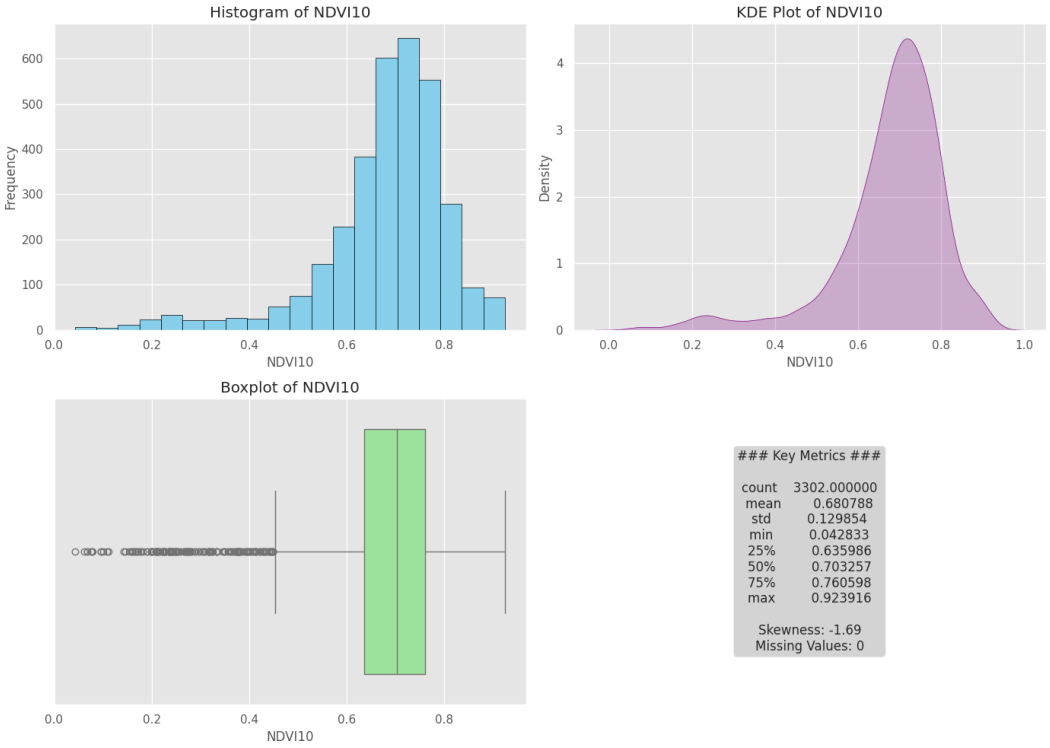


Key Metrics

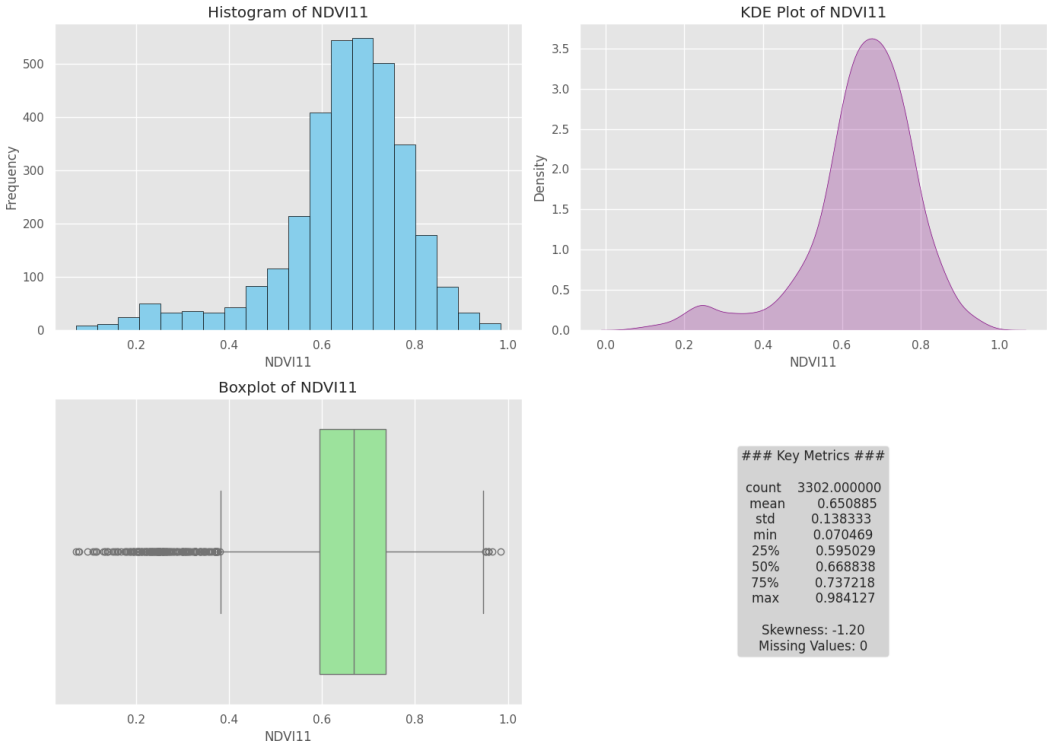
count	3302.000000
mean	0.735249
std	0.158848
min	0.017911
25%	0.697656
50%	0.770038
75%	0.825826
max	0.999485

Skewness: -2.08
Missing Values: 0

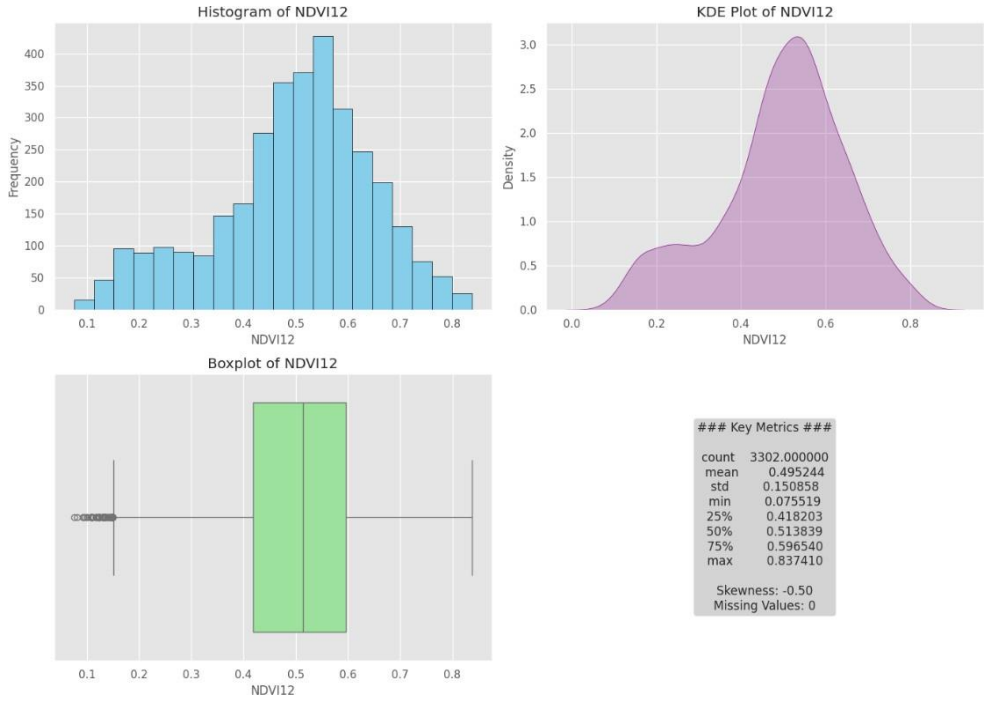
10. NDVI10


Histogram of NDVI10
Frequency
NDVI10
KDE Plot of NDVI10
Density
NDVI10
Boxplot of NDVI10
NDVI10
Key Metrics
count 3302.000000
mean 0.680788
std 0.129854
min 0.042833
25% 0.635986
50% 0.703257
75% 0.760598
max 0.923916
Skewness: -1.69
Missing Values: 0

11. NDVI11

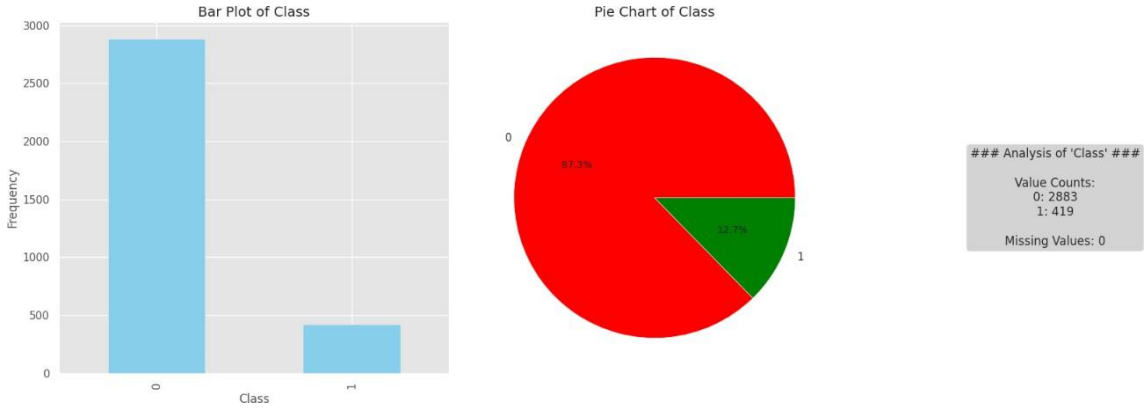

Histogram of NDVI11
Frequency
NDVI11
KDE Plot of NDVI11
Density
NDVI11
Boxplot of NDVI11
NDVI11
Key Metrics
count 3302.000000
mean 0.650885
std 0.138333
min 0.070469
25% 0.595029
50% 0.668838
75% 0.737218
max 0.984127
Skewness: -1.20
Missing Values: 0

12. NDVI12



4.1.1.3 Categorical Variables

The categorical variable Class (indicating crop type: rice or cotton) is analyzed to understand the distribution of data points between the two crop types.



4.1.2 Bivariate Analysis

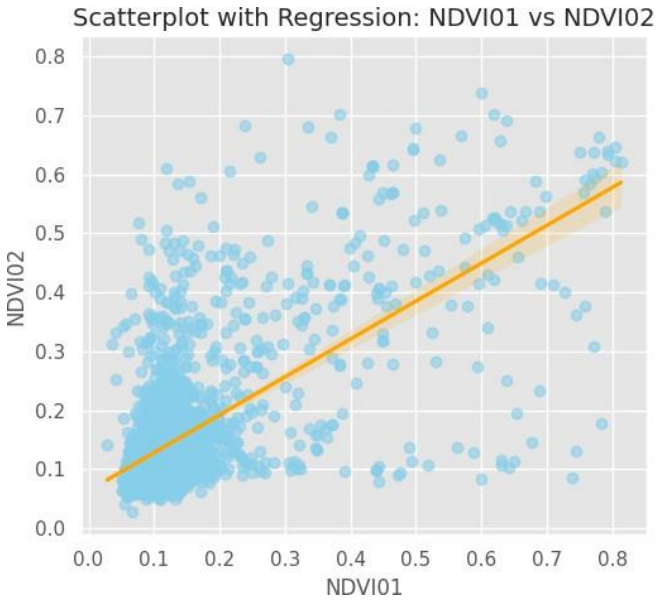
Bivariate analysis investigates relationships between pairs of features to identify potential correlations, trends, and interactions.

4.1.2.1 Steps of Bivariate Analysis

1. **Numerical-Numerical Relationships**:
    - Scatterplots, regression plots, 2D histograms, and 2D KDE (Kernel Density Estimation) plots are used to visualize relationships between two numerical features.
    - Correlation coefficients (e.g., Pearson) are computed to assess the strength and direction of linear relationships.
2. **Numerical-Categorical Relationships**:
    - Bar plots, box plots, violin plots, and KDE plots are created to compare numerical features across different categories of the Class variable (rice vs. cotton).
3. **Categorical-Categorical Relationships**:
    - Cross-tabulations or contingency tables are used to study the interaction between two categorical features (though no such features are present in this dataset).

4.1.2.2 Numerical-Numerical Analysis

The relationships between NDVI01 and all other NDVI features are analyzed iteratively. Similar analyses are conducted for NDVI02, NDVI03, and so on, to explore pairwise correlations between the NDVI features.

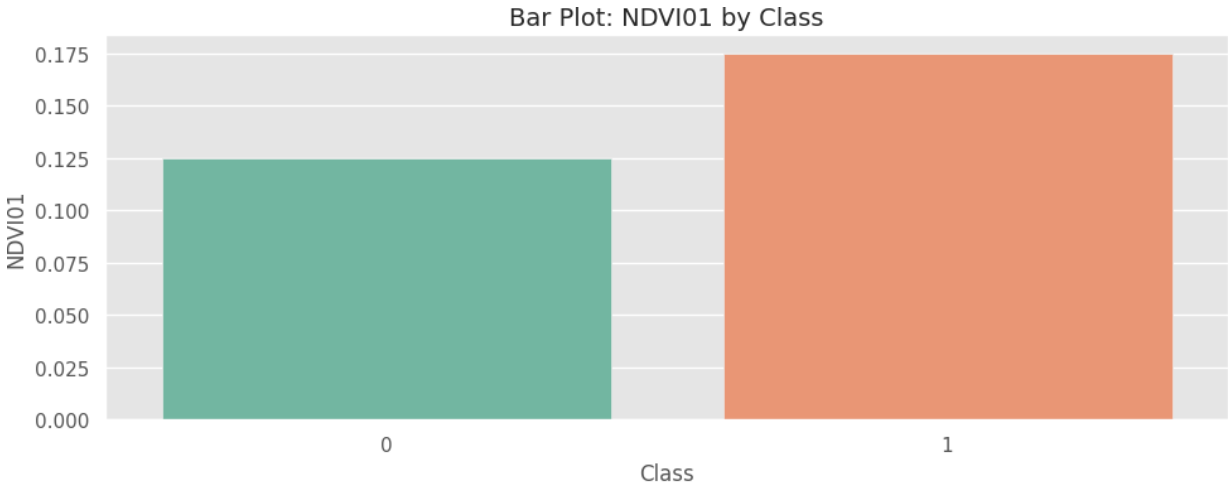


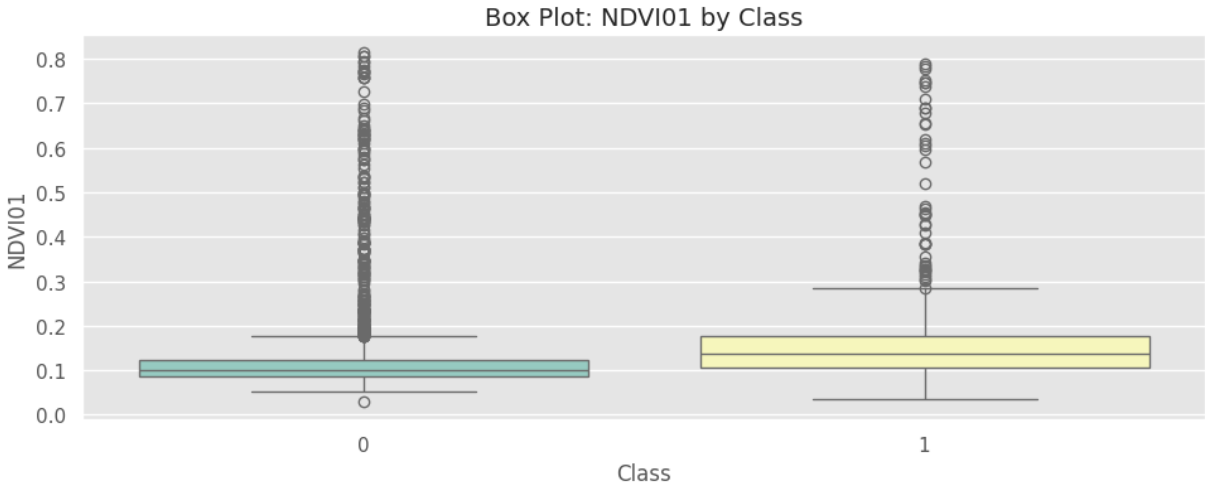
Correlation Analysis

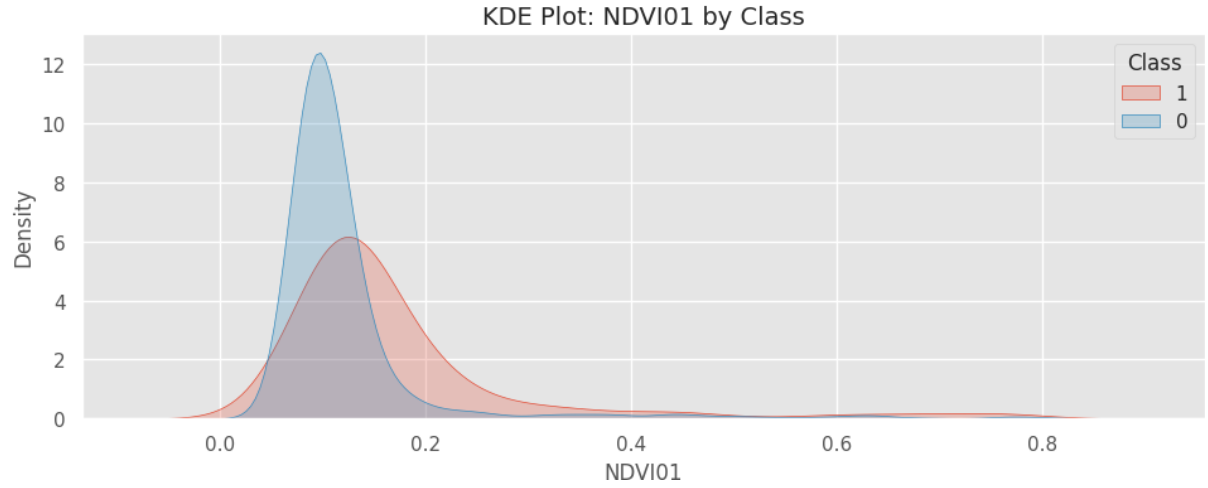
Correlation Coefficient: 0.66

4.1.2.3 Numerical-Categorical Analysis

Each numerical feature (NDVI01 to NDVI12) is analyzed against the Class variable to examine how NDVI values differ between rice and cotton crops.







4.1.2.4 Categorical-Categorical Analysis

Since the dataset does not contain more than one categorical feature, this type of analysis is not applicable.

4.2 Exploratory Data Analysis for Year 2

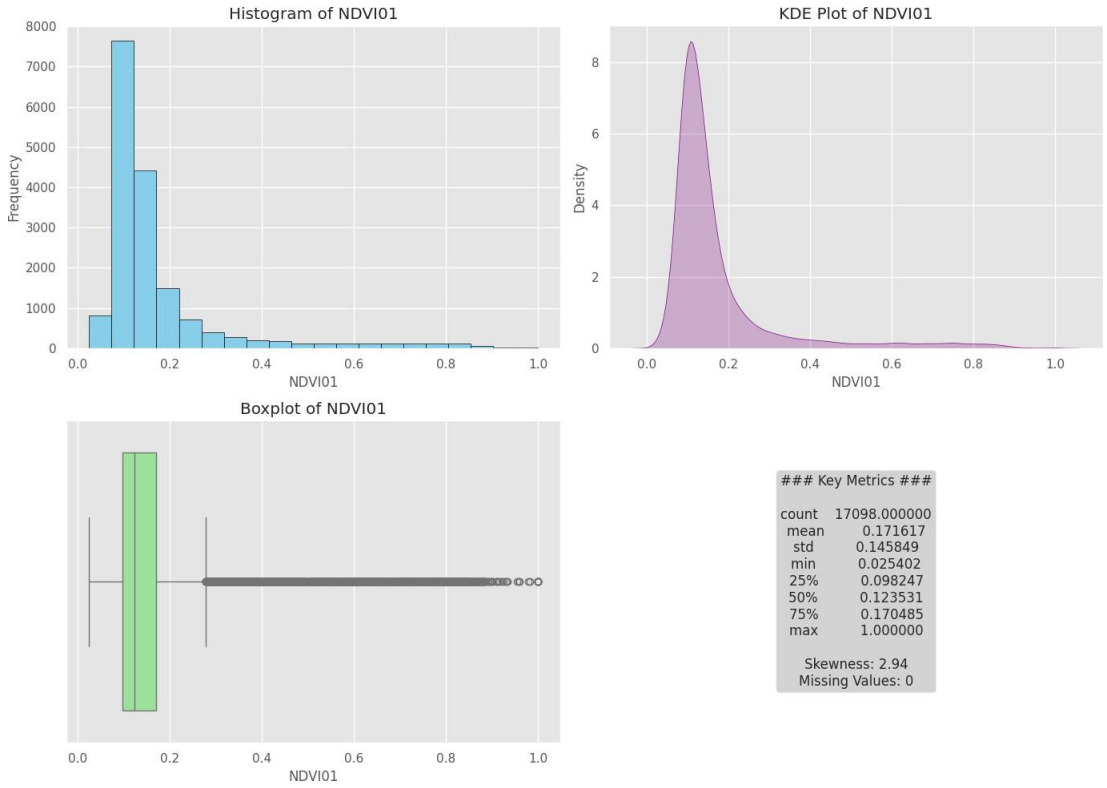
The Year 2 dataset, combining rice and cotton NDVI values for 2022, is analyzed using the same approach as Year 1.

4.2.1 Univariate Analysis

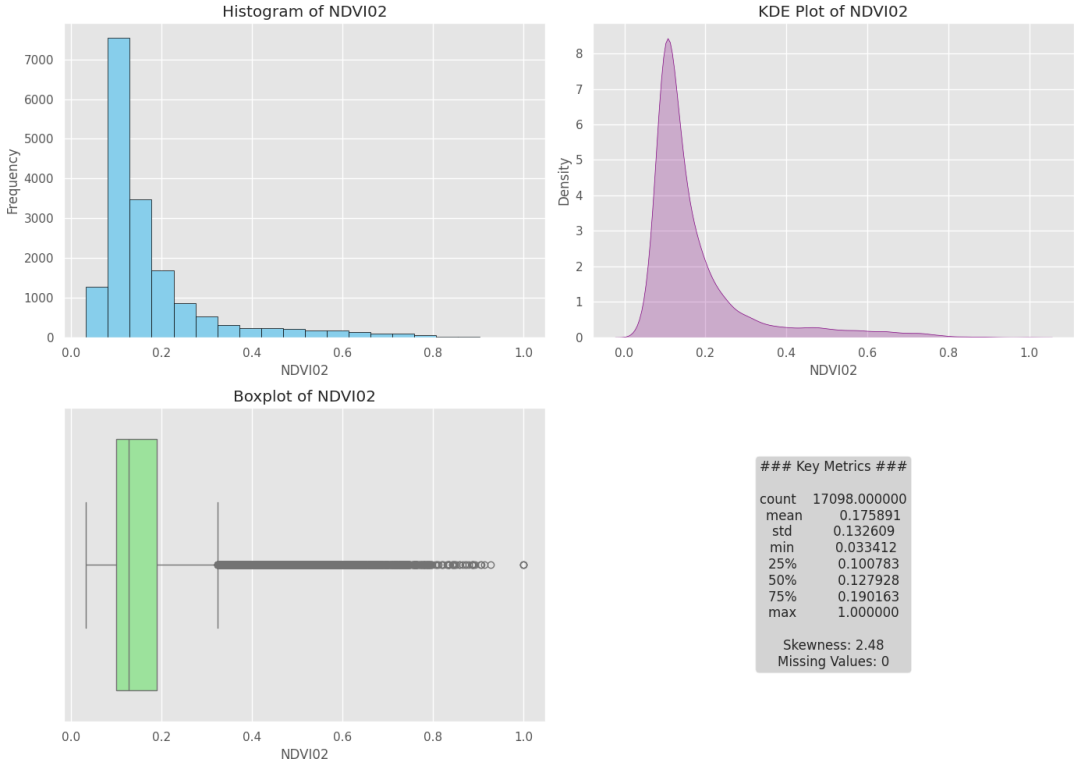
1. **Descriptive Statistics and Visualizations**:
    - The distribution of each NDVI feature (NDVI01 to NDVI12) is examined through histograms, box plots, and density plots.
    - Descriptive statistics are computed to summarize central tendencies and variability.
2. **Identifying Outliers and Skewness**:
    - Outliers and skewness are identified, and potential transformations or data cleaning methods are considered.
3. **Categorical Variable Analysis**:
    - The Class variable is analyzed to ensure an even distribution of data points between rice and cotton crops.

4.2.1.1 Numerical Features

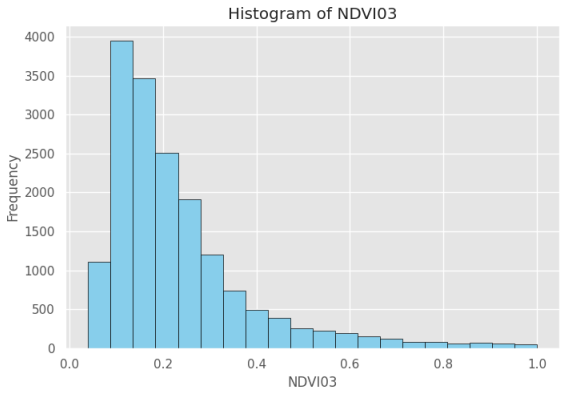
1. **NDVI01**

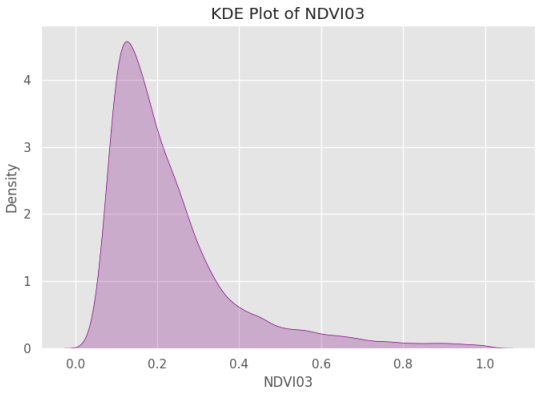


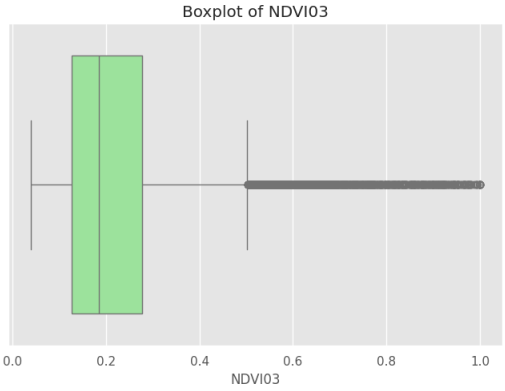
2. NDVI02



3. NDVI03





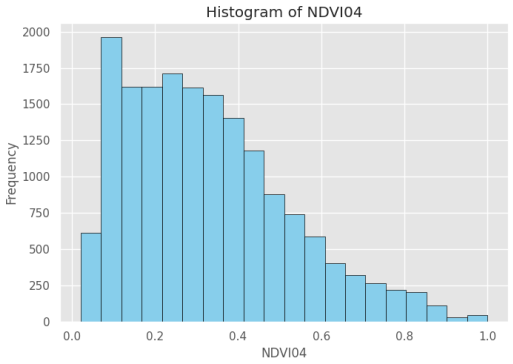


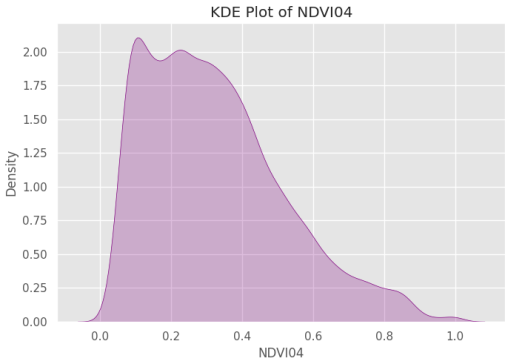
Key Metrics

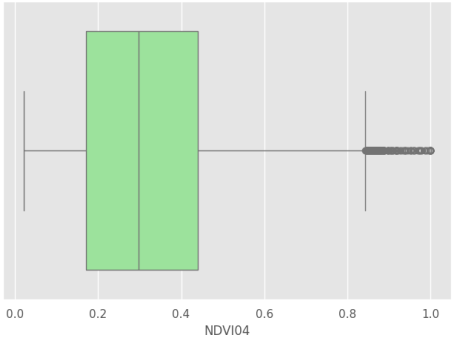
count	17098.000000
mean	0.230918
std	0.157491
min	0.039514
25%	0.125735
50%	0.184260
75%	0.276473
max	1.000000

Skewness: 2.03
Missing Values: 0

4. NDVI04





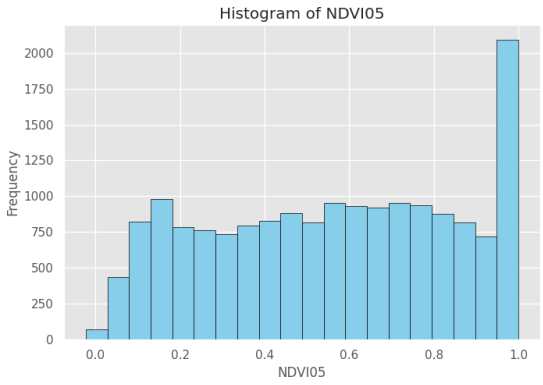


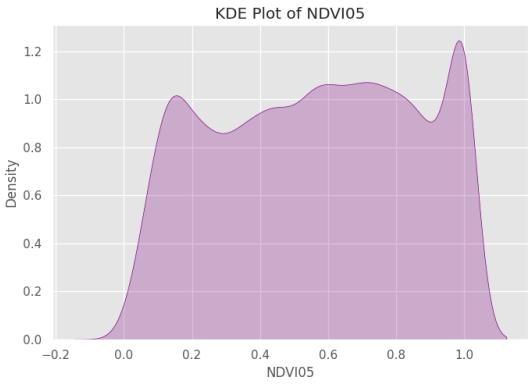
Key Metrics

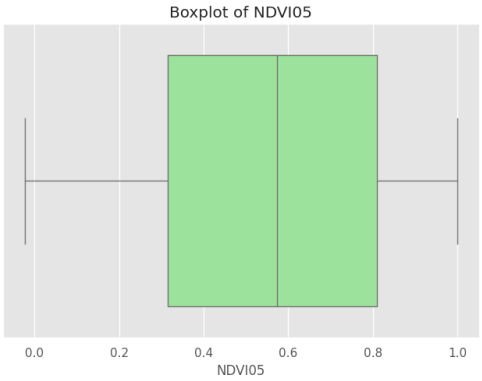
count	17098.000000
mean	0.325971
std	0.194446
min	0.021010
25%	0.170619
50%	0.297309
75%	0.439397
max	1.000000

Skewness: 0.78
Missing Values: 0

5. NDVI05



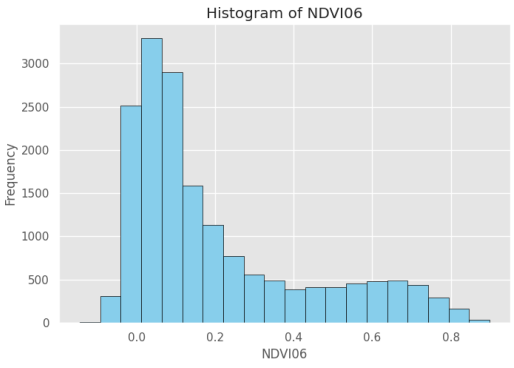


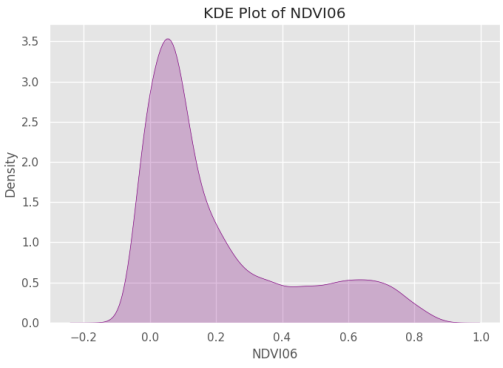


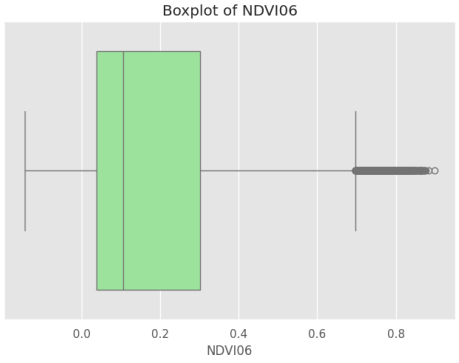
### Key Metrics ###	
count	17098.000000
mean	0.561830
std	0.289879
min	-0.022998
25%	0.314445
50%	0.573208
75%	0.809642
max	1.000000

Skewness: -0.08
Missing Values: 0

6. NDVI06



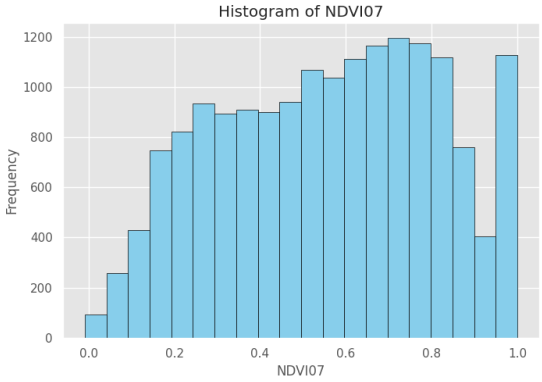


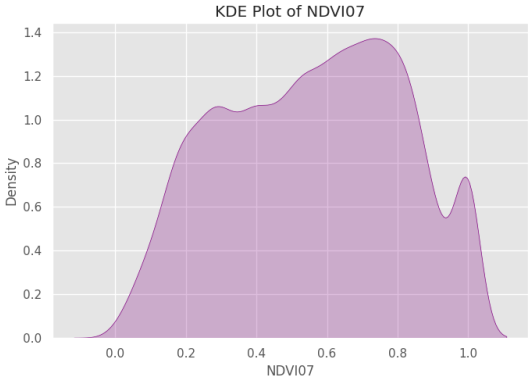


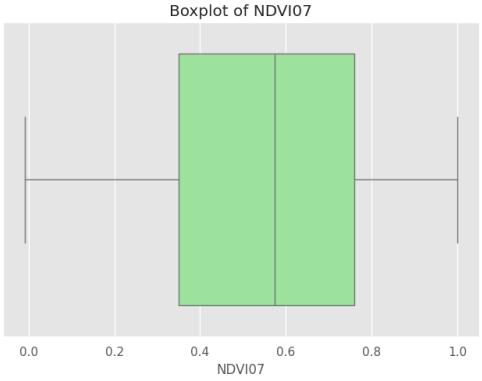
### Key Metrics ###	
count	17098.000000
mean	0.200044
std	0.228619
min	-0.144406
25%	0.037813
50%	0.105478
75%	0.301481
max	0.898720

Skewness: 1.18
Missing Values: 0

7. NDVI07



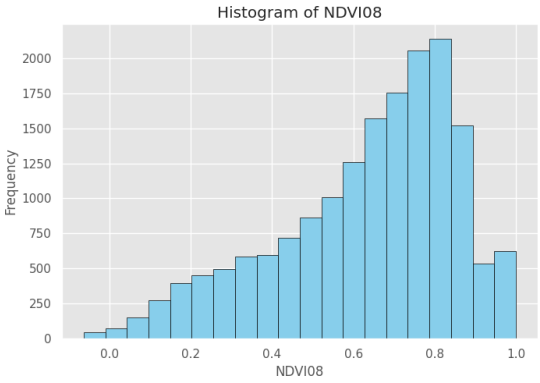


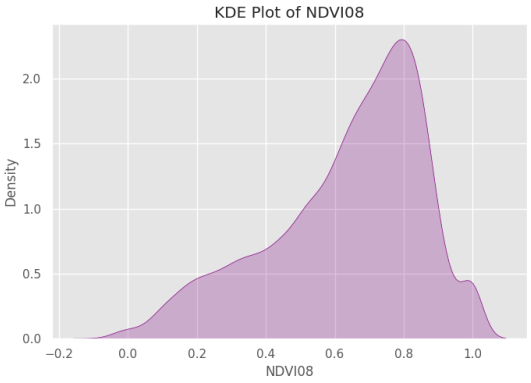


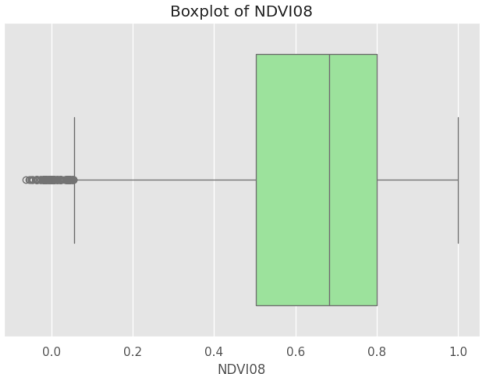
### Key Metrics ###	
count	17098.000000
mean	0.558217
std	0.253309
min	-0.008705
25%	0.349338
50%	0.573882
75%	0.760229
max	1.000000

Skewness: -0.09
Missing Values: 0

8. NDVI08



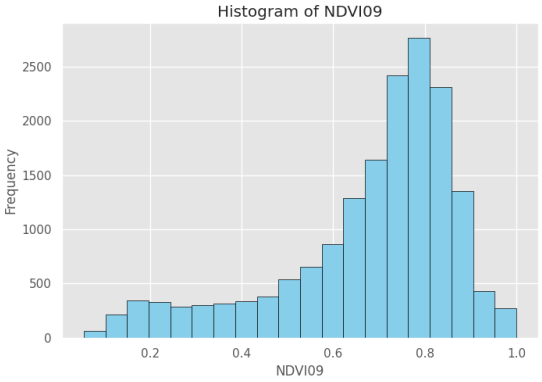


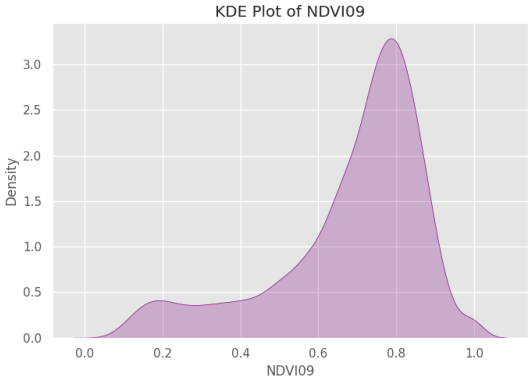


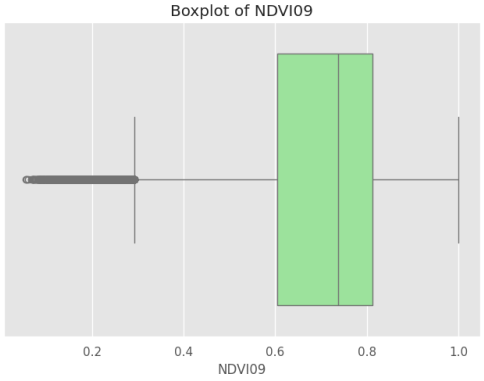
### Key Metrics ###	
count	17098.000000
mean	0.633866
std	0.221482
min	-0.063648
25%	0.502542
50%	0.683506
75%	0.800540
max	1.000000

Skewness: -0.74
Missing Values: 0

9. NDVI09



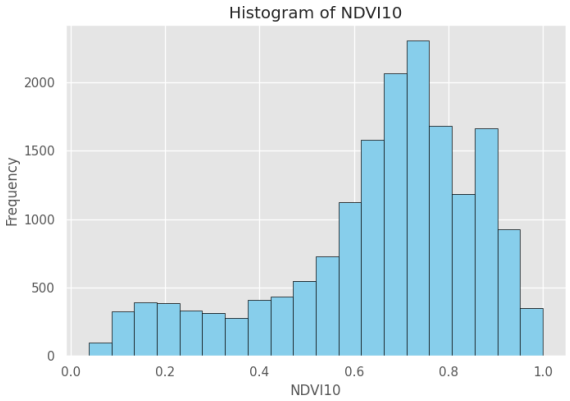


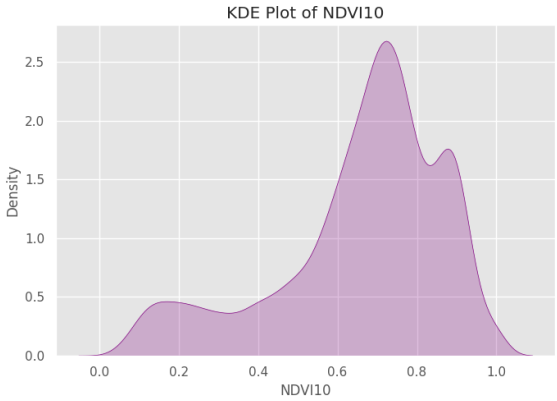


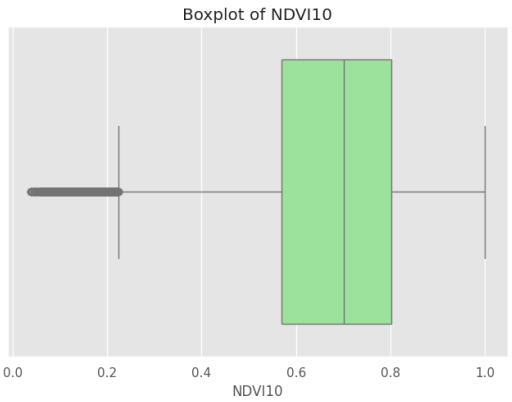
### Key Metrics ###	
count	17098.000000
mean	0.680701
std	0.194294
min	0.055522
25%	0.604905
50%	0.738385
75%	0.812978
max	1.000000

Skewness: -1.19
Missing Values: 0

10. NDVI10



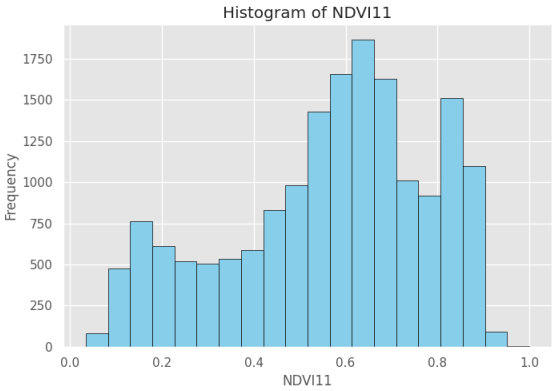


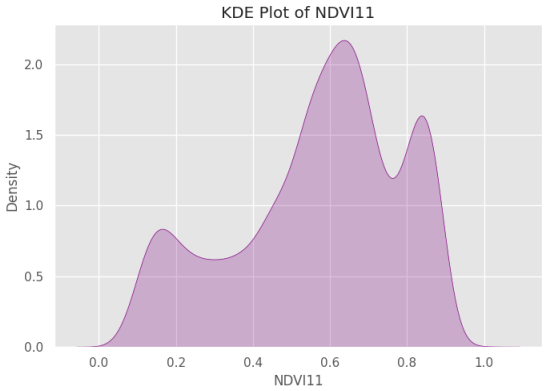


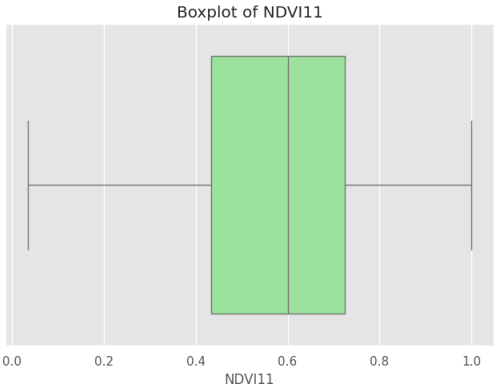
### Key Metrics ###	
count	17098.000000
mean	0.656828
std	0.211748
min	0.037831
25%	0.570149
50%	0.702184
75%	0.801435
max	1.000000

Skewness: -0.96
Missing Values: 0

11. NDVI11



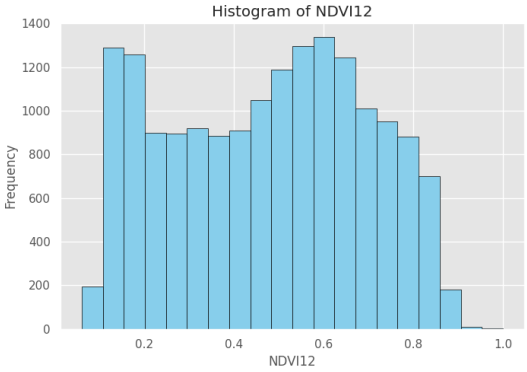


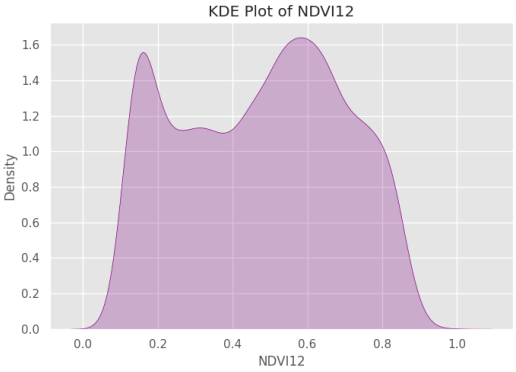


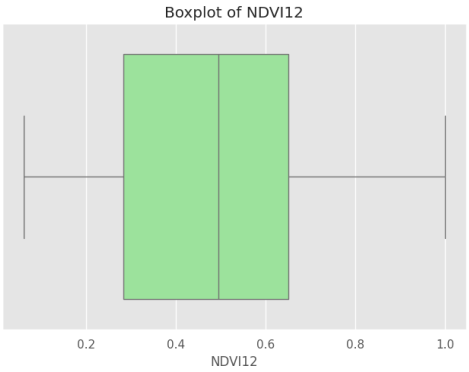
### Key Metrics ###	
count	17098.000000
mean	0.566137
std	0.217242
min	0.034250
25%	0.433338
50%	0.601662
75%	0.724917
max	1.000000

Skewness: -0.52
Missing Values: 0

12. NDVI12



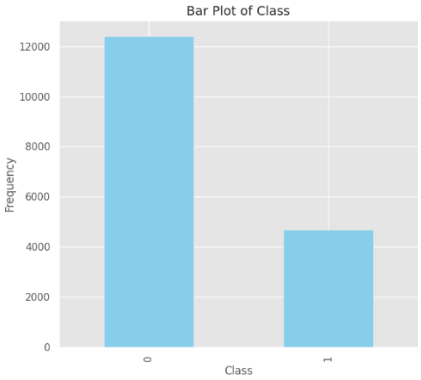


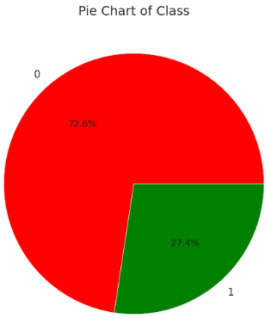


### Key Metrics ###	
count	17098.000000
mean	0.474677
std	0.218313
min	0.060553
25%	0.282425
50%	0.494684
75%	0.650031
max	1.000000

Skewness: -0.06
Missing Values: 0

4.2.1.2 Categorical Features





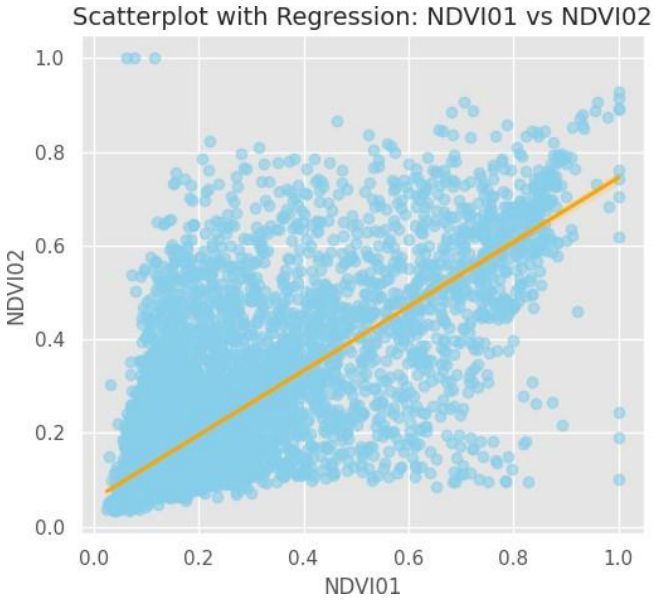

```
### Analysis of 'Class' ###

Value Counts:
0: 12411
1: 4687

Missing Values: 0
```

4.2.2 Bivariate Analysis

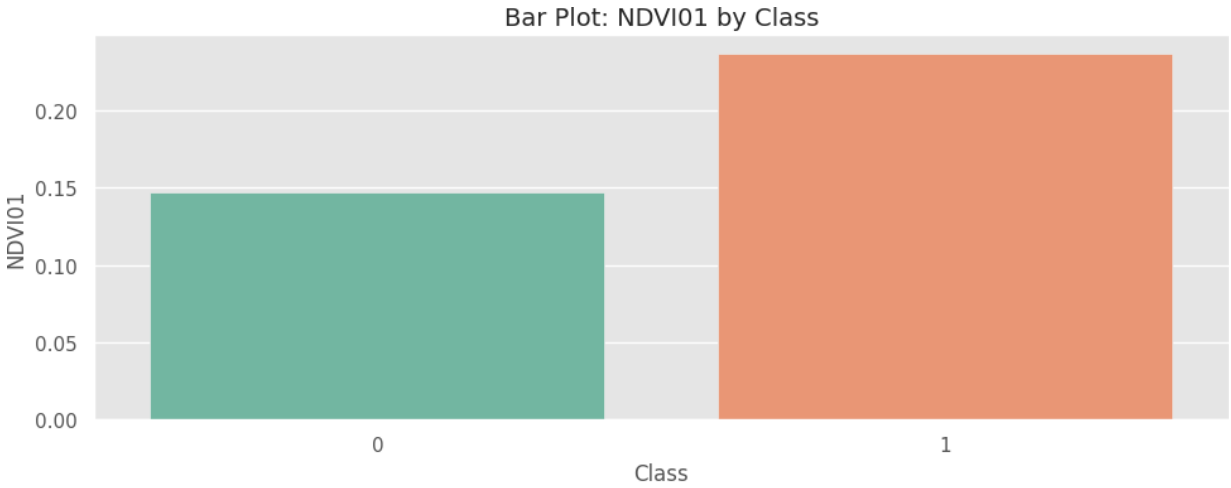
1. **Numerical-Numerical**:
    - Pairwise relationships between NDVI features are analyzed using scatterplots and correlation coefficients.

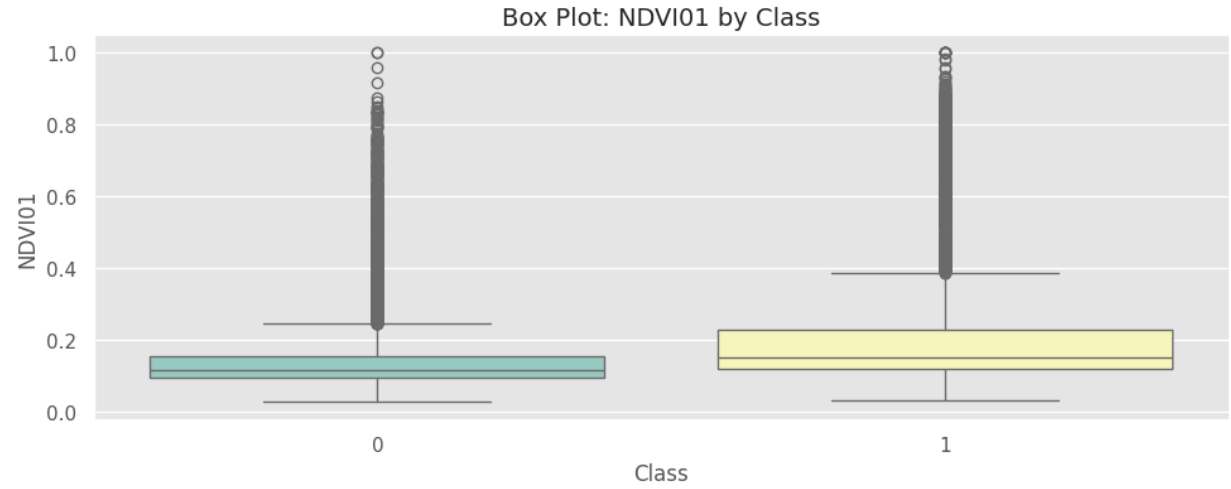


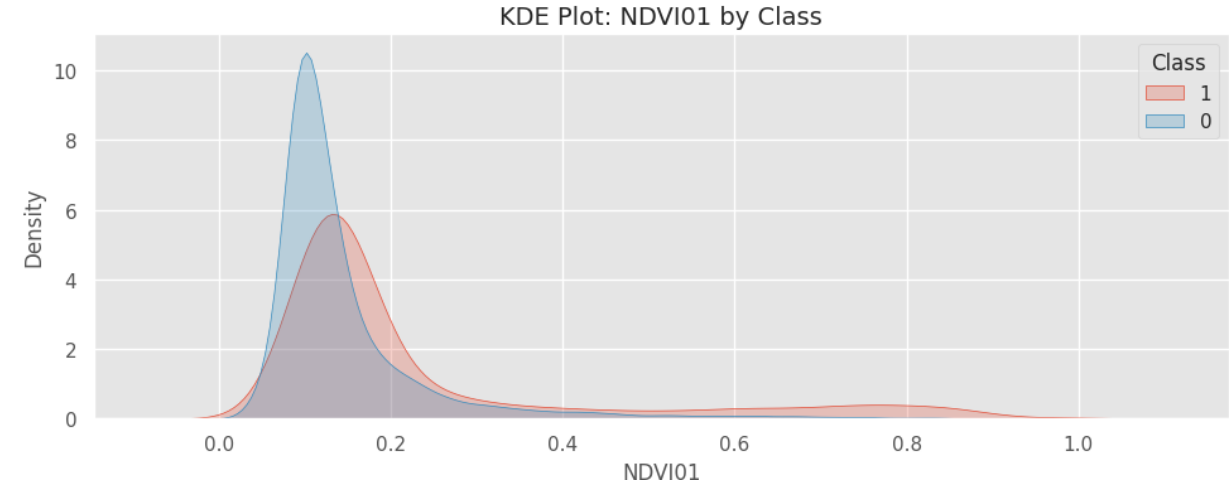

```
### Correlation Analysis ###

Correlation Coefficient: 0.76
```

2. **Numerical-Categorical**:
    - Box plots and violin plots are used to compare NDVI values for rice and cotton crops.







3. **Categorical-Categorical**:
    - Not applicable for this dataset.

4.3 Exploratory Data Analysis for Year 3

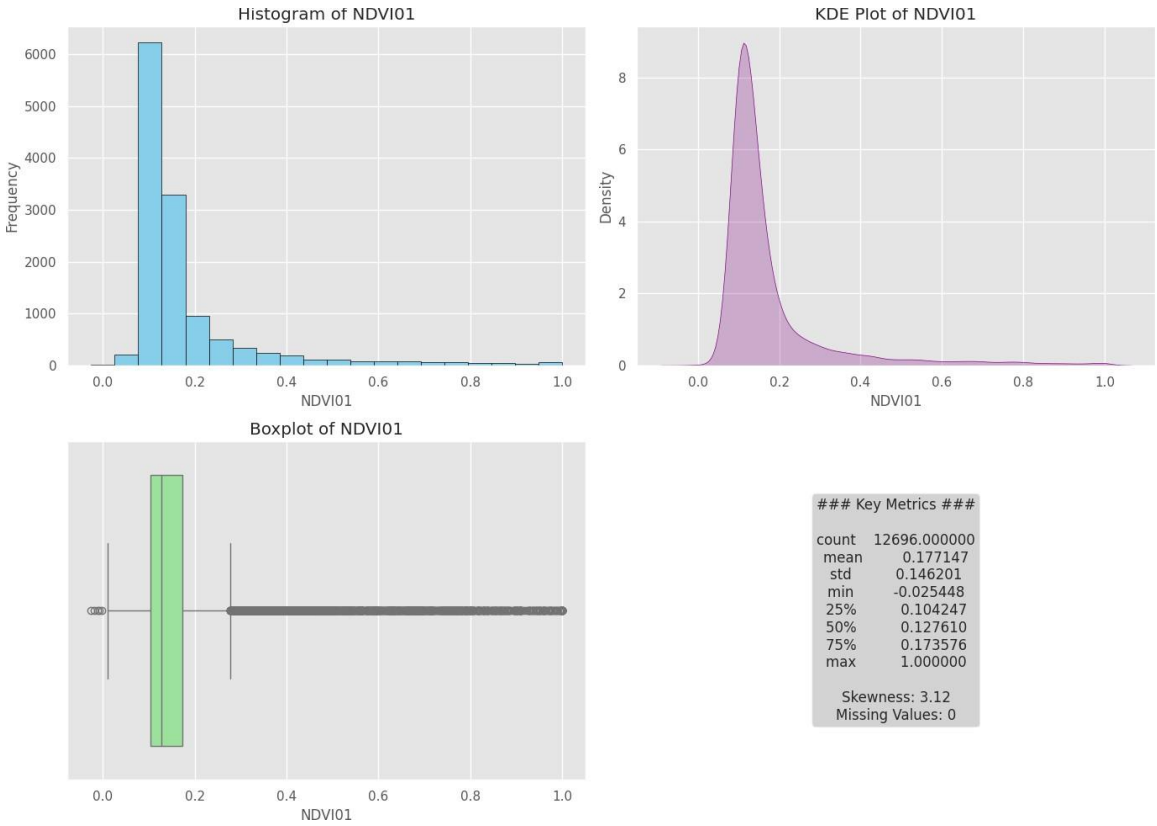
The Year 3 dataset combines NDVI data for rice and cotton crops in 2023 and is analyzed similarly to the datasets for Years 1 and 2.

4.3.1 Univariate Analysis

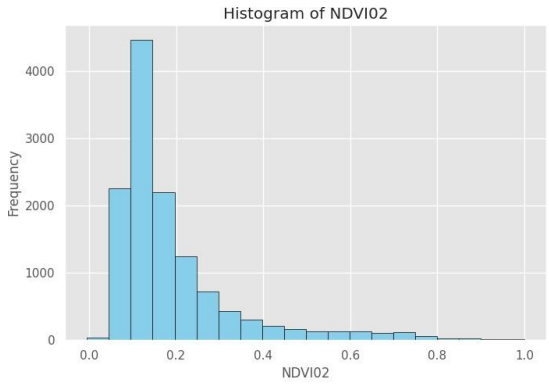
1. **Descriptive Statistics**:
    - Summary statistics provide an overview of the central tendencies and variability of NDVI features.
2. **Visualizations**:
    - Histograms, box plots, and density plots reveal patterns, outliers, and skewness in the data.
3. **Categorical Analysis**:
    - The distribution of Class is examined to ensure balanced representation of rice and cotton data.

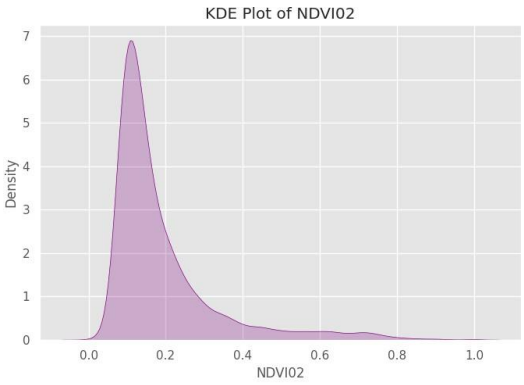
4.3.1.1 Numerical Features

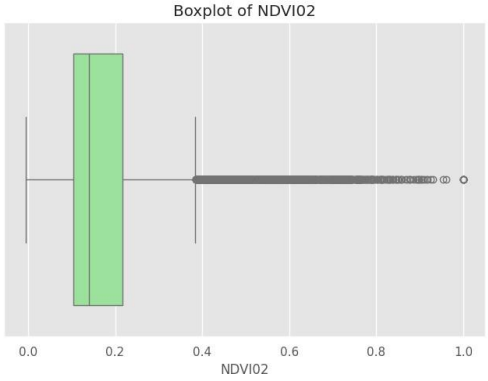
1. **NDVI01**


Histogram of NDVI01
Frequency
NDVI01
KDE Plot of NDVI01
Density
NDVI01
Boxplot of NDVI01
NDVI01
Key Metrics
count 12696.000000
mean 0.177147
std 0.146201
min -0.025448
25% 0.104247
50% 0.127610
75% 0.173576
max 1.000000
Skewness: 3.12
Missing Values: 0

2. NDVI02



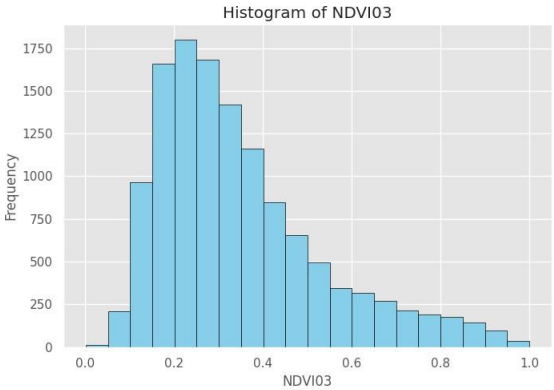


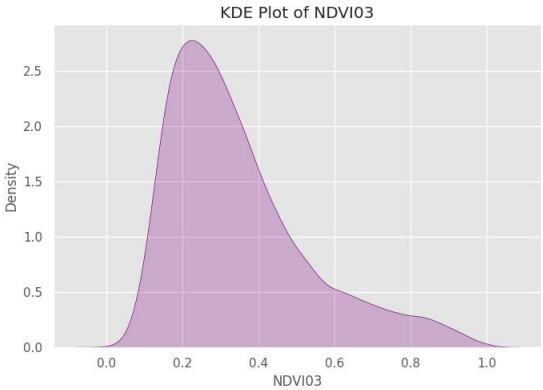


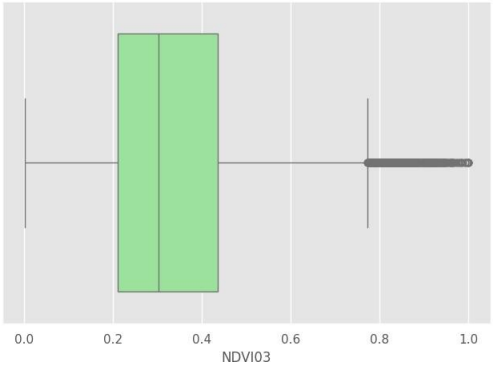
### Key Metrics ###	
count	12696.000000
mean	0.190523
std	0.142232
min	-0.004479
25%	0.104276
50%	0.139753
75%	0.216499
max	1.000000

Skewness: 2.32
Missing Values: 0

3. NDVI03



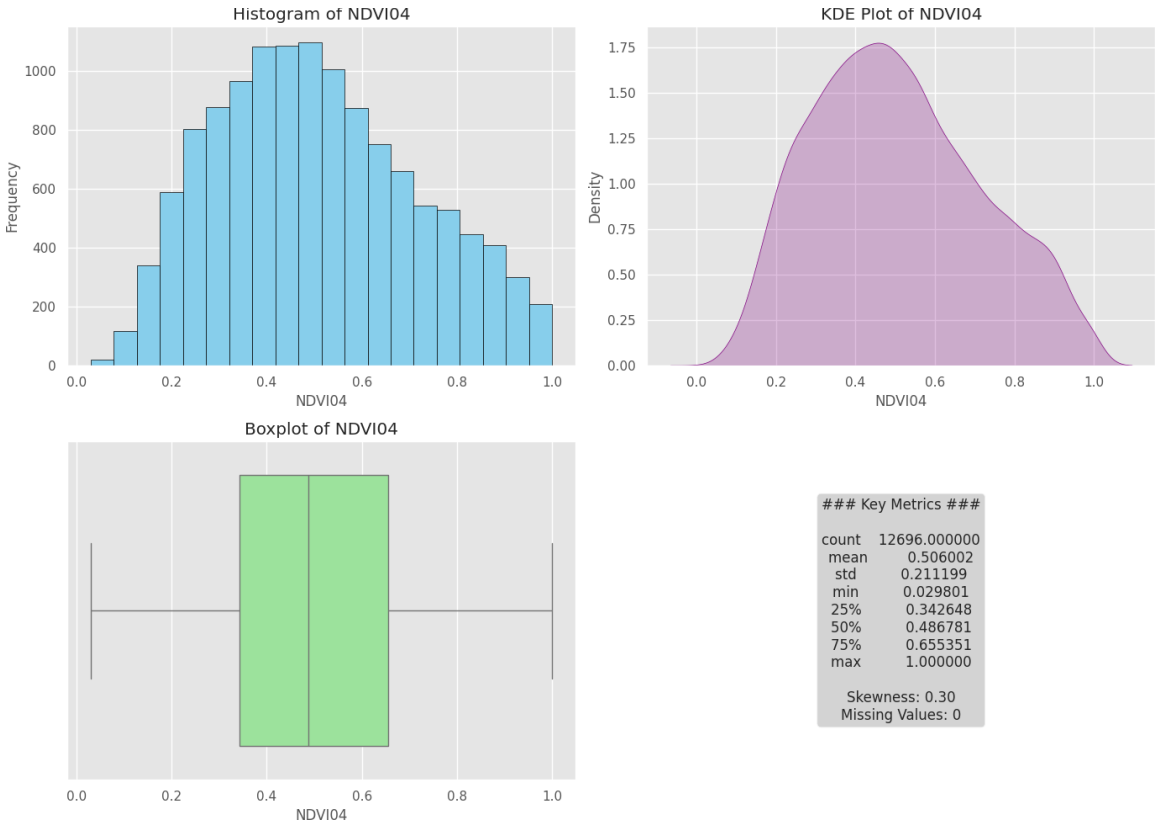




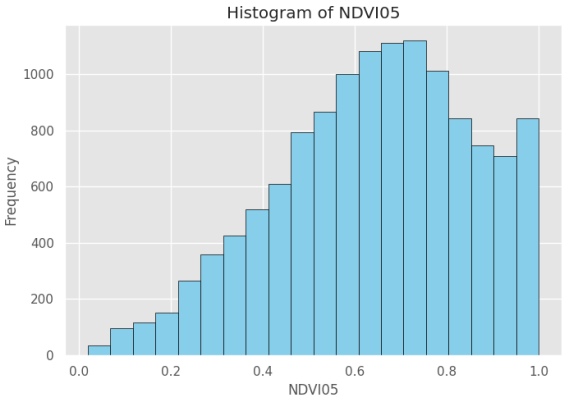
### Key Metrics ###	
count	12696.000000
mean	0.347986
std	0.186310
min	0.002020
25%	0.210620
50%	0.302645
75%	0.435668
max	1.000000

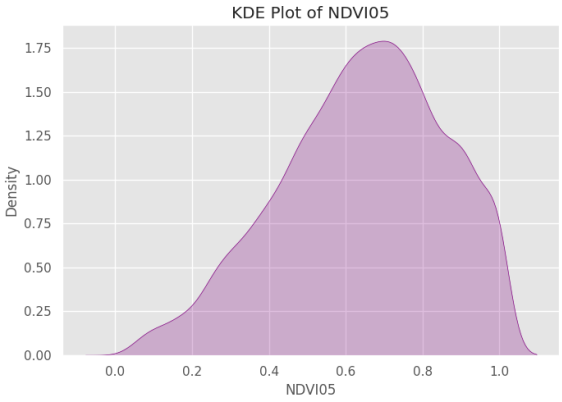
Skewness: 1.12
Missing Values: 0

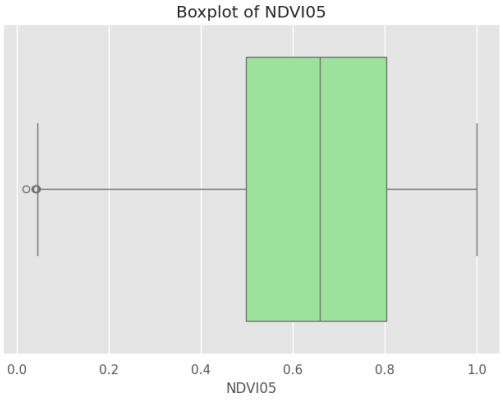
4. NDVI04


Histogram of NDVI04
Frequency
NDVI04
KDE Plot of NDVI04
Density
NDVI04
Boxplot of NDVI04
NDVI04
Key Metrics
count 12696.000000
mean 0.506002
std 0.211199
min 0.029801
25% 0.342648
50% 0.486781
75% 0.655351
max 1.000000
Skewness: 0.30
Missing Values: 0

5. NDVI05



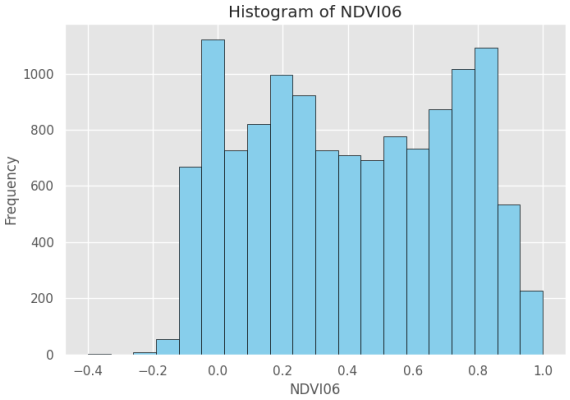


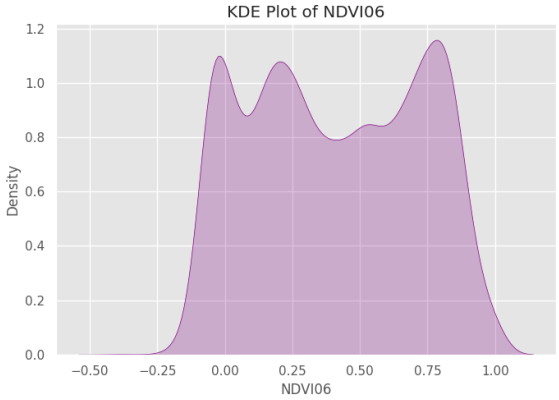


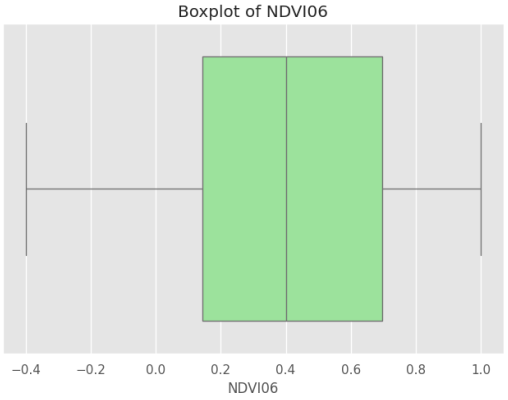
### Key Metrics ###	
count	12696.000000
mean	0.641992
std	0.213128
min	0.018450
25%	0.498181
50%	0.658113
75%	0.802006
max	1.000000

Skewness: -0.36
Missing Values: 0

6. NDVI06



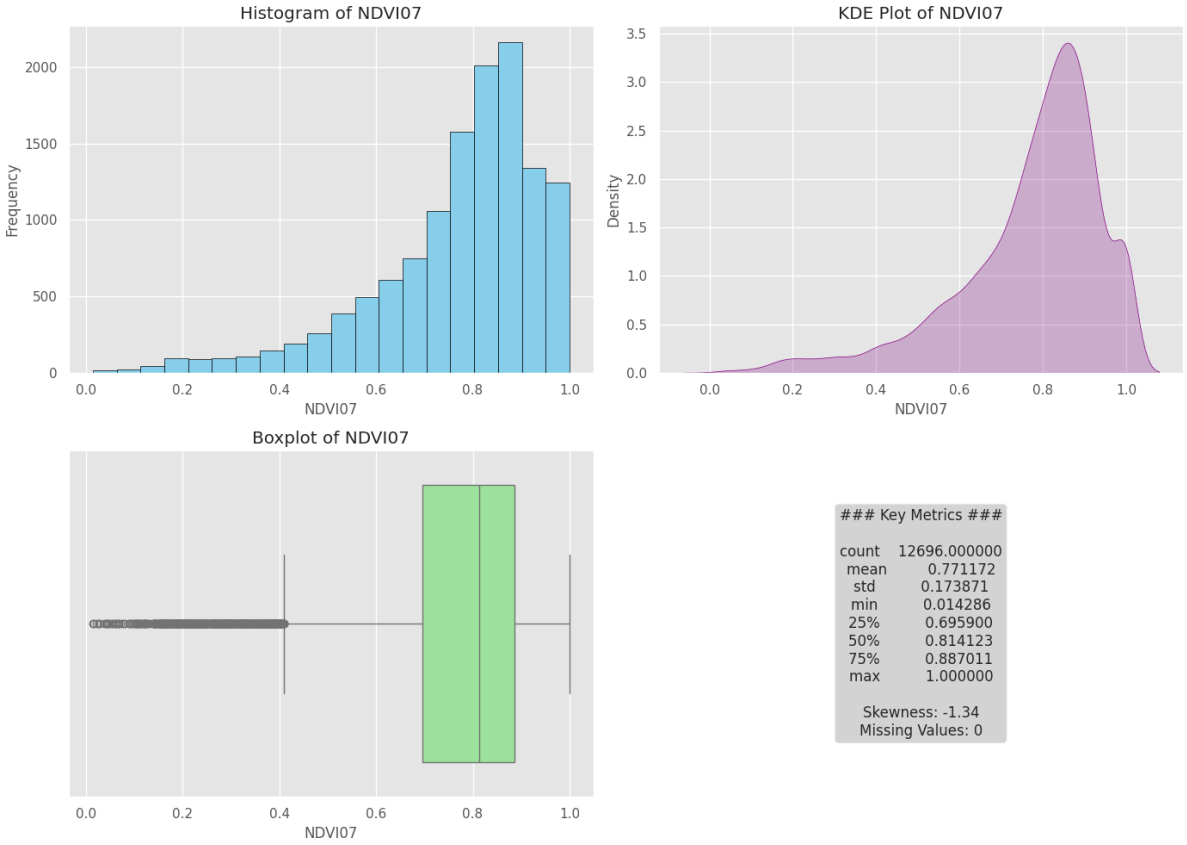




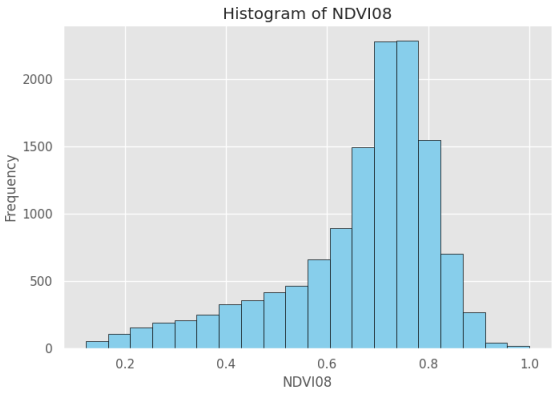
### Key Metrics ###	
count	12696.000000
mean	0.409309
std	0.311598
min	-0.400086
25%	0.143910
50%	0.401572
75%	0.697527
max	1.000000

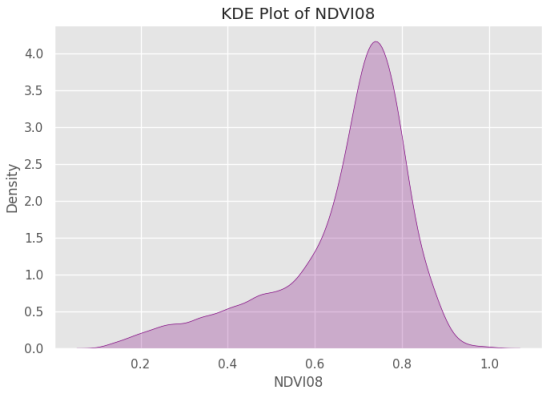
Skewness: 0.02
Missing Values: 0

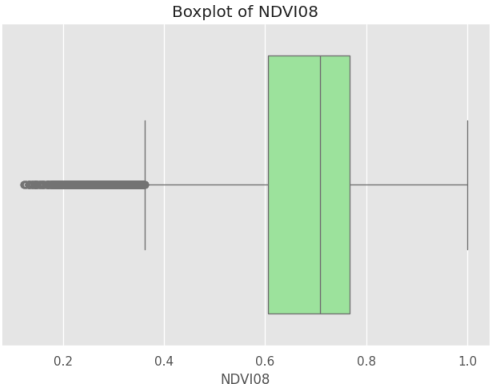
7. NDVI07



8. NDVI08



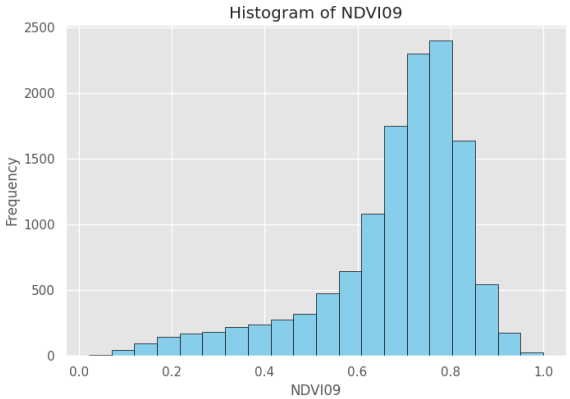


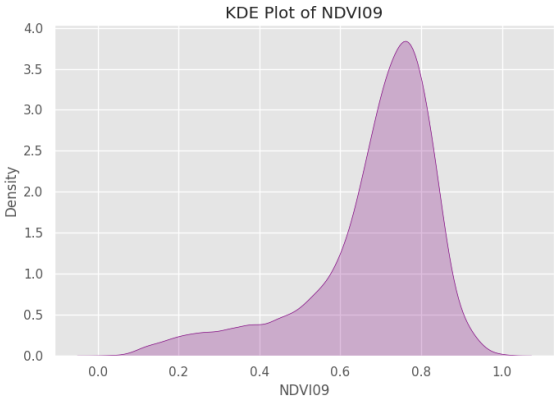


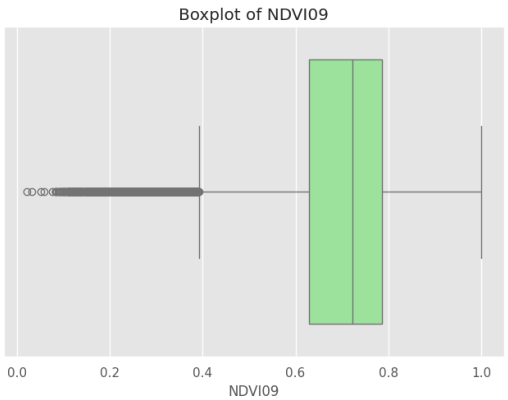
### Key Metrics ###	
count	12696.000000
mean	0.666104
std	0.153354
min	0.122353
25%	0.605182
50%	0.709357
75%	0.767672
max	1.000000

Skewness: -1.22
Missing Values: 0

9. NDVI09



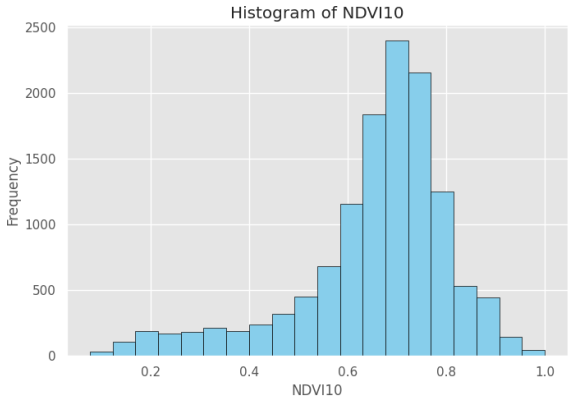


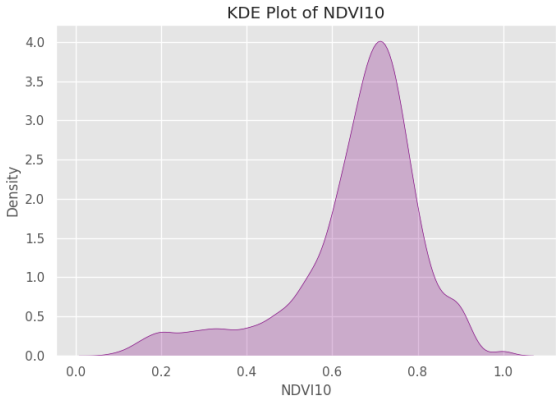


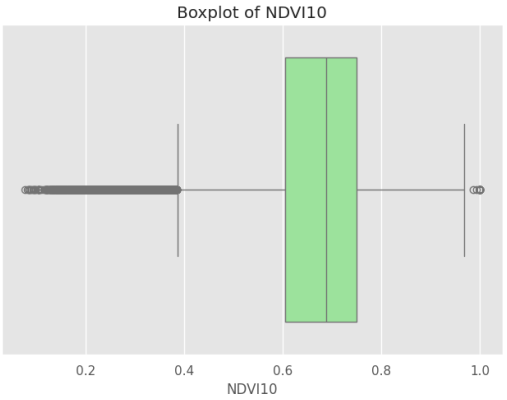
### Key Metrics ###	
count	12696.000000
mean	0.681207
std	0.160149
min	0.021189
25%	0.628723
50%	0.722274
75%	0.786628
max	1.000000

Skewness: -1.41
Missing Values: 0

10. NDVI10



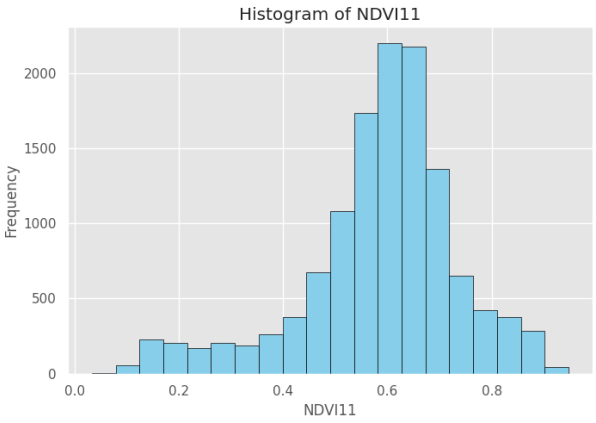


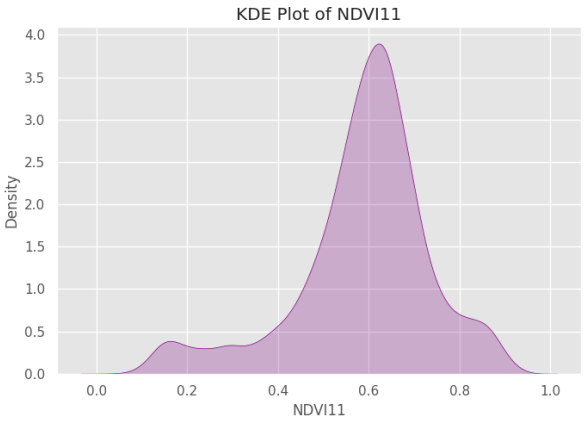


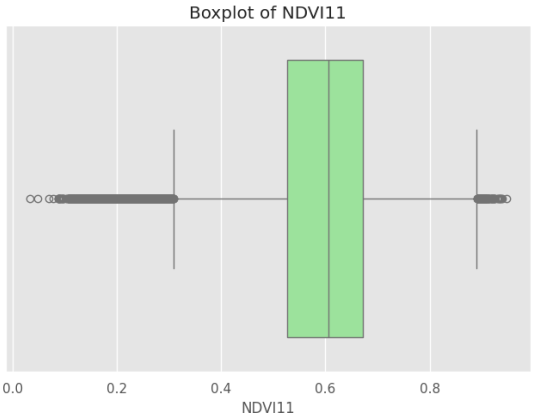
### Key Metrics ###	
count	12696.000000
mean	0.656300
std	0.155842
min	0.076431
25%	0.604465
50%	0.689170
75%	0.750709
max	1.000000

Skewness: -1.24
Missing Values: 0

11. NDVI11





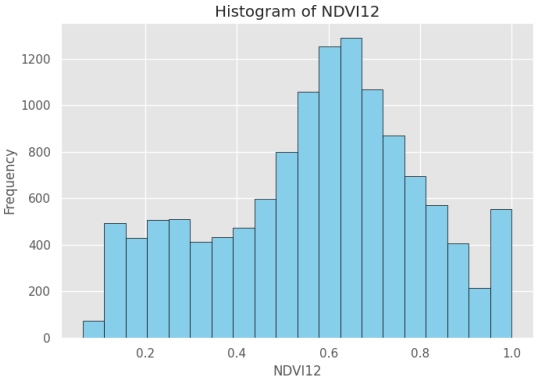


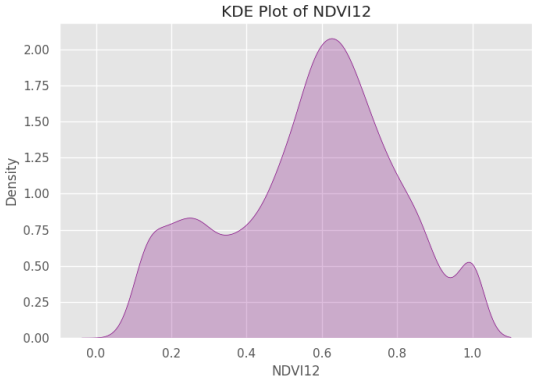
Key Metrics

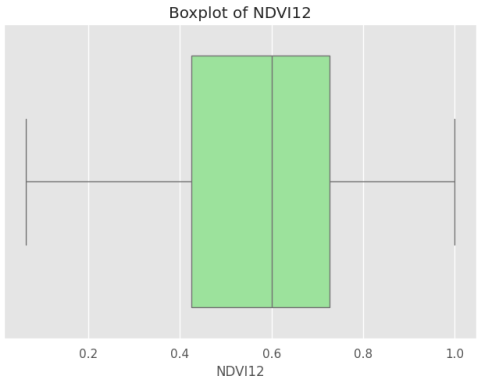
count	12696.000000
mean	0.586659
std	0.150598
min	0.033582
25%	0.526941
50%	0.606270
75%	0.672331
max	0.947242

Skewness: -0.87
Missing Values: 0

12. NDVI12





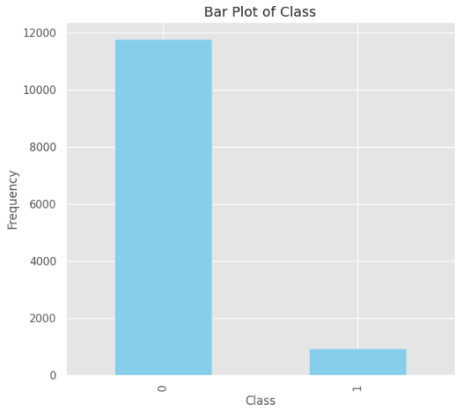


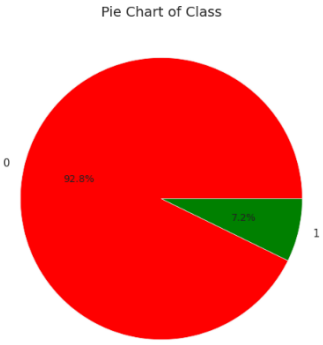
Key Metrics

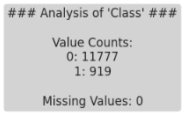
count	12696.000000
mean	0.572473
std	0.224205
min	0.062907
25%	0.424289
50%	0.599762
75%	0.725743
max	1.000000

Skewness: -0.25
Missing Values: 0

4.3.1.2 Categorical Features

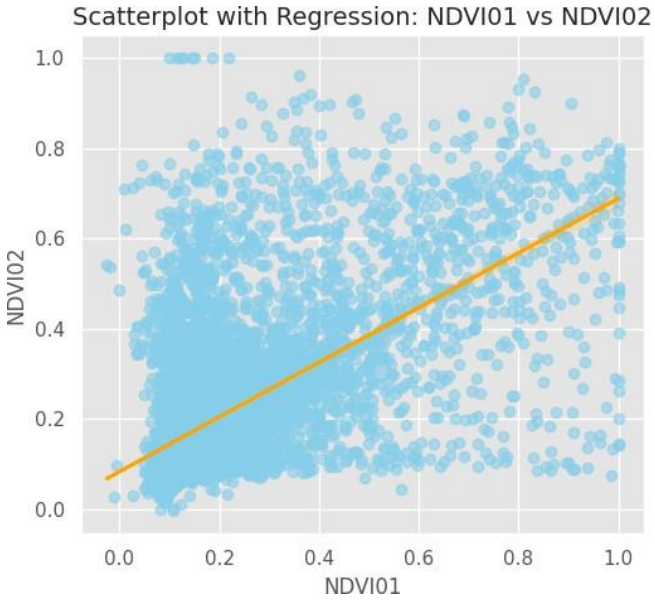


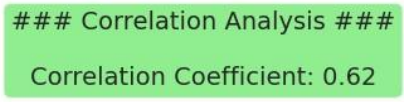




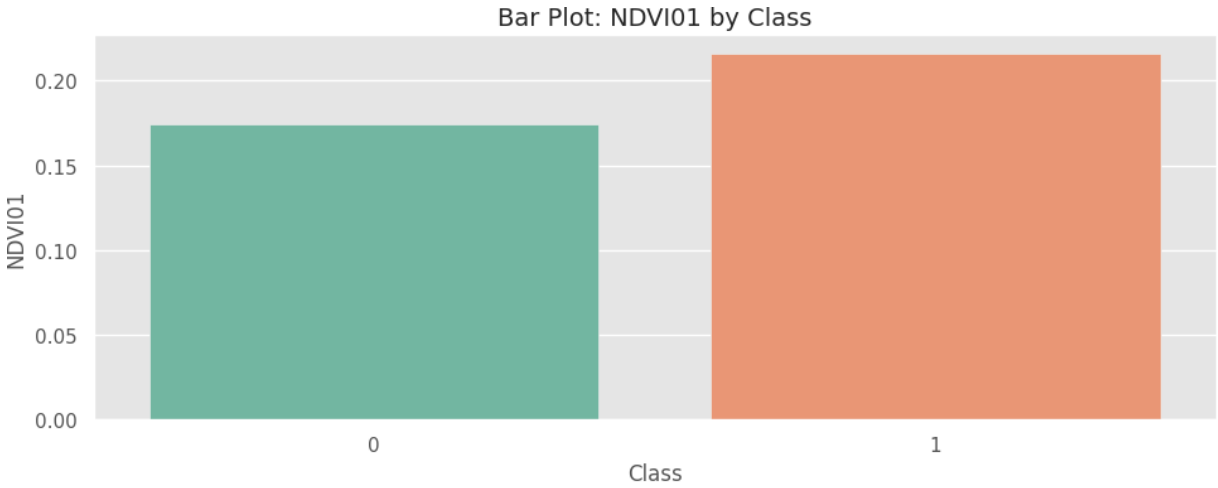
4.3.2 Bivariate Analysis

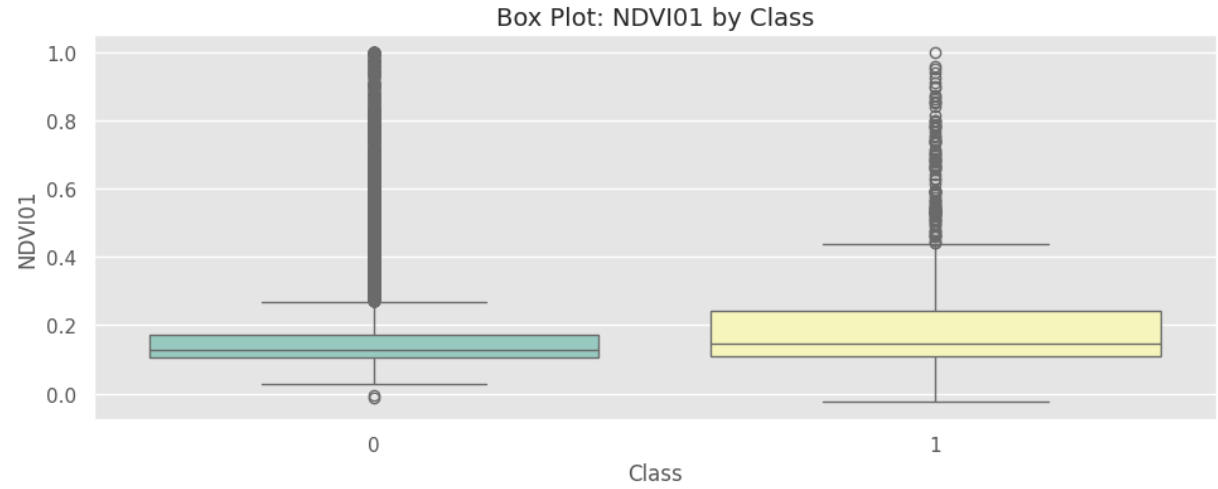
1. **Numerical-Numerical**:
    - Correlations between NDVI features are visualized using scatterplots and regression plots.

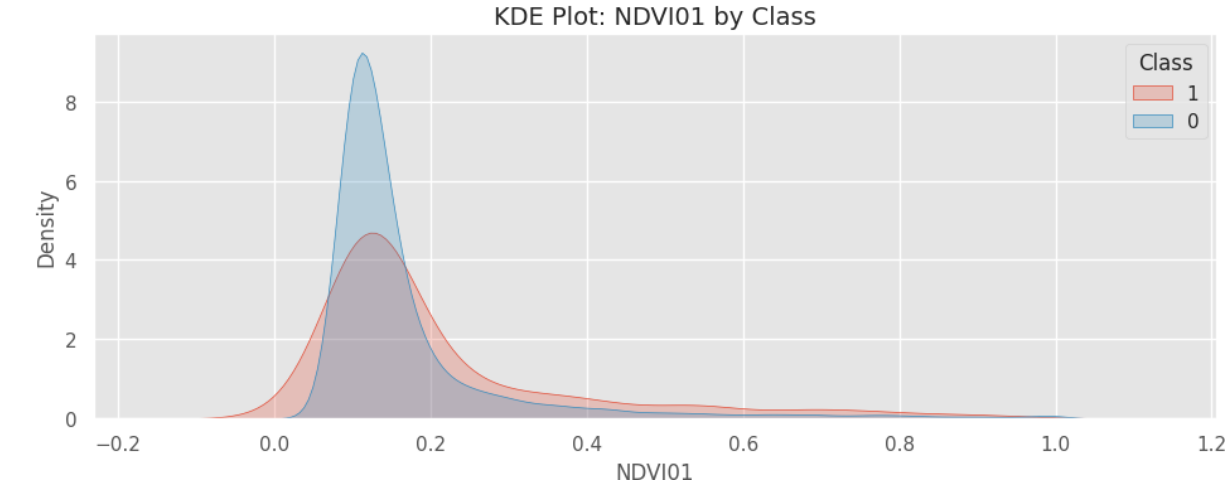




2. **Numerical-Categorical**:
    - NDVI distributions are compared across crop types (rice and cotton) using and box plots.







3. **Categorical-Categorical**:

- Not applicable due to the absence of multiple categorical features.

4.4 Summary

Exploratory Data Analysis for all three years provided the following insights:

1. **Univariate Analysis**:
    - Distributions of NDVI features were examined to identify potential outliers and skewness.
    - Crop types were balanced across the datasets.
2. **Bivariate Analysis**:
    - Strong correlations were observed between certain NDVI features, reflecting temporal patterns in vegetation growth.
    - Distinct distributions of NDVI values between rice and cotton crops were identified.

These findings guide the next steps in data preprocessing, feature engineering, and model selection.

5. Time Series Plot Analysis

Time series analysis is a crucial step in understanding temporal patterns and trends in NDVI (Normalized Difference Vegetation Index) values. This section presents a seasonal analysis of NDVI values for rice and cotton crops over three years (2021, 2022, and 2023) using time series plots.

5.1 Purpose of Time Series Analysis

The goal of this analysis is to:

1. **Visualize Seasonal Patterns**:
    - Observe how NDVI values fluctuate across the growing season.
    - Identify peak growth periods for rice and cotton crops.
2. **Compare Crop Trends**:
    - Highlight differences in temporal NDVI patterns between rice and cotton.
3. **Provide Insights for Modeling**:
    - Understand the data's time-dependent structure to guide feature engineering and model selection.

5.2 Seasonal NDVI Trends

5.2.1 NDVI Columns

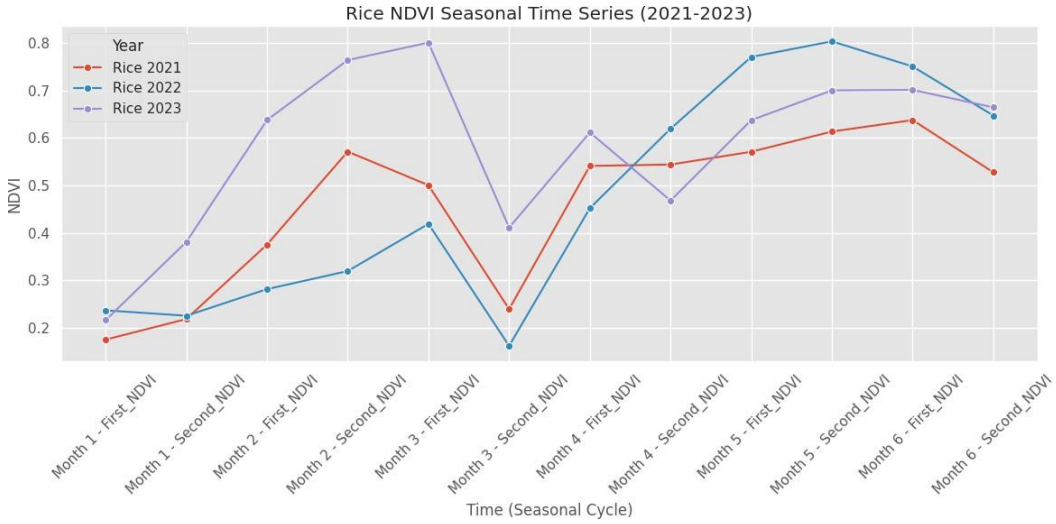
The analysis uses the following 12 NDVI features, which represent monthly NDVI values throughout the year:

- NDVI01, NDVI02, NDVI03, NDVI04, NDVI05, NDVI06, NDVI07, NDVI08, NDVI09, NDVI10, NDVI11, and NDVI12.

5.2.2 Seasonal Time Series for Rice

The **seasonal_time_series_rice()** function generates a time series plot for rice crops using the NDVI features. The plot illustrates:

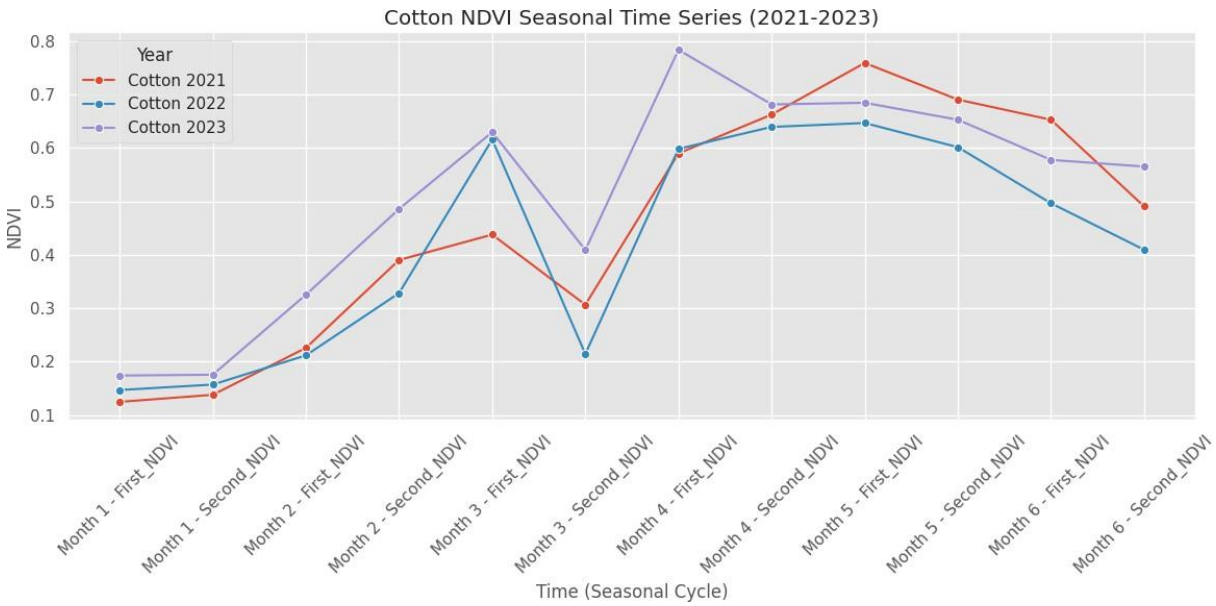
- The average NDVI values for rice crops across three years (2021, 2022, and 2023).
- Seasonal trends, such as the gradual increase in NDVI values during the early growing season, reaching a peak during the middle of the season, followed by a decline towards the end.



5.2.3 Seasonal Time Series for Cotton

The **seasonal_time_series_cotton()** function generates a similar plot for cotton crops. The plot shows:

- The average NDVI values for cotton crops across three years (2021, 2022, and 2023).
- Distinct growth patterns compared to rice, with earlier peaks in NDVI values and a faster decline, reflecting differences in the growth cycles of cotton and rice.



5.3 Observations

1. **Distinct Growth Cycles**:
    - **Rice**: NDVI values for rice generally peak later in the growing season, reflecting its longer growth cycle.
    - **Cotton**: NDVI values for cotton peak earlier and decline faster, indicating its shorter growth cycle.
2. **Seasonal Consistency Across Years**:
    - The NDVI trends for both rice and cotton exhibit consistent seasonal patterns across 2021, 2022, and 2023, suggesting reliable data collection and clear crop phenology.
3. **Crop Differentiation**:
    - The temporal differences in NDVI trends provide a strong basis for distinguishing between rice and cotton crops, which is valuable for crop classification models.

5.4 Visualizations

- **Rice NDVI Seasonal Time Series**: A line plot showing the mean NDVI values for rice crops across all 12 months for the years 2021, 2022, and 2023.
- **Cotton NDVI Seasonal Time Series**: A similar line plot for cotton crops, highlighting its unique seasonal NDVI behavior.

Both plots include:

- **X-axis**: Months (NDVI01 to NDVI12).
- **Y-axis**: Mean NDVI values.
- **Title**: Indicating the crop type and time span.

5.5 Summary

The time series plot analysis provides a clear understanding of the seasonal NDVI behavior for rice and cotton crops. These insights will be critical for:

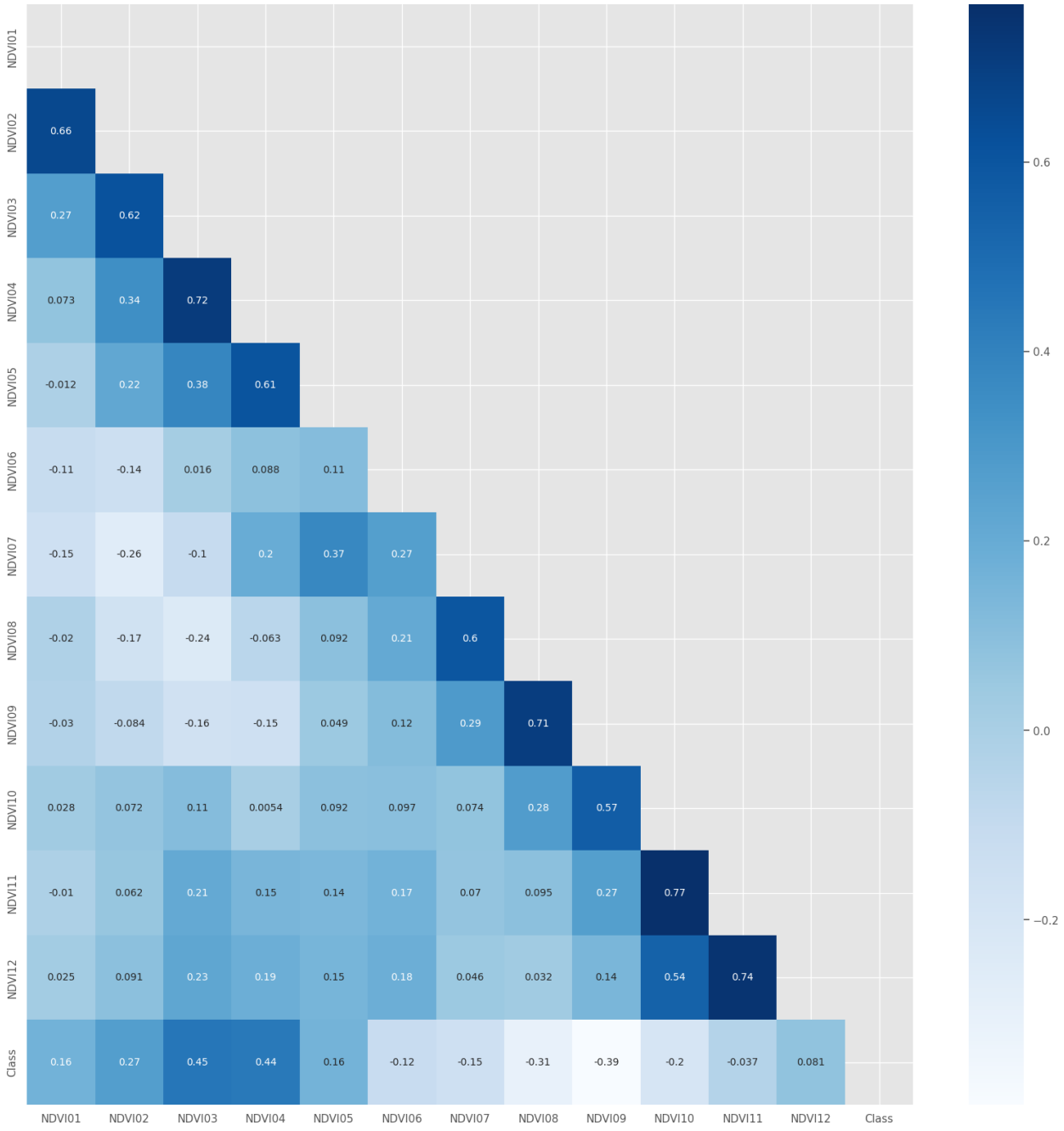
- **Feature Engineering**: Incorporating temporal patterns into models.

6. Investigation of Correlation Matrix

The correlation matrix is used to evaluate the relationships between variables in a dataset. Each value represents the correlation coefficient, where 1 indicates a perfect positive correlation, -1 indicates a perfect negative correlation, and 0 indicates no correlation. Analyzing these relationships helps identify patterns and dependencies between variables.

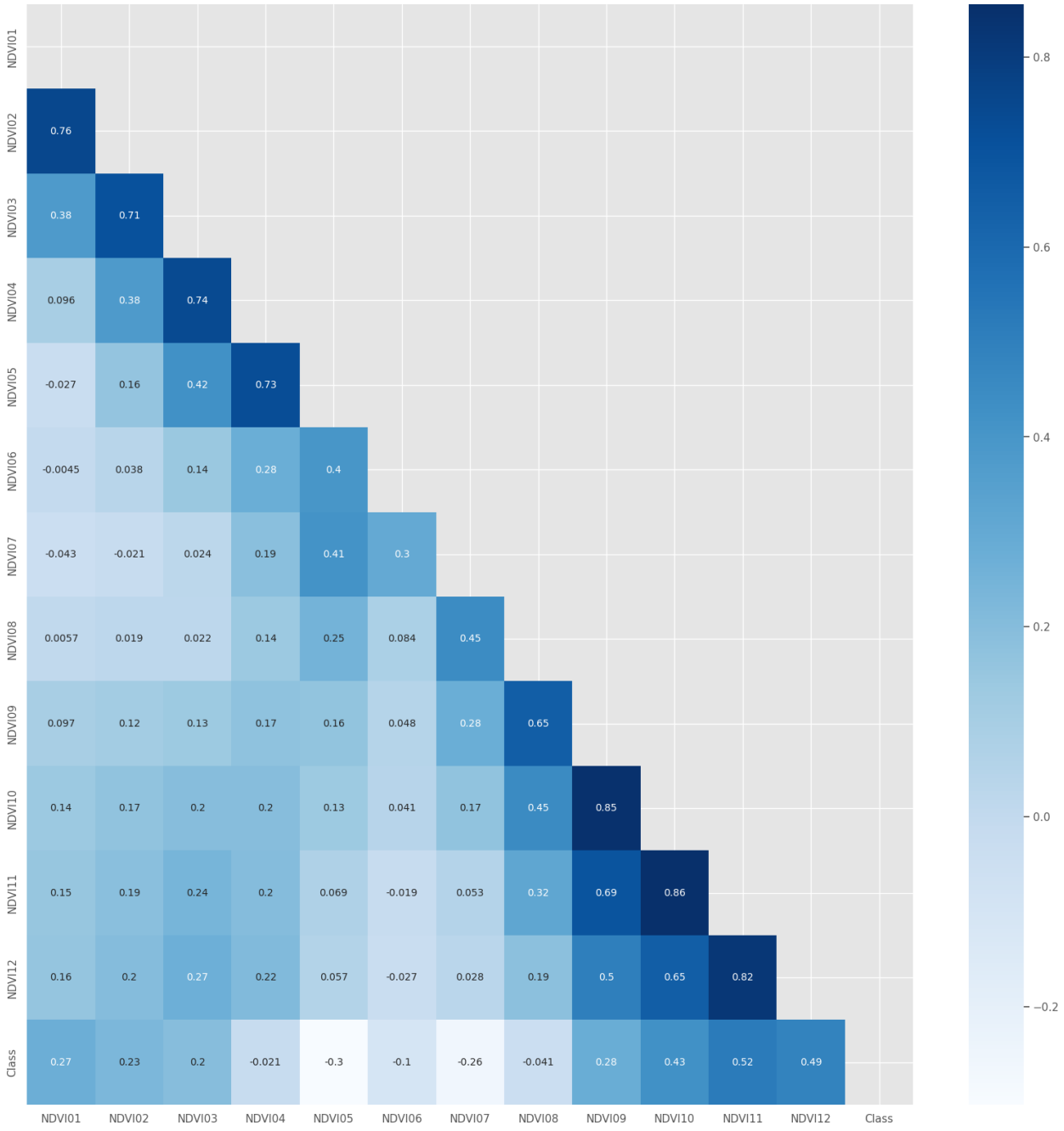
6.1 Correlation Matrix for Year 1

For Year 1, the correlation matrix is computed by calculating pairwise correlation coefficients between numerical variables. A heatmap is used to visualize the relationships, where the upper triangle is masked to avoid redundancy, and the **cmap='Blues'** color scheme highlights correlation strengths.



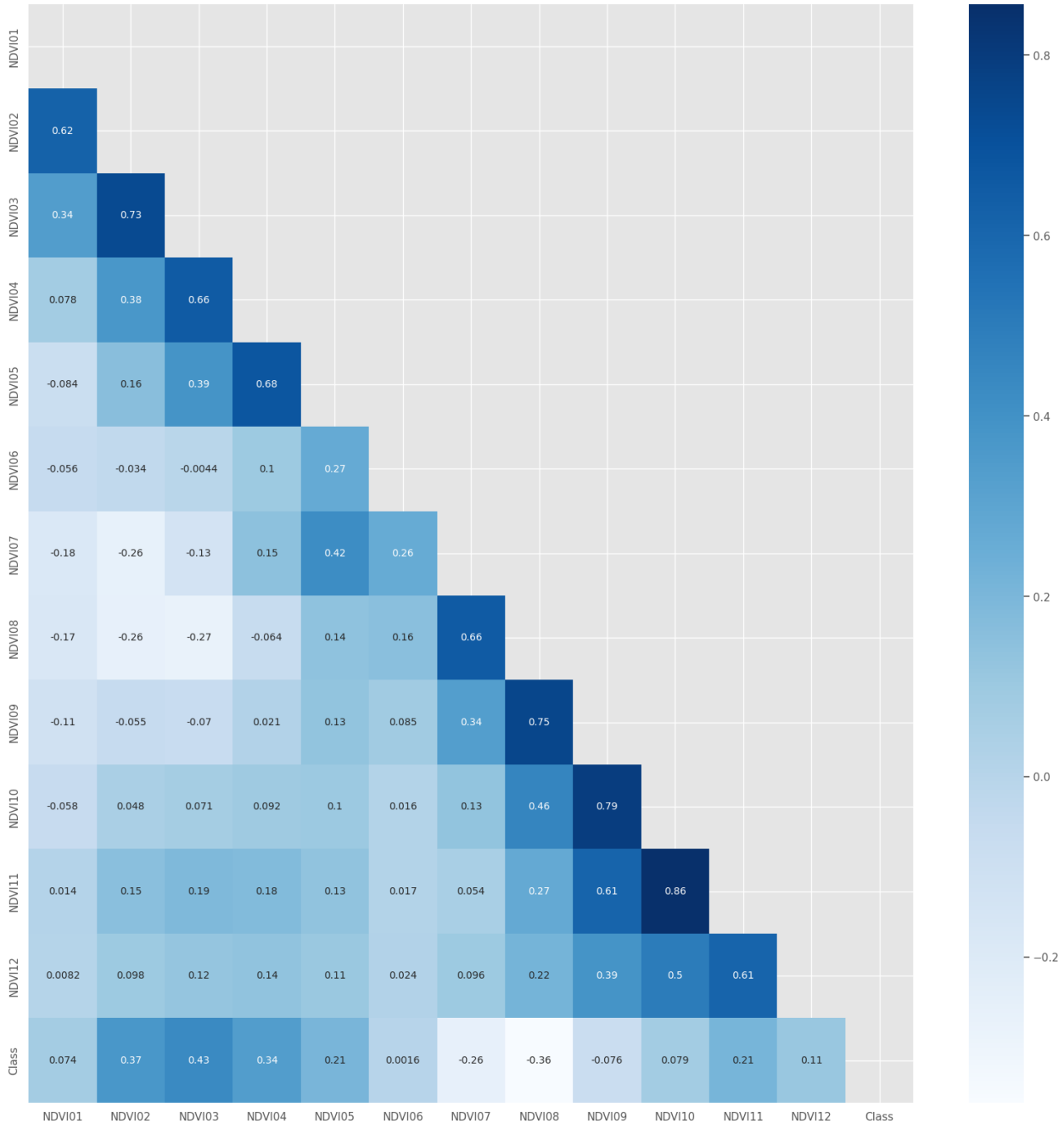
6.2 Correlation Matrix for Year 2

Similarly, the correlation matrix for Year 2 is generated and visualized using a heatmap. Comparing this matrix with Year 1's allows us to observe changes or emerging trends in the relationships between variables.



6.3 Correlation Matrix for Year 3

The same procedure is followed for Year 3, generating a correlation matrix and visualizing it with a heatmap. Comparing all three years reveals any long-term trends, consistent relationships, or shifts over time.



6.4 Summary

In summary, the correlation matrices for the three years provide a clear view of variable relationships. The heatmap visualizations allow easy identification of strong correlations and trends across years, offering valuable insights for predictive modeling and data-driven decision-making.

Preprocessing data is a crucial step in any machine learning pipeline. It ensures that the dataset is clean, balanced, and properly formatted, which leads to better model performance and generalizability. The following sections outline the steps taken for preprocessing the dataset for Year 1, and similar methods were applied to the data for Years 2 and 3.

7. Preprocessing

7.1.1 Handling Missing Values

Missing values in the dataset can distort statistical analyses and affect model performance. The first step was to inspect the dataset for missing values using the check_missing_values function. Missing values were then either imputed or dropped, depending on their nature and prevalence.

- **Importance**: Ensures no bias or misrepresentation of data due to missing entries.
- **Action Taken**: The dataset was checked, and appropriate handling strategies were implemented.

7.1.2 Handling Duplicates and Low Variance Features

Duplicate rows and features with low variance can reduce model performance and increase computational overhead.

Duplicates:

- The dataset was checked for duplicates using the check_duplicates function and analyzed with calculate_duplicate_percentage.
- The duplicates were further examined to determine whether they were genuine or artifacts of data collection.
    - **Genuine Duplicates**: Rows where all NDVI values and class labels were identical were considered true duplicates.

- **Decision**: True duplicates were dropped, which removed 537 rows from the dataset, improving class balance to some extent.

Low Variance Features:

- Features with little to no variance were identified using inspect_data. Such features add no informational value to the model and were removed.

Impact:

- Improved data quality.
- Reduced computational complexity.

7.1.3 Outlier Detection and Removal

Outliers can skew model predictions, especially for algorithms sensitive to distance metrics like Support Vector Machines (SVM).

Detection:

- **Z-Score Method**: The Z-score method was used to identify data points lying beyond a threshold.

Removal:

- Rows identified as outliers were dropped from the dataset.

Post-Processing:

- A correlation matrix was generated and visualized using a heatmap to ensure the removal of outliers did not disturb meaningful relationships within the data.

Impact:

- Enhanced robustness of the dataset.
- Reduced model sensitivity to extreme values.

7.1.4 Addressing Class Imbalance

Class imbalance can lead to biased model predictions. To address this, the following techniques were applied:

Analyze Class Imbalance:

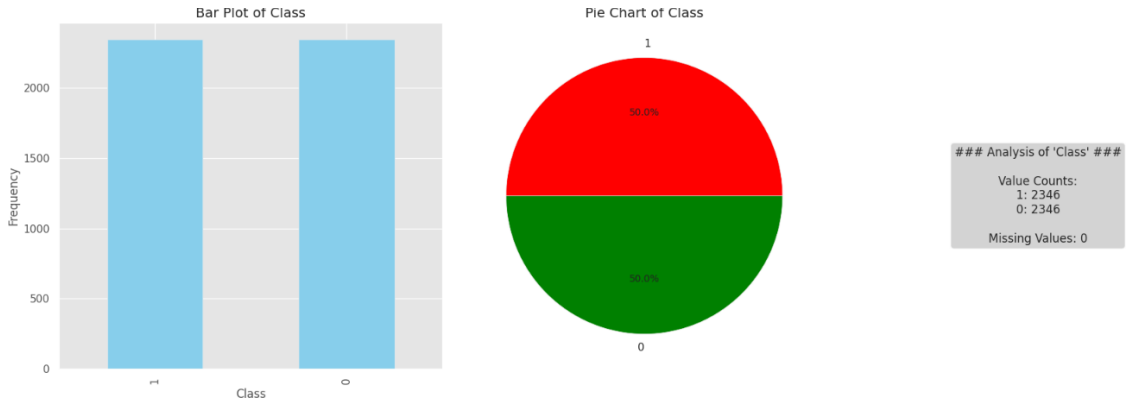
- The dataset's class distribution was analyzed using analyze_categorical_feature.
- An exact balance was preferred to ensure unbiased predictions across classes.

Resampling Techniques:

1. **Random Oversampling**:
    - Minority class samples were duplicated until class distribution was balanced.
    - Used random_oversample_dataset for this purpose.
2. **Random Undersampling**:
    - Majority class samples were reduced to match the minority class size.
    - Used random_undersample_dataset for implementation.
3. **Synthetic Minority Oversampling Technique (SMOTE)**:
    - Synthetic samples were generated for the minority class using SMOTE.
    - Used smote_oversample_dataset for this step.

Result:

- Class balance was achieved, ensuring improved model fairness.



7.1.5 Feature Scaling

Feature scaling is essential for distance-based models like SVM to ensure fair contributions from all features.

Skewness Analysis:

- The skewness of features was calculated and visualized.

Transformation and Scaling:

1. **Power Transformation**:
    - Applied Yeo-Johnson transformation to reduce skewness and make feature distributions more symmetric.
2. **Standardization**:
    - Features were standardized to have a mean of 0 and standard deviation of 1 using the Z-score method.

Scaling for Resampled Datasets:

- Resampled datasets (oversampled, undersampled, SMOTE) were scaled separately to ensure consistency across all variations.

Impact:

- Improved compatibility with SVM and other models.
- Reduced potential biases caused by feature scaling differences.

7.1.6 Feature Engineering and Selection

Feature engineering and selection aim to improve model performance by reducing irrelevant or redundant features.

Importance from Ensemble Models:

- Ensemble methods like Random Forest and XGBoost were used to calculate feature importance scores.
- The most significant features were selected for SVM training, improving computational efficiency and reducing overfitting risk.

Workflow:

- **Rank Features**: Features were ranked based on importance.
- **Top-K Selection**: Top-K features were retained for further modeling.

Impact:

- Optimized dataset for SVM training.
- Balanced computational efficiency with model accuracy.

7.1.7 Validation Split

To evaluate model performance effectively, the dataset was split into training and testing sets.

Workflow:

1. Resampled datasets were split into features (X) and target labels (y).

2. Data was split into training and testing sets using an 80-20 split with train_test_split.

Separate Splits for Each Resampling Technique:

- Separate training and testing sets were created for:
    - Random Oversampling
    - Random Undersampling
    - SMOTE

Impact:

- Ensured fair evaluation of models trained on different resampled datasets.
- Maintained data consistency for comparative analysis.

7.1.8 Visualization

Scatter plots were used to visualize the class distribution in:

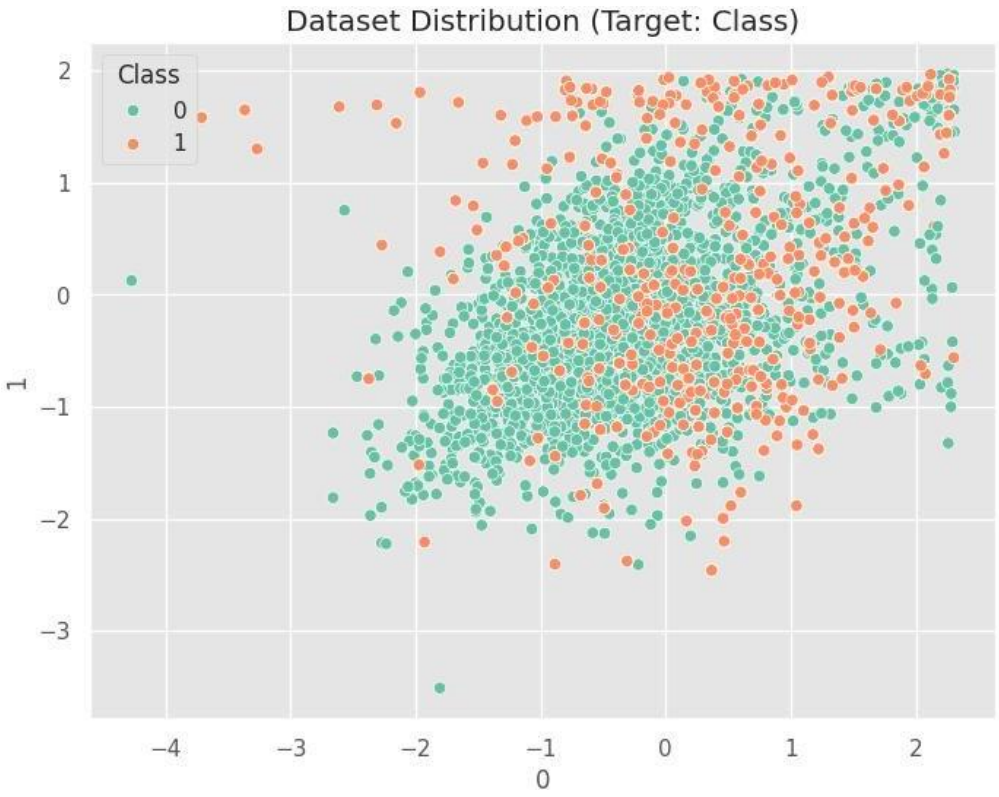
- Original dataset.
- Oversampled, undersampled, and SMOTE datasets.

Purpose:

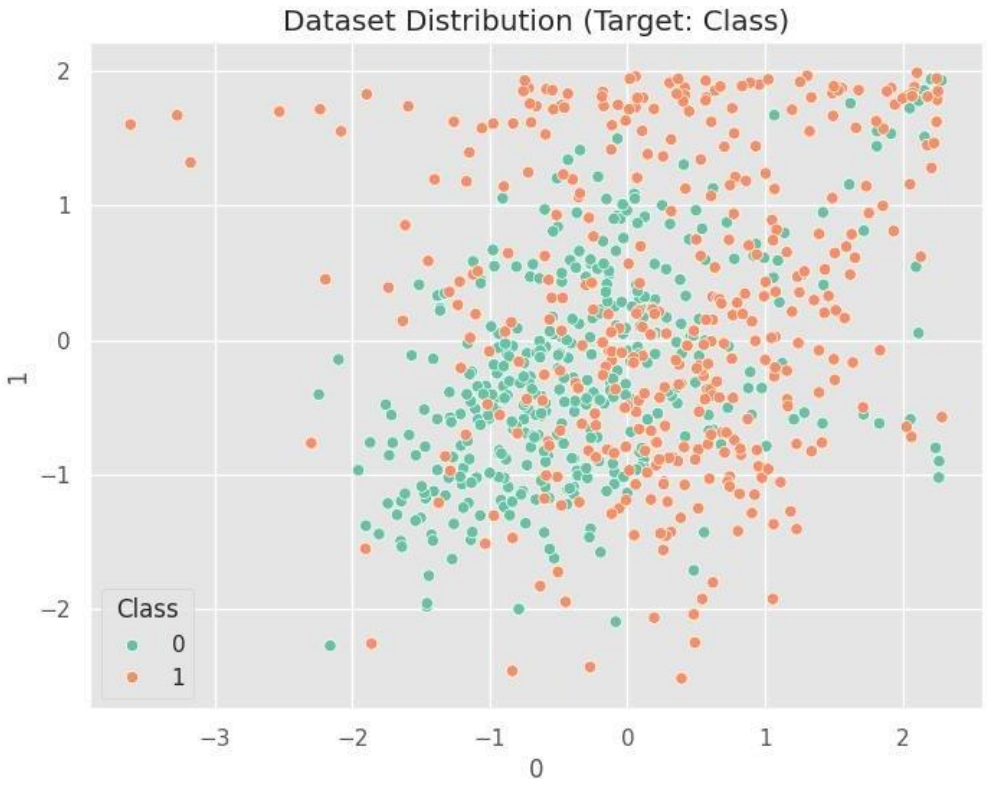
- Validate the effectiveness of resampling techniques.
- Observe class separation and overlap in feature space.

7.1.8.1 Balanced Data Of Year 1

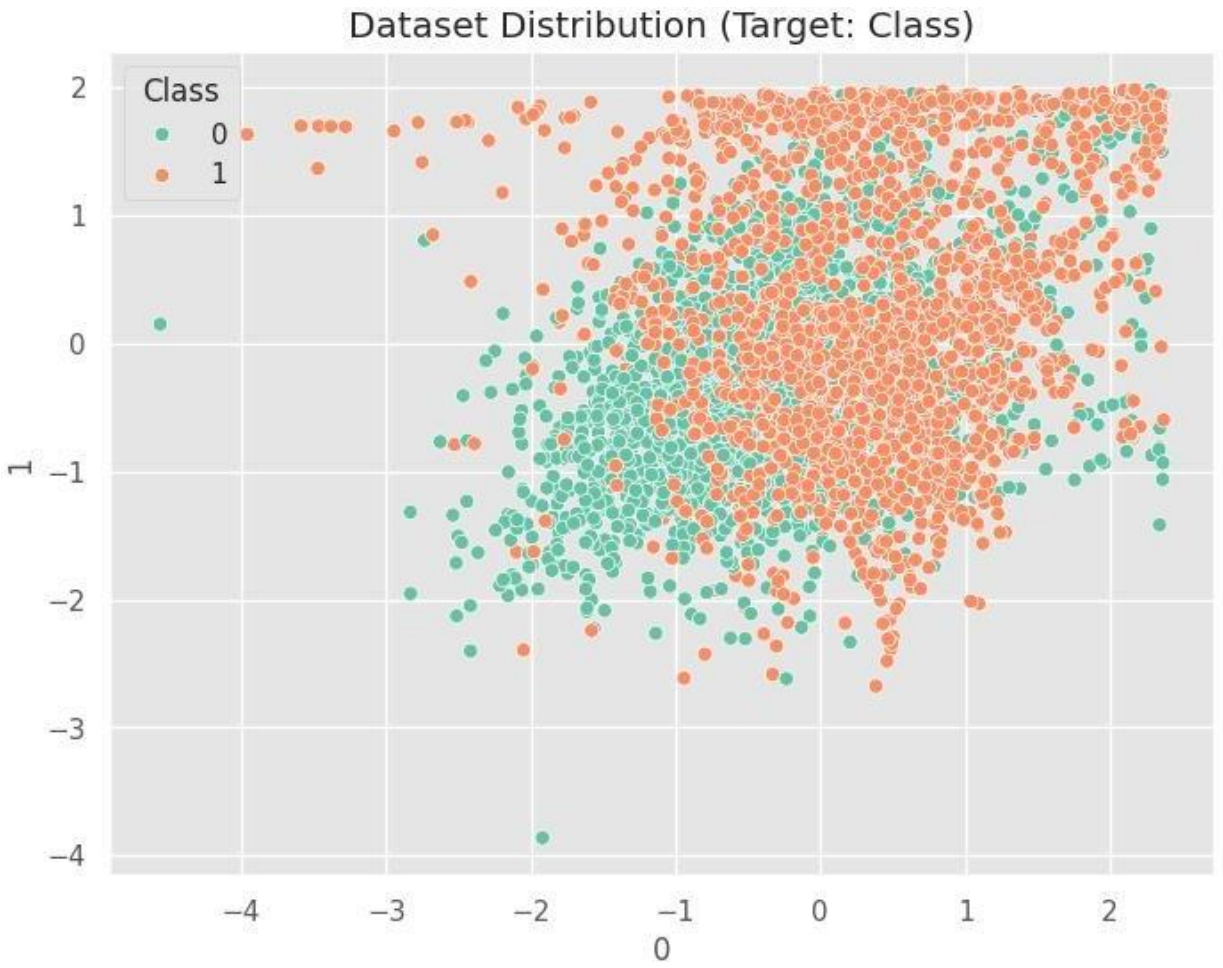
Randomly Oversampled Data



Randomly Undersampled Data

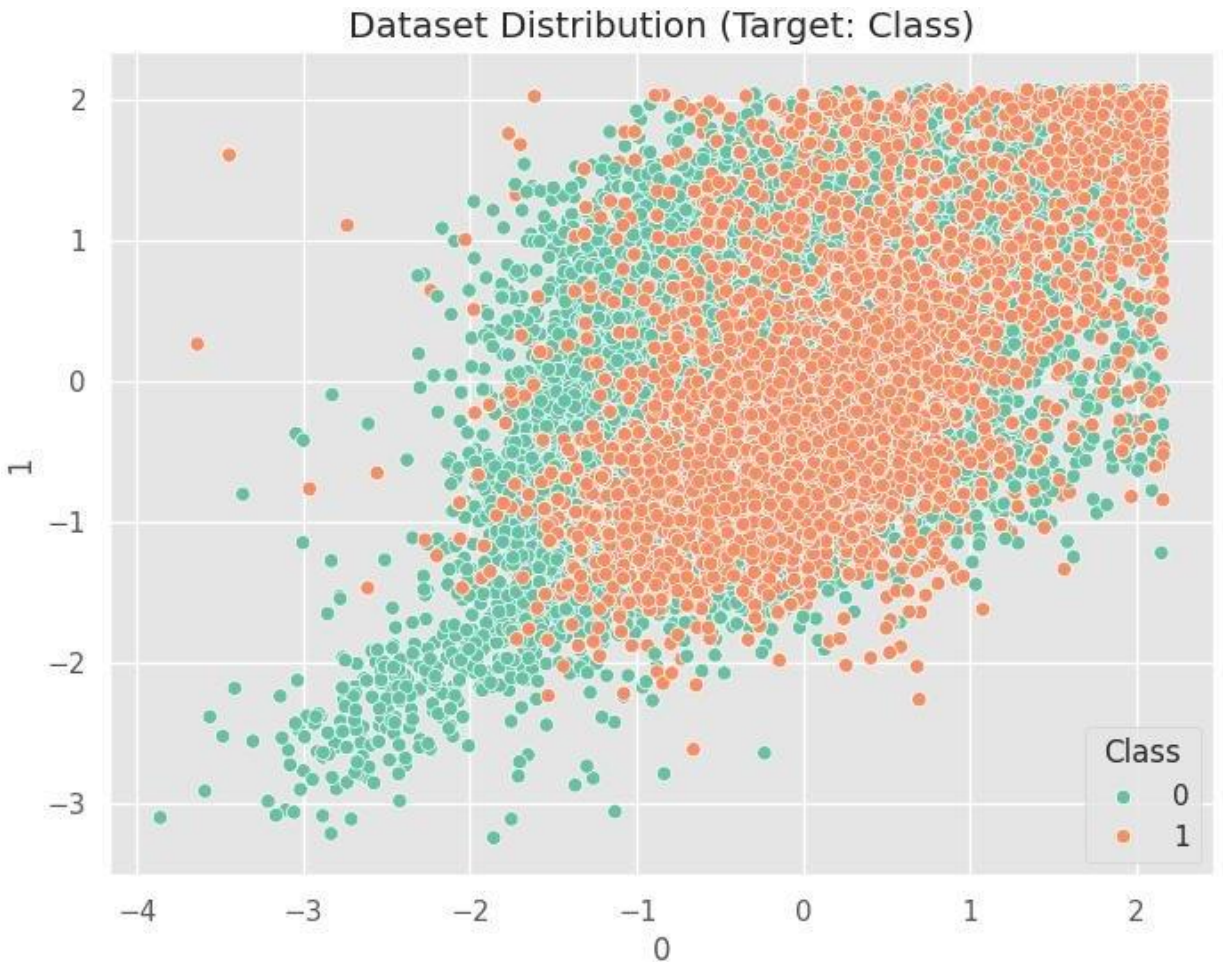

Dataset Distribution (Target: Class)
2
1
0
−1
−2
1
Class
0
1
−3
−2
−1
0
1
2
0

SMOTE Data

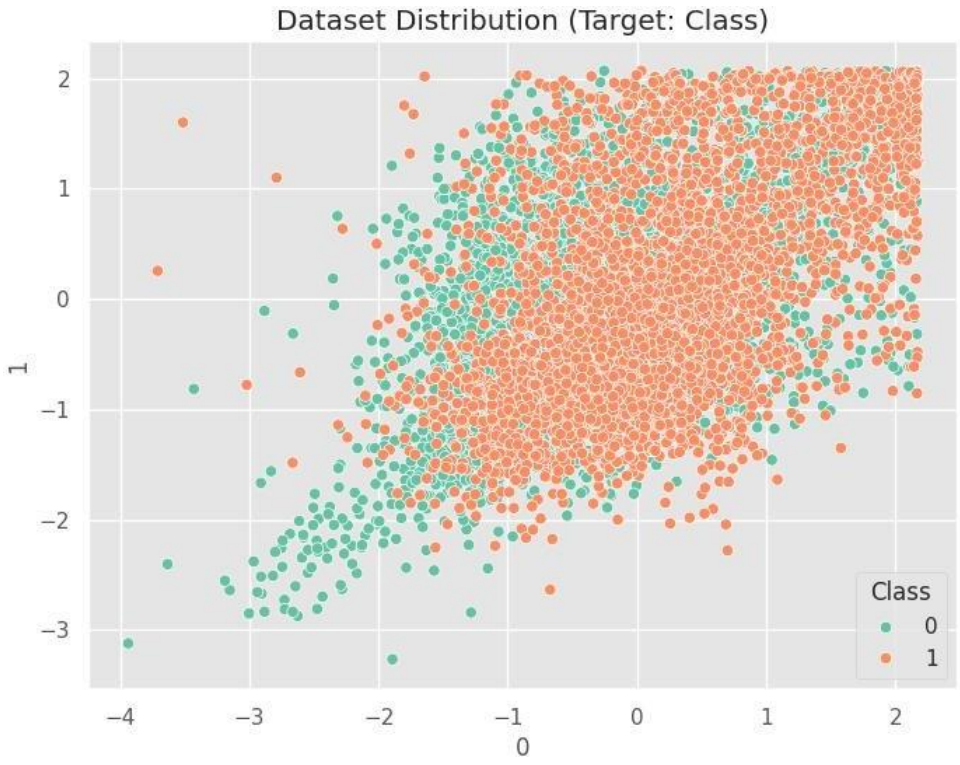


7.1.8.2 Balanced Data Of Year 2

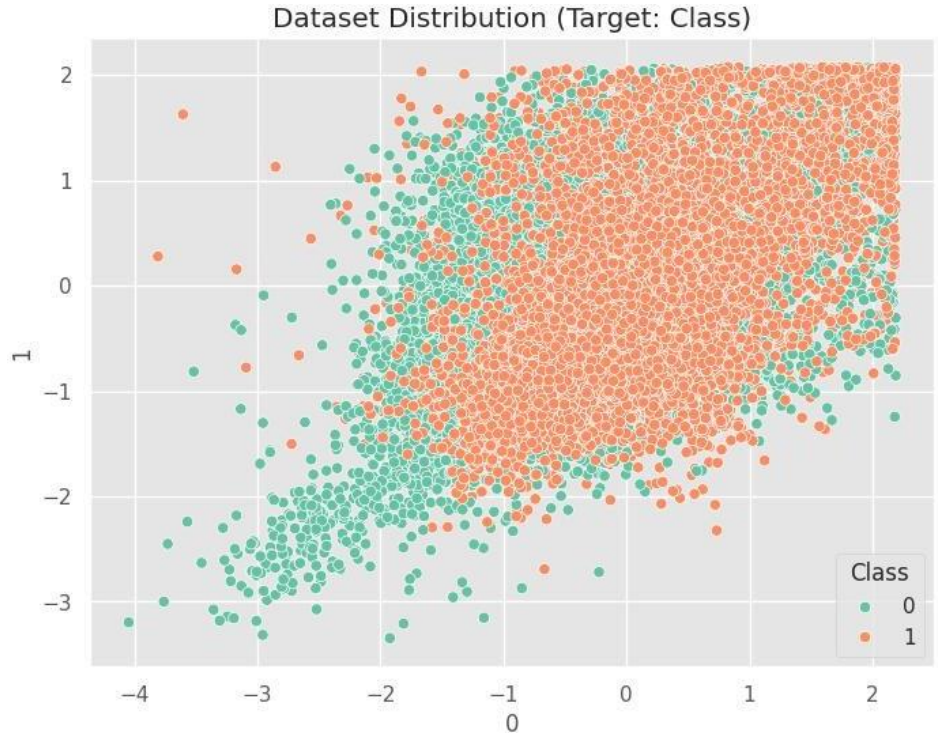
Randomly Oversampled Data



Randomly Undersampled Data

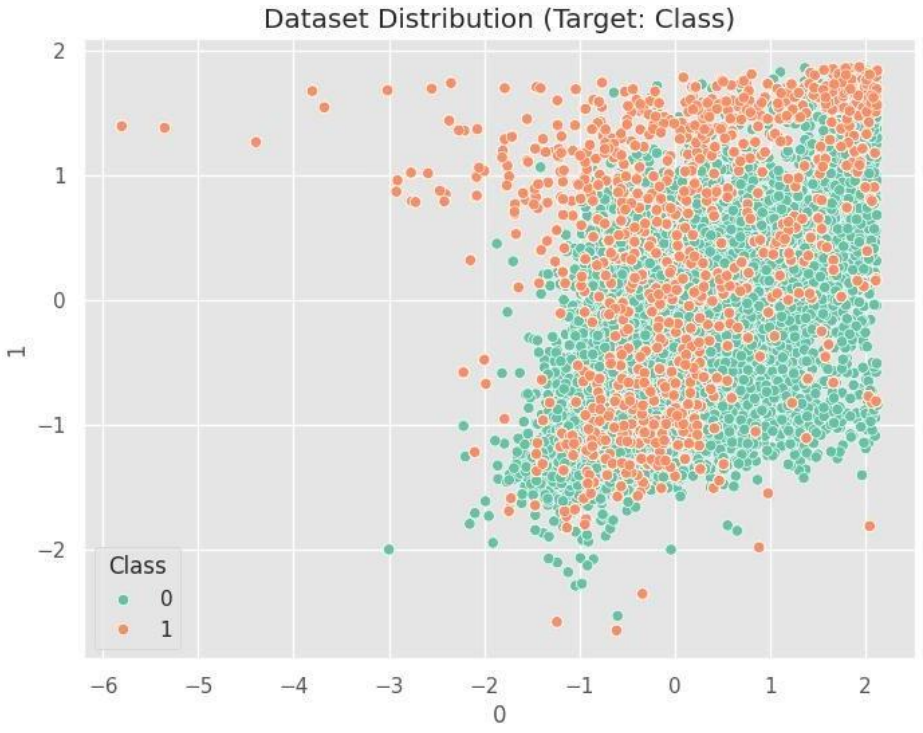

Dataset Distribution (Target: Class)
Class
0
1
0
1

SMOTE Data

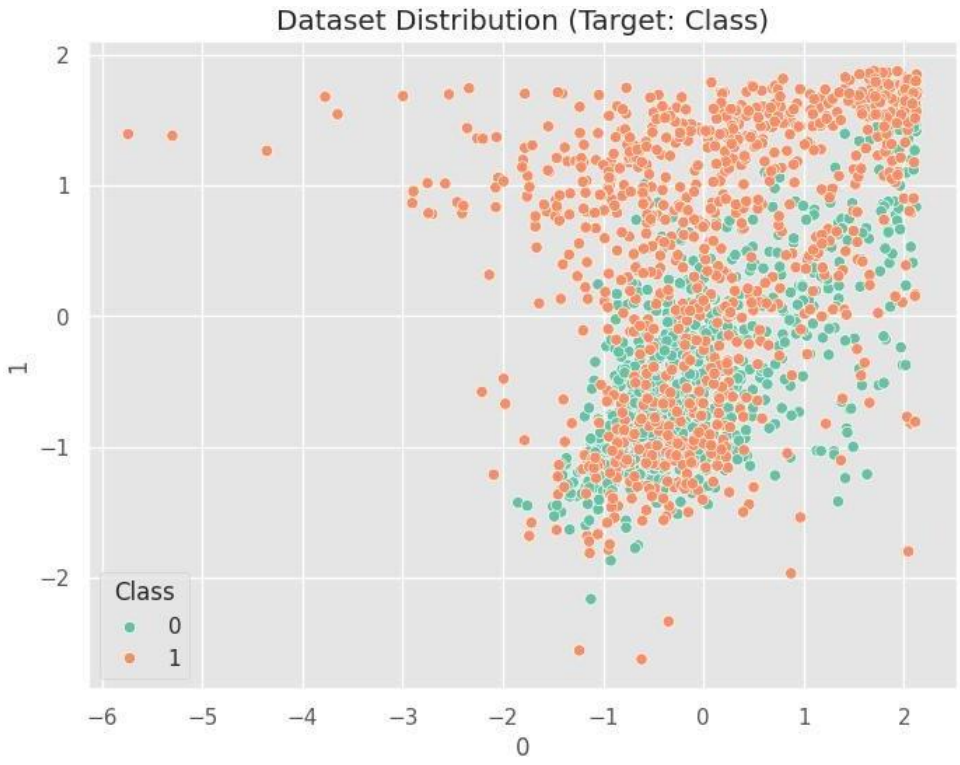

Dataset Distribution (Target: Class)
Class
0
1
0
1

7.1.8.3 Balanced Data Of Year 3

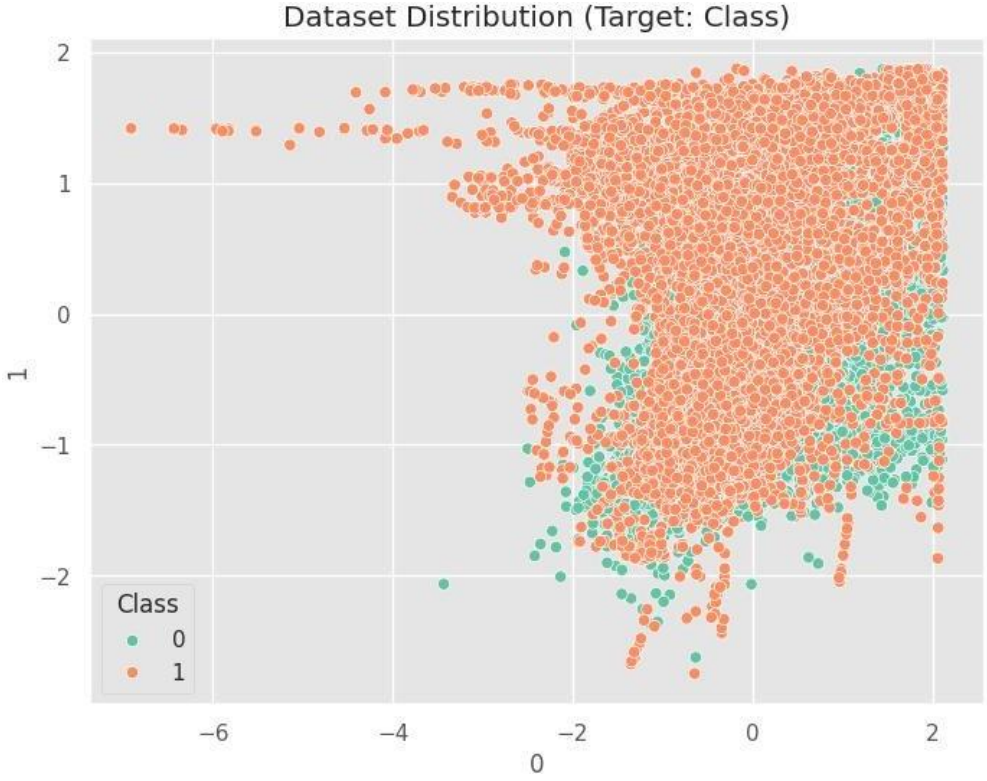
Randomly Oversampled Data



Randomly Undersampled Data


Dataset Distribution (Target: Class)
Class
0
1
0
1

SMOTE Data


Dataset Distribution (Target: Class)
Class
0
1
0
1

8. Grid Search: Hyperparameters Tuning Of XGBoost, Bagging, and Random Forest

Hyperparameter tuning is an essential step in optimizing machine learning models to achieve the best possible performance on a given dataset. In this project, hyperparameter tuning is conducted for three powerful ensemble learning algorithms—XGBoost, Bagging (Bootstrap Aggregation), and Random Forest. These models are further evaluated for their feature selection capability based on feature importance scores. Feature importance is crucial in identifying and retaining the most relevant predictors while potentially eliminating redundant or irrelevant features.

8.1 Use of Resampling Techniques for Imbalanced Datasets

The dataset used in this study is imbalanced, which can hinder the performance of machine learning models by biasing them toward the majority class. To address this, three resampling techniques are employed:

1. **Random Oversampling**: Replicates samples from the minority class to balance the class distribution.
2. **Random Undersampling**: Removes samples from the majority class to balance the class distribution.
3. **SMOTE (Synthetic Minority Oversampling Technique)**: Synthesizes new samples for the minority class by interpolating between existing instances.

8.2 Train-Test Split Approach

The temporal nature of the data necessitates splitting into training and testing sets based on distinct time periods to ensure the model's ability to generalize across different time spans. Three configurations are explored:

1. Train on Year 1 and Year 2, Test on Year 3.
2. Train on Year 2 and Year 3, Test on Year 1.
3. Train on Year 3 and Year 1, Test on Year 2.

These splits ensure robust evaluation of the models over varying temporal conditions.

8.3 Model-Specific Hyperparameter Tuning

1. **XGBoost**: XGBoost (Extreme Gradient Boosting) is a highly efficient and flexible implementation of gradient boosting. Grid search is applied to tune hyperparameters such as:
    - Learning rate
    - Maximum depth of trees
    - Subsample ratio
    - Number of estimators

    The tuned model is trained on the merged datasets (e.g., Year 1 and Year 2) and tested on the designated test set (e.g., Year 3). Performance metrics such as accuracy, precision, recall, and F1-score are calculated to evaluate the model.

2. **Bagging**: Bagging enhances model stability and accuracy by training multiple instances of a weak learner (e.g., Decision Trees) on bootstrapped subsets of the data. Key hyperparameters tuned include:
    - Number of base estimators
    - Maximum features for each estimator
    - Maximum samples for bootstrapping

    The tuned bagging model is evaluated in a similar fashion to XGBoost, with feature importance scores derived from the aggregate model.

3. **Random Forest**: Random Forest combines the strengths of bagging and decision trees by introducing randomness in feature selection for tree splits. Hyperparameters tuned for Random Forest include:
    - Number of trees in the forest
    - Maximum depth of trees
    - Minimum samples required for splits
    - Number of features considered for splits

The importance of each feature is assessed using mean decrease impurity or permutation importance.

8.4 Feature Importance-Based Selection

The feature importance scores derived from each model provide insights into the relative contributions of predictors to the model's performance. This allows for feature selection, which is critical in improving model interpretability and reducing computational cost. The selected features can be used in subsequent iterations to retrain and further refine the models.

8.5 Key Contributions of Resampling Techniques

Each resampling technique (Random Oversampling, Random Undersampling, and SMOTE) is separately applied to ensure fair comparisons and assess their impact on:

1. Model performance across different time splits.
2. Stability and reliability of feature importance rankings.

By using resampling techniques in conjunction with robust hyperparameter tuning, this approach ensures models are both effective in handling imbalanced data and capable of identifying the most relevant features.

9. Summarizing Results Of Grid Search

9.1 Random Oversampling

9.1.1 XGBoost

9.1.1.1 Model Results for XGBoost

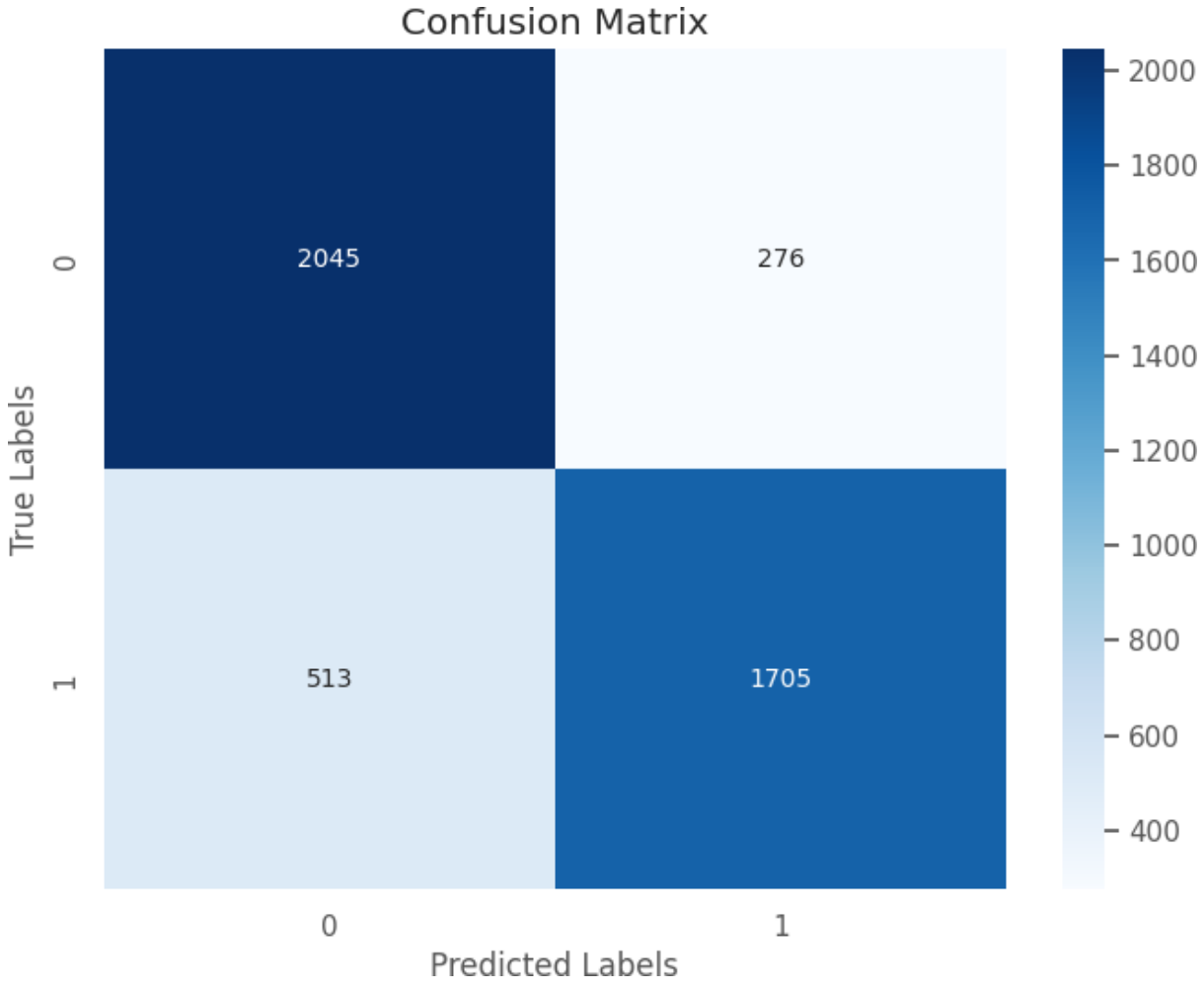
Train Years	Test Year	Best Parameters	Precision		Recall		F1-Score		Accuracy
			0	1	0	1	0	1	
Year1 + Year 2	Year3	Learning rate: 0.1 Max_length: 10 N_estimators: 300	0.80	0.86	0.88	0.77	0.84	0.81	0.83
Year2 + Year 3	Year1	Learning rate: 0.2 Max_length: 10 N_estimators: 300	0.73	0.83	0.90	0.59	0.80	0.69	0.76
Year1 + Year 3	Year2	Learning rate: 0.2 Max_length: 10 N_estimators: 200	0.61	0.82	0.93	0.35	0.74	0.49	0.66

9.1.1.2 Feature Importance(features range from 0 to 11)

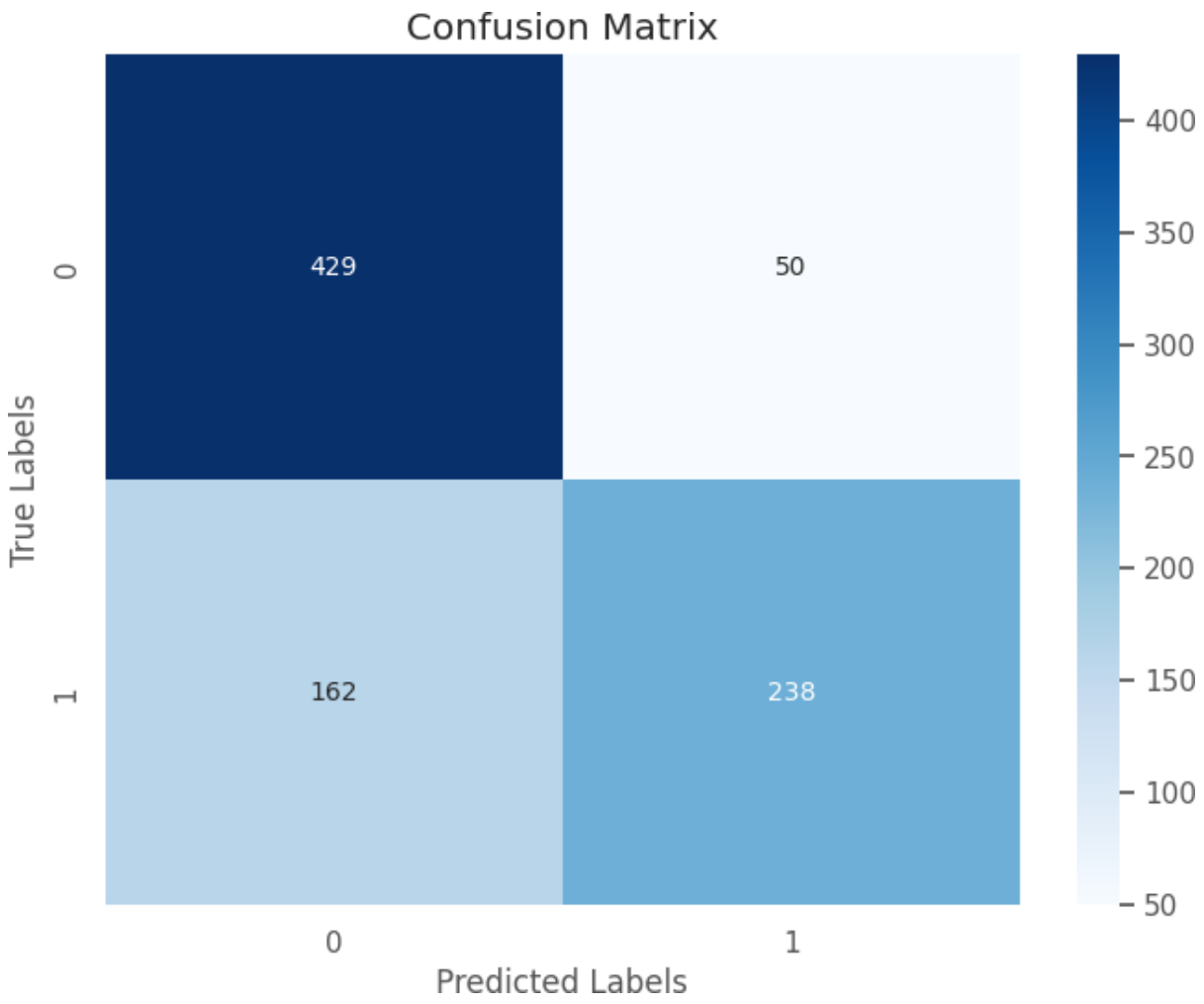
Rank	Year1 + Year 2 -> Year3	Year2 + Year3 -> Year1	Year1 + Year3 -> Year2
1	4	4	8
2	5	6	5
3	7	5	11
4	10	7	7
5	6	11	0
6	3	10	6
7	11	8	4
8	8	3	3
9	2	0	9
10	0	9	1
11	9	1	2
12	1	2	10

9.1.1.3 Confusion Matrix

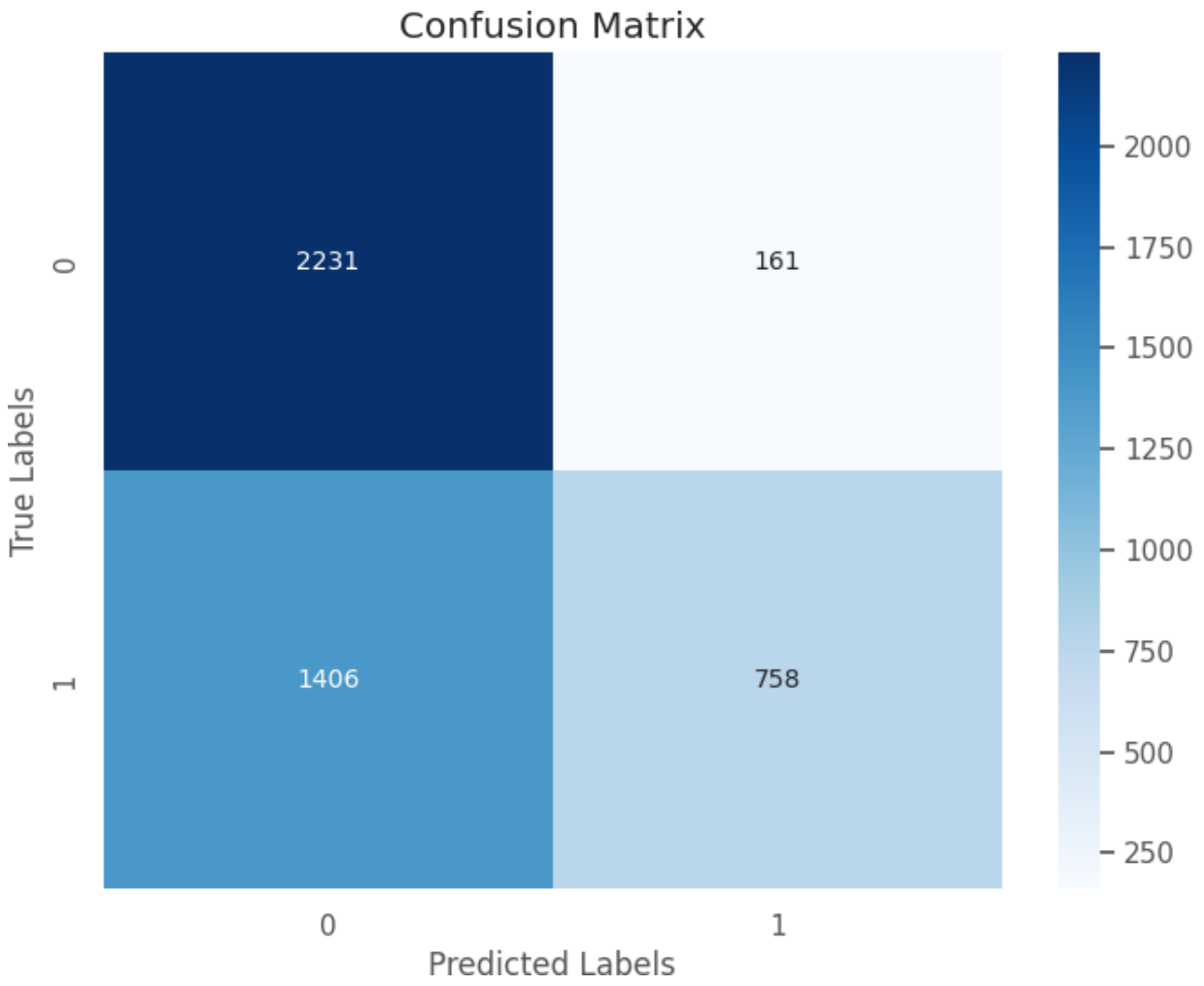
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



9.1.2 Bagging(Bootstrap Aggregation)

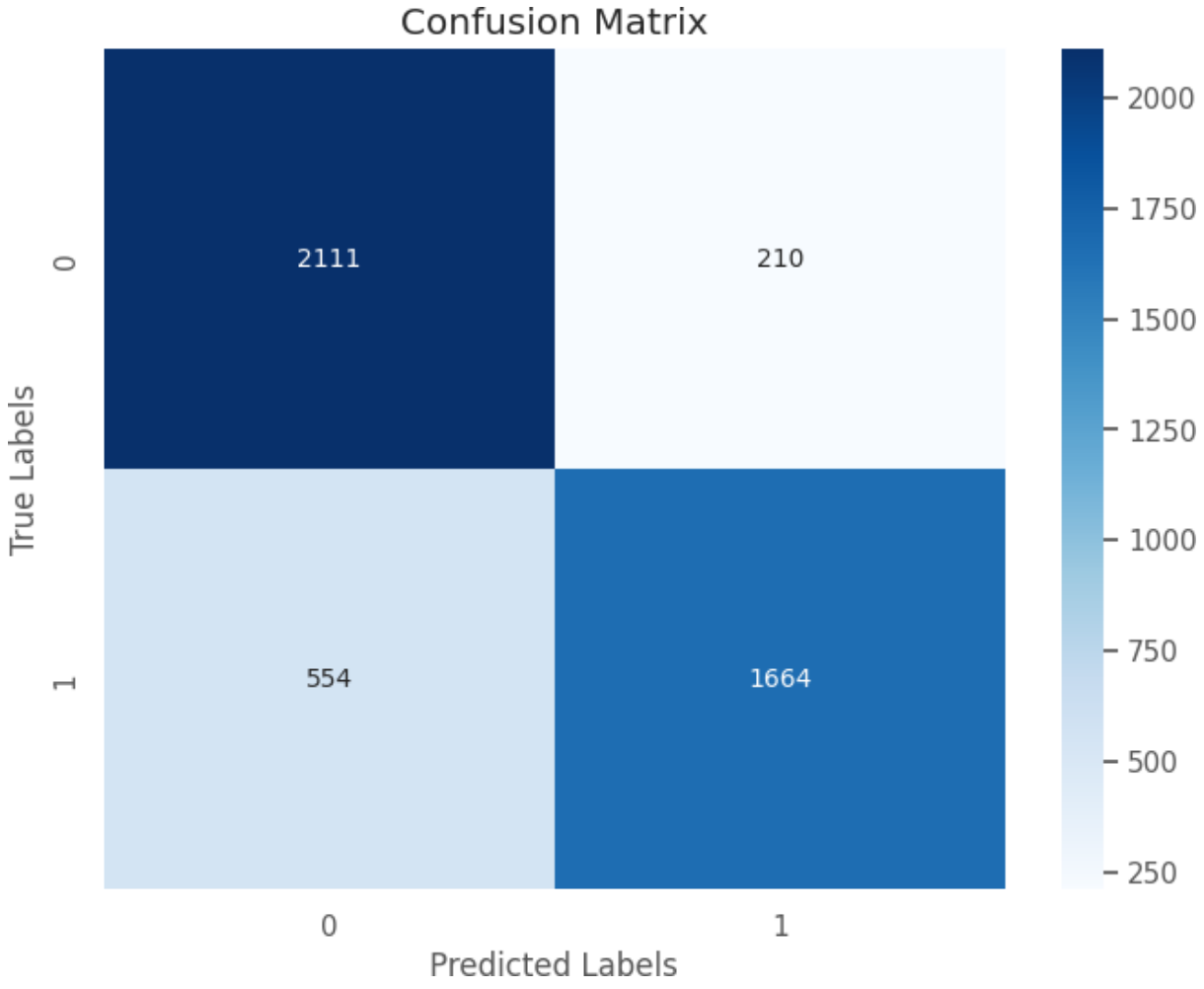
9.1.2.1 Model Results

Train Years	Test Year	Best Parameters	Precision		Recall		F1-Score		Accuracy
			0	1	0	1	0	1	
Year1 + Year 2	Year3	estimator_max_depth: None Max_samples: 1.0 Max_features: 0.7	0.79	0.89	0.91	0.75	0.85	0.81	0.83

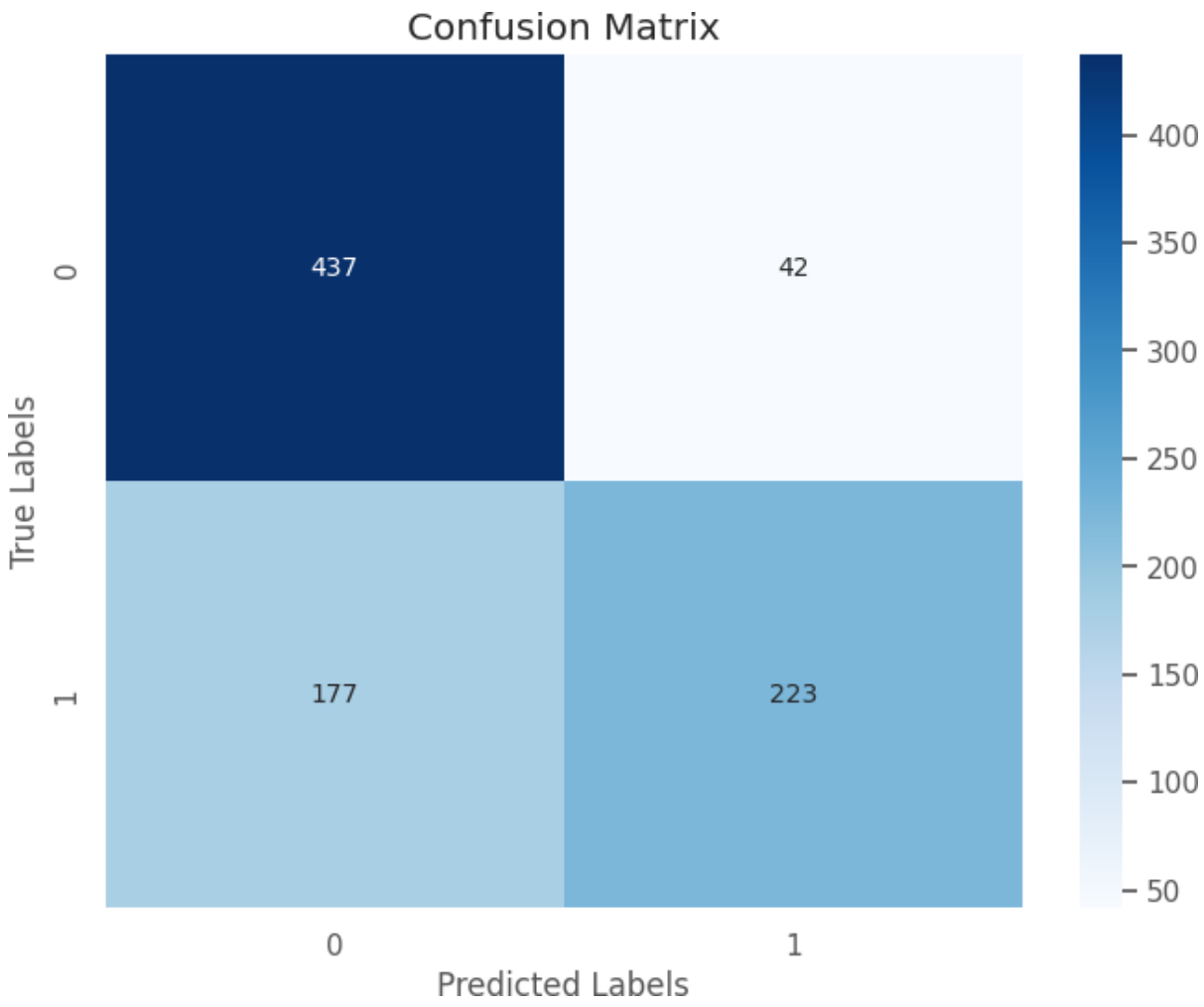
		N_estimators: 100							
Year2 + Year 3	Year1	estimator_max_depth: None Max_samples: 1.0 Max_features: 0.7 N_estimators: 100	0.71	0.84	0.91	0.56	0.80	0.67	0.75
Year1 + Year 3	Year2	estimator_max_depth: None Max_samples: 0.7 Max_features: 0.5 N_estimators: 100	0.62	0.92	0.96	0.42	0.75	0.58	0.69

9.1.2.2 Confusion Matrix

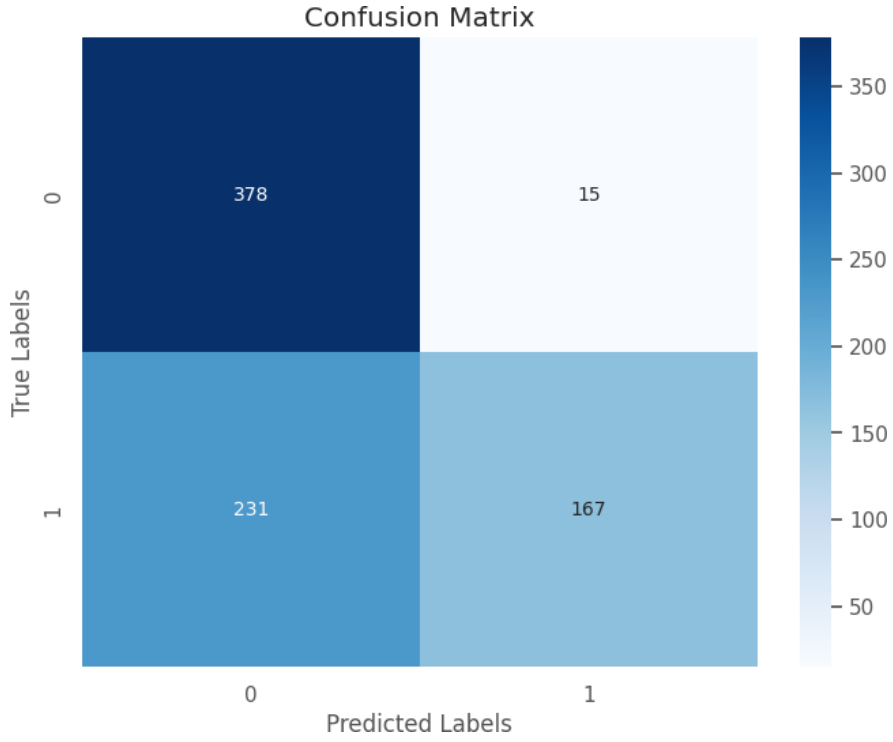
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



9.1.3 Random Forest

9.1.3.1 Model Results

Train Years	Test Year	Best Parameters	Precision		Recall		F1-Score		Accuracy
			0	1	0	1	0	1	
Year1 + Year 2	Year3	Max_depth: None Min_samples_leaf: 1 N_estimators: 200	0.80	0.89	0.91	0.76	0.85	0.82	0.84
Year2 + Year 3	Year1	Max_depth: None Min_samples_leaf: 1 N_estimators: 100	0.71	0.85	0.92	0.55	0.80	0.67	0.75
Year1 + Year 3	Year2	Max_depth: None	0.61	0.87	0.96	0.33	0.75	0.48	0.66

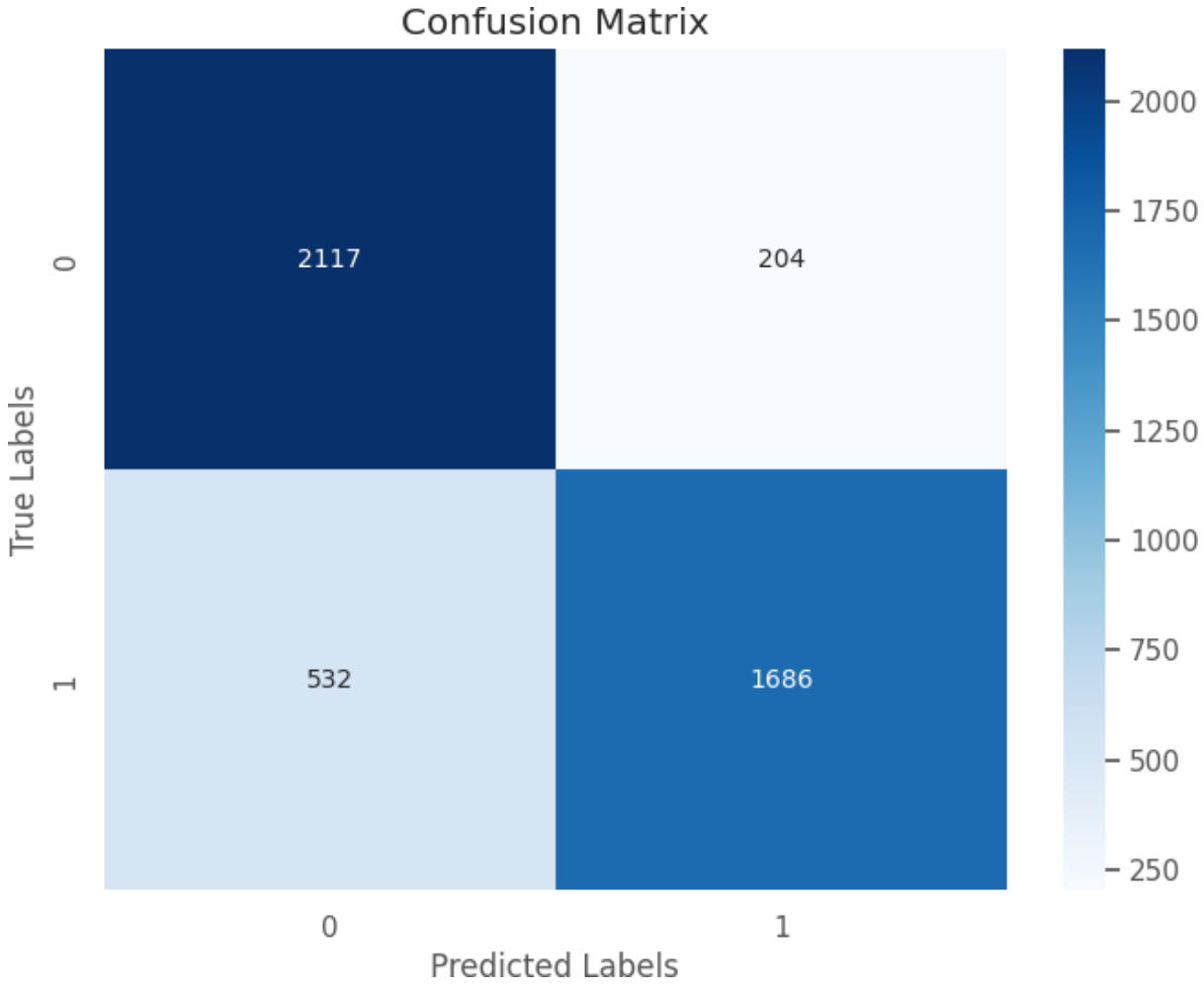
		Min_samples_leaf: 2 N_estimators: 100							

9.1.3.2 Feature Importance

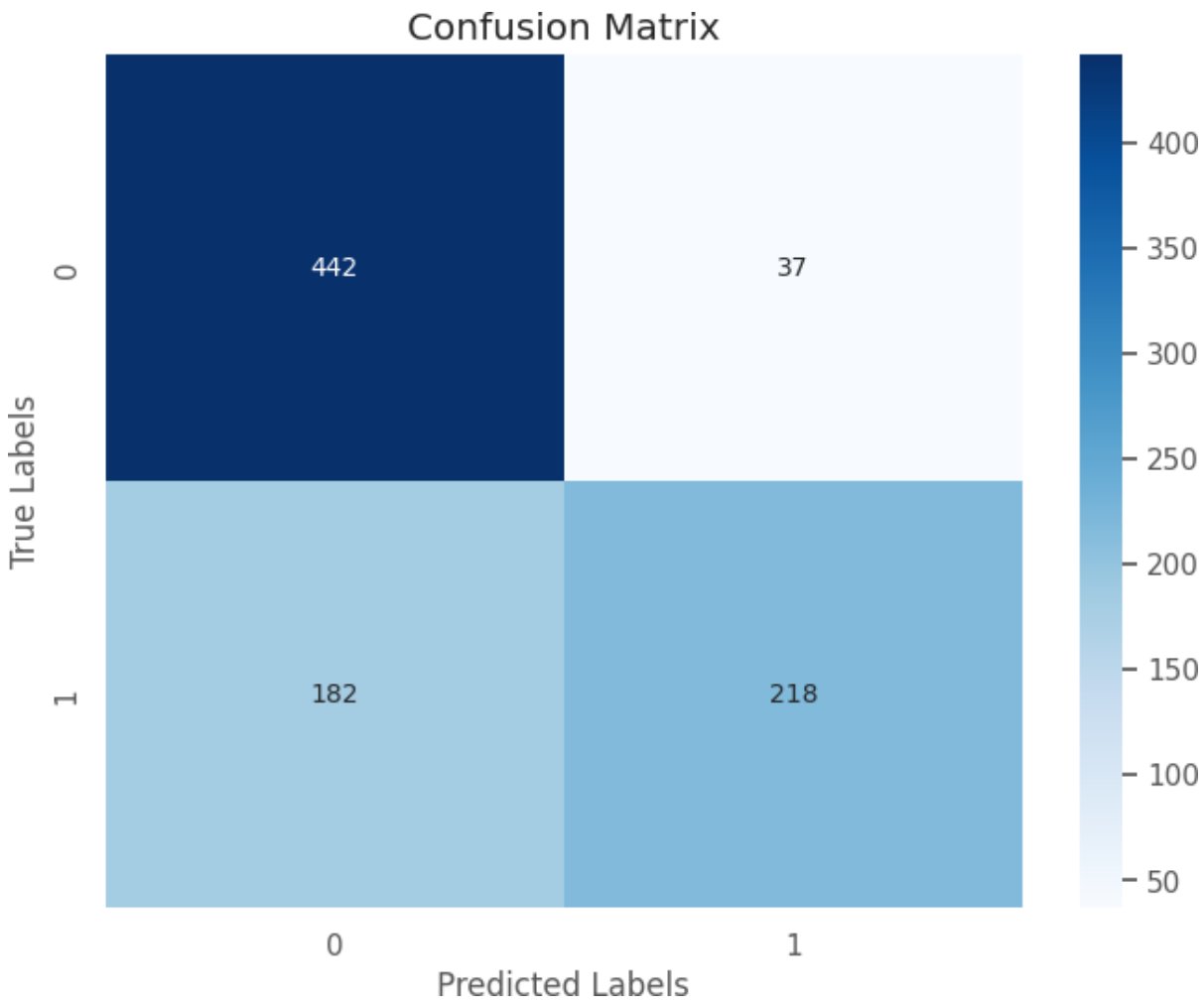
Rank	**Year1 + Year 2 -> Year3**	**Year2 + Year3 -> Year1**	**Year1 + Year3 -> Year2**
1	10	10	10
2	9	9	9
3	11	11	3
4	4	8	7
5	8	7	2
6	3	2	4
7	0	4	8
8	2	6	6
9	6	3	1
10	7	1	11
11	5	0	0
12	1	5	5

9.1.3.3 Confusion Matrix

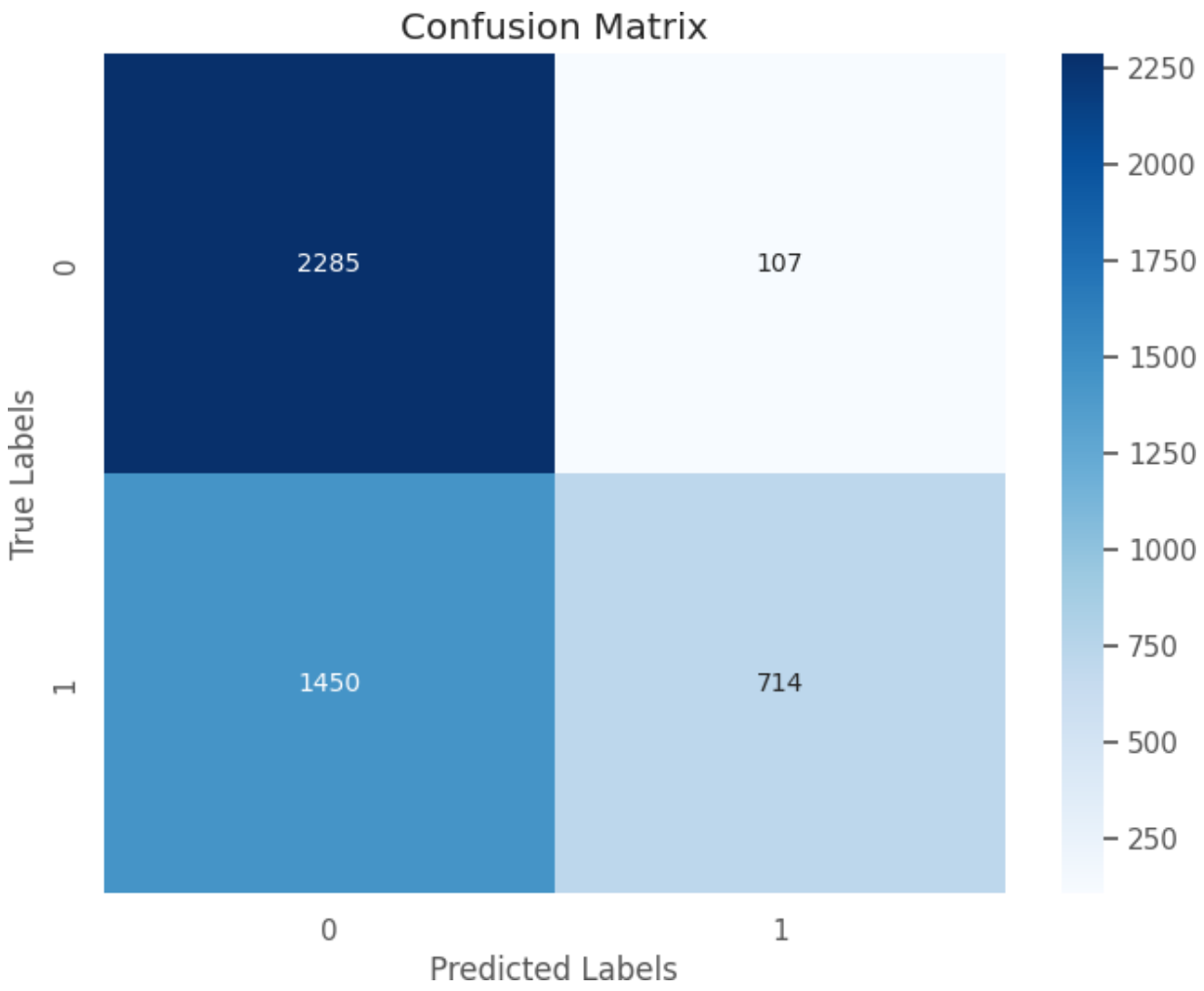
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



9.1.4 SVM

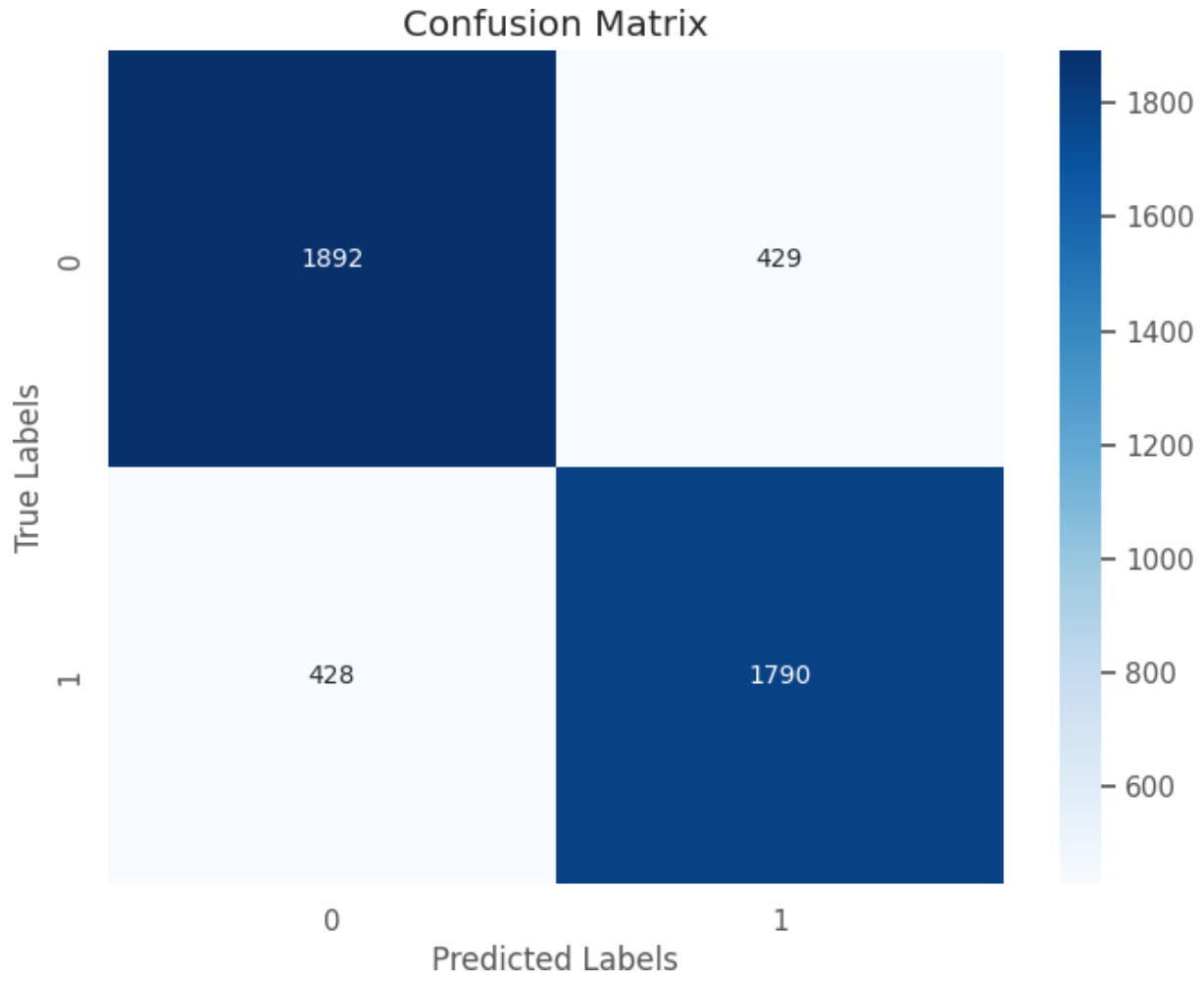
9.1.4.1 Model Results

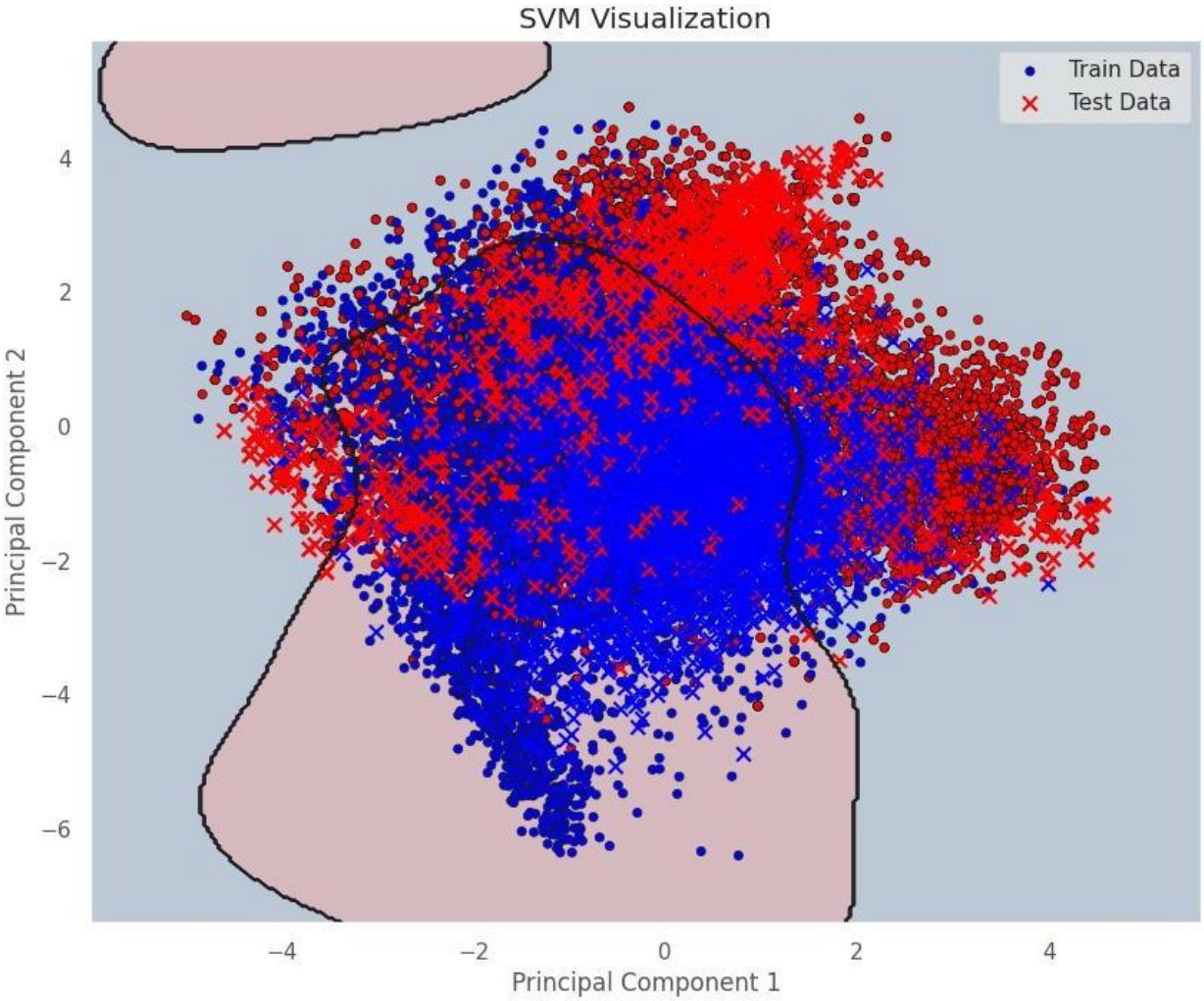
Train Years	Test Year	Best Parameters	Precision		Recall		F1-Score		Accuracy
			0	1	0	1	0	1	
Year1 + Year 2	Year3	'C': 10 'gamma': 'scale', 'kernel': 'rbf'	0.82	0.81	0.82	0.81	0.82	0.81	0.81
Year2 + Year 3	Year1	'C': 10	0.78	0.81	0.86	0.70	0.82	0.75	0.79

		'gamma': 'scale', 'kernel': 'rbf'							
Year1 + Year 3	Year2	'C': 10 'gamma': 'auto', 'kernel': 'rbf'	0.63	0.76	0.83	0.52	0.72	0.62	0.68

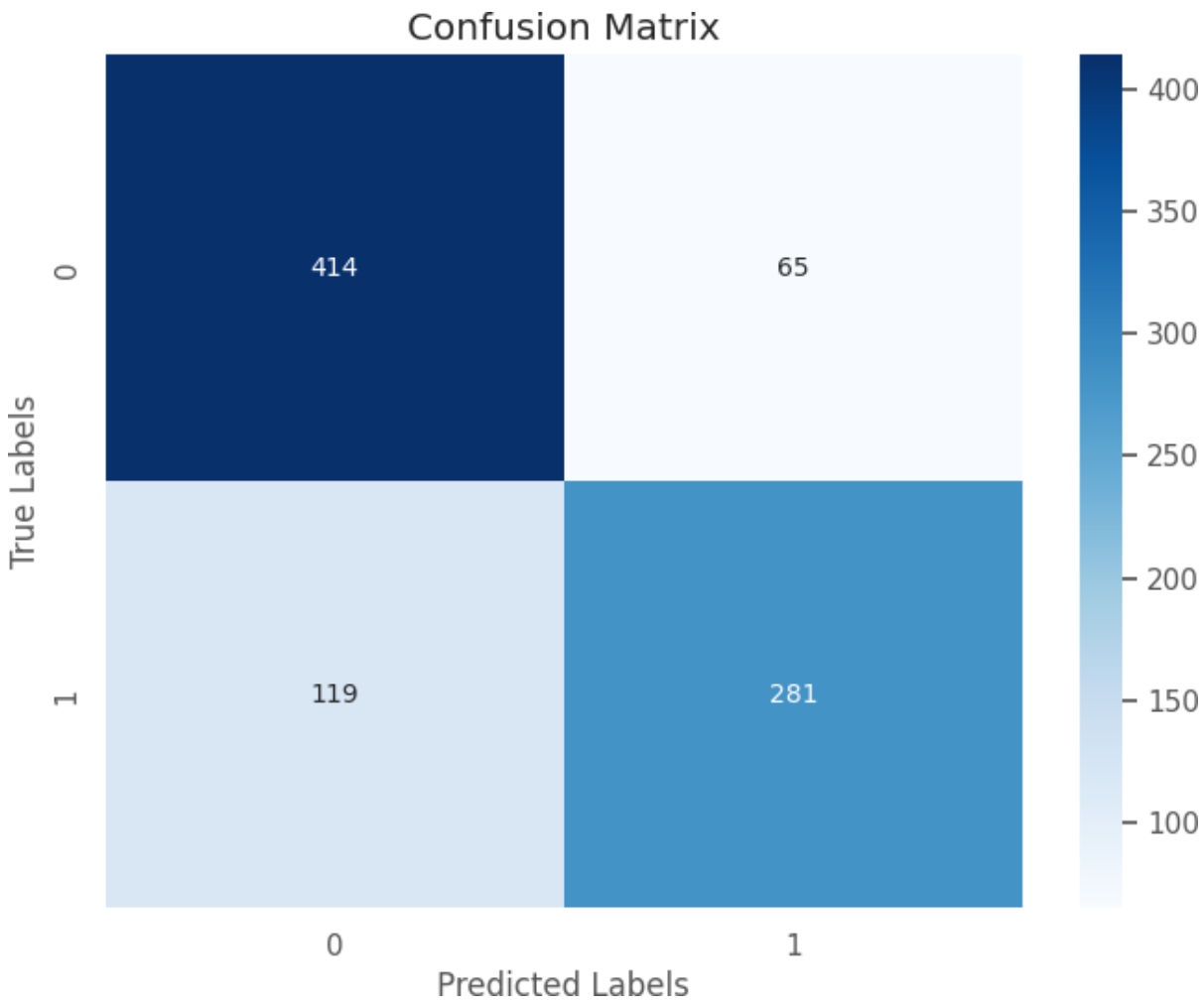
9.1.4.2 Confusion Matrix

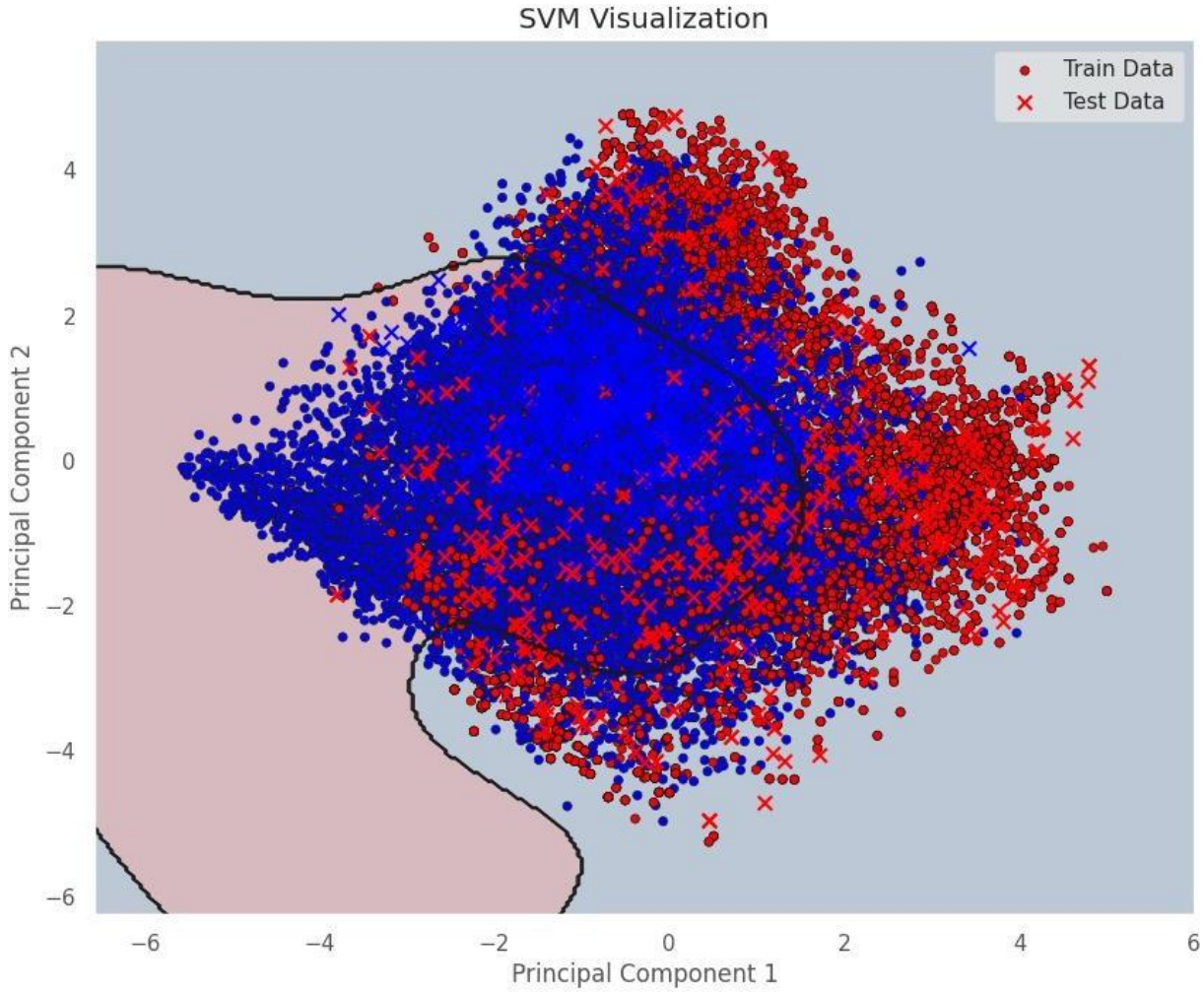
Year 3 As Testing Year



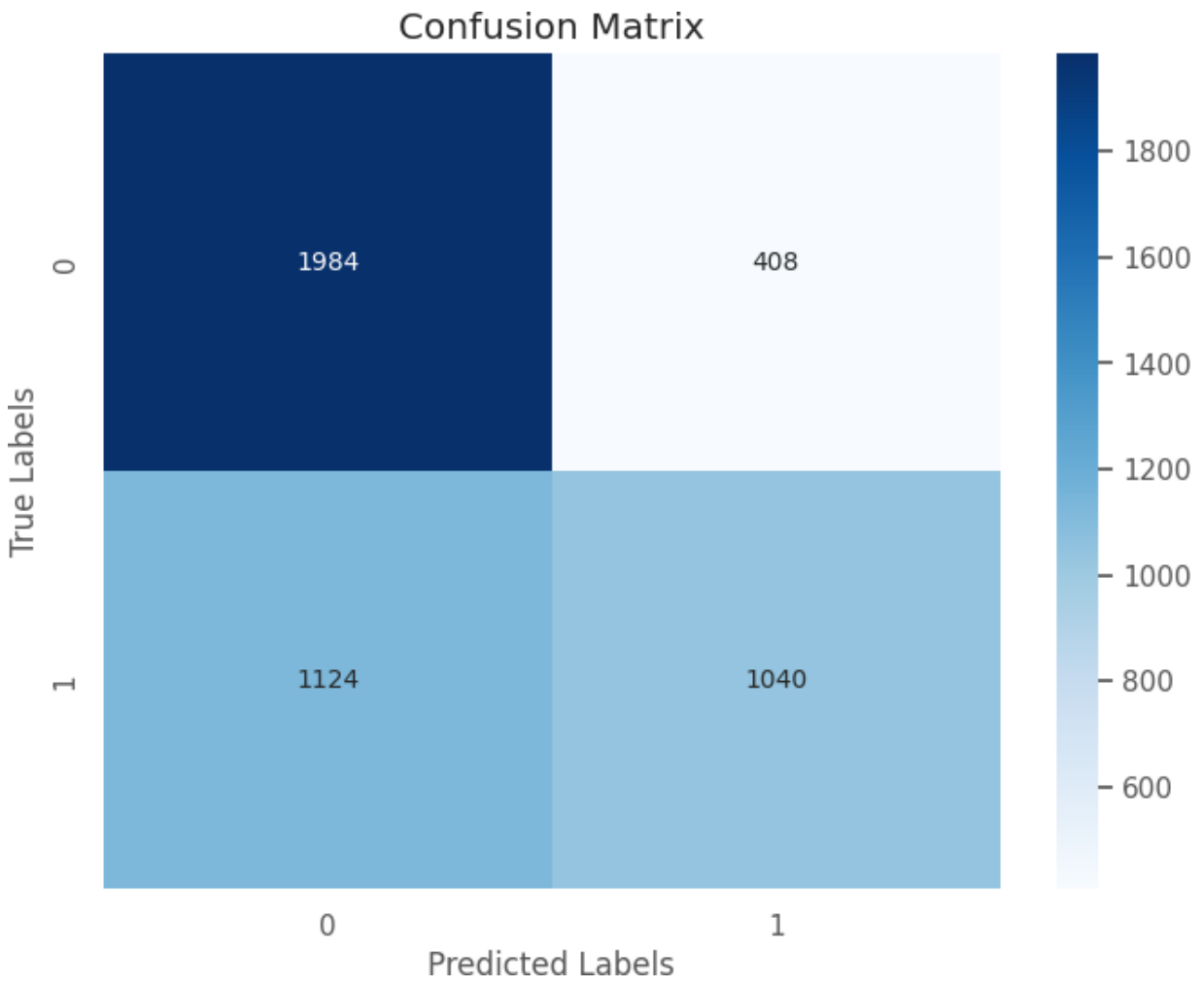

SVM Visualization
Train Data
Test Data
Principal Component 2
Principal Component 1
4
2
0
−2
−4
−6
−4
−2
0
2
4

Year 1 As Testing Year




SVM Visualization
Train Data
Test Data
Principal Component 2
Principal Component 1
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Year 2 As Testing Year



9.2 Random Undersampling

9.2.1 XGBoost

9.2.1.1 Model Results for XGBoost

Train Years	Test Year	Best Parameters	Precision	Recall	F1-Score	Accuracy

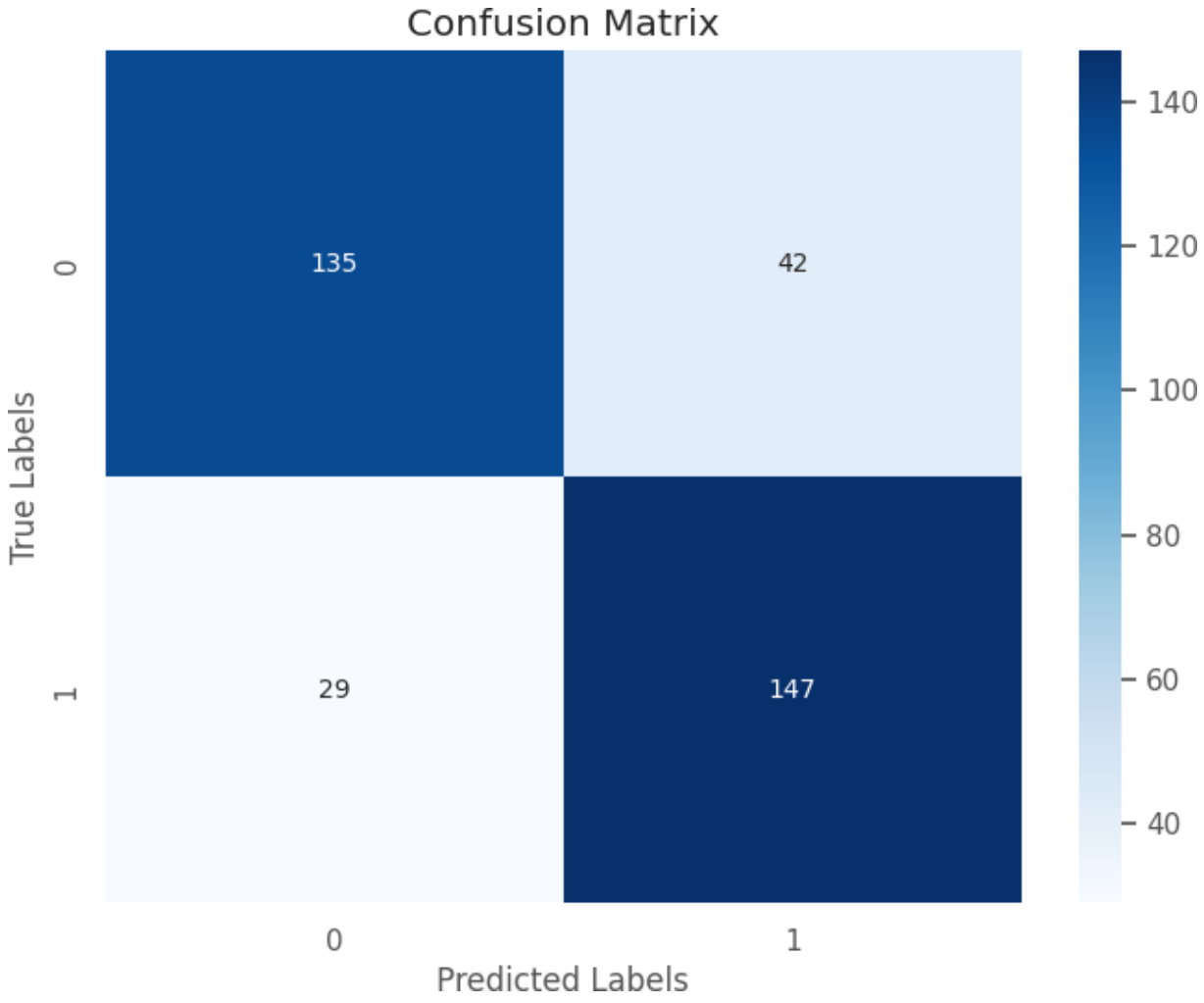
			0	**1**	**0**	**1**	**0**	**1**	
Year1 + Year 2	Year3	Learning rate: 0.1 Max_length: 7 N_estimators: 300	0.82	0.78	0.76	0.84	0.79	0.81	0.80
Year2 + Year 3	Year1	Learning rate: 0.1 Max_length: 5 N_estimators: 300	0.85	0.87	0.88	0.84	0.87	0.87	0.86
Year1 + Year 3	Year2	Learning rate: 0.05 Max_length: 10 N_estimators: 200	0.77	0.73	0.72	0.78	0.74	0.75	0.75

9.2.1.2 Feature Importance(features range from 0 to 11)

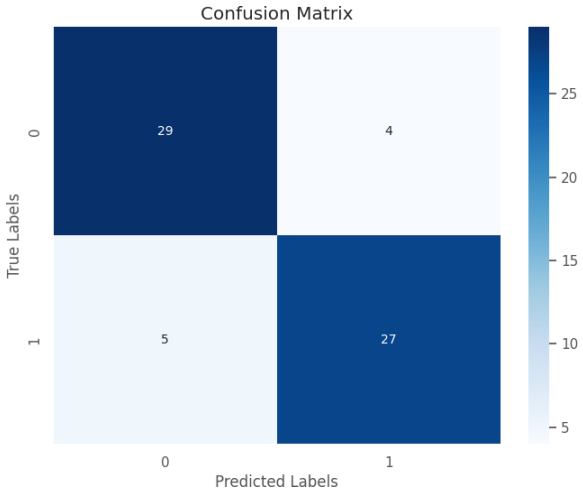
Rank	**Year1 + Year 2 -> Year3**	**Year2 + Year3 -> Year1**	**Year1 + Year3 -> Year2**
1	7	3	10
2	6	10	4
3	4	4	3
4	5	7	11
5	3	11	9
6	11	2	8
7	2	6	6
8	0	5	2
9	8	9	5
10	10	8	7
11	9	0	1
12	1	1	0

9.2.1.3 Confusion Matrix

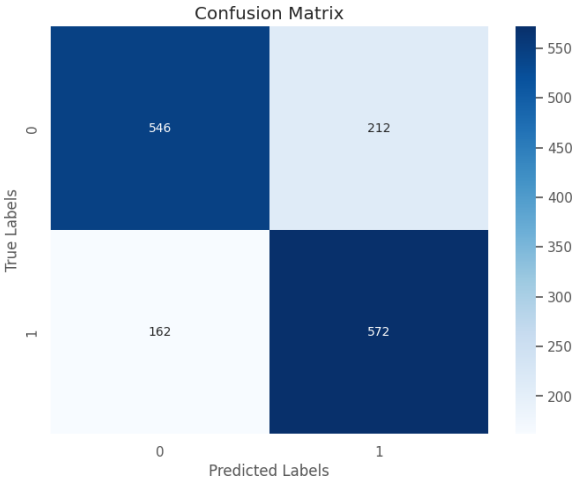
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



9.2.2 Bagging(Bootstrap Aggregation)

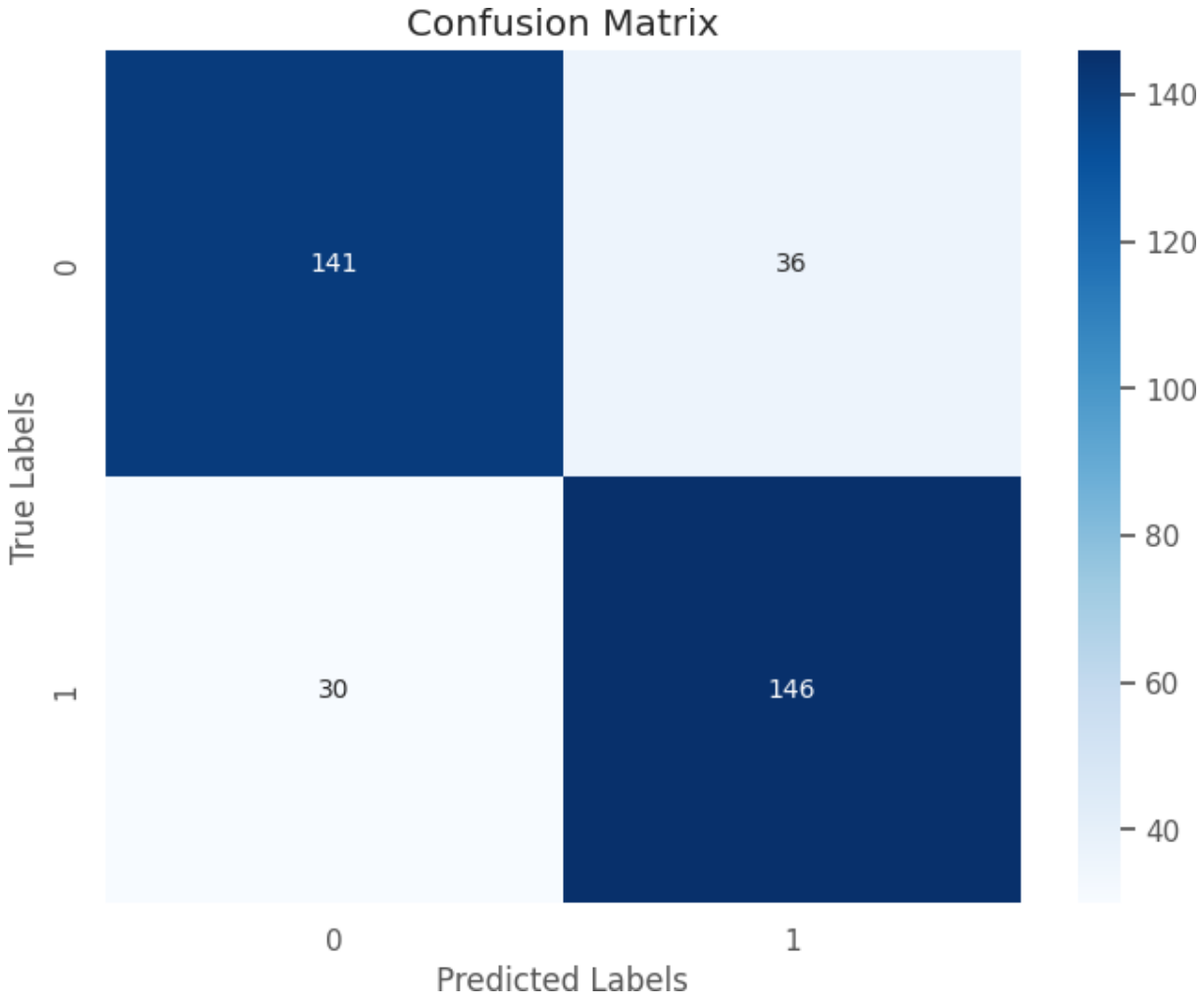
9.2.2.1 Model Results

Train Years	Test Year	Best Parameters	Precision		Recall		F1-Score		Accuracy
			0	1	0	1	0	1	
Year1 + Year 2	Year3	estimator_max_depth: None	0.82	0.80	0.80	0.83	0.81	0.82	0.81

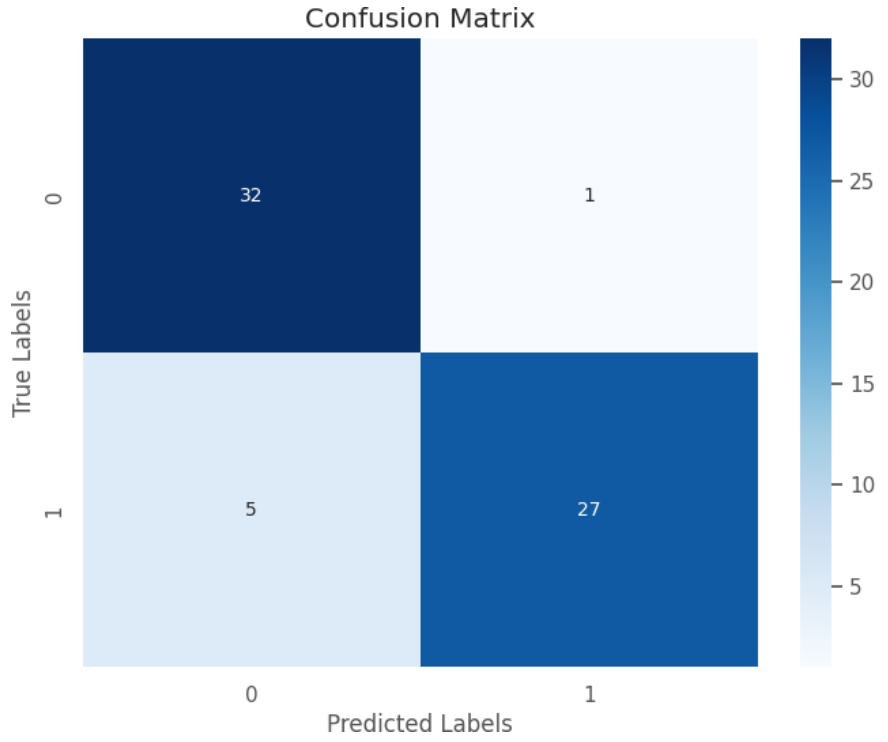
		Max_samples: 1.0 Max_features: 0.7 N_estimators: 100							
Year2 + Year 3	Year1	estimator_max_depth: None Max_samples: 0.5 Max_features: 1.0 N_estimators: 100	0.86	0.96	0.97	0.84	0.91	0.90	0.91
Year1 + Year 3	Year2	estimator_max_depth: None Max_samples: 1.0 Max_features: 0.7 N_estimators: 100	0.78	0.72	0.71	0.79	0.74	0.76	0.75

9.2.2.2 Confusion Matrix

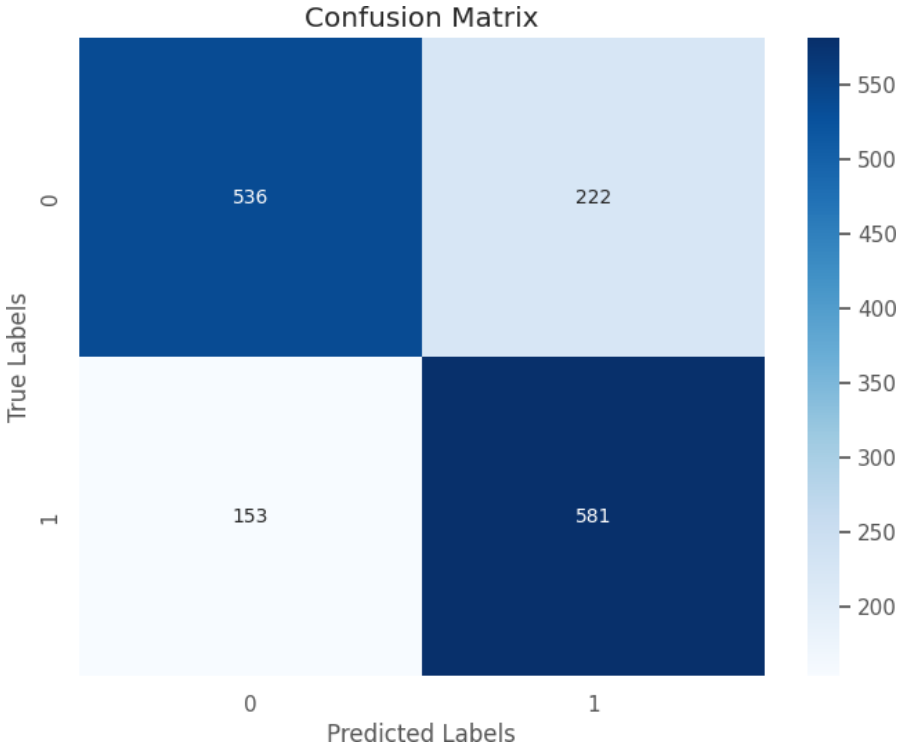
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



9.2.3 Random Forest

9.2.3.1 Model Results

Train Years	Test Year	Best Parameters	Precision		Recall		F1-Score		Accuracy
			0	1	0	1	0	1	
Year1 + Year 2	Year3	Max_depth: None Min_samples_leaf: 1 N_estimators: 100	0.84	0.81	0.80	0.85	0.82	0.83	0.82
Year2 + Year 3	Year1	Max_depth: 20 Min_samples_leaf: 1 N_estimators: 100	0.89	0.93	0.94	0.88	0.91	0.90	0.91
Year1 + Year 3	Year2	Max_depth: None Min_samples_leaf: 1 N_estimators: 200	0.79	0.72	0.70	0.81	0.74	0.76	0.75

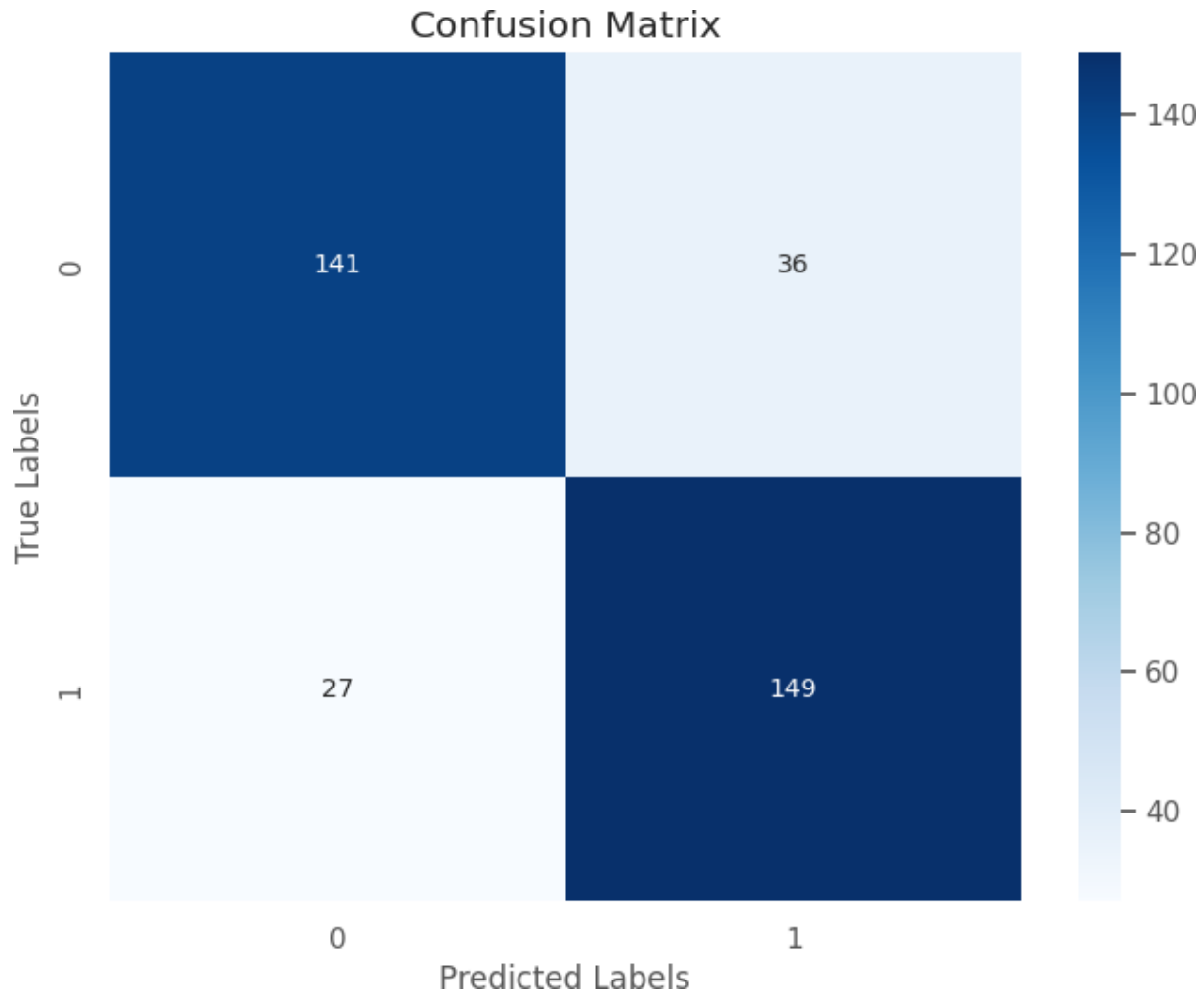
9.2.2.2 Feature Importance

Rank	Year1 + Year 2 -> Year3	Year2 + Year3 -> Year1	Year1 + Year3 -> Year2
1	10	10	10
2	9	9	9
3	11	11	3
4	8	8	2
5	4	3	7
6	3	4	4
7	2	2	8
8	0	6	11
9	6	0	6

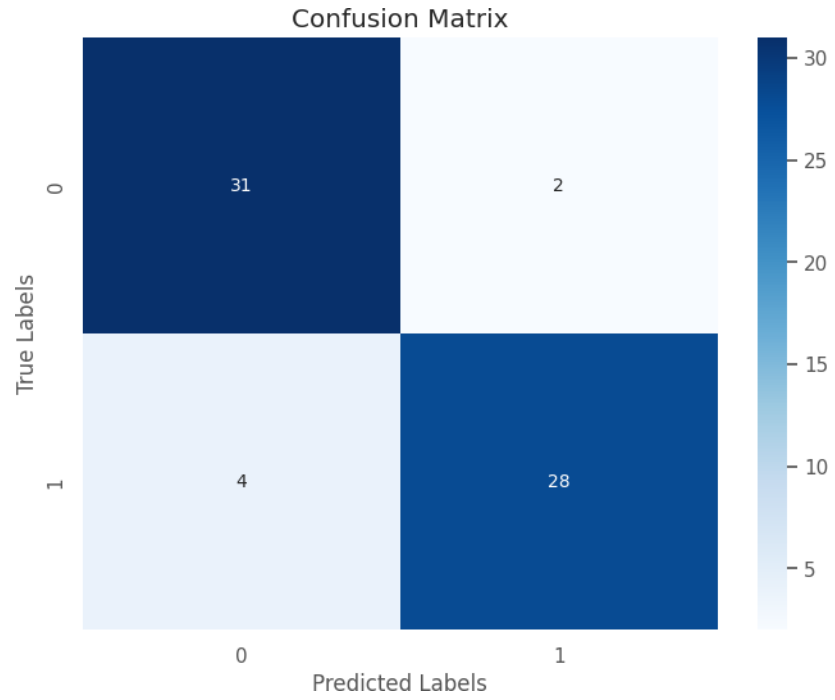
10	7	7	1
11	5	5	0
12	1	1	5

9.2.2.3 Confusion Matrix

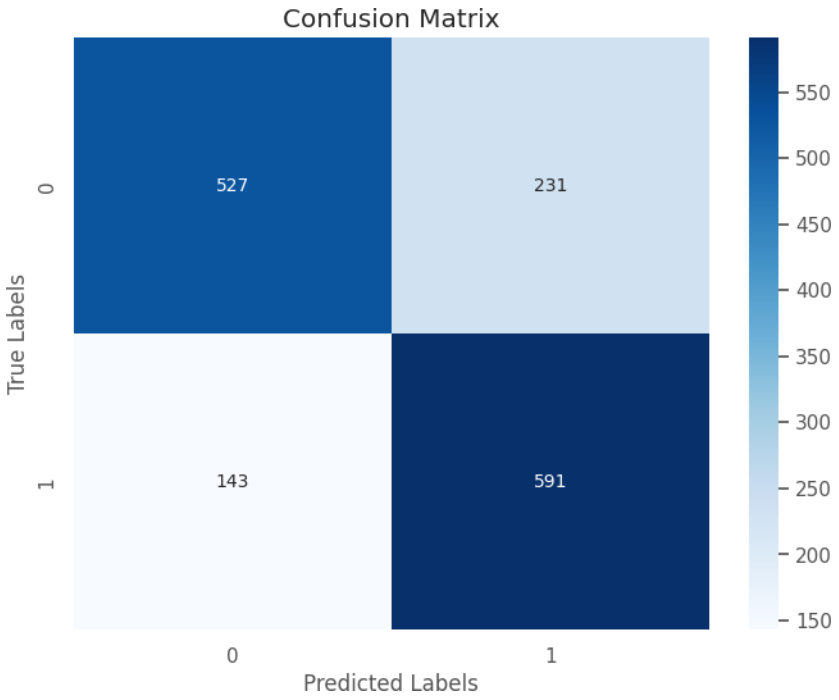
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



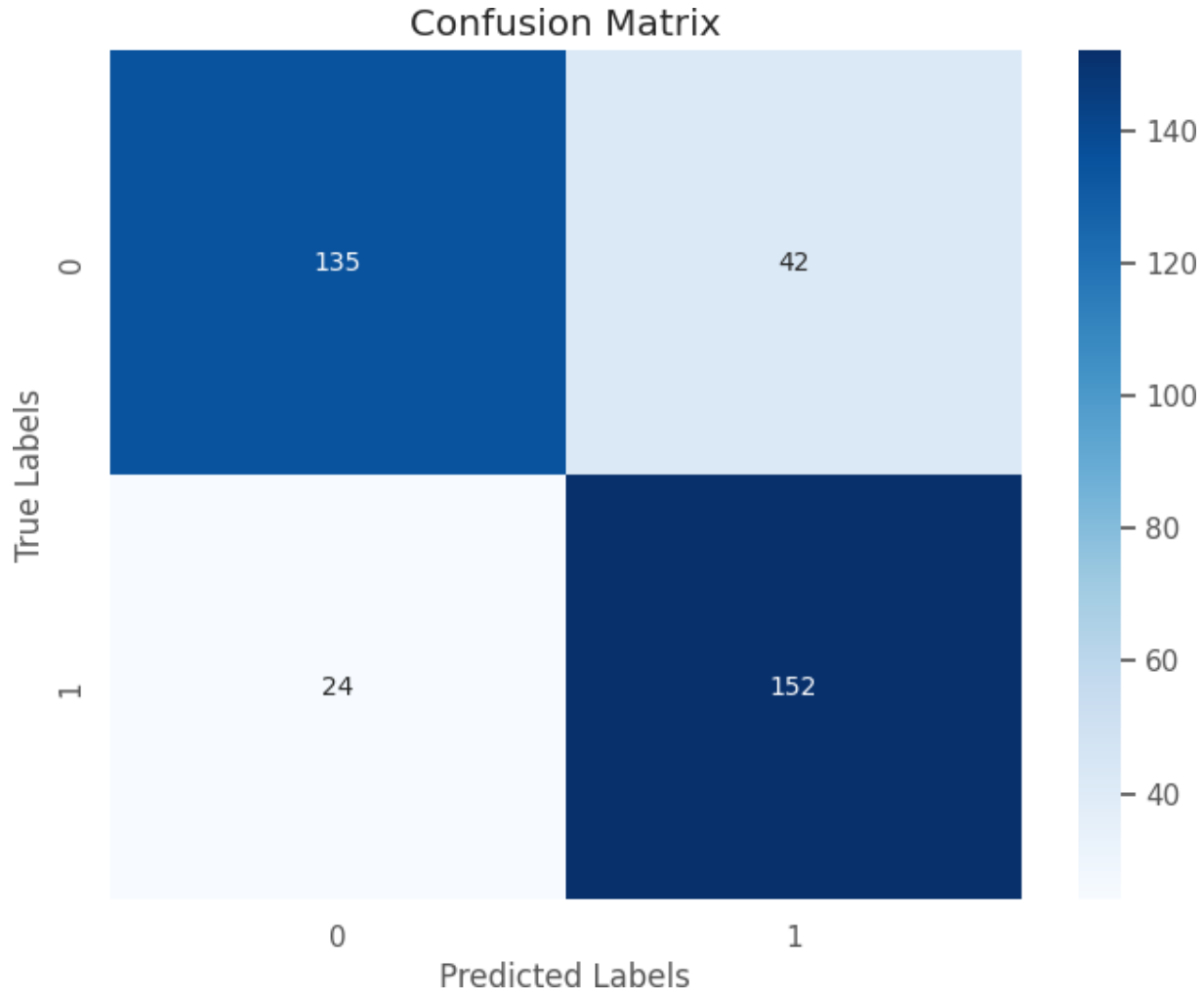
9.2.4 SVM

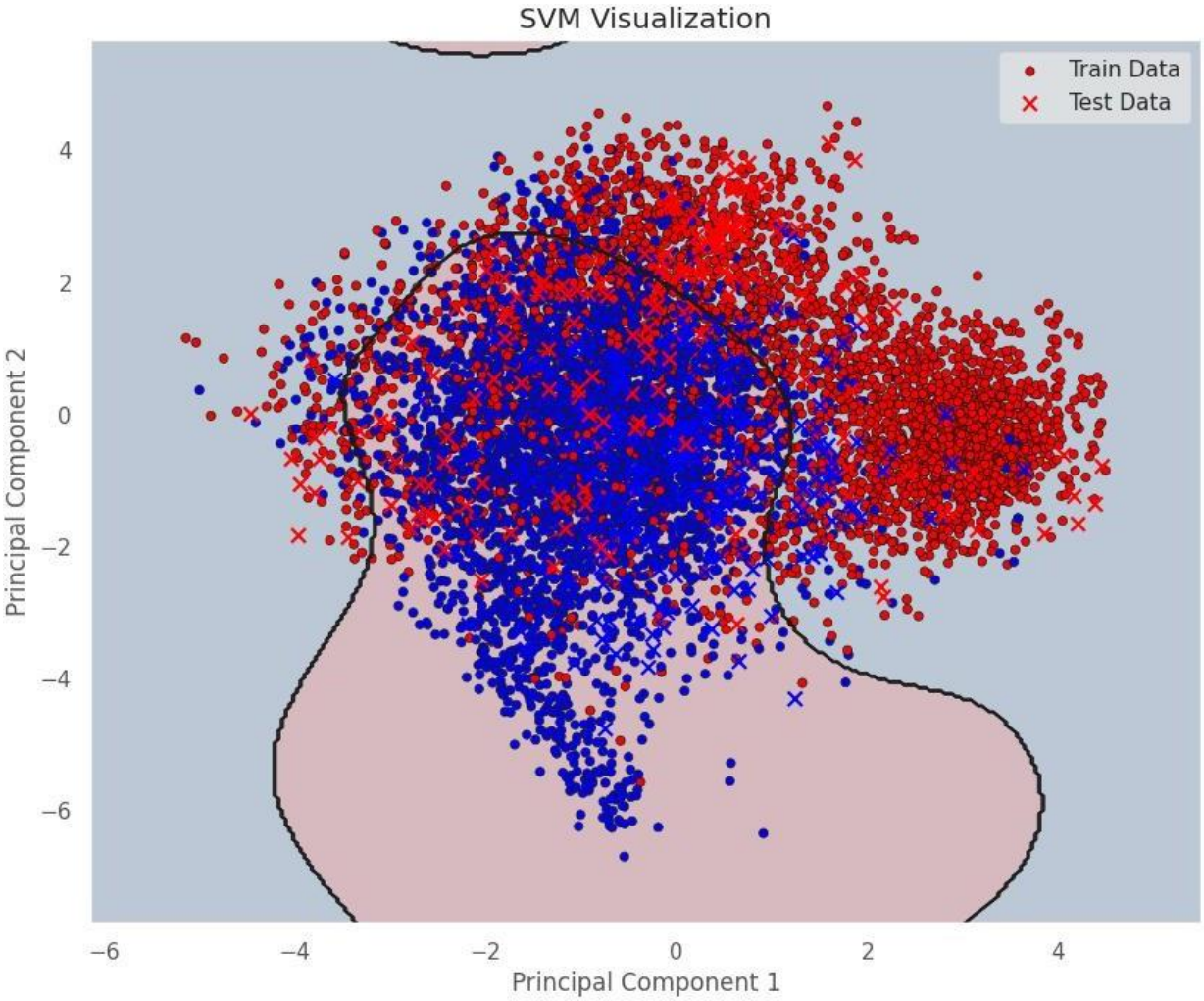
9.2.4.1 Model Results

Train Years	Test Year	Best Parameters	Precision		Recall		F1-Score		Accuracy
			0	1	0	1	0	1	
Year1 + Year 2	Year3	'C': 10 'gamma': 'scale', 'kernel': 'rbf'	0.85	0.78	0.76	0.86	0.80	0.82	0.81
Year2 + Year 3	Year1	'C': 10 'gamma': 'scale', 'kernel': 'rbf'	0.87	0.80	0.79	0.88	0.83	0.84	0.84
Year1 + Year 3	Year2	'C': 1 'gamma': 'scale', 'kernel': 'rbf'	0.76	0.75	0.76	0.75	0.76	0.75	0.75

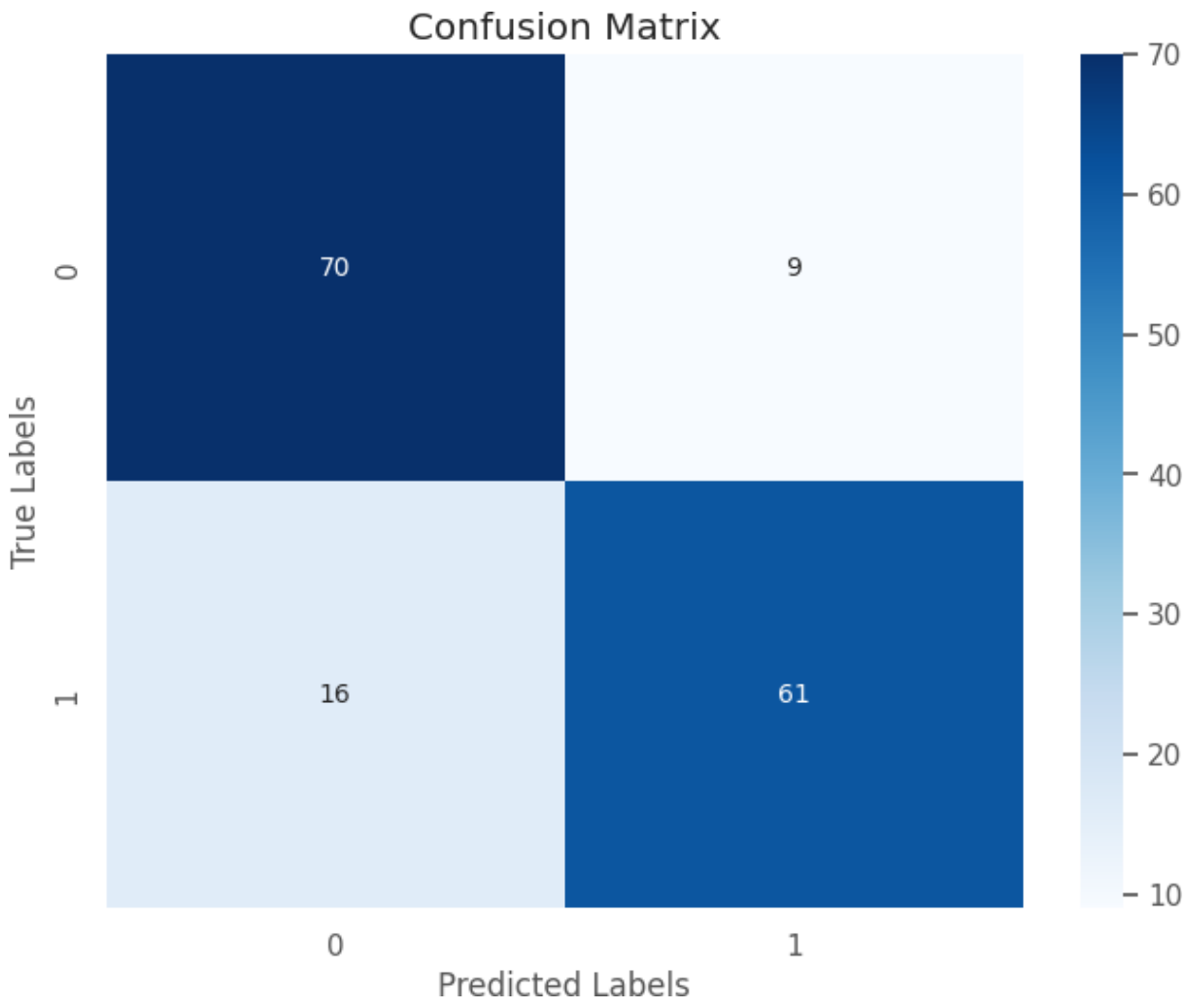
9.2.4.2 Confusion Matrix

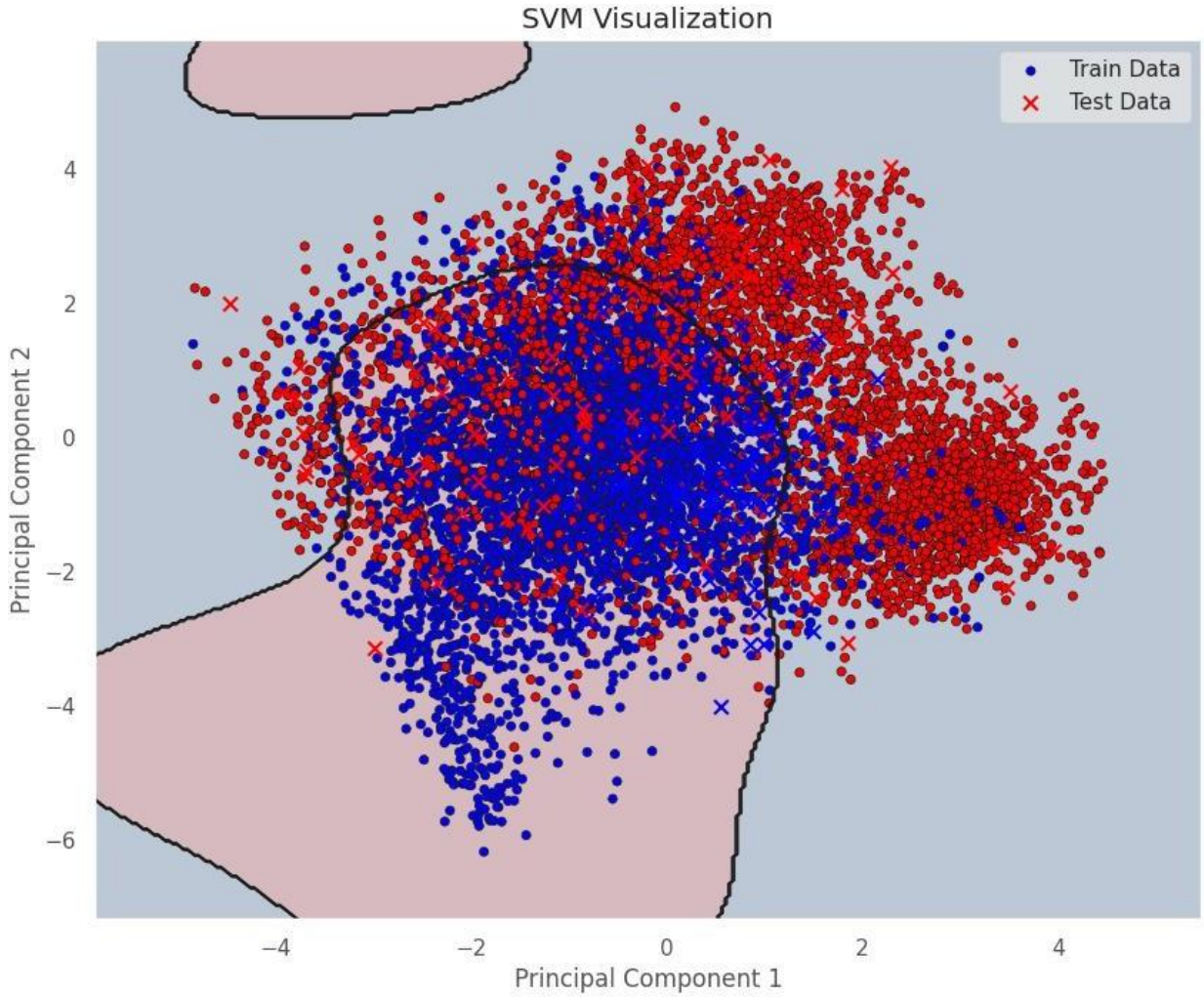
Year 3 As Testing Year



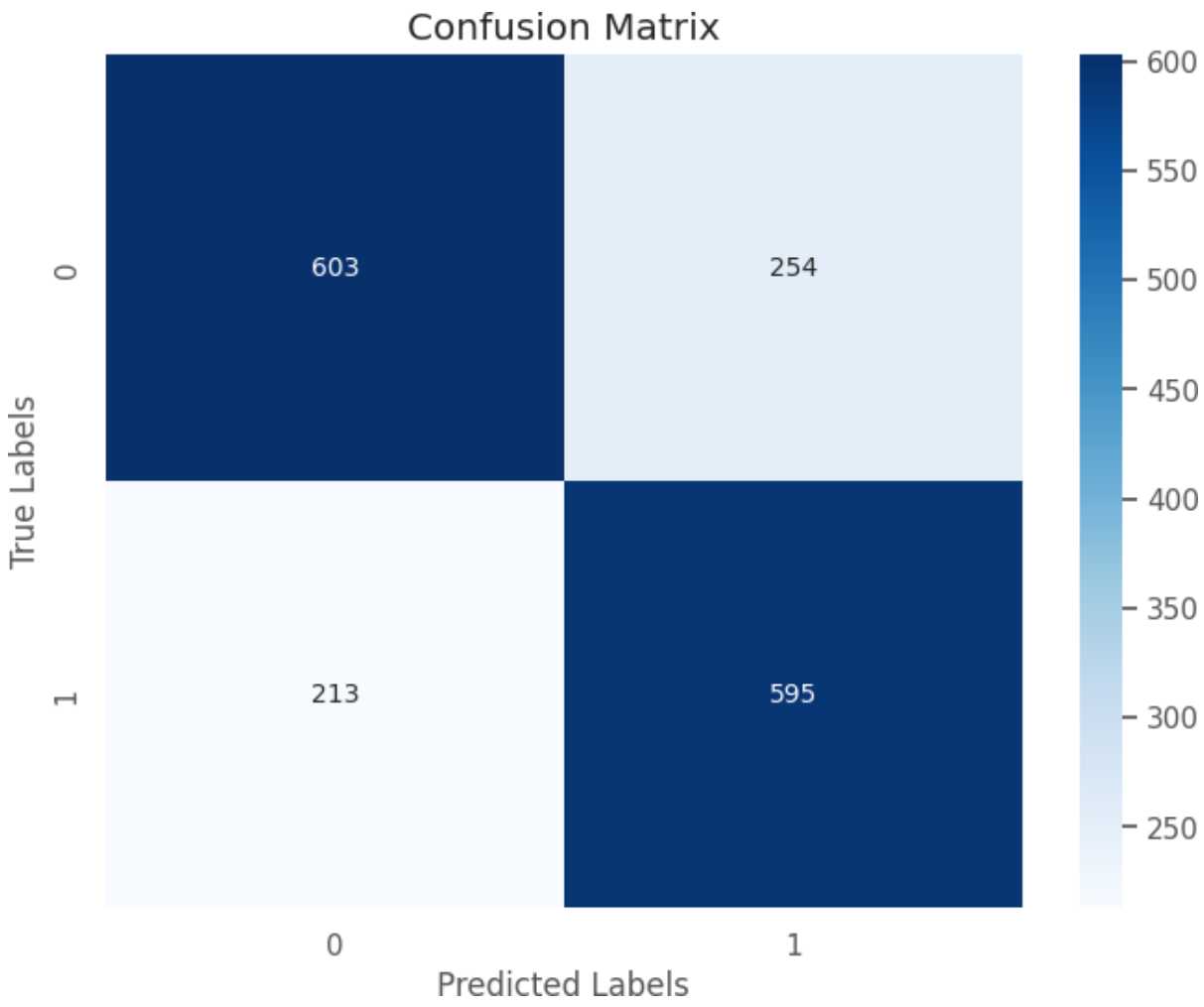

SVM Visualization
Train Data
Test Data
Principal Component 2
Principal Component 1
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4

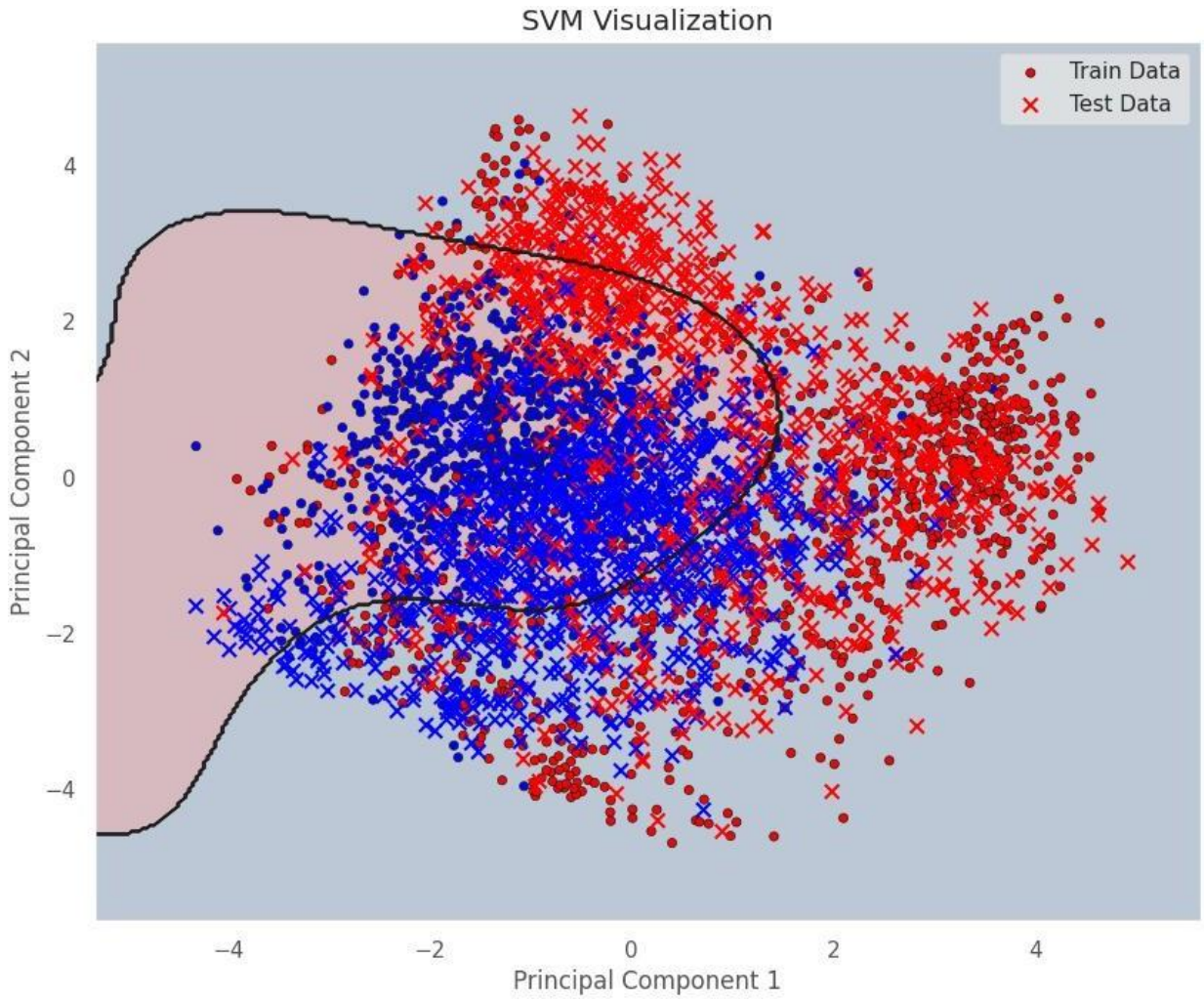
Year 1 As Testing Year




SVM Visualization
Train Data
Test Data
Principal Component 2
Principal Component 1
4
2
0
−2
−4
−6
−4
−2
0
2
4

Year 2 As Testing Year





9.3 SMOTE

9.3.1 XGBoost

9.3.1.1 Model Results for XGBoost

Train Years	Test Year	Best Parameters	Precision	Recall	F1-Score	Accuracy

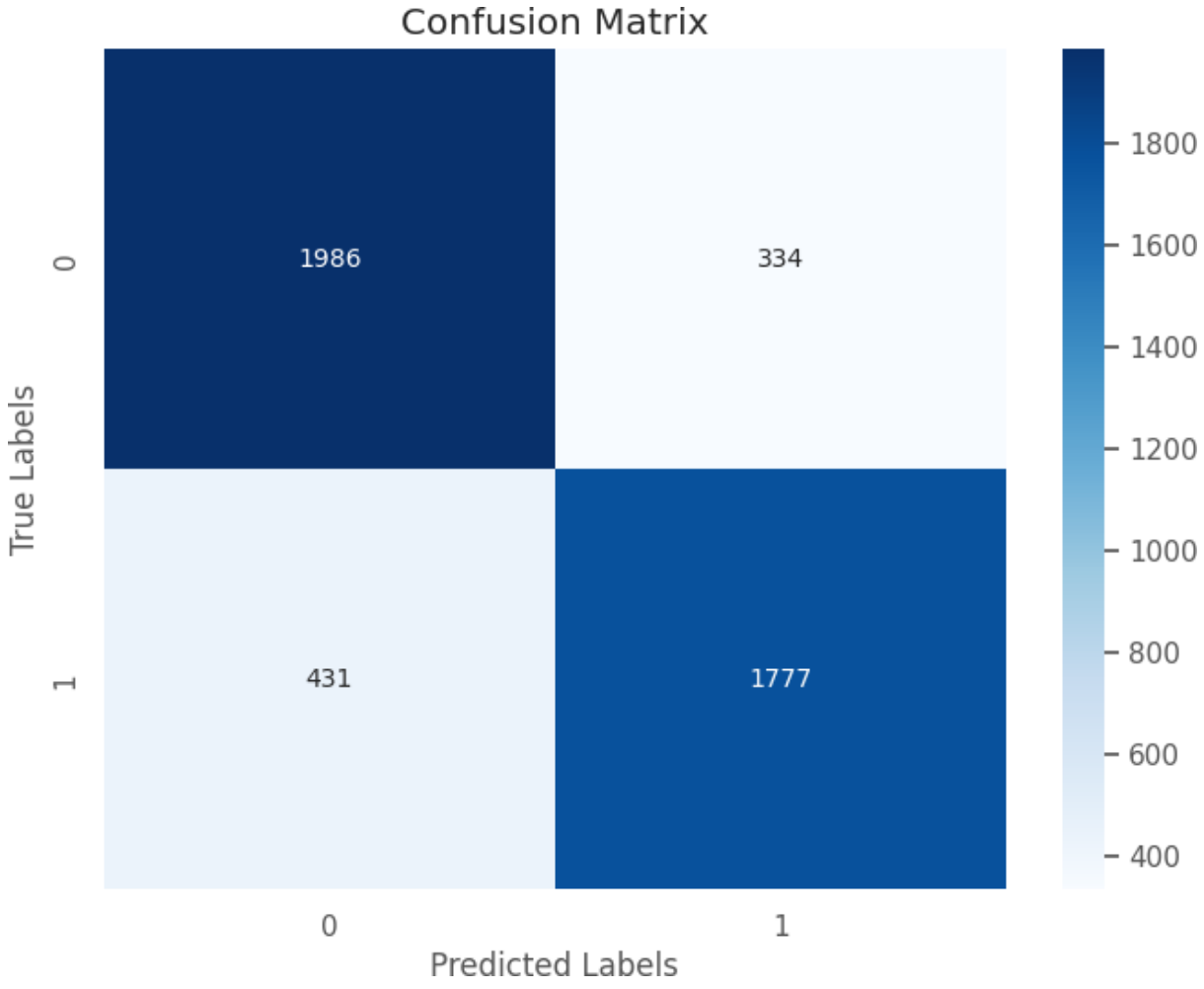
			0	**1**	**0**	**1**	**0**	**1**	
Year1 + Year 2	Year3	Learning rate: 0.2 Max_length: 7 N_estimators: 300	0.82	0.84	0.86	0.80	0.84	0.82	0.83
Year2 + Year 3	Year1	Learning rate: 0.2 Max_length: 10 N_estimators: 300	0.73	0.86	0.91	0.62	0.81	0.72	0.77
Year1 + Year 3	Year2	Learning rate: 0.2 Max_length: 7 N_estimators: 300	0.64	0.87	0.93	0.49	0.76	0.63	0.71

9.3.1.2 Feature Importance(features range from 0 to 11)

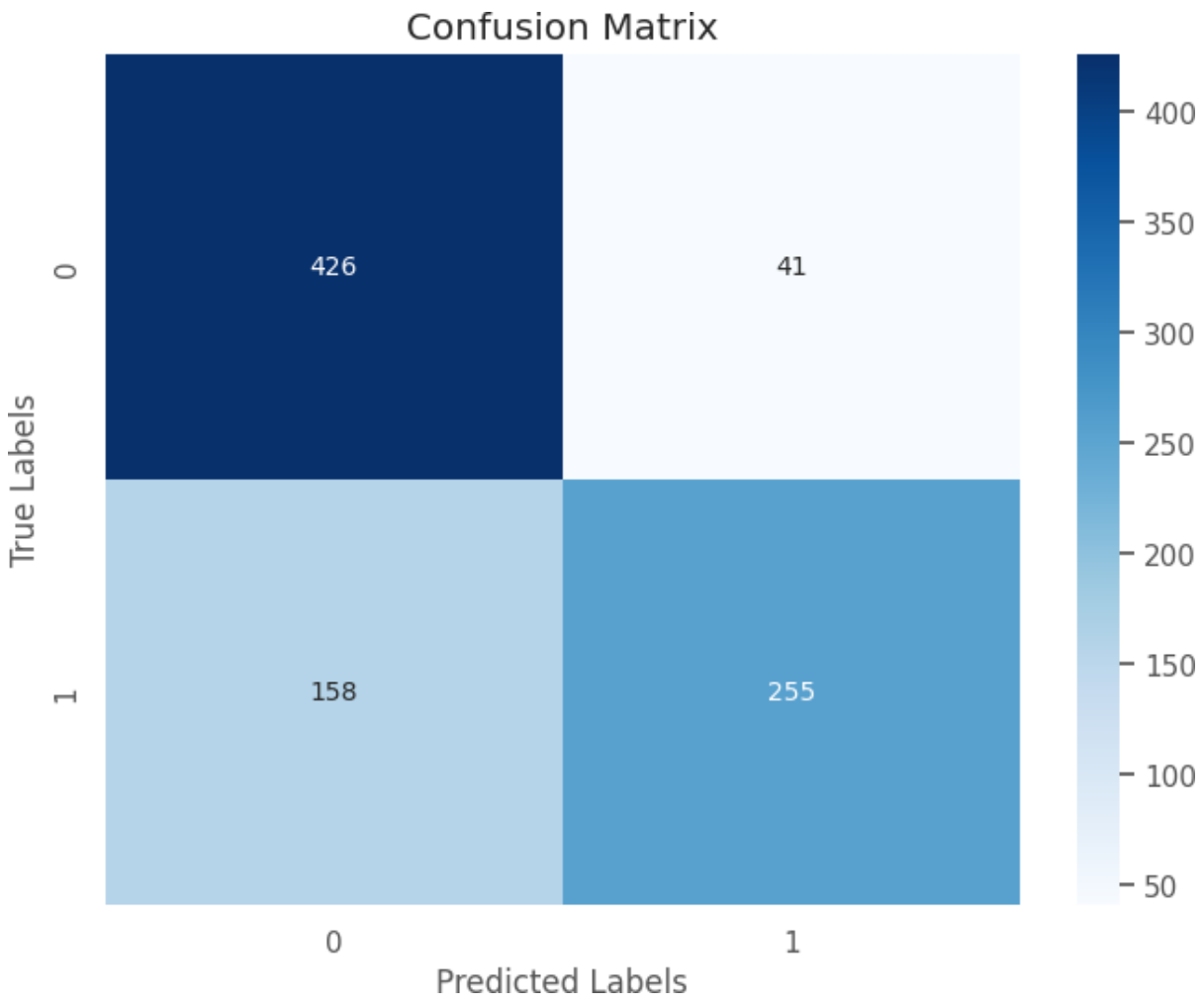
Rank	**Year1 + Year 2 -> Year3**	**Year2 + Year3 -> Year1**	**Year1 + Year3 -> Year2**
1	5	5	11
2	4	4	5
3	6	11	4
4	7	6	8
5	8	7	6
6	11	3	10
7	10	8	3
8	3	0	7
9	0	2	1
10	9	10	2
11	2	1	0
12	1	9	9

9.3.1.3 Confusion Matrix:

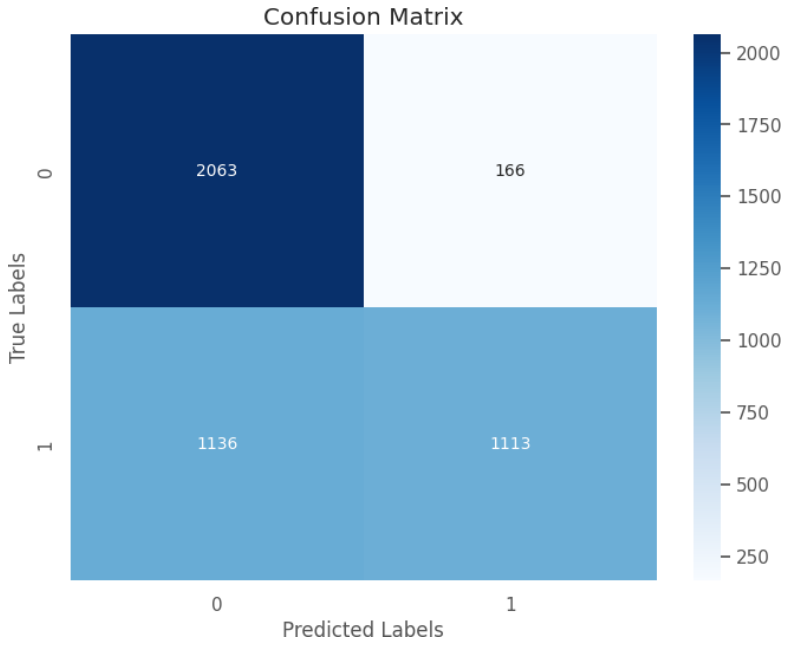
Year3 as test year



Year1 as test year



Year2 as test year



9.3.2 Bagging(Bootstrap Aggregation)

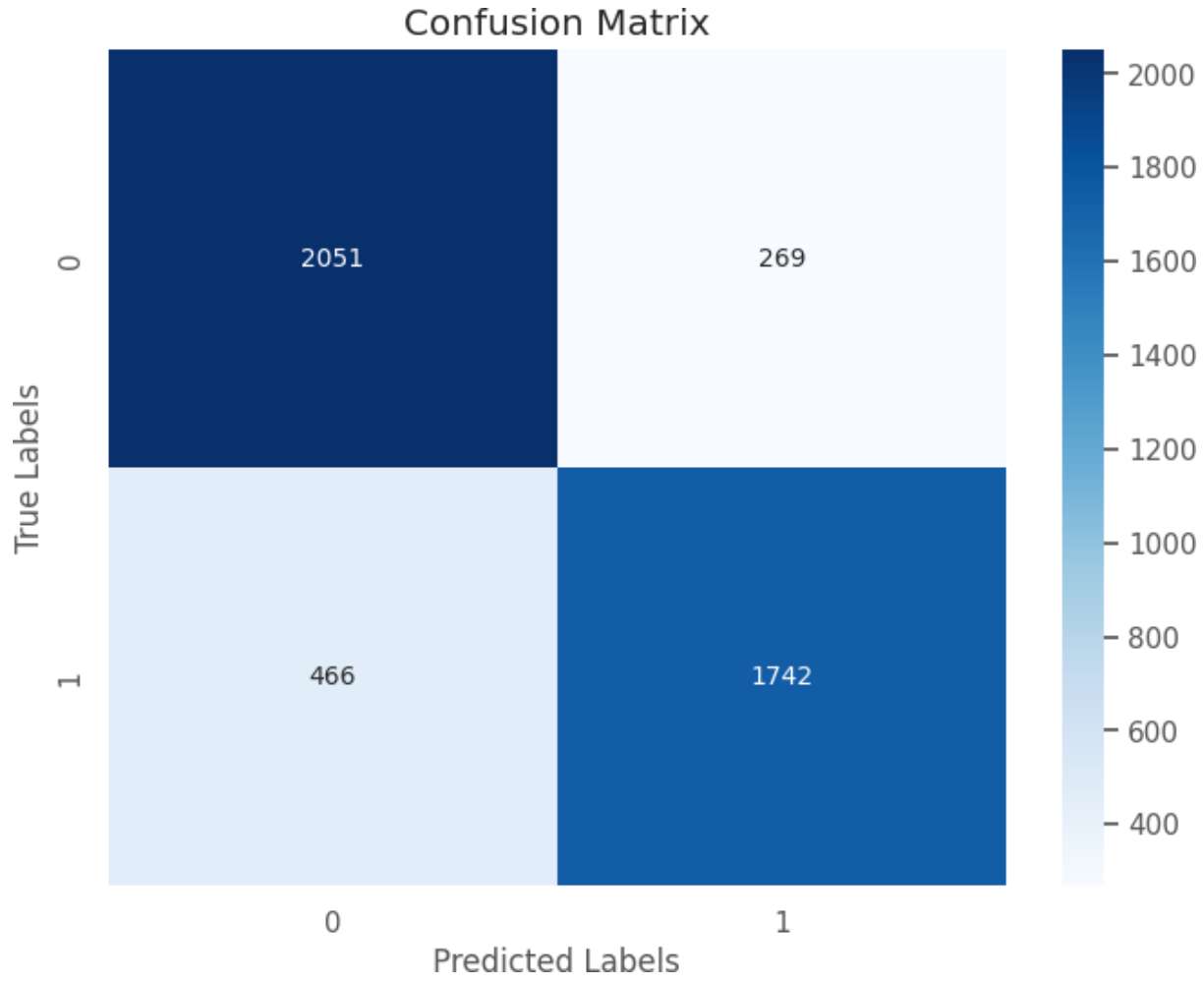
9.3.2.1 Model Results

Train Years	Test Year	Best Parameters	Precision		Recall		F1-Score		Accuracy
			0	1	0	1	0	1	
Year1 + Year 2	Year3	estimator__max_depth: None Max_samples: 1.0 Max_features: 0.7 N_estimators: 100	0.81	0.87	0.88	0.79	0.85	0.83	0.84
Year2 + Year 3	Year1	estimator__max_depth: None Max_samples: 1.0 Max_features: 0.7 N_estimators: 100	0.73	0.86	0.91	0.61	0.81	0.72	0.77

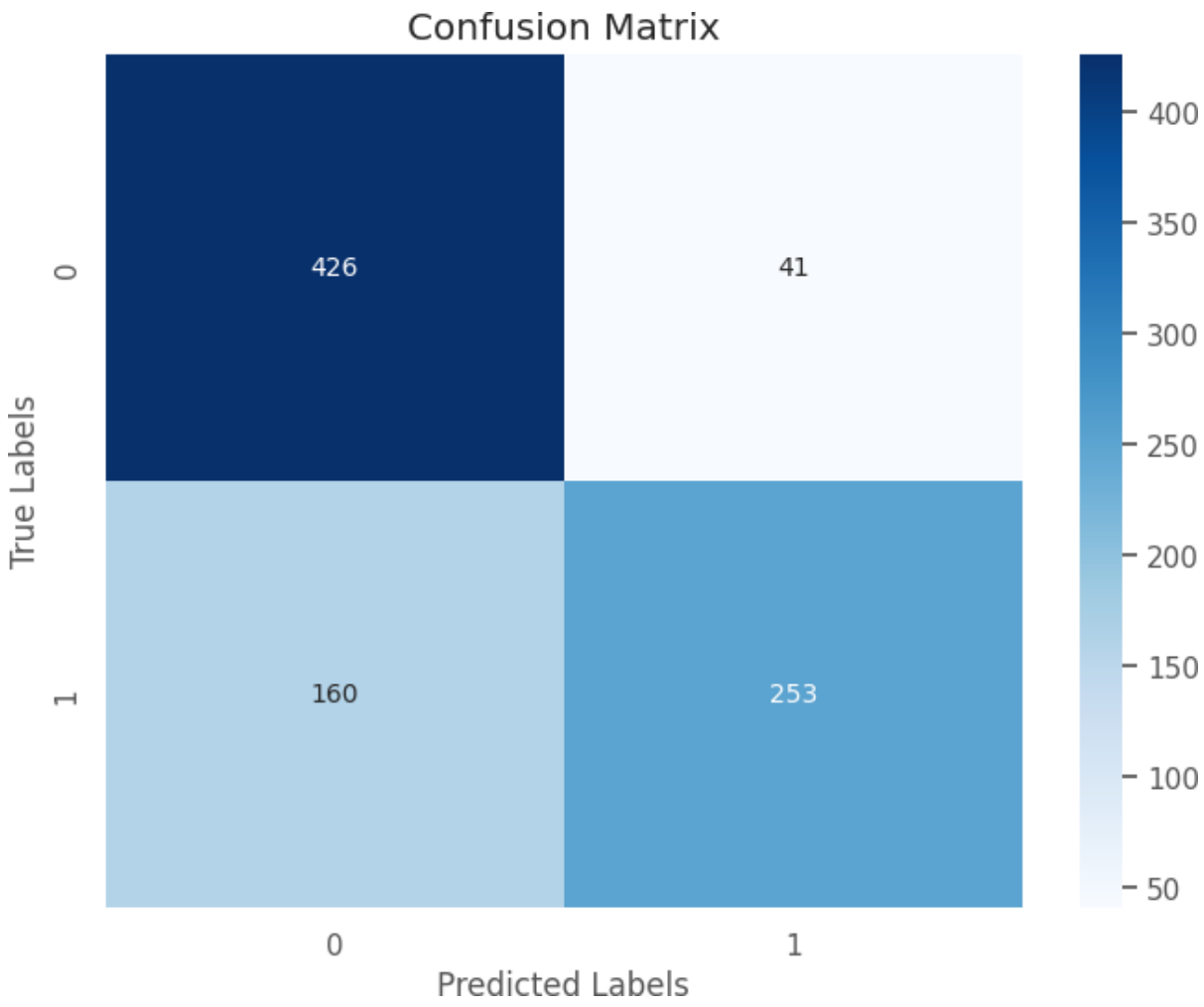
Year1 + Year 3	Year2	estimator_max_depth: None Max_samples: 1.0 Max_features: 0.7 N_estimators: 100	0.66	0.89	0.93	0.53	0.78	0.67	0.73

9.3.2.2 Confusion Matrix

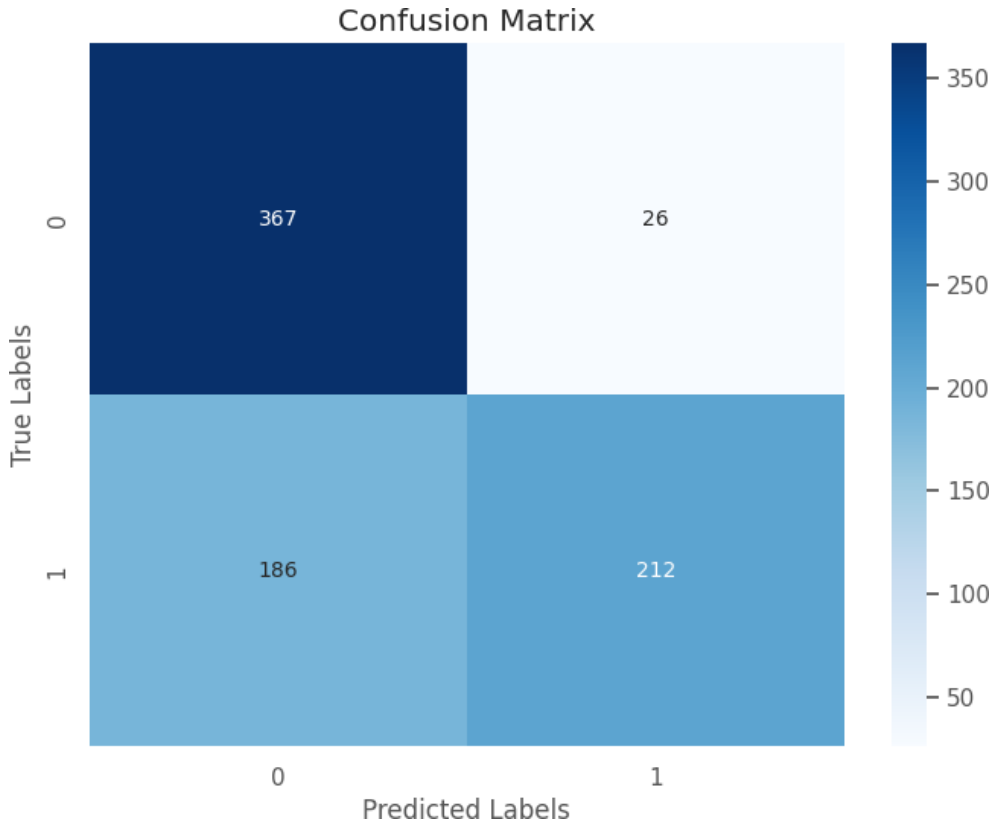
Year 3 As Testing year



Year 1 As Testing year



Year 2 As Testing year



9.3.3 Random Forest

9.3.2.1 Model Results

Train Years	Test Year	Best Parameters	Precision		Recall		F1-Score		Accuracy
			0	1	0	1	0	1	
Year1 + Year 2	Year3	Max_depth: None Min_samples_leaf: 1 N_estimators: 200	0.82	0.86	0.88	0.80	0.85	0.83	0.84
Year2 + Year 3	Year1	Max_depth: None Min_samples_leaf: 1 N_estimators: 200	0.73	0.86	0.91	0.62	0.81	0.72	0.78

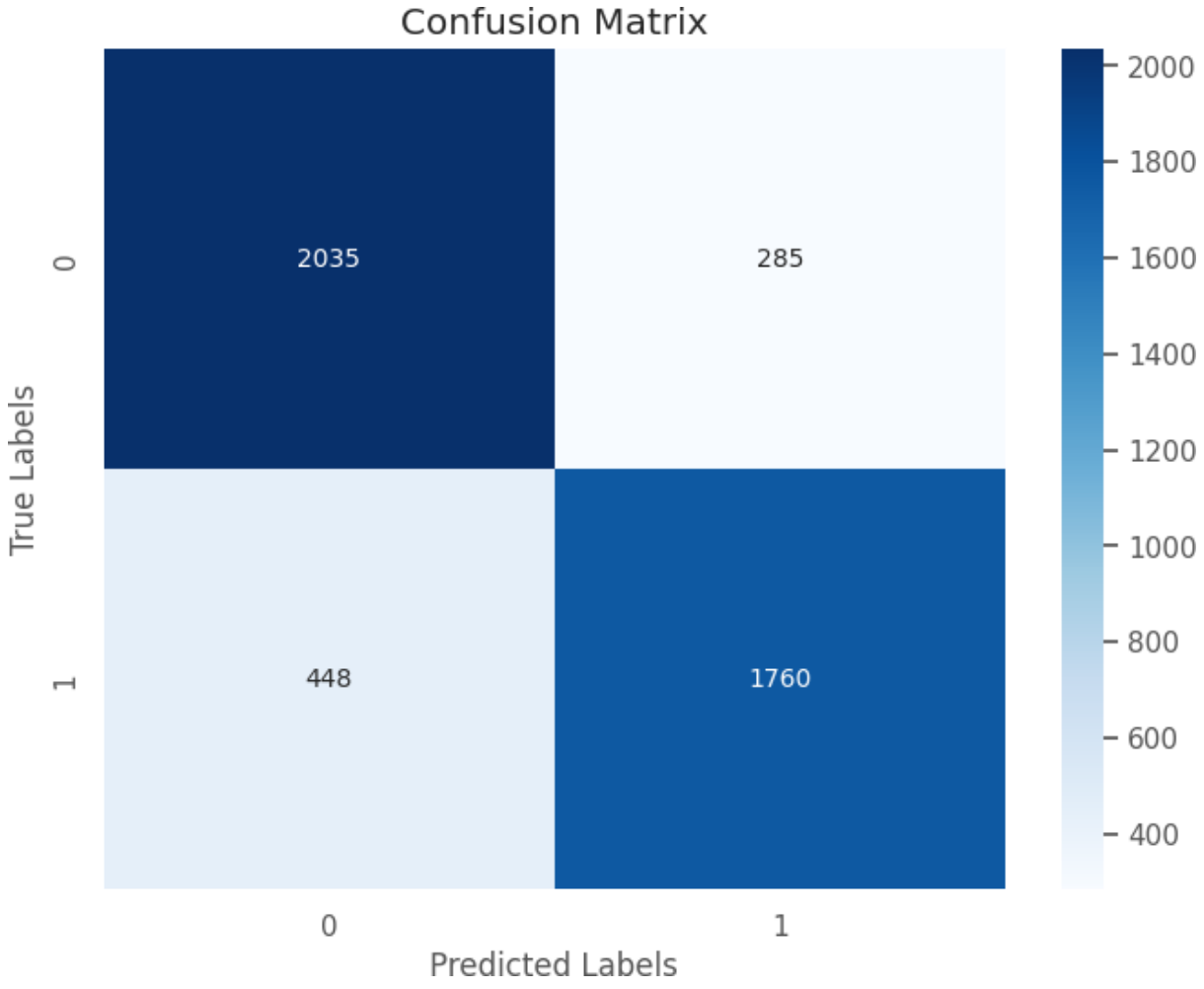
Year1 + Year 3	Year2	Max_depth: None Min_samples_leaf: 2 N_estimators: 100	0.65	0.81	0.91	0.45	0.76	0.58	0.69

9.3.2.2 Feature Importance

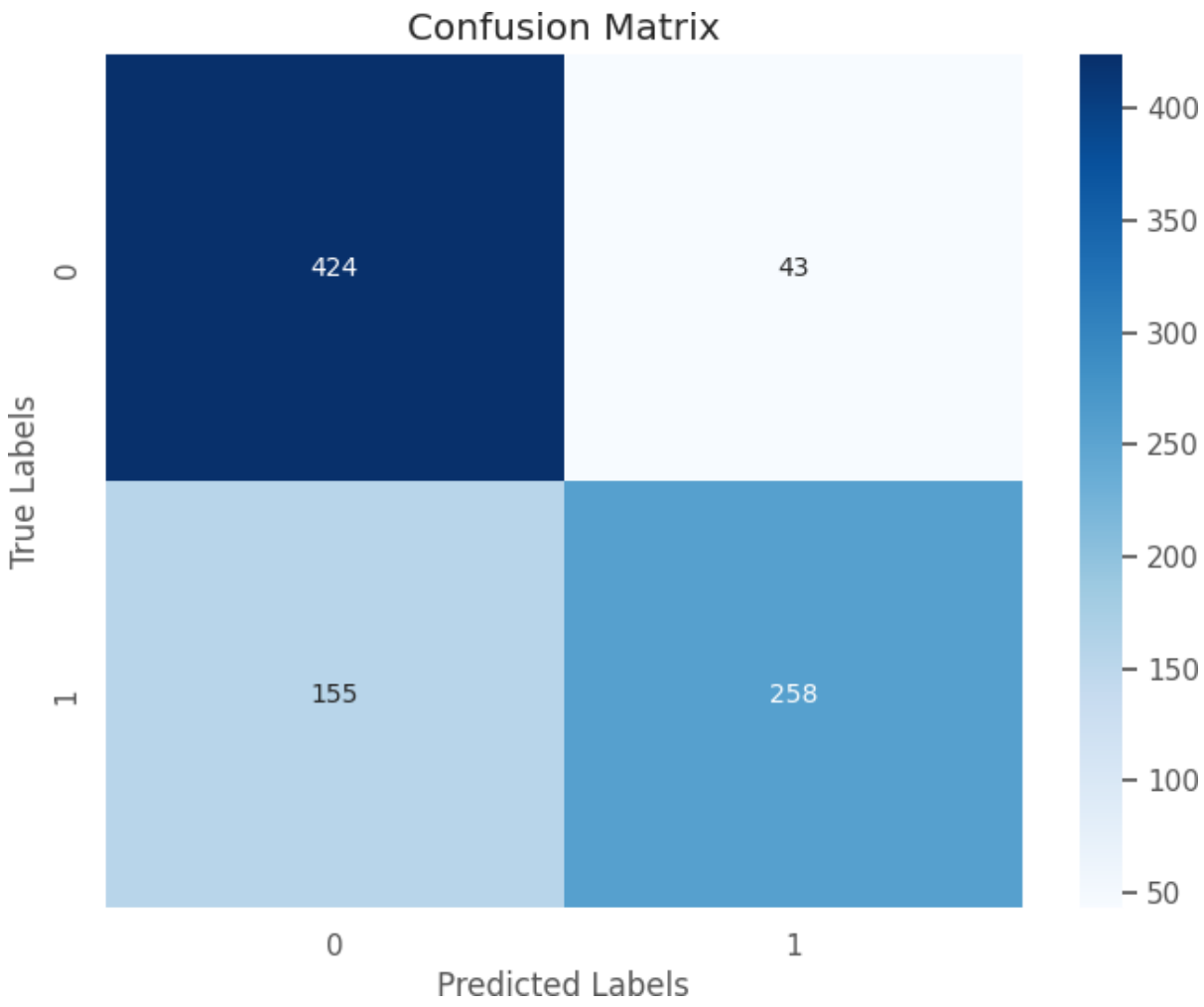
Rank	**Year1 + Year 2 -> Year3**	**Year2 + Year3 -> Year1**	**Year1 + Year3 -> Year2**
1	10	10	10
2	9	9	9
3	11	11	3
4	4	8	2
5	8	7	7
6	3	2	4
7	0	4	8
8	2	3	6
9	6	6	11
10	7	5	5
11	5	1	0
12	1	0	9

9.3.2.3 Confusion Matrix

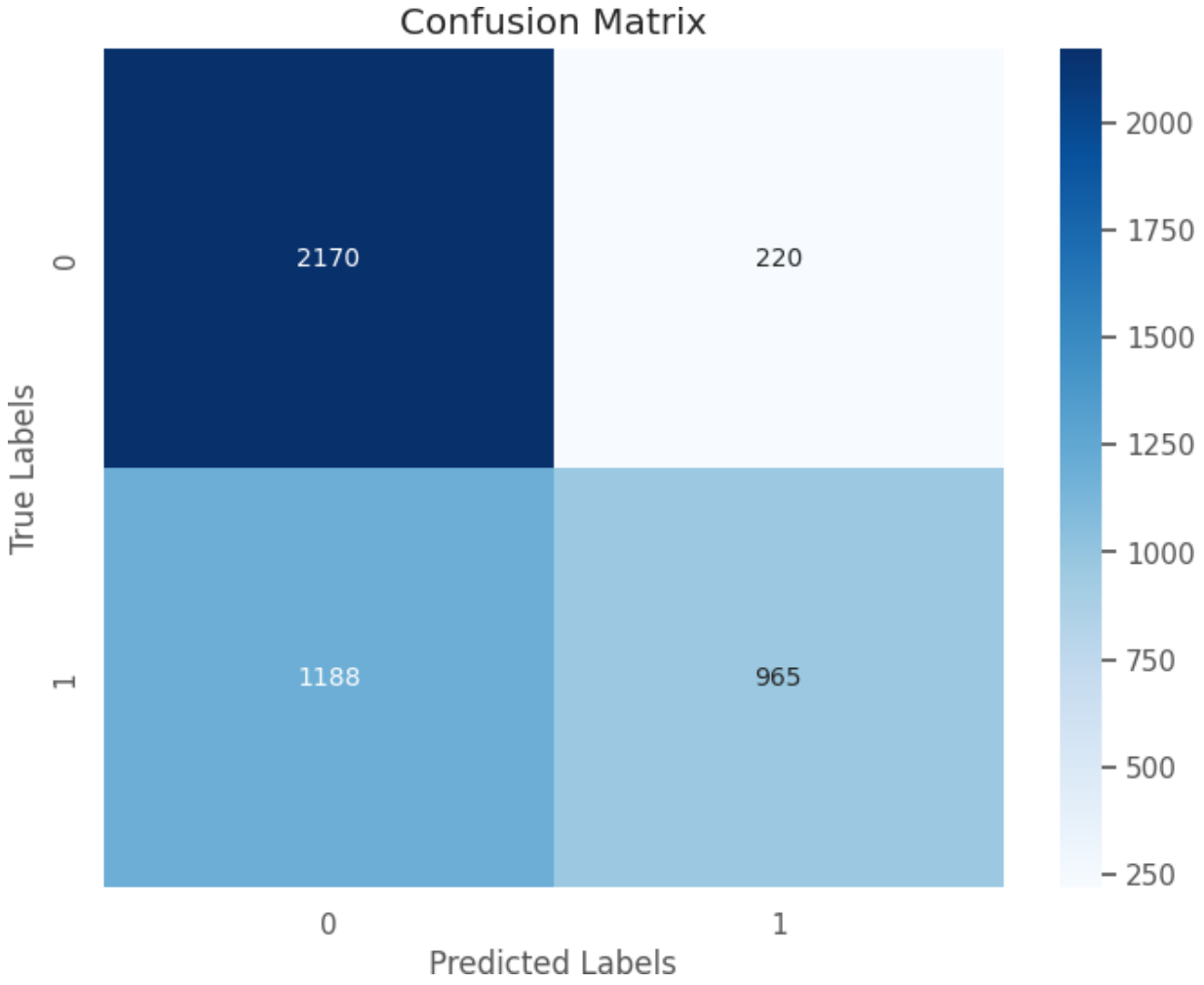
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



9.3.4 SVM

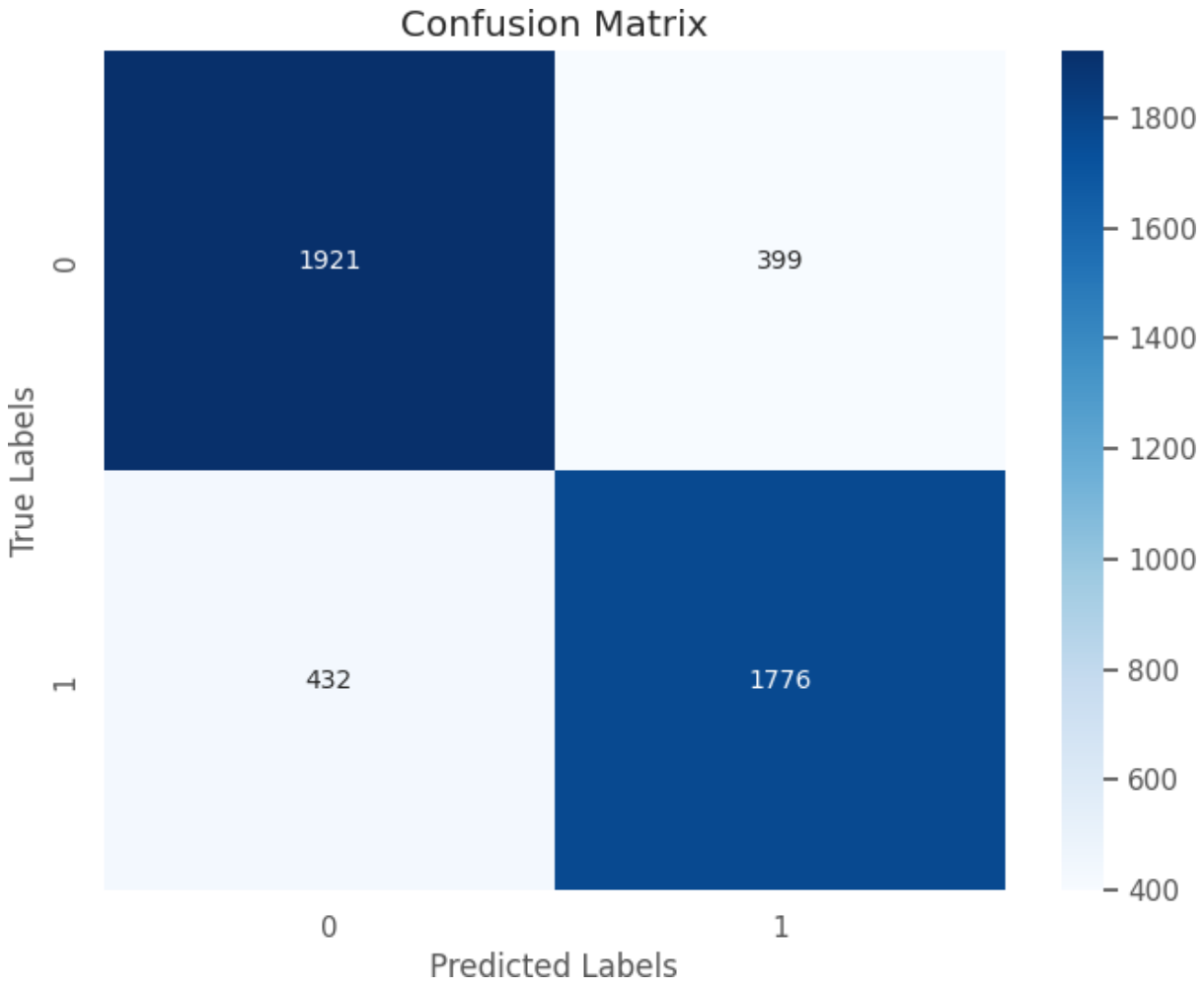
9.3.4.1 Model Results

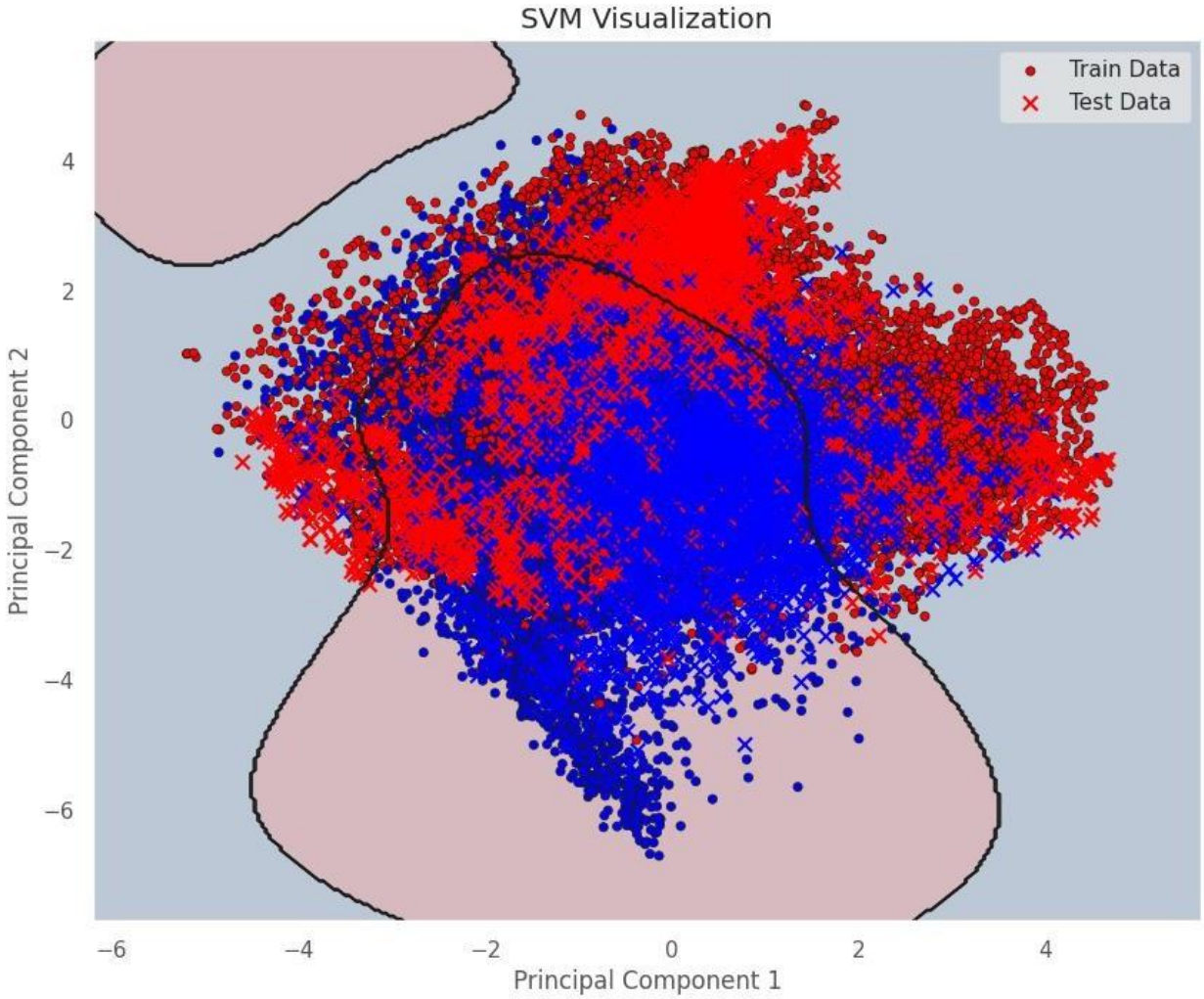
Train Years	Test Year	Best Parameters	Precision		Recall		F1-Score		Accuracy
			0	1	0	1	0	1	
Year1 + Year 2	Year3	'C': 10 'gamma': 'scale', 'kernel': 'rbf'	0.82	0.82	0.83	0.80	0.82	0.81	0.82
Year2 + Year 3	Year1	'C': 10	0.77	0.85	0.89	0.70	0.83	0.77	0.80

		'gamma': 'auto', 'kernel': 'rbf'							
Year1 + Year 3	Year2	'C': 10 'gamma': 'scale', 'kernel': 'rbf'	0.65	0.76	0.86	0.49	0.74	0.59	0.68

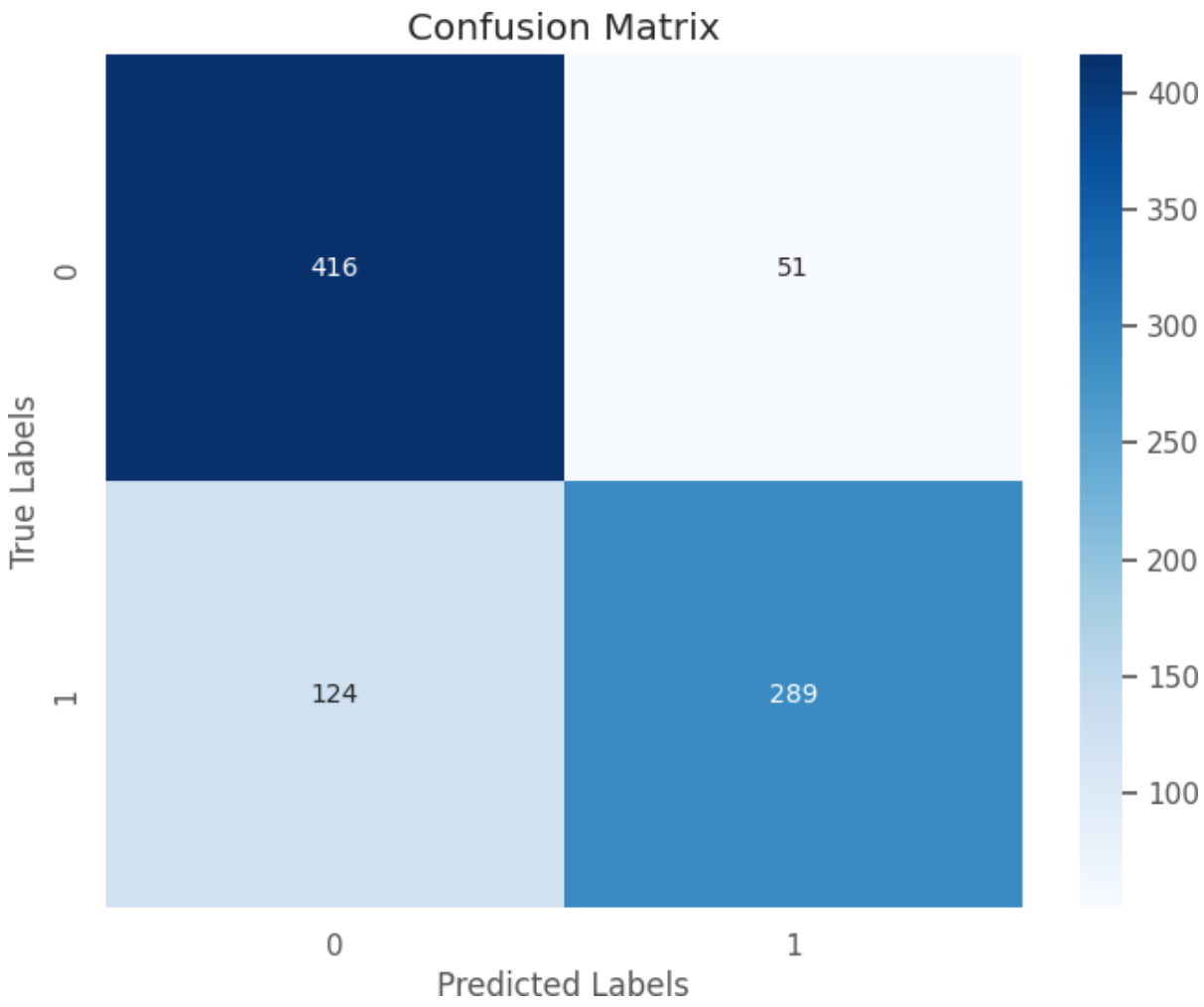
9.3.4.2 Confusion Matrix

Year 3 As Testing Year

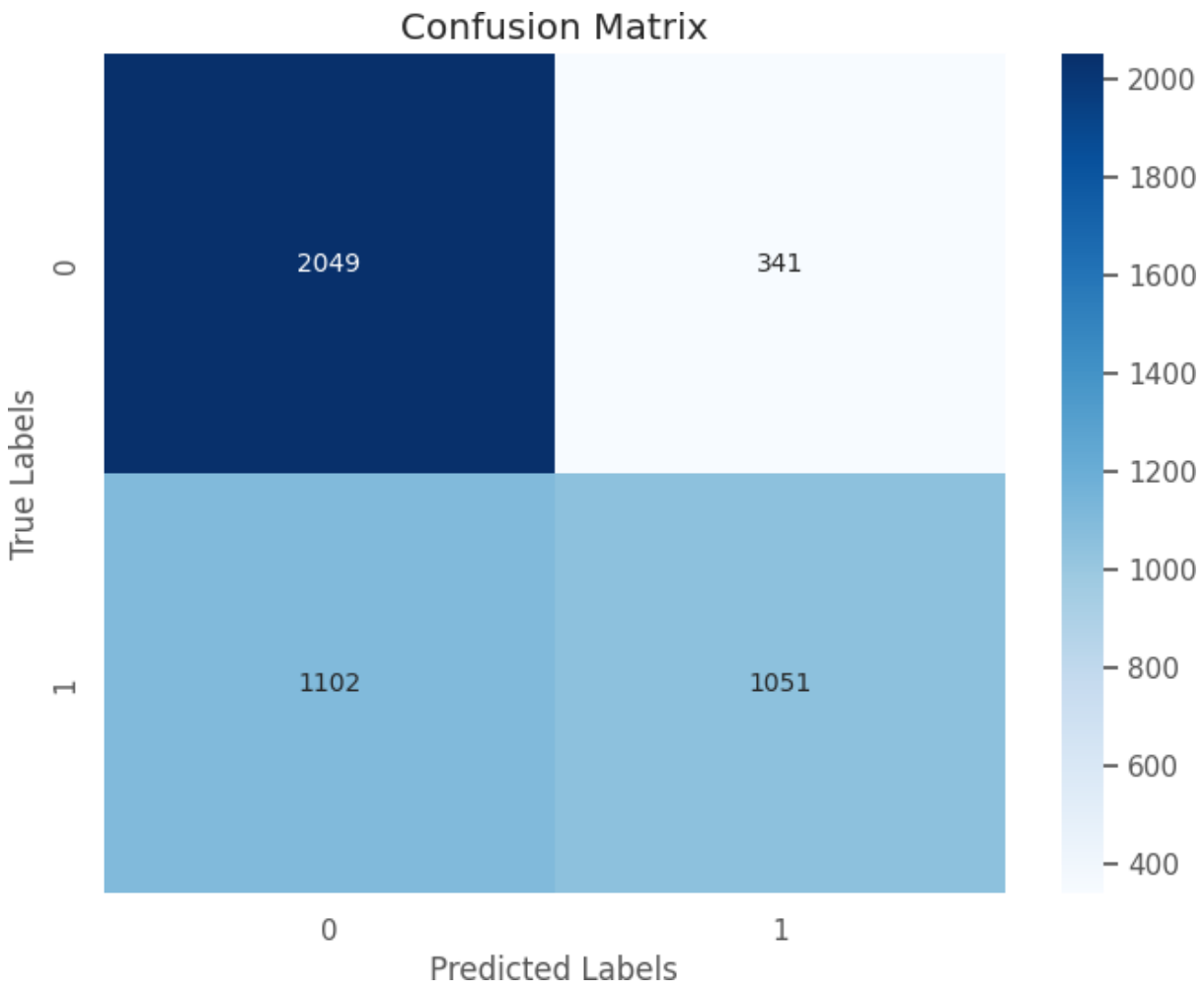


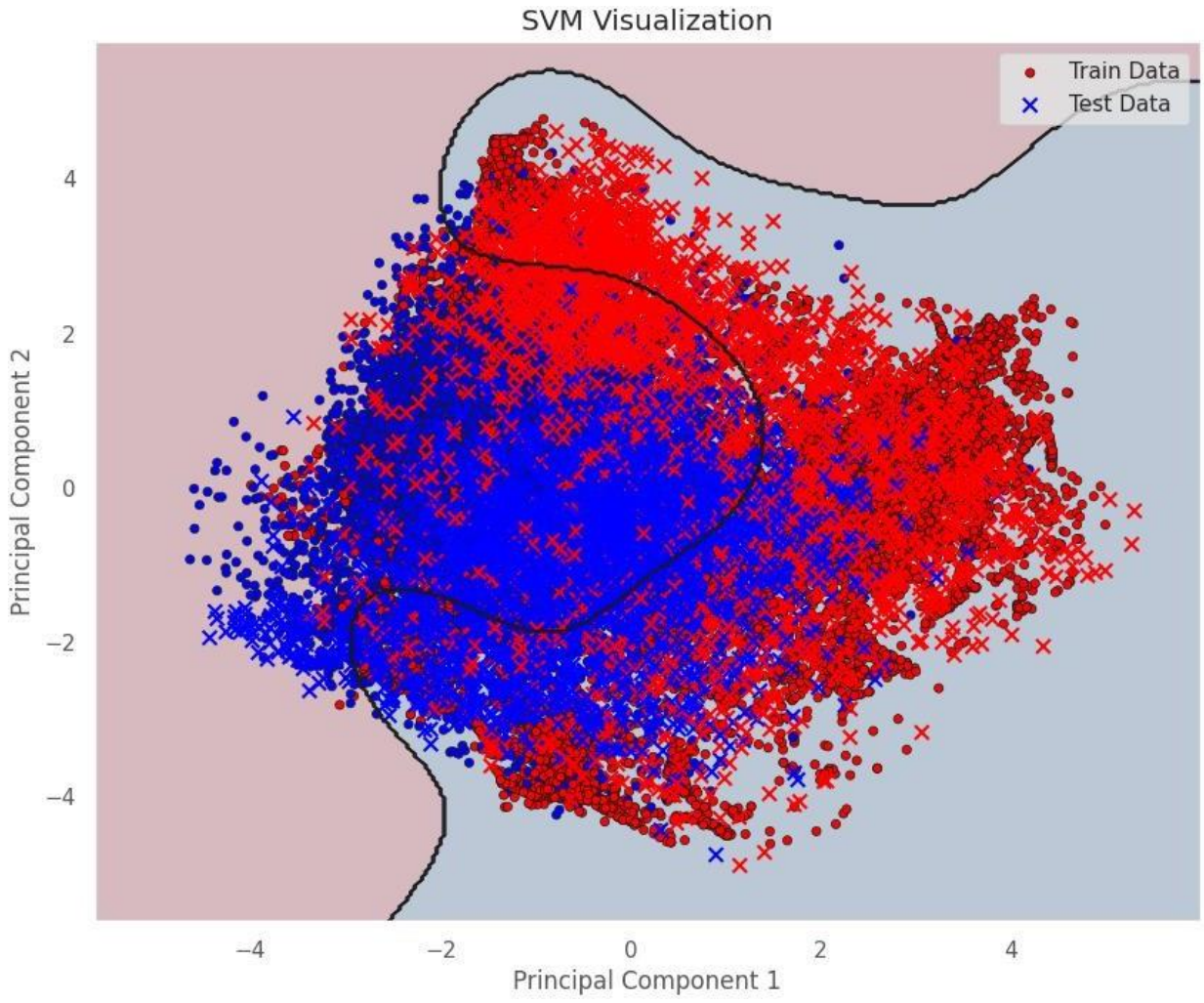

SVM Visualization
Train Data
Test Data
Principal Component 2
Principal Component 1
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4

Year 1 As Testing Year



Year 2 As Testing Year





10. Analyzing The Results Of Grid Search

10.1 Analysis Of XGBoost Results

1. **Performance Across Sampling Methods:**
    - **Random Oversampling**: Accuracy ranges from 67% to 83%.
    - **Random Undersampling**: Accuracy ranges from 70% to 86%, showing stable performance with better precision-recall trade-offs.
    - **SMOTE**: Accuracy ranges from 71% to 77%, with good recall for class 1 but moderate precision.
2. **Feature Importance Consistency**:

- Top-ranking features **(e.g., Feature 5, 6, 3, 4, 10, and 11)** occur consistently across the three sampling methods and train-test splits.
- Features ranked lower (e.g., **0, 1, 2, 9**) consistently appear in the bottom across all scenarios.

3. Based on the aggregated feature importance, we can divide features into **"retain"** and **"remove"** groups:
   - **Features to Retain** (Highly Important - Appears Consistently in Top Ranks):

     Feature 5, 6, 3, 4, 11, 10, 8

   - **Features to Remove** (Low Importance - Appears in Bottom Ranks):

     Feature 0, 1, 2, 9

4. Best parameters to use for further final training:
   - **Learning rate**: 0.1 or 0.2
   - **Max_depth**: 10 (frequent in high-performing models)
   - **N_estimators**: 300 (most consistent value)

10.2 Analysis Of Bagging Results

1. Performance Across Sampling Methods

Random Oversampling

- **Accuracy**: Ranges from **67% to 78%**.
- The **Year1 + Year2 → Year3** split achieves the highest accuracy (**78%**) and balanced metrics.
- **Year2 + Year3 → Year1** and **Year1 + Year3 → Year2** show weaker results, with lower F1-scores and imbalanced precision-recall trade-offs for class 1.

Random Undersampling

- **Accuracy**: Ranges from **66% to 91%**.
- **Year2 + Year3 → Year1** consistently outperforms with a high accuracy of **91%** and balanced precision-rqsecall values, making it the most effective split for undersampling.
- The other two splits show moderate accuracy (~66%-75%), with relatively balanced precision and recall.

SMOTE

- **Accuracy**: Ranges from **73% to 77%**.
- Results are consistent across splits, with the **Year1 + Year2 → Year3** split achieving the best accuracy (**77%**).
- Performance is stable, with well-balanced precision and recall across classes.

2. Sampling Strategy Insights

- **Random Undersampling** achieves the best performance in one split (**91% accuracy**) but struggles with stability across splits.
- **SMOTE** is more consistent across splits but does not reach the peak performance of undersampling.
- **Random Oversampling** achieves a good balance but slightly lower performance compared to undersampling.

3. Best Parameters for Bagging

- **Base Estimator**: Decision Tree with max_depth=None.
- **Max Samples**: 1.0 (Use all samples per tree).
- **Max Features**: 0.7 (Balanced performance with a subset of features).
- **Number of Estimators (n_estimators)**: 100.

10.3 Analysis Of Random Forest Results

1. Performance Across Sampling Methods:

- **Random Oversampling**:
    - **Accuracy**: Ranges from 67% to 78%.
    - Performance is moderate with fluctuations, indicating some variability in handling class imbalances.
- **Random Undersampling**:
    - **Accuracy**: Ranges from 67% to 91%.
    - Better performance stability, with the highest accuracy (91%) being achieved in one of the splits.
    - Shows good precision-recall trade-offs, particularly in situations where the data is more balanced.
- **SMOTE**:
    - **Accuracy**: Ranges from 67% to 77%.
    - Good recall for class 1, but precision may be moderate due to the generation of synthetic samples.

2. Feature Importance Consistency:

Consistent Top-Ranking Features:

- Features consistently ranked highly across all sampling methods and train-test splits include:
    - **Feature 10, Feature 9, Feature 11, Feature 4, Feature 8**.
    - These features tend to play a central role in the model's predictions, indicating they are **highly important**.

Consistent Low-Ranking Features:

- Features consistently ranked low or at the bottom across all scenarios include:

- **Feature 3, Feature 1, Feature 0, Feature 2, Feature 7**.
- These features are less impactful for model performance, and removing them could simplify the model.

Best Parameters for Final Model:

- **Max_depth**: None (or 10 for regularization)
- **Min_samples_leaf**: 1 (or 2 for regularization)
- **N_estimators**: 100 to 200

11. Final Model Training Using Best Parameters And All Features

The goal of this section is to train various machine learning models on the dataset using different sampling techniques, such as Random Oversampling (RO), Random Undersampling (RU), and Synthetic Minority Over-sampling Technique (SMOTE). These techniques are used to address class imbalance in the dataset by either oversampling the minority class (RO and SMOTE) or undersampling the majority class (RU).

Each model is trained on multiple training datasets corresponding to different years, and their performance is evaluated on the data from the remaining year. This process ensures a thorough evaluation across different time periods.

11.1 Random Oversampling

11.1.1 XGBoost

- **Model Overview**: XGBoost (Extreme Gradient Boosting) is an efficient and scalable implementation of gradient boosting. It is used for both classification and regression tasks and is known for its high performance in machine learning competitions.
- **Parameters**:

    - **Learning rate (0.2)**: Controls how much the model weights each new tree in the boosting process.
    - **Max depth (10)**: Defines the maximum depth of each tree, helping to control overfitting.
    - **N_estimators (300)**: The number of trees in the ensemble model.
- **Approach**: The model is trained using data from two years (Year 1 and Year 2) and tested on Year 3. The process is repeated for other combinations, training on two years and testing on one.

11.1.2 Bagging

- **Model Overview**: Bagging (Bootstrap Aggregating) is an ensemble method that combines the predictions of multiple base models (often decision trees) to improve generalization and reduce overfitting.
- **Best Parameters**:
    - **Base estimator**: Decision tree with no depth restriction (max_depth=None).
    - **Max samples (1.0)**: All samples are used in each iteration.
    - **Max features (0.7)**: Only 70% of the features are randomly selected for each base model.
    - **N_estimators (100)**: 100 base models are created.
- **Approach**: Similar to XGBoost, bagging models are trained using data from two years and tested on the third.

11.1.3 Random Forest

- **Model Overview**: Random Forest is another ensemble technique based on decision trees. It builds multiple decision trees and aggregates their predictions for better accuracy.
- **Best Parameters**:
    - **Max_depth (None or 10)**: Controls the depth of the trees; no limit or regularized depth.
    - **Min_samples_leaf (1)**: The minimum number of samples required to be at a leaf node.

    - **N_estimators (100 to 200)**: The number of trees in the forest.
- **Approach**: Like XGBoost and bagging, the model is trained on data from two years and tested on the remaining year.

11.1.4 SVM (Support Vector Machine)

- **Model Overview**: Support Vector Machine (SVM) is a supervised learning algorithm that can perform both classification and regression tasks. It finds a hyperplane that best separates the data points from different classes.
- **Best Parameters**:
    - **C (10)**: Regularization parameter that determines the trade-off between achieving a low error on the training data and minimizing the model complexity.
    - **Gamma ('scale')**: A kernel coefficient that controls the decision boundary's smoothness.
    - **Kernel ('rbf')**: The radial basis function kernel, commonly used for non-linear classification tasks.
- **Approach**: SVM models are trained using two years' worth of data and tested on the third year.

11.2 Random Undersampling

11.2.1 XGBoost

- **Model Overview**: The XGBoost model here is trained on data that has undergone random undersampling, which reduces the size of the majority class to balance the dataset.
- **Approach**: The training and testing steps follow a similar process to Random Oversampling, with different training sets composed of undersampled data.

11.2.2 Bagging

- **Approach**: Bagging is applied to undersampled data to mitigate class imbalance. The training procedure remains similar, with different training and testing data combinations.

11.2.3 Random Forest

- **Approach**: Random Forest models are trained on undersampled data, following the same training and testing methodology as in the oversampling case.

11.2.4 SVM

- **Approach**: The SVM model is trained on undersampled data and tested on data from the remaining year, following the same methodology as the other models.

11.3 SMOTE (Synthetic Minority Over-sampling Technique)

11.3.1 XGBoost

- **Model Overview**: SMOTE is an over-sampling technique that generates synthetic examples in the feature space for the minority class to balance the dataset.
- **Approach**: The training and testing steps are similar to those used in the oversampling section, with synthetic data generated for the minority class.

11.3.2 Bagging

- **Approach**: Bagging is applied after performing SMOTE to balance the dataset. The model is trained using two years' data (with synthetic samples) and tested on the remaining year.

11.3.3 Random Forest

- **Approach**: Random Forest is trained on SMOTE data to address class imbalance and evaluate model performance.

11.3.4 SVM

- **Approach**: The SVM model is trained using synthetic samples from SMOTE and evaluated in the same manner as the other models.

Summary of Methodology

For each of the three sampling methods (Random Oversampling, Random Undersampling, and SMOTE), the classifiers (XGBoost, Bagging, Random Forest, and SVM) are trained on various combinations of training data from different years and tested on the remaining year. This allows for comprehensive evaluation of each model's performance in the context of class imbalance mitigation techniques. Each model's parameters are optimized to improve performance and ensure that the models generalize well on unseen data.

12. Summarizing Final Results

12.1 Random Oversampling

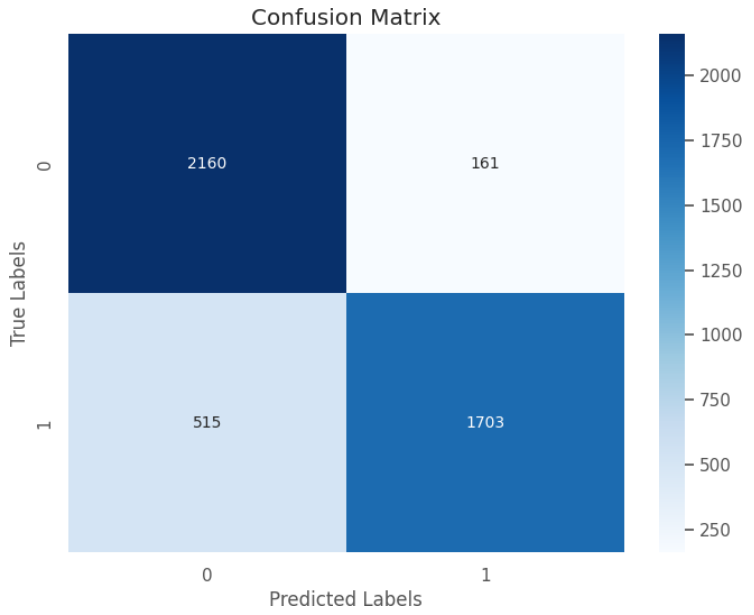
12.1.1 XGBoost

12.1.1.1 Model Results for XGBoost

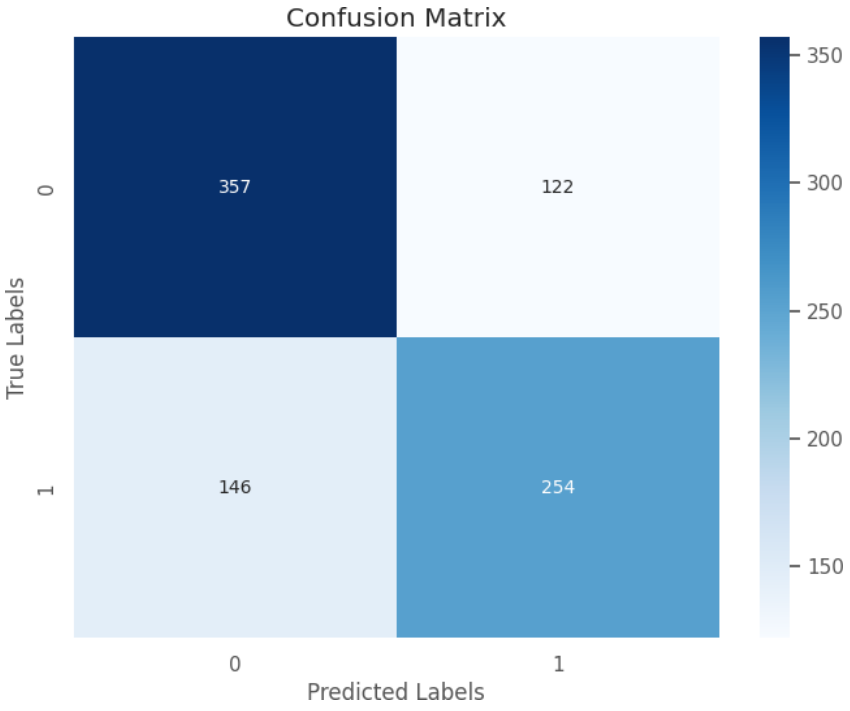
Train Years	Test Year	Precision		Recall		F1-Score		Accuracy
		0	1	0	1	0	1	
Year1 + Year 2	Year3	0.81	0.91	0.93	0.77	0.86	0.83	0.85
Year2 + Year 3	Year1	0.71	0.68	0.75	0.64	0.73	0.65	0.70
Year1 + Year 3	Year2	0.62	0.82	0.93	0.36	0.74	0.50	0.66

12.1.1.2 Confusion Matrix

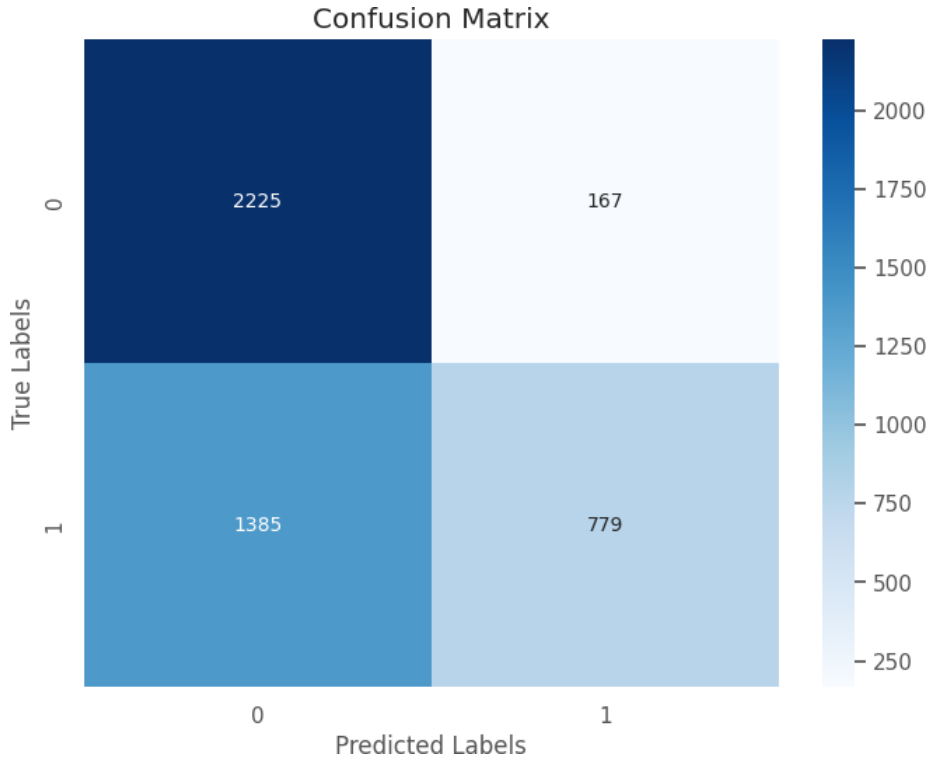
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



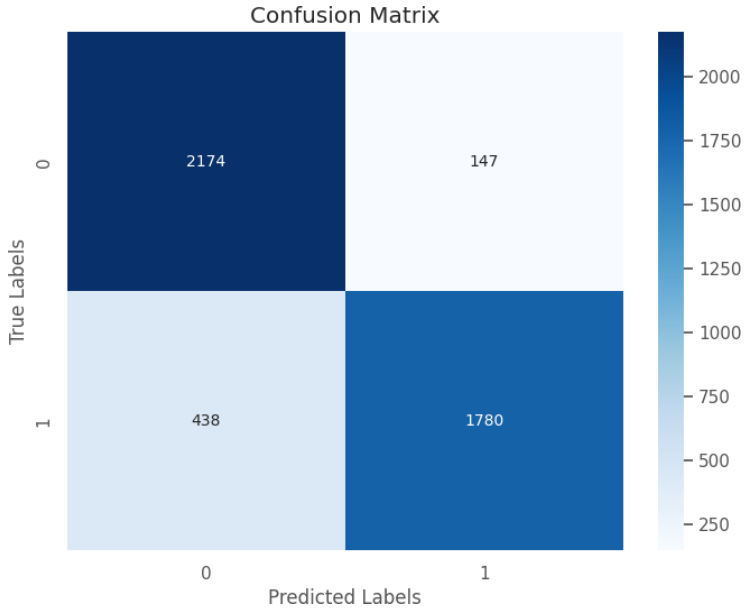
12.1.2 Bagging(Bootstrap Aggregation)

12.1.2.1 Model Results

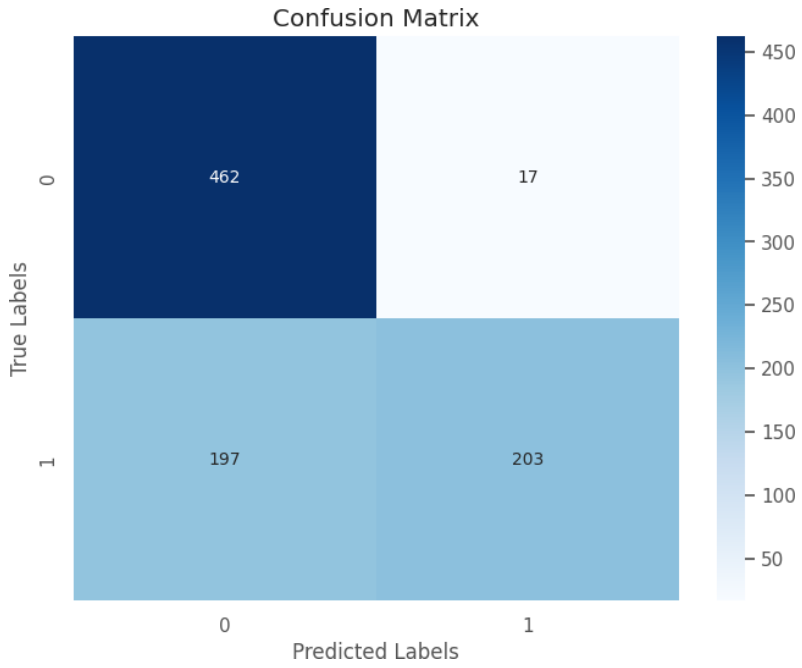
Train Years	Test Year	Precision		Recall		F1-Score		Accuracy
		0	1	0	1	0	1	
Year1 + Year 2	Year3	0.83	0.92	0.94	0.80	0.88	0.86	0.87
Year2 + Year 3	Year1	0.70	0.92	0.96	0.51	0.81	0.65	0.76
Year1 + Year 3	Year2	0.60	0.86	0.96	0.29	0.74	0.44	0.64

12.1.2.2 Confusion Matrix

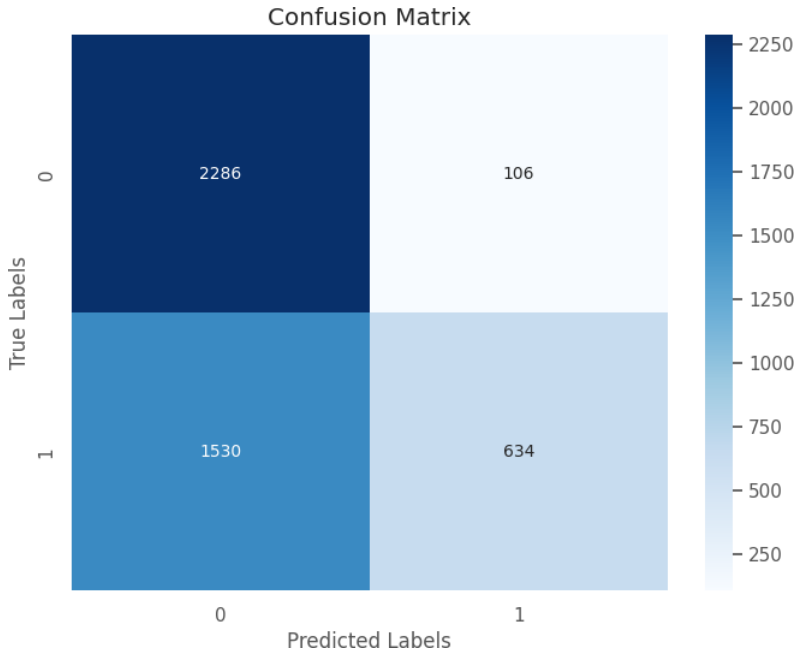
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



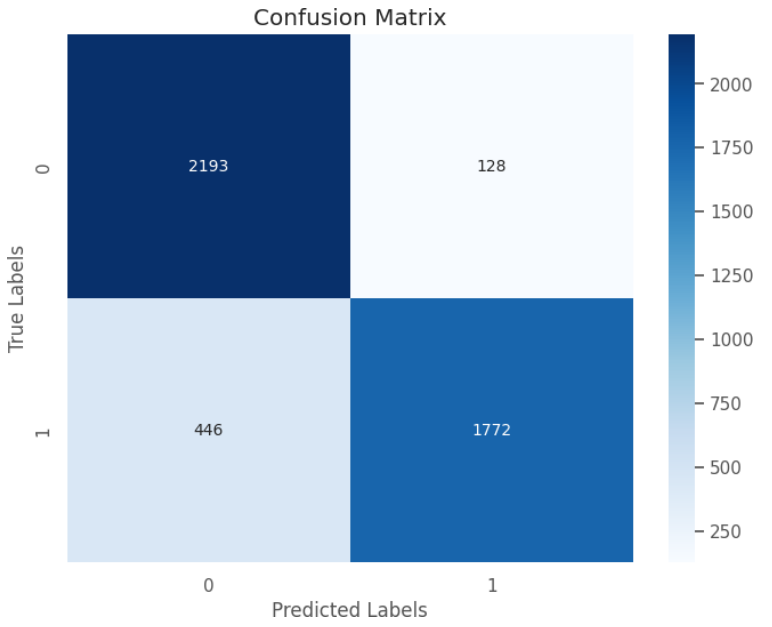
12.1.3 Random Forest

12.1.3.1 Model Results

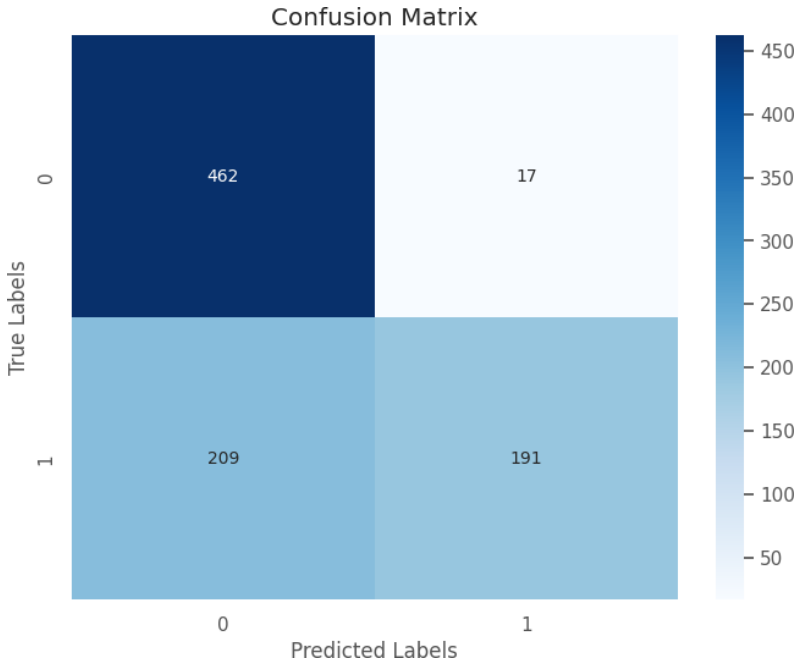
Train Years	Test Year	Precision		Recall		F1-Score		Accuracy
		0	1	0	1	0	1	
Year1 + Year 2	Year3	0.83	0.93	0.94	0.80	0.88	0.86	0.87
Year2 + Year 3	Year1	0.69	0.92	0.96	0.48	0.80	0.63	0.74
Year1 + Year 3	Year2	0.59	0.89	0.97	0.27	0.74	0.41	0.64

12.1.3.2 Confusion Matrix

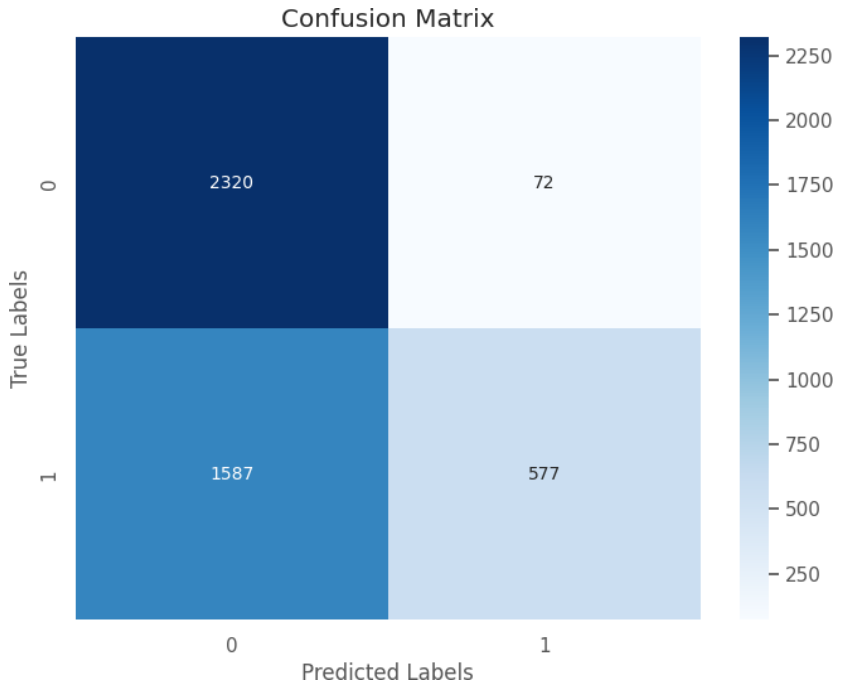
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



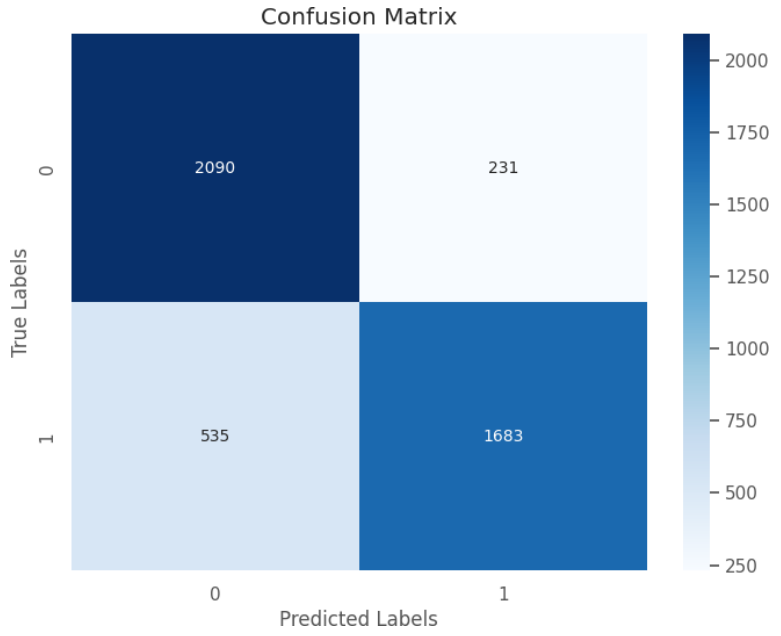
12.1.4 SVM

12.1.4.1 Model Results

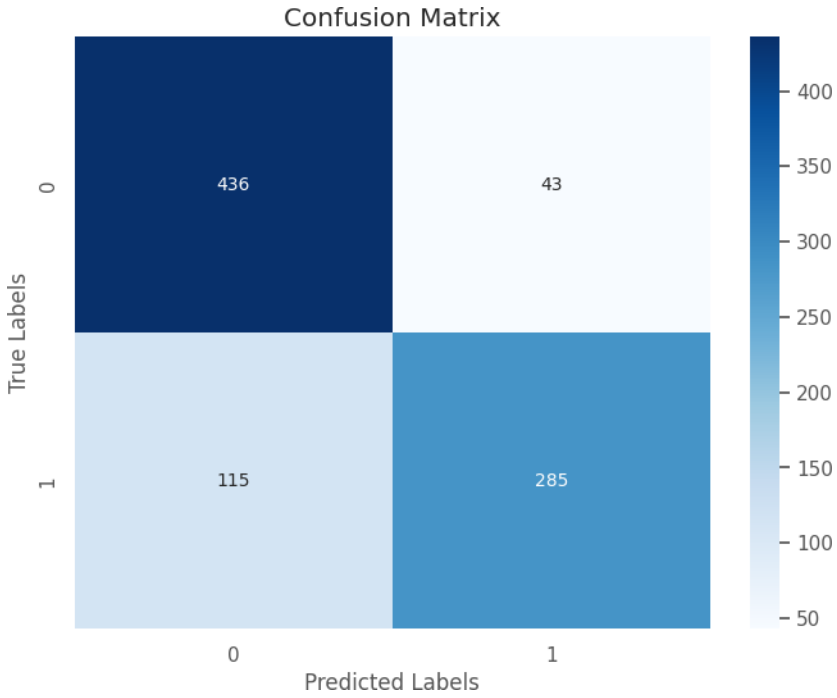
Train Years	Test Year	Precision		Recall		F1-Score		Accuracy
		0	1	0	1	0	1	
Year1 + Year 2	Year3	0.80	0.88	0.90	0.76	0.85	0.81	0.83
Year2 + Year 3	Year1	0.79	0.87	0.91	0.71	0.85	0.78	0.82
Year1 + Year 3	Year2	0.64	0.74	0.84	0.48	0.73	0.58	0.67

12.1.4.2 Confusion Matrix

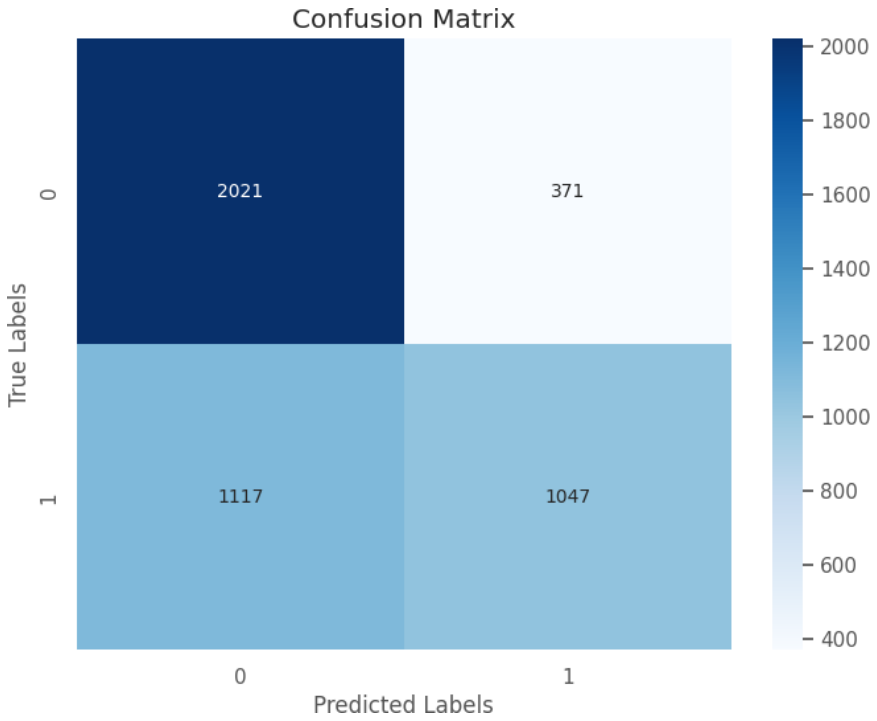
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



12.2 Random Undersampling

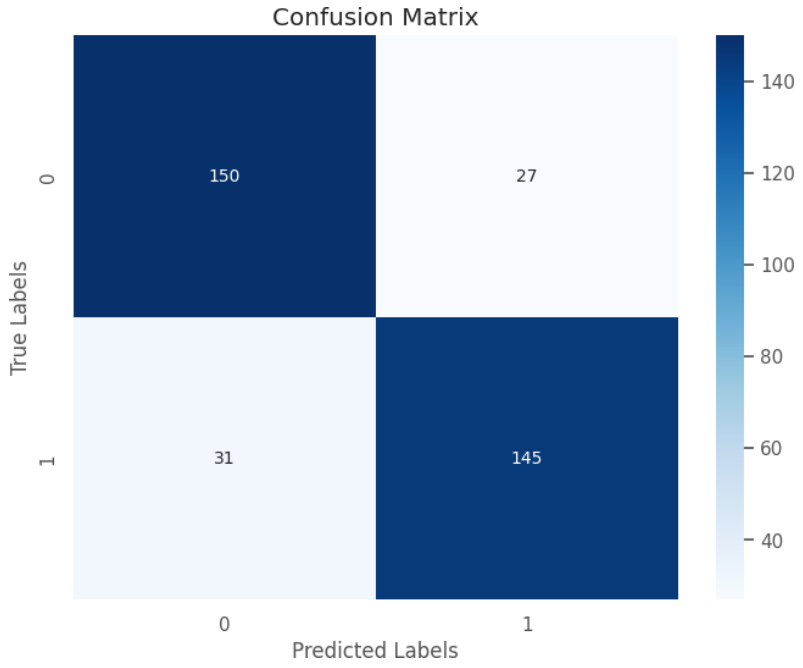
12.2.1 XGBoost

12.2.1.1 Model Results for XGBoost

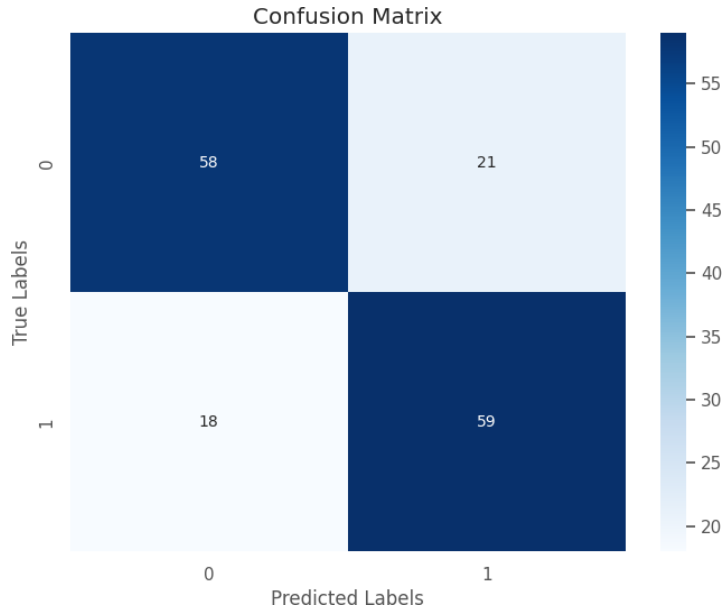
Train Years	Test Year	Precision		Recall		F1-Score		Accuracy
		0	1	0	1	0	1	
Year1 + Year 2	Year3	0.83	0.84	0.85	0.82	0.84	0.83	0.84
Year2 + Year 3	Year1	0.76	0.74	0.73	0.77	0.75	0.75	0.75
Year1 + Year 3	Year2	0.66	0.63	0.65	0.64	0.65	0.64	0.65

12.2.1.2 Confusion Matrix

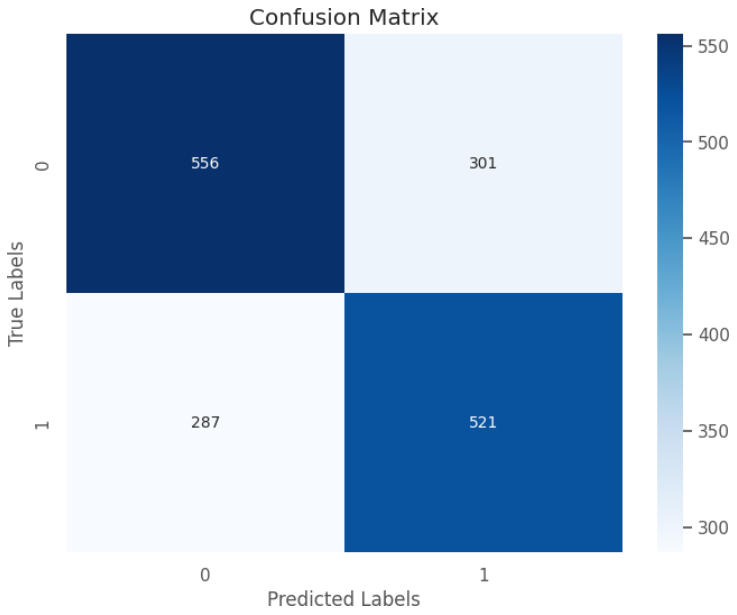
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



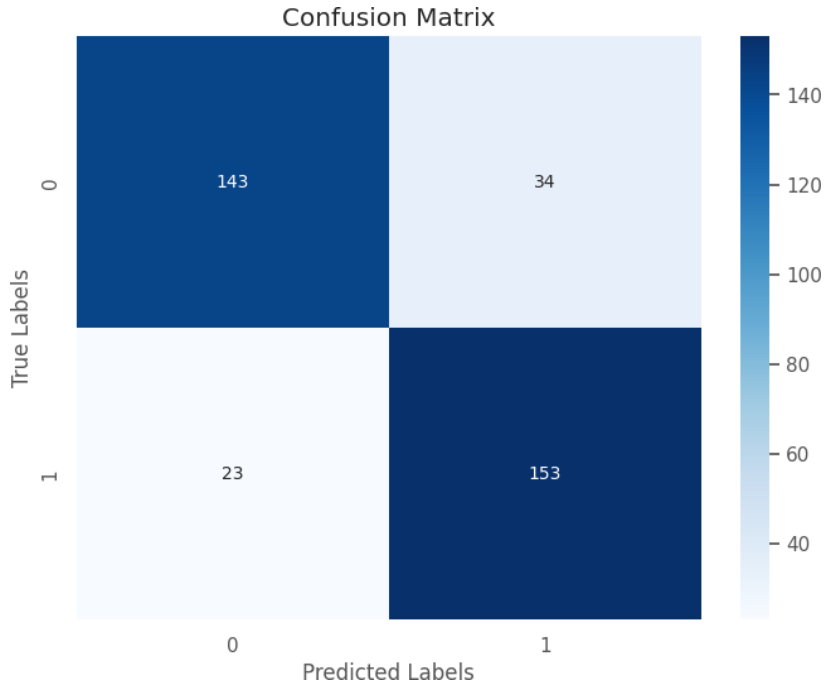
12.2.2 Bagging(Bootstrap Aggregation)

12.2.2.1 Model Results

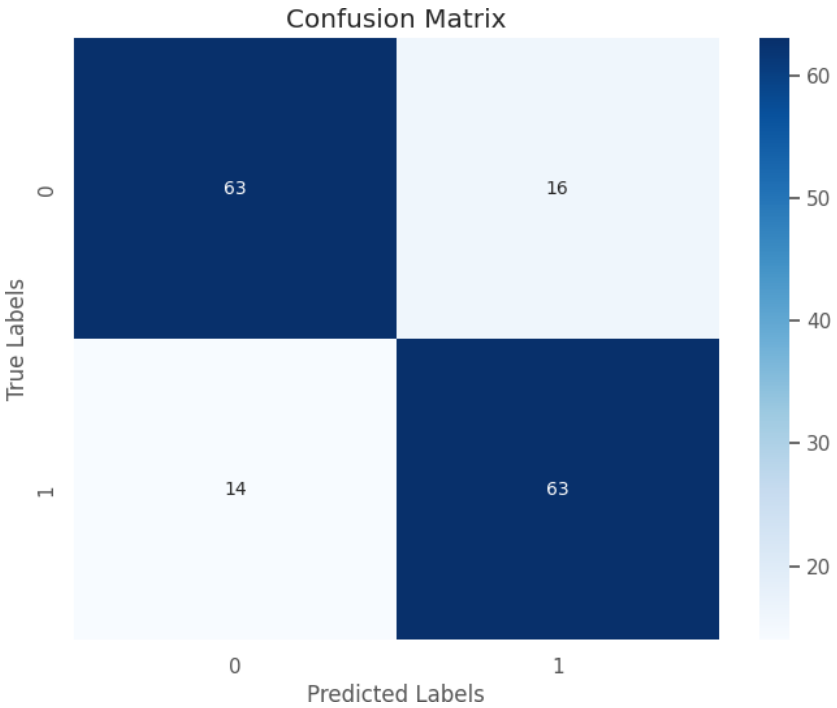
Train Years	Test Year	Precision		Recall		F1-Score		Accuracy
		0	1	0	1	0	1	
Year1 + Year 2	Year3	0.86	0.82	0.81	0.87	0.83	0.84	0.84
Year2 + Year 3	Year1	0.82	0.80	0.80	0.82	0.81	0.81	0.81
Year1 + Year 3	Year2	0.74	0.66	0.63	0.77	0.68	0.71	0.70

12.2.2.2 Confusion Matrix

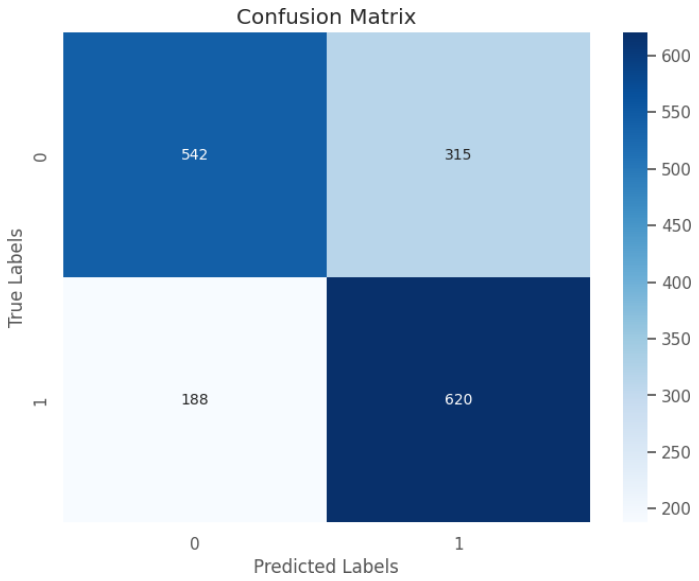
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



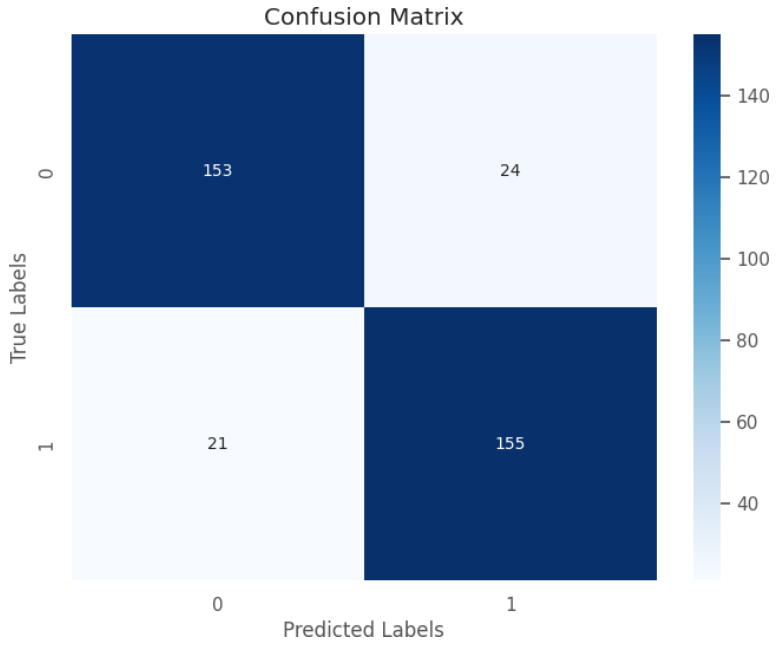
12.2.3 Random Forest

12.2.3.1 Model Results

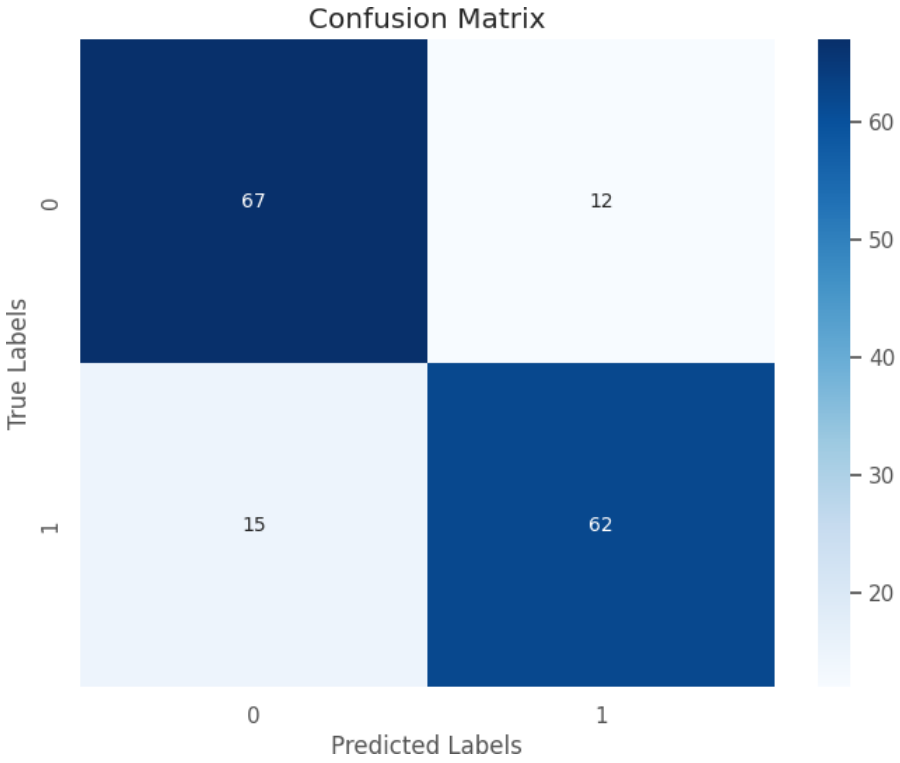
Train Years	Test Year	Precision		Recall		F1-Score		Accuracy
		0	1	0	1	0	1	
Year1 + Year 2	Year3	0.88	0.87	0.86	0.88	0.87	0.87	0.87
Year2 + Year 3	Year1	0.82	0.84	0.85	0.81	0.83	0.82	0.83
Year1 + Year 3	Year2	0.75	0.70	0.69	0.75	0.72	0.72	0.72

12.2.2.2 Confusion Matrix

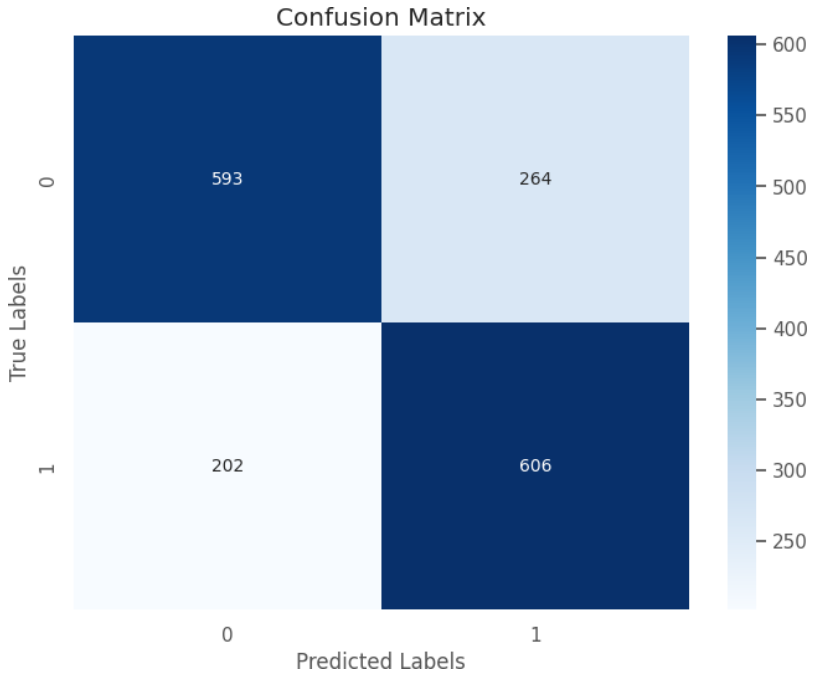
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



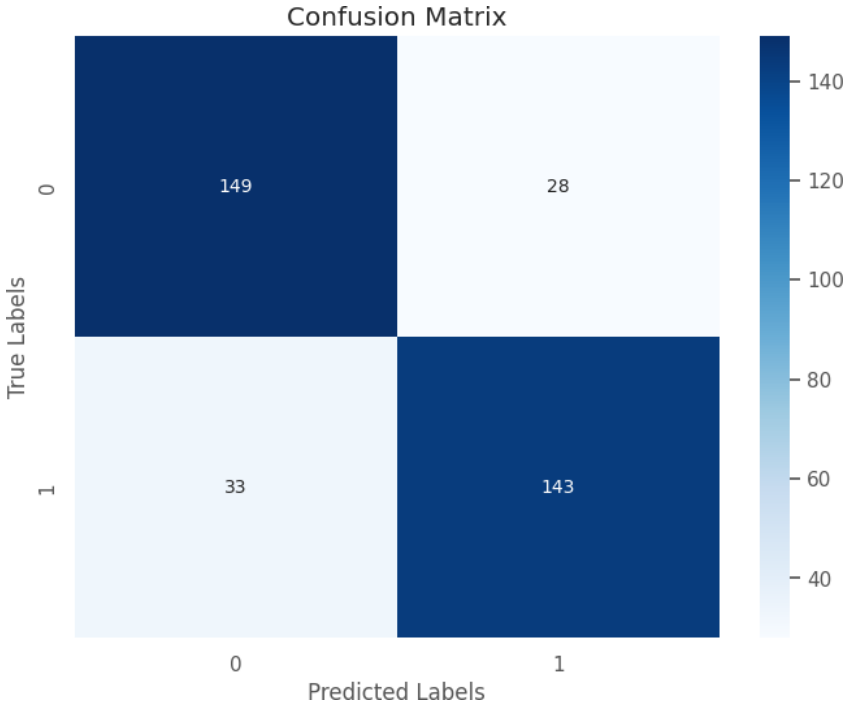
12.2.4 SVM

12.2.4.1 Model Results

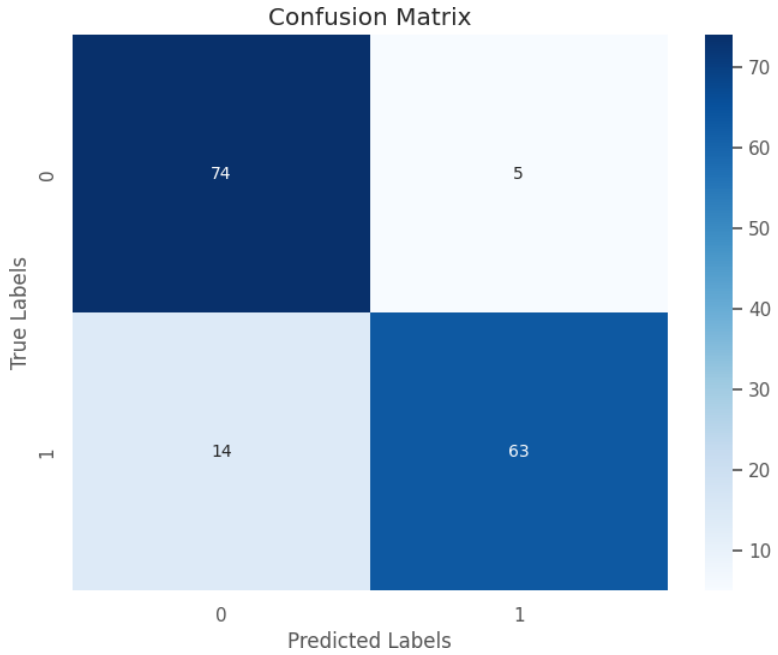
Train Years	Test Year	Precision		Recall		F1-Score		Accuracy
		0	1	0	1	0	1	
Year1 + Year 2	Year3	0.82	0.84	0.84	0.81	0.83	0.82	0.83
Year2 + Year 3	Year1	0.84	0.93	0.94	0.82	0.89	0.87	0.88
Year1 + Year 3	Year2	0.71	0.68	0.70	0.70	0.70	0.69	0.70

12.2.4.2 Confusion Matrix

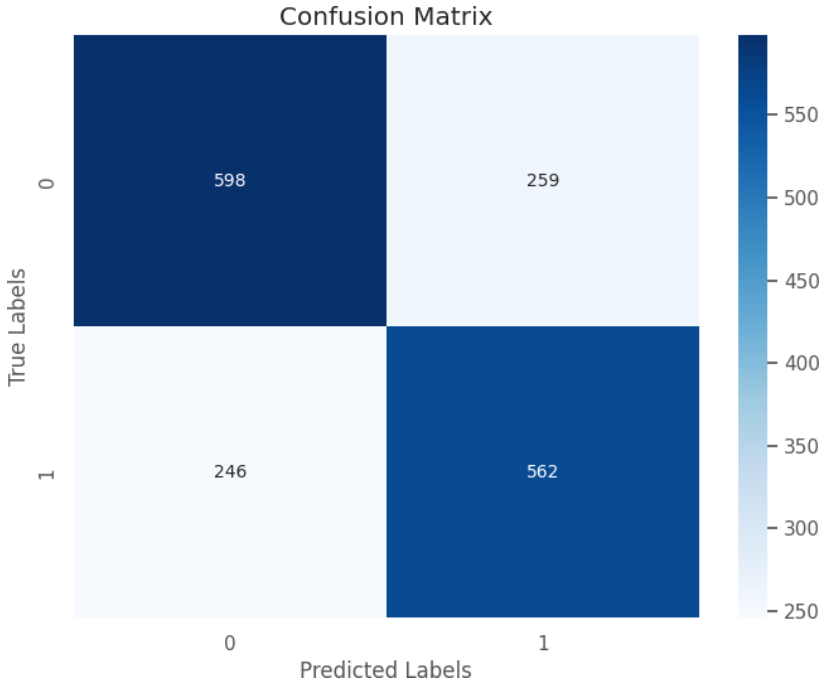
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



12.3 SMOTE

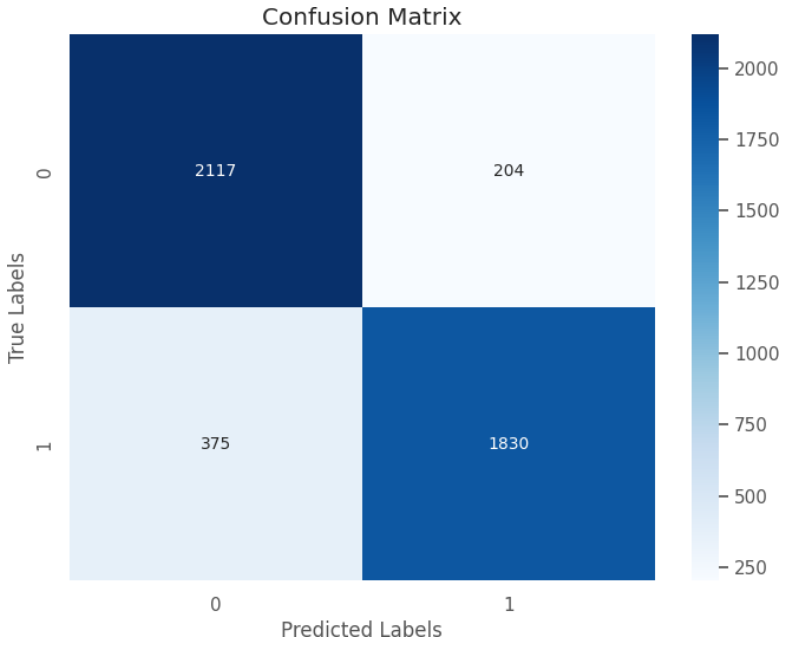
12.3.1 XGBoost

12.3.1.1 Model Results for XGBoost

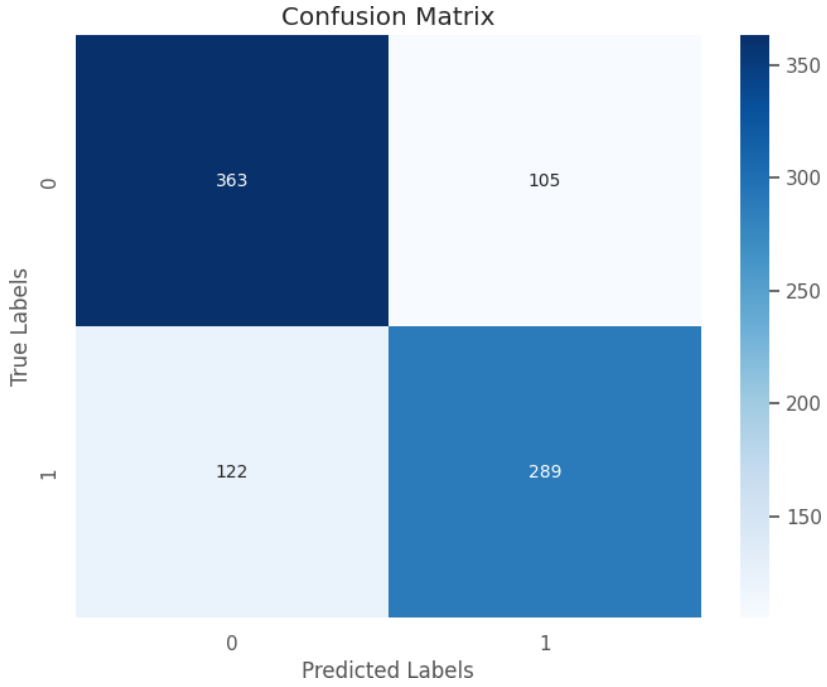
Train Years	Test Year	Precision		Recall		F1-Score		Accuracy
		0	1	0	1	0	1	
Year1 + Year 2	Year3	0.85	0.91	0.91	0.83	0.88	0.86	0.87
Year2 + Year 3	Year1	0.75	0.73	0.78	0.70	0.76	0.72	0.74
Year1 + Year 3	Year2	0.68	0.79	0.88	0.52	0.77	0.63	0.71

12.3.1.2 Confusion Matrix:

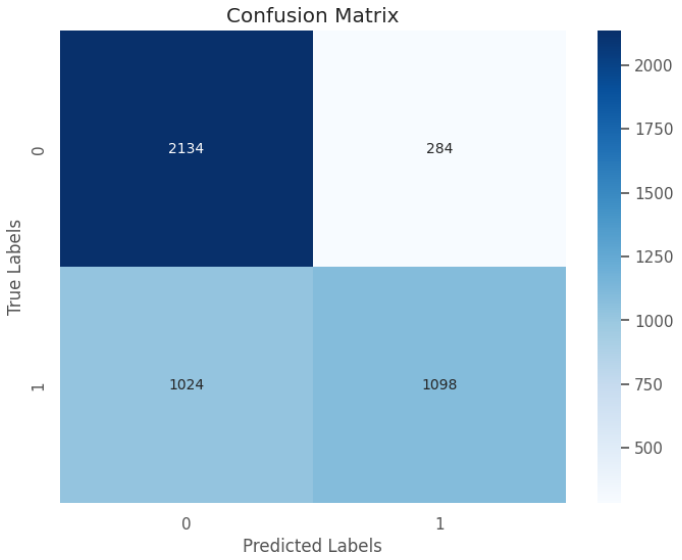
Year3 as test year



Year1 as test year



Year2 as test year



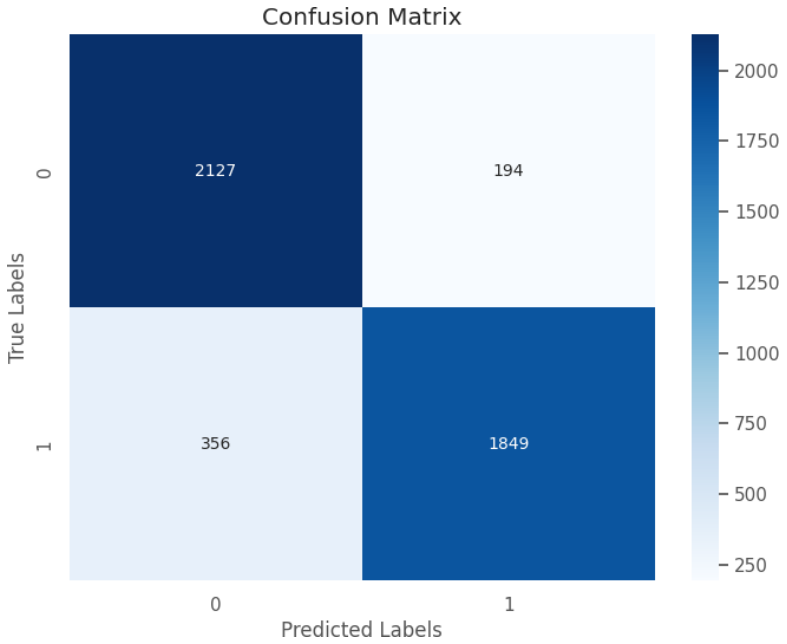
12.3.2 Bagging(Bootstrap Aggregation)

12.3.2.1 Model Results

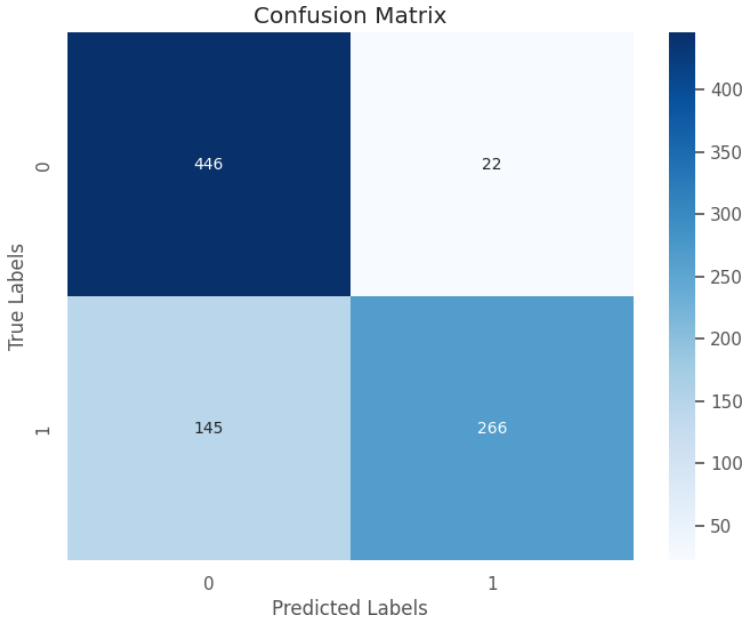
Train Years	Test Year	Precision		Recall		F1-Score		Accuracy
		0	1	0	1	0	1	
Year1 + Year 2	Year3	0.86	0.91	0.92	0.84	0.89	0.87	0.88
Year2 + Year 3	Year1	0.75	0.92	0.95	0.65	0.84	0.76	0.81
Year1 + Year 3	Year2	0.66	0.81	0.90	0.59	0.76	0.59	0.70

12.3.2.2 Confusion Matrix

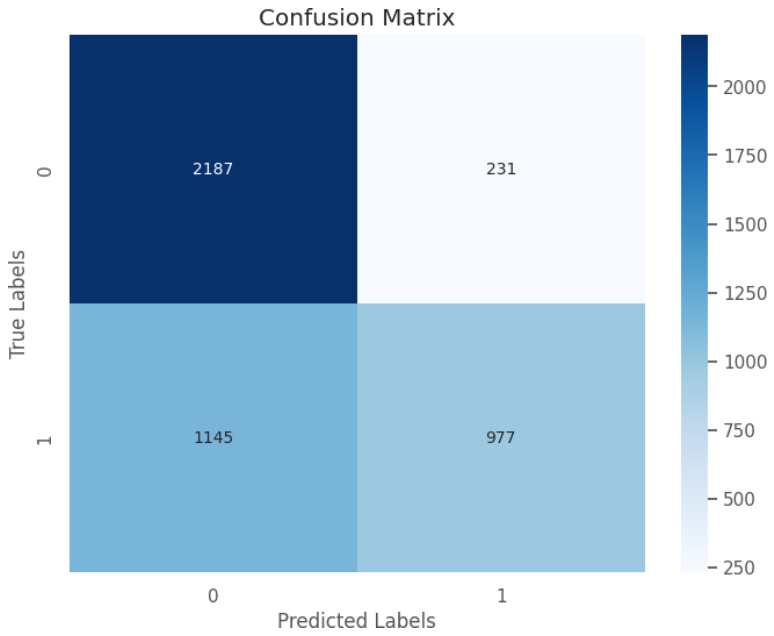
Year 3 As Testing year



Year 1 As Testing year



Year 2 As Testing year



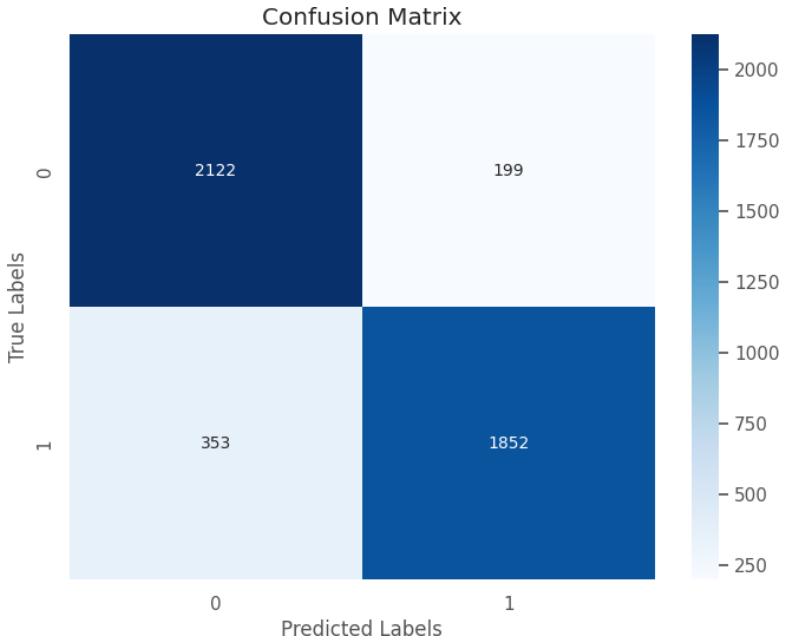
12.3.3 Random Forest

12.3.2.1 Model Results

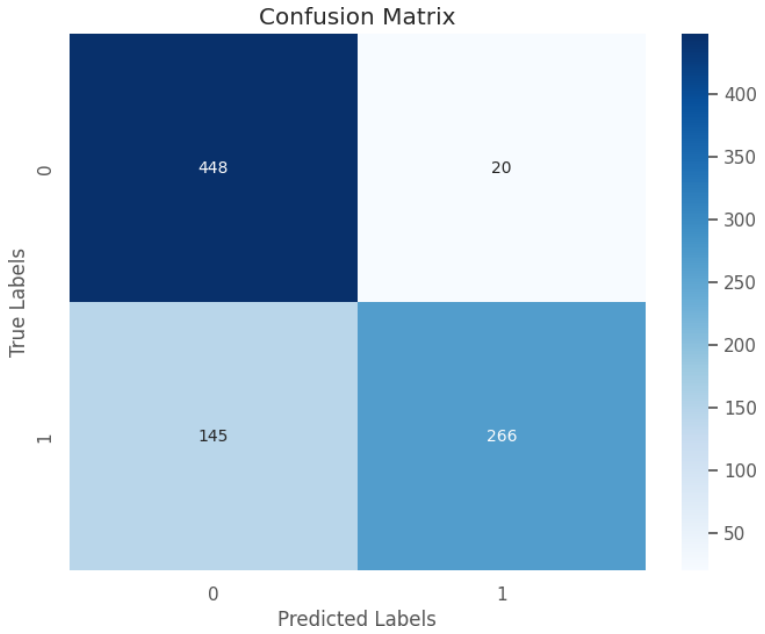
Train Years	Test Year	Precision		Recall		F1-Score		Accuracy
		0	1	0	1	0	1	
Year1 + Year 2	Year3	0.86	0.90	0.91	0.84	0.88	0.87	0.88
Year2 + Year 3	Year1	0.76	0.93	0.96	0.65	0.84	0.76	0.81
Year1 + Year 3	Year2	0.65	0.83	0.92	0.43	0.76	0.57	0.69

12.3.2.3 Confusion Matrix

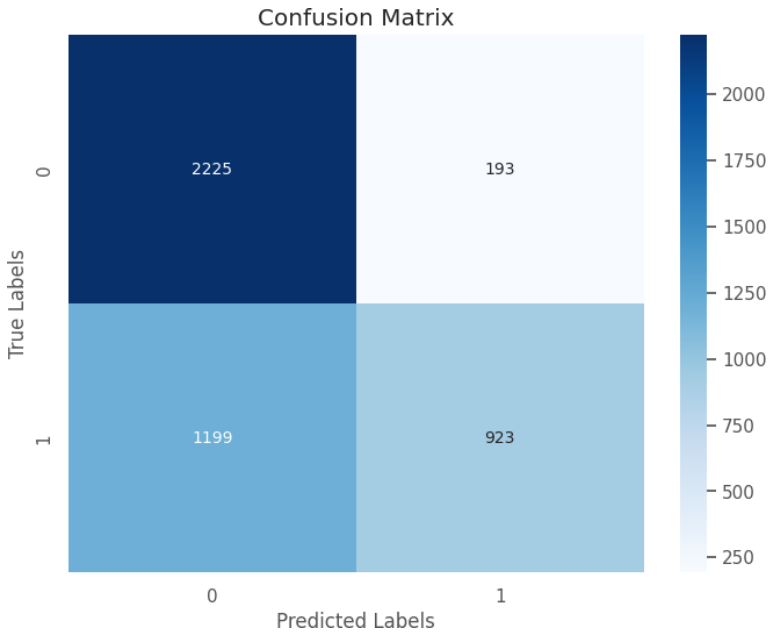
Year 3 As Testing Year



Year 1 As Testing Year



Year 2 As Testing Year



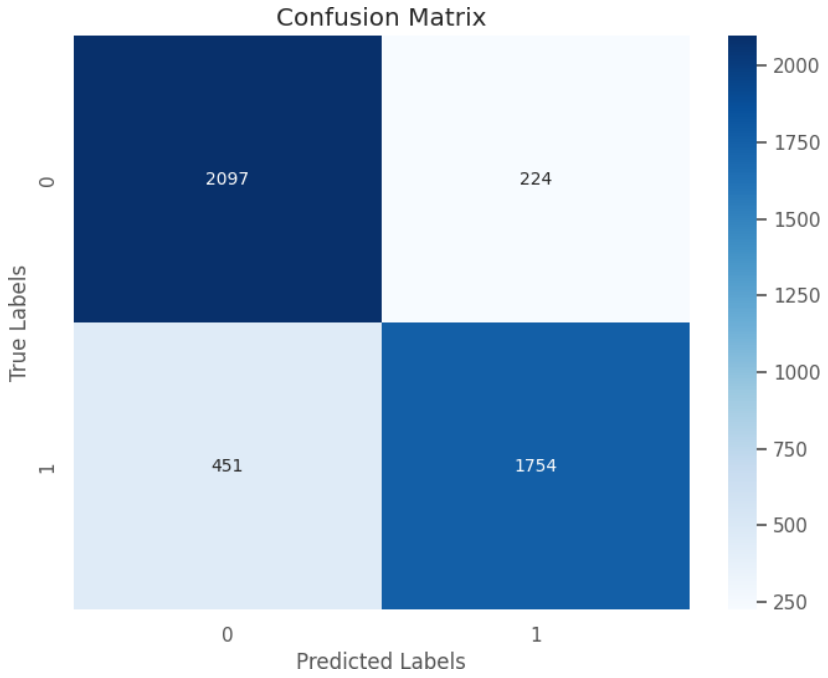
12.3.4 SVM

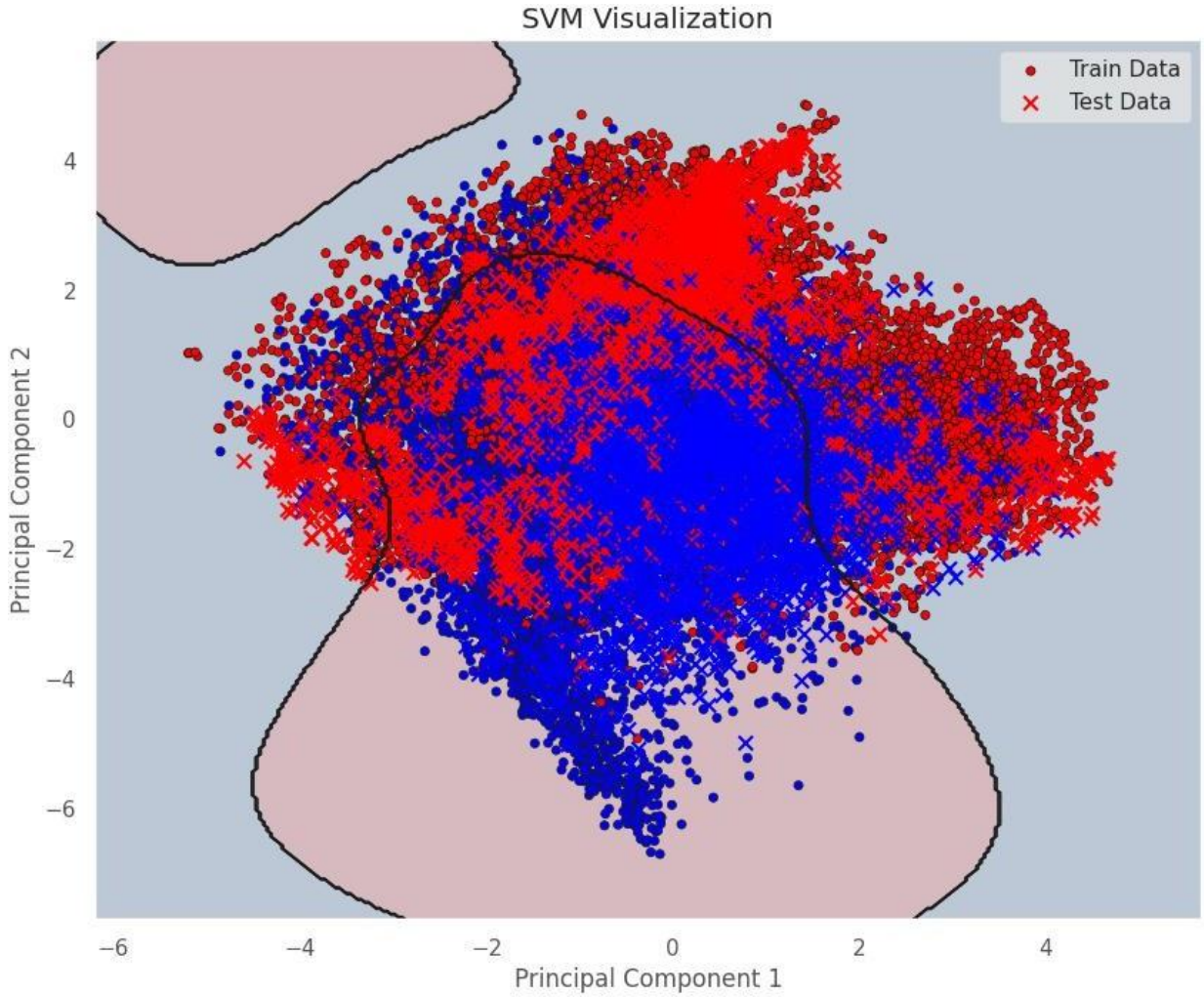
12.3.4.1 Model Results

Train Years	Test Year	Precision		Recall		F1-Score		Accuracy
		0	1	0	1	0	1	
Year1 + Year 2	Year3	0.82	0.89	0.90	0.80	0.86	0.84	0.85
Year2 + Year 3	Year1	0.80	0.89	0.92	0.74	0.86	0.81	0.84
Year1 + Year 3	Year2	0.66	0.75	0.85	0.51	0.75	0.61	0.69

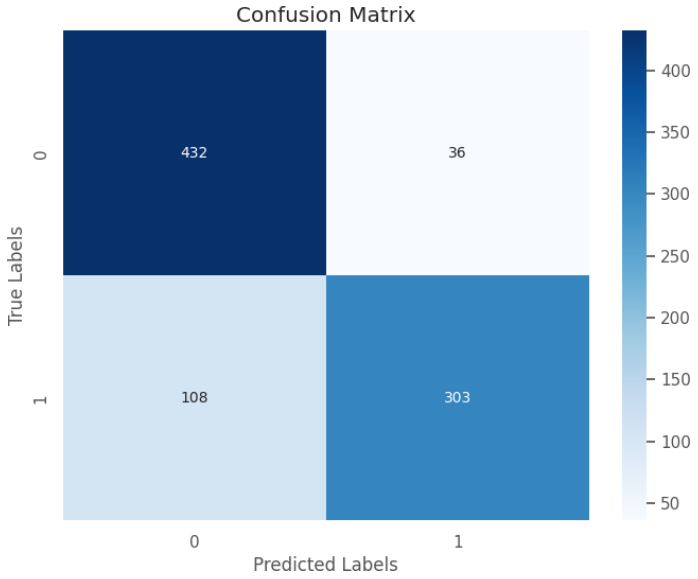
12.3.4.2 Confusion Matrix

Year 3 As Testing Year

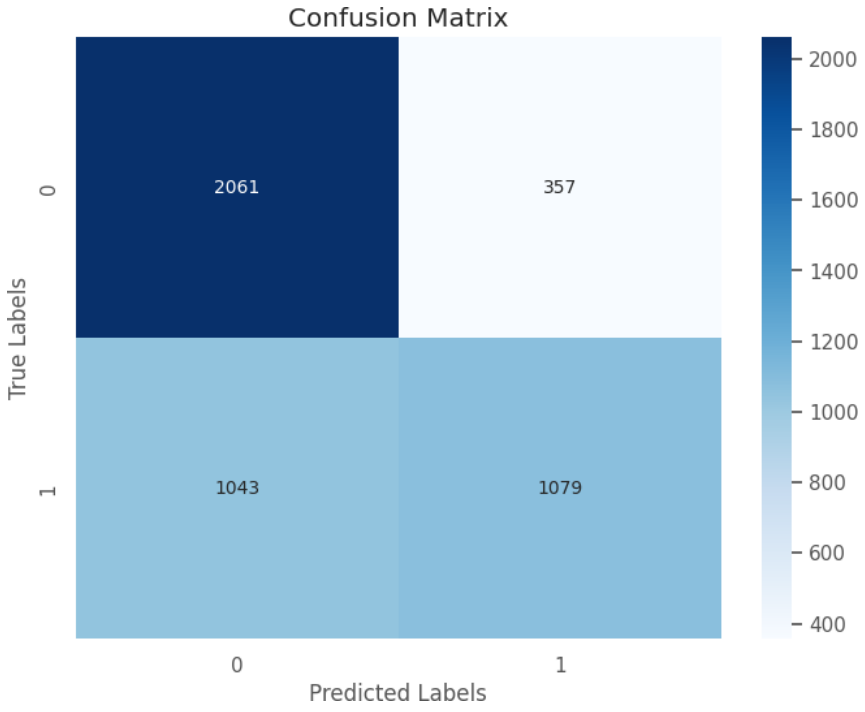


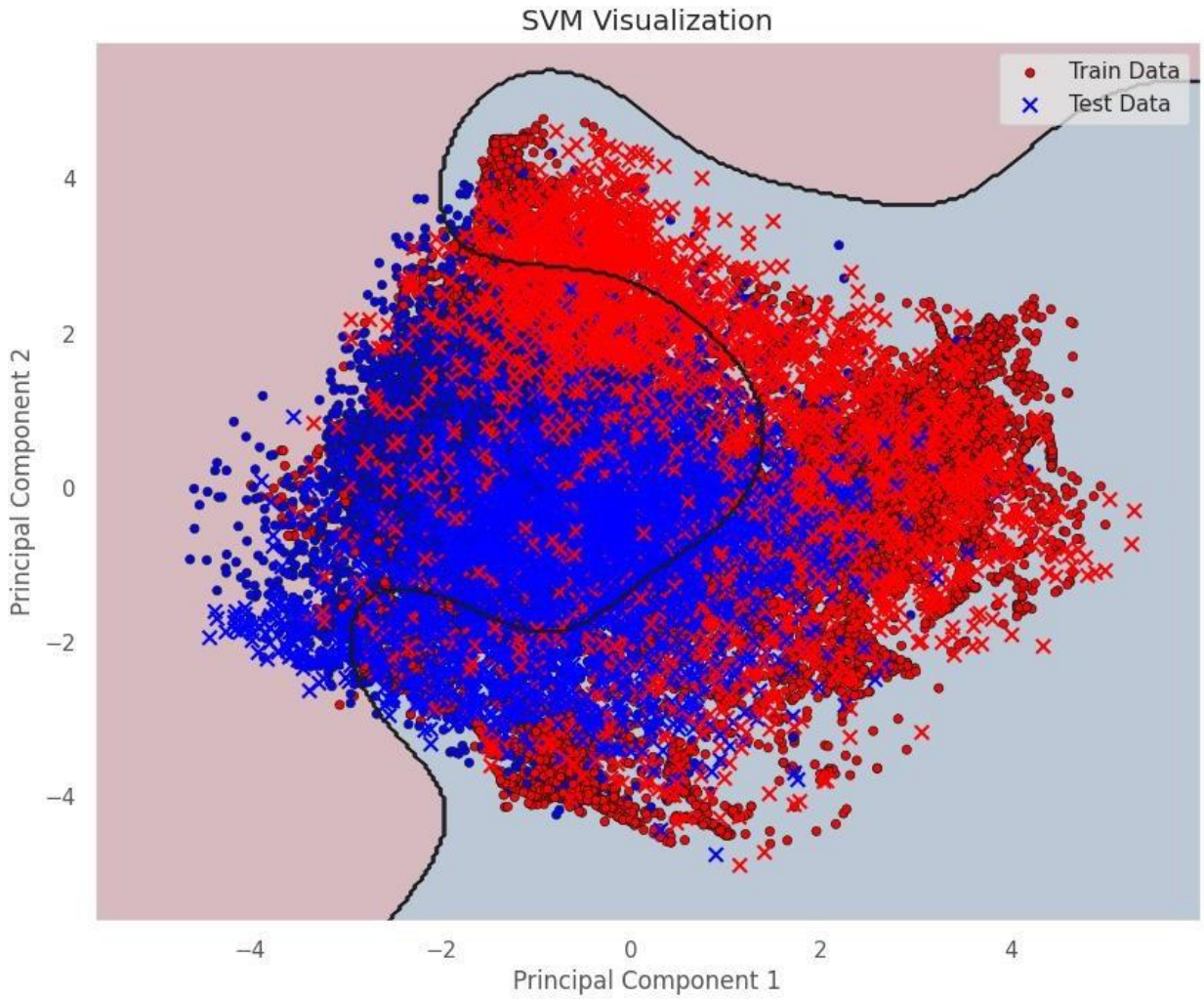

SVM Visualization
Train Data
Test Data
Principal Component 2
Principal Component 1
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4

Year 1 As Testing Year



Year 2 As Testing Year





Conclusion On Supervised Learning Implementation:

In this project, we explored crop classification using NDVI data for rice and cotton over three consecutive years (2021-2023). Various machine learning models, including XGBoost, Bagging, Random Forest, and SVM, were applied to address the classification challenge. The results demonstrated that ensemble methods generally performed well, especially when using hyperparameter tuning and class balancing techniques like Random Oversampling, Random Undersampling, and SMOTE.

Notably, feature reduction significantly impacted model performance, leading to a considerable drop in accuracy. This suggests that all 12 NDVI features contribute meaningfully to the classification task, and reducing the feature set compromises the models' ability to differentiate

between rice and cotton effectively. Additionally, attempts to apply time series augmentation methods also resulted in a decrease in accuracy, indicating that the temporal patterns in the original data are crucial for maintaining model performance.

These findings underscore the importance of preserving the full set of spectral features and respecting the natural time series structure of NDVI data for accurate crop classification. Future work may focus on developing more sophisticated augmentation techniques or exploring alternative feature selection strategies to improve model robustness without sacrificing accuracy.

13. Unsupervised Learning without PCA

1. Cluster Purity

Cluster purity measures the degree to which a cluster contains data points from a single true class.

2. Normalized Mutual Information (NMI)

NMI quantifies the amount of information shared between the clustering assignments and the true labels, normalized to account for the total information in the labels.

3. Silhouette Score

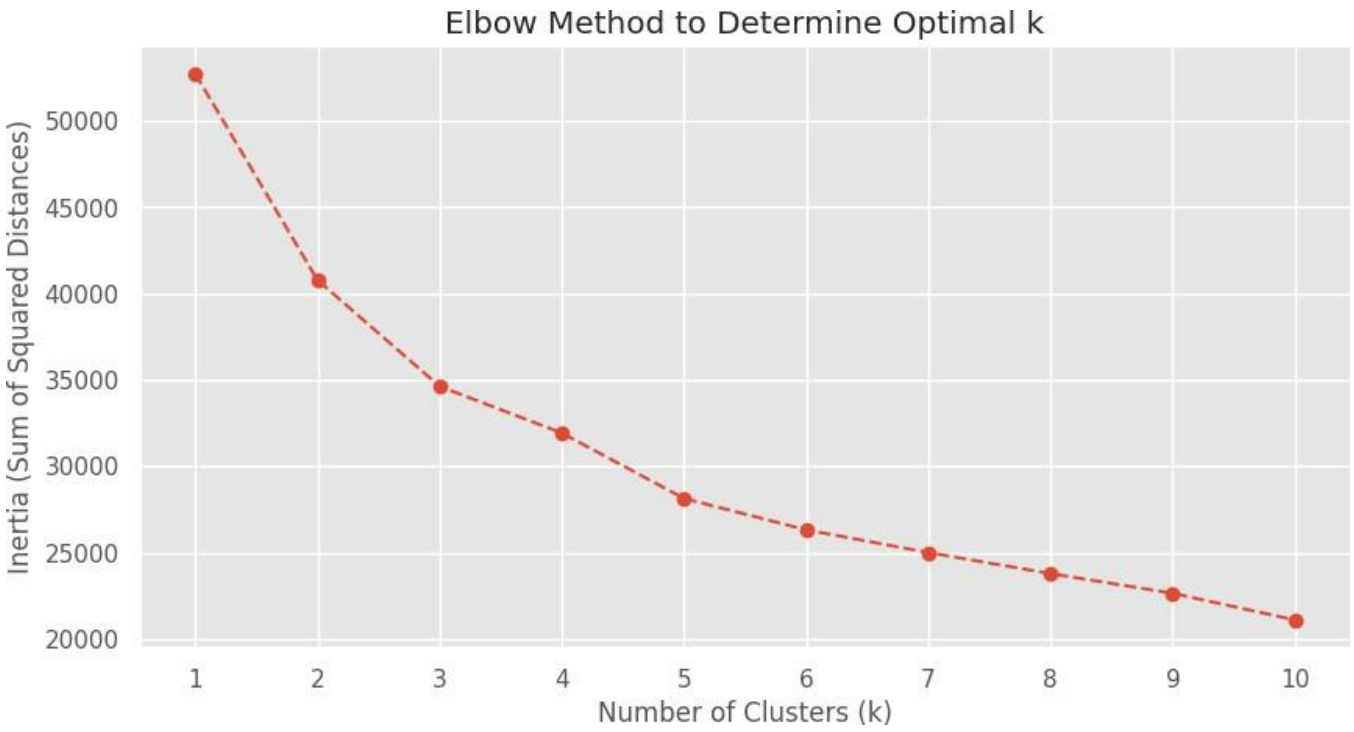
Silhouette score evaluates the quality of clustering by measuring how similar data points are to their own cluster compared to other clusters, ranging from -1 (poor clustering) to 1 (good clustering).

13.1 Year1

13.1.1 SMOTE:

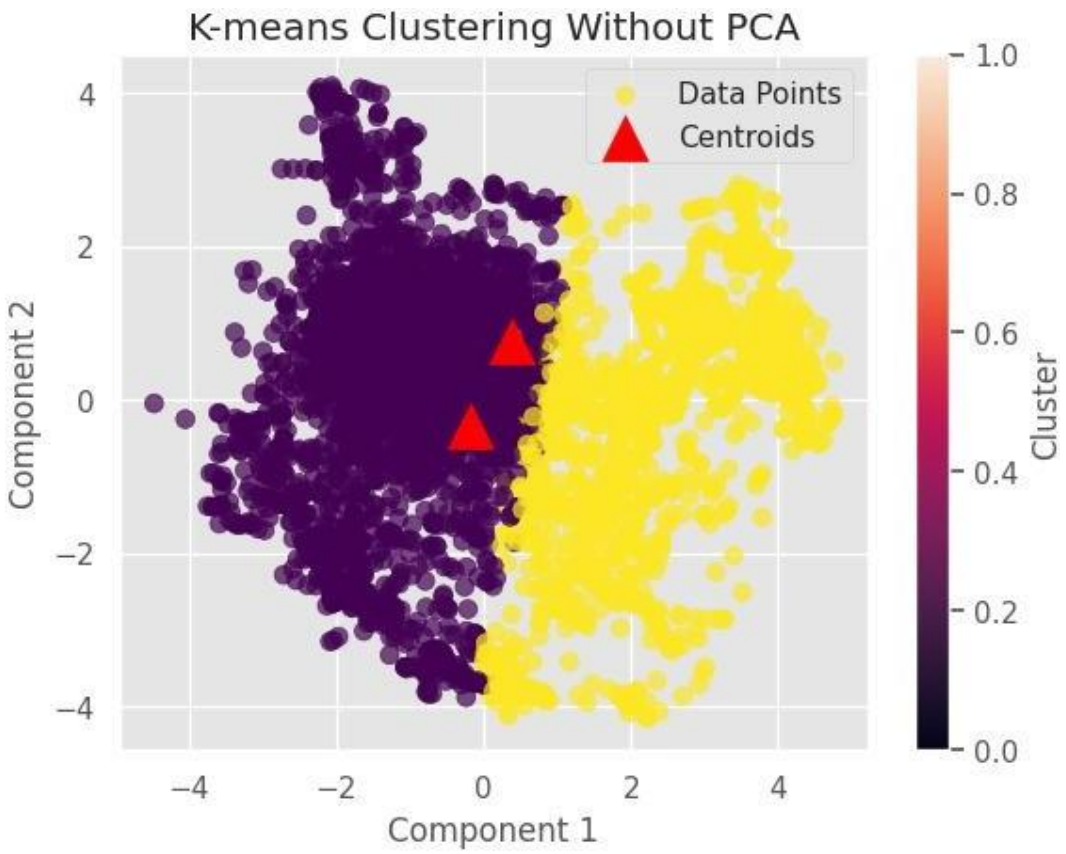
13.1.1.1 K Means Clustering:

Elbow method:



Elbow method shows k=3 but since our dataset is labelled and we are dealing with 2 classes, k = 2.

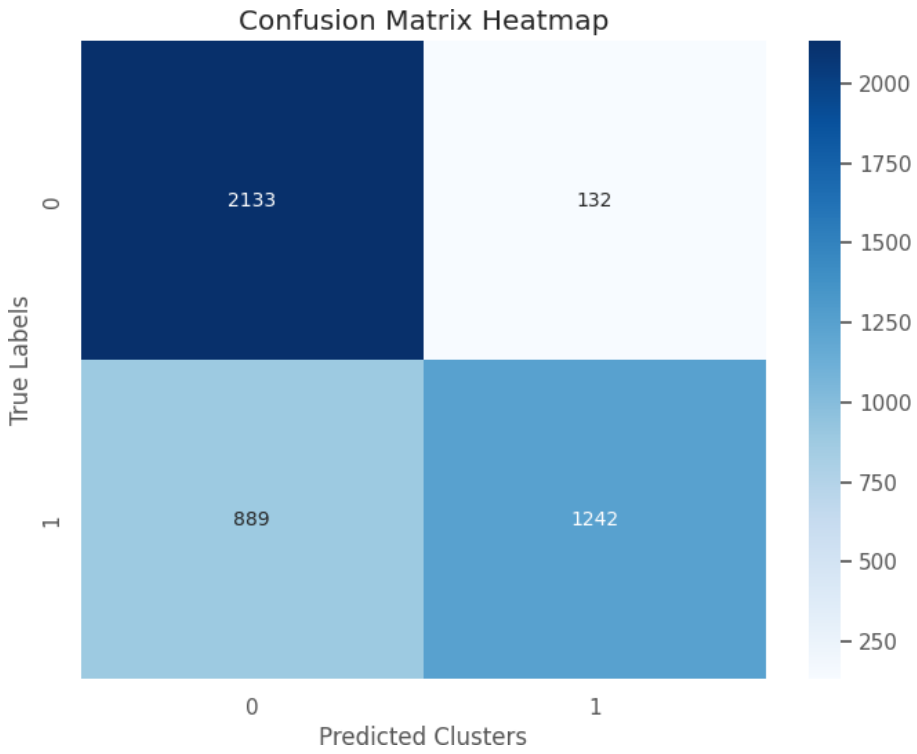
Visualization:



Evaluation Metrics:

Cluster Purity: 0.7677434030937216

Confusion matrix:

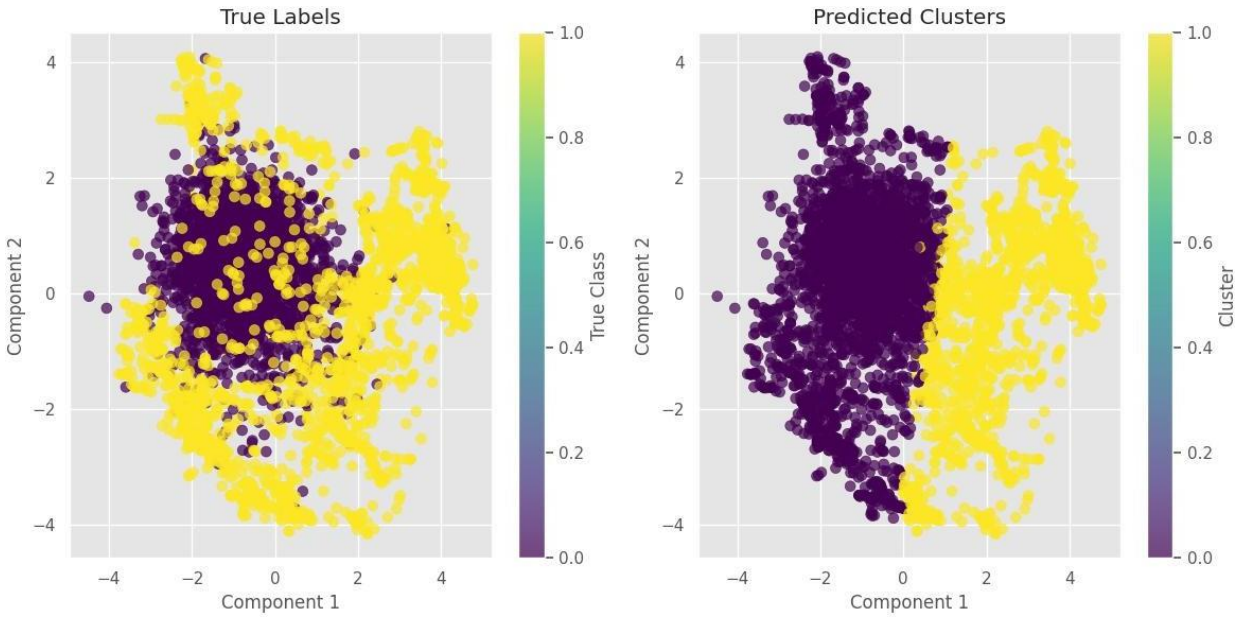


Other Metrics:

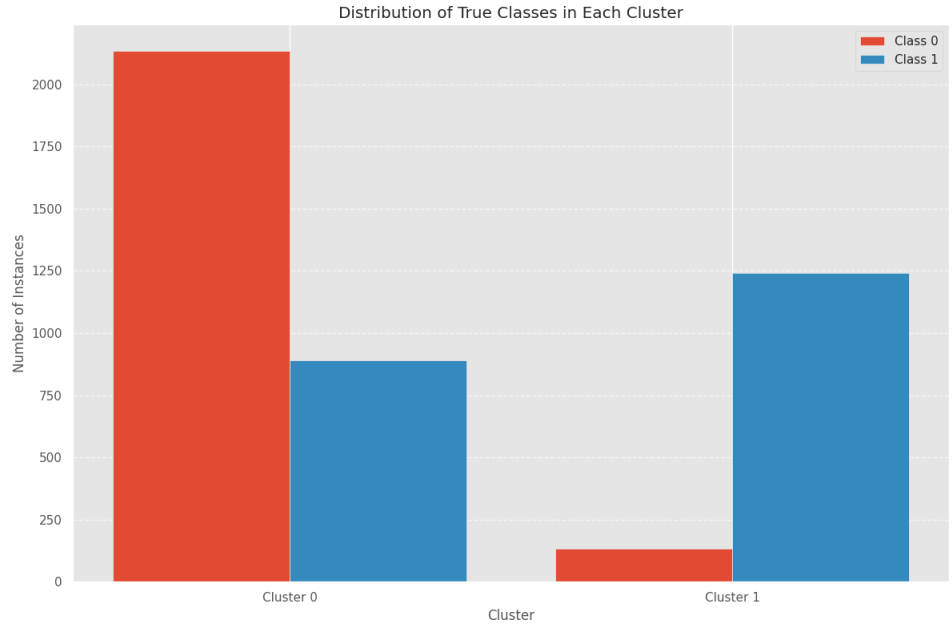
Normalized Mutual Information (NMI): 0.26992460623936787

silhouette_score: 0.2505446650723351

Comparison with true labels:



Class Distribution among clusters:

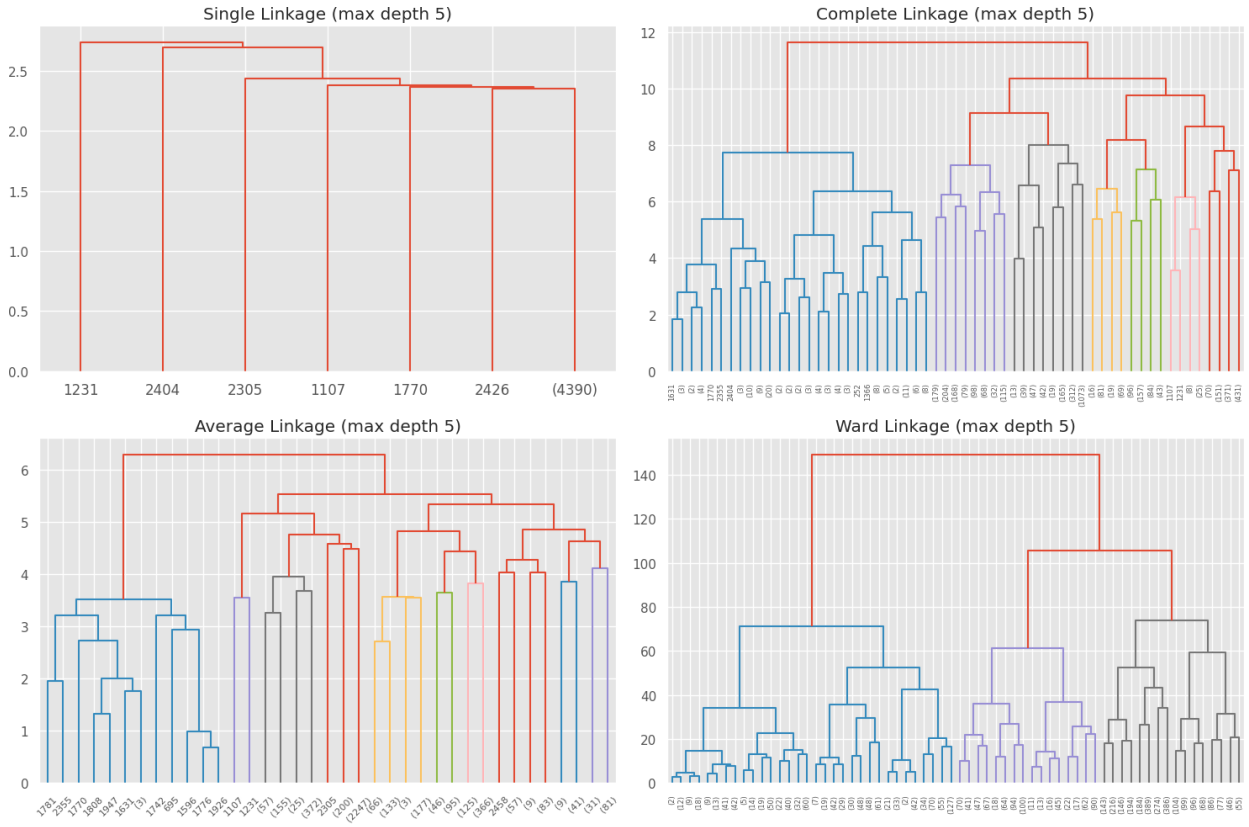


Cluster 0 has 2133 datapoints with label 0 and 889 datapoints with label 1.

Cluster 1 has 132 datapoints with label 0 and 1242 datapoints with label 1.

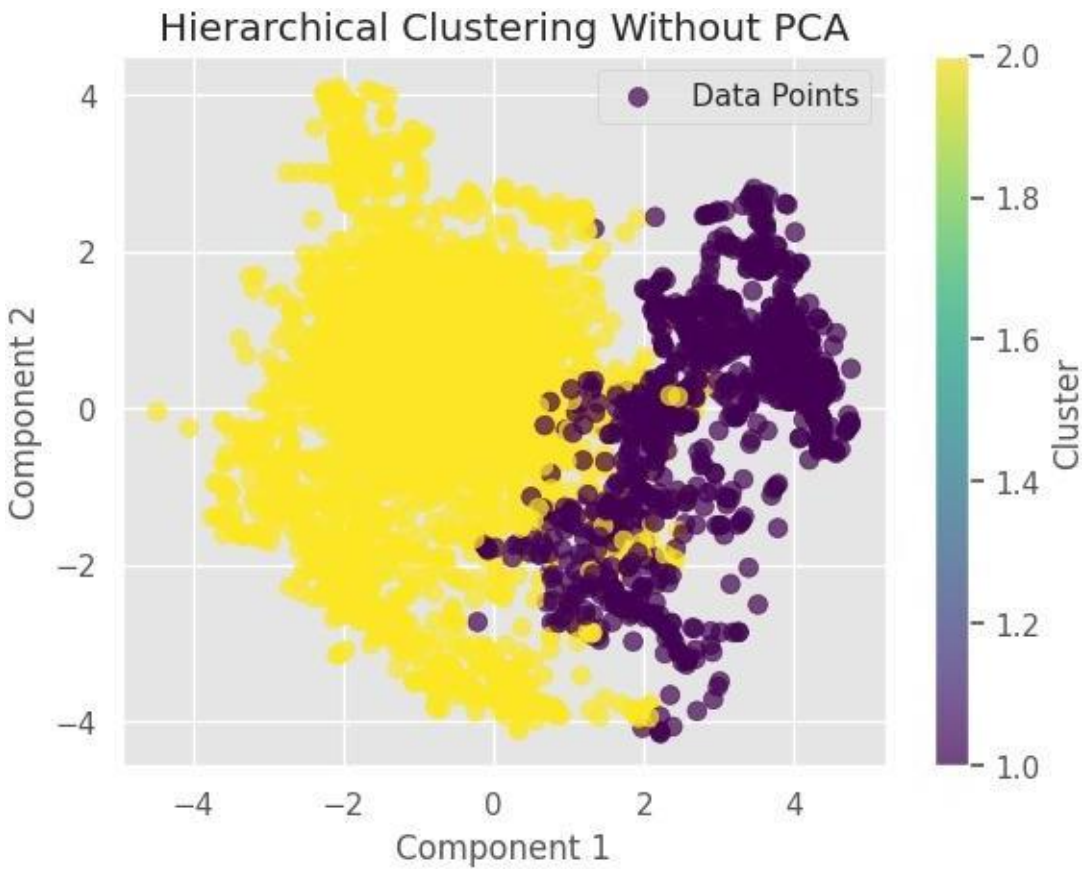
13.1.1.2 Hierarchical Clustering*:*

Dendrogram:



Ward Linkage:

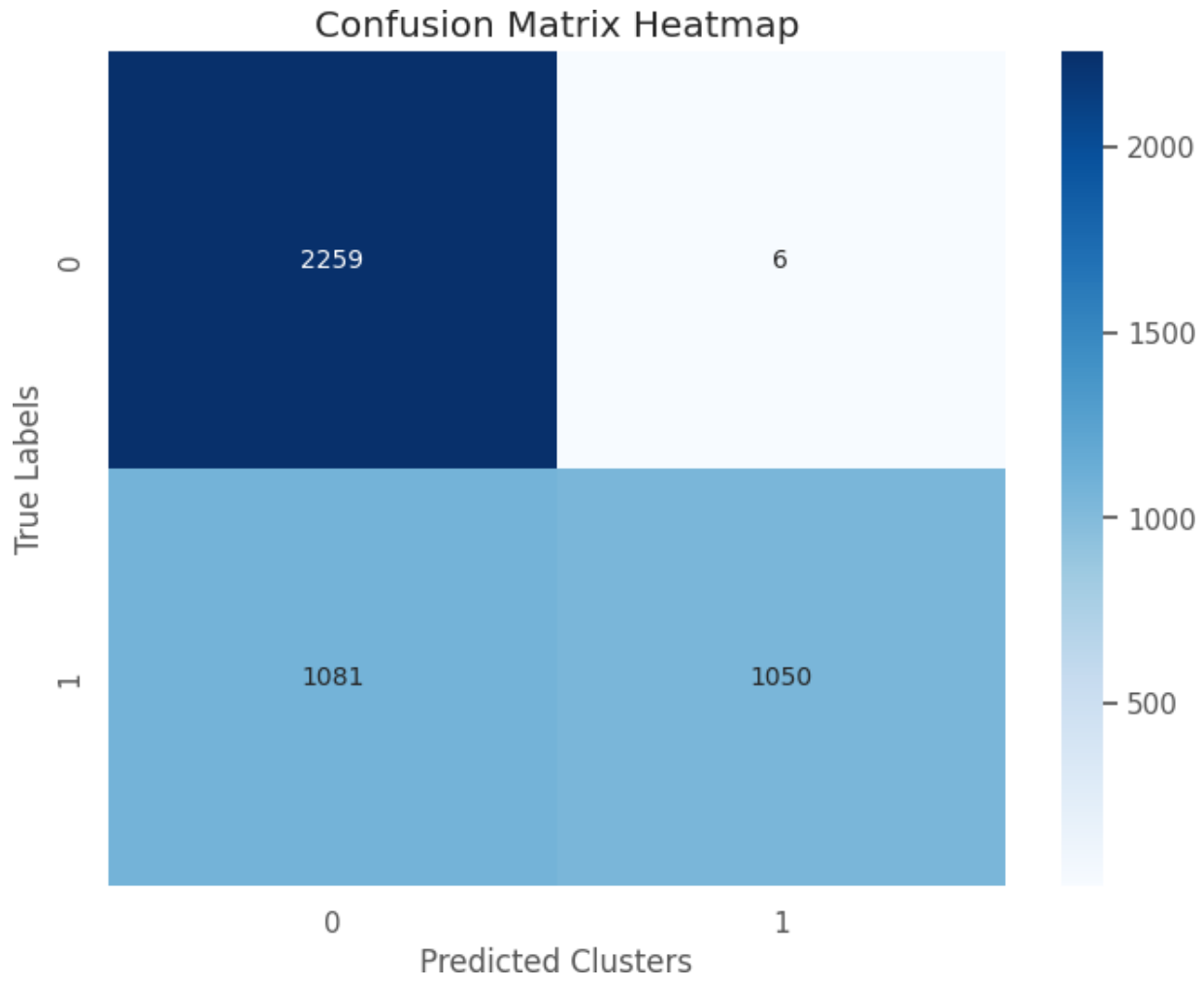
Visualization:



Evaluation Metrics:

Cluster Purity: 0.752729754322111

Confusion matrix:

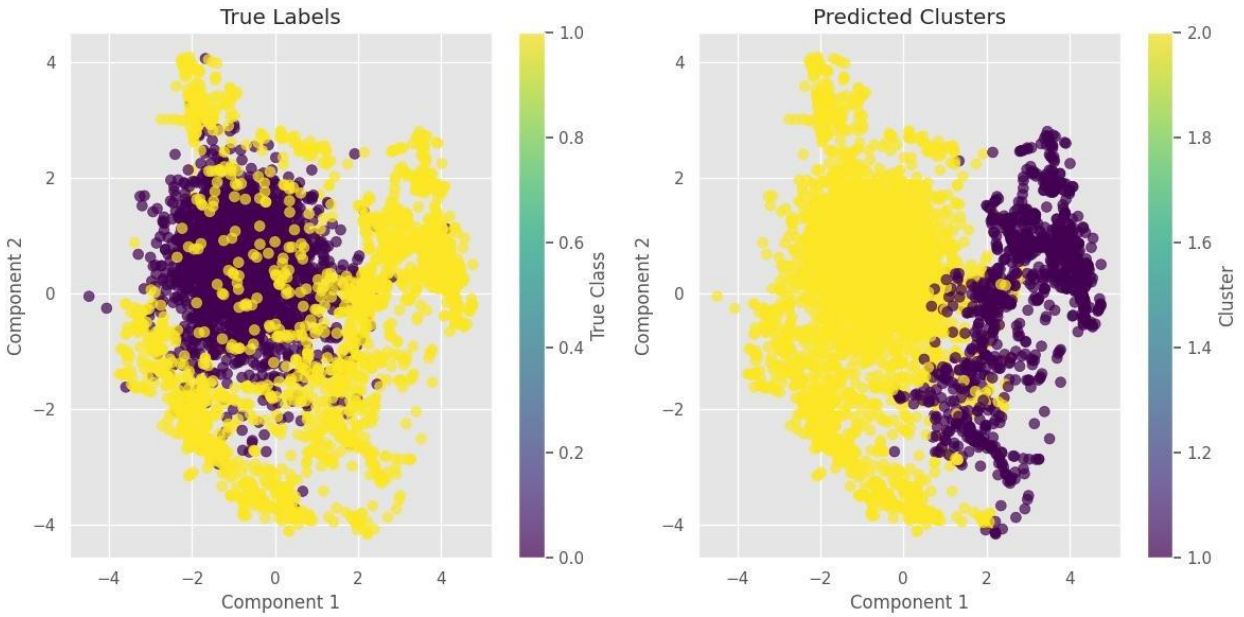


Other Metrics:

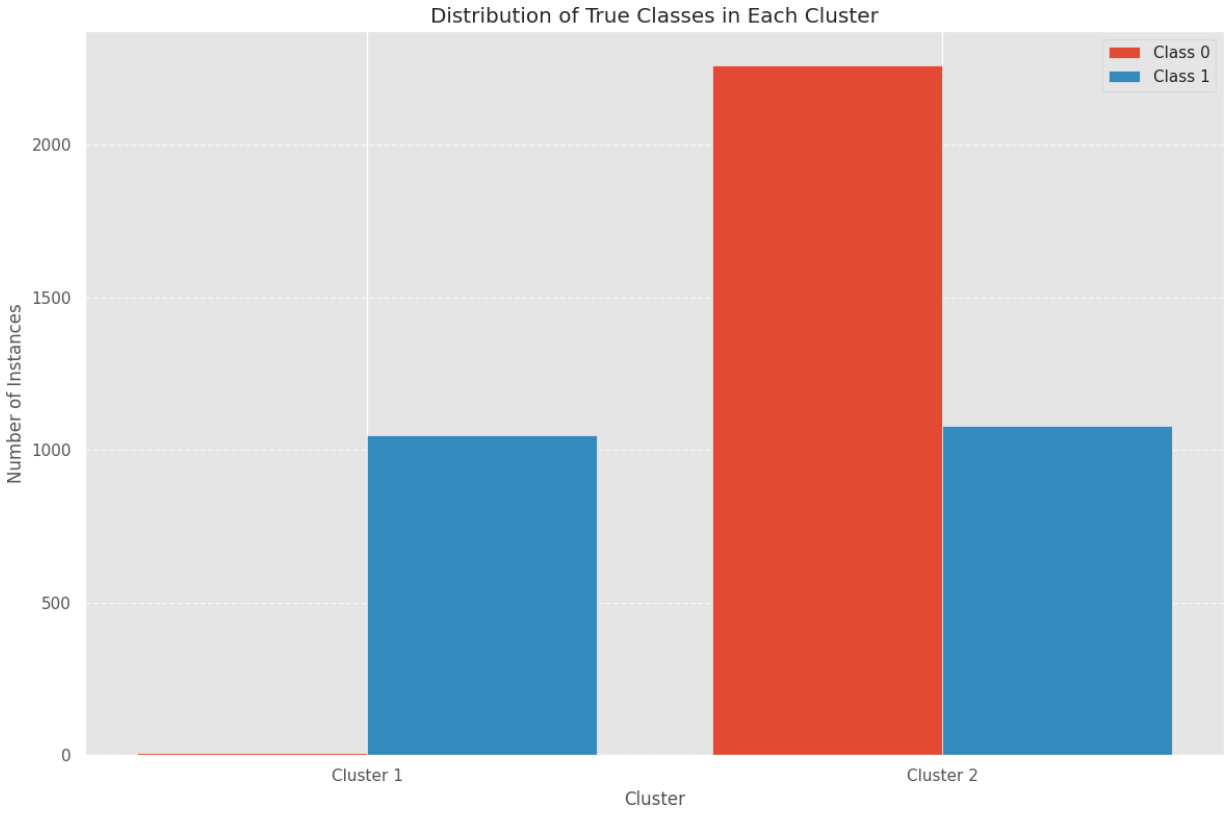
Normalized Mutual Information (NMI): 0.3310443431981989

silhouette_score: 0.25332418420807157

Comparison with true labels:



Class Distribution among clusters:

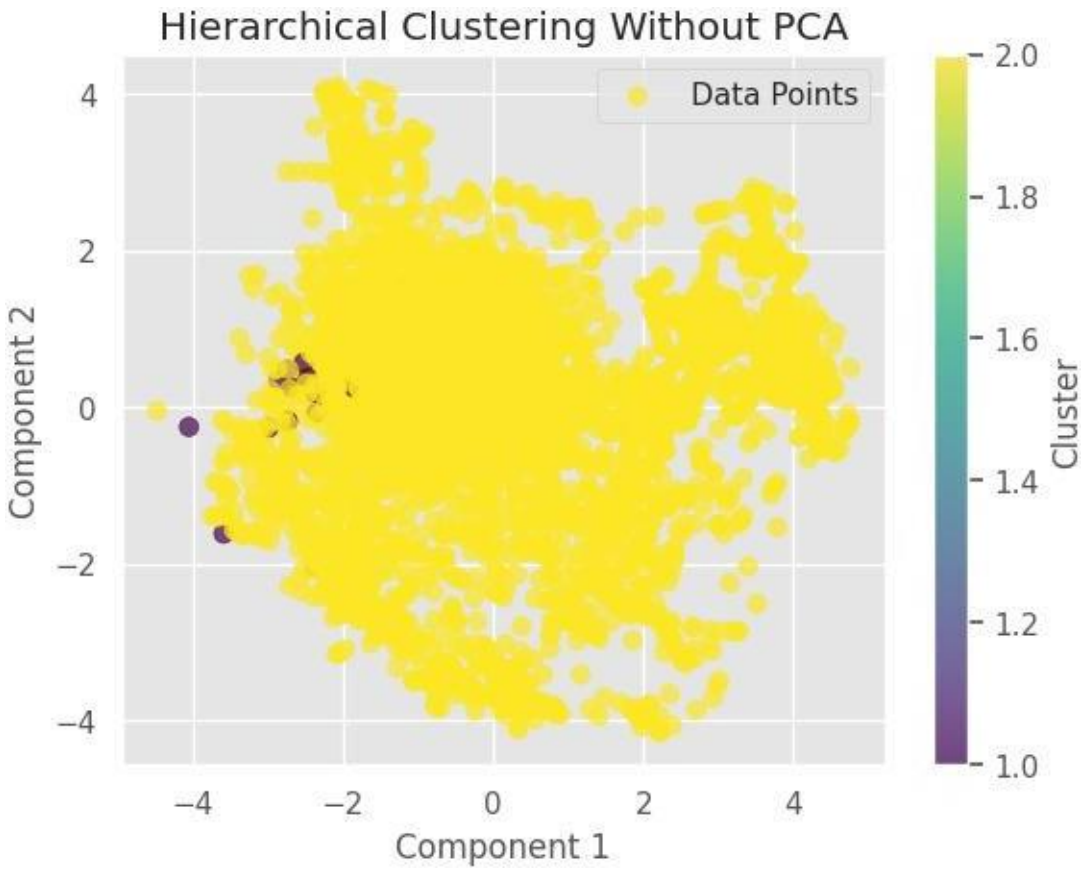


Cluster 1 has 6 datapoints with label 0 and 1050 datapoints with label 1.

Cluster 2 has 2259 datapoints with label 0 and 1081 datapoints with label 1.

Average Linkage:

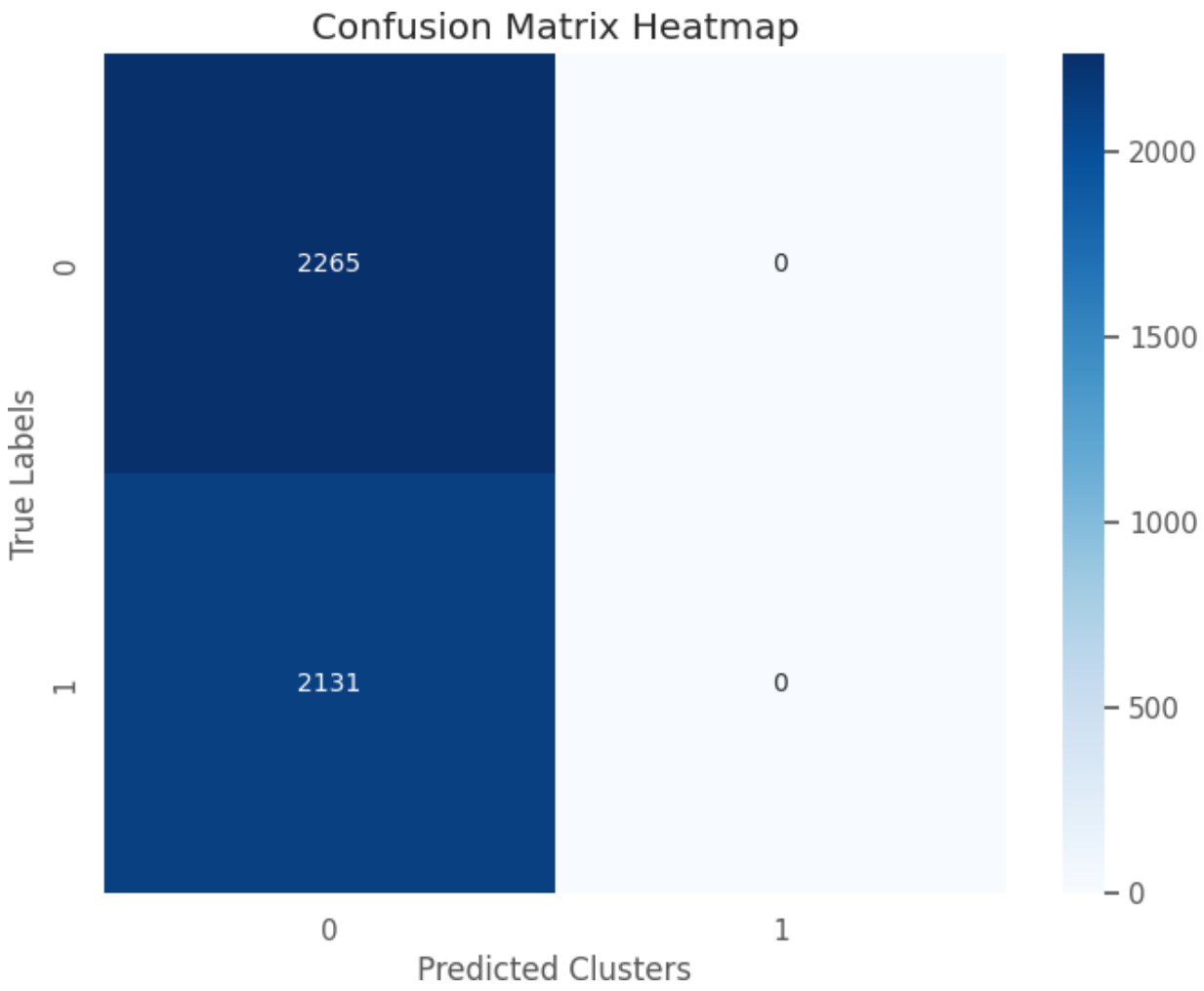
Visualization:



Evaluation Metrics:

Cluster Purity: 0.5152411282984531

Confusion matrix:

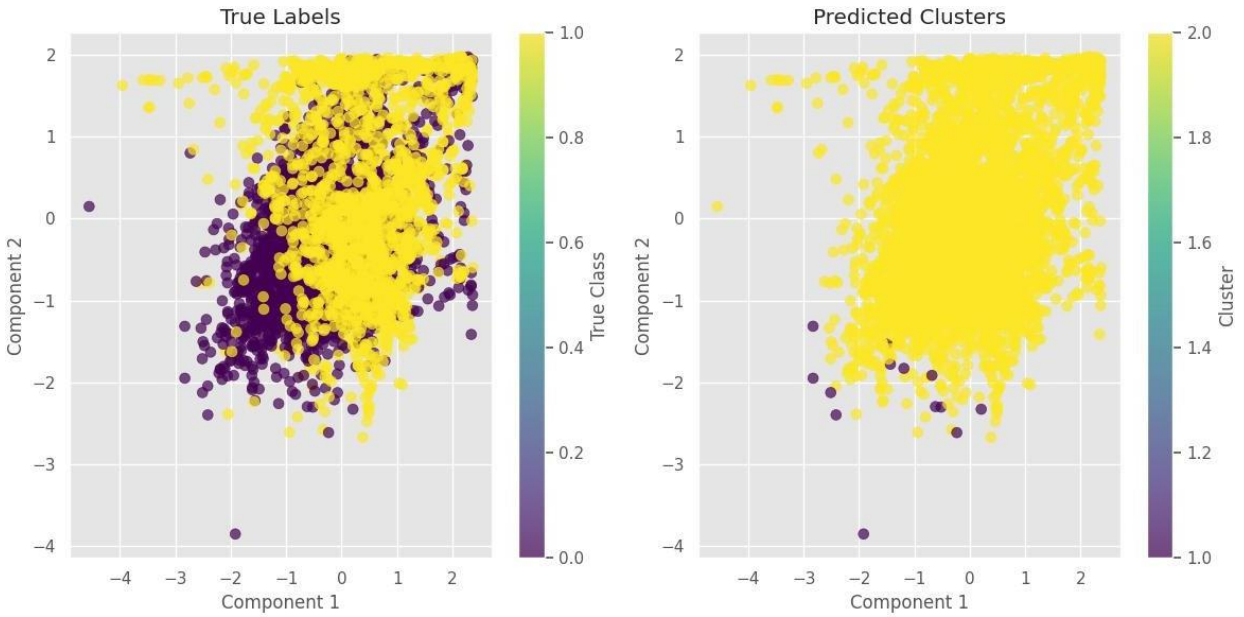


Other Metrics:

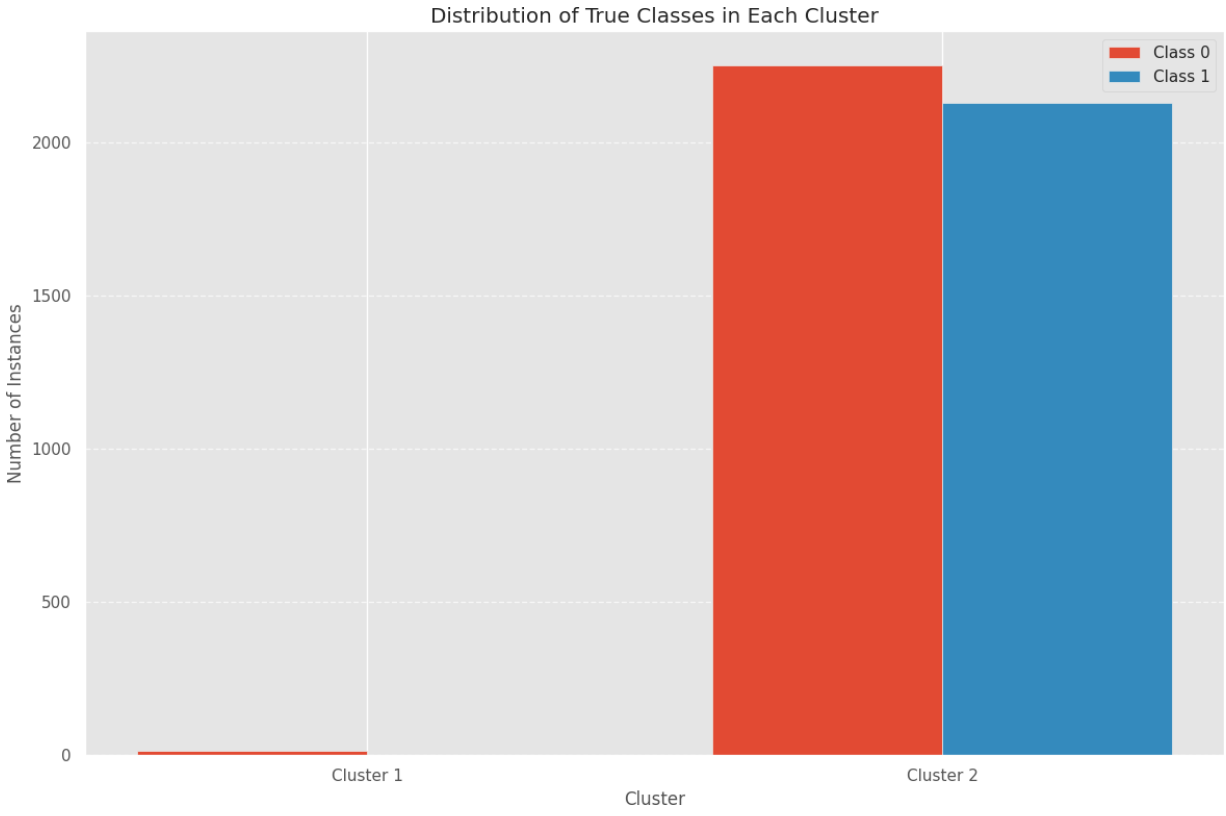
Normalized Mutual Information (NMI): 0.005927516538275286

silhouette_score: 0.23441420256998302

Comparison with true labels:



Class Distribution among clusters:



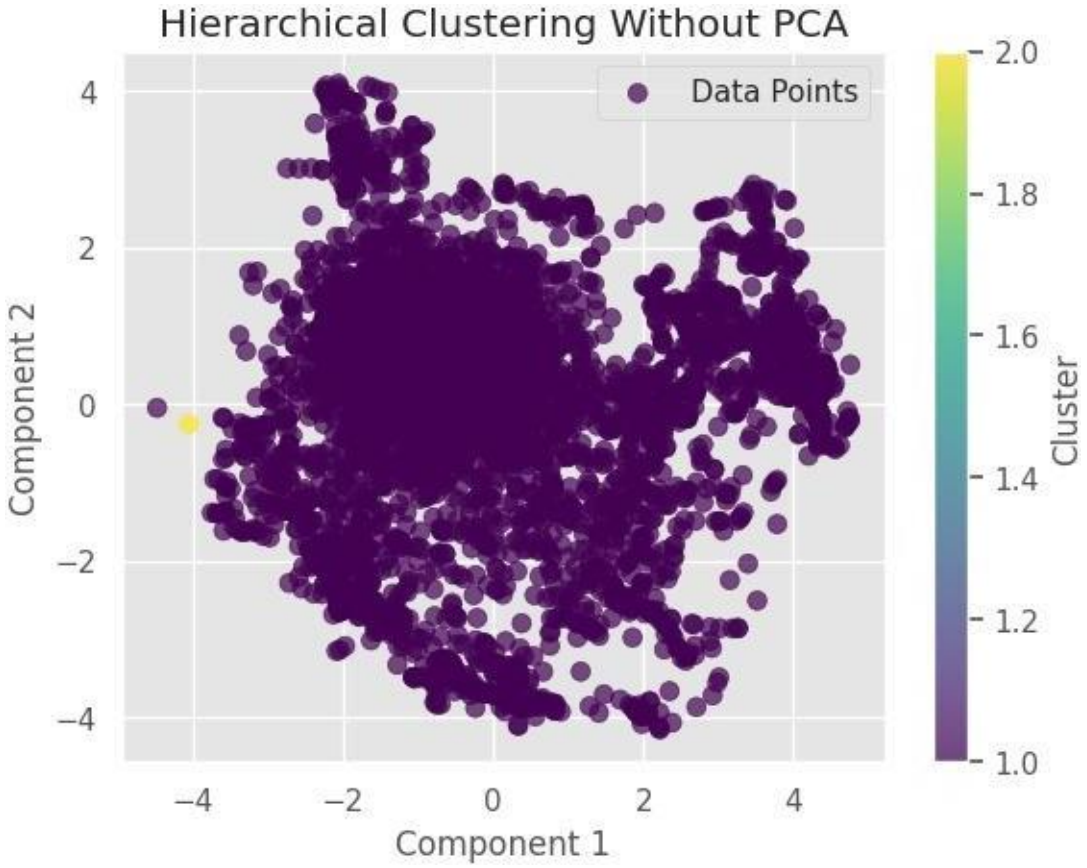
Cluster 1 has 14 datapoints with label 0 and 0 datapoints with label 1.

Cluster 2 has 2251 datapoints with label 0 and 2131 datapoints with label 1.

Average linkage gives horrible performance on our data.

Centroid Linkage:

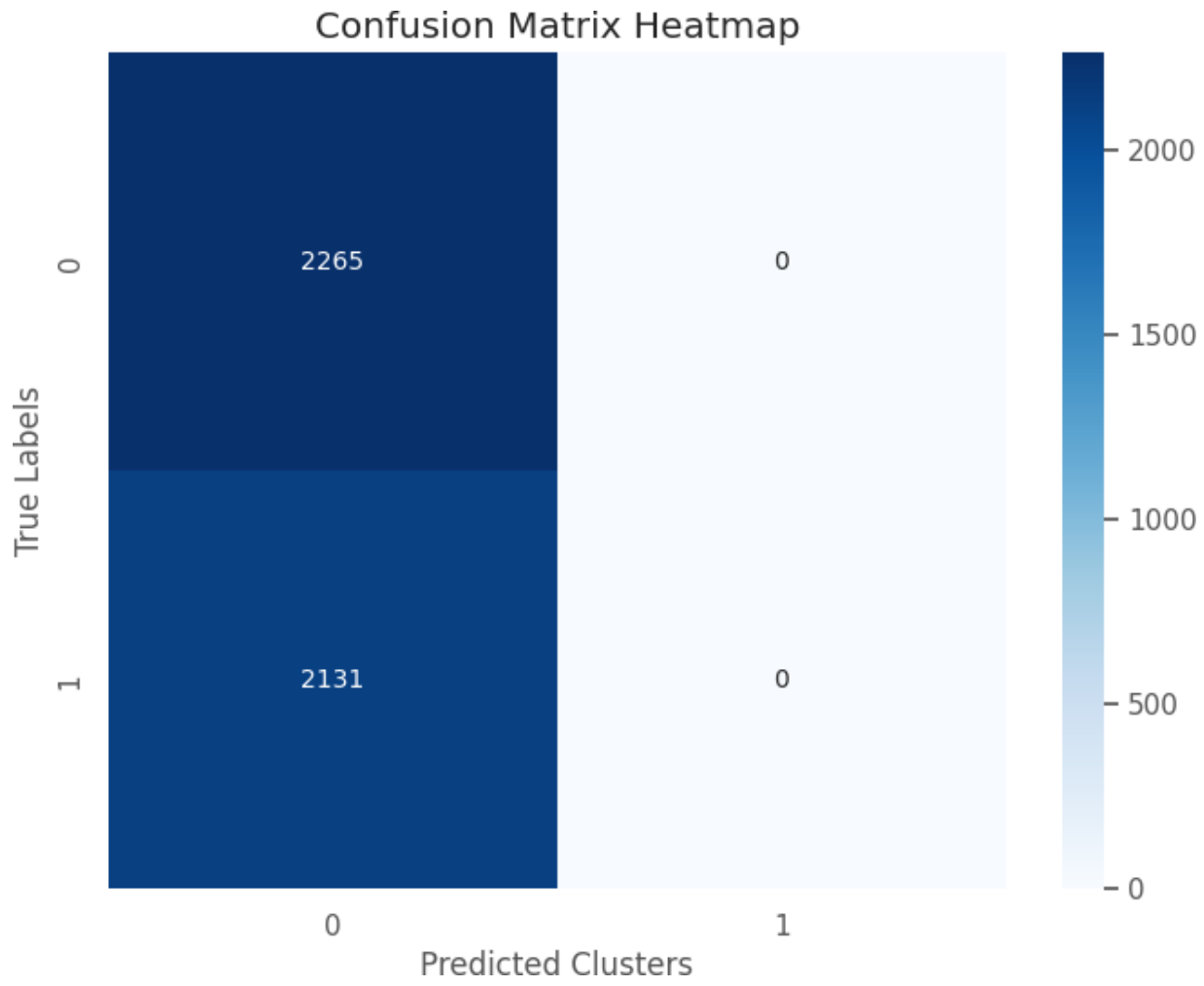
Visualization:



Evaluation Metrics:

Cluster Purity: 0.5152411282984531

Confusion matrix:

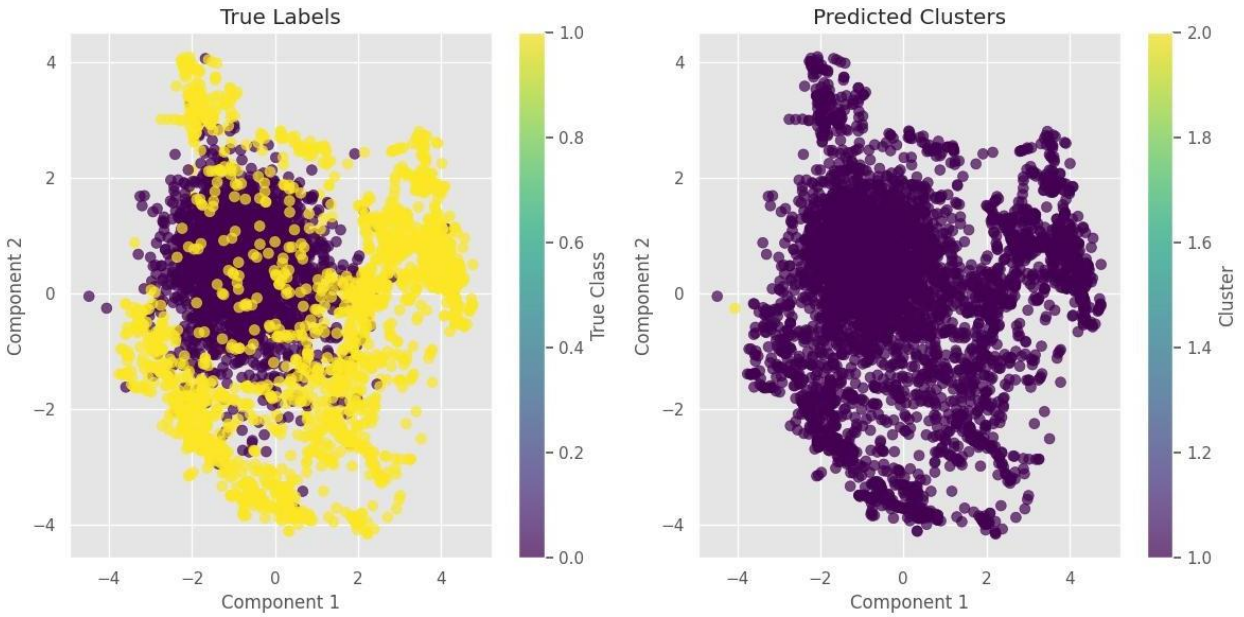


Other Metrics:

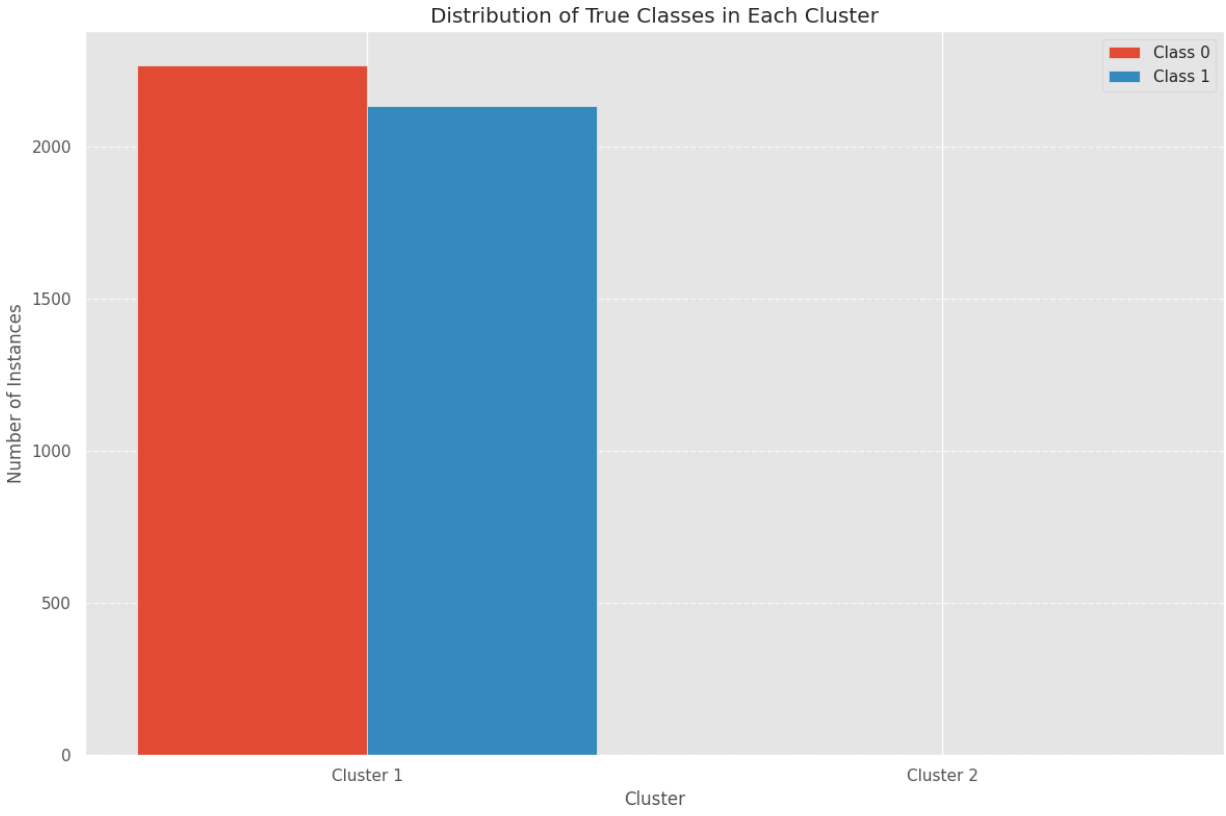
Normalized Mutual Information (NMI): 0.00043427371289392515

silhouette_score: 0.37181424598218865

Comparison with true labels:



Class Distribution among clusters:



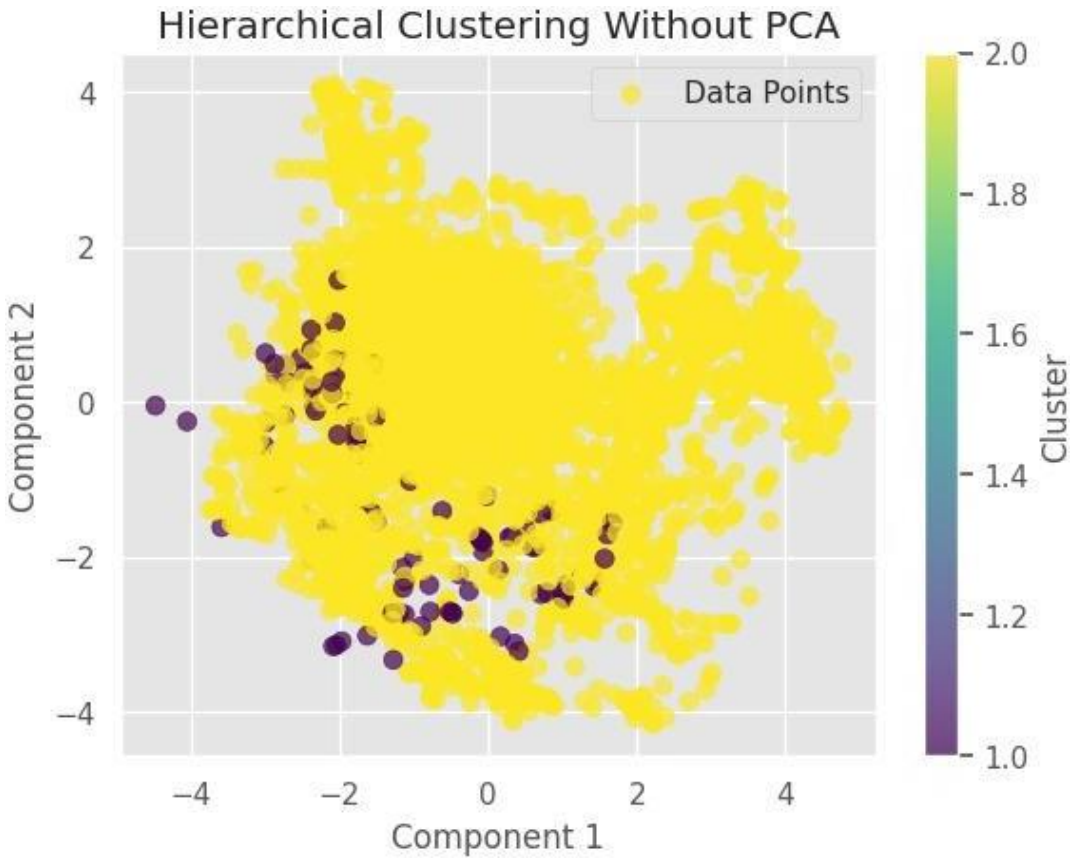
Cluster 1 has 2264 datapoints with label 0 and 2131 datapoints with label 1.

Cluster 2 has 1 datapoints with label 0 and 0 datapoints with label 1.

Centroid linkage gives horrible performance on our data.

Complete Linkage:

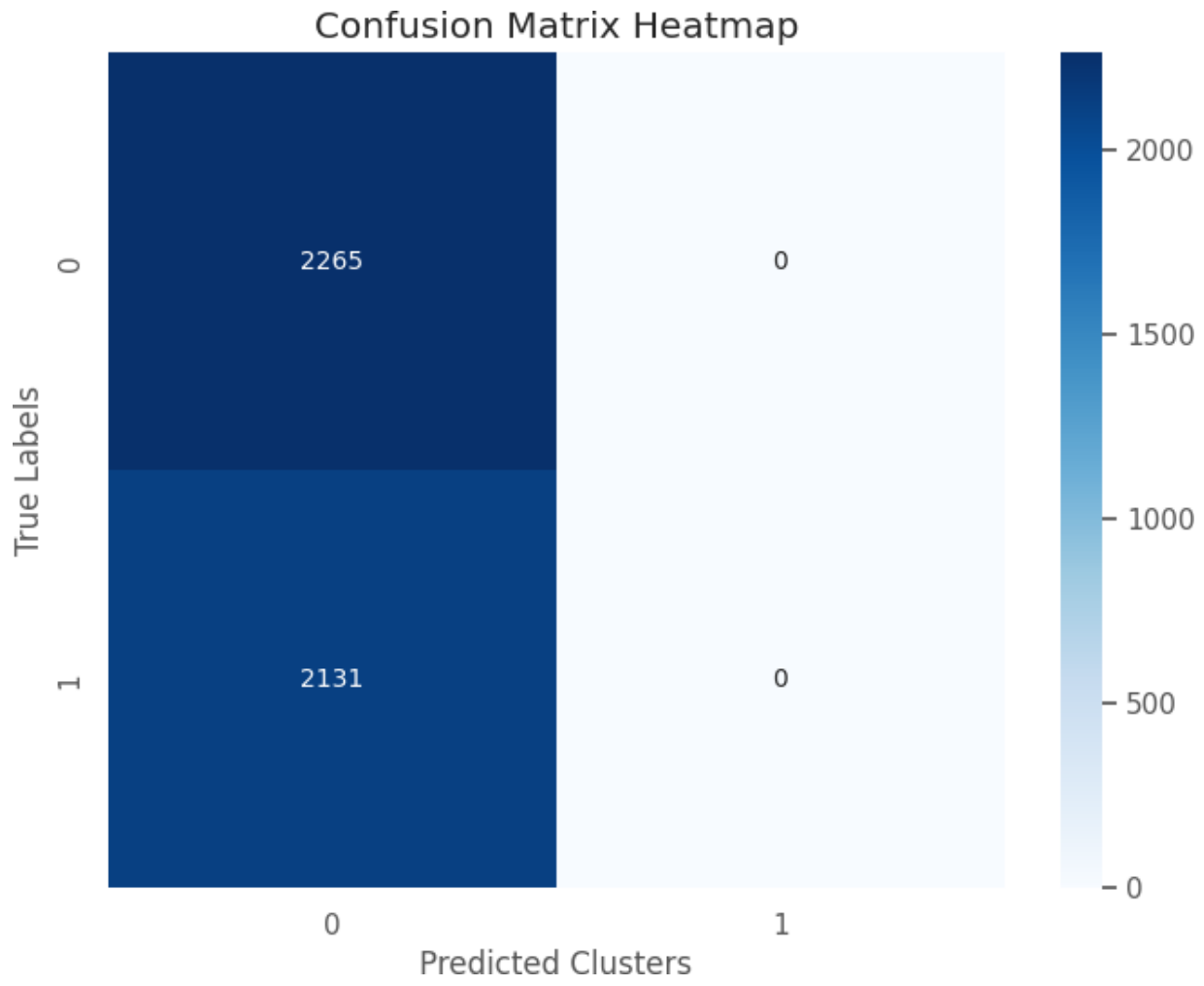
Visualization:



Evaluation Metrics:

Cluster Purity: 0.5152411282984531

Confusion matrix:

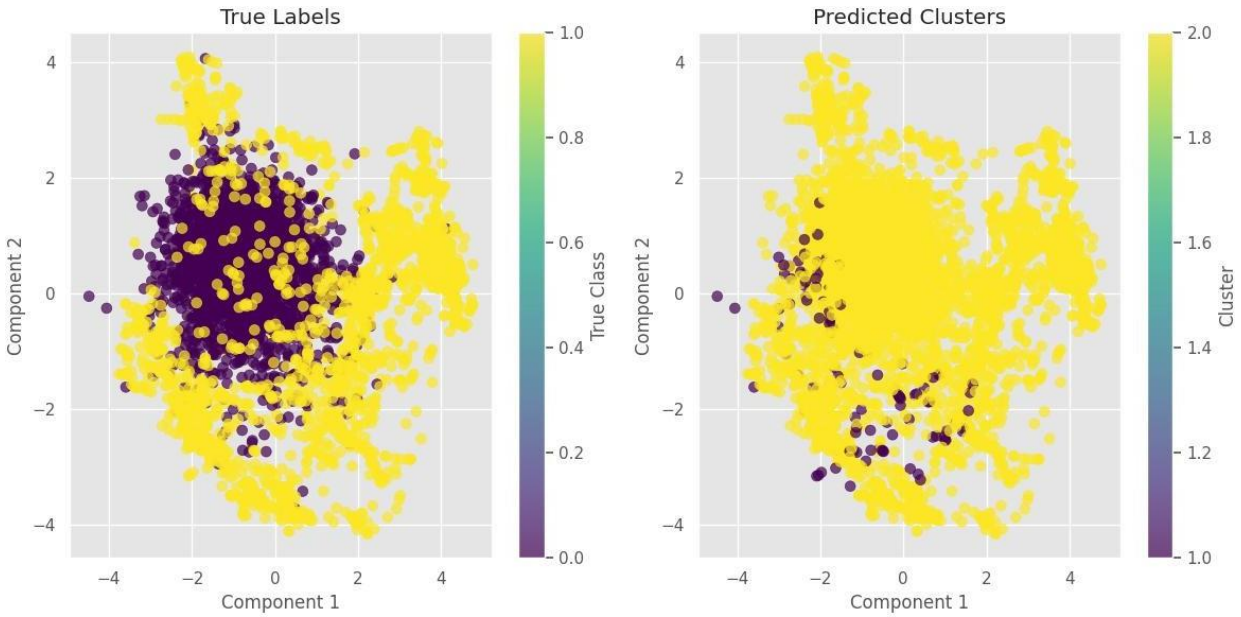


Other Metrics:

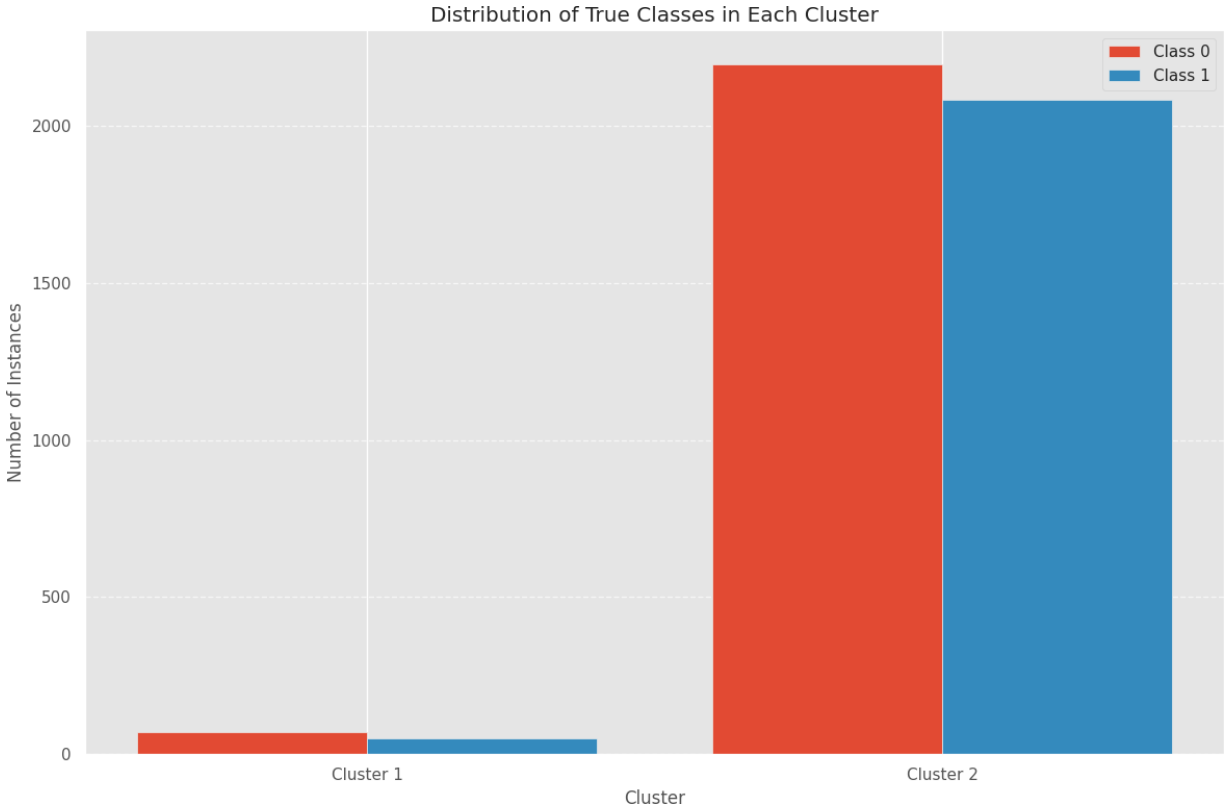
Normalized Mutual Information (NMI): 0.0008079278139194699

silhouette_score: 0.10249557820759768

Comparison with true labels:



Class Distribution among clusters:



Cluster 1 has 71 datapoints with label 0 and 49 datapoints with label 1.

Cluster 2 has 2194 datapoints with label 0 and 2082 datapoints with label 1.

Linkage Comparison in Hierarchical Clustering:

1. Ward Linkage:

- **Cluster Purity:** 0.7527
- **Normalized Mutual Information (NMI):** 0.3310
- **Silhouette Score:** 0.2533
- **Class Distribution:**
  - Cluster 1: 6 datapoints with label 0, 1050 datapoints with label 1
  - Cluster 2: 2259 datapoints with label 0, 1081 datapoints with label 1
- **Summary:** Ward linkage performs well in terms of cluster purity and NMI. Despite some misdistribution, it is a solid performer compared to the other linkages.

2. Average Linkage:

- **Cluster Purity:** 0.5152
- **Normalized Mutual Information (NMI):** 0.0059
- **Silhouette Score:** 0.2344
- **Class Distribution:**
  - Cluster 1: 14 datapoints with label 0, 0 datapoints with label 1
  - Cluster 2: 2251 datapoints with label 0, 2131 datapoints with label 1
- **Summary:** Average linkage gives poor performance. The cluster purity and NMI are very low, indicating poor clustering quality.

3. Centroid Linkage:

- **Cluster Purity:** 0.5152
- **Normalized Mutual Information (NMI):** 0.0004
- **Silhouette Score:** 0.3718
- **Class Distribution:**
  - Cluster 1: 2264 datapoints with label 0, 2131 datapoints with label 1
  - Cluster 2: 1 datapoint with label 0, 0 datapoints with label 1
- **Summary:** Centroid linkage also shows poor performance, with a bad class distribution and a very low NMI, despite a slightly better silhouette score.

4. Complete Linkage:

- **Cluster Purity:** 0.5152
- **Normalized Mutual Information (NMI):** 0.0008

- **Silhouette Score:** 0.1025
- **Class Distribution:**
    - Cluster 1: 71 datapoints with label 0, 49 datapoints with label 1
    - Cluster 2: 2194 datapoints with label 0, 2082 datapoints with label 1
- **Summary:** Complete linkage also provides poor results, with low cluster purity and NMI, and a very low silhouette score.

Comparison:

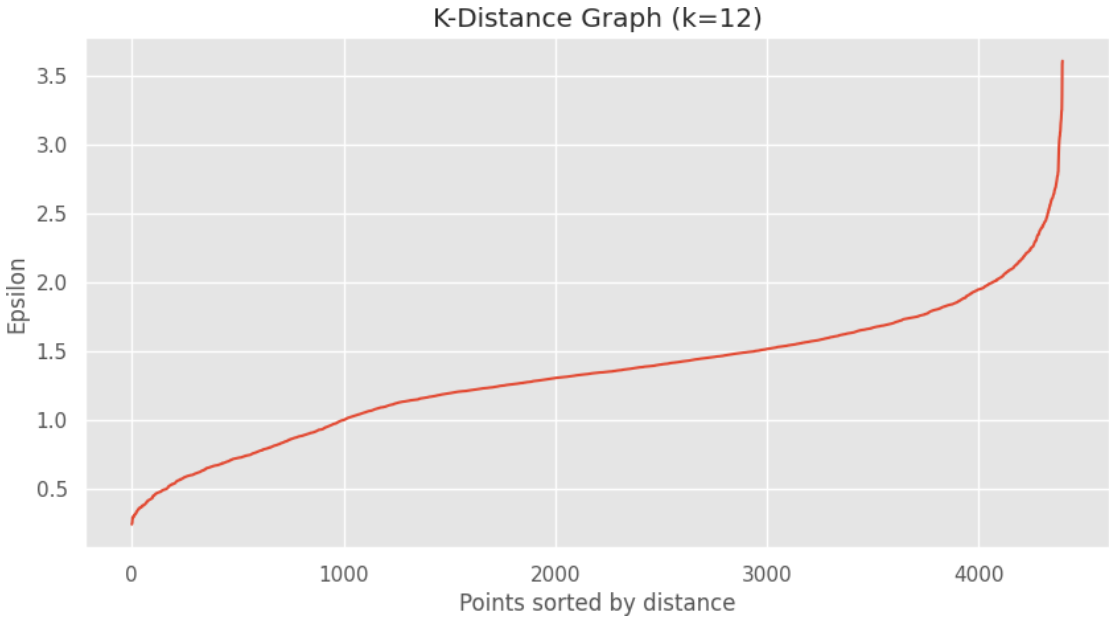
- **Ward Linkage** outperforms all the other methods in terms of **cluster purity (0.7527)** and **NMI (0.3310)**, making it the best-performing linkage method here.
- The other linkages (Average, Centroid, and Complete) show much poorer results, particularly in **NMI** and **cluster purity**, which indicate less accurate clustering.

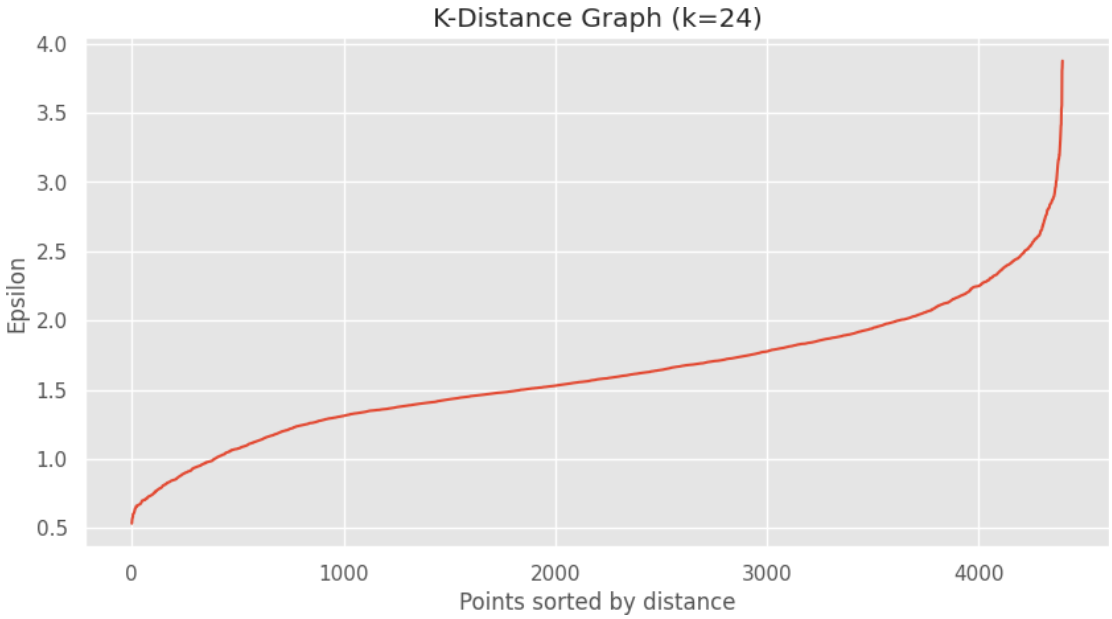
Conclusion:

- **Ward Linkage** is the best performer among the four methods. Despite some misdistribution, it gives the highest cluster purity, NMI, and reasonable silhouette scores. Since our labels (original clusters) are overlapping, that's why we are getting these types of results.

13.1.1.3 DBScan Clustering:

Elbow method:





Elbow method shows eps=2 for min samples 12 and 24. We usually take value of k>= dimension of data. Dimension of our data is 12. It is recommended to keep k between Dimentionality + 1 and 2 * Dimentionality.

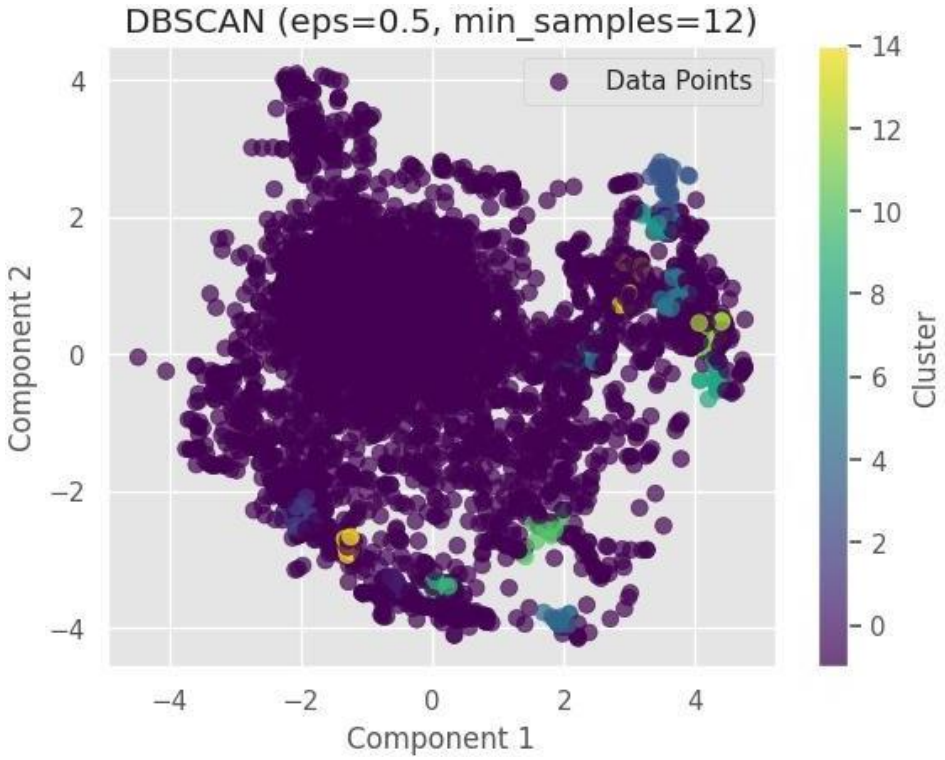
We tried experimenting DBScan for various values.

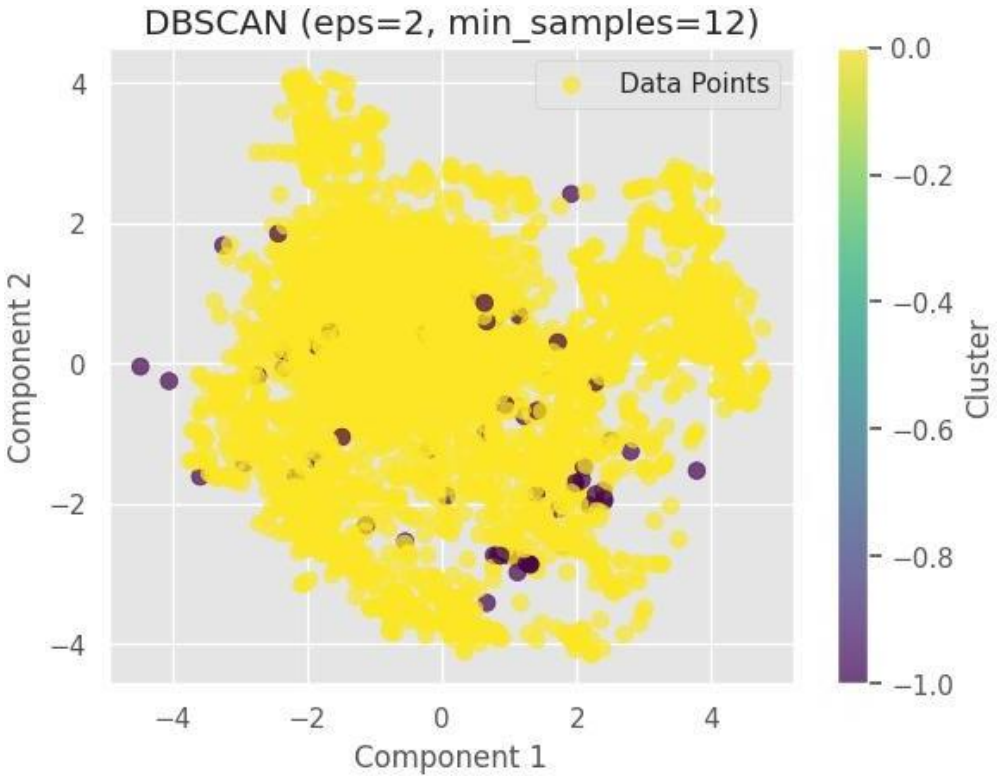
eps_range = [0.5, 1, 1.5, 2, 2.5, 3]

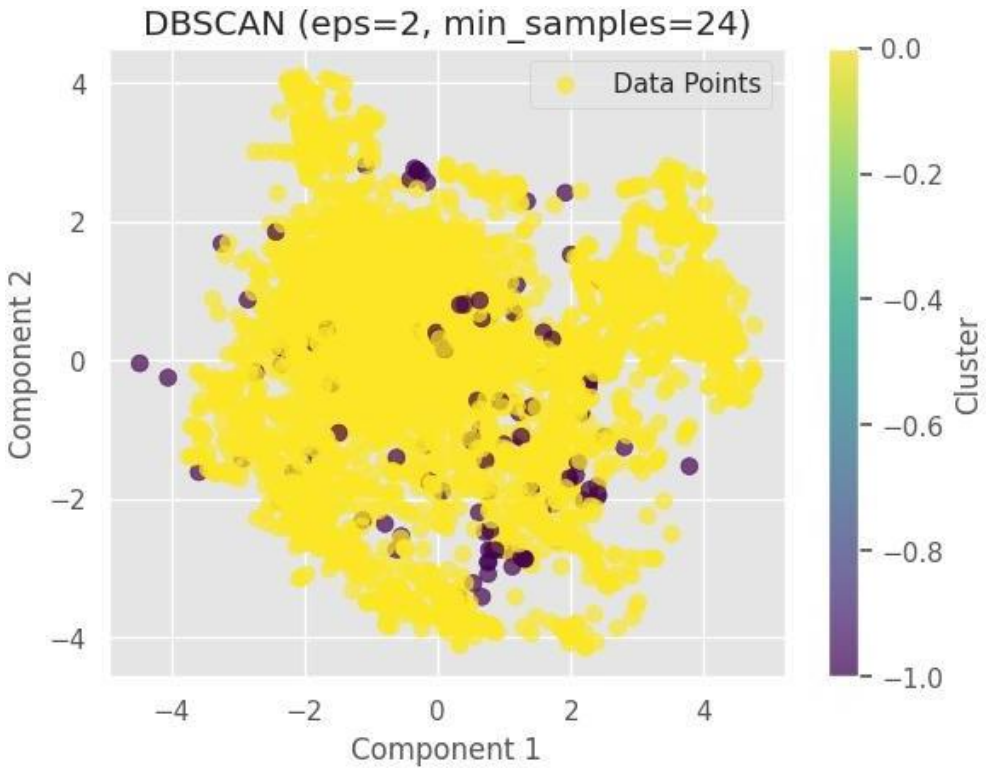
min_samples_range = [12, 15, 18, 21, 24]

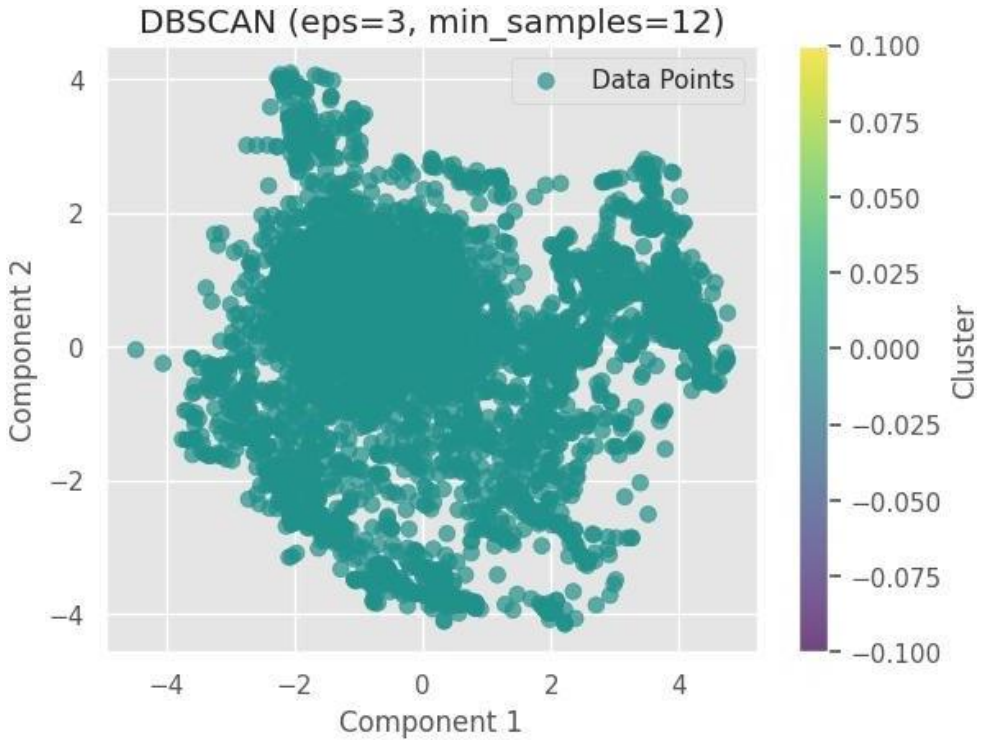
None of them gave satisfactory results. Either it makes a lot of clusters, or it makes 1 cluster with some noise points. This experiment clearly shows that DBScan is not suitable for our dataset because of overlapping clusters.

Visualization:




DBSCAN (eps=2, min_samples=12)
Data Points
Component 2
Component 1
Cluster
4
2
0
−2
−4
−4
−2
0
2
4
0.0
−0.2
−0.4
−0.6
−0.8
−1.0

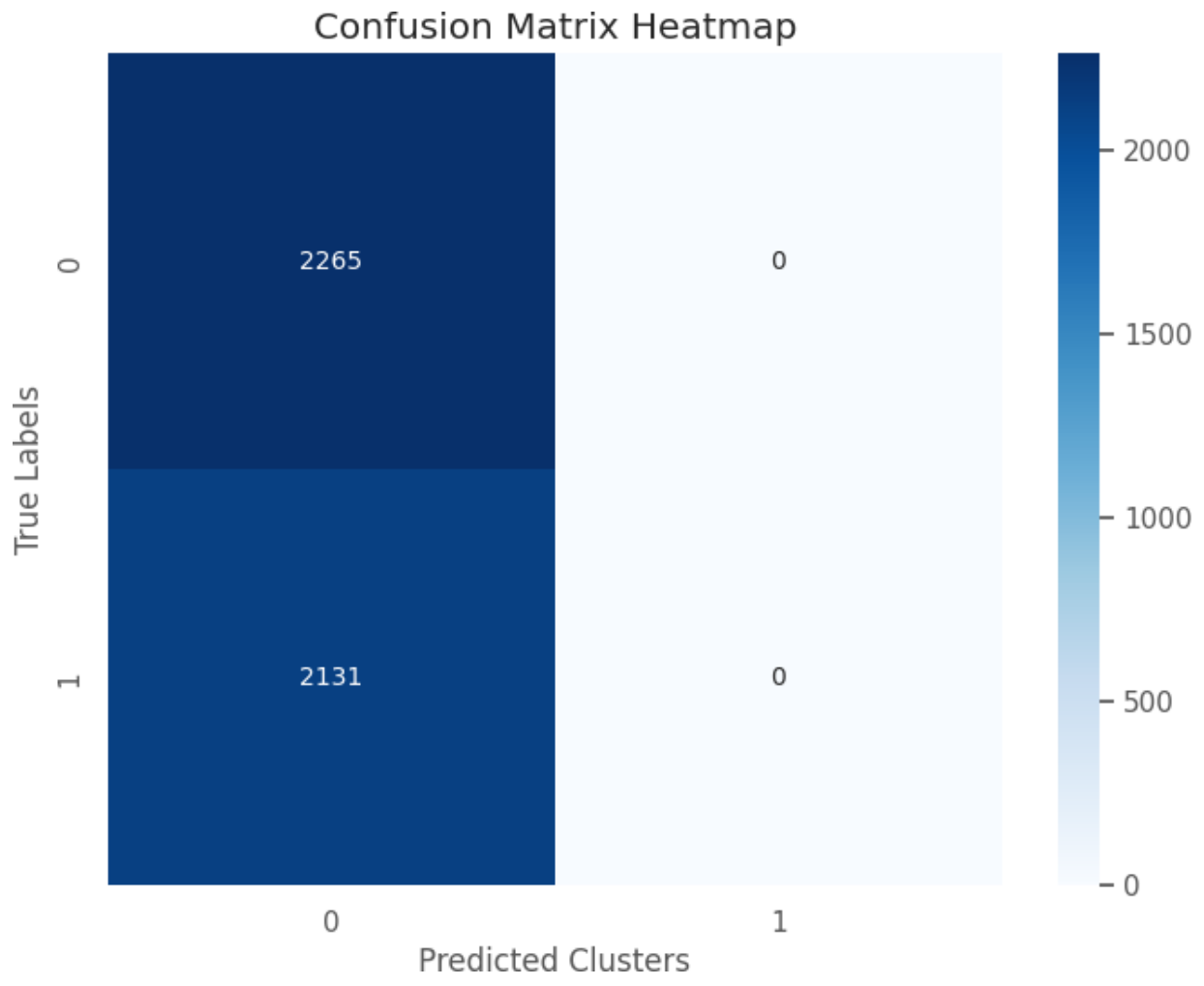

DBSCAN (eps=2, min_samples=24)
Data Points
Component 2
Component 1
Cluster
4
2
0
−2
−4
−4
−2
0
2
4
0.0
−0.2
−0.4
−0.6
−0.8
−1.0



Evaluation Metrics:

Cluster Purity: 0.5152411282984531

Confusion matrix:

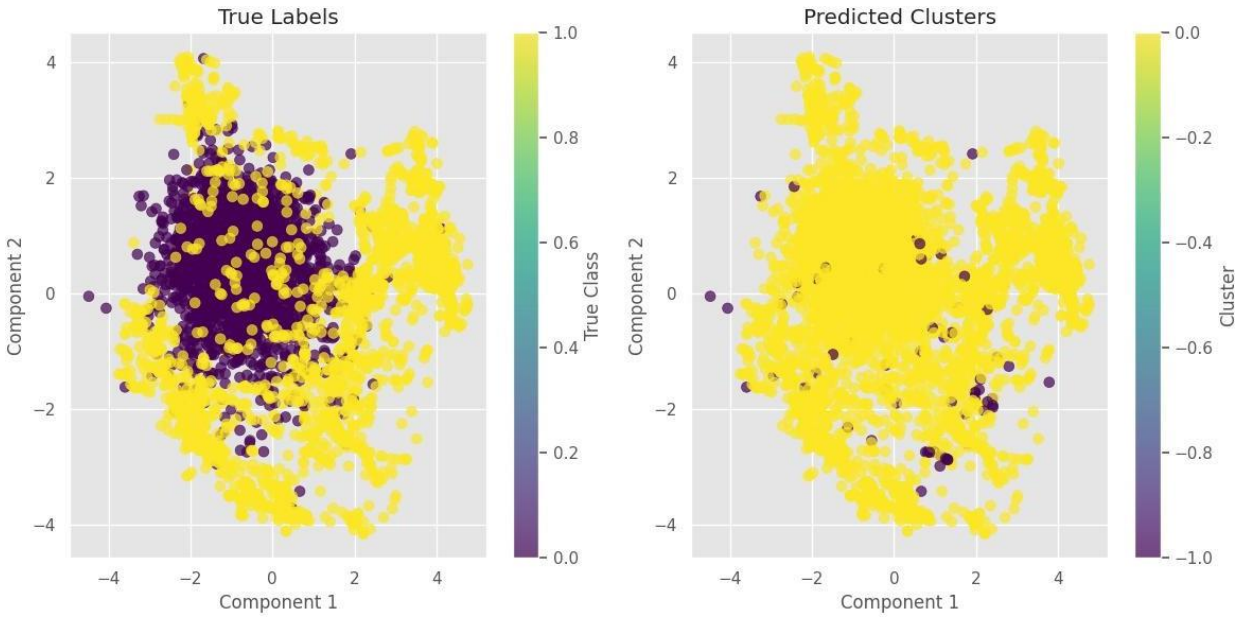


Other Metrics:

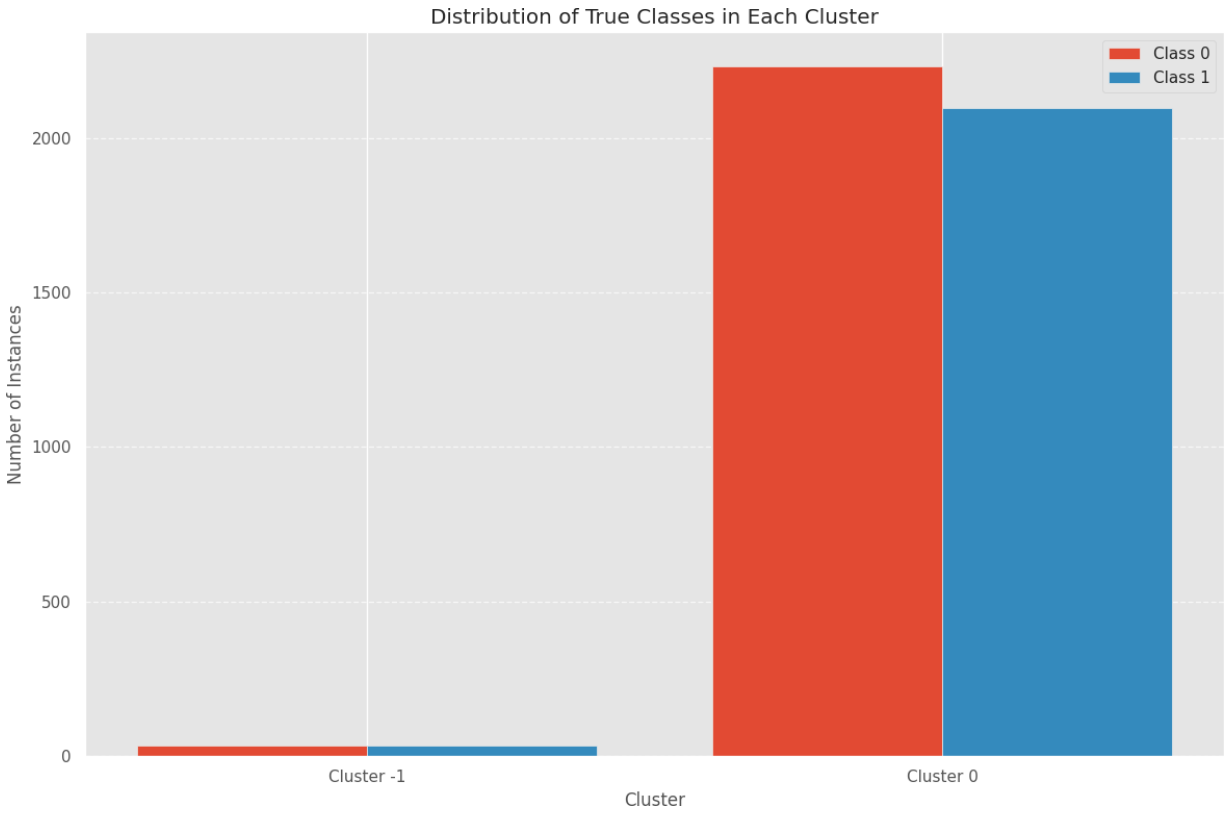
Normalized Mutual Information (NMI): 5.274439163009122e-06

silhouette_score: 0.15552789801523648

Comparison with true labels:



Class Distribution among clusters:

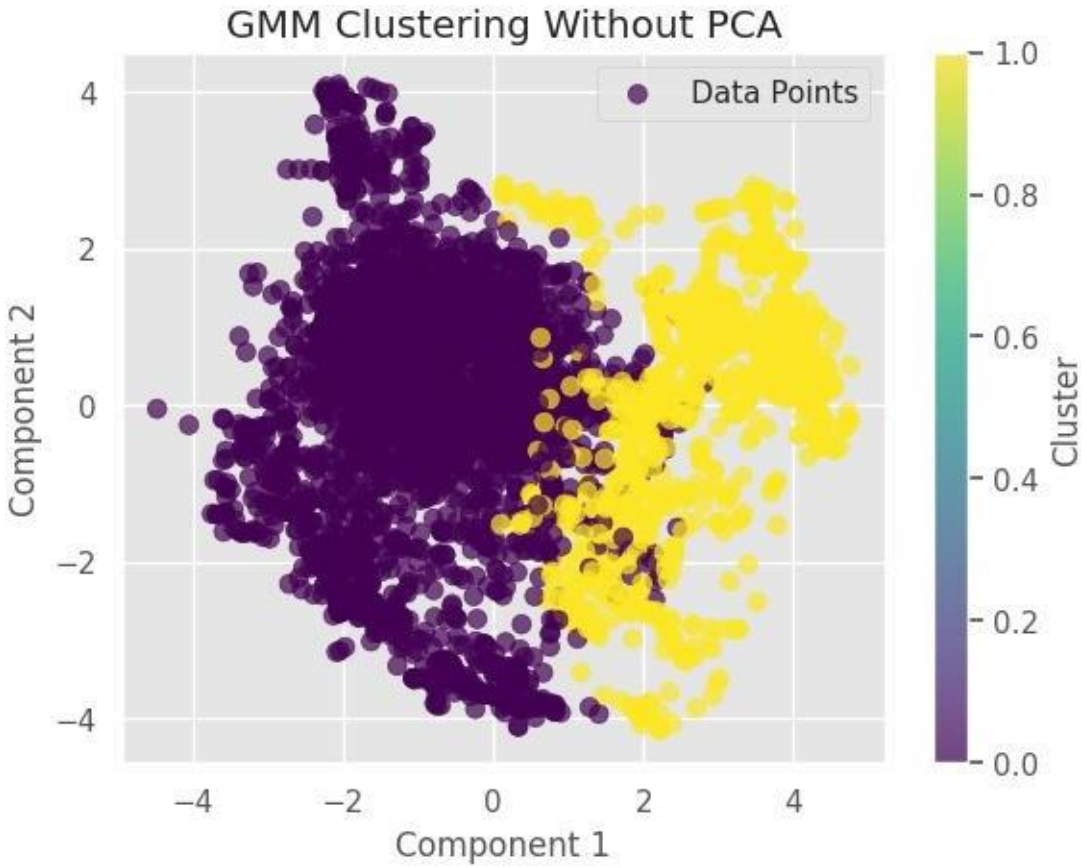


Cluster -1 has 35 datapoints with label 0 and 34 datapoints with label 1.

Cluster 0 has 2230 datapoints with label 0 and 2097 datapoints with label 1.

13.1.1.4 Gaussian Mixture Model Clustering*:*

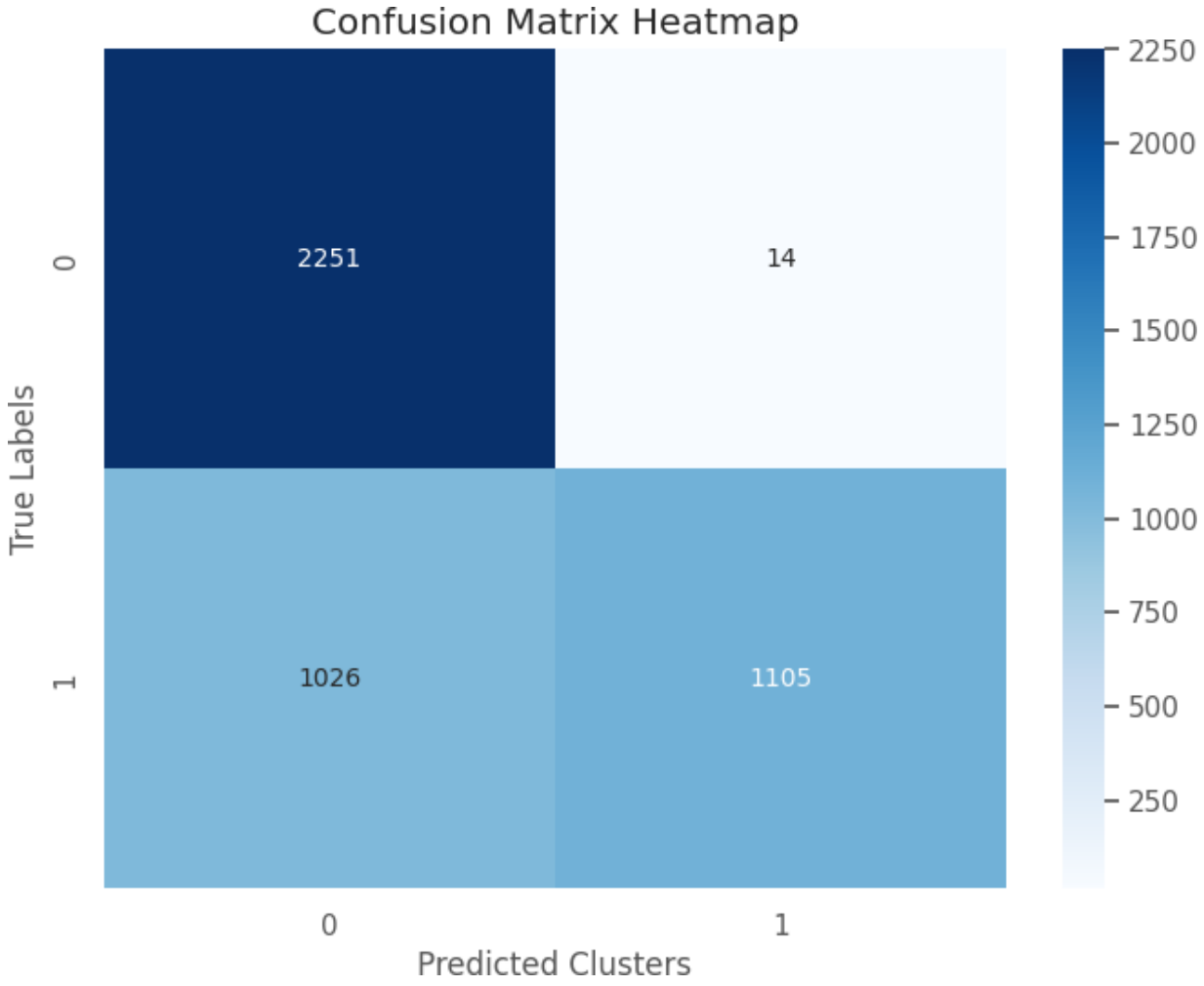
Visualization:



Evaluation Metrics:

Cluster Purity: 0.7634212920837125

Confusion matrix:

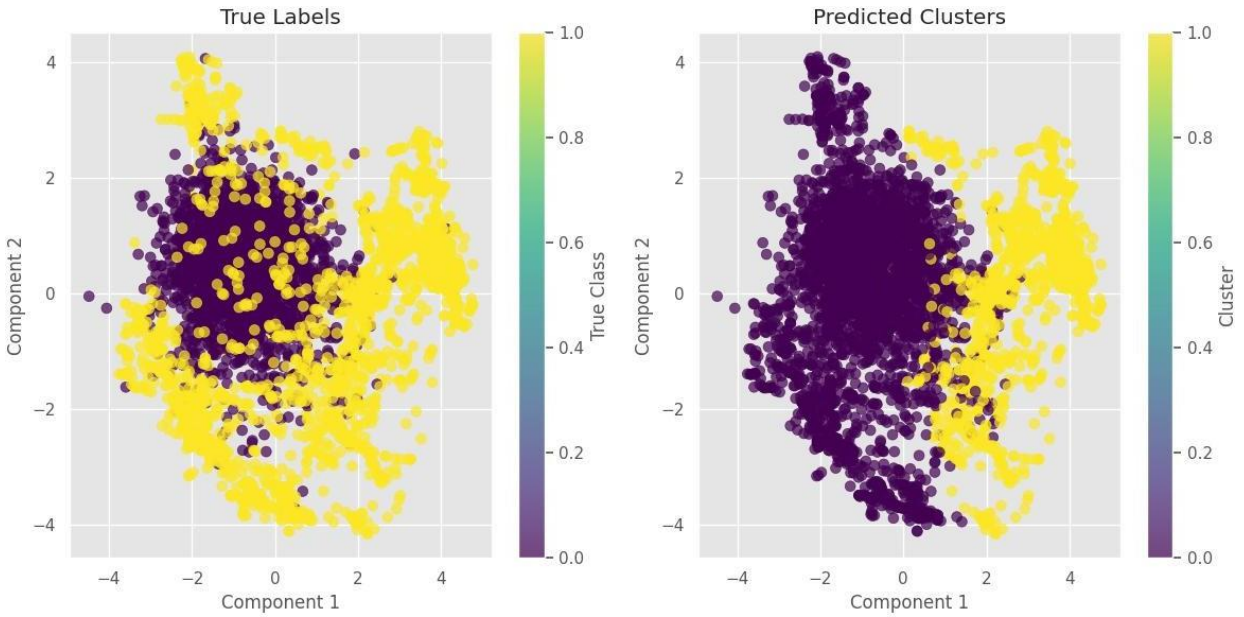


Other Metrics:

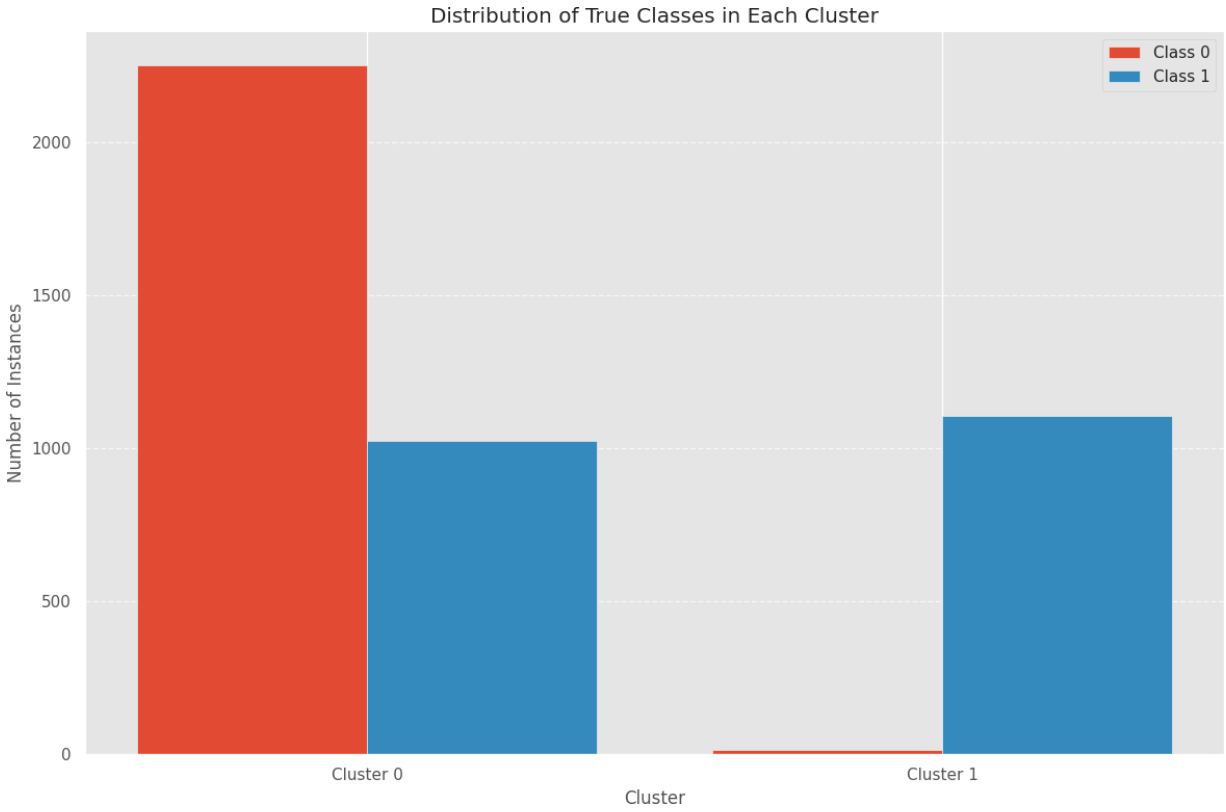
Normalized Mutual Information (NMI 0.3368821463025168

silhouette_score: 0.2542486252113705

Comparison with true labels:



Class Distribution among clusters:



Cluster 0 has 2251 datapoints with label 0 and 1026 datapoints with label 1.

Cluster 1 has 14 datapoints with label 0 and 1105 datapoints with label 1.

13.1.1.5 Model Comparison:

1. K-Means Clustering

- **Elbow Method**: Despite the elbow method suggesting k=3k = 3, kk was set to 2, aligning with the two classes in the labeled dataset.
- **Cluster Purity**: 0.7677 (highest among all models).
- **Normalized Mutual Information (NMI)**: 0.2699.
- **Silhouette Score**: 0.2505.
- **Confusion Matrix**: Showed good distribution across the two clusters.
- **Class Distribution**:
    - Cluster 0: 2133 datapoints with label 0, 889 datapoints with label 1.
    - Cluster 1: 132 datapoints with label 0, 1242 datapoints with label 1.

Summary:
K-Means performed reasonably well, with the highest cluster purity and decent silhouette score, reflecting well-separated clusters. However, the NMI suggests moderate agreement between true labels and clustering.

2. Hierarchical Clustering

a. Ward Linkage

- **Cluster Purity**: 0.7527.
- **NMI**: 0.3310 (highest among all hierarchical methods).
- **Silhouette Score**: 0.2533.
- **Class Distribution**:
    - Cluster 1: 6 datapoints with label 0, 1050 datapoints with label 1.
    - Cluster 2: 2259 datapoints with label 0, 1081 datapoints with label 1.

Summary:
Ward linkage performed best among hierarchical clustering methods, achieving high cluster purity and the best NMI. However, some misclassifications indicate overlapping clusters.

b. Average Linkage

- **Cluster Purity**: 0.5152.
- **NMI**: 0.0059 (very low).
- **Silhouette Score**: 0.2344.
- **Class Distribution**:
    - Cluster 1: 14 datapoints with label 0, 0 datapoints with label 1.
    - Cluster 2: 2251 datapoints with label 0, 2131 datapoints with label 1.

Summary:
Average linkage showed poor performance, with very low NMI and a skewed class distribution.

c. Centroid Linkage

- **Cluster Purity**: 0.5152.
- **NMI**: 0.0004.
- **Silhouette Score**: 0.3718 (highest among hierarchical methods).
- **Class Distribution**:
    - Cluster 1: 2264 datapoints with label 0, 2131 datapoints with label 1.
    - Cluster 2: 1 datapoint with label 0, 0 datapoints with label 1.

Summary:
Centroid linkage provided a slightly better silhouette score but failed in NMI and cluster purity, indicating poor clustering quality.

d. Complete Linkage

- **Cluster Purity**: 0.5152.
- **NMI**: 0.0008.
- **Silhouette Score**: 0.1025.
- **Class Distribution**:
    - Cluster 1: 71 datapoints with label 0, 49 datapoints with label 1.
    - Cluster 2: 2194 datapoints with label 0, 2082 datapoints with label 1.

Summary:
Complete linkage performed the worst, with extremely low scores and poor clustering.

3. DBScan Clustering

- **Cluster Purity**: 0.5152.
- **NMI**: 0.000005 (almost negligible).
- **Silhouette Score**: 0.1555.
- **Class Distribution**:
    - Cluster -1: 35 datapoints with label 0, 34 datapoints with label 1 (noise).
    - Cluster 0: 2230 datapoints with label 0, 2097 datapoints with label 1.

Summary:
DBScan struggled to form meaningful clusters due to overlapping data. It either formed one large cluster or multiple fragmented clusters, making it unsuitable for this dataset.

4. Gaussian Mixture Model (GMM) Clustering

- **Cluster Purity**: 0.7634.
- **NMI**: 0.3368 (highest overall).
- **Silhouette Score**: 0.2542.
- **Class Distribution**:
    - Cluster 0: 2251 datapoints with label 0, 1026 datapoints with label 1.
    - Cluster 1: 14 datapoints with label 0, 1105 datapoints with label 1.

Summary:
GMM performed well, achieving high NMI and cluster purity. Its probabilistic nature helped it handle overlapping clusters better than K-Means or hierarchical methods.

Model Comparison

Model	Cluster Purity	NMI	Silhouette Score	Notes
K-Means	**0.7677**	0.2699	0.2505	Best cluster purity; struggles with NMI.
Ward Linkage	0.7527	0.3310	0.2533	Best hierarchical method.
Average Linkage	0.5152	0.0059	0.2344	Poor performance; skewed clusters.
Centroid Linkage	0.5152	0.0004	0.3718	High silhouette but poor clustering.
Complete Linkage	0.5152	0.0008	0.1025	Worst hierarchical method.
DBScan	0.5152	0.000005	0.1555	Unsuitable for overlapping data.
GMM	0.7634	**0.3368**	**0.2542**	Best NMI; strong performance overall.

Conclusion

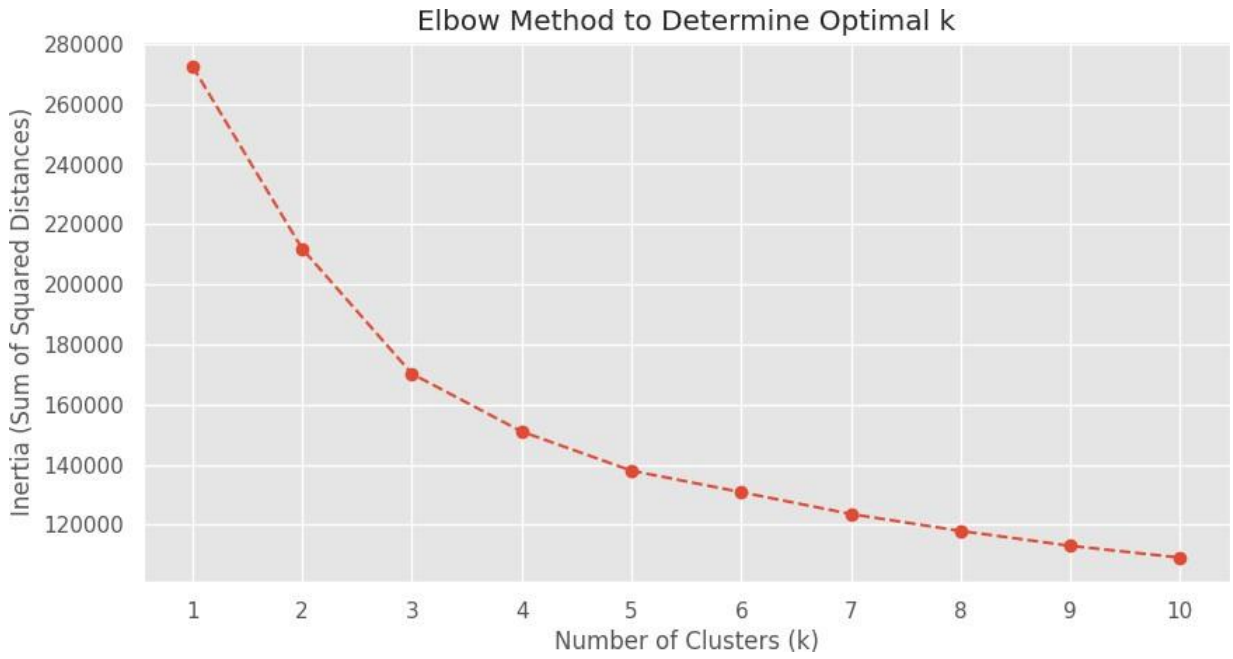
1. **Best Model**: Gaussian Mixture Model (GMM) due to its ability to handle overlapping clusters and achieving the highest NMI and strong cluster purity.
2. **Runner-Up**: K-Means Clustering, with the highest cluster purity but slightly lower NMI.
3. **Hierarchical Clustering**: Ward linkage is the best option, but overall, hierarchical clustering methods are less suitable for this dataset.
4. **Unsuitable Models**: DBScan and other linkage methods in hierarchical clustering fail to handle the overlapping clusters effectively.

13.2 Year2

13.2.1 SMOTE:

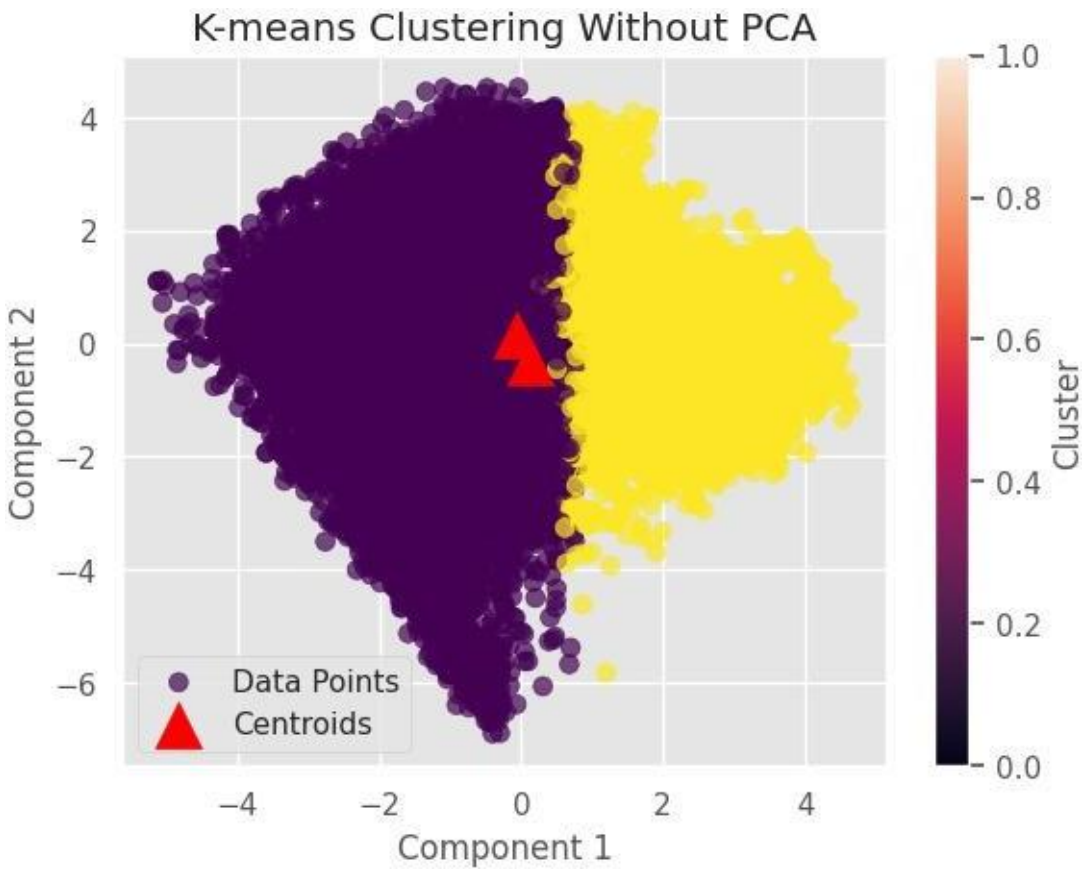
13.2.1.1 K Means Clustering:

Elbow method:



Elbow method shows k=3 but since our dataset is labelled and we are dealing with 2 classes, k = 2.

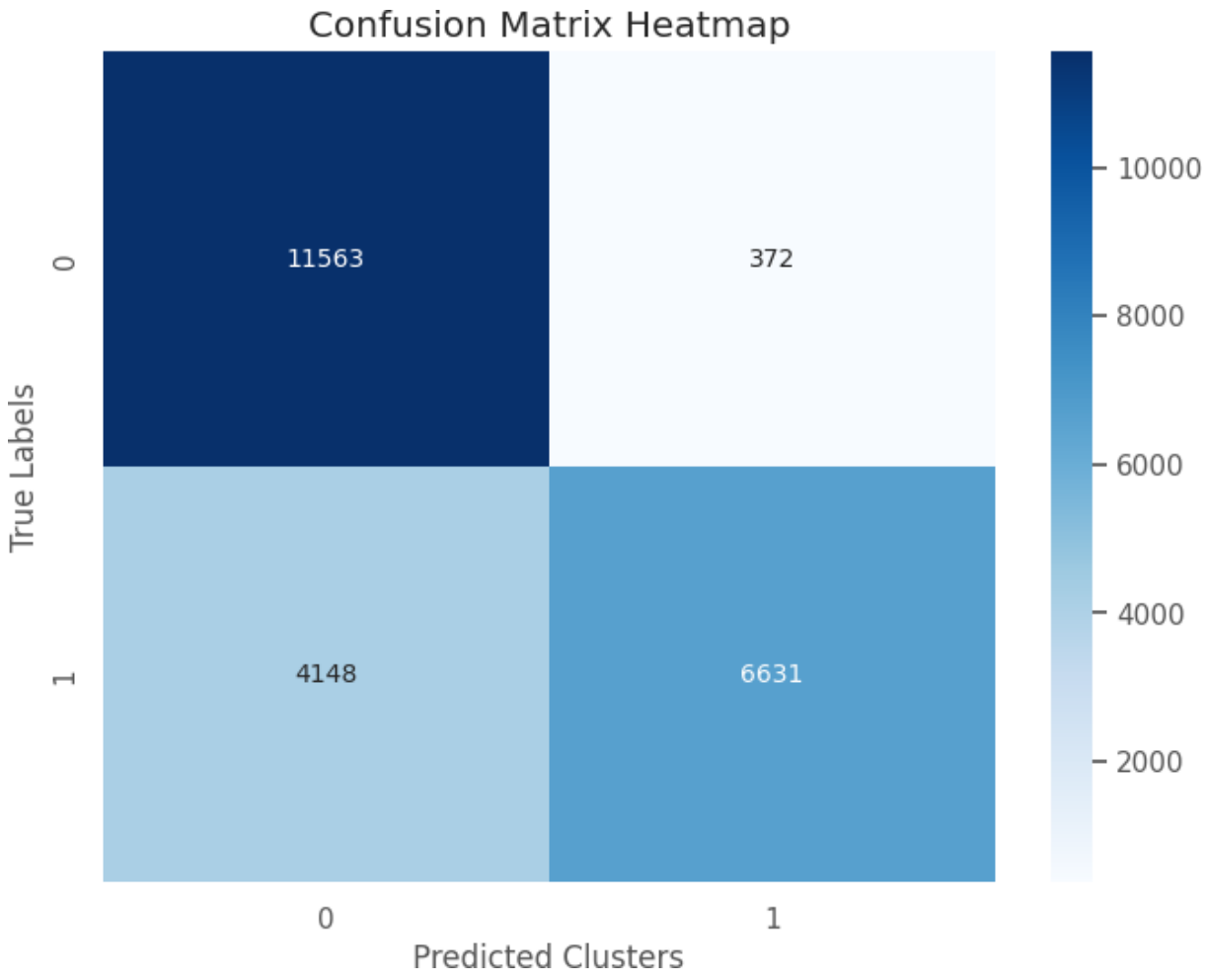
Visualization:



Evaluation Metrics:

Cluster Purity: 0.8010037862111473

Confusion matrix:

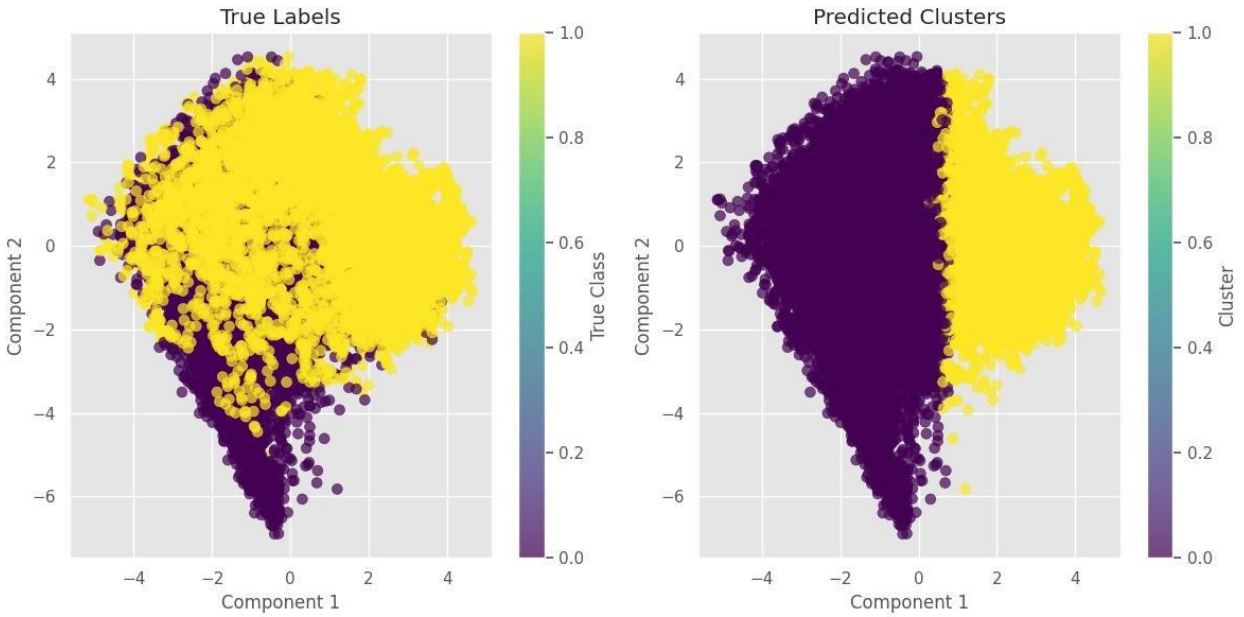


Other Metrics:

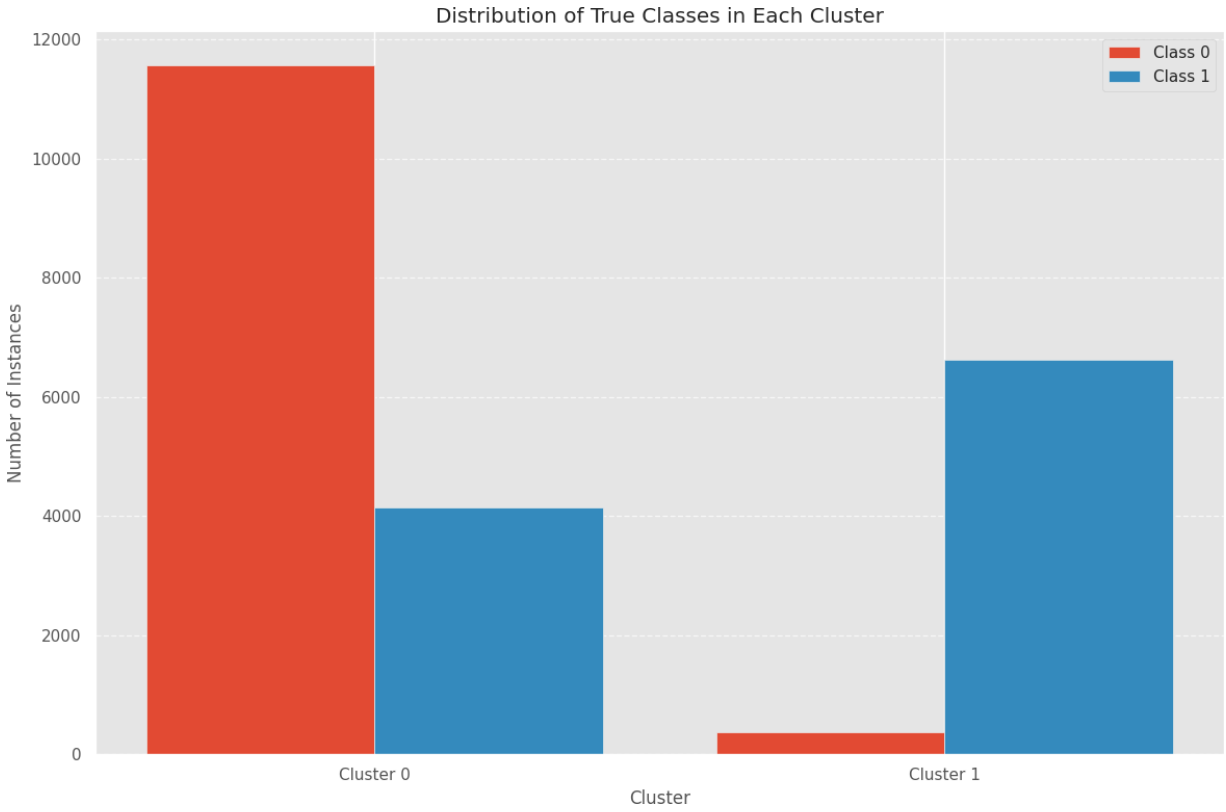
Normalized Mutual Information (NMI): 0.349100687606029

silhouette_score: 0.22434101297080272

Comparison with true labels:



Class Distribution among clusters:

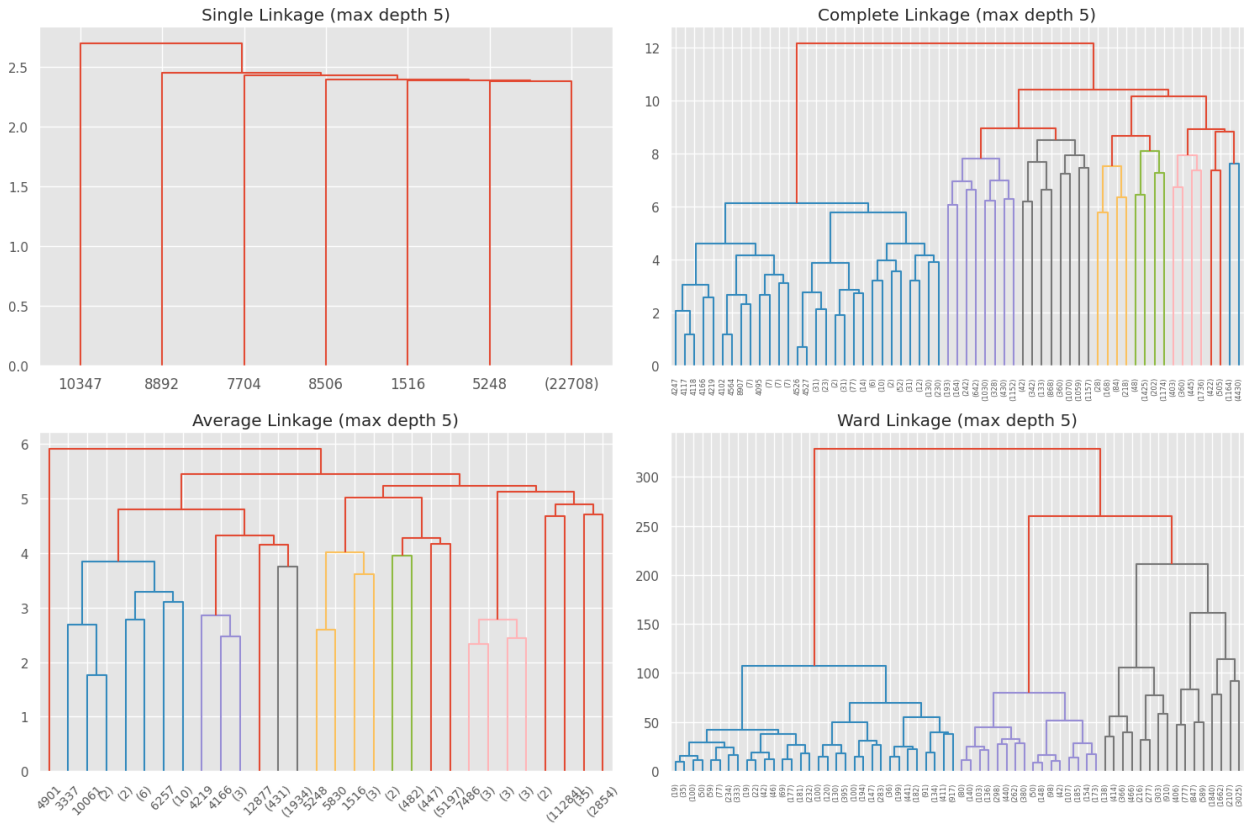


Cluster 0 has 11563 datapoints with label 0 and 4148 datapoints with label 1.

Cluster 1 has 372 datapoints with label 0 and 6631 datapoints with label 1.

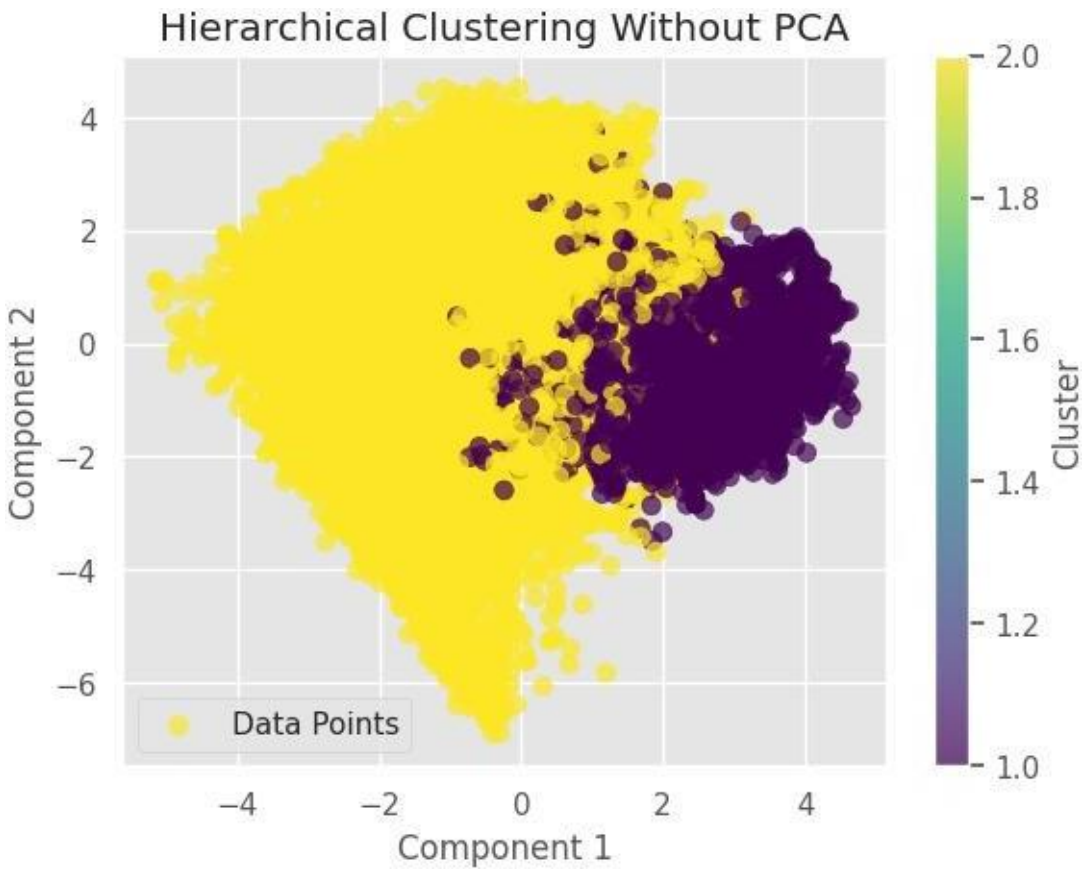
13.2.1.2 Hierarchical Clustering*:*

Dendrogram:



Ward Linkage:

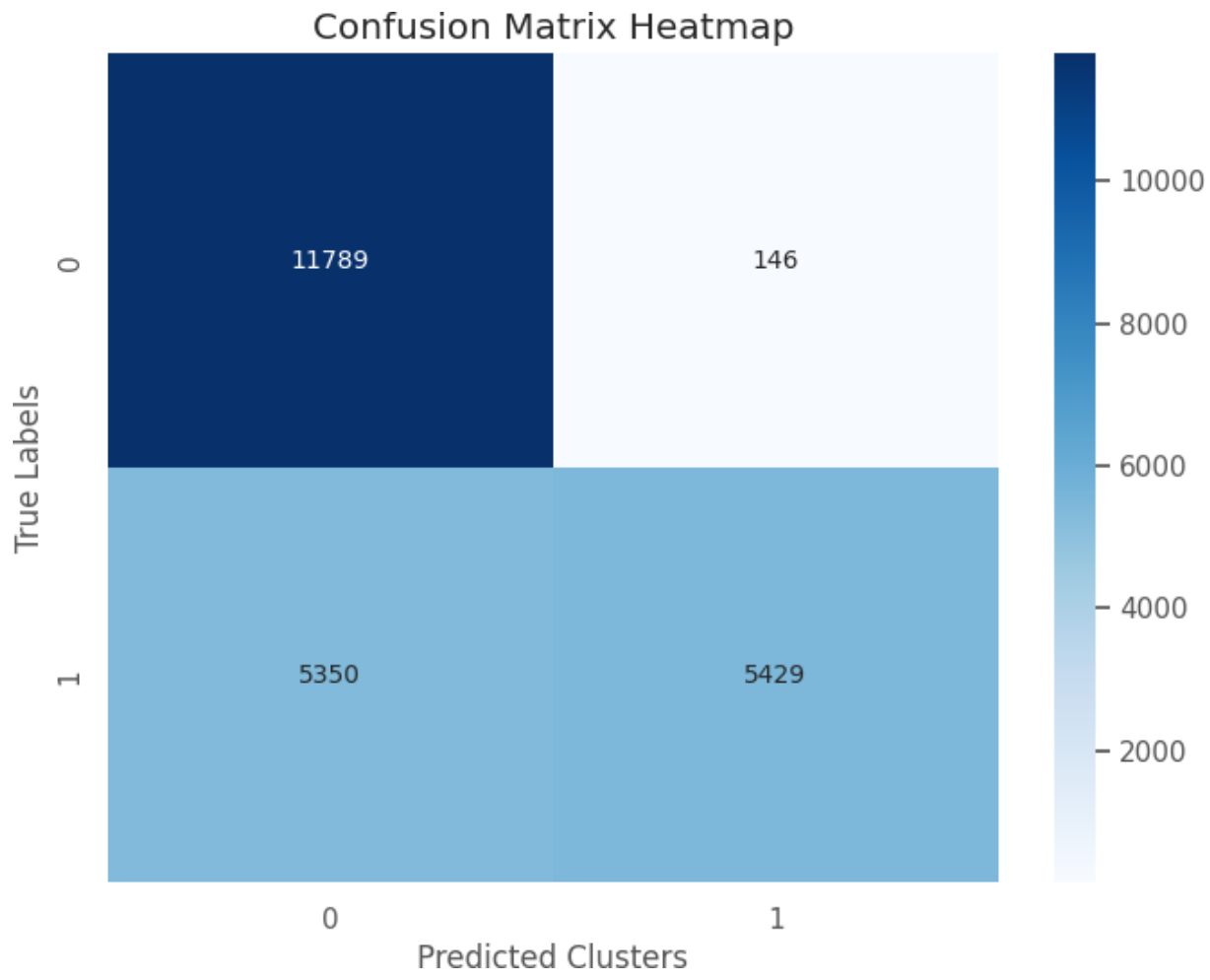
Visualization:



Evaluation Metrics:

Cluster Purity: 0.75803469226028

Confusion matrix:

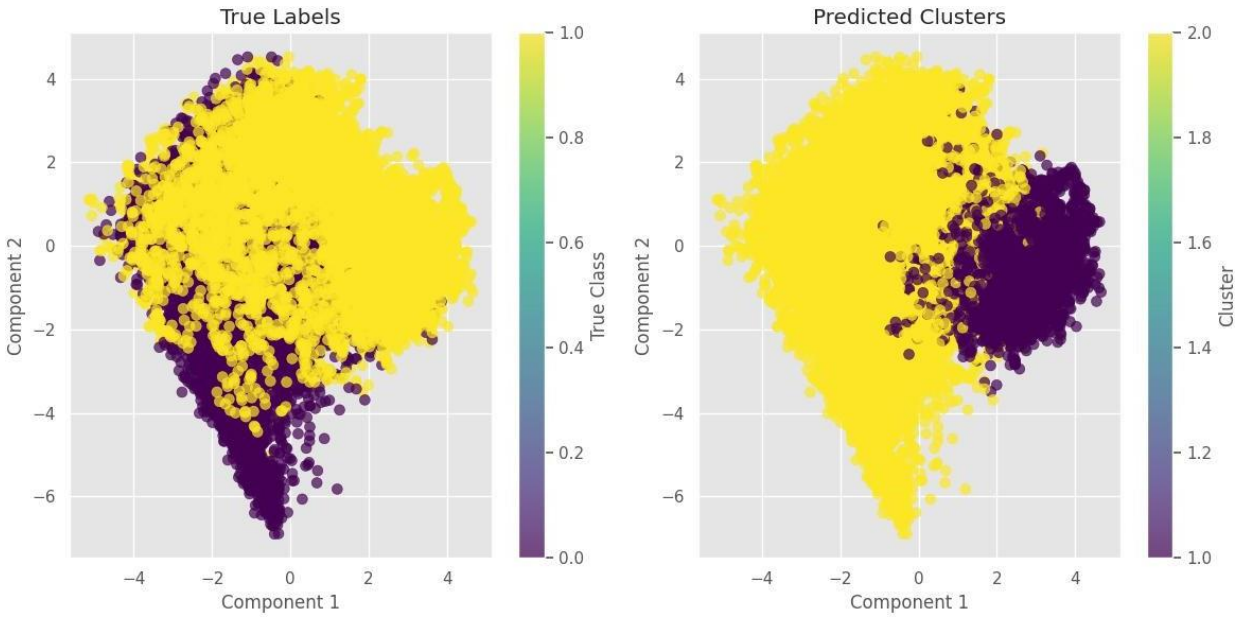


Other Metrics:

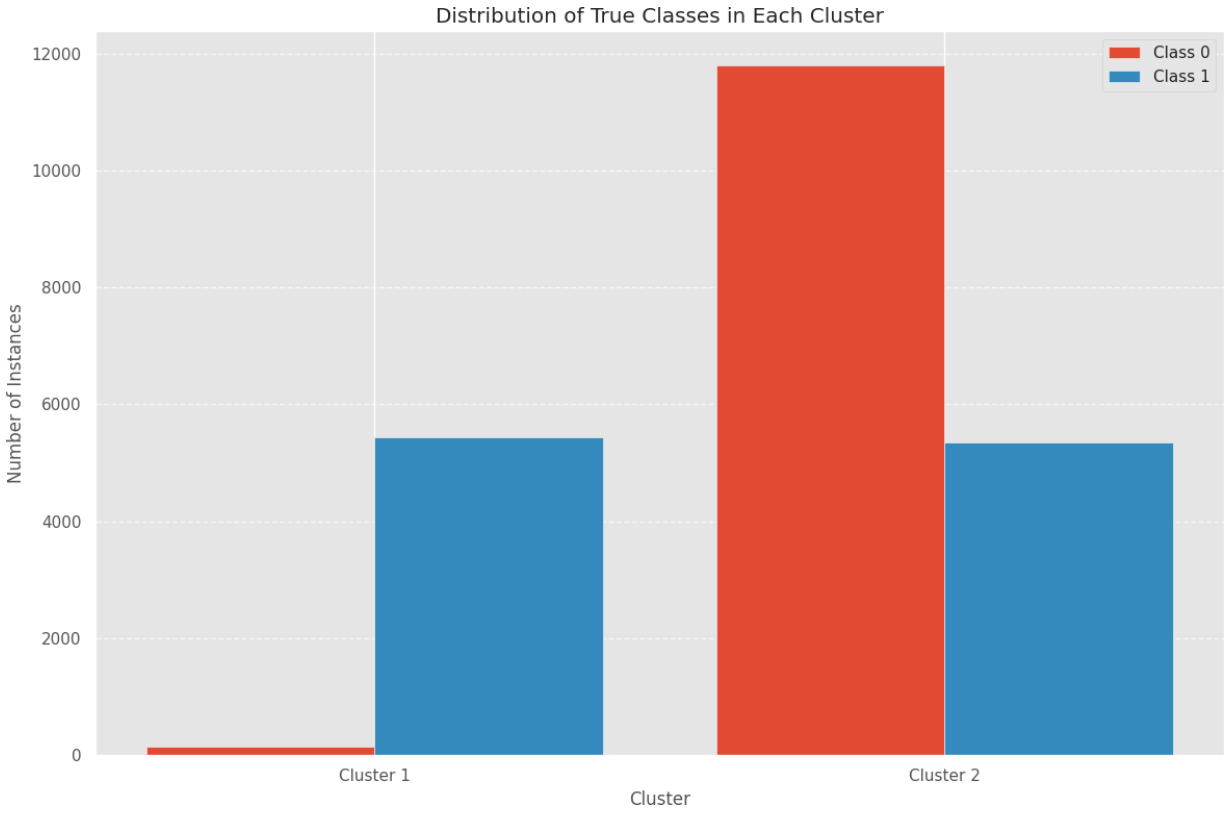
Normalized Mutual Information (NMI): 0.31006892092096316

silhouette_score: 0.20429838814470888

Comparison with true labels:



Class Distribution among clusters:

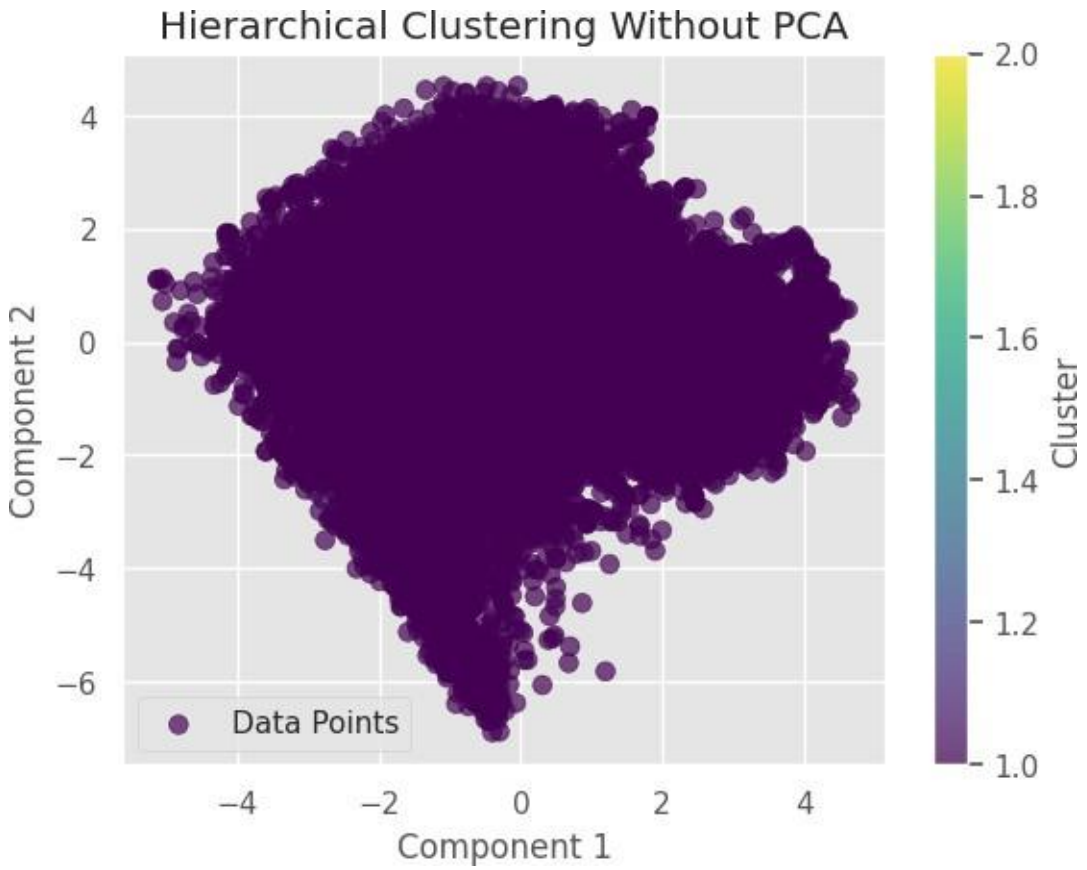


Cluster 1 has 146 datapoints with label 0 and 5429 datapoints with label 1.

Cluster 2 has 11789 datapoints with label 0 and 5350 datapoints with label 1.

Average Linkage:

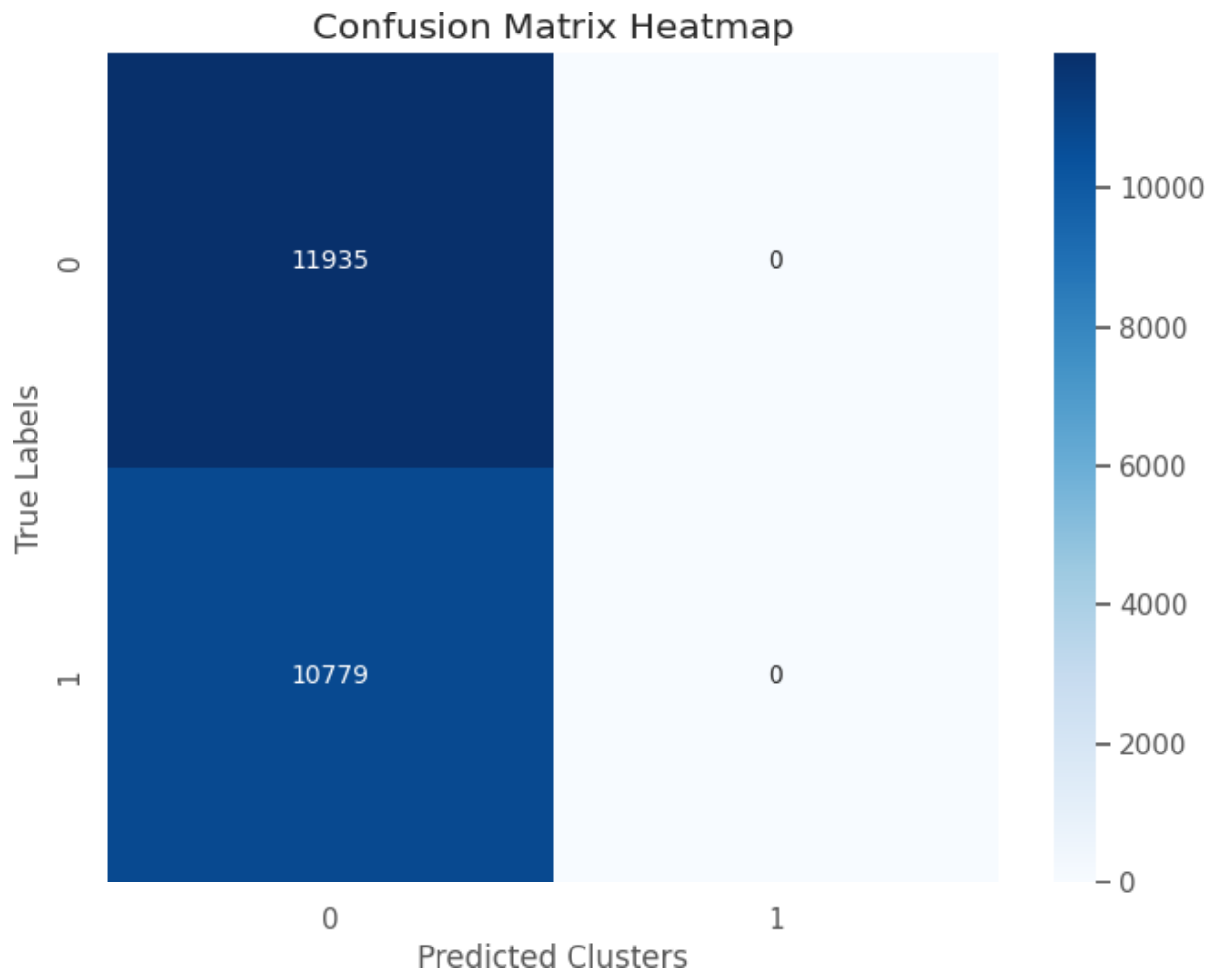
Visualization:



Evaluation Metrics:

Cluster Purity: 0.5254468609668046

Confusion matrix:

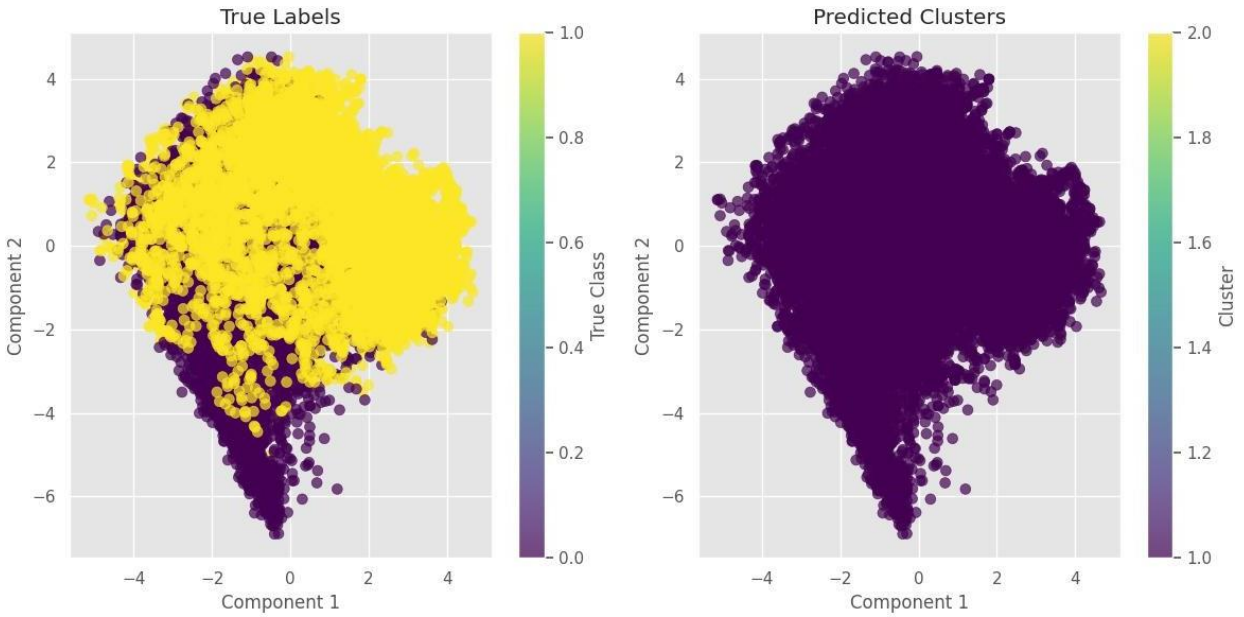


Other Metrics:

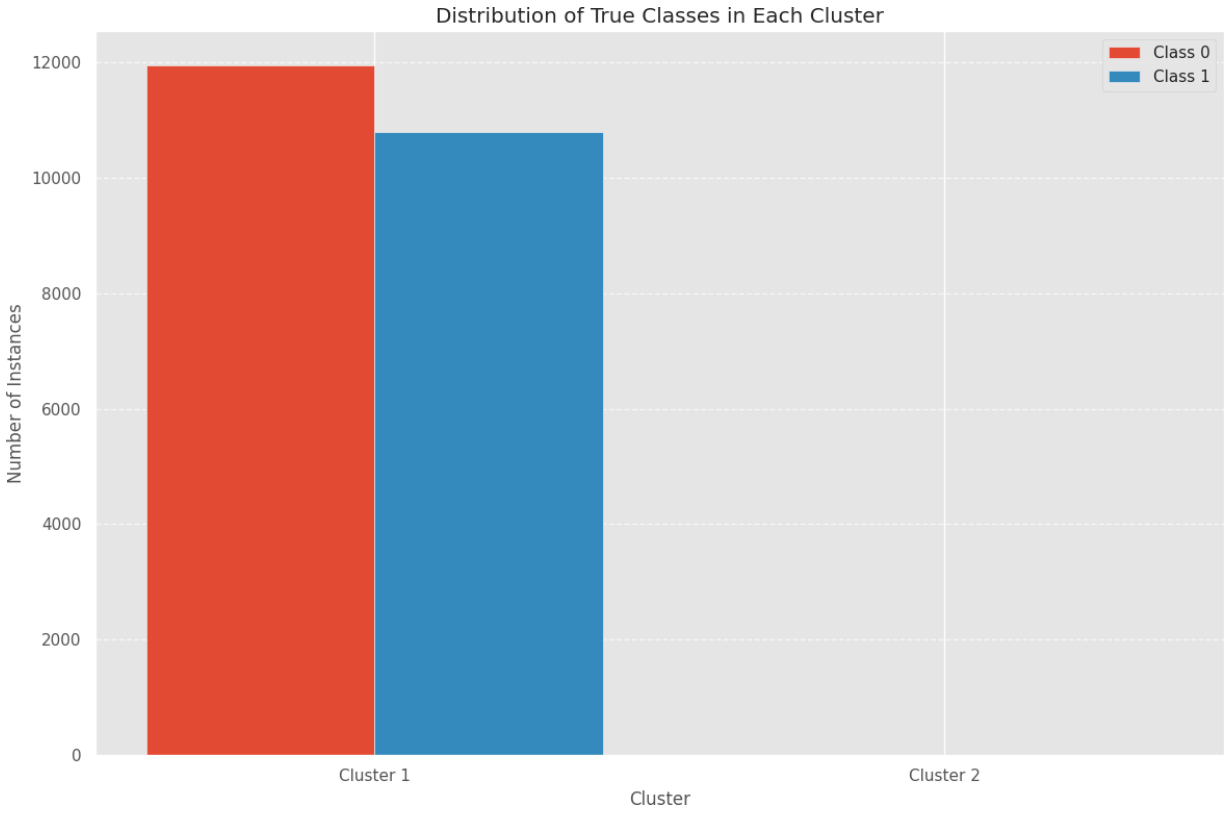
Normalized Mutual Information (NMI): 8.184363097684478e-05

silhouette_score: 0.18316844556585554

Comparison with true labels:



Class Distribution among clusters:



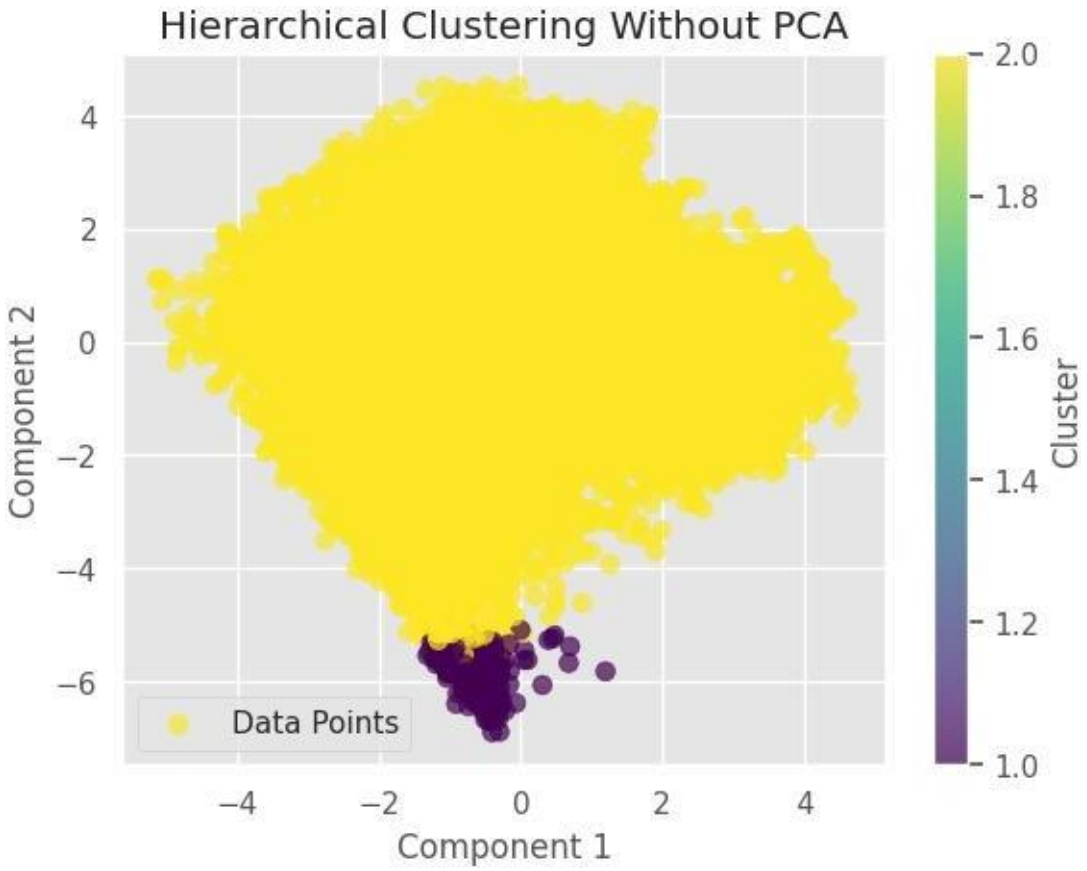
Cluster 1 has 11934 datapoints with label 0 and 10779 datapoints with label 1.

Cluster 2 has 1 datapoints with label 0 and 0 datapoints with label 1.

Average linkage gives horrible performance on our data.

Centroid Linkage:

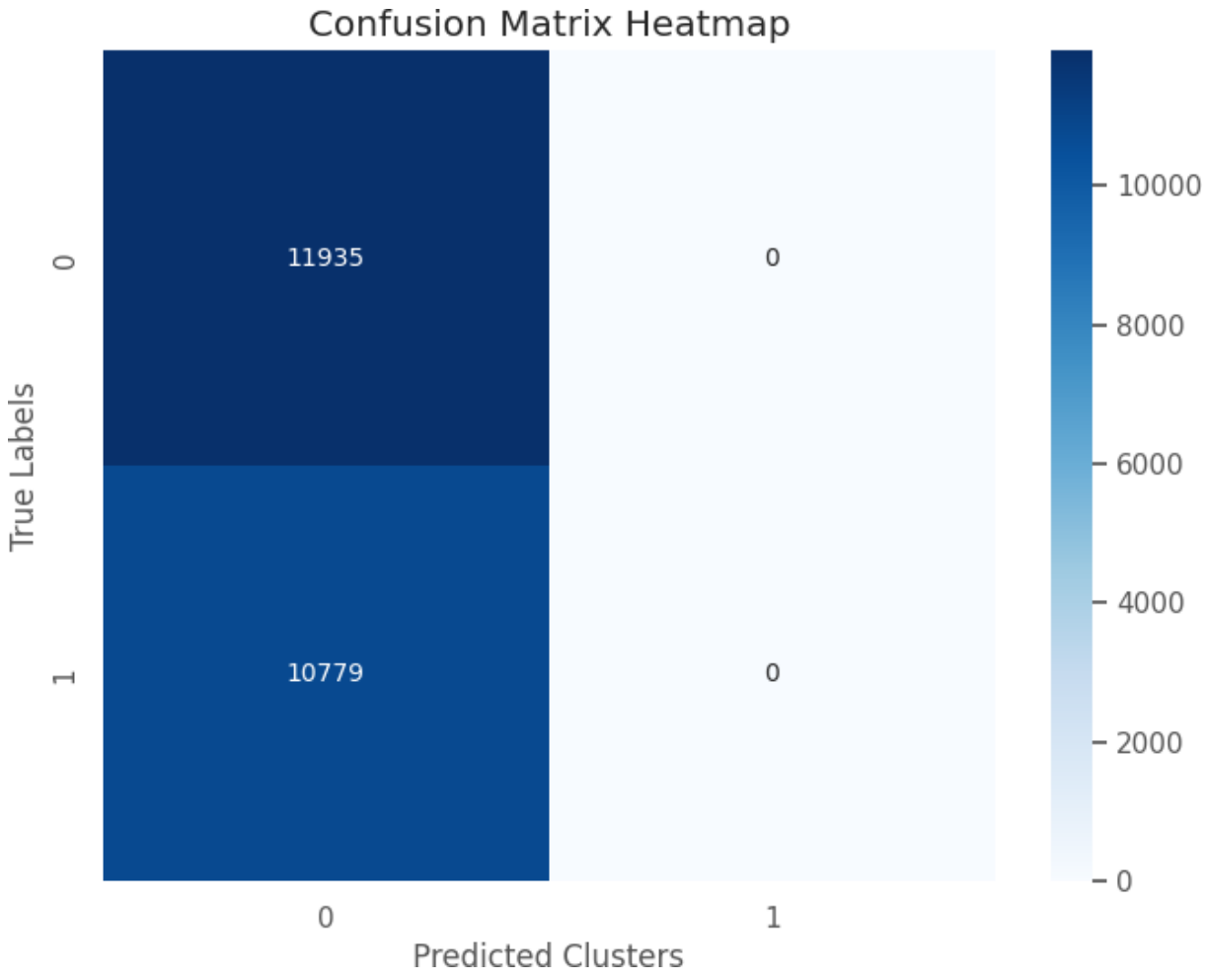
Visualization:



Evaluation Metrics:

Cluster Purity: 0.5254468609668046

Confusion matrix:

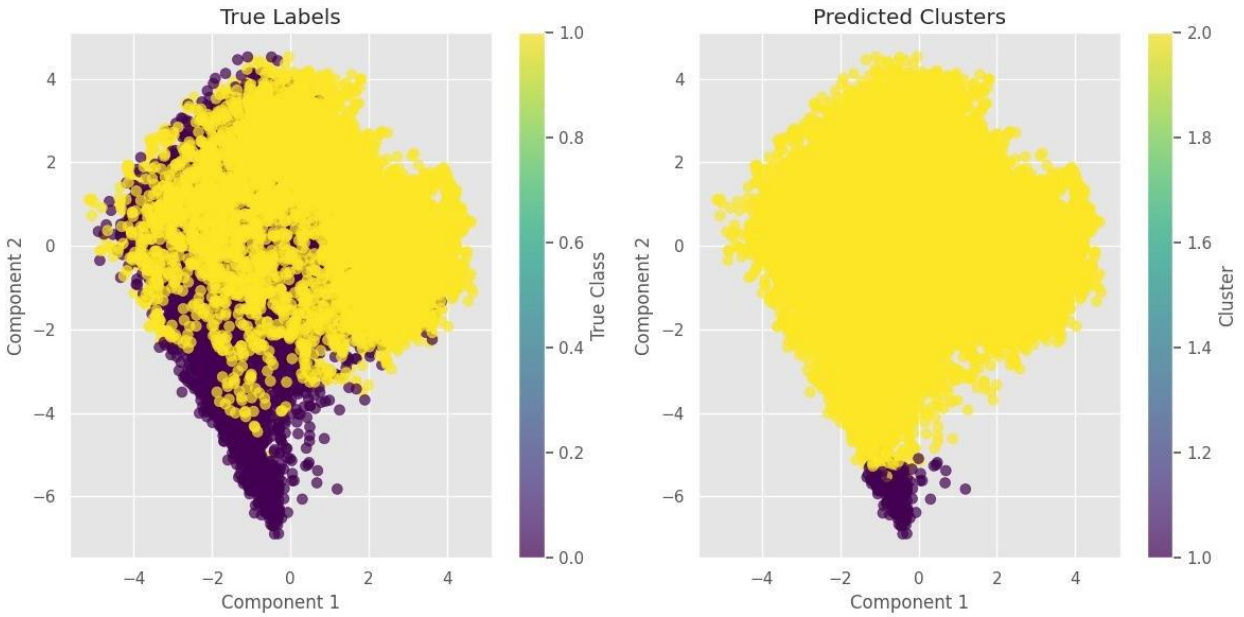


Other Metrics:

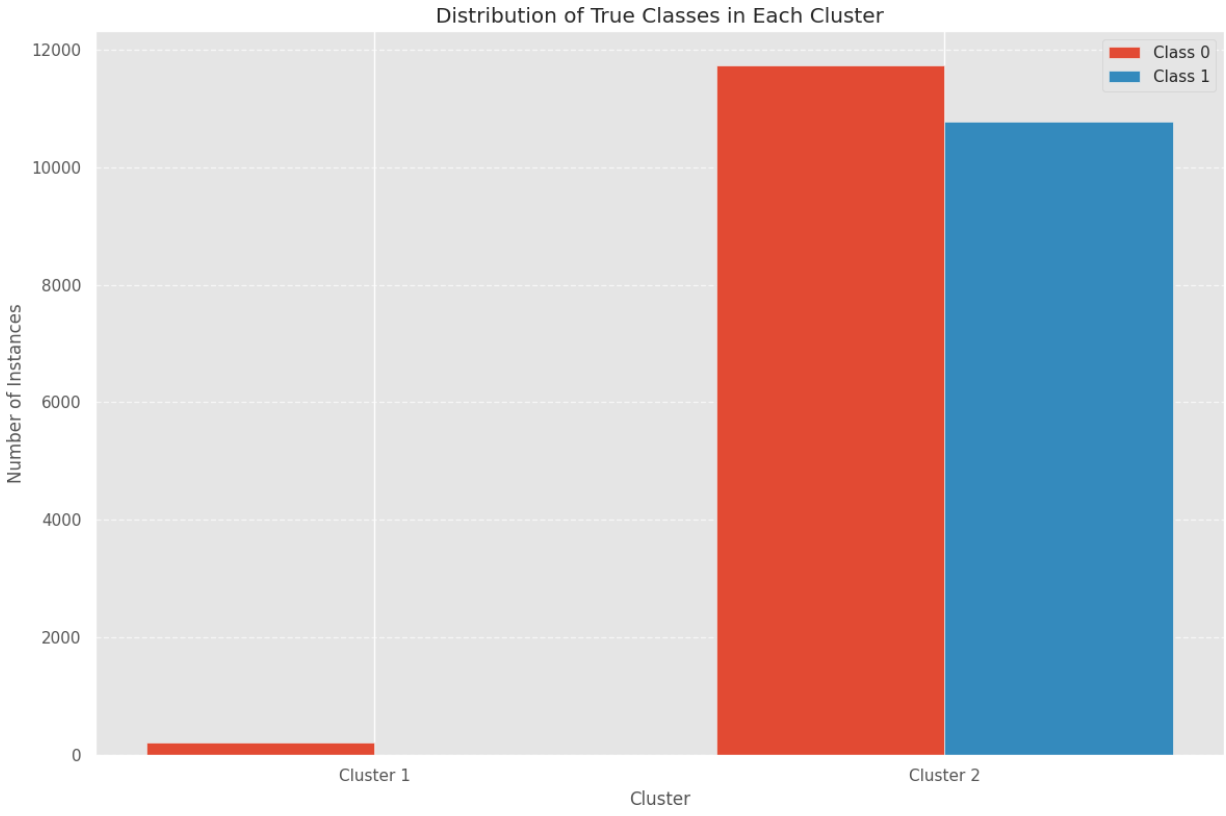
Normalized Mutual Information (NMI): 0.016308087347091608

silhouette_score: 0.29811604141691084

Comparison with true labels:



Class Distribution among clusters:



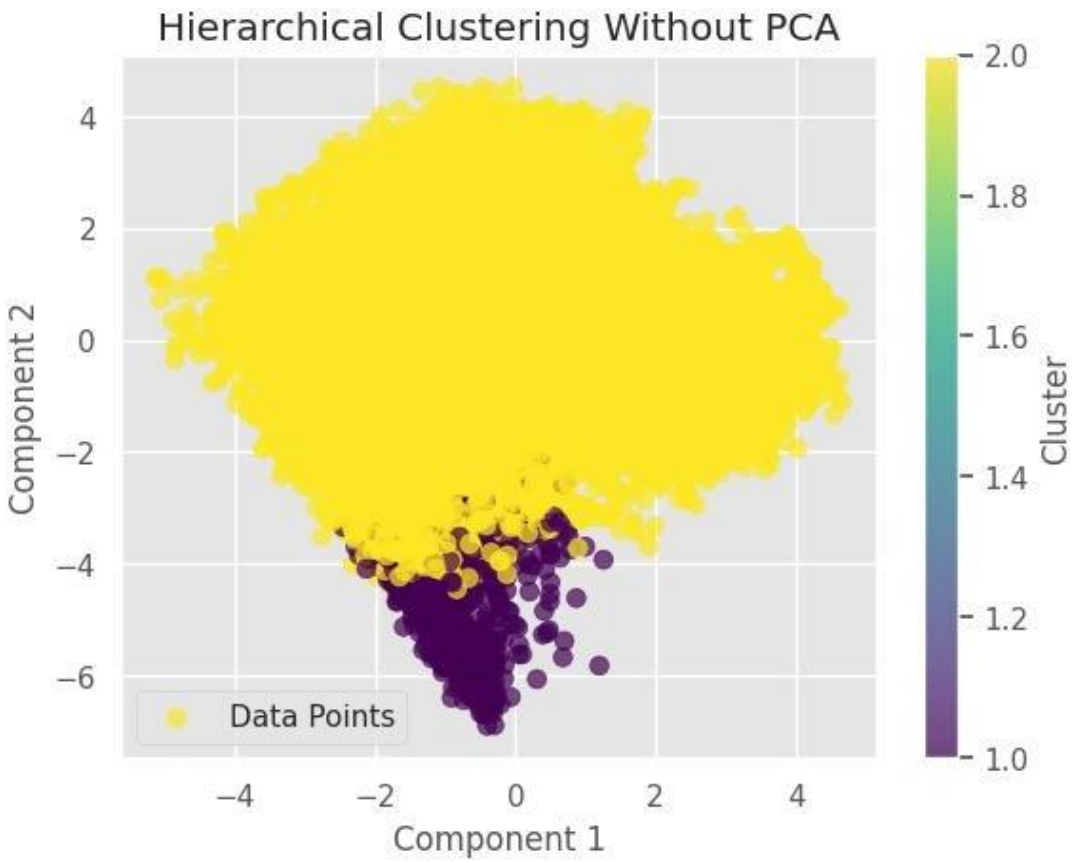
Cluster 1 has 213 datapoints with label 0 and 0 datapoints with label 1.

Cluster 2 has 11722 datapoints with label 0 and 10779 datapoints with label 1.

Centroid linkage gives horrible performance on our data.

Complete Linkage:

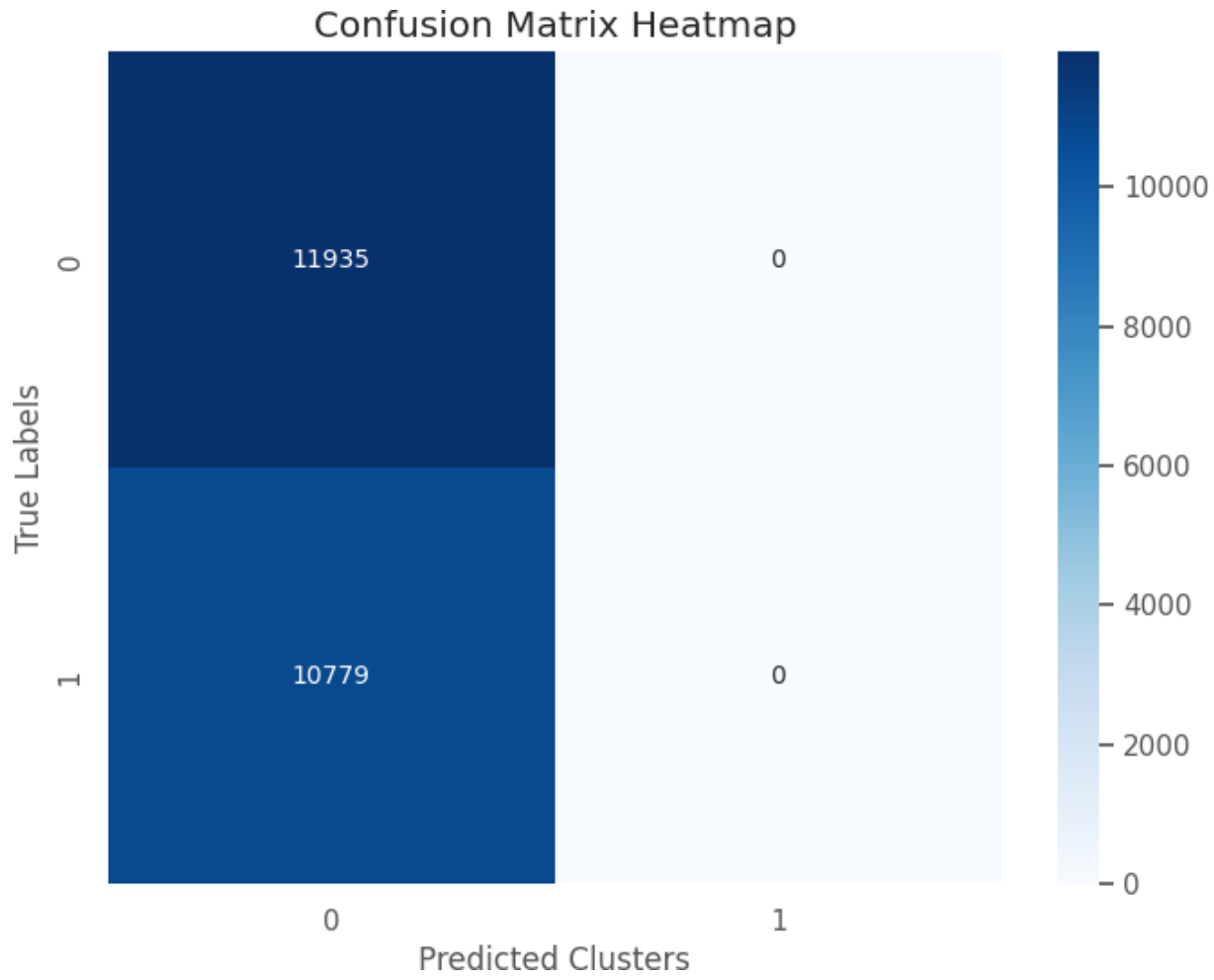
Visualization:



Evaluation Metrics:

Cluster Purity: 0.5254468609668046

Confusion matrix:

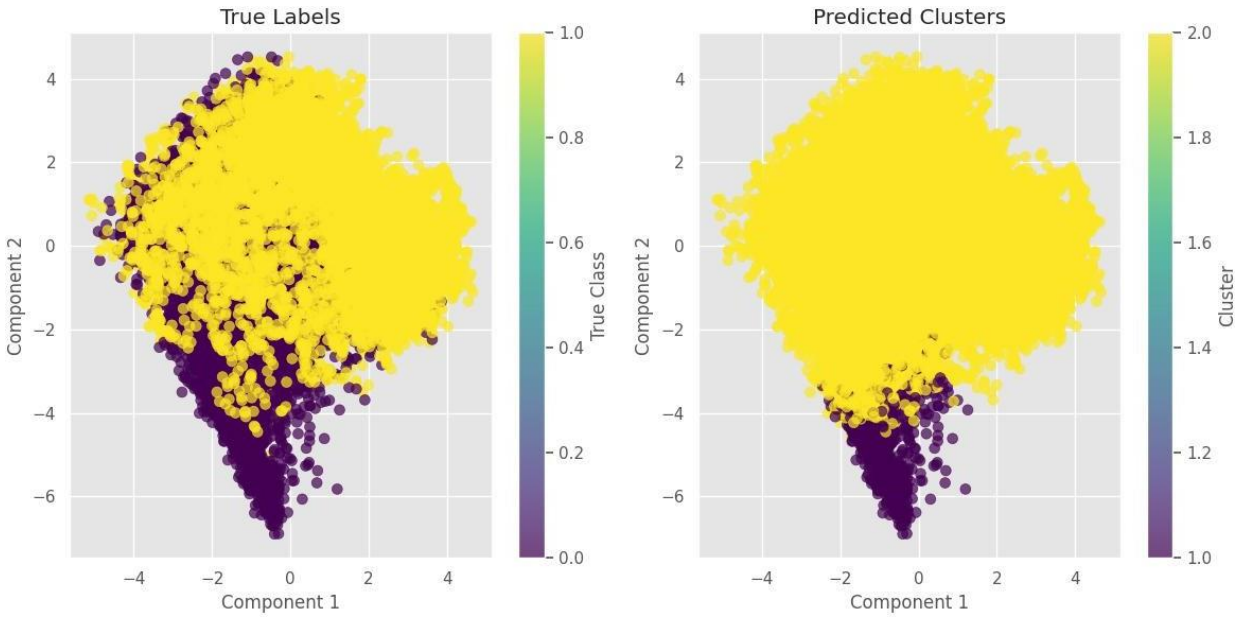


Other Metrics:

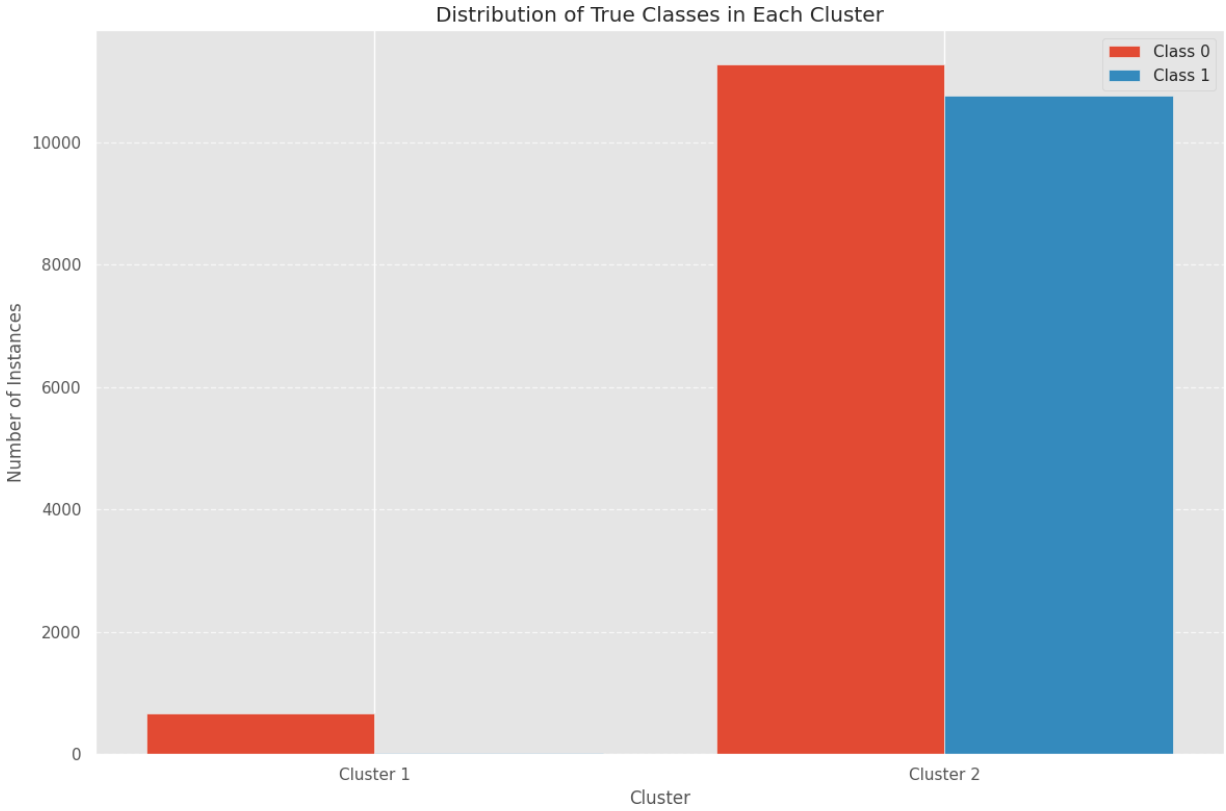
Normalized Mutual Information (NMI): 0.039098158483605164

silhouette_score: 0.23334808537779725

Comparison with true labels:



Class Distribution among clusters:

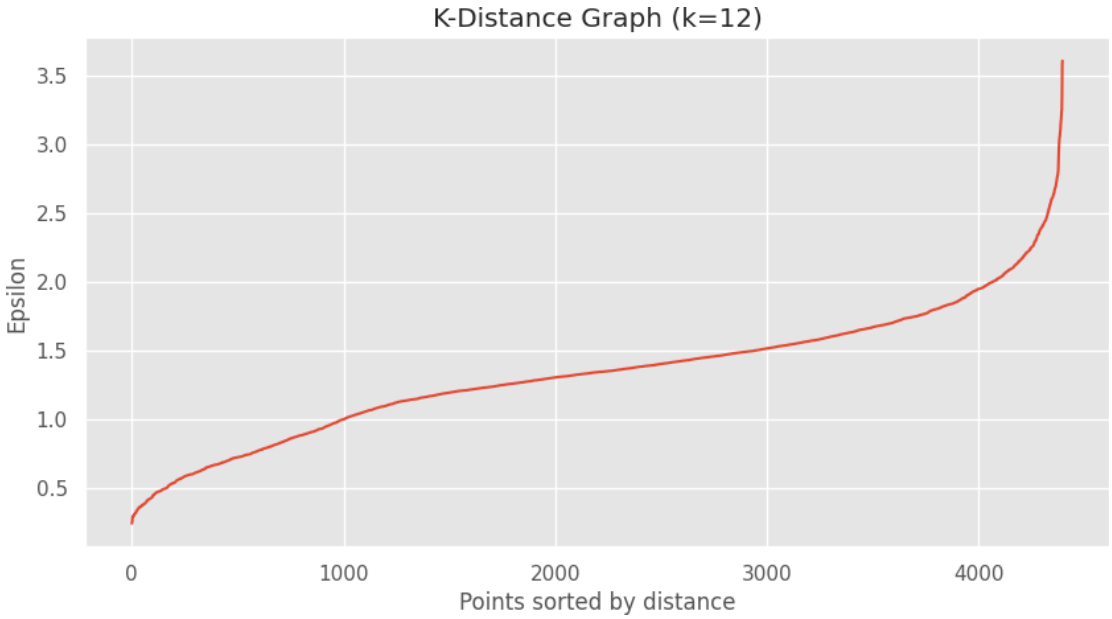


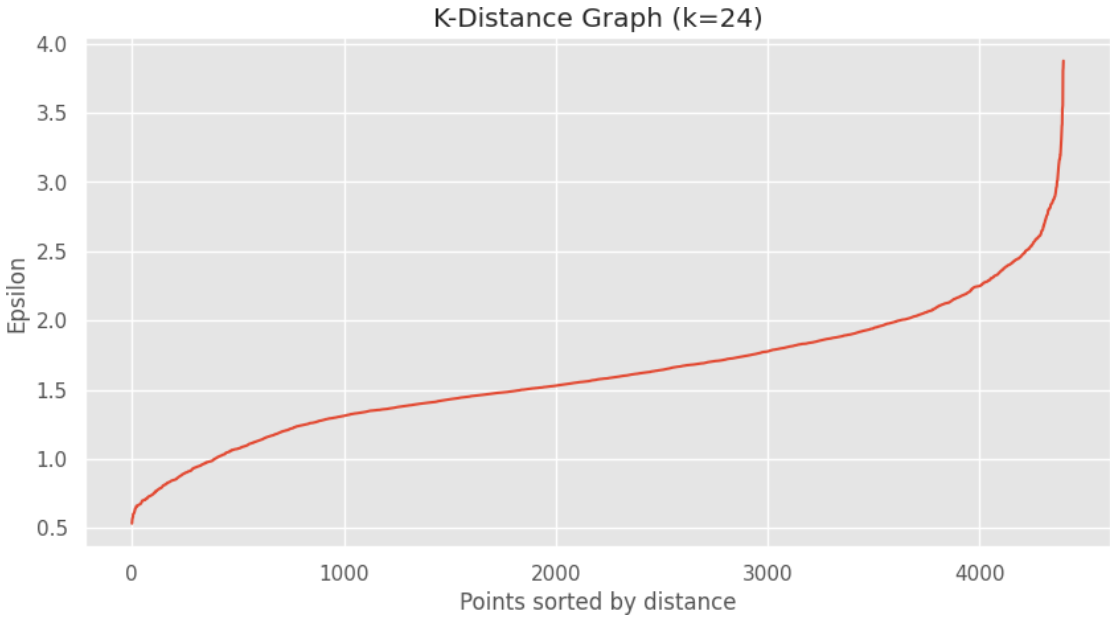
Cluster 1 has 671 datapoints with label 0 and 19 datapoints with label 1.

Cluster 2 has 11264 datapoints with label 0 and 10760 datapoints with label 1.

13.2.1.3 DBScan Clustering:

Elbow method:





Elbow method shows eps=2 for min samples 12 and 24. We usually take value of k>= dimension

of data. Dimension of our data is 12. It is recommended to keep k between Dimentionality + 1 and 2 * Dimentionality.

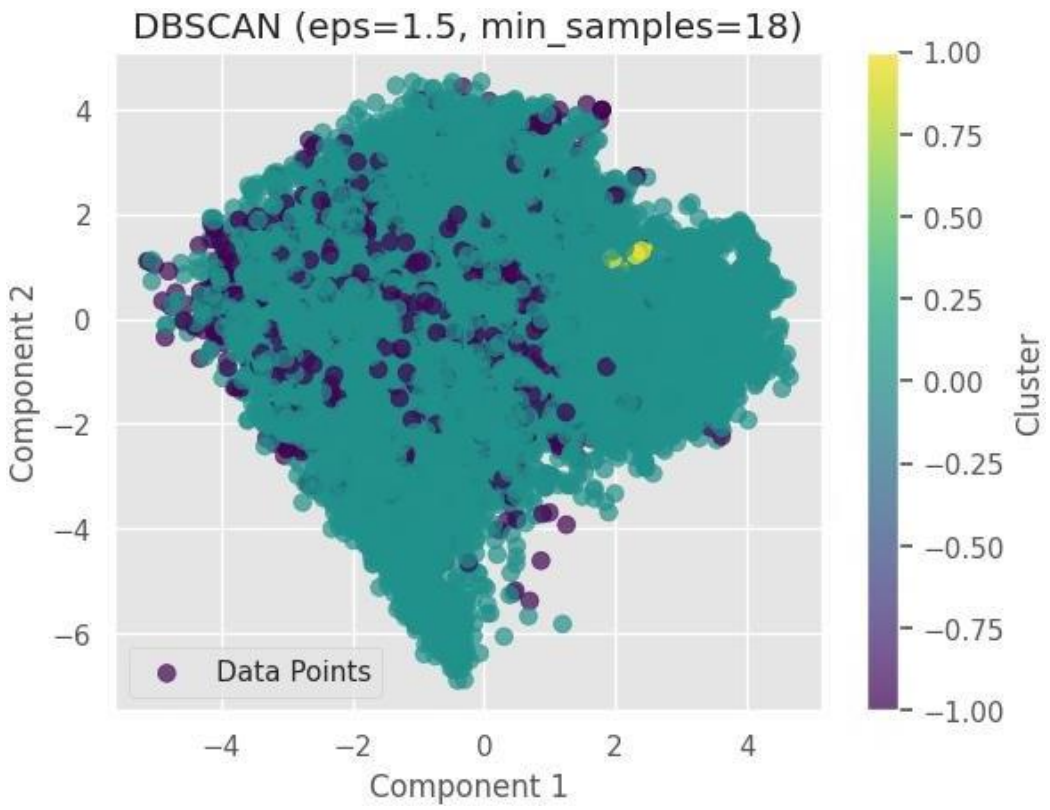
We tried experimenting DBScan for various values.

eps_range = [0.5, 1, 1.5, 2, 2.5, 3]

min_samples_range = [12, 15, 18, 21, 24]

None of them gave satisfactory results. Either it makes a lot of clusters, or it makes 1 cluster with some noise points. This experiment clearly shows that DBScan is not suitable for our dataset because of overlapping clusters. We got 2 clusters only for

Visualization:

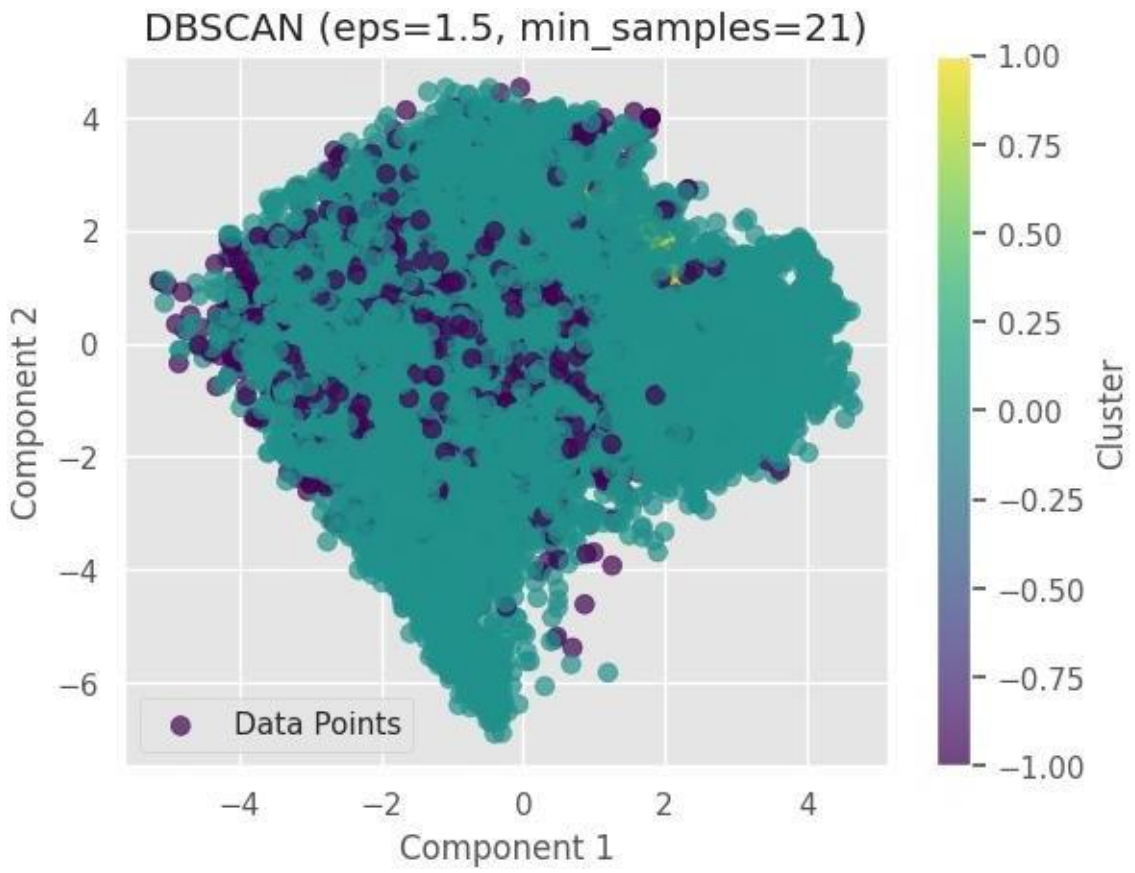


eps=1.5, min_samples=18

Number of clusters (excluding noise): 2

Number of noise points: 1848

Silhouette score: -0.012321333696672376



eps=1.5, min_samples=21

Number of clusters (excluding noise): 2

Number of noise points: 2089

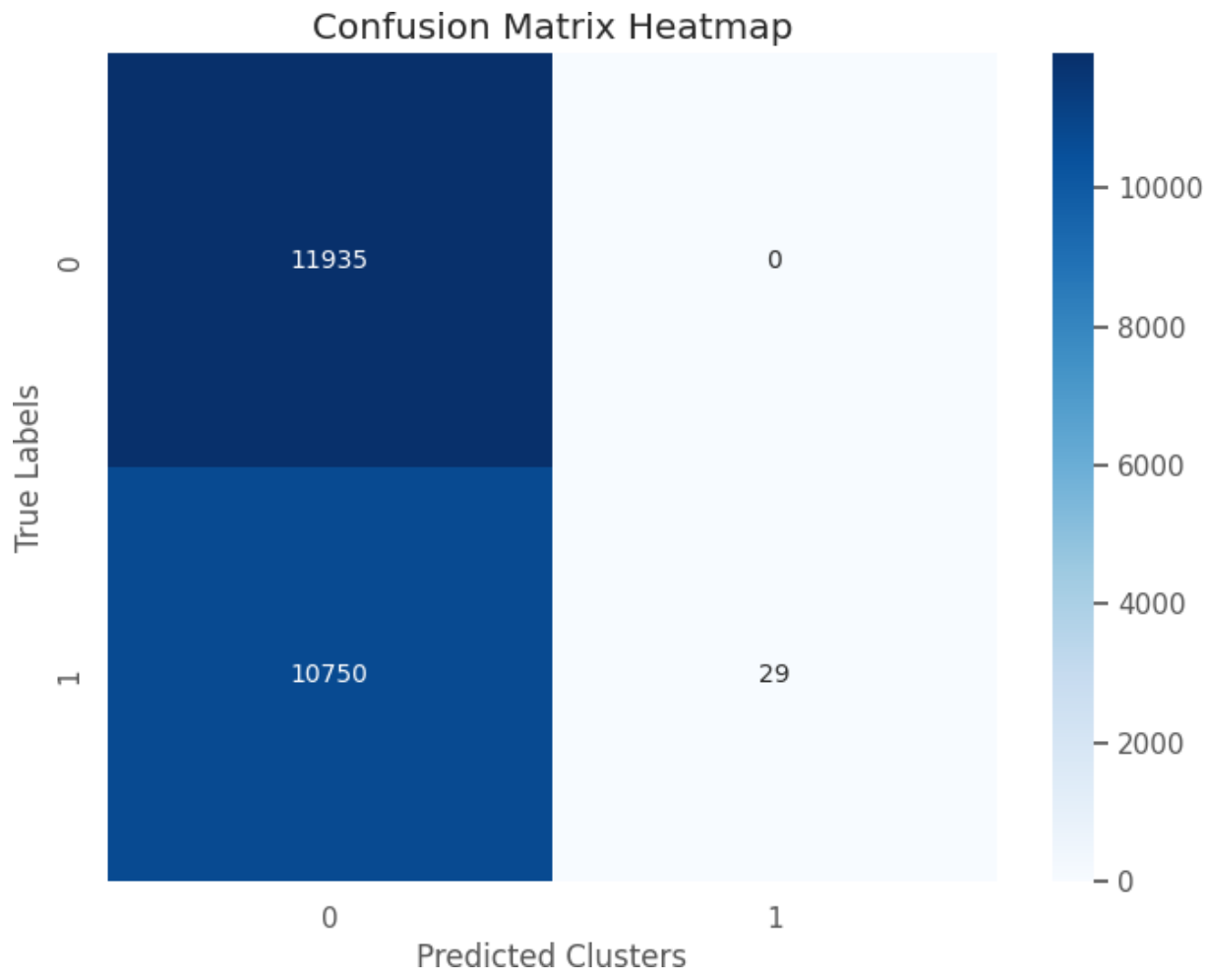
Silhouette score: 0.016089331140922394

We choose eps=1.5, min_samples=18

Evaluation Metrics:

Cluster Purity: 0.5267236065862464

Confusion matrix:

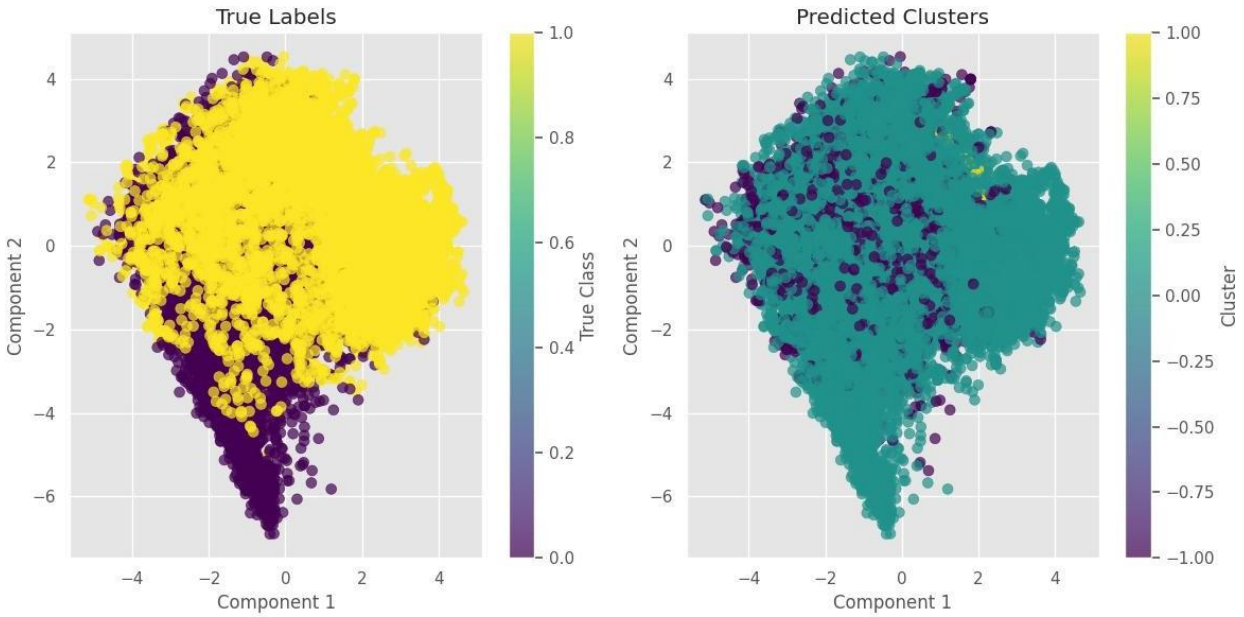


Other Metrics:

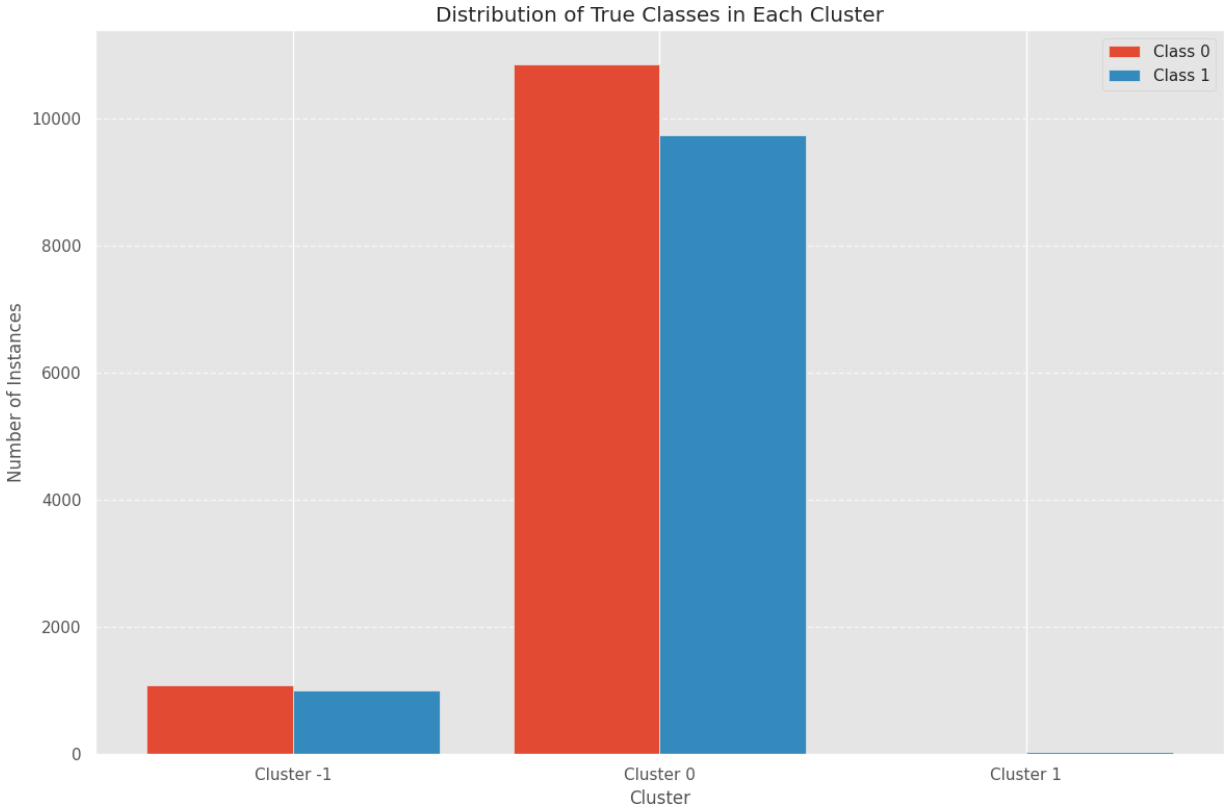
Normalized Mutual Information (NMI): 0.0019071657256047455

silhouette_score: 0.016089331140922394

Comparison with true labels:



Class Distribution among clusters:



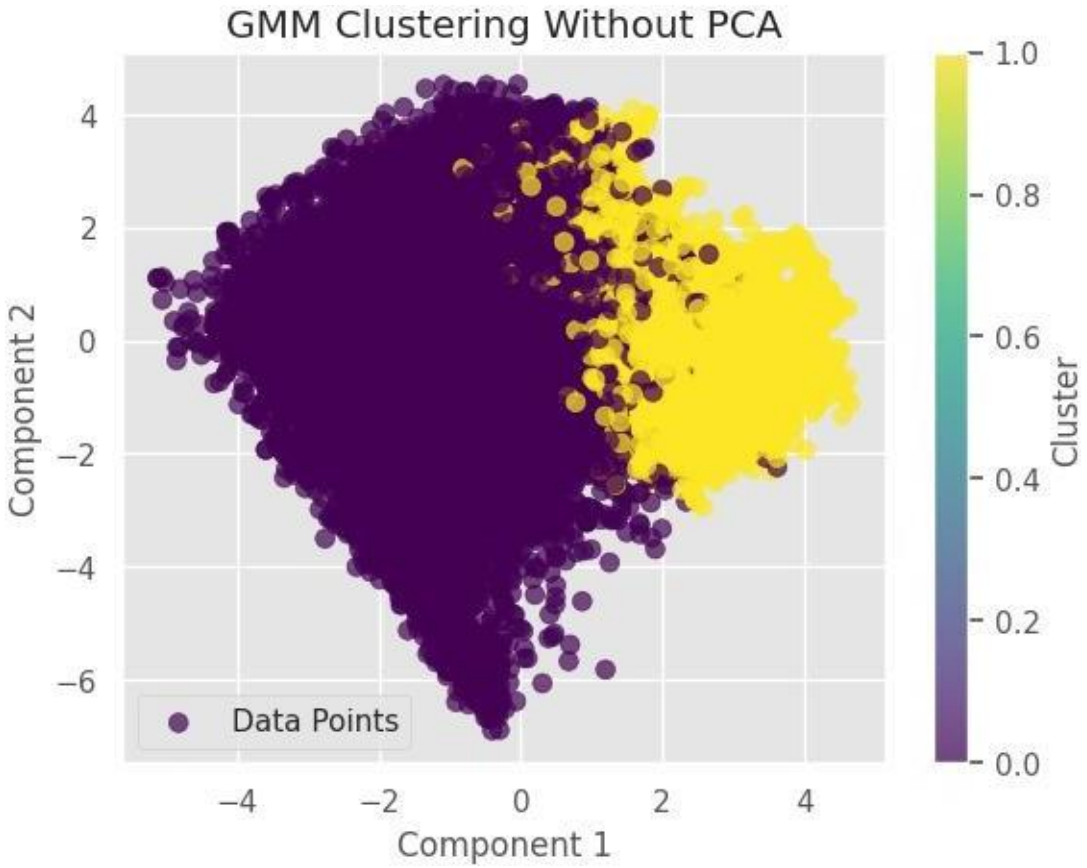
Cluster -1 has 1085 datapoints with label 0 and 1004 datapoints with label 1.

Cluster 0 has 10850 datapoints with label 0 and 9746 datapoints with label 1.

Cluster 1 has 0 datapoints with label 0 and 29 datapoints with label 1.

13.2.1.4 Gaussian Mixture Model Clustering*:*

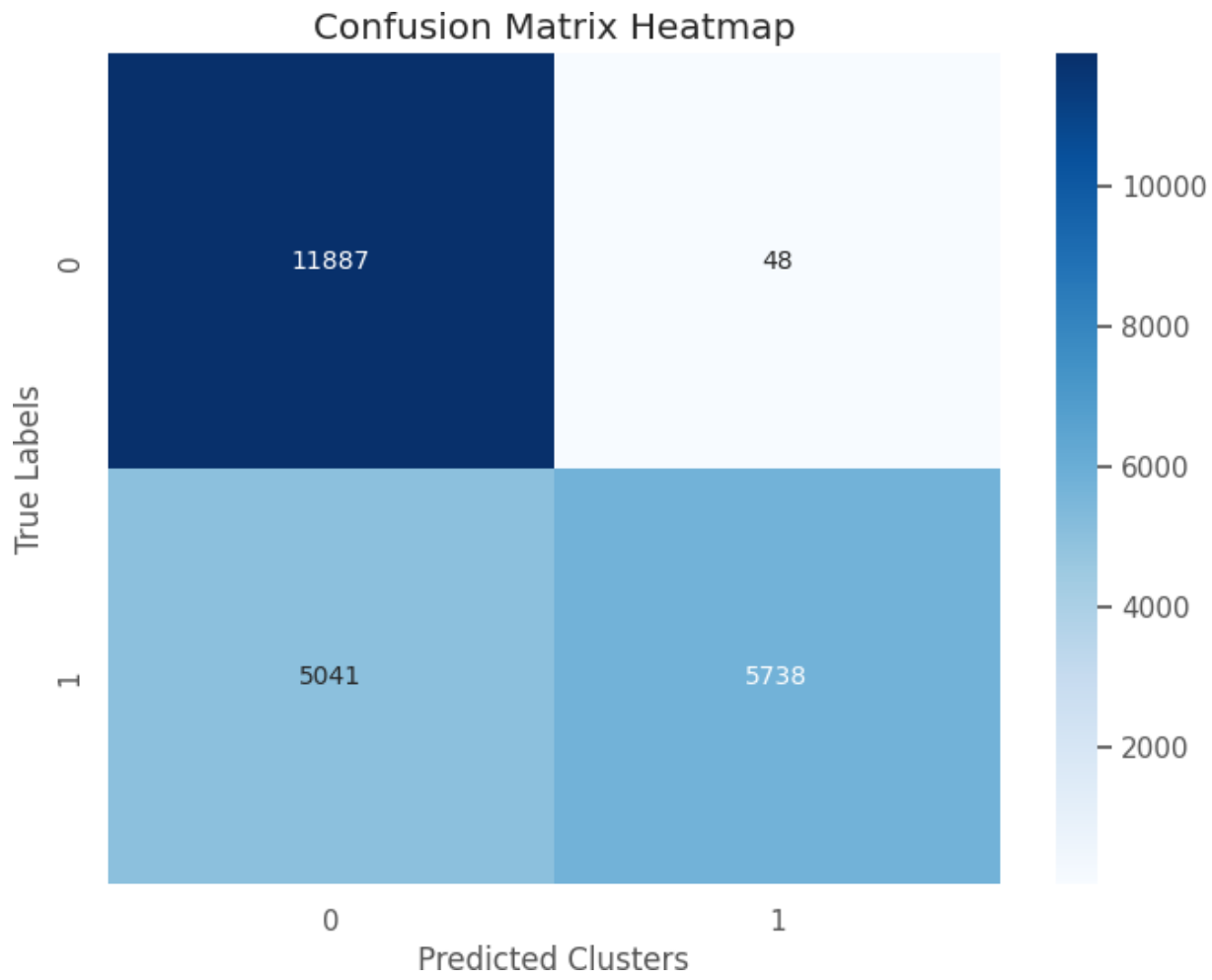
Visualization:



Evaluation Metrics:

Cluster Purity: 0.7759531566434797

Confusion matrix:

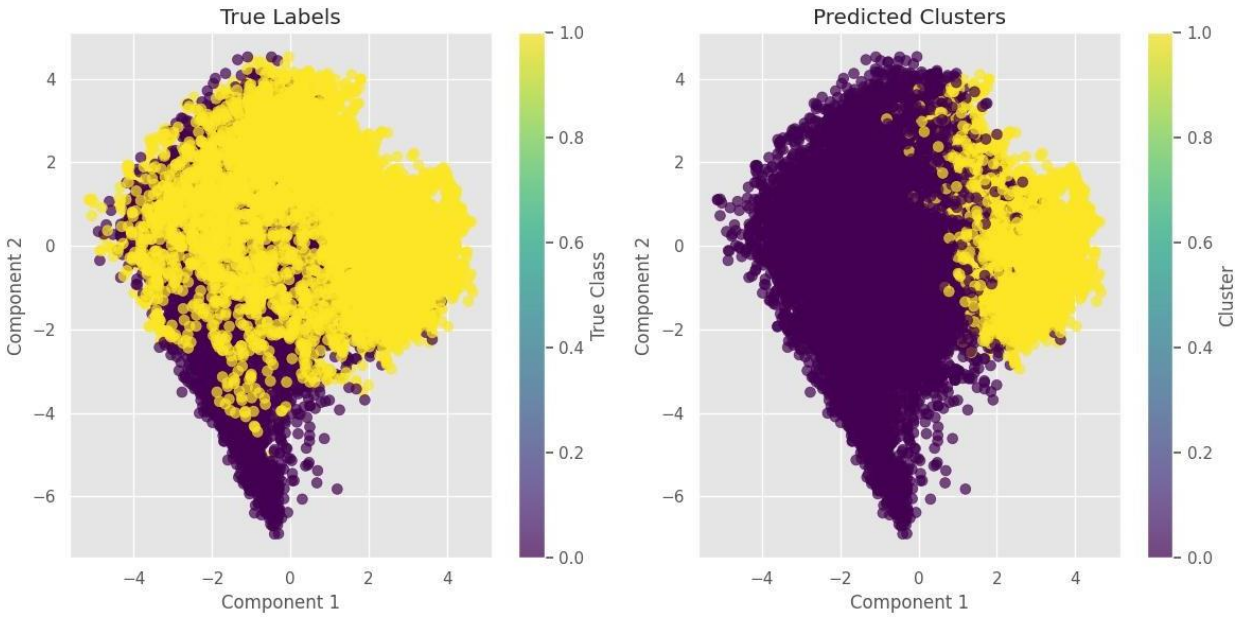


Other Metrics:

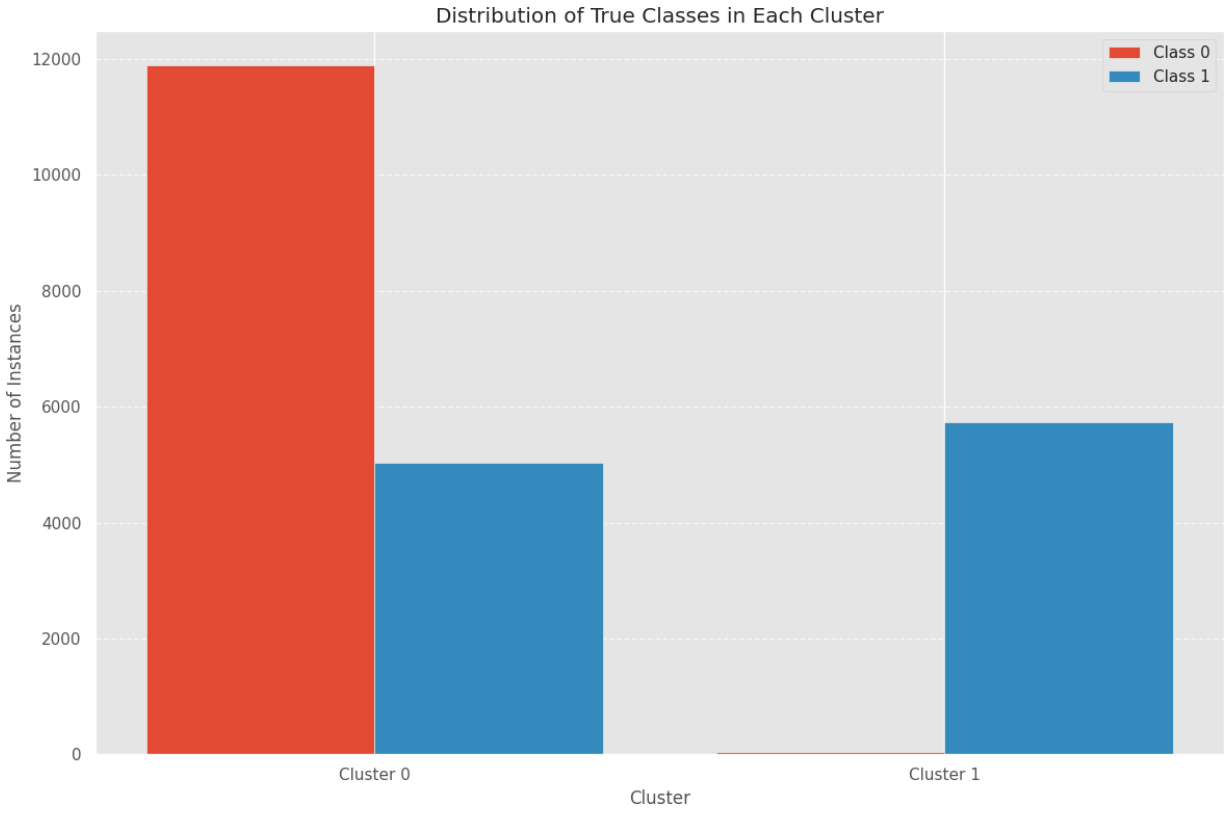
Normalized Mutual Information (NMI: 0.35855213741615977

silhouette_score: 0.2128829224023907

Comparison with true labels:



Class Distribution among clusters:



Cluster 0 has 11887 datapoints with label 0 and 5041 datapoints with label 1.

Cluster 1 has 48 datapoints with label 0 and 5738 datapoints with label 1.

13.2.1.5 Model Comparison:

Here's a detailed comparison of clustering models for our dataset:

1. K-Means Clustering

Evaluation Metrics

- **Cluster Purity:** 0.801
- **NMI:** 0.3491
- **Silhouette Score:** 0.2243

Performance

- Visualization reveals clear separations when k=2k=2 (forced due to labels).
- **Class Distribution:**
  - Cluster 0: 11563 (label 0), 4148 (label 1)
  - Cluster 1: 372 (label 0), 6631 (label 1)

Strengths

- Best purity score among all methods.
- Suitable for datasets with distinct spherical clusters.

Limitations

- Lower silhouette score suggests suboptimal cluster compactness.

2. Hierarchical Clustering

Evaluation Metrics

Ward Linkage:

- **Cluster Purity:** 0.758
- **NMI:** 0.3101

- **Silhouette Score:** 0.2043

Average Linkage:

- **Cluster Purity:** 0.525
- **NMI:** ~0
- **Silhouette Score:** 0.1832

Centroid Linkage:

- **Cluster Purity:** 0.525
- **NMI:** 0.0163
- **Silhouette Score:** 0.2981

Complete Linkage:

- **Cluster Purity:** 0.525
- **NMI:** 0.0391
- **Silhouette Score:** 0.2333

Performance

- Best configuration: **Ward Linkage**.
- **Class Distribution (Ward Linkage):**
    - Cluster 1: 146 (label 0), 5429 (label 1)
    - Cluster 2: 11789 (label 0), 5350 (label 1)

Strengths

- Can capture different shapes and densities of clusters.
- Good performance with Ward linkage.

Limitations

- Poor performance with other linkage methods.
- Lower purity compared to K-Means.

3. DBSCAN Clustering

Evaluation Metrics

- **Cluster Purity:** 0.5267
- **NMI:** 0.0019
- **Silhouette Score:** 0.0161

Performance

- Best configuration: epsilon=1.5, min_samples=18.
- **Class Distribution:**
  - Cluster -1 (Noise): 1085 (label 0), 1004 (label 1)
  - Cluster 0: 10850 (label 0), 9746 (label 1)
  - Cluster 1: 0 (label 0), 29 (label 1)

Strengths

- Can detect noise and anomalies.

Limitations

- Overlapping clusters and noise points lead to suboptimal results.
- Negative silhouette score indicates poor clustering quality.

4. Gaussian Mixture Model (GMM)

Evaluation Metrics

- **Cluster Purity:** 0.776
- **NMI:** 0.3586
- **Silhouette Score:** 0.2129

Performance

- **Class Distribution:**
  - Cluster 0: 11887 (label 0), 5041 (label 1)
  - Cluster 1: 48 (label 0), 5738 (label 1)

Strengths

- Handles overlapping clusters better than DBSCAN.
- Comparable NMI to K-Means.

Limitations

- Slightly lower purity than K-Means.
- Assumes clusters follow a Gaussian distribution, which may not align with the data.

Summary of Model Comparison

Model	Cluster Purity	NMI	Silhouette Score	Best Use Case
K-Means	**0.801**	0.3491	0.2243	Well-separated clusters with equal densities.
Hierarchical (Ward)	0.758	0.3101	0.2043	Non-spherical clusters with varying densities.
DBSCAN	0.5267	0.0019	0.0161	Noise detection and arbitrary-shaped clusters.
GMM	0.776	**0.3586**	0.2129	Overlapping clusters with Gaussian distributions.

Recommendations

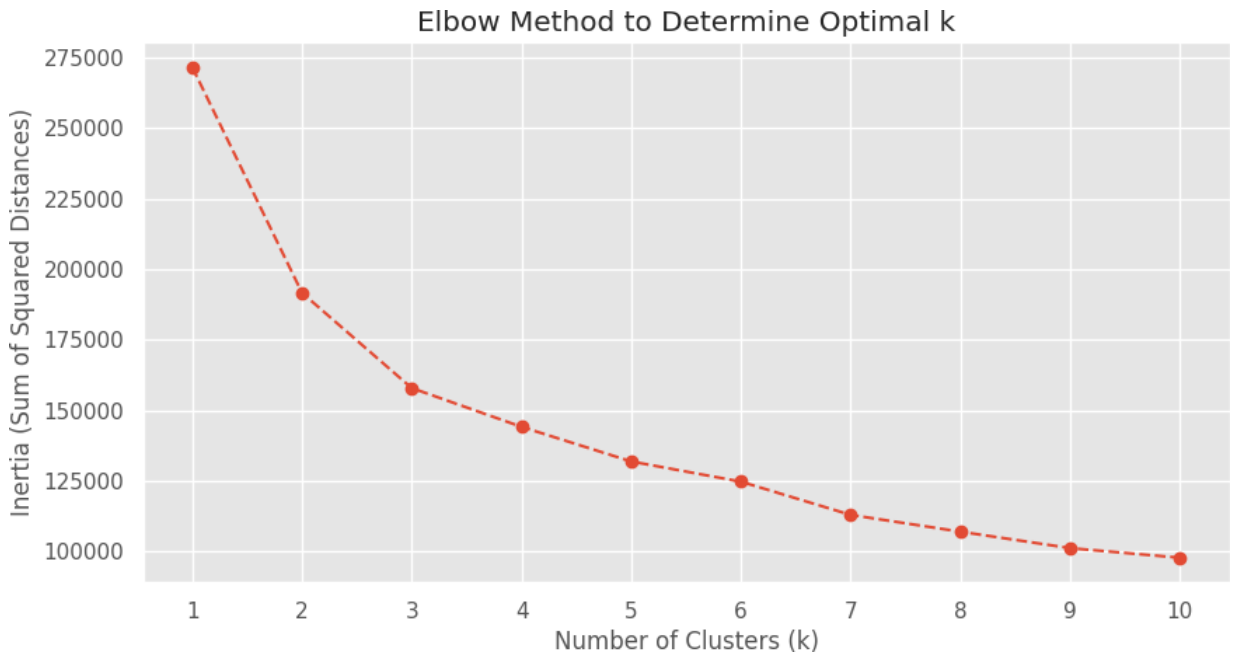
1. **K-Means:** Best overall performance for purity and NMI; a good fit for your labeled data with two classes.
2. **Hierarchical Clustering (Ward):** Suitable as an alternative if more interpretability is needed.
3. **DBSCAN:** Avoid unless the goal is to identify noise points.
4. **GMM:** Useful if cluster overlaps are expected and Gaussian assumptions hold.

13.3 Year3

13.3.1 SMOTE:

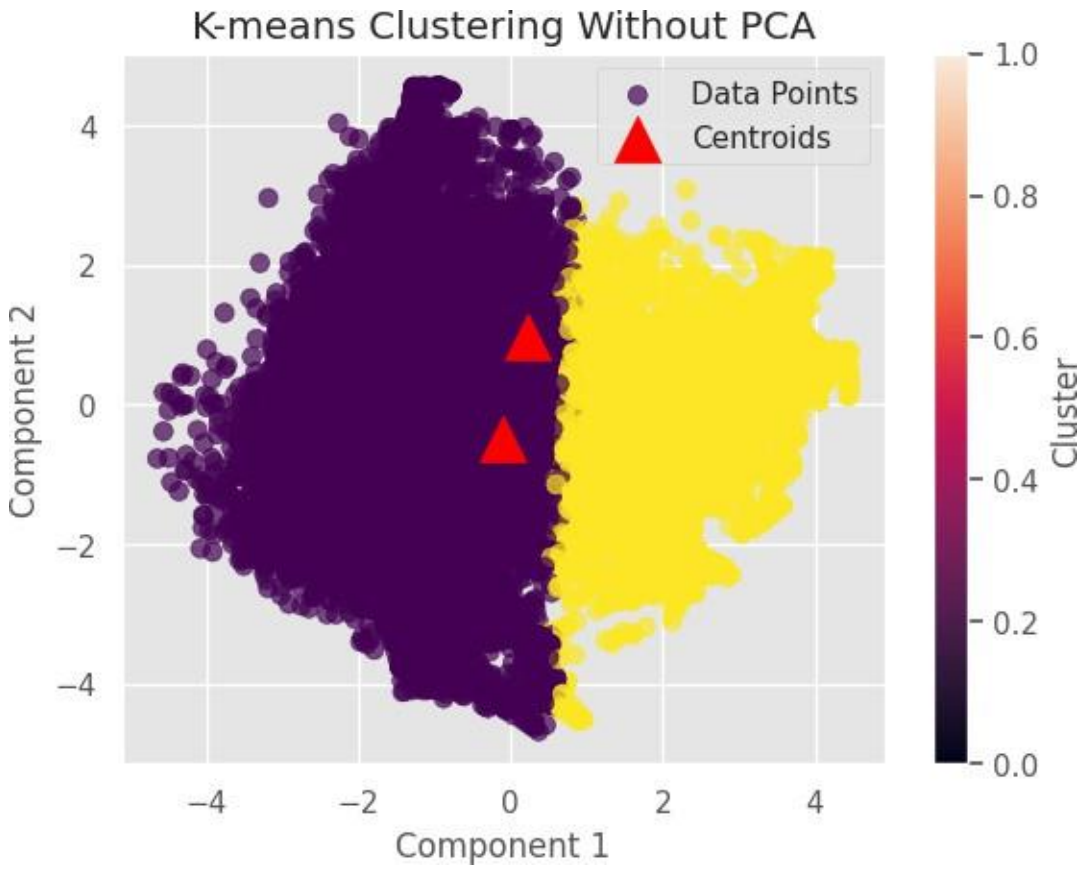
13.3.1.1 K Means Clustering:

Elbow method:



Elbow method shows k=3 but since our dataset is labelled and we are dealing with 2 classes, k = 2.

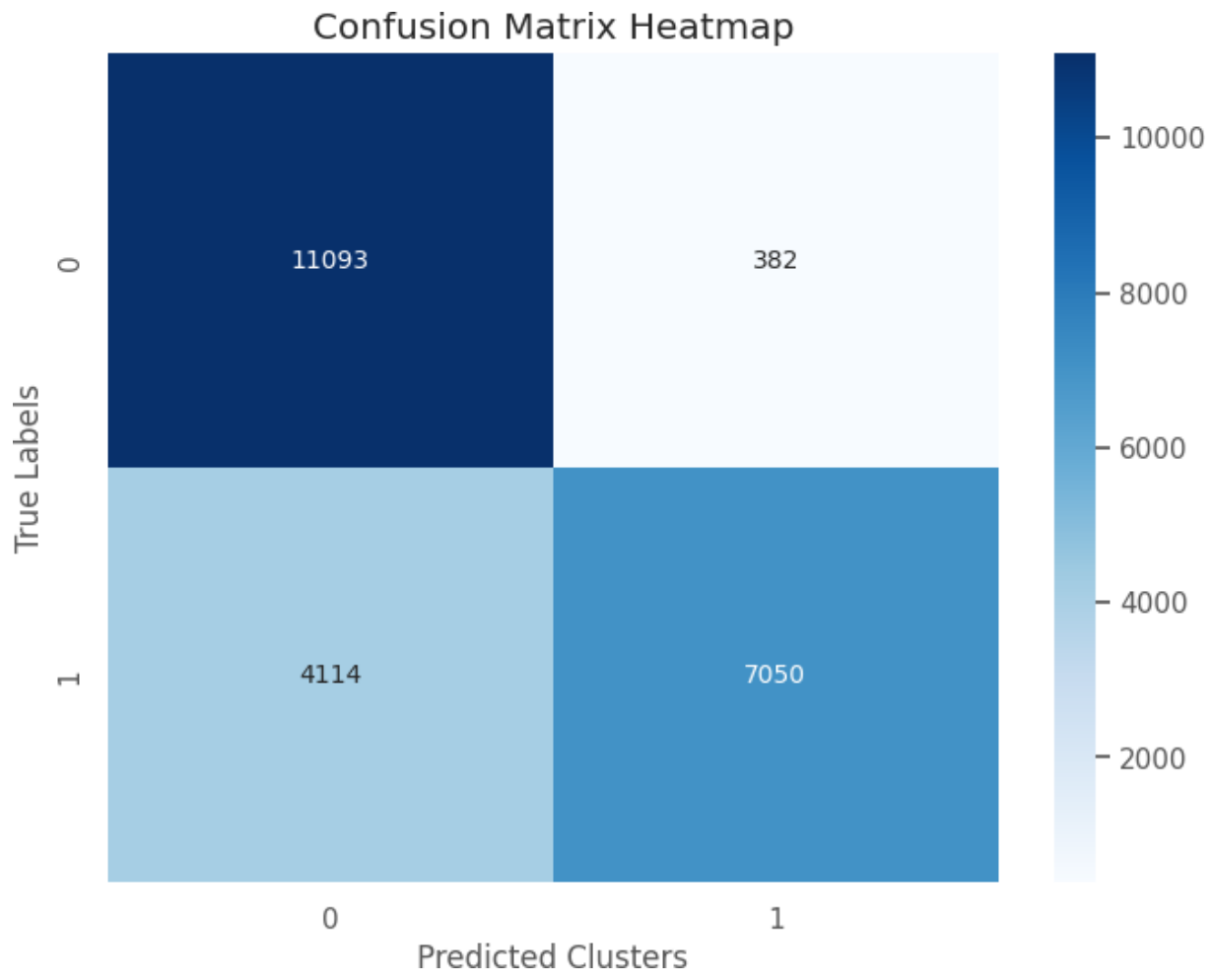
Visualization:



Evaluation Metrics:

Cluster Purity: 0.8014046556826715

Confusion matrix:

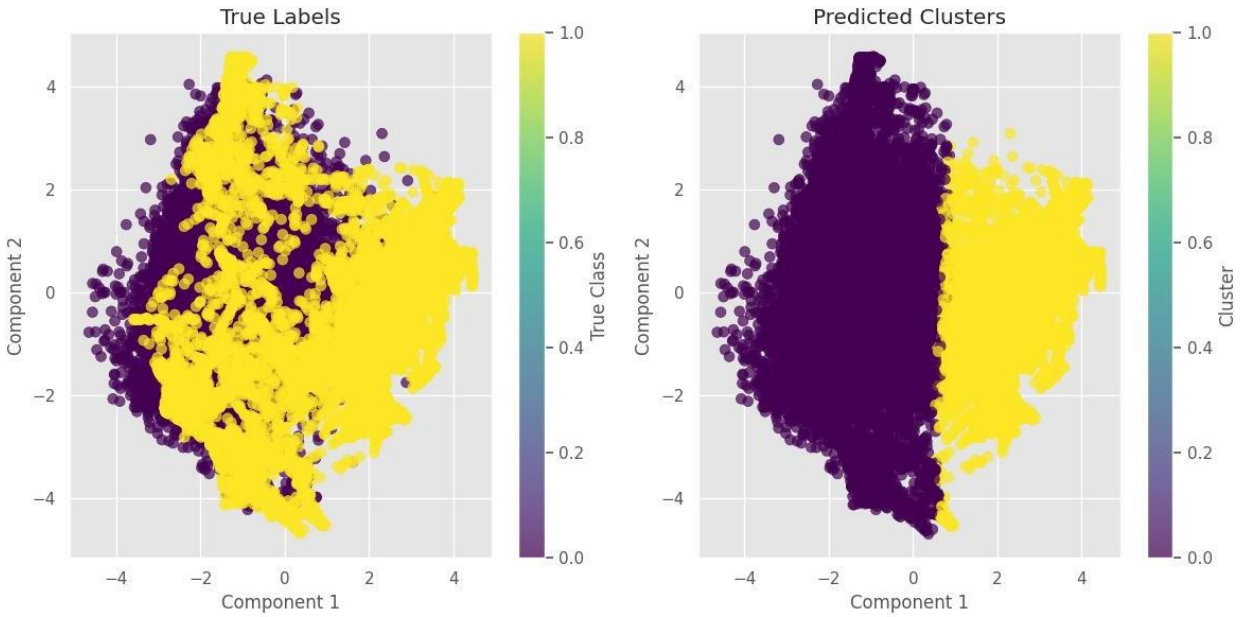


Other Metrics:

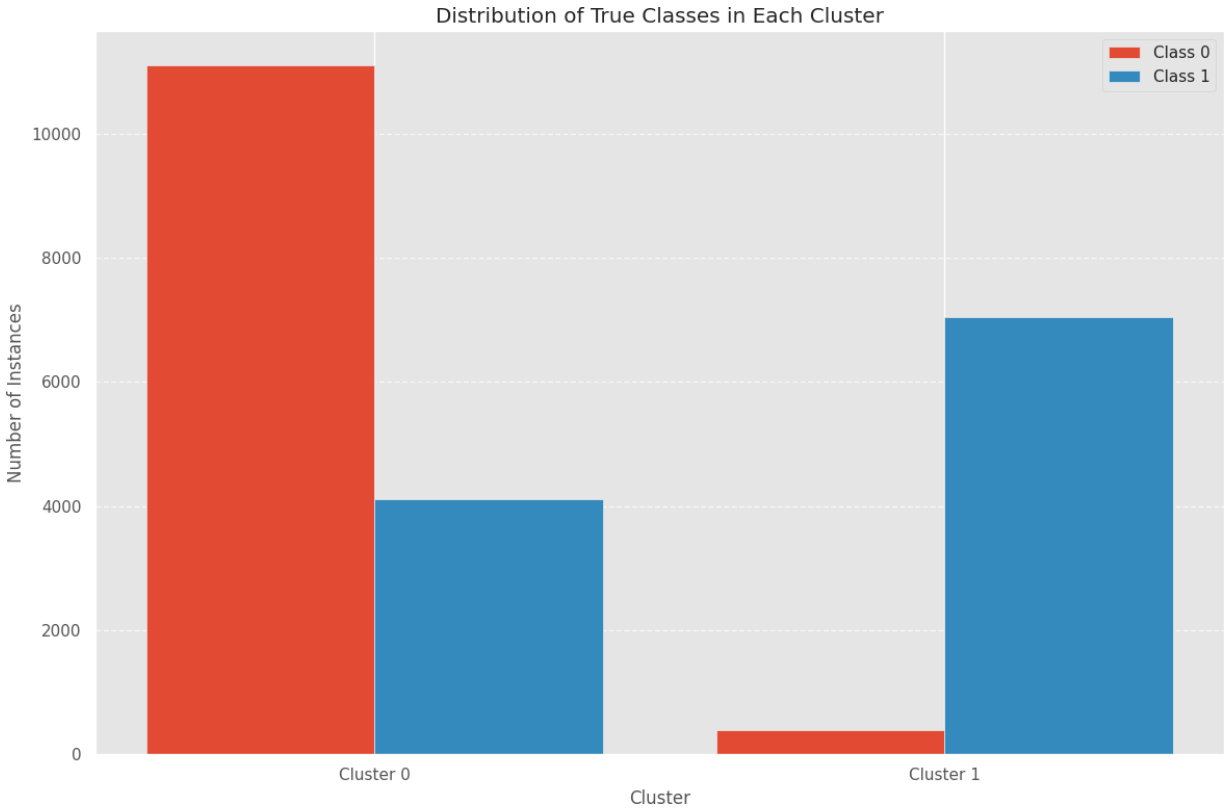
Normalized Mutual Information (NMI): 0.35353766667509484

silhouette_score: 0.292755094889243

Comparison with true labels:



Class Distribution among clusters:

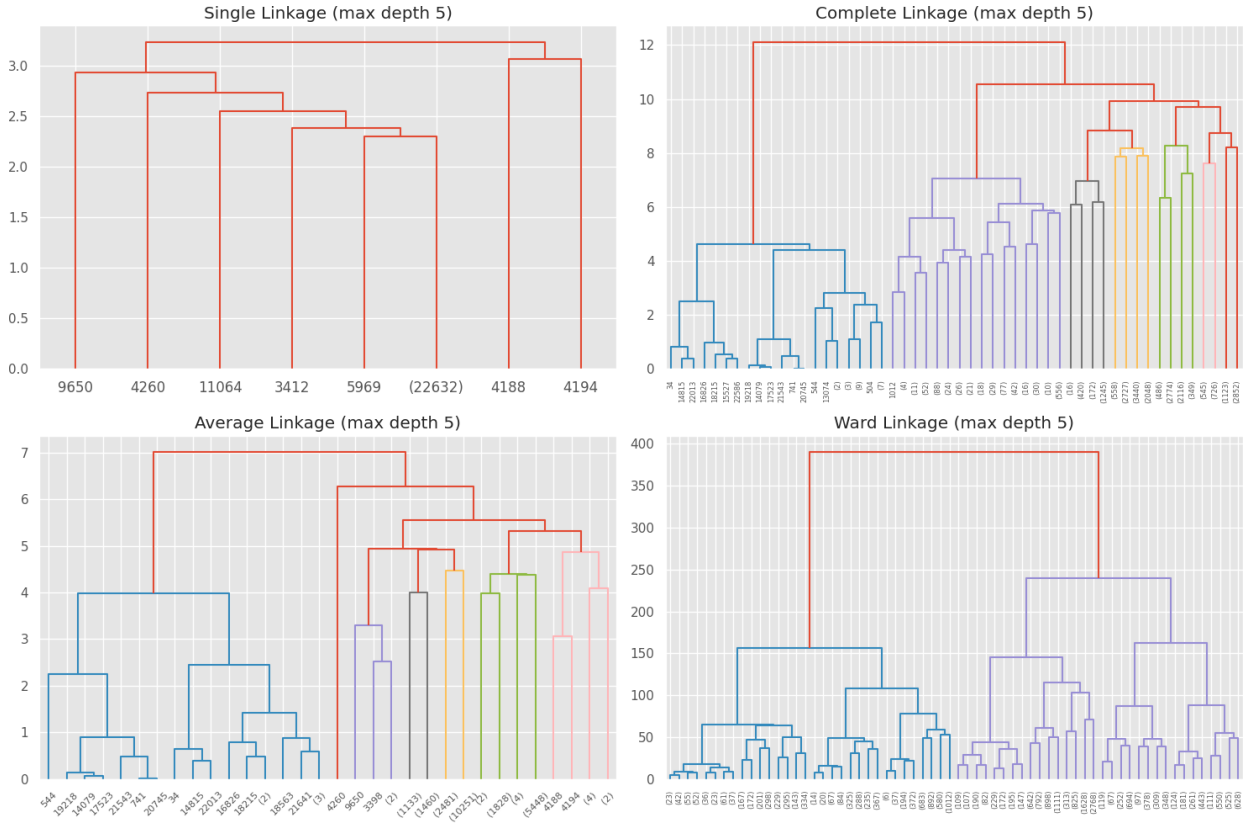


Cluster 0 has 11093 datapoints with label 0 and 4114 datapoints with label 1.

Cluster 1 has 382 datapoints with label 0 and 7050 datapoints with label 1.

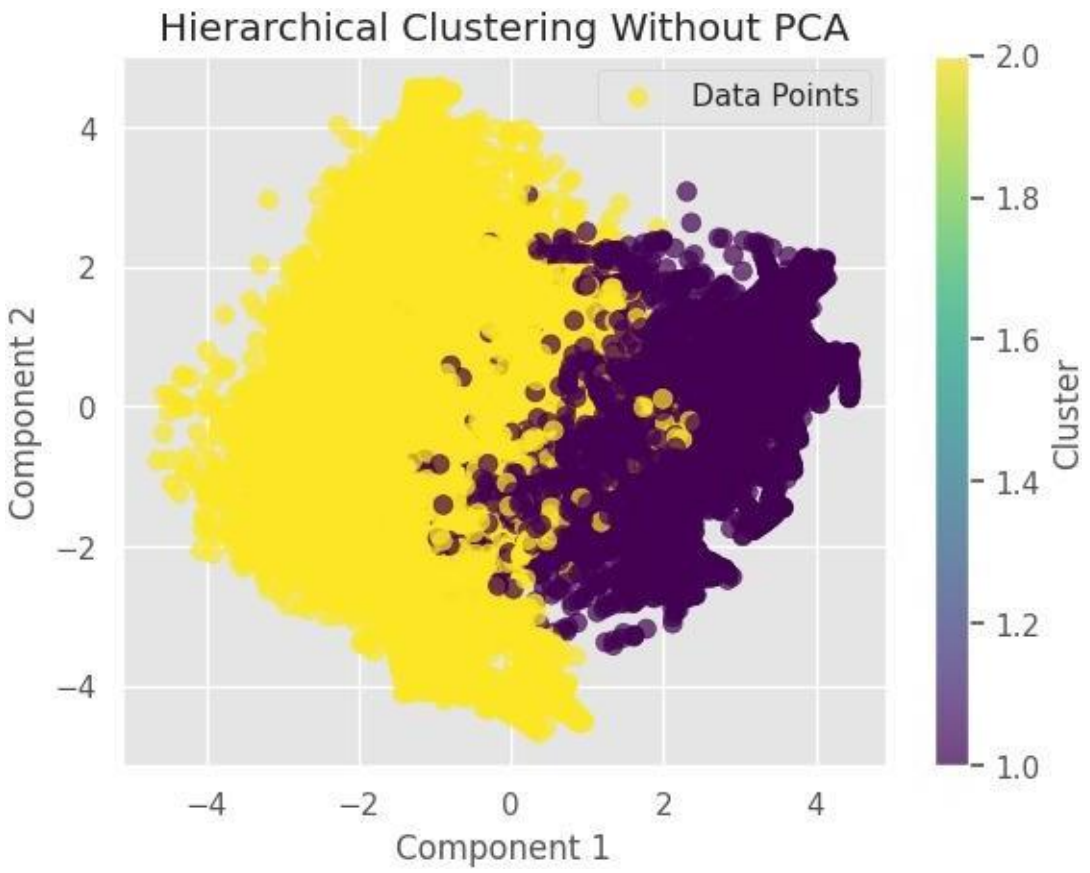
13.3.1.2 Hierarchical Clustering*:*

Dendrogram:



Ward Linkage:

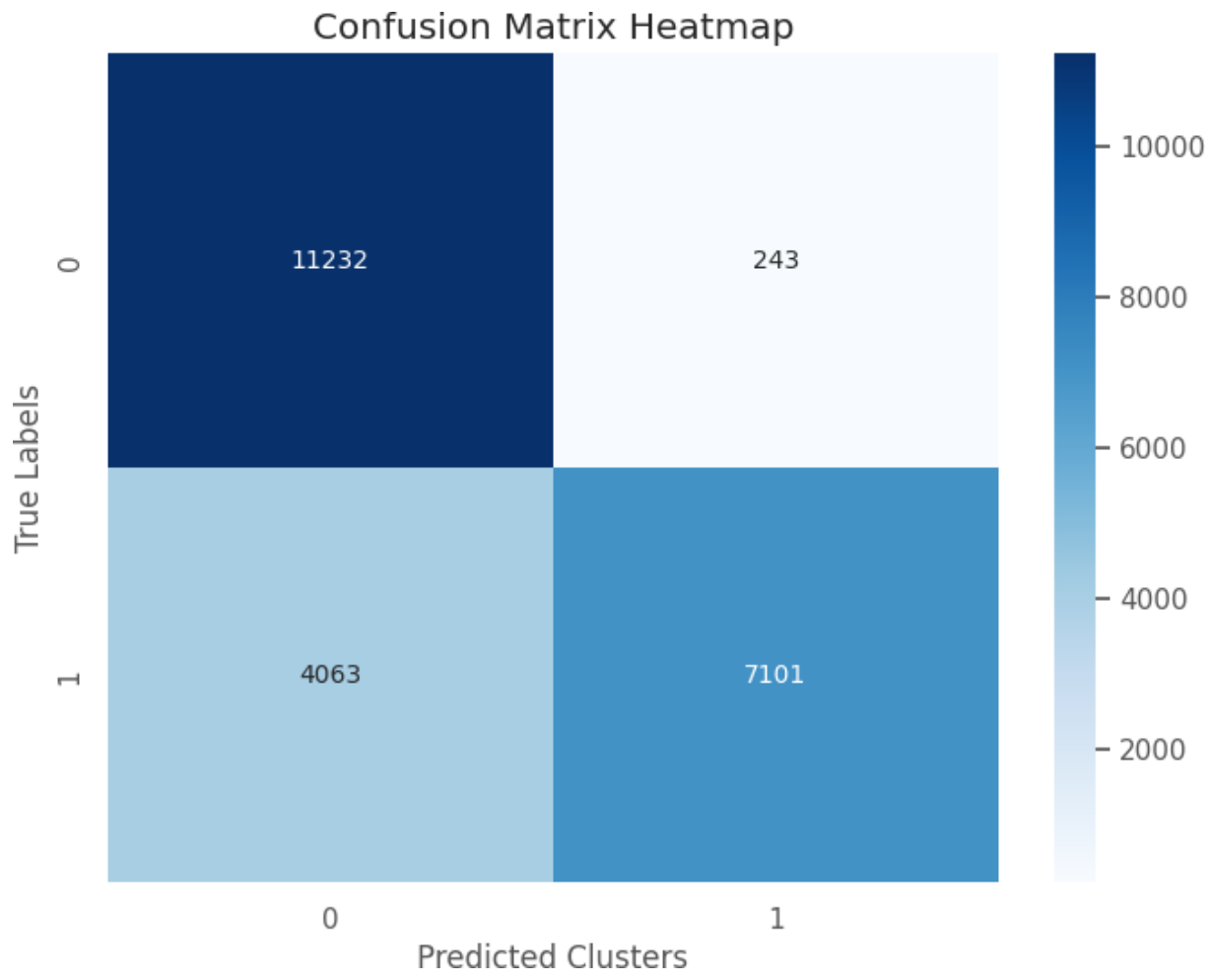
Visualization:



Evaluation Metrics:

Cluster Purity: 0.809797252528822

Confusion matrix:

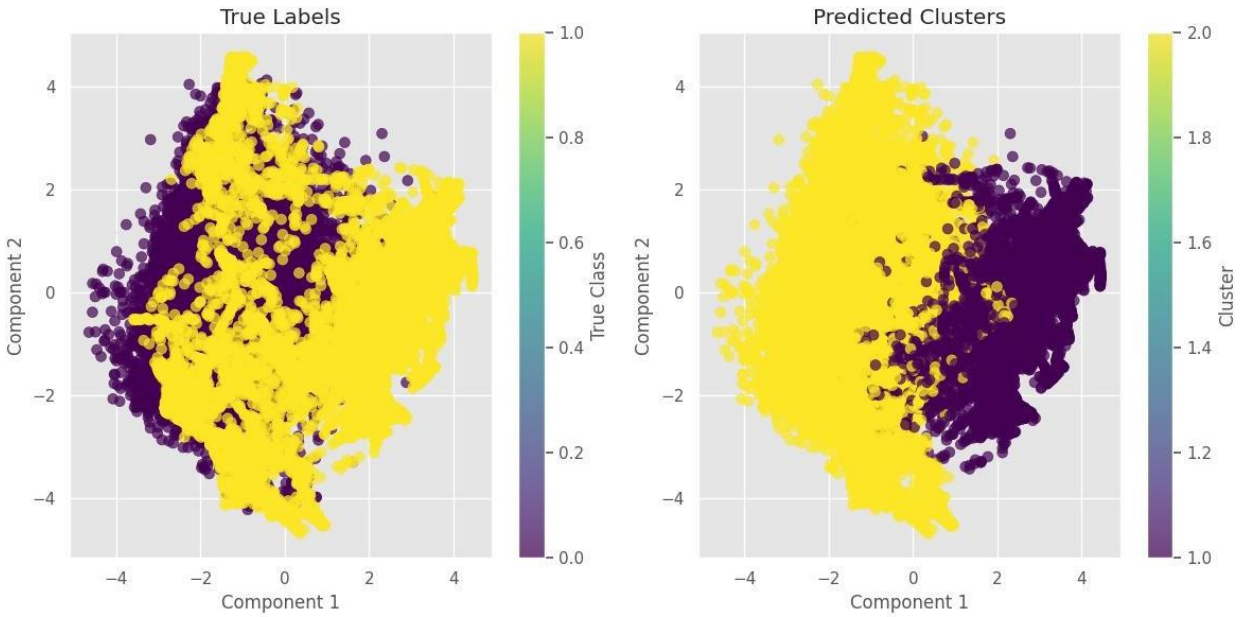


Other Metrics:

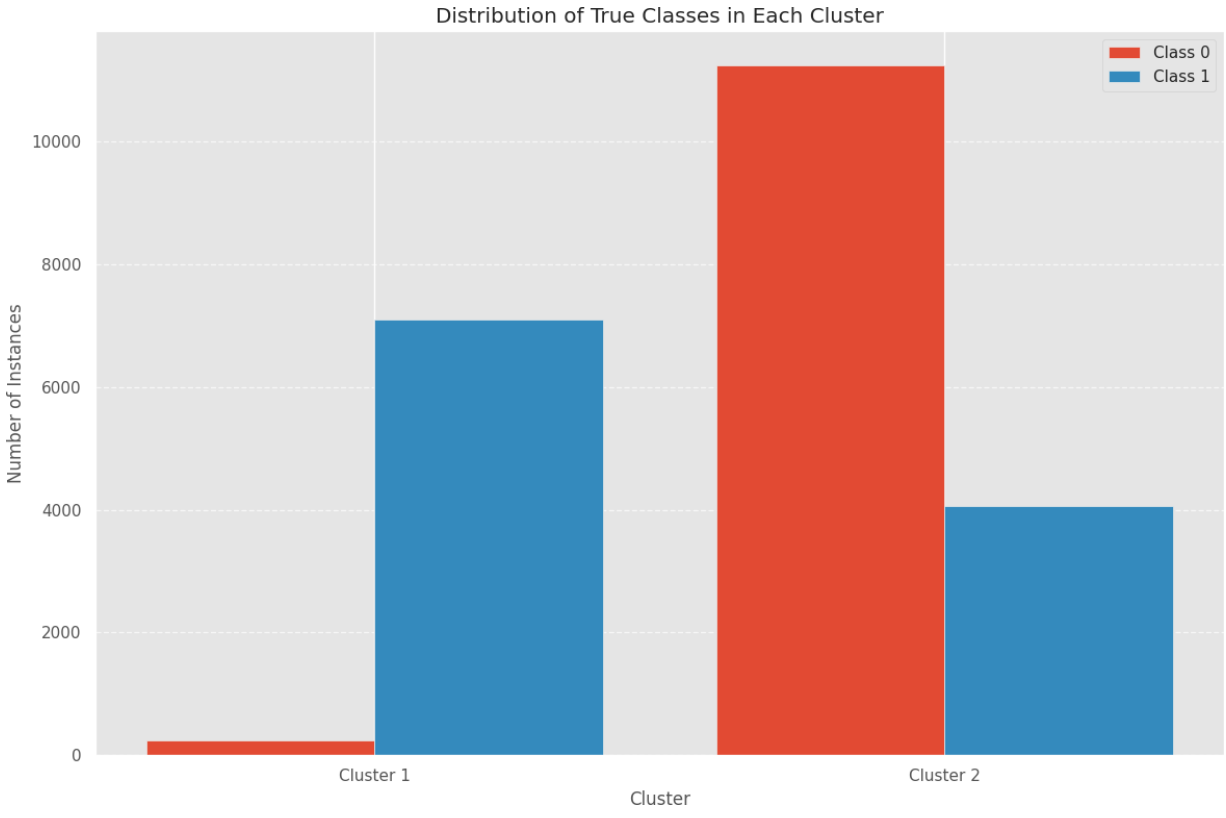
Normalized Mutual Information (NMI): 0.3851610448140514

silhouette_score: 0.2802339397708227

Comparison with true labels:



Class Distribution among clusters:

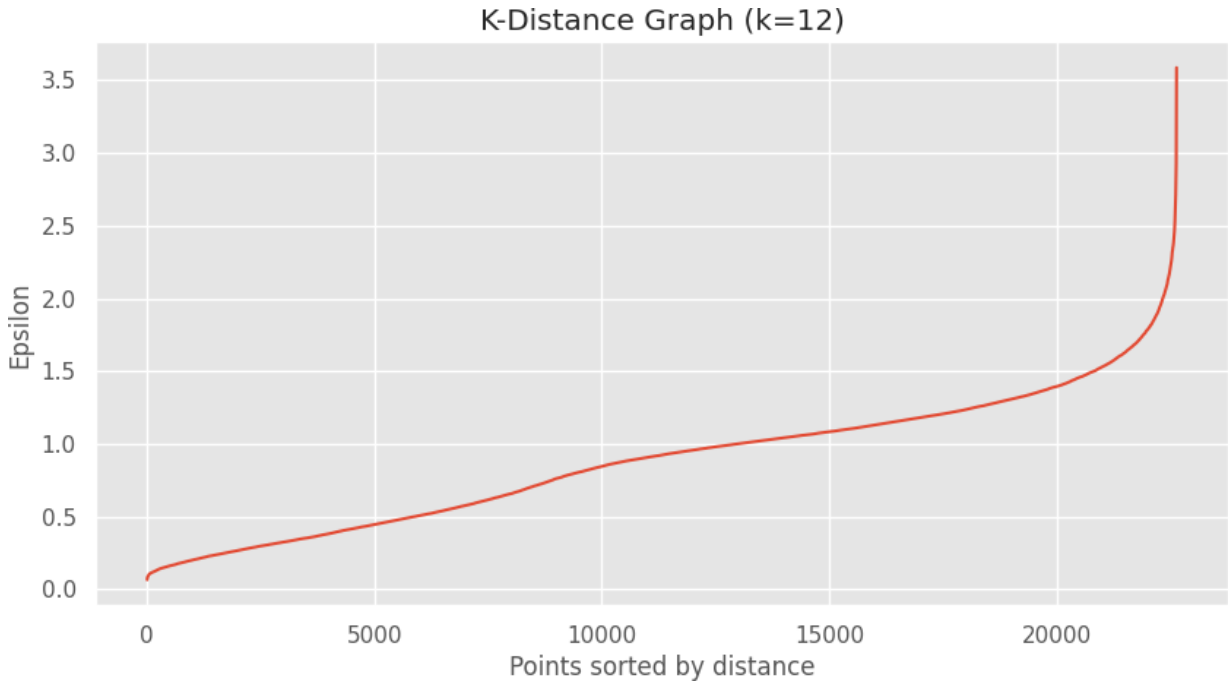


Cluster 1 has 6 datapoints with label 0 and 1050 datapoints with label 1.

Cluster 2 has 2259 datapoints with label 0 and 1081 datapoints with label 1.

13.3.1.3 DBScan Clustering:

Elbow method:



Elbow method shows eps=1.5 for min samples 12 and 24. We usually take value of k>= dimension of data. Dimension of our data is 12. It is recommended to keep k between Dimentionality + 1 and 2 * Dimentionality.

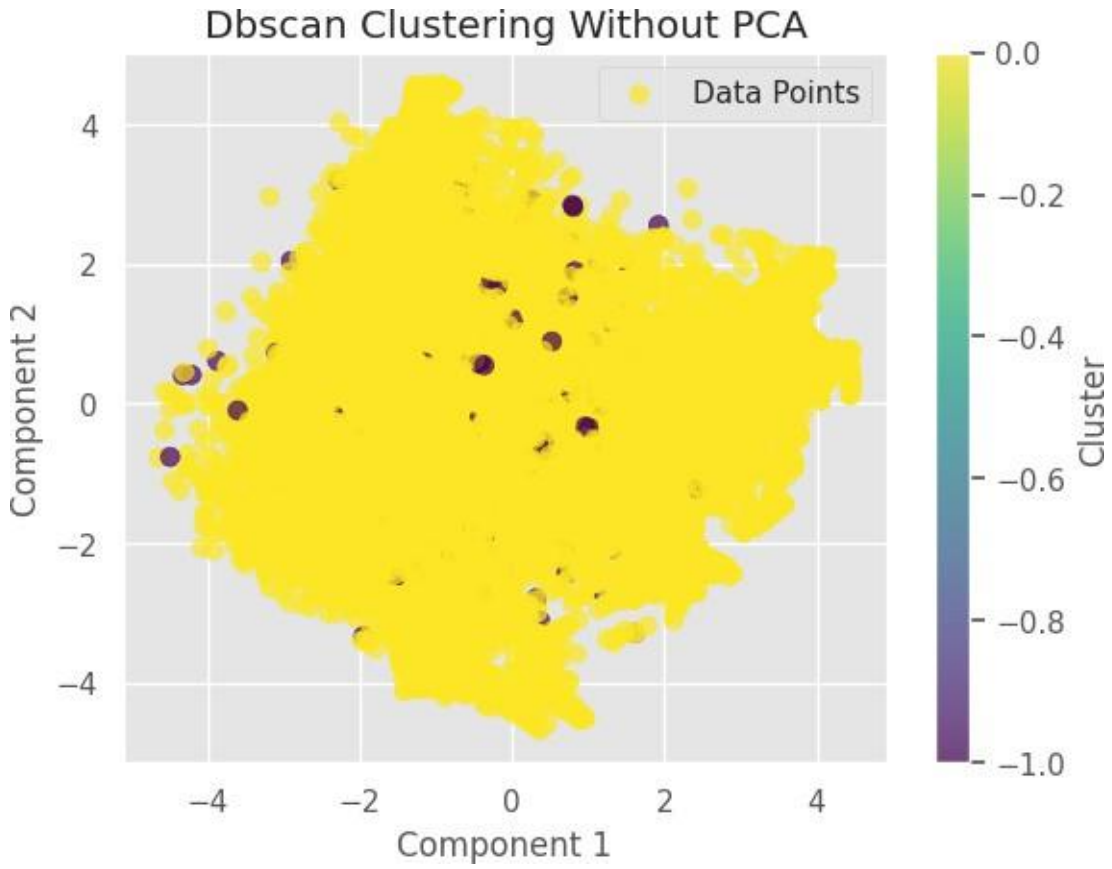
We tried experimenting DBScan for various values.

eps_range = [0.5, 1, 1.5, 2, 2.5, 3]

min_samples_range = [12, 15, 18, 21, 24]

None of them gave satisfactory results. Either it makes a lot of clusters, or it makes 1 cluster with some noise points. This experiment clearly shows that DBScan is not suitable for our dataset because of overlapping clusters.

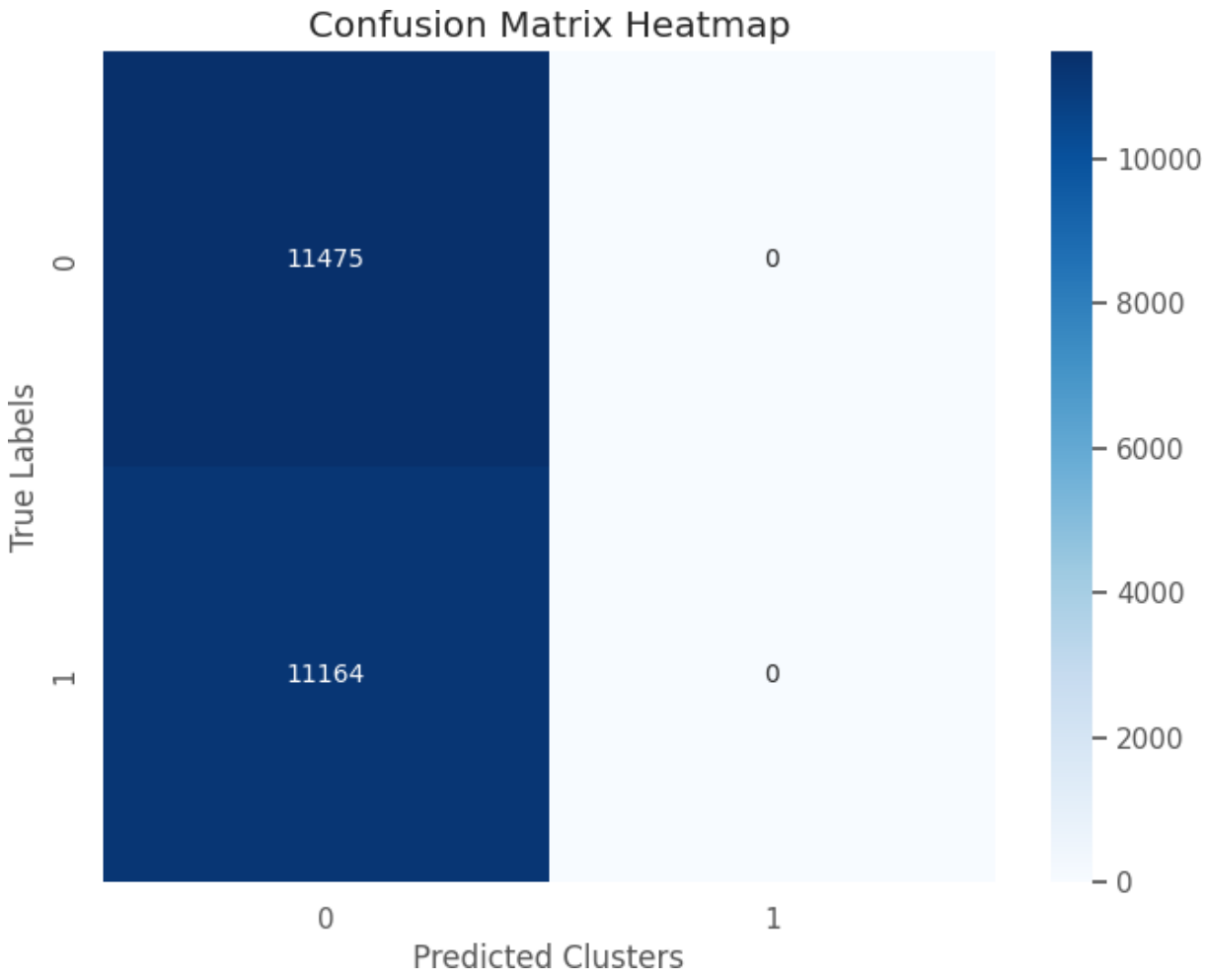
Visualization:



Evaluation Metrics:

Cluster Purity: 0.5068686779451389

Confusion matrix:

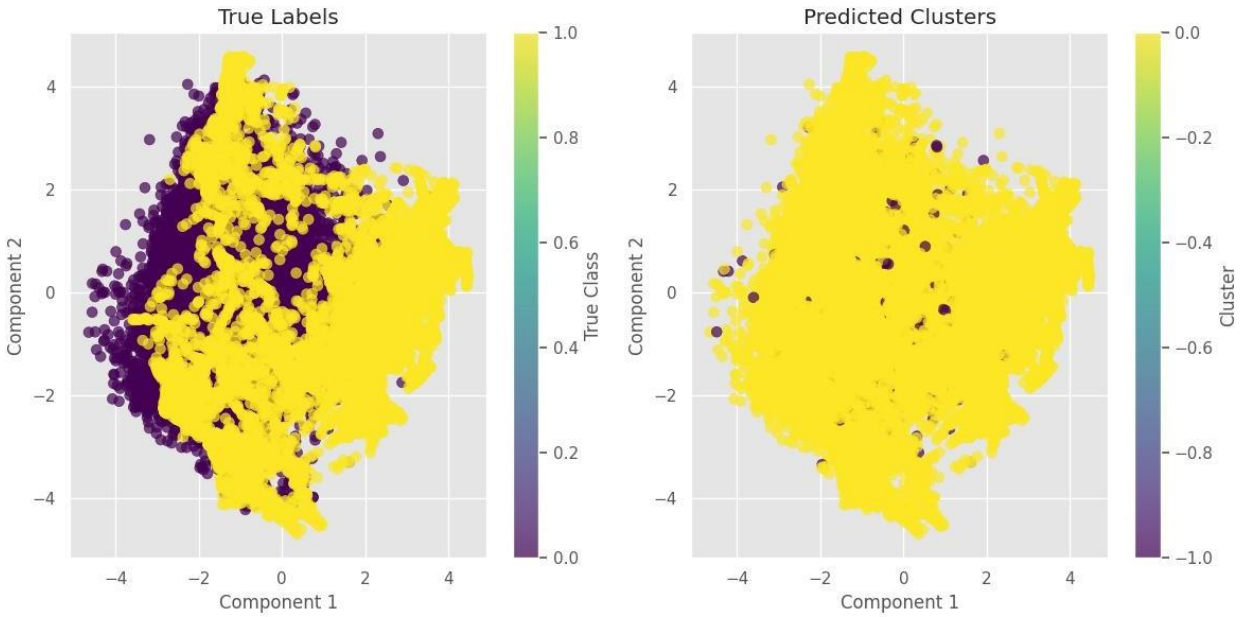


Other Metrics:

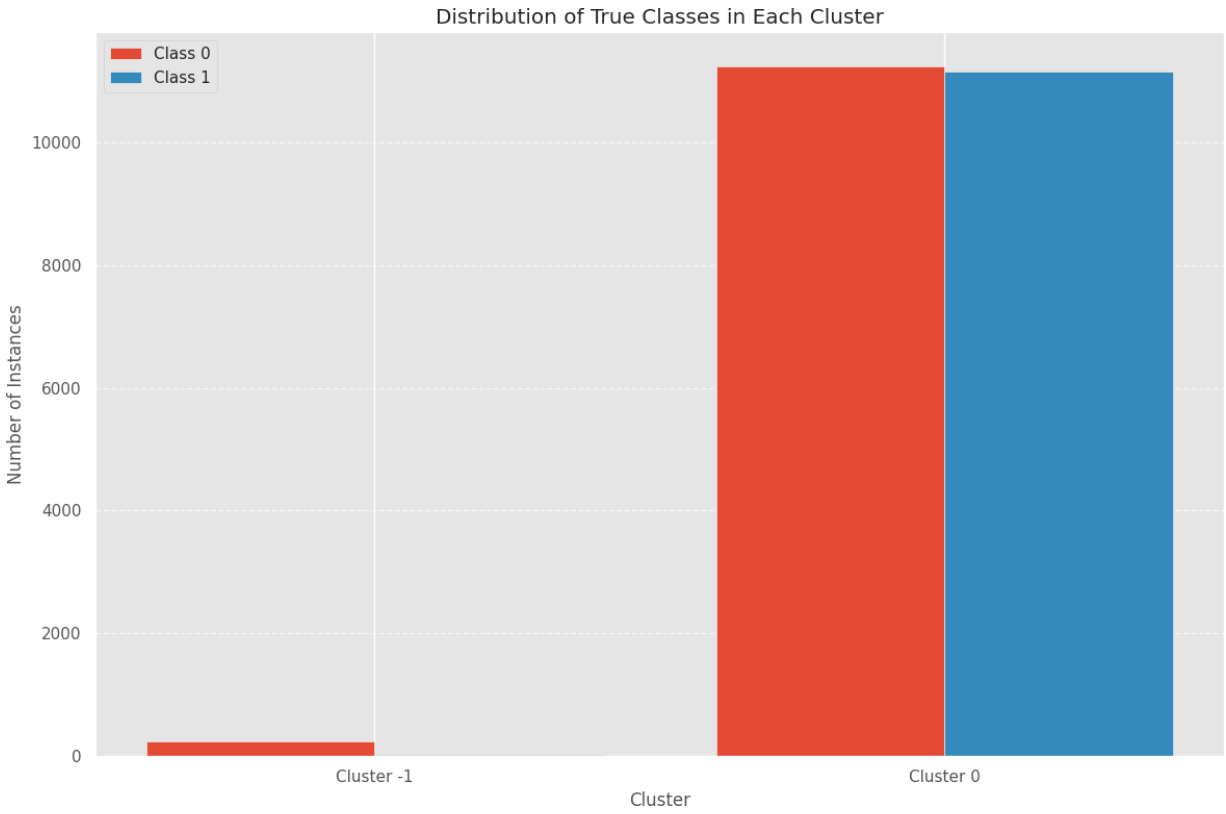
Normalized Mutual Information (NMI): 0.014199873250172239

silhouette_score: 0.08971229775965911

Comparison with true labels:



Class Distribution among clusters:

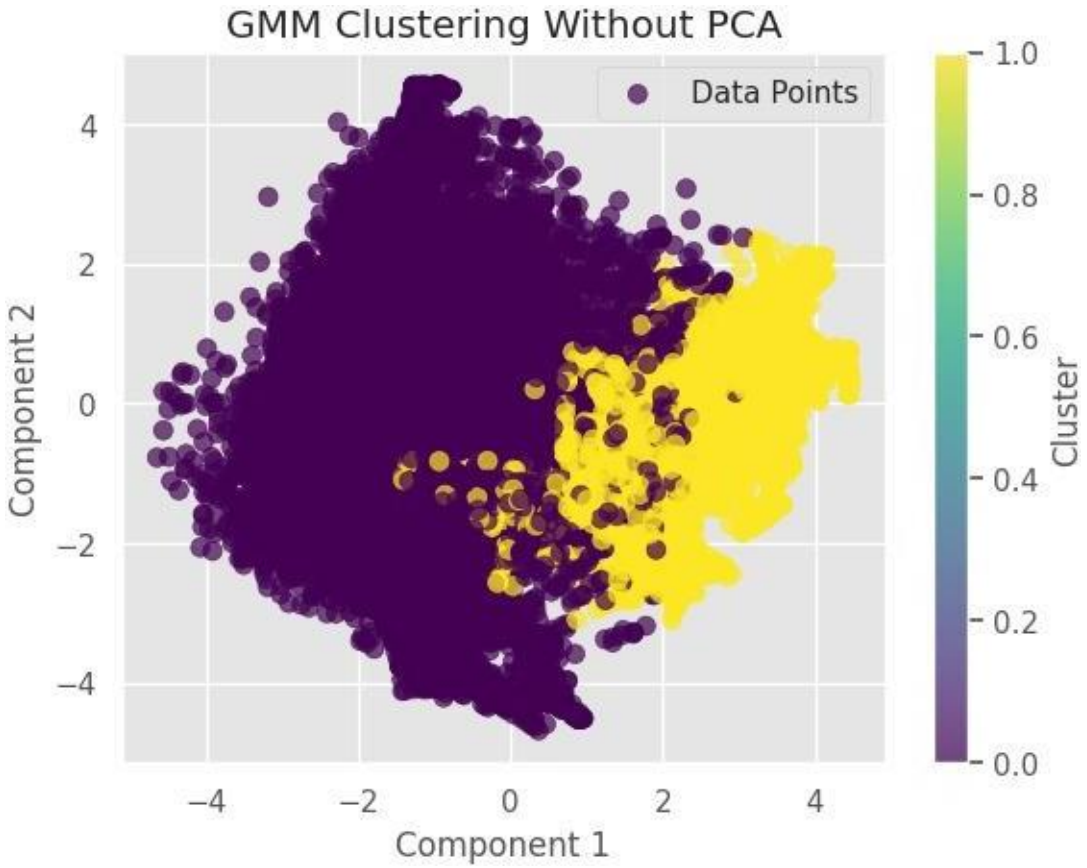


Cluster -1 has 246 datapoints with label 0 and 15 datapoints with label 1.

Cluster 0 has 11229 datapoints with label 0 and 11149 datapoints with label 1.

13.3.1.4 Gaussian Mixture Model Clustering*:*

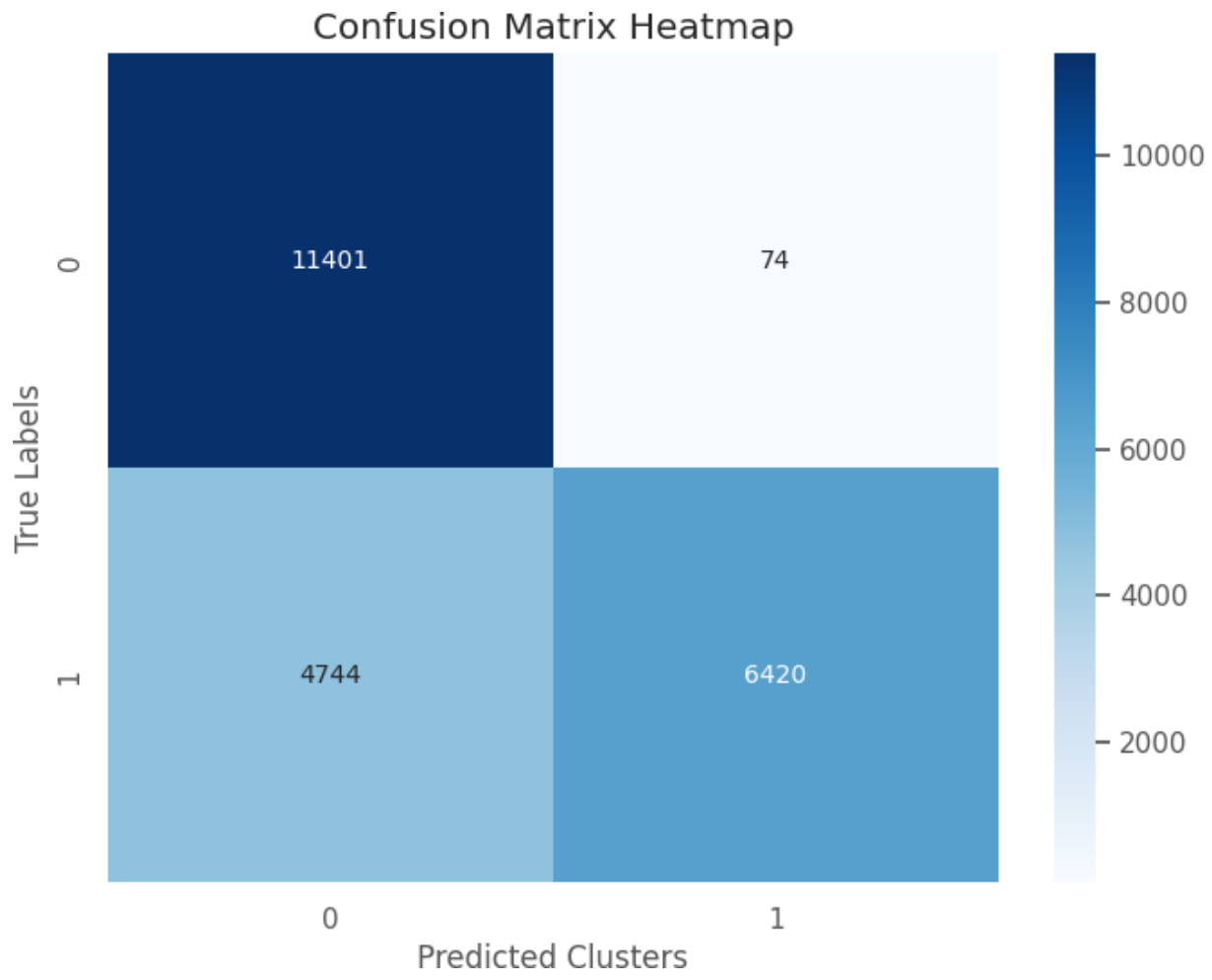
Visualization:



Evaluation Metrics:

Cluster Purity: 0.7871814126065639

Confusion matrix:

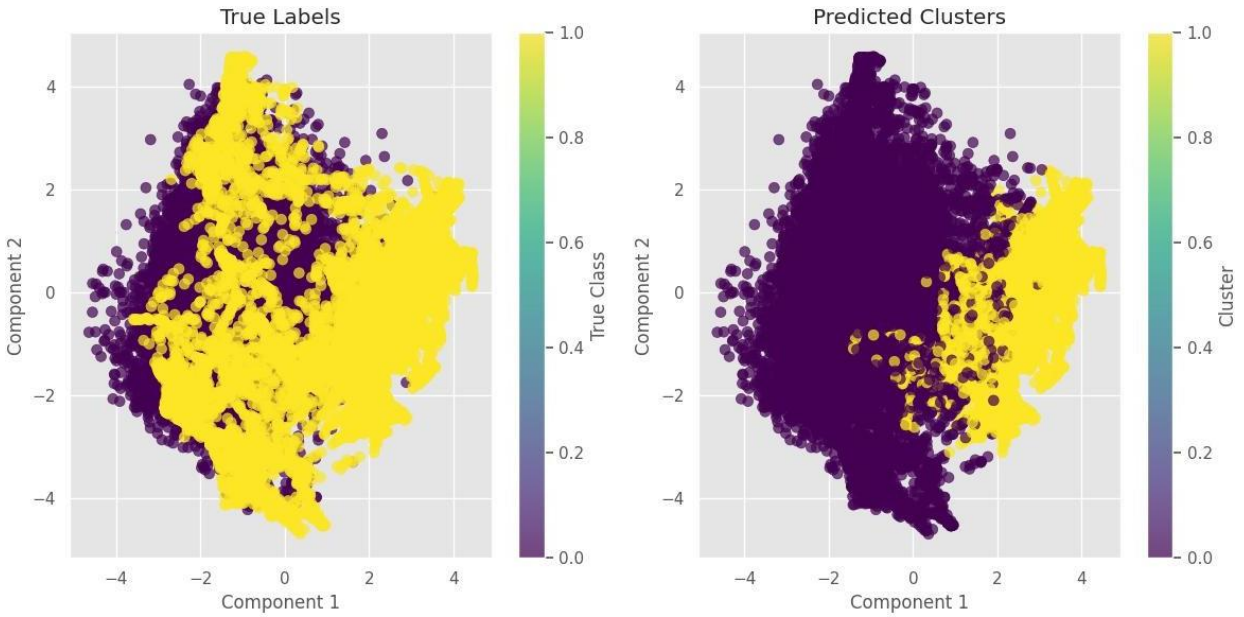


Other Metrics:

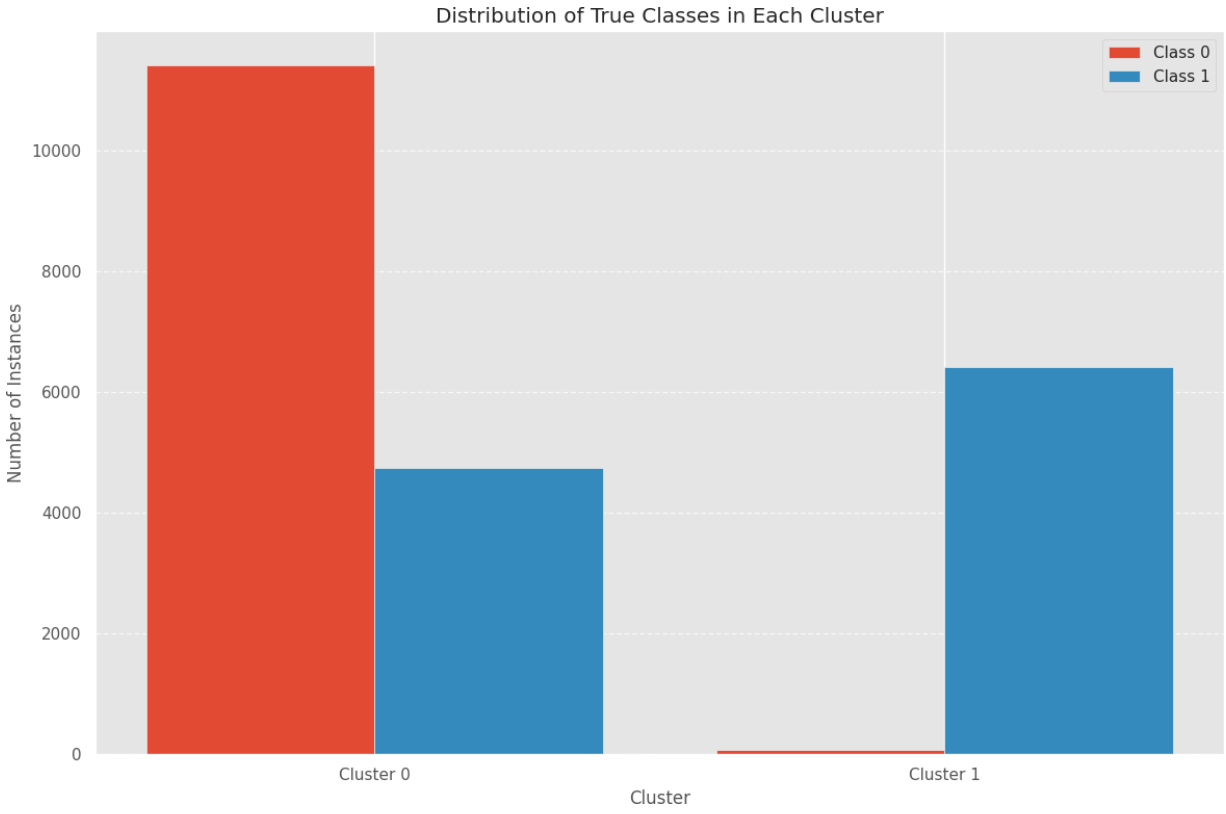
Normalized Mutual Information (NMI 0.3765633331268388

silhouette_score: 0.279942348517655

Comparison with true labels:



Class Distribution among clusters:



Cluster 0 has 11401 datapoints with label 0 and 4744 datapoints with label 1.

Cluster 1 has 74 datapoints with label 0 and 6420 datapoints with label 1.

13.3.1.5 Model Comparison:

SMOTE

SMOTE was used to address class imbalance in the dataset, ensuring more robust clustering by generating synthetic samples for the minority class. After applying SMOTE, various clustering methods were evaluated.

K-Means Clustering

Elbow Method

- **Optimal k:** Although the elbow method suggested k=3k = 3, we set k=2k = 2 as the dataset is labeled for binary classification.

Visualization

Clustering results were visualized, showing distinct clusters for the majority and minority classes.

Evaluation Metrics

- **Cluster Purity:** 0.8014
- **Normalized Mutual Information (NMI):** 0.3535
- **Silhouette Score:** 0.2928

Class Distribution Among Clusters

- Cluster 0: Predominantly label 0 (11,093) with fewer label 1 samples (4,114).
- Cluster 1: Predominantly label 1 (7,050) with fewer label 0 samples (382).

13.3.1.2 Hierarchical Clustering

Dendrogram

- Dendrogram suggested the dataset has moderate cluster separation.

Ward Linkage Visualization

Visualization confirmed the existence of two primary clusters.

Evaluation Metrics

- **Cluster Purity:** 0.8098
- **Normalized Mutual Information (NMI):** 0.3852
- **Silhouette Score:** 0.2802

Class Distribution Among Clusters

- Cluster 1: Predominantly label 1 (1,050).
- Cluster 2: Predominantly label 0 (2,259).

13.3.1.3 DBSCAN Clustering

Elbow Method

- Optimal parameters: eps=1.5\text{eps} = 1.5, min_samples=12\text{min_samples} = 12.
- Experimentation across multiple parameter combinations did not yield satisfactory results due to overlapping clusters.

Observations

- DBSCAN struggled with overlapping clusters, making either many clusters or one large cluster with noise.

Evaluation Metrics

- **Cluster Purity:** 0.5069
- **Normalized Mutual Information (NMI):** 0.0142
- **Silhouette Score:** 0.0897

Class Distribution Among Clusters

- Cluster -1: Mostly noise.
- Cluster 0: Mixed labels, showing poor separation.

13.3.1.4 Gaussian Mixture Model (GMM) Clustering

Visualization

- GMM clustering displayed a smoother cluster separation compared to K-Means, but still struggled with perfect segmentation.

Evaluation Metrics

- **Cluster Purity:** 0.7872
- **Normalized Mutual Information (NMI):** 0.3766
- **Silhouette Score:** 0.2799

Class Distribution Among Clusters

- Cluster 0: Predominantly label 0.
- Cluster 1: Predominantly label 1.

Summary of Model Performance

Metric/Model	K-Means	Hierarchical	DBSCAN	GMM
Cluster Purity	0.8014	0.8098	0.5069	0.7872
NMI	0.3535	0.3852	0.0142	0.3766
Silhouette Score	0.2928	0.2802	0.0897	0.2799

Conclusion

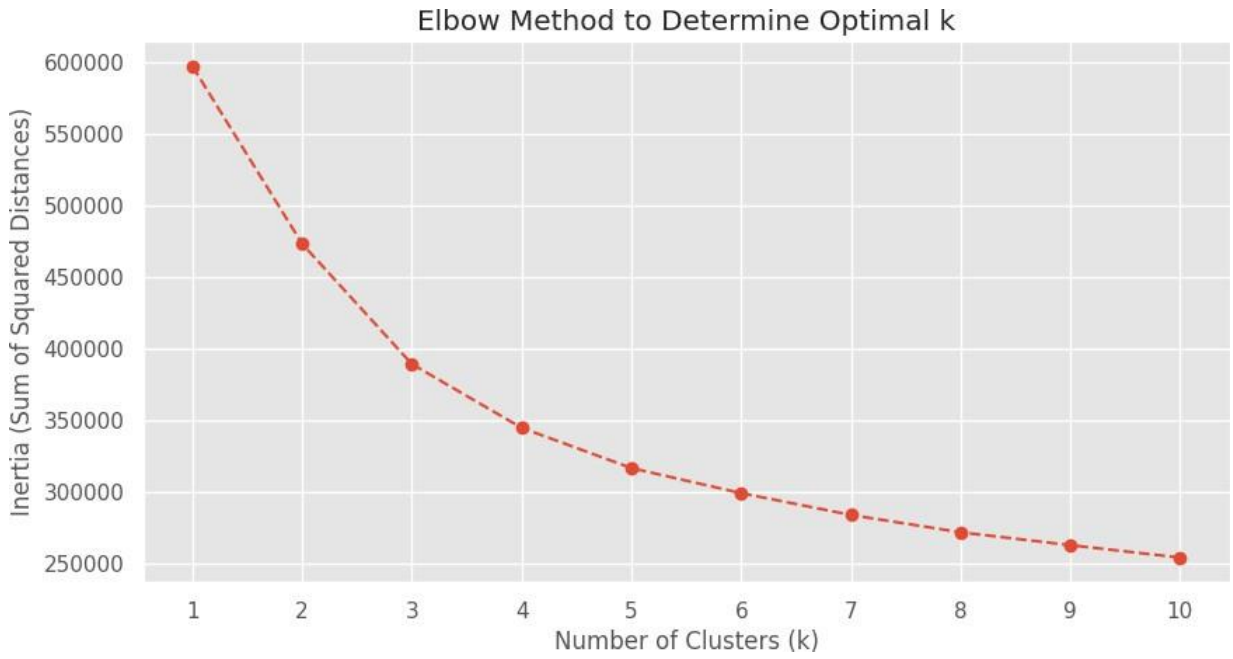
- **Best Performer: Hierarchical Clustering** achieved the highest cluster purity and NMI.
- **Worst Performer: DBSCAN** failed due to overlapping clusters.
- **GMM and K-Means:** Performed similarly, with GMM slightly better in handling data complexity.
- **Recommendation:** Use Hierarchical Clustering for this dataset as it balances performance and interpretability effectively.

13.4 All years

13.4.1 SMOTE:

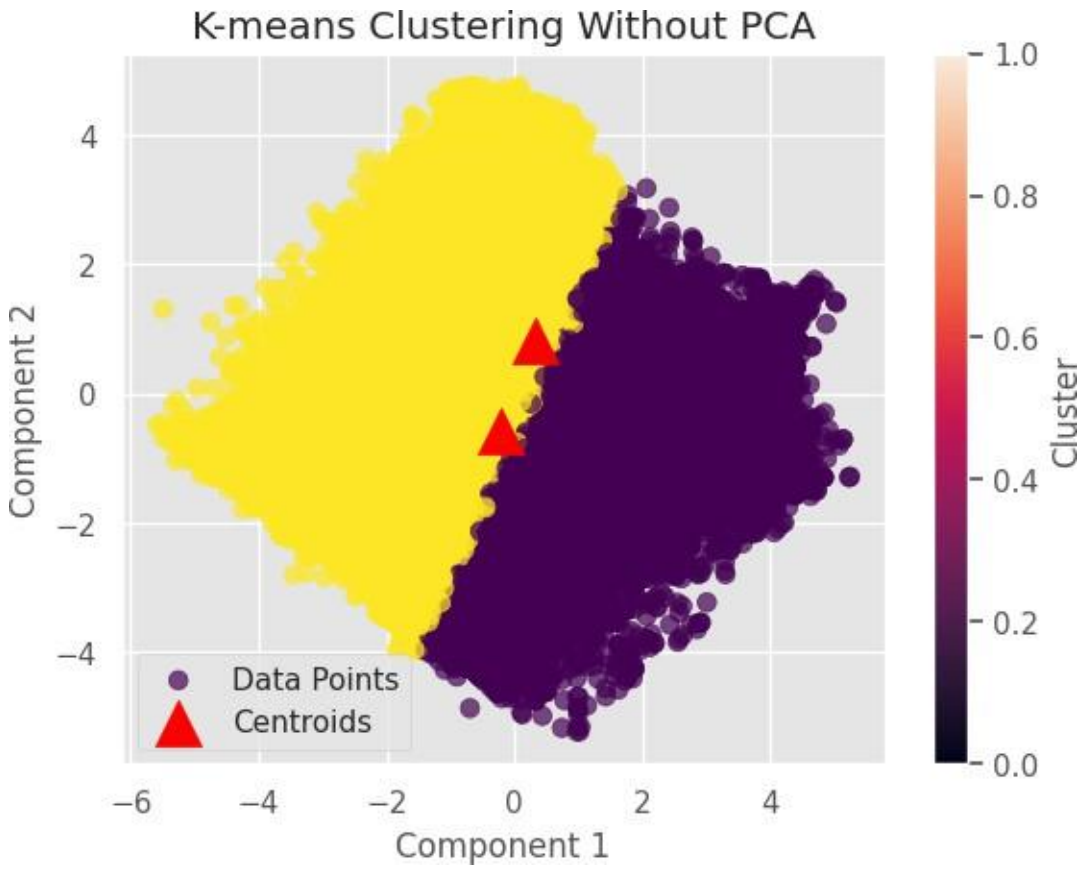
13.4.1.1 K Means Clustering:

Elbow method:



Elbow method shows k=3 but since our dataset is labelled and we are dealing with 2 classes, k = 2.

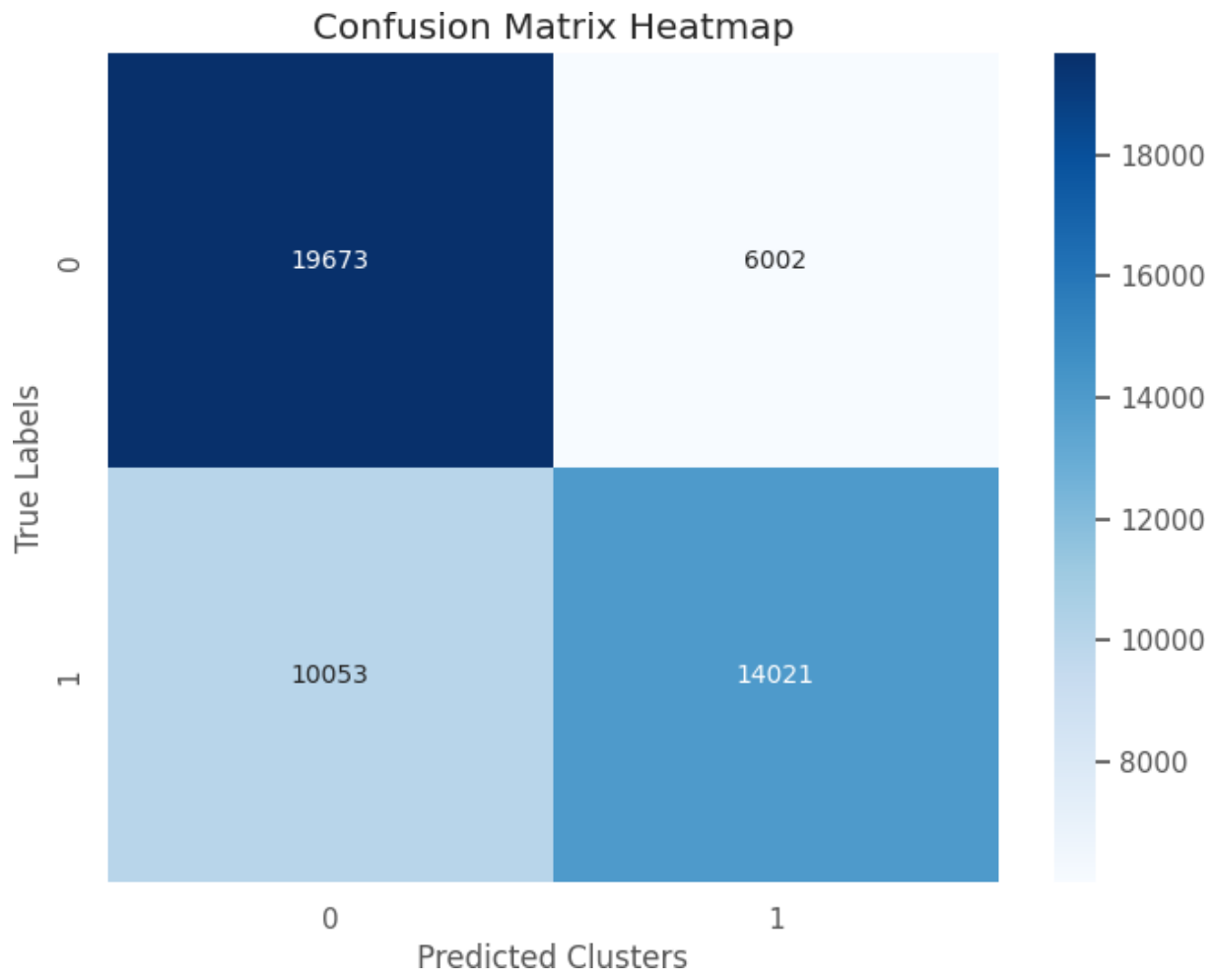
Visualization:



Evaluation Metrics:

Cluster Purity: 0.6772799453255341

Confusion matrix:

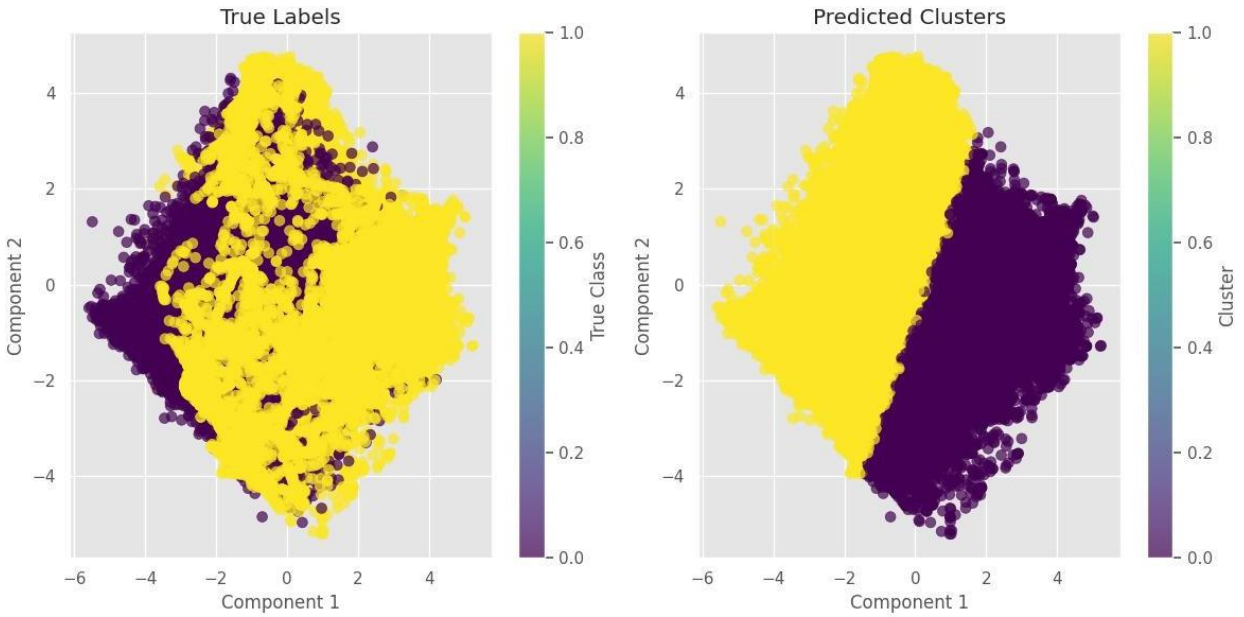


Other Metrics:

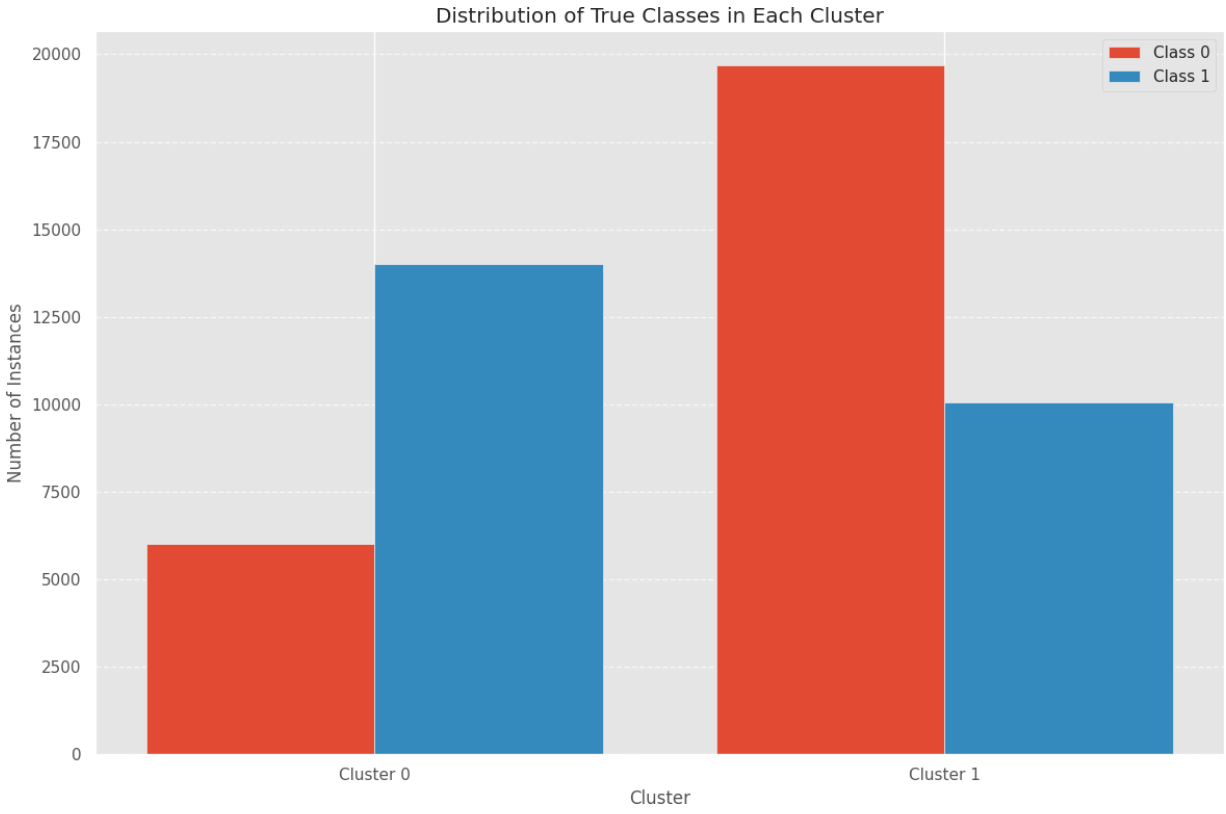
Normalized Mutual Information (NMI): 0.09445517066773207

silhouette_score: 0.202638435430059

Comparison with true labels:



Class Distribution among clusters:

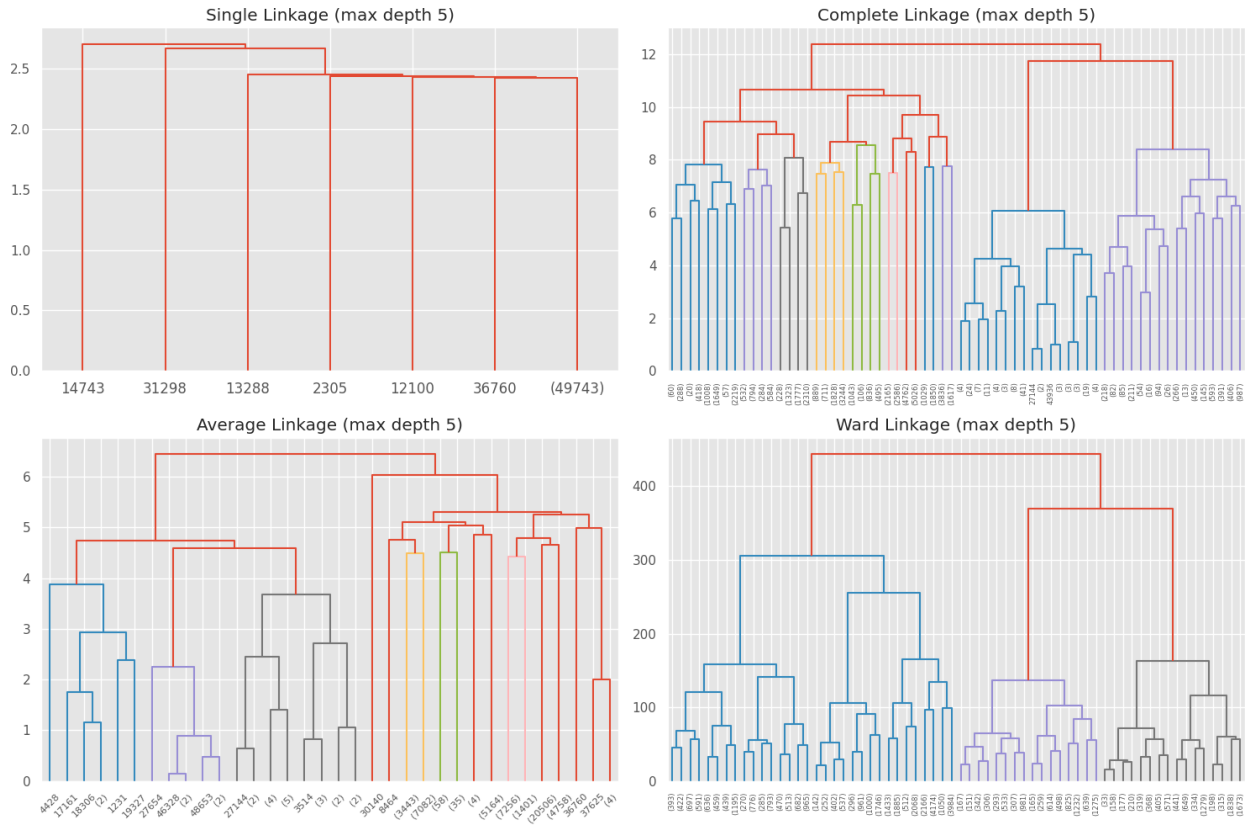


Cluster 0 has 6002 datapoints with label 0 and 14021 datapoints with label 1.

Cluster 1 has 19673 datapoints with label 0 and 10053 datapoints with label 1.

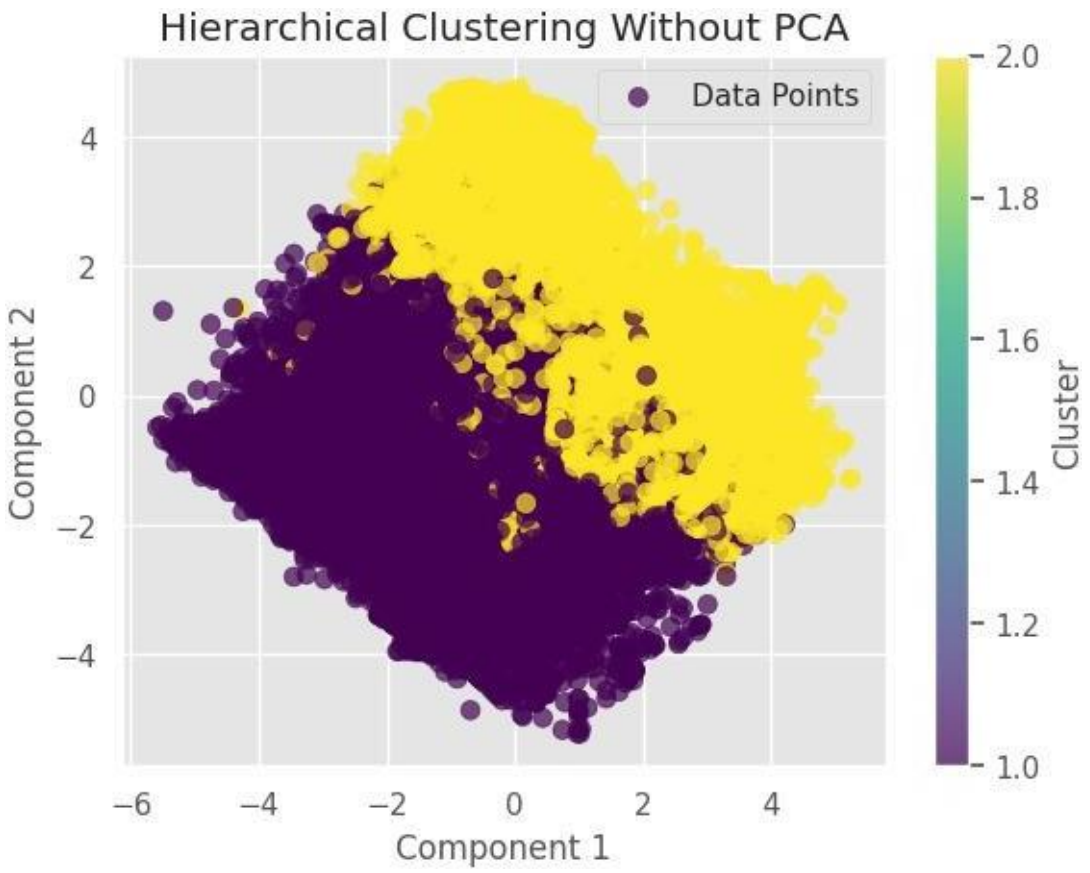
13.4.1.2 Hierarchical Clustering*:*

Dendrogram:



Ward Linkage:

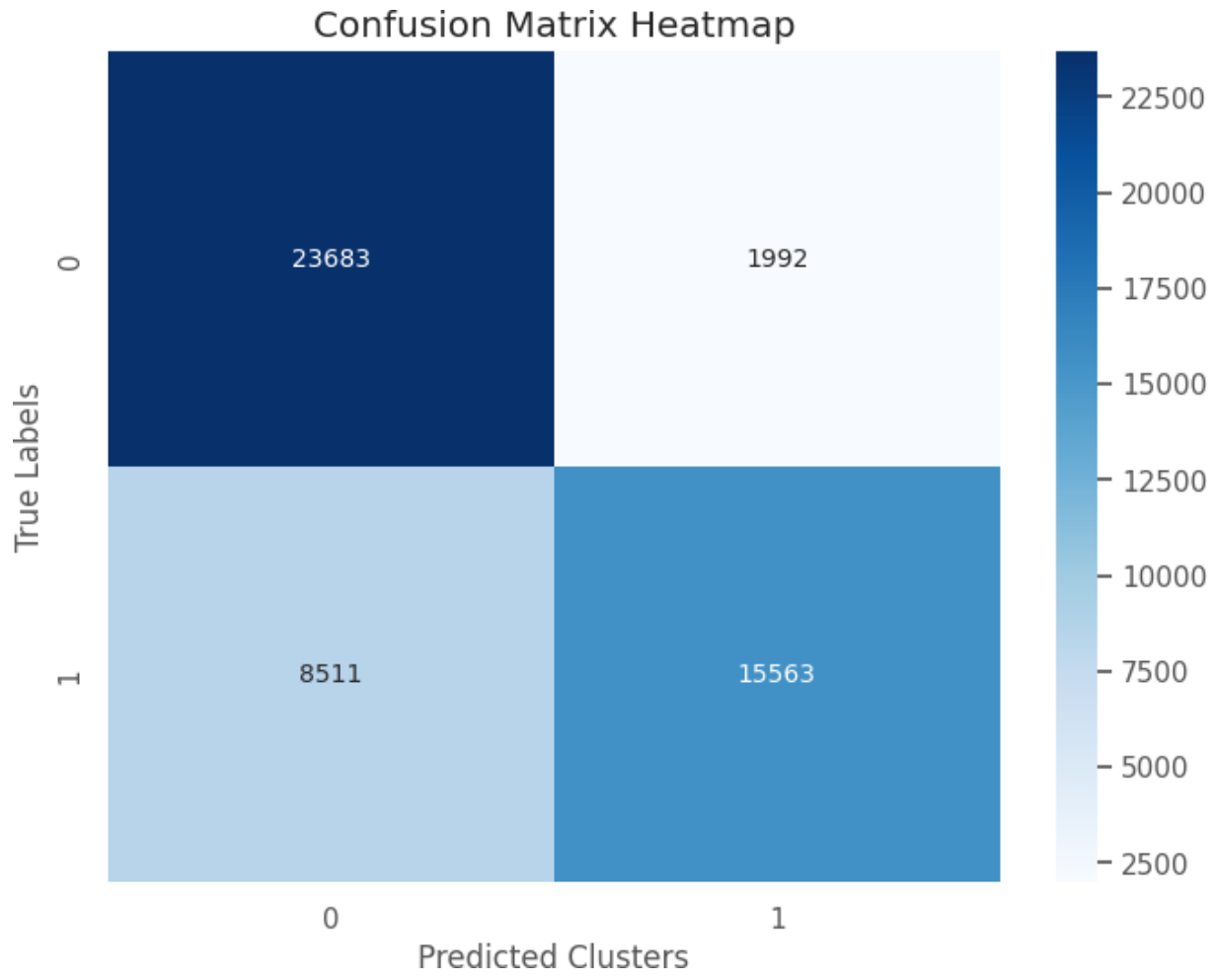
Visualization:



Evaluation Metrics:

Cluster Purity: 0.7888801784960502

Confusion matrix:

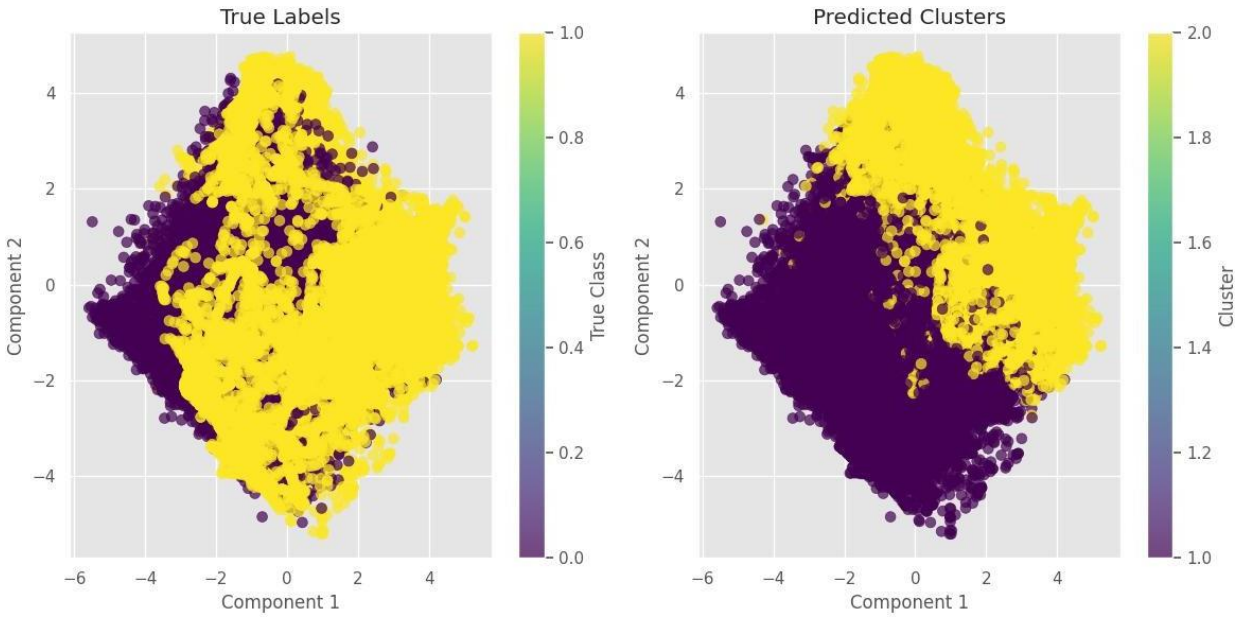


Other Metrics:

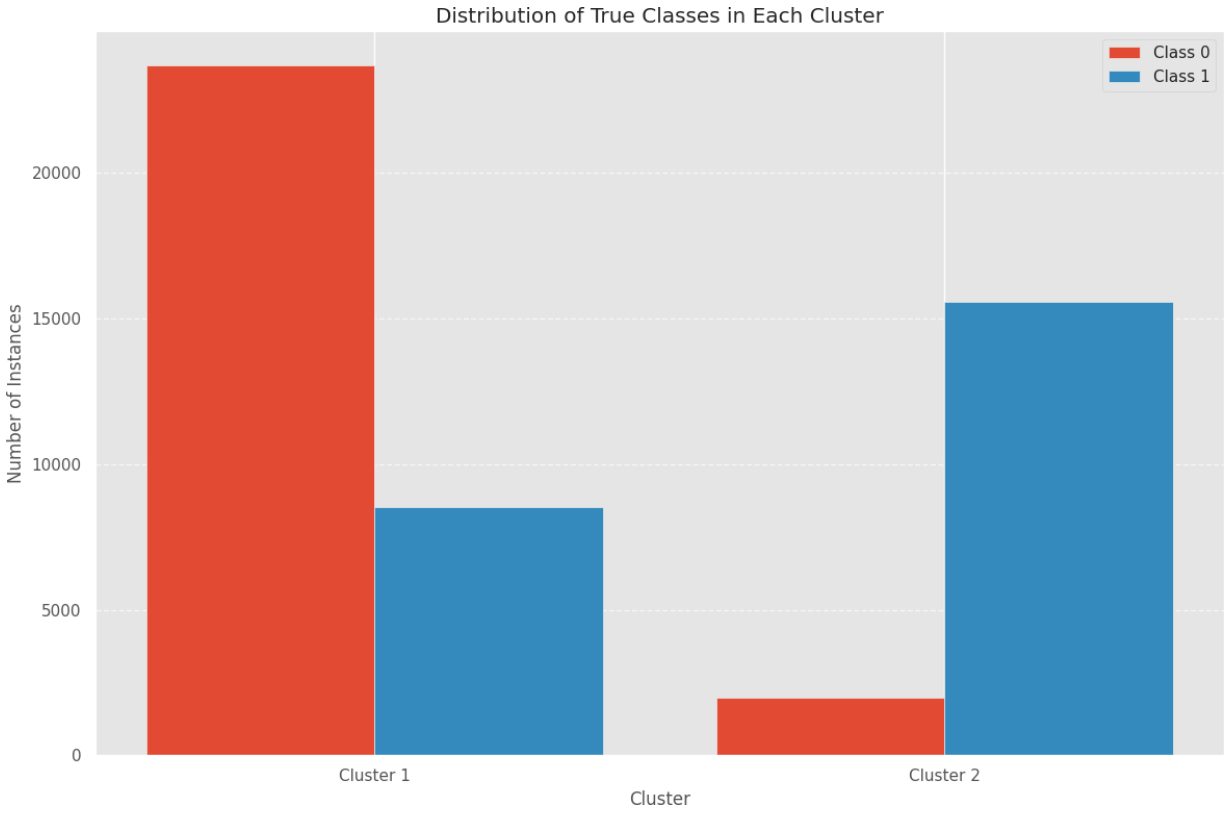
Normalized Mutual Information (NMI): 0.2892294272057773

silhouette_score: 0.1707953850648645

Comparison with true labels:



Class Distribution among clusters:

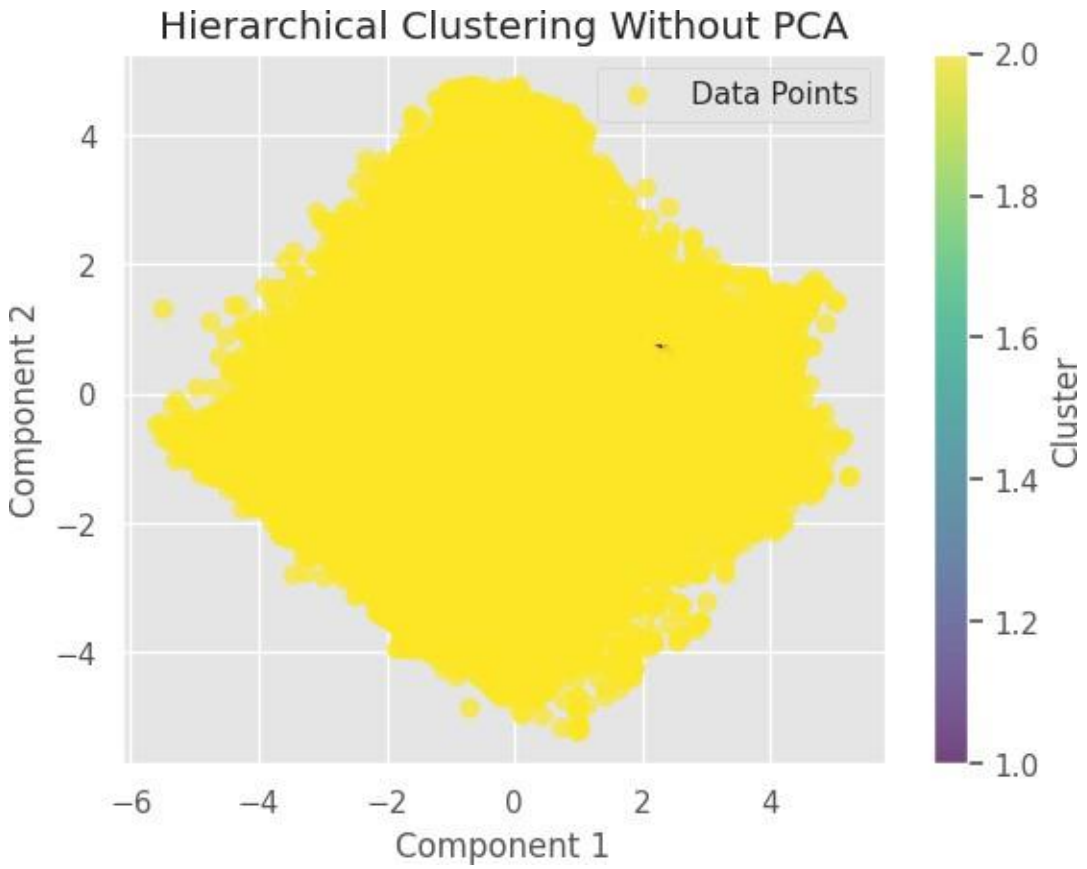


Cluster 1 has 23683 datapoints with label 0 and 8511 datapoints with label 1.

Cluster 2 has 1992 datapoints with label 0 and 15563 datapoints with label 1.

Centroid Linkage:

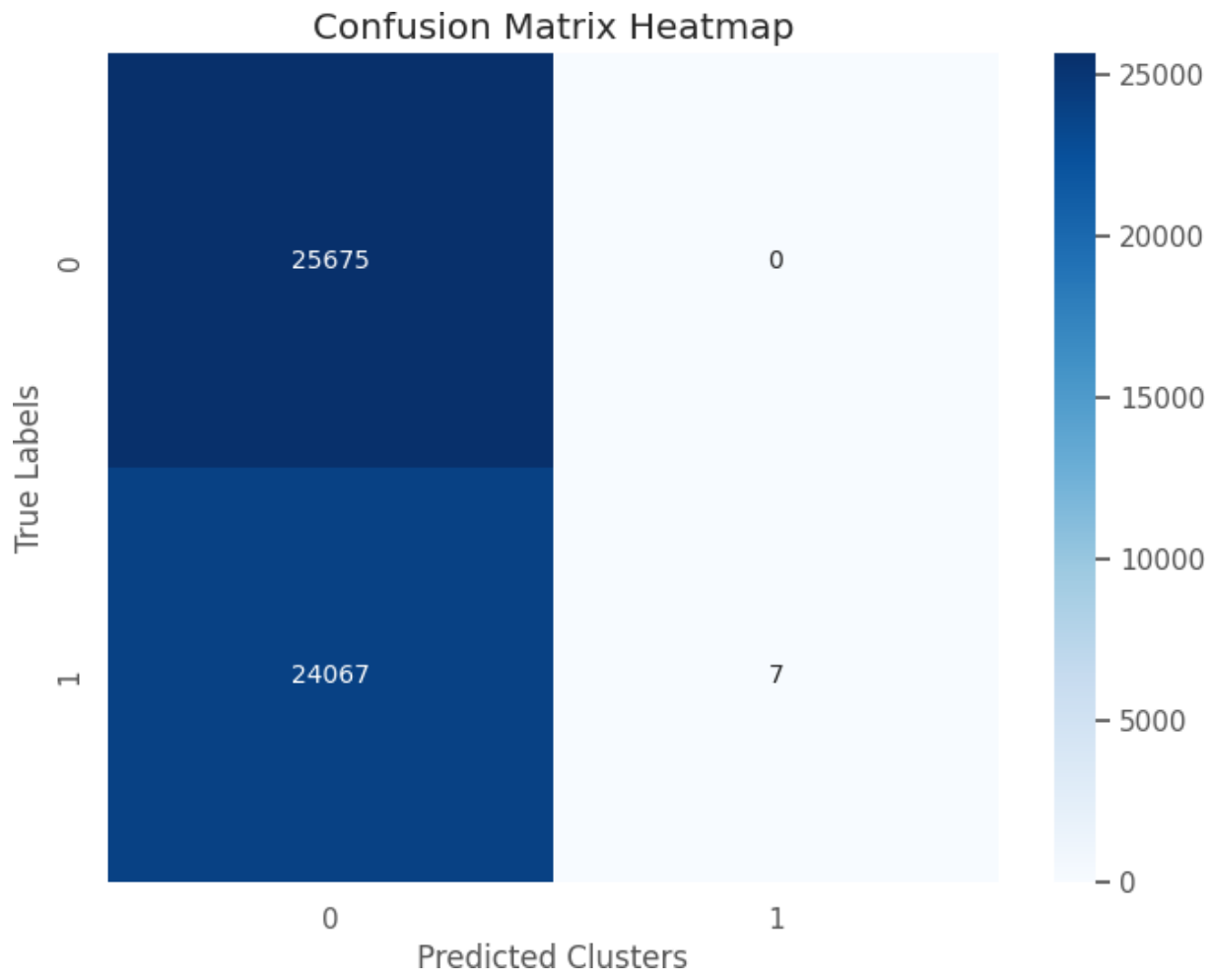
Visualization:



Evaluation Metrics:

Cluster Purity: 0.51623148203984

Confusion matrix:

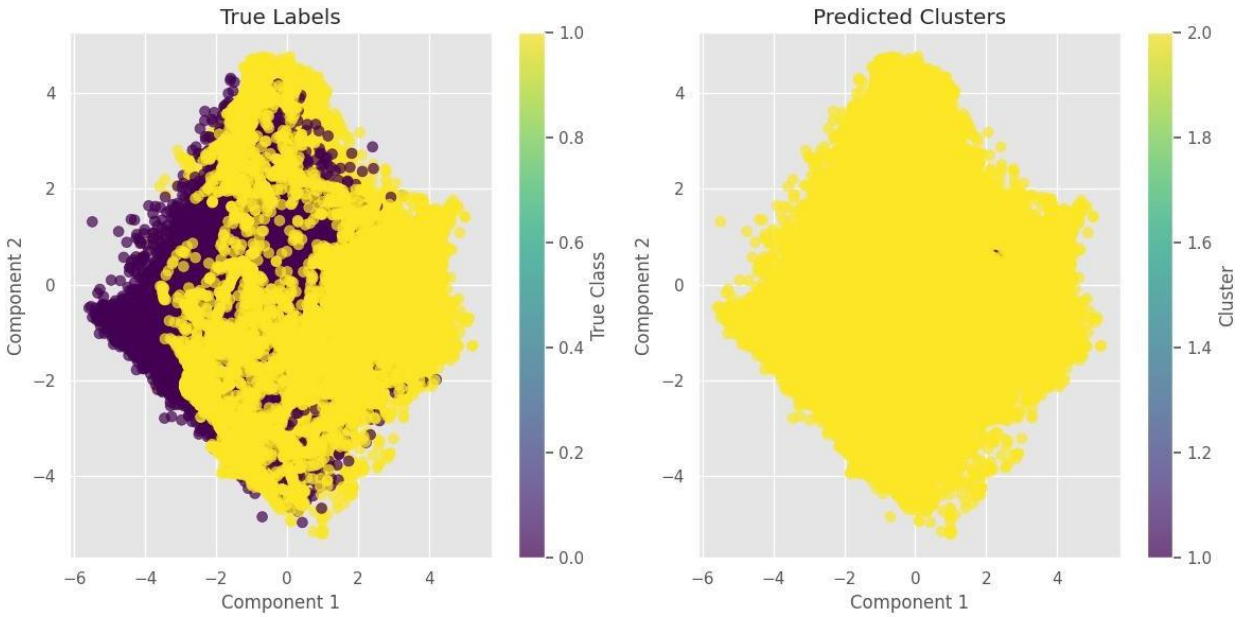


Other Metrics:

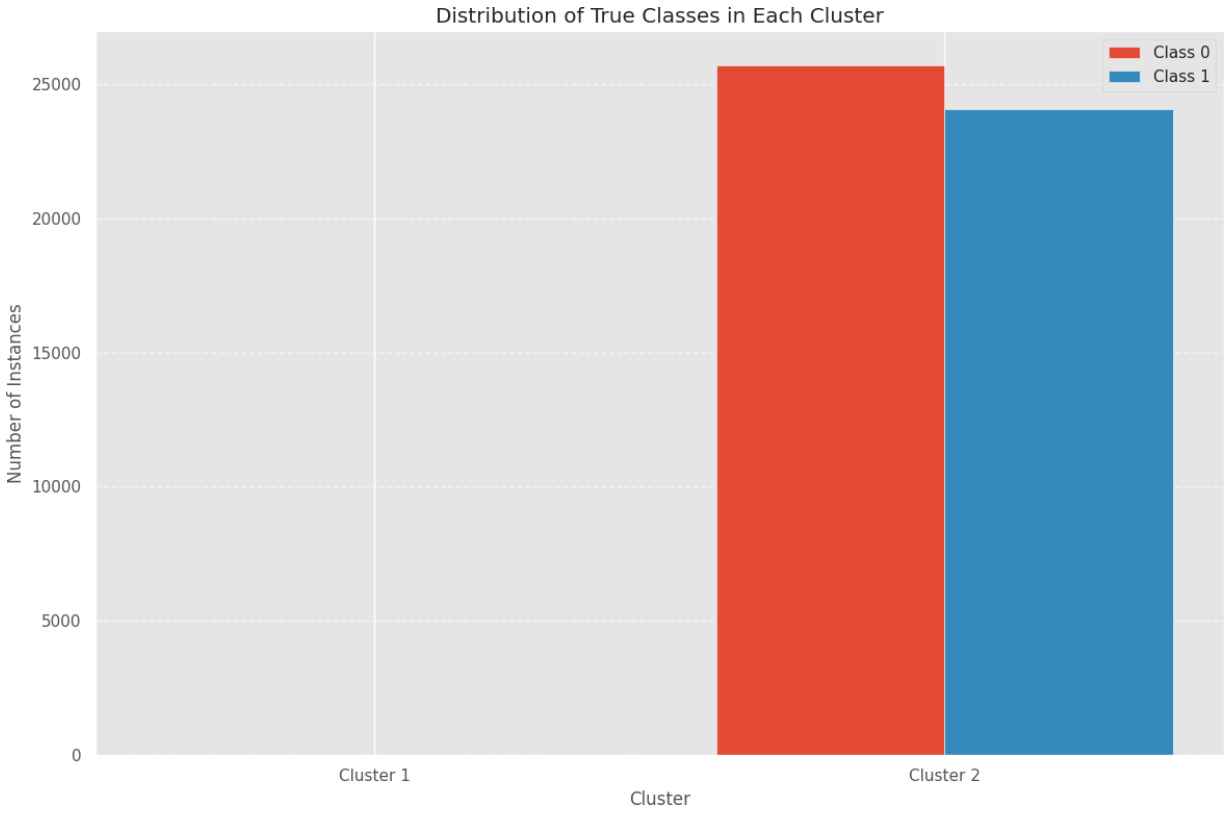
Normalized Mutual Information (NMI): 0.0002943537427640311

silhouette_score: 0.4064844805065774

Comparison with true labels:



Class Distribution among clusters:



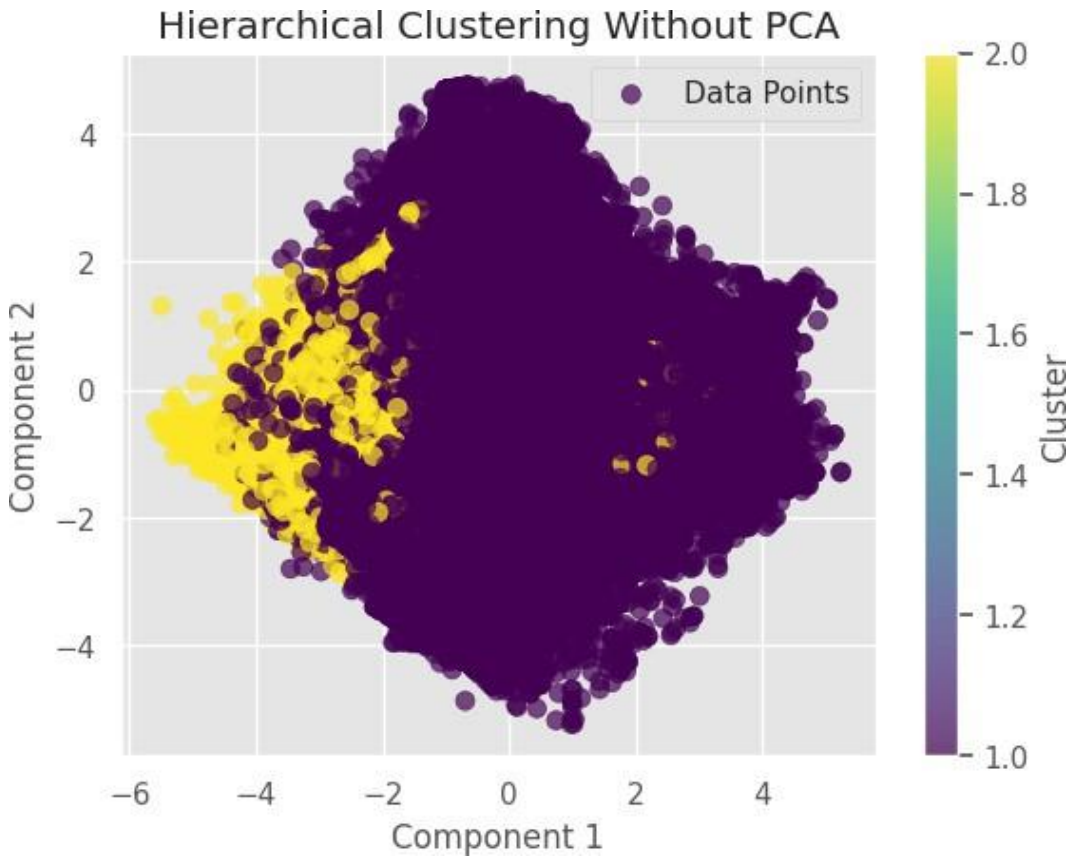
Cluster 1 has 0 datapoints with label 0 and 7 datapoints with label 1.

Cluster 2 has 25675 datapoints with label 0 and 24067 datapoints with label 1.

Centroid linkage gives horrible performance on our data.

Complete Linkage:

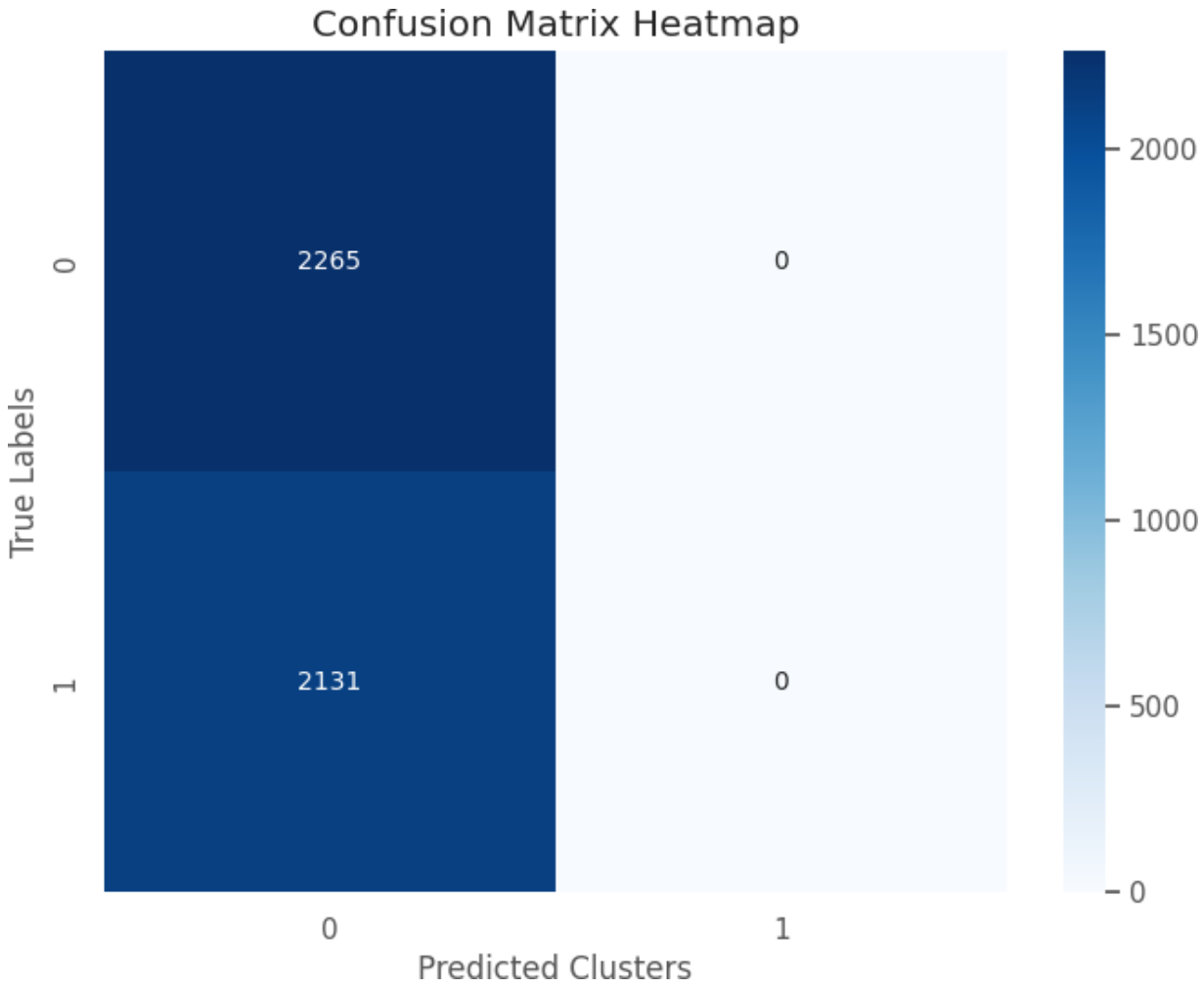
Visualization:



Evaluation Metrics:

Cluster Purity: 0.541860137892219

Confusion matrix:

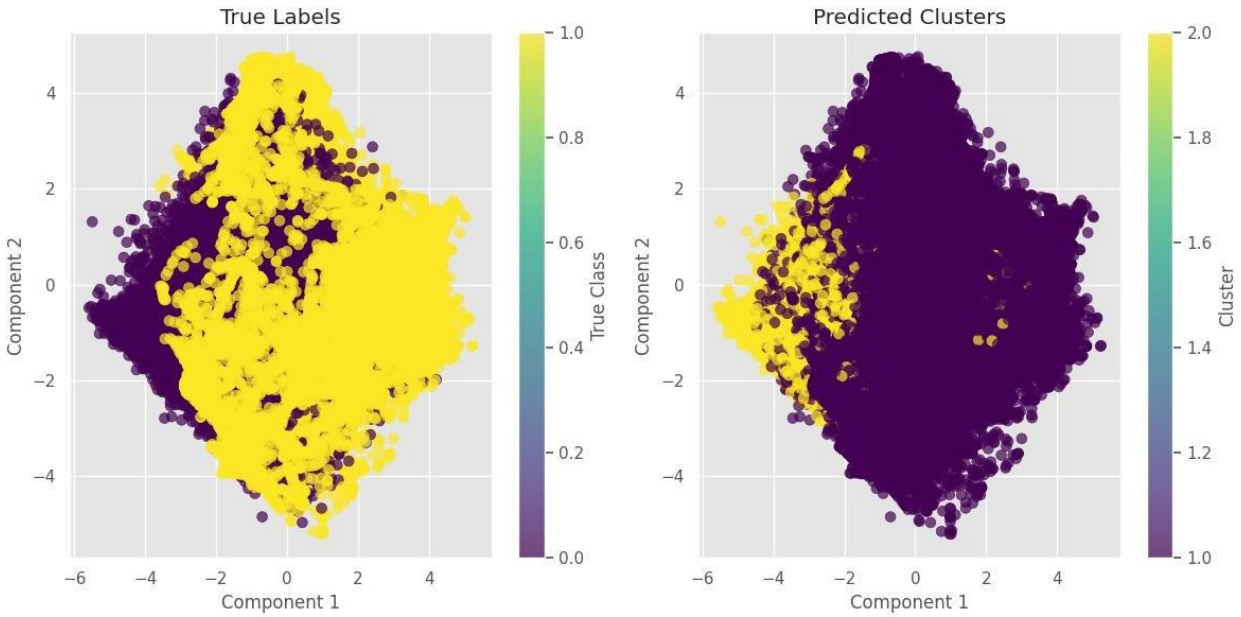


Other Metrics:

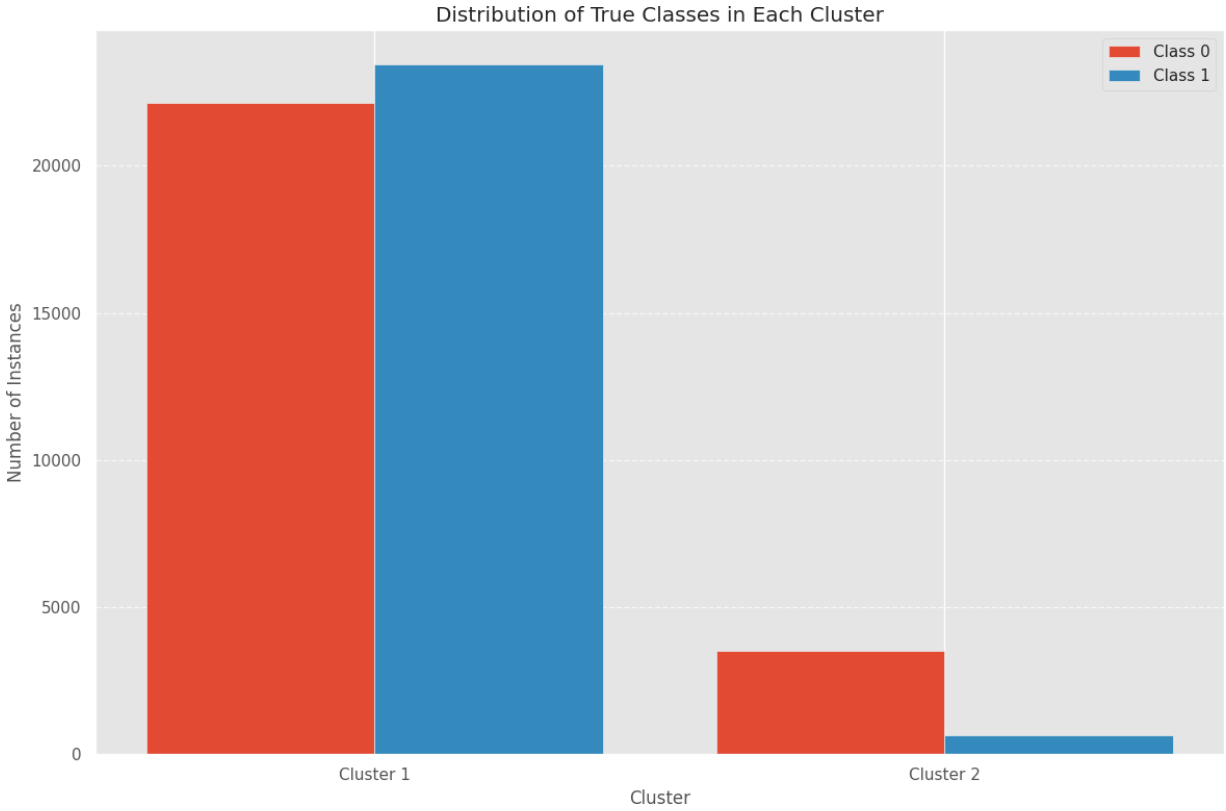
Normalized Mutual Information (NMI): 0.04457003668016213

silhouette_score: 0.11143880756849377

Comparison with true labels:



Class Distribution among clusters:

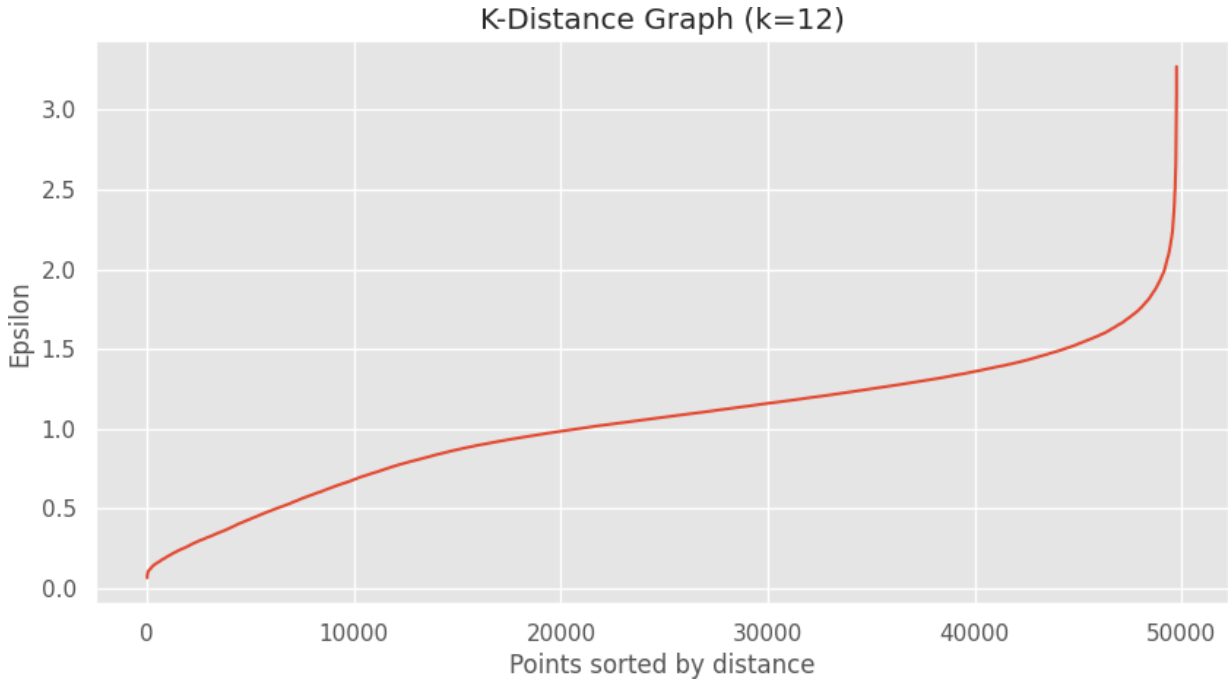


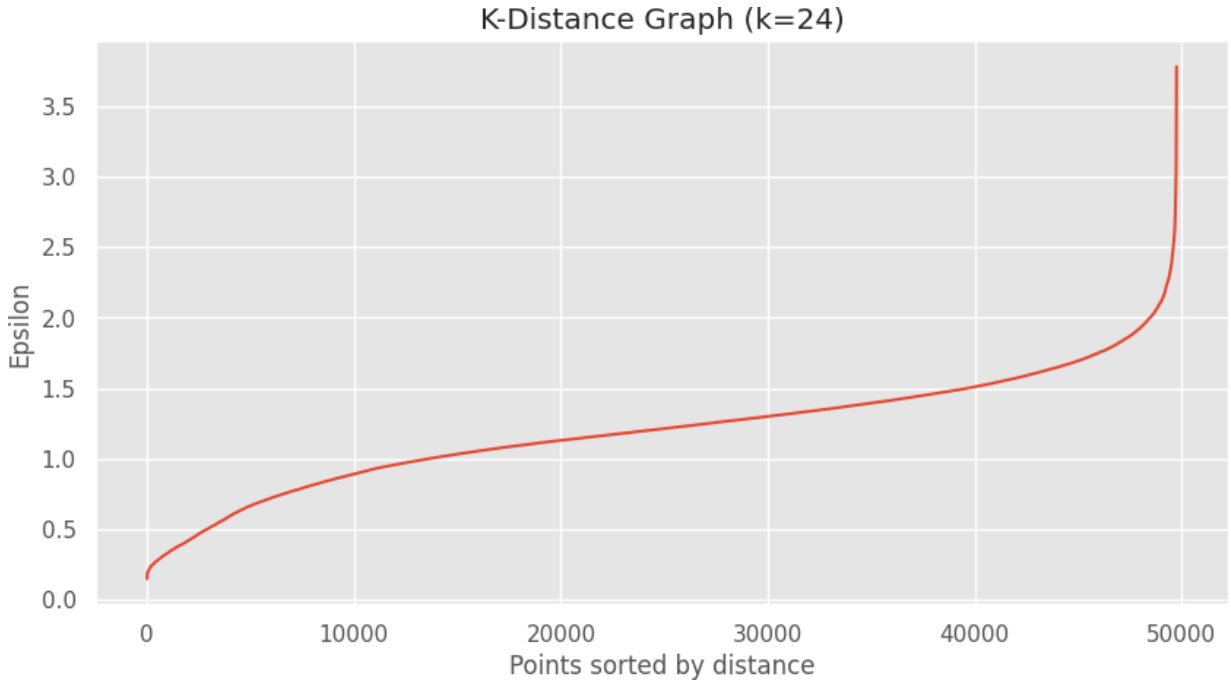
Cluster 1 has 22146 datapoints with label 0 and 23428 datapoints with label 1.

Cluster 2 has 3529 datapoints with label 0 and 646 datapoints with label 1.

13.4.1.3 DBScan Clustering:

Elbow method:





Elbow method shows eps=1.5 for min samples 12 and 24. We usually take value of k>=

dimension of data. Dimension of our data is 12. It is recommended to keep k between Dimentionality + 1 and 2 * Dimentionality.

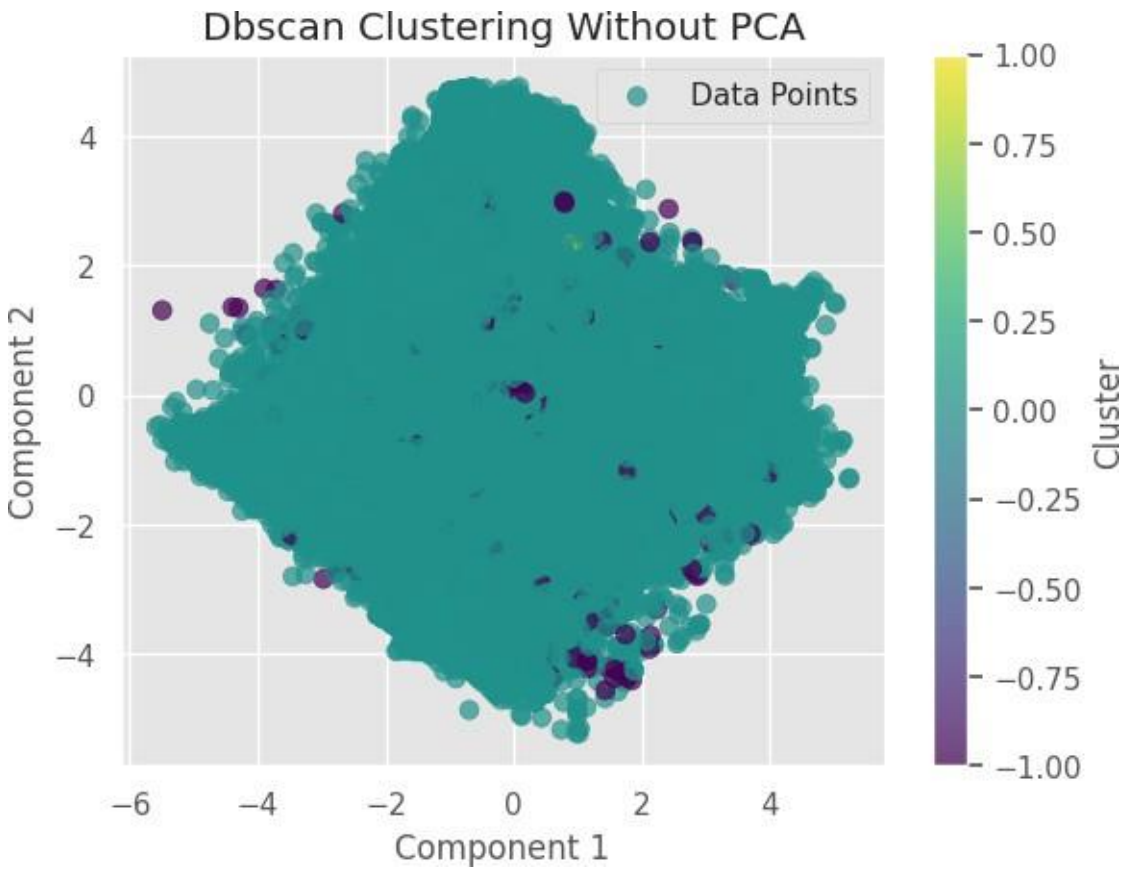
We tried experimenting DBScan for various values.

eps_range = [0.5, 1, 1.55, 2, 2.5, 3]

min_samples_range = [12, 15, 18, 21, 24]

We found 1.55 most suitable.

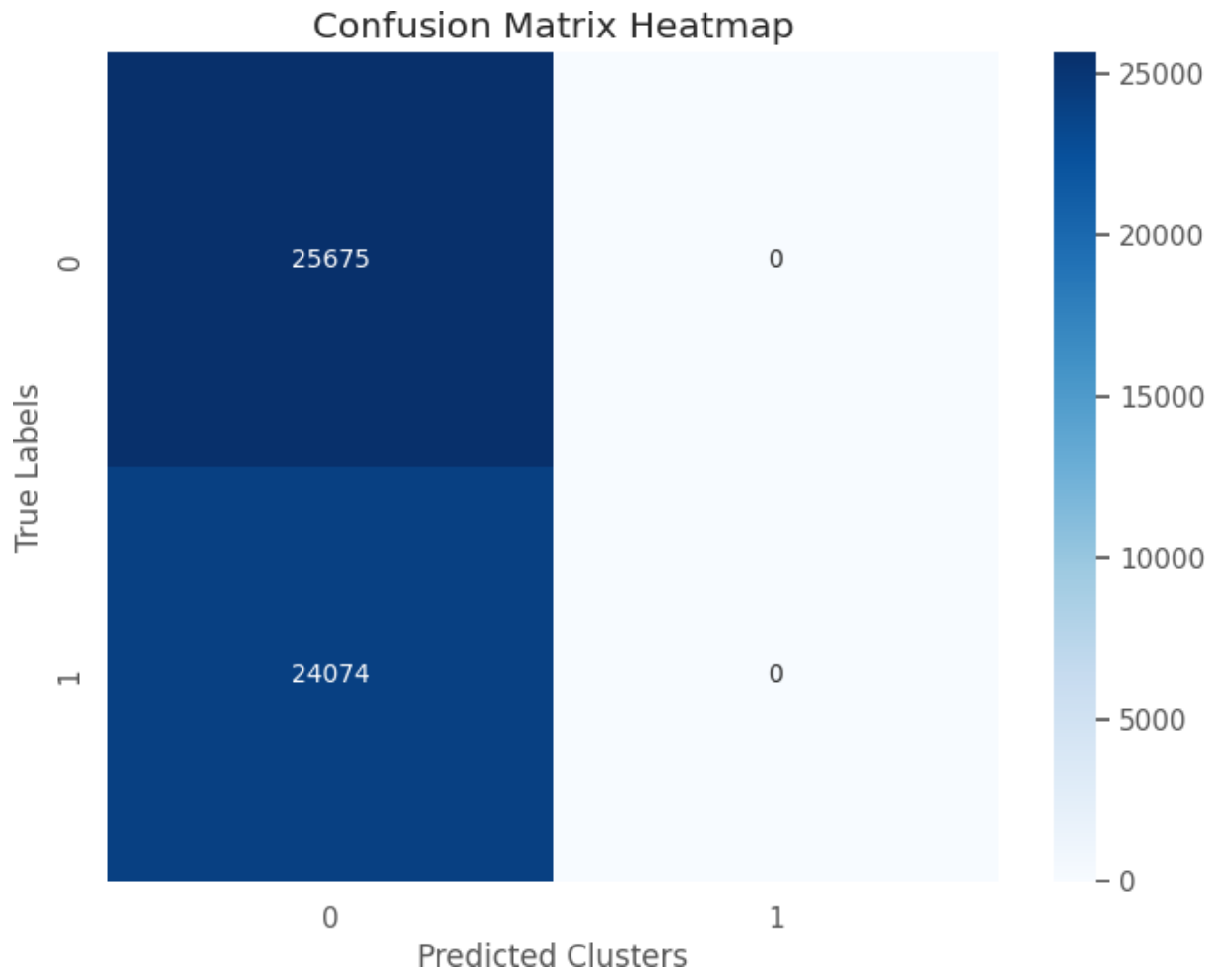
Visualization:



Evaluation Metrics:

Cluster Purity: 0.5160907756939838

Confusion matrix:

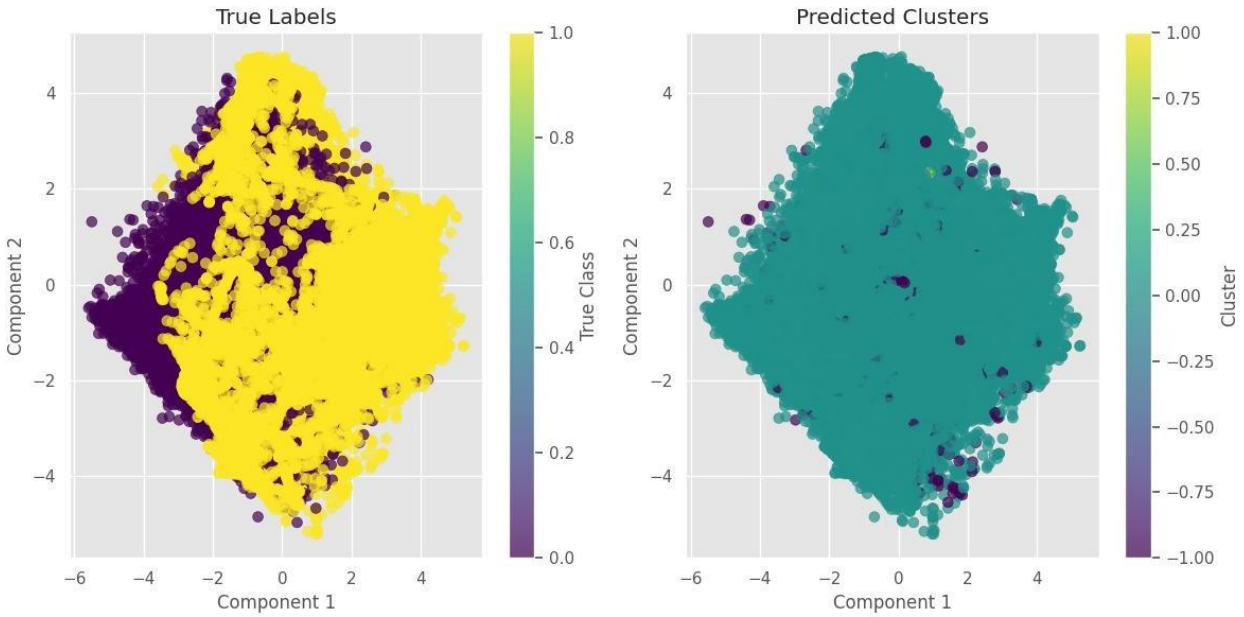


Other Metrics:

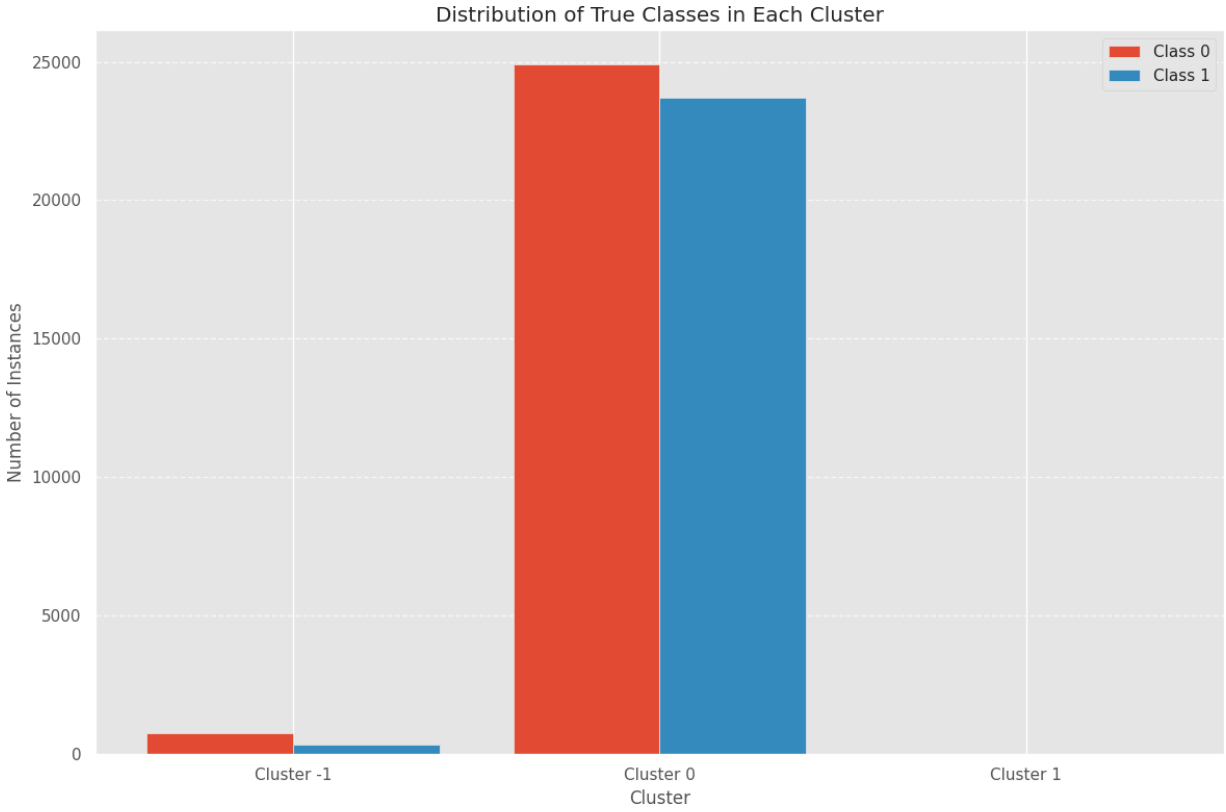
Normalized Mutual Information (NMI): 0.003746102025084575

silhouette_score: -0.05341848963104839

Comparison with true labels:



Class Distribution among clusters:



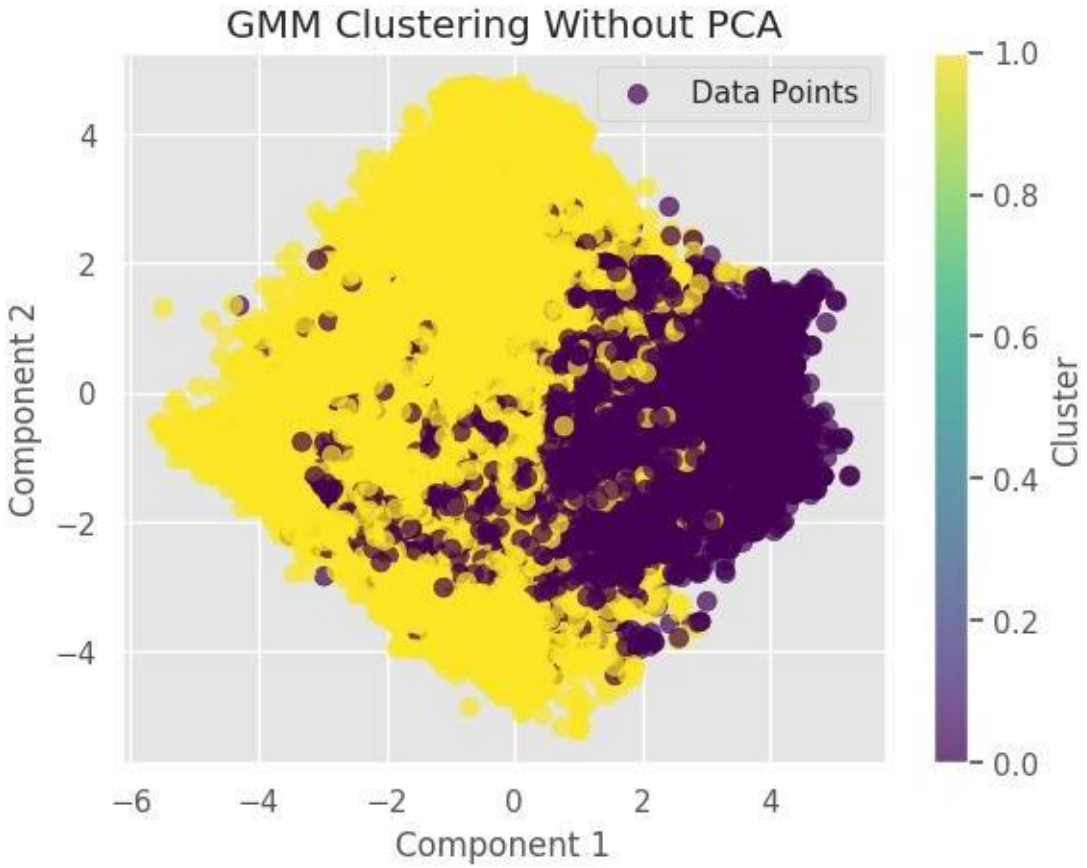
Cluster -1 has 775 datapoints with label 0 and 353 datapoints with label 1.

Cluster 0 has 24892 datapoints with label 0 and 23721 datapoints with label 1.

Cluster 1 has 8 datapoints with label 0 and 0 datapoints with label 1.

13.4.1.4 Gaussian Mixture Model Clustering*:*

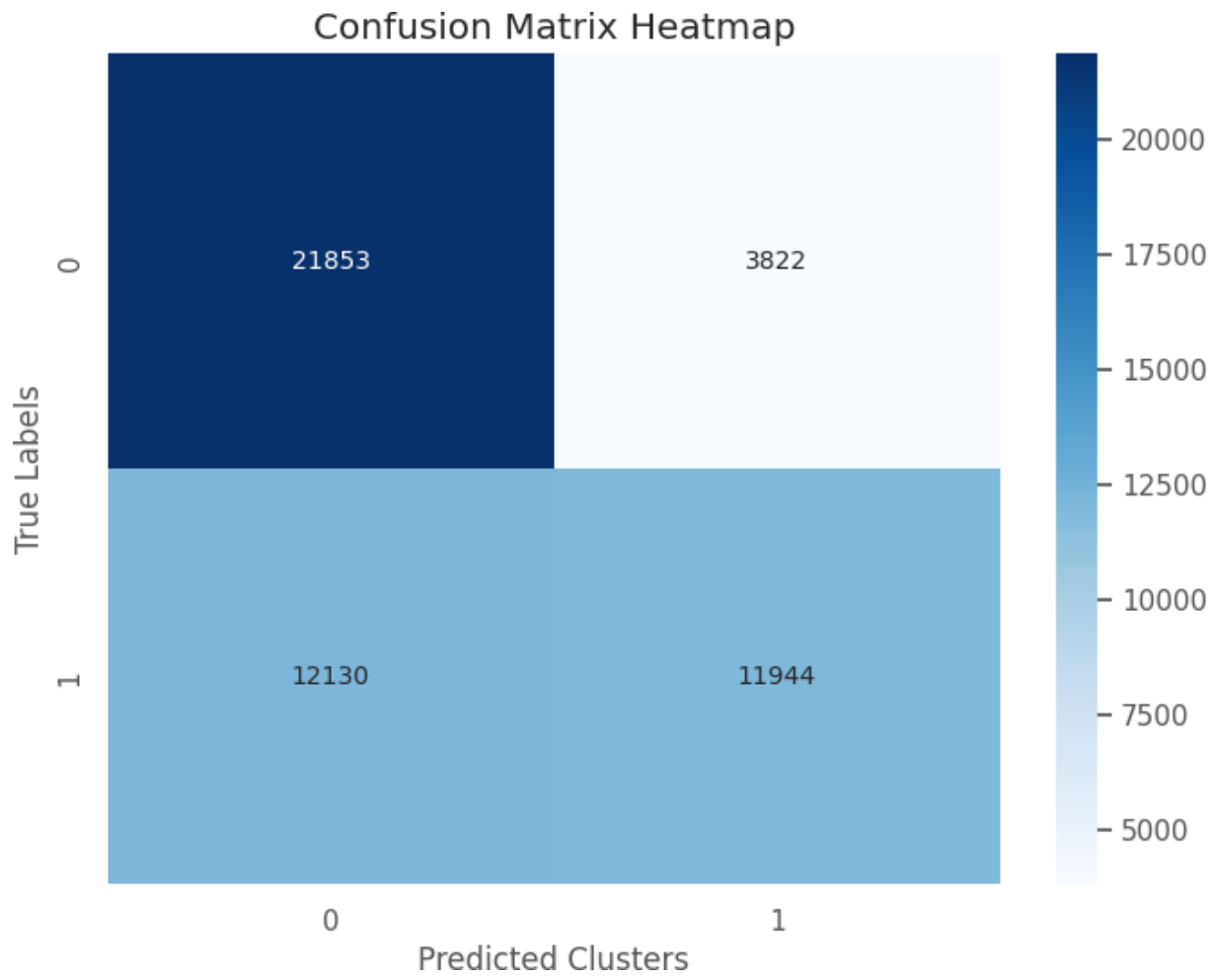
Visualization:



Evaluation Metrics:

Cluster Purity: 0.6793503387002754

Confusion matrix:

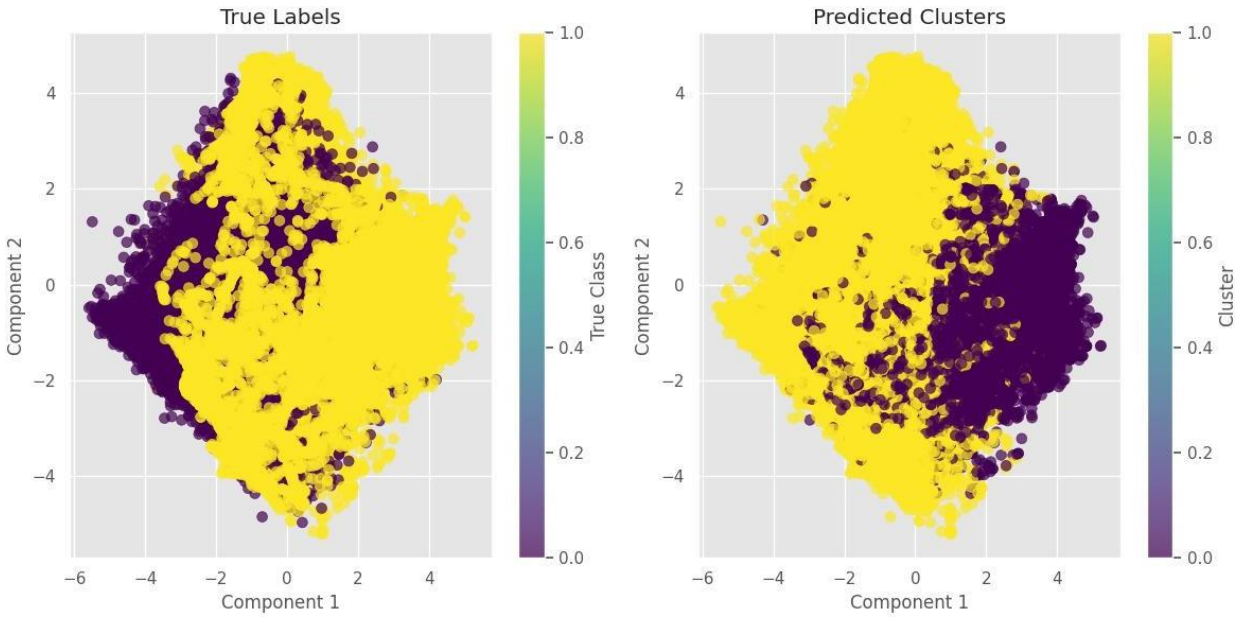


Other Metrics:

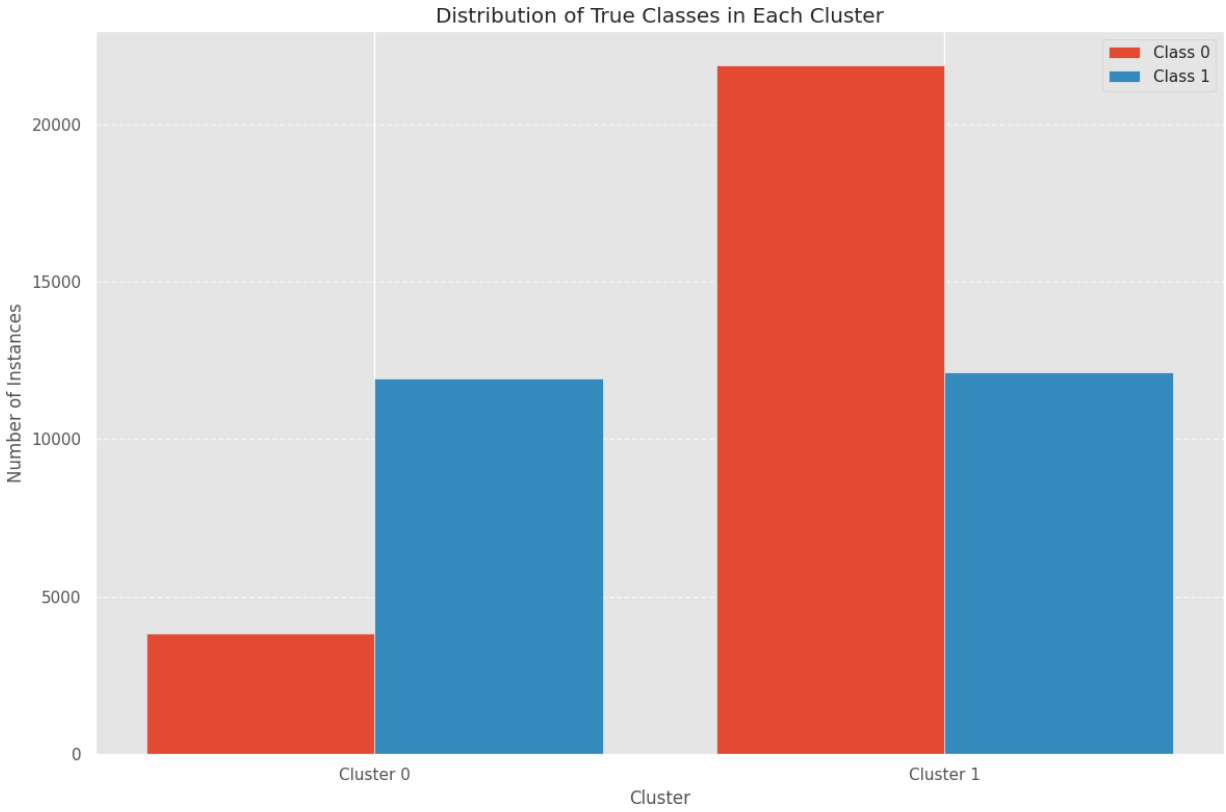
Normalized Mutual Information (NMI 0.10929577753138509

silhouette_score: 0.1437901284450953

Comparison with true labels:



Class Distribution among clusters:



Cluster 0 has 3822 datapoints with label 0 and 11944 datapoints with label 1.

Cluster 1 has 21853 datapoints with label 0 and 12130 datapoints with label 1.

13.4.1.5 Model Comparison:

13.4.1 SMOTE:

13.4.1.1 K-Means Clustering:

Elbow Method:

- **k = 3** is suggested by the elbow method, but since the dataset has two classes, **k = 2** is chosen.

Visualization:

- The clusters are visualized, showing separation based on the feature distributions.

Evaluation Metrics:

- **Cluster Purity:** 0.6773
  This indicates that around 67.73% of the data points in each cluster are correctly labeled.
- **Confusion Matrix:**
  The confusion matrix reflects how well the clusters match with the true labels.

Other Metrics:

- **Normalized Mutual Information (NMI):** 0.0945
  NMI is quite low, indicating a modest association between predicted and true labels.
- **Silhouette Score:** 0.2026
  This value indicates weak separation between clusters, as it's closer to 0. Higher values (closer to 1) would indicate better-defined clusters.

Class Distribution Among Clusters:

- **Cluster 0:**
  - 6002 datapoints with label 0
  - 14021 datapoints with label 1
- **Cluster 1:**
  - 19673 datapoints with label 0
  - 10053 datapoints with label 1

13.4.1.2 Hierarchical Clustering:

Dendrogram:

- A hierarchical dendrogram is used to visualize the clustering process. Different linkage methods are tested.

Ward Linkage:

- **Cluster Purity:** 0.7889
  Shows that about 78.89% of the points in each cluster are correctly labeled.
- **Confusion Matrix:**
  The confusion matrix indicates a reasonable match between predicted clusters and true labels.

Other Metrics:

- **Normalized Mutual Information (NMI):** 0.2892
  This is higher than K-Means, suggesting that hierarchical clustering is more aligned with the true class labels.
- **Silhouette Score:** 0.1708
  Still a relatively low value, indicating that the clusters are not well-separated.

Class Distribution Among Clusters:

- **Cluster 1:**
  - 23683 datapoints with label 0
  - 8511 datapoints with label 1
- **Cluster 2:**
  - 1992 datapoints with label 0
  - 15563 datapoints with label 1

Centroid Linkage:

- **Cluster Purity:** 0.5162
  This shows worse performance compared to the Ward linkage. Centroid linkage struggles to capture the structure of the dataset.
- **Confusion Matrix:**
  Poor performance as reflected in the low cluster purity.

Other Metrics:

- **Normalized Mutual Information (NMI):** 0.0003
  Extremely low, indicating that centroid linkage does not correspond well with true labels.
- **Silhouette Score:** 0.4065
  A better score compared to Ward linkage, but still not great for this dataset.

Class Distribution Among Clusters:

- **Cluster 1:**
  - 0 datapoints with label 0
  - 7 datapoints with label 1
- **Cluster 2:**
  - 25675 datapoints with label 0
  - 24067 datapoints with label 1

Complete Linkage:

- **Cluster Purity:** 0.5419
  This shows moderate performance, but still inferior to Ward linkage.

Other Metrics:

- **Normalized Mutual Information (NMI):** 0.0446
  This is still relatively low, suggesting that complete linkage doesn't map well to the true classes.
- **Silhouette Score:** 0.1114
  A very low score, indicating poor clustering quality.

Class Distribution Among Clusters:

- **Cluster 1:**
  - 22146 datapoints with label 0
  - 23428 datapoints with label 1
- **Cluster 2:**
  - 3529 datapoints with label 0
  - 646 datapoints with label 1

13.4.1.3 DBSCAN Clustering:

Elbow Method:

- **eps = 1.5** and **min_samples = 12** are chosen as the most suitable values after experimenting with multiple parameters.

Visualization:

- Visualizations show the effect of DBSCAN on the dataset.

Evaluation Metrics:

- **Cluster Purity:** 0.5161
  Low purity, indicating that the clusters formed by DBSCAN do not match the true class distribution effectively.

- **Confusion Matrix:**
  The confusion matrix reveals that DBSCAN struggles to correctly identify the true classes.

Other Metrics:

- **Normalized Mutual Information (NMI):** 0.0037
  Extremely low, suggesting that DBSCAN fails to align with the true labels.
- **Silhouette Score:** -0.0534
  A negative score, indicating that DBSCAN produces clusters that are poorly separated.

Class Distribution Among Clusters:

- **Cluster -1 (Noise):**
  - 775 datapoints with label 0
  - 353 datapoints with label 1
- **Cluster 0:**
  - 24892 datapoints with label 0
  - 23721 datapoints with label 1
- **Cluster 1:**
  - 8 datapoints with label 0
  - 0 datapoints with label 1

13.4.1.4 Gaussian Mixture Model Clustering:

Visualization:

- Visualizations show the clustering results of the Gaussian Mixture Model.

Evaluation Metrics:

- **Cluster Purity:** 0.6794
  This is a moderate purity score, showing that GMM can cluster some data points correctly.
- **Confusion Matrix:**
  The confusion matrix shows a reasonable match, though not perfect.

Other Metrics:

- **Normalized Mutual Information (NMI):** 0.1093
  Shows a weak alignment with the true labels.
- **Silhouette Score:** 0.1438
  A low score, indicating that GMM has suboptimal clustering performance for this dataset.

Class Distribution Among Clusters:

- **Cluster 0:**
  - 3822 datapoints with label 0
  - 11944 datapoints with label 1
- **Cluster 1:**
  - 21853 datapoints with label 0
  - 12130 datapoints with label 1

Summary of Model Comparison:

Clustering Method	Evaluation Metric	Cluster Purity	NMI	Silhouette Score	Class Distribution (Cluster 1)	Class Distribution (Cluster 2)
K-Means Clustering	-	0.6773	0.0945	0.2026	6002 (label 0), 14021 (label 1)	19673 (label 0), 10053 (label 1)
Hierarchical Clustering	**Ward Linkage**	0.7889	0.2892	0.1708	23683 (label 0), 8511 (label 1)	1992 (label 0), 15563 (label 1)
	Centroid Linkage	0.5162	0.0003	0.4065	0 (label 0), 7 (label 1)	25675 (label 0), 24067 (label 1)
	Complete Linkage	0.5419	0.0446	0.1114	22146 (label 0), 23428 (label 1)	3529 (label 0), 646 (label 1)
DBScan Clustering	-	0.5161	0.0037	-0.0534	775 (label 0), 353 (label 1)	24892 (label 0), 23721 (label 1)
Gaussian Mixture Model (GMM)	-	0.6794	0.1093	0.1438	3822 (label 0), 11944 (label 1)	21853 (label 0), 12130 (label 1)

Key Observations:

1. **Best Cluster Purity**: Hierarchical Clustering with Ward Linkage (0.7889).
2. **Best NMI**: Hierarchical Clustering with Ward Linkage (0.2892).
3. **Best Silhouette Score**: Hierarchical Clustering with Centroid Linkage (0.4065).
4. **Worst Performance**: DBScan, with a negative silhouette score indicating poor clustering structure.

- **K-Means Clustering:** Shows moderate cluster purity but weak cluster separation based on silhouette score.

- **Hierarchical Clustering (Ward Linkage):** Performs better than K-Means, with a higher NMI and reasonable class separation, though still suboptimal for this dataset.
- **DBSCAN:** Poor performance, with a low cluster purity and a negative silhouette score, suggesting that DBSCAN is not suitable for this dataset.
- **Gaussian Mixture Model (GMM):** Provides moderate clustering performance, but the clusters are not perfectly aligned with the true labels, as shown by the low NMI and silhouette score.

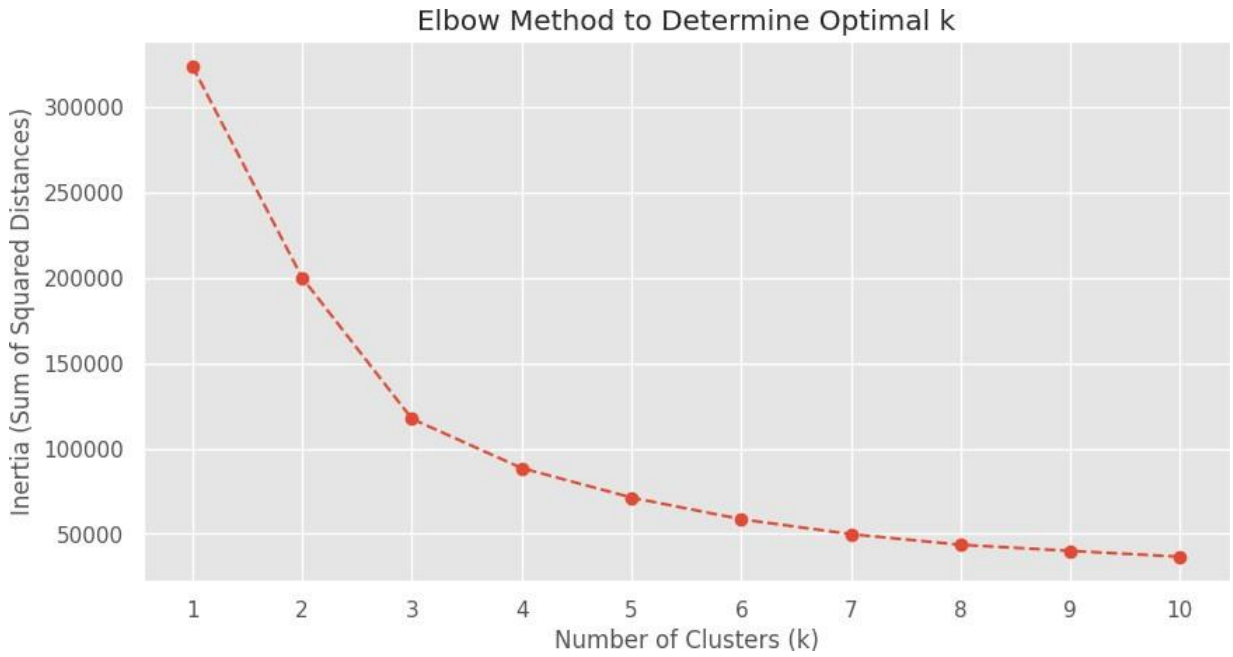
In conclusion, **K-Means** and **Hierarchical Clustering (Ward Linkage)** provide the best results, with K-Means being the more stable method in terms of simplicity, while **DBSCAN** and **GMM** struggle to provide meaningful clusters for this dataset.

14. Unsupervised with PCA All years

13.4.1 SMOTE:

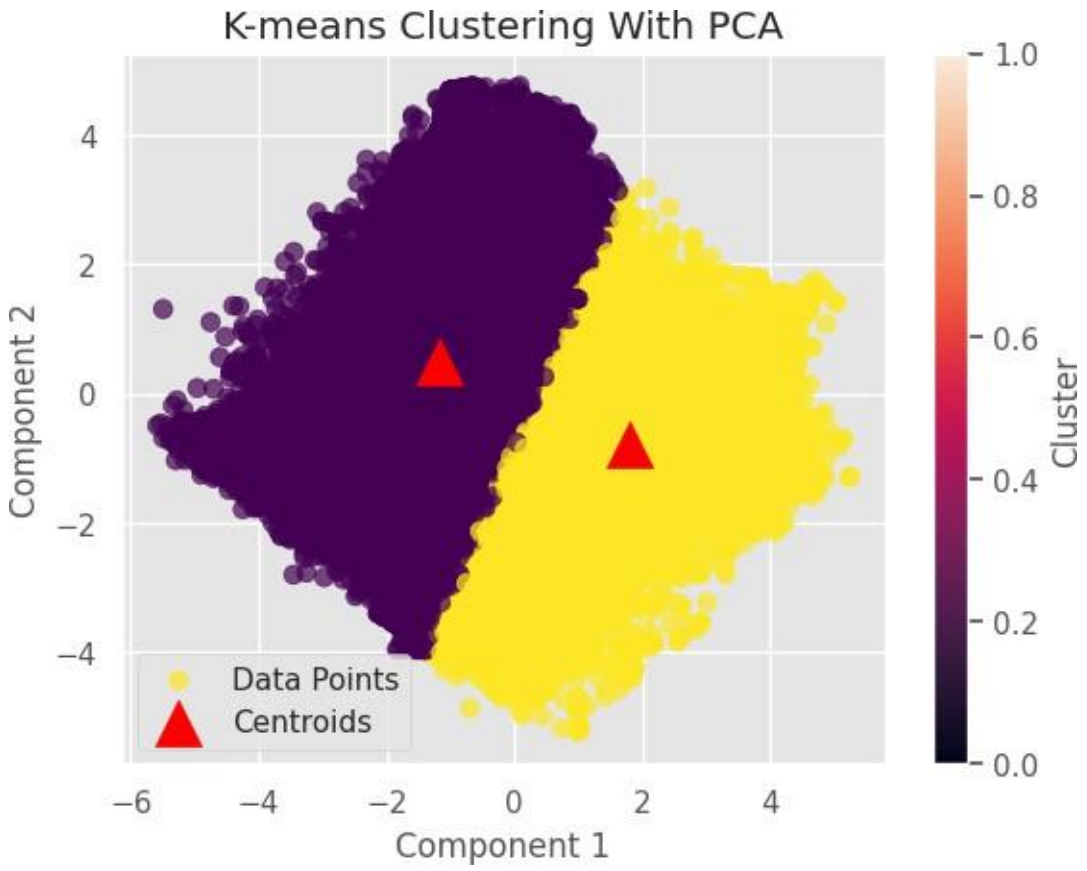
13.4.1.1 K Means Clustering:

Elbow method:



Elbow method shows k=3 but since our dataset is labelled and we are dealing with 2 classes, k = 2.

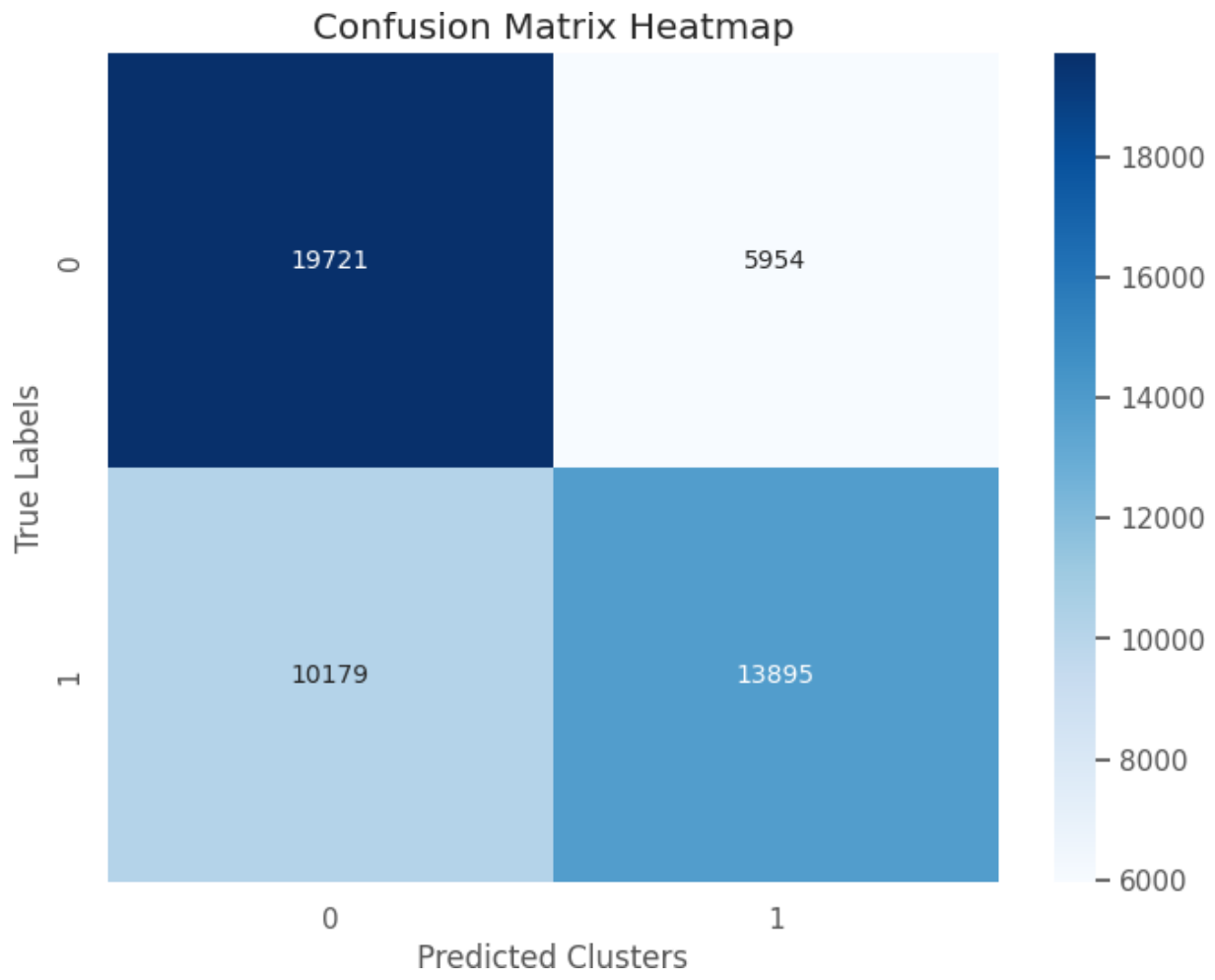
Visualization:



Evaluation Metrics:

Cluster Purity: 0.6757120746145651

Confusion matrix:

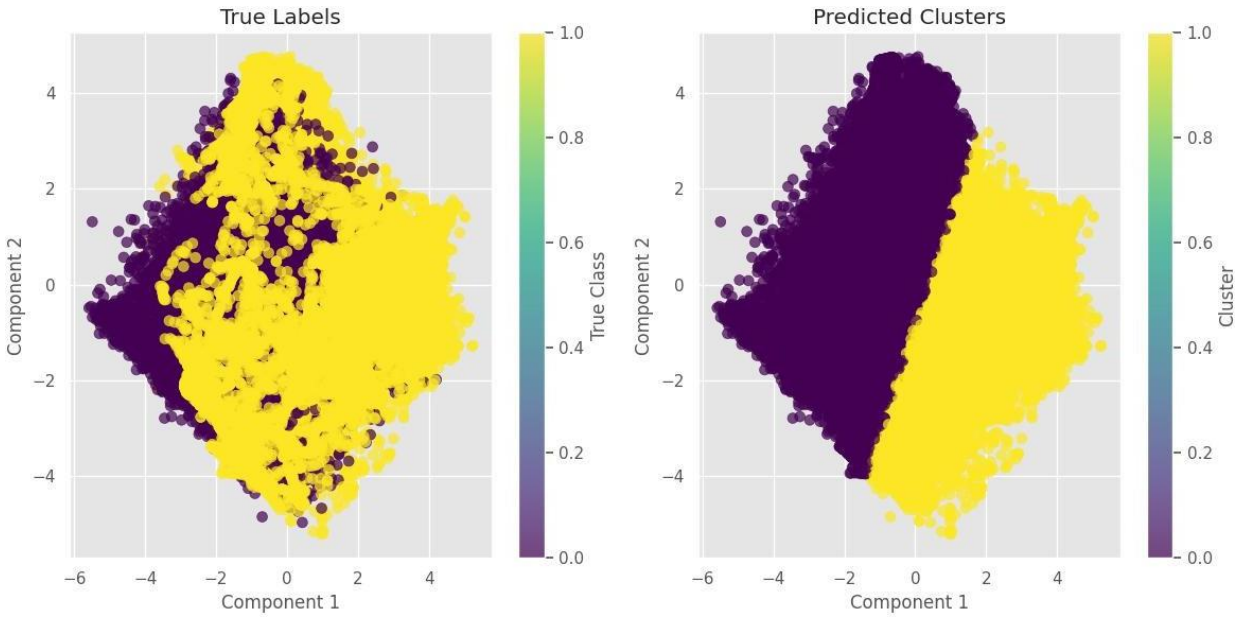


Other Metrics:

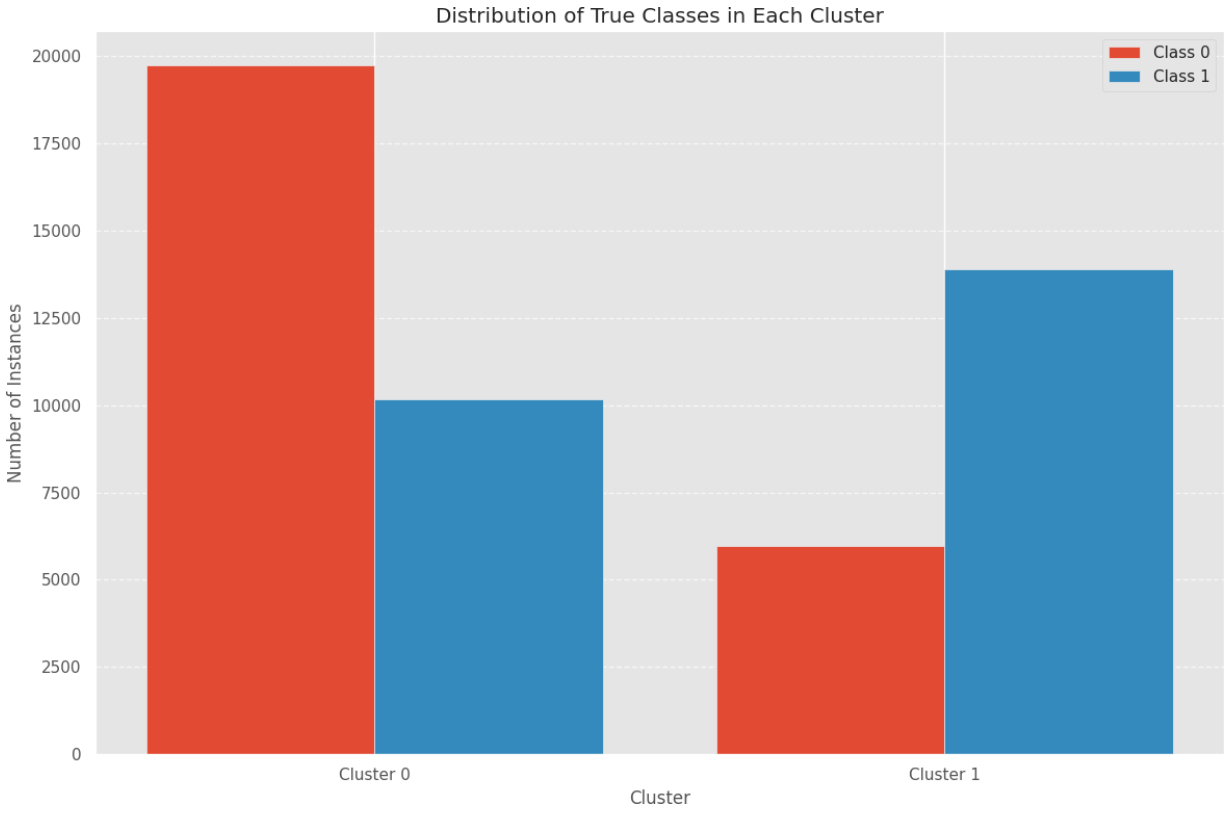
Normalized Mutual Information (NMI): 0.09298153191056266

silhouette_score: 0.3694185027617268

Comparison with true labels:



Class Distribution among clusters:

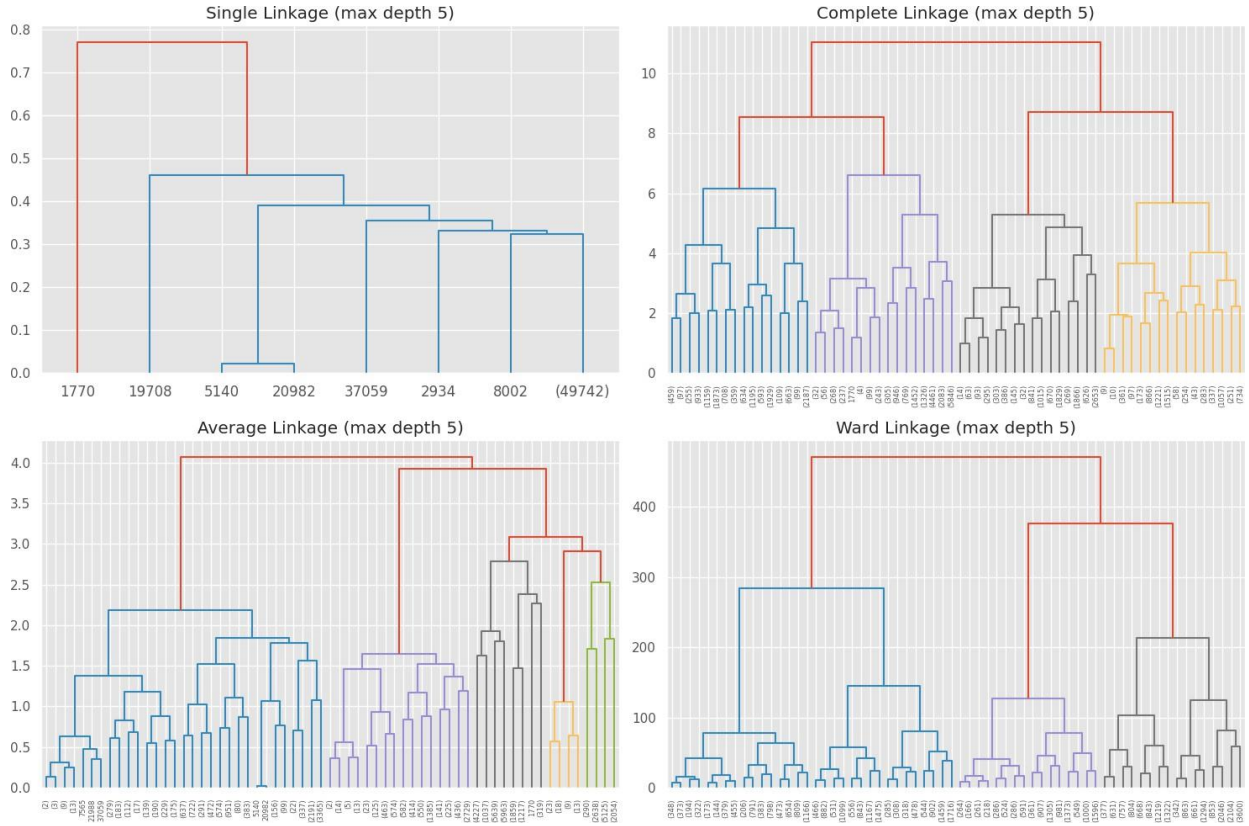


Cluster 0 has 6002 datapoints with label 0 and 14021 datapoints with label 1.

Cluster 1 has 19673 datapoints with label 0 and 10053 datapoints with label 1.

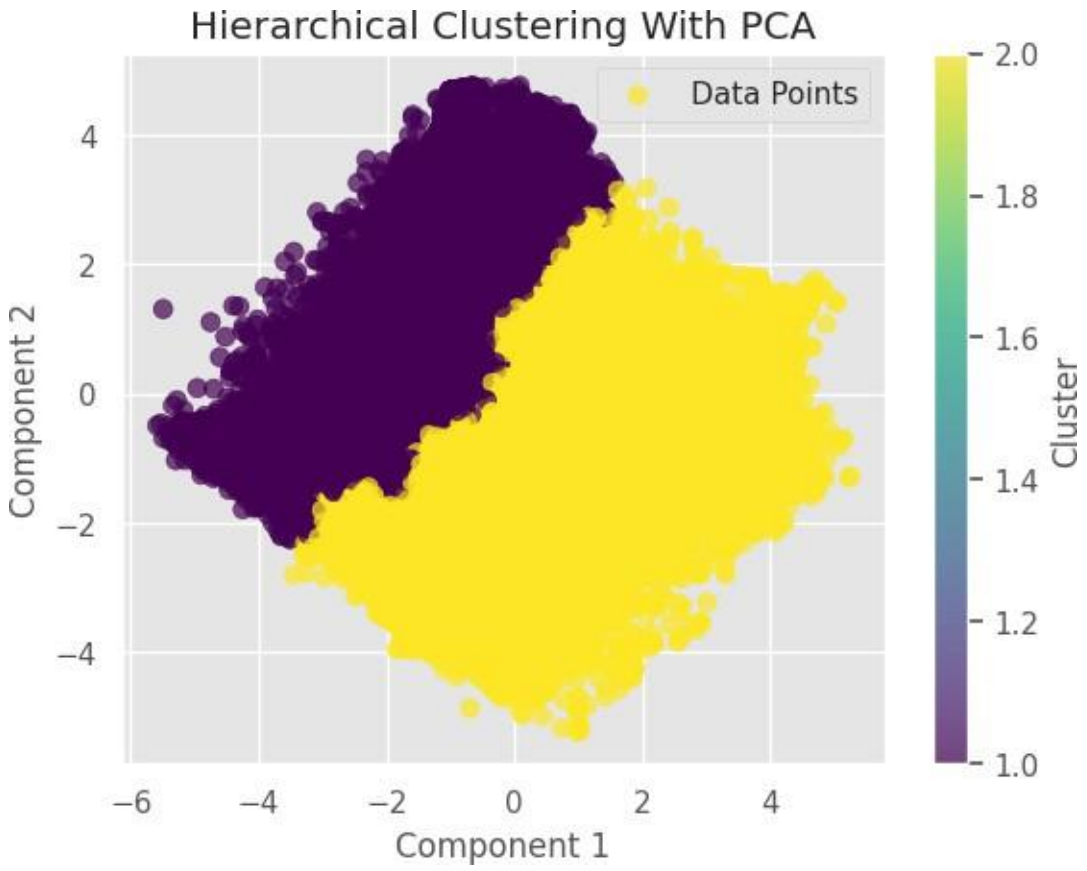
13.4.1.2 Hierarchical Clustering*:*

Dendrogram:



Ward Linkage:

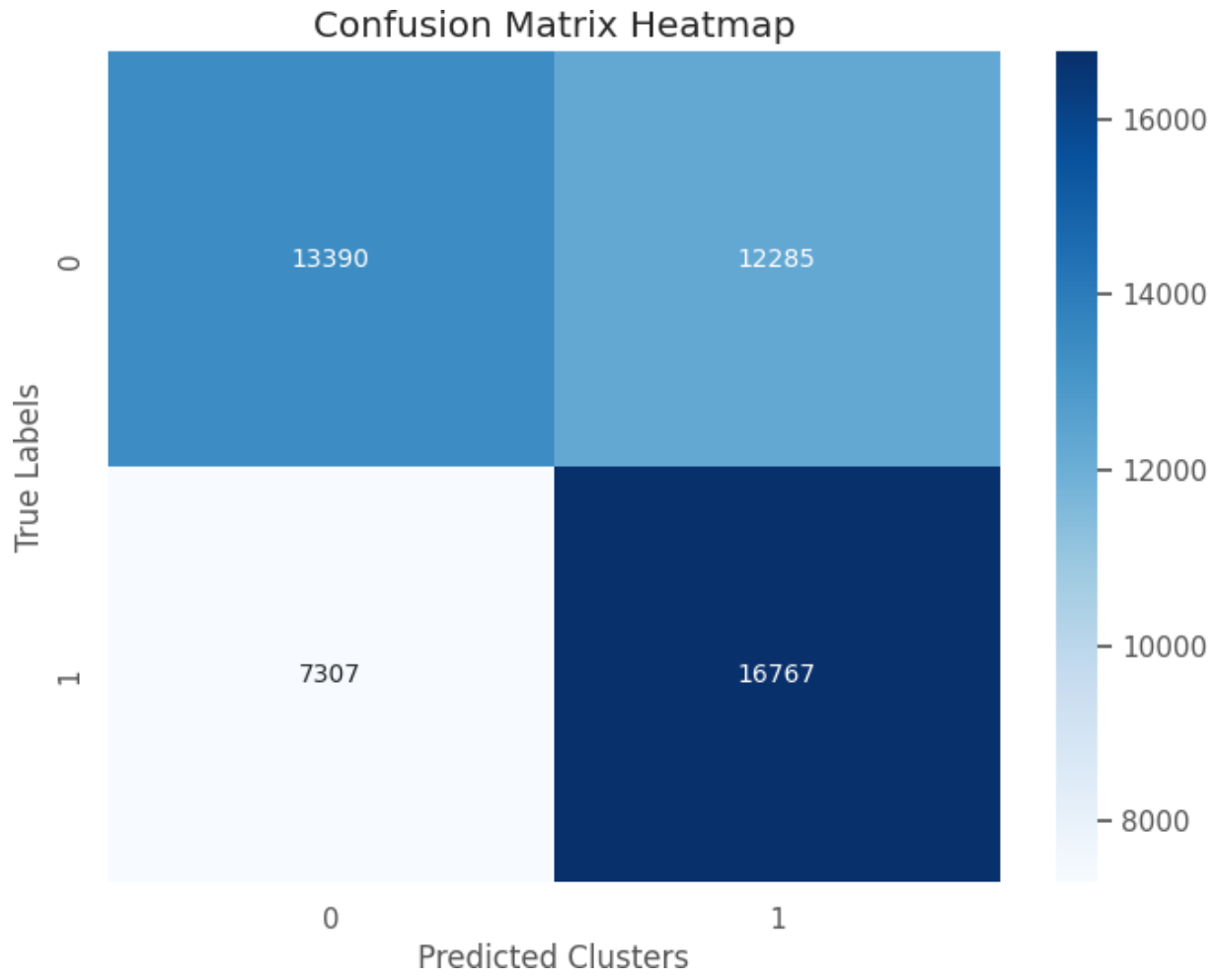
Visualization:



Evaluation Metrics:

Cluster Purity: 0.6061830388550523

Confusion matrix:

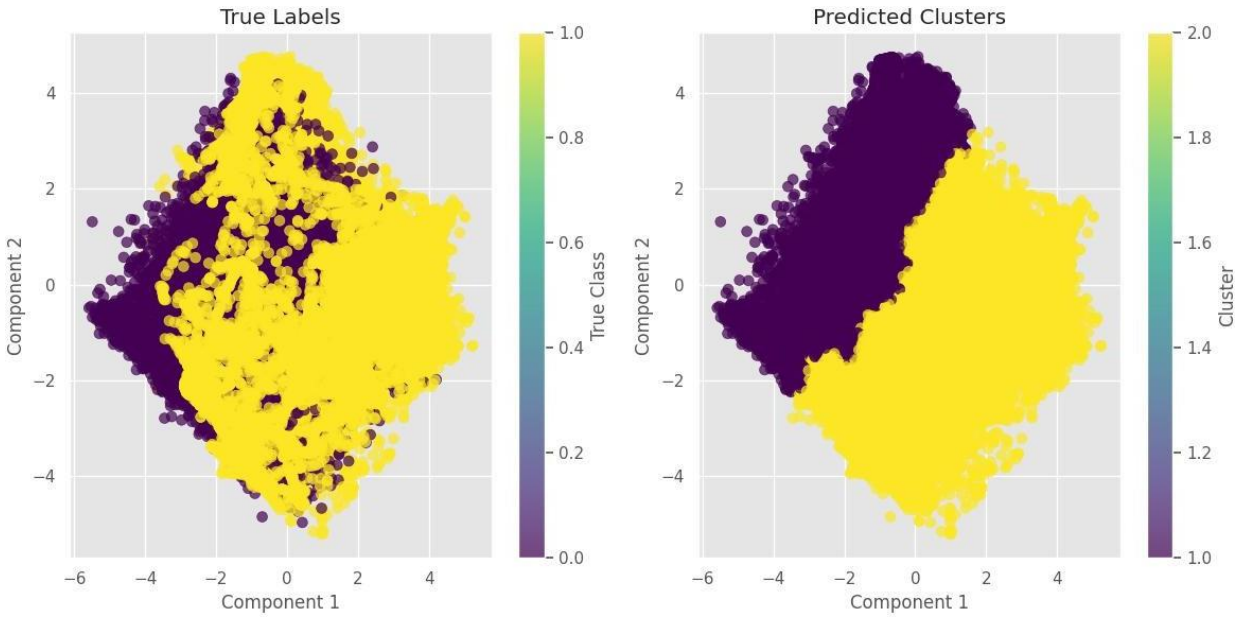


Other Metrics:

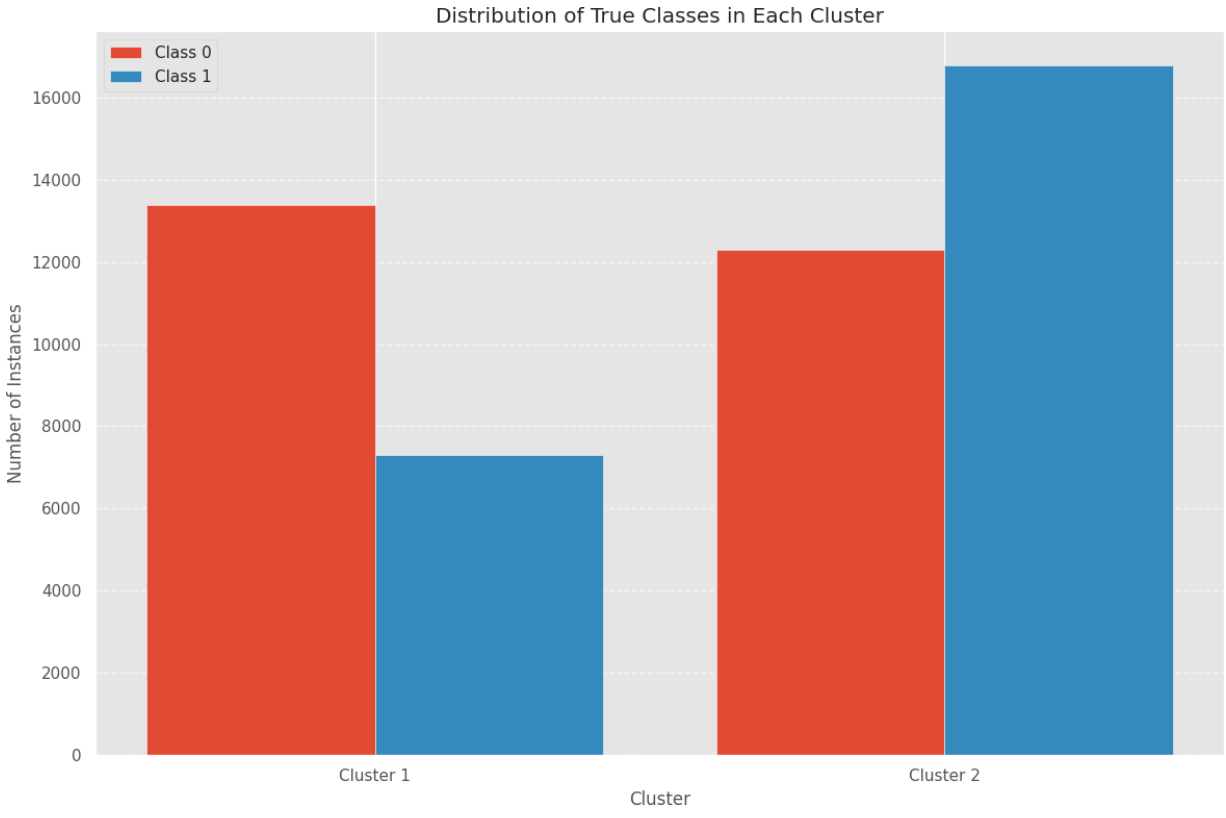
Normalized Mutual Information (NMI): 0.036010413629915264

silhouette_score: 0.3240694043231268

Comparison with true labels:



Class Distribution among clusters:

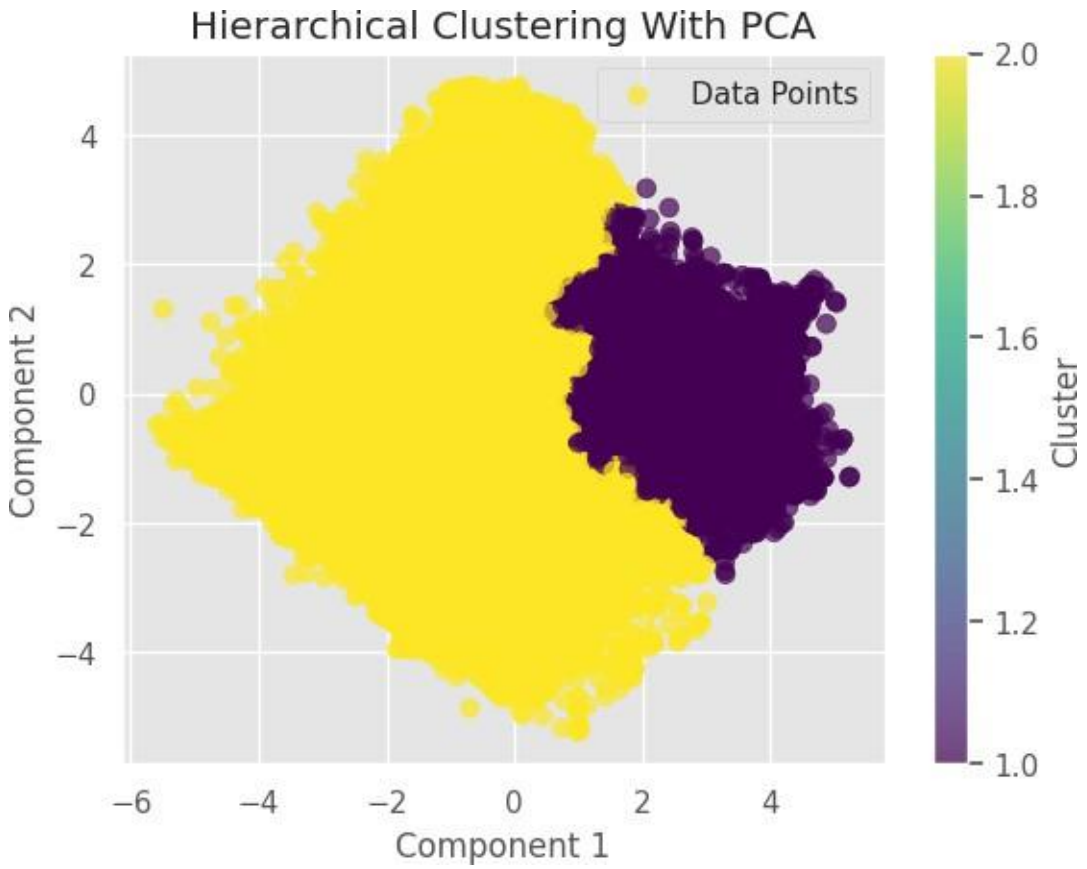


Cluster 1 has 13390 datapoints with label 0 and 7307 datapoints with label 1.

Cluster 2 has 12285 datapoints with label 0 and 16767 datapoints with label 1.

Average Linkage:

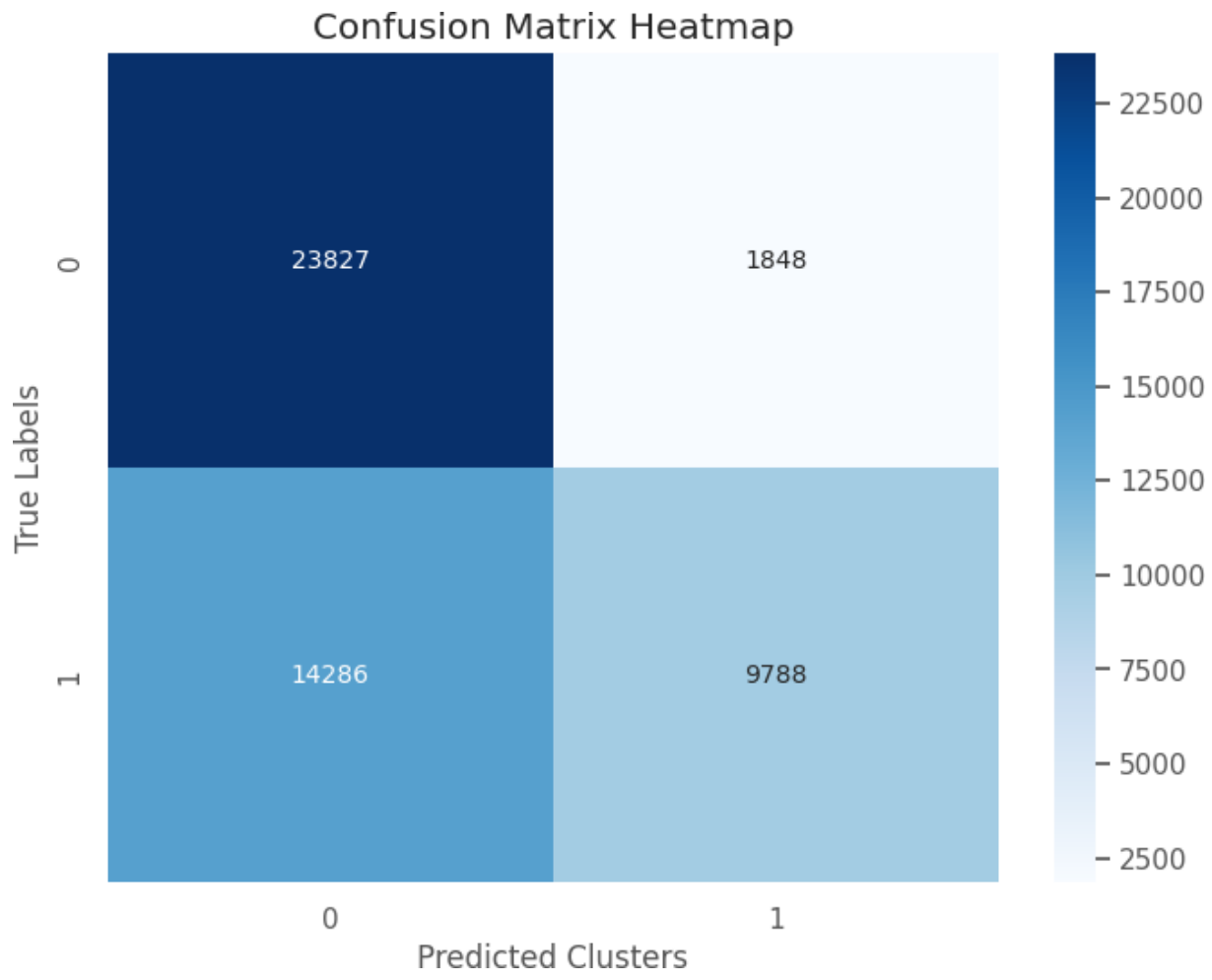
Visualization:



Evaluation Metrics:

Cluster Purity: 0.6756919737080143

Confusion matrix:

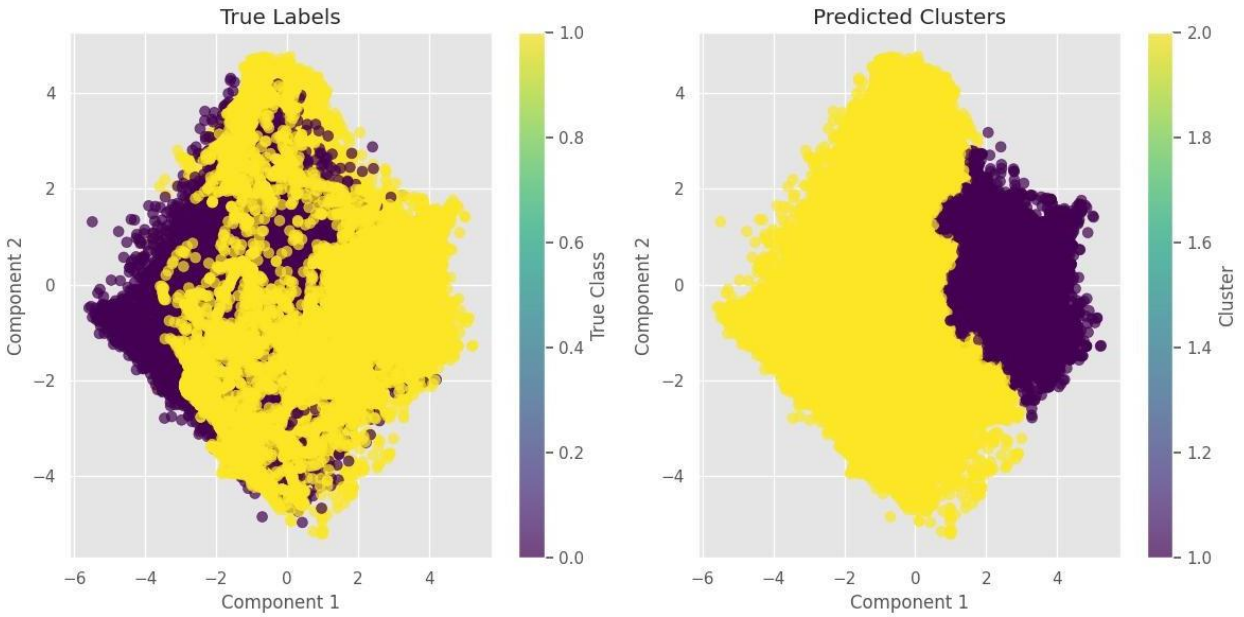


Other Metrics:

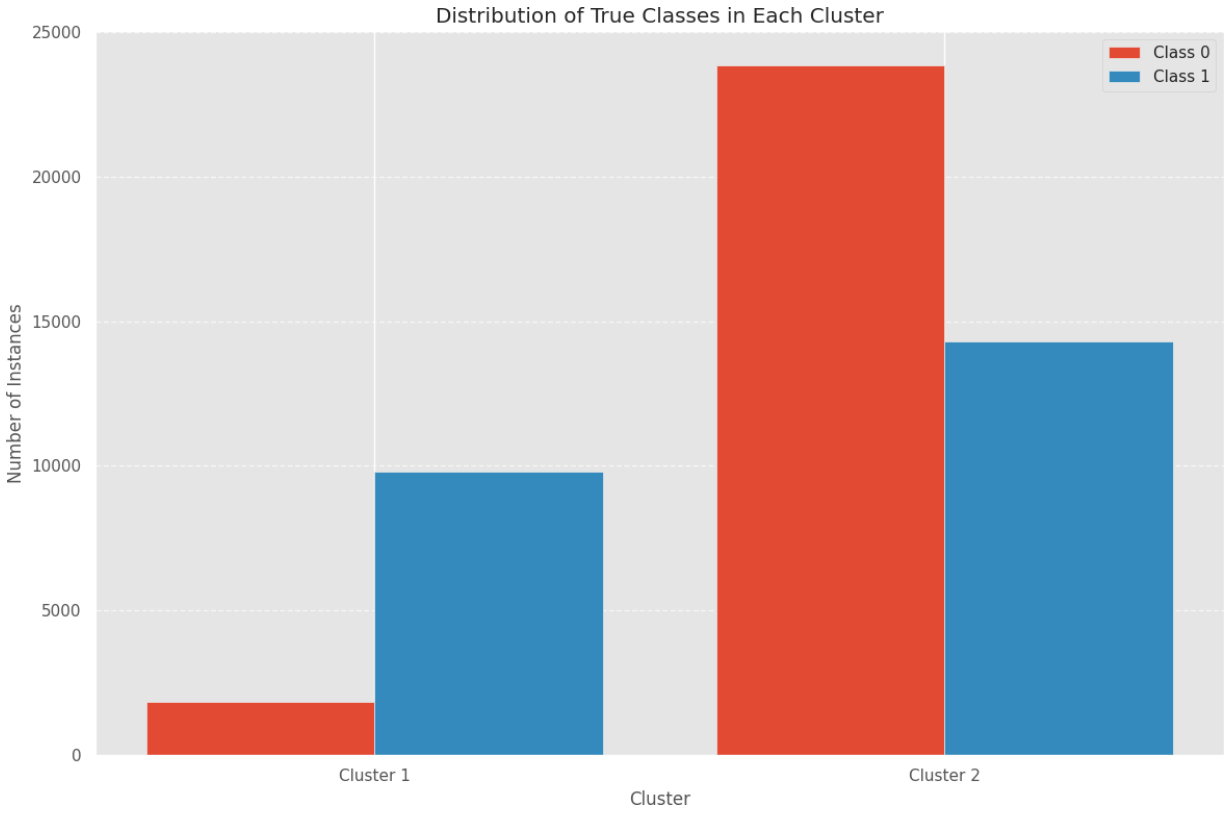
Normalized Mutual Information (NMI): 0.13503654490092615

silhouette_score: 0.34166607370162444

Comparison with true labels:



Class Distribution among clusters:

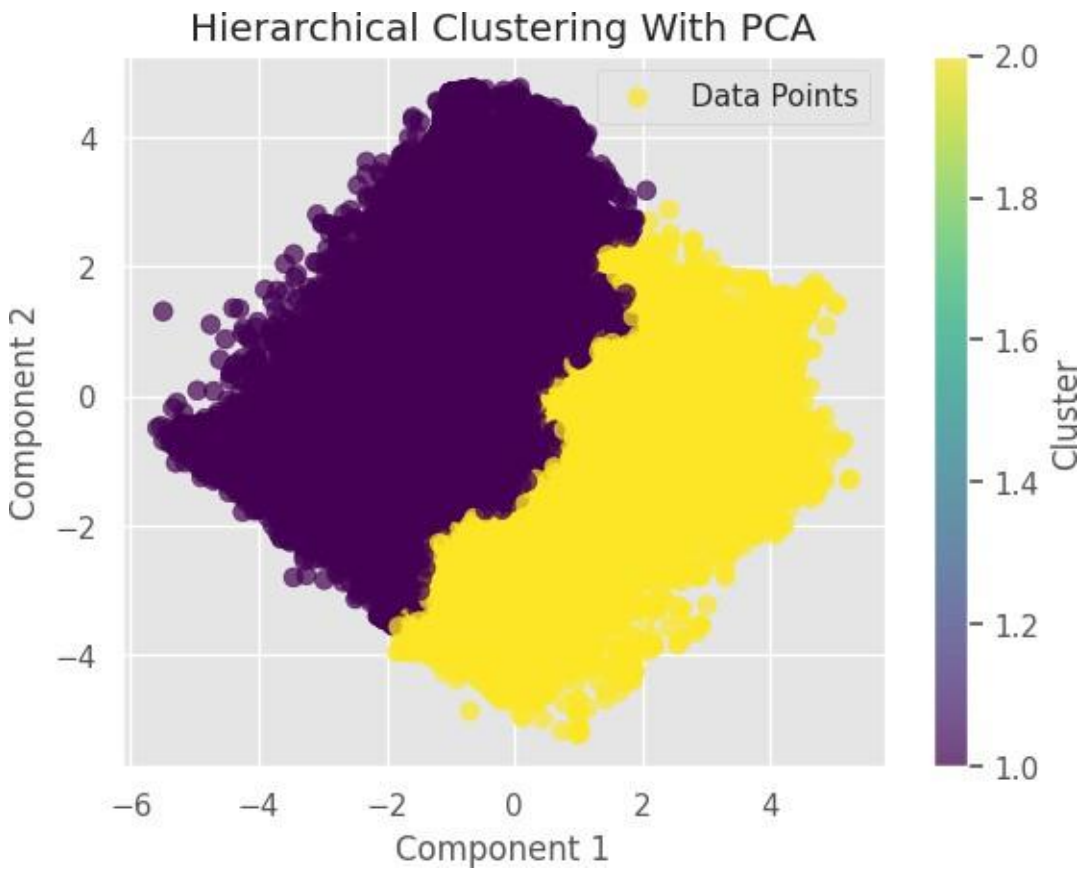


Cluster 1 has 1848 datapoints with label 0 and 9788 datapoints with label 1.

Cluster 2 has 23827 datapoints with label 0 and 14286 datapoints with label 1.

Complete Linkage:

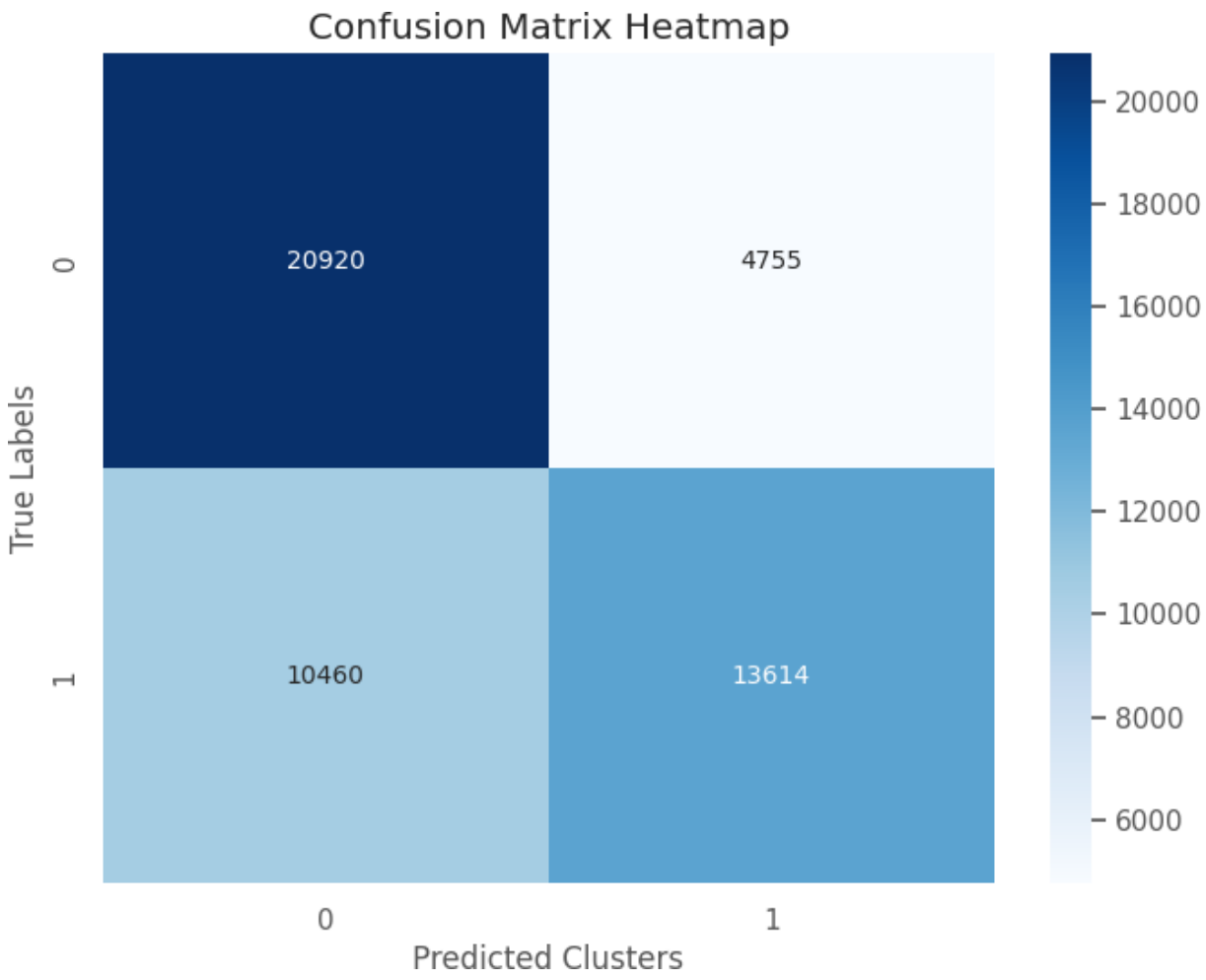
Visualization:



Evaluation Metrics:

Cluster Purity: 0.6941647068282779

Confusion matrix:

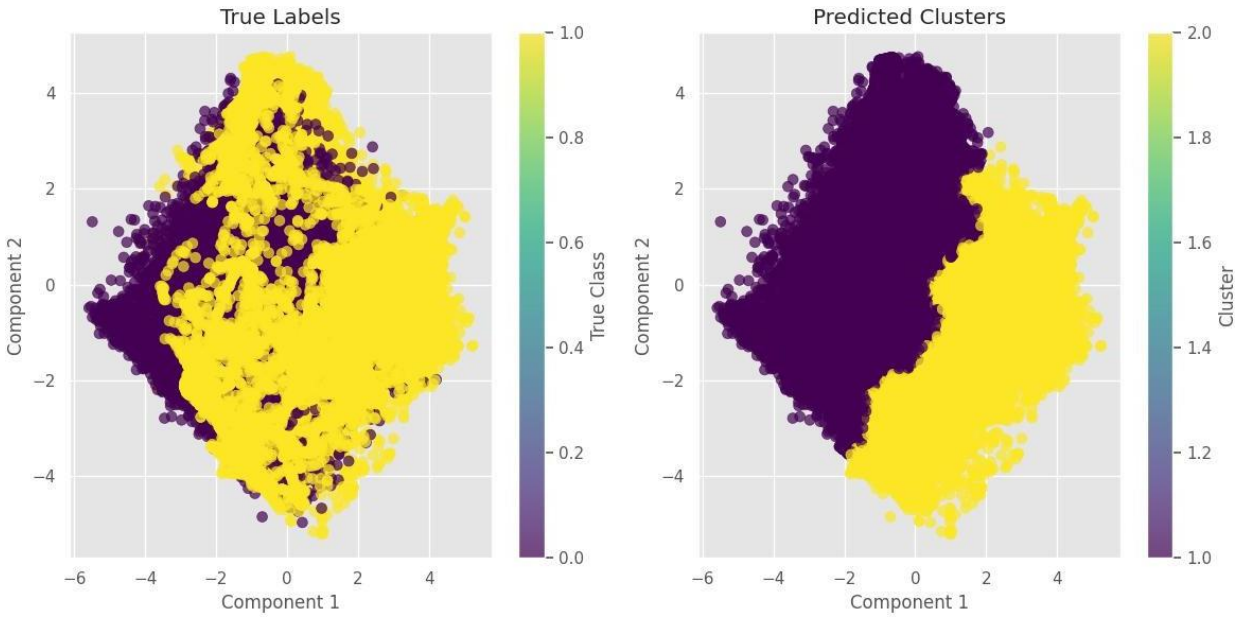


Other Metrics:

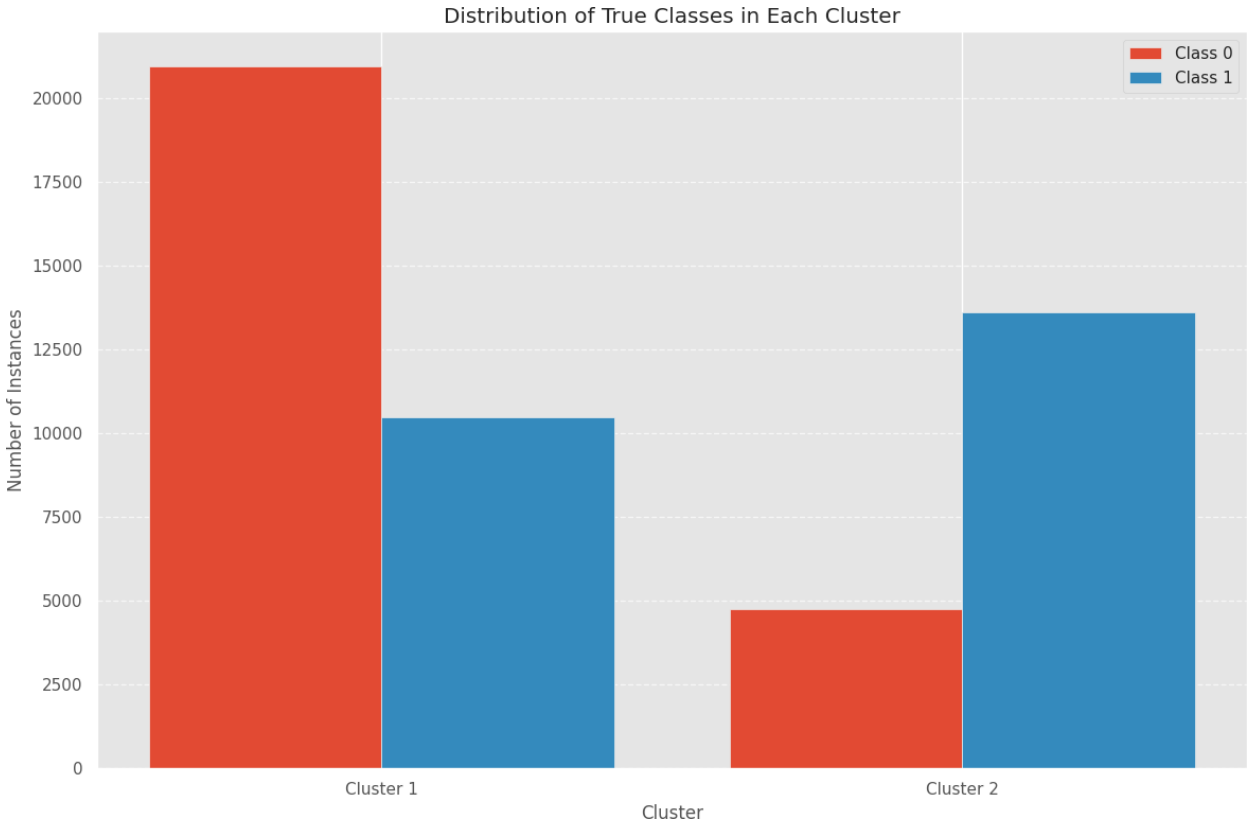
Normalized Mutual Information (NMI): 0.11839636960698631

silhouette_score: 0.36433467376577516

Comparison with true labels:



Class Distribution among clusters:

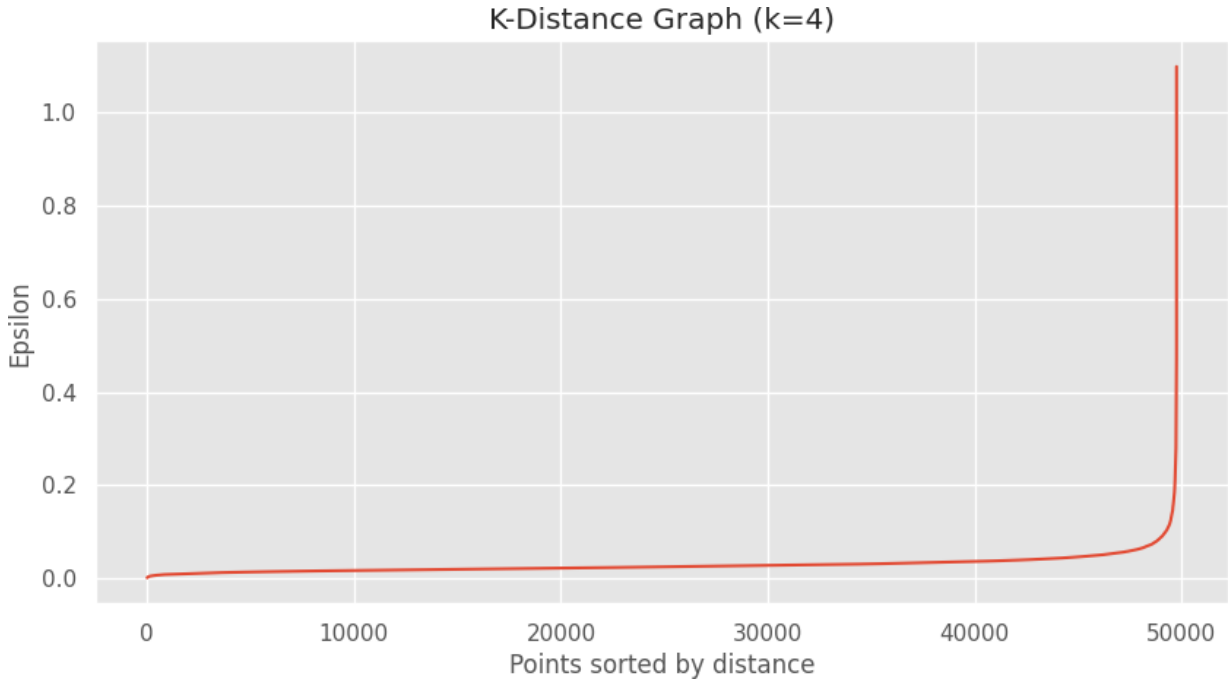


Cluster 1 has 20920 datapoints with label 0 and 10460 datapoints with label 1.

Cluster 2 has 4755 datapoints with label 0 and 13614 datapoints with label 1.

13.4.1.3 DBScan Clustering:

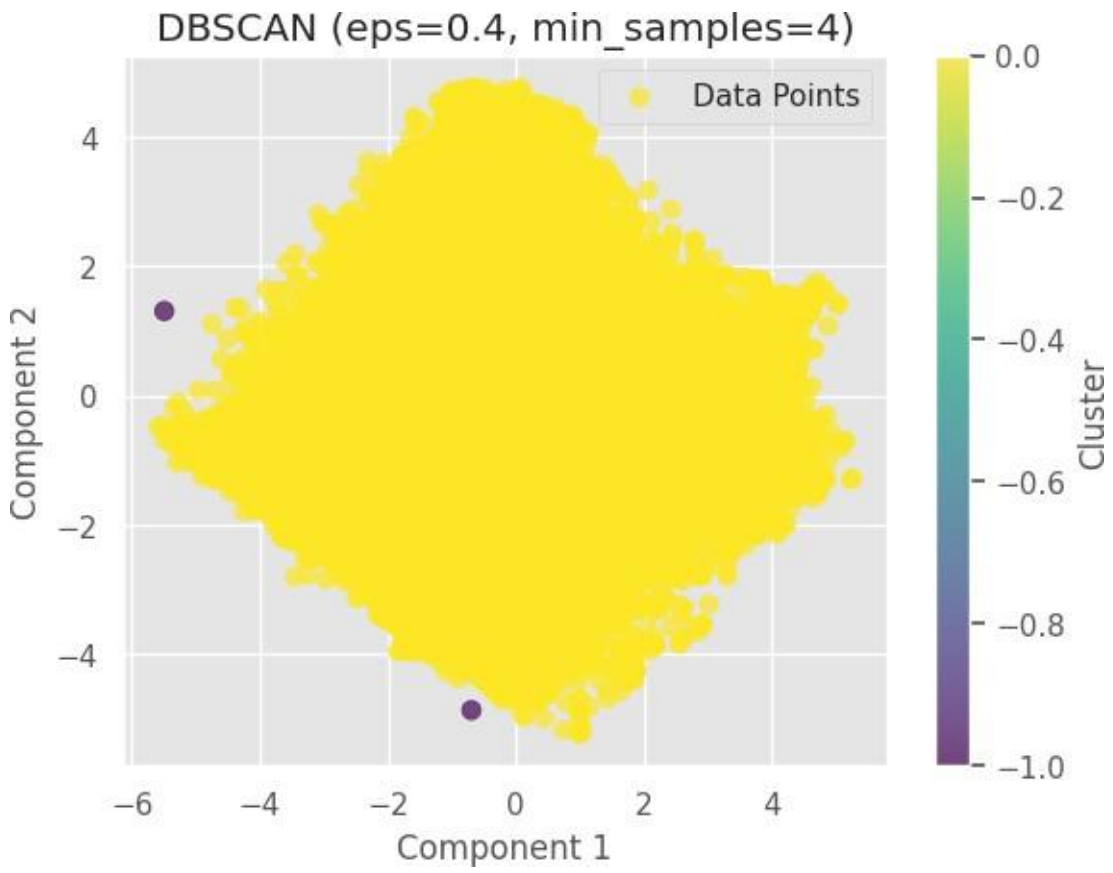
Elbow method:



Elbow method shows eps=0.1 for min samples 4 We usually take value of k>= dimension of data. Dimension of our data is 12. It is recommended to keep k between Dimentionality + 1 and 2 * Dimentionality.

We check for different values of eps and minpts.

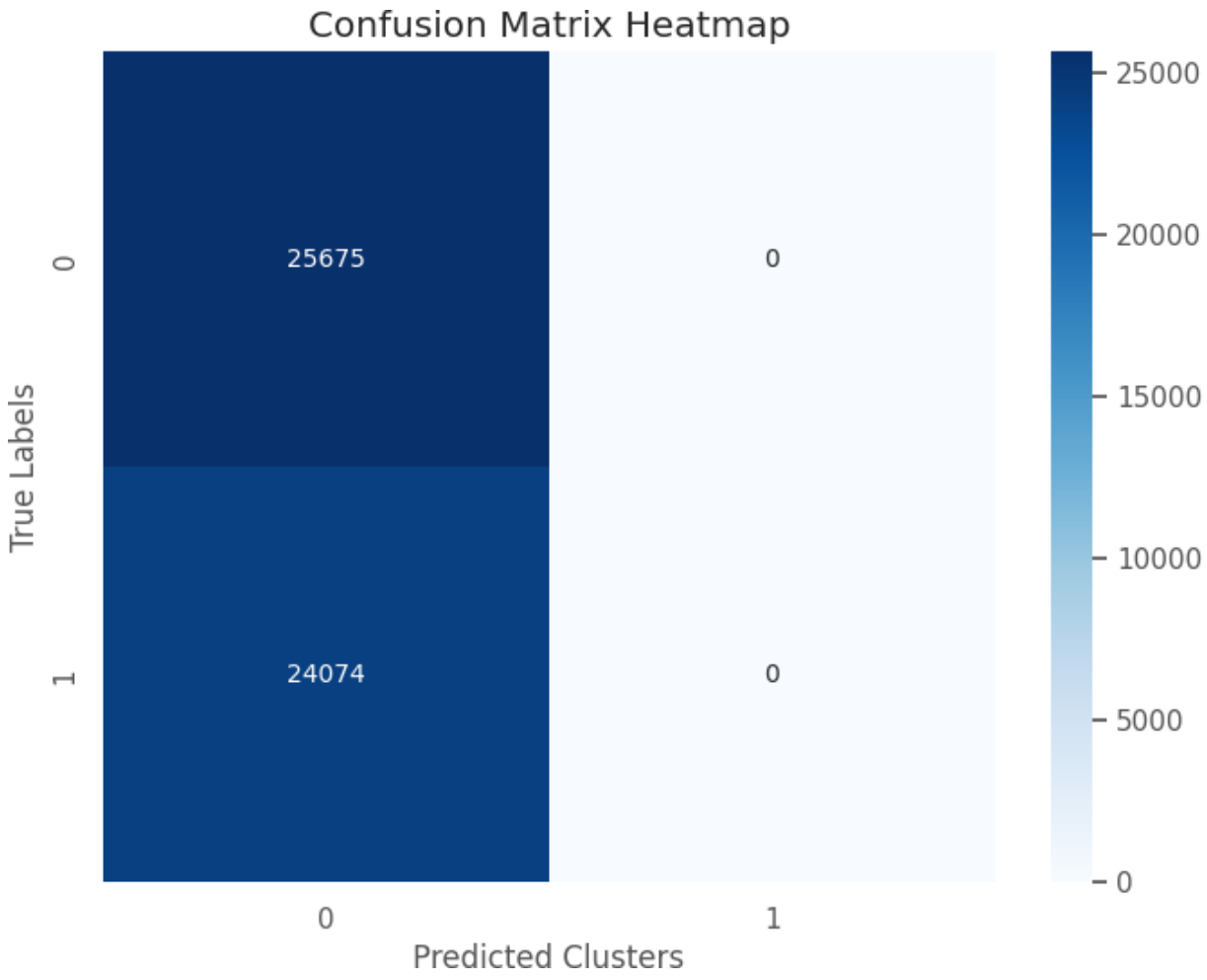
Visualization:



Evaluation Metrics:

Cluster Purity: 0.5160907756939838

Confusion matrix:

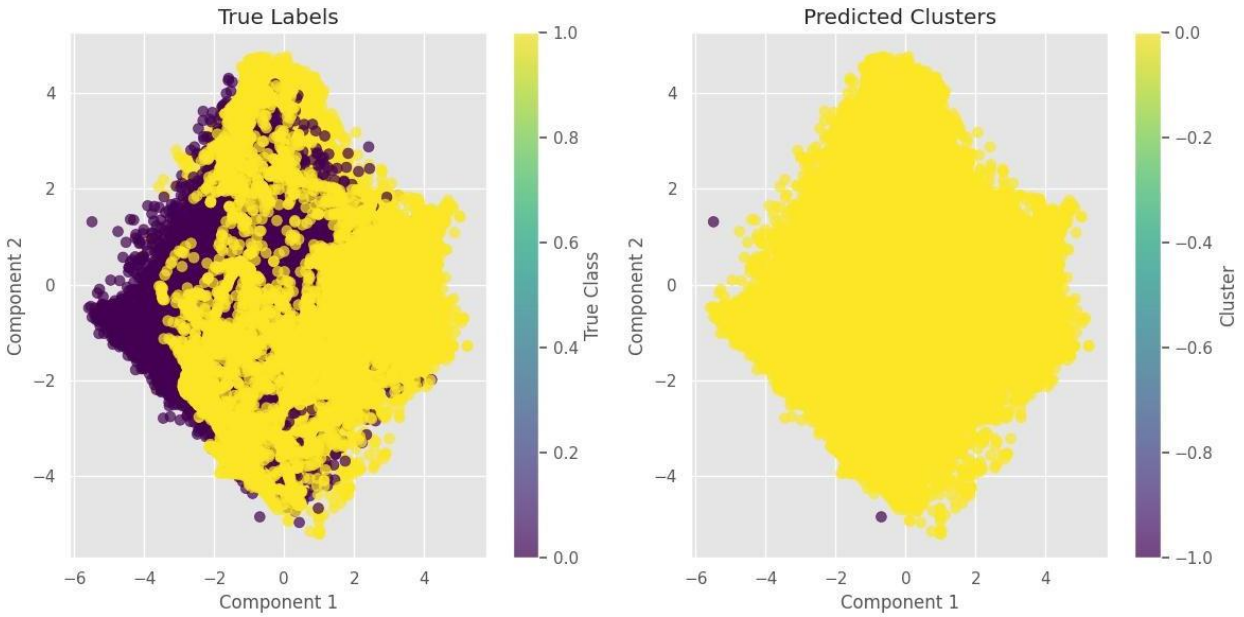


Other Metrics:

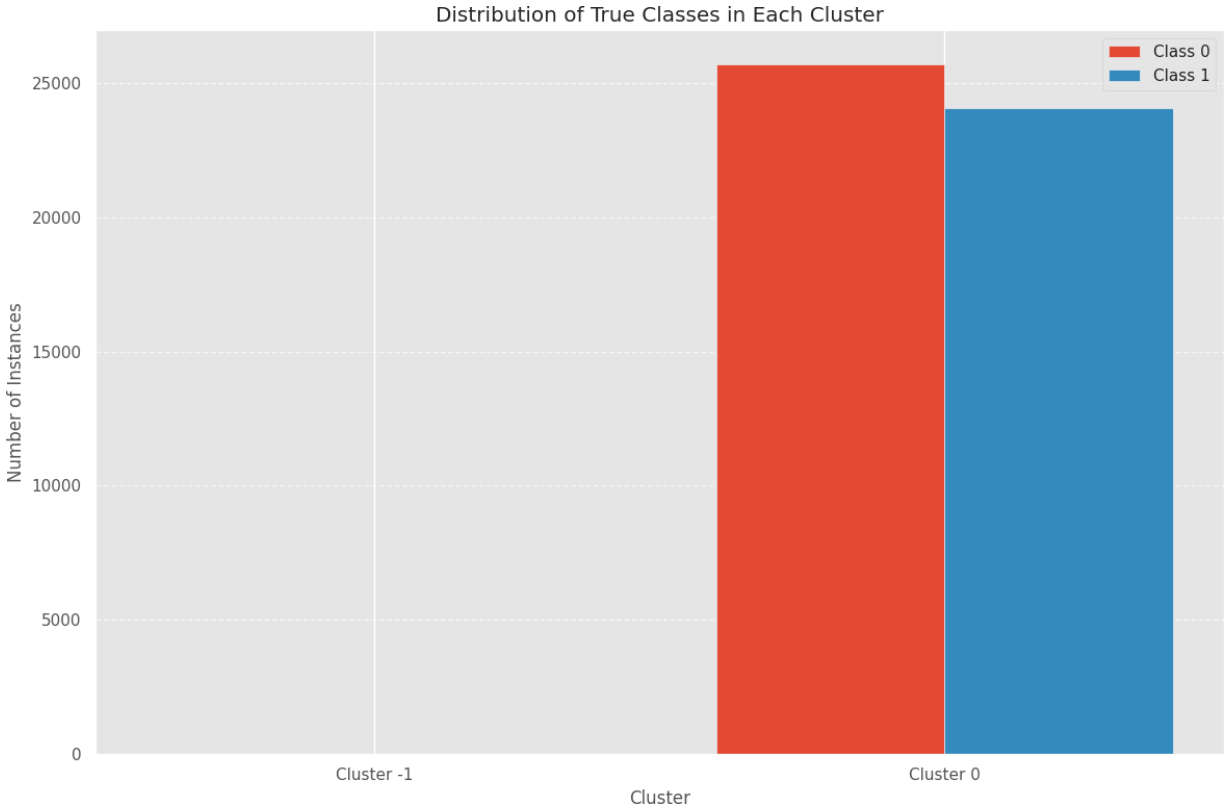
Normalized Mutual Information (NMI): 0.003746102025084575

silhouette_score: -0.05341848963104839

Comparison with true labels:



Class Distribution among clusters:

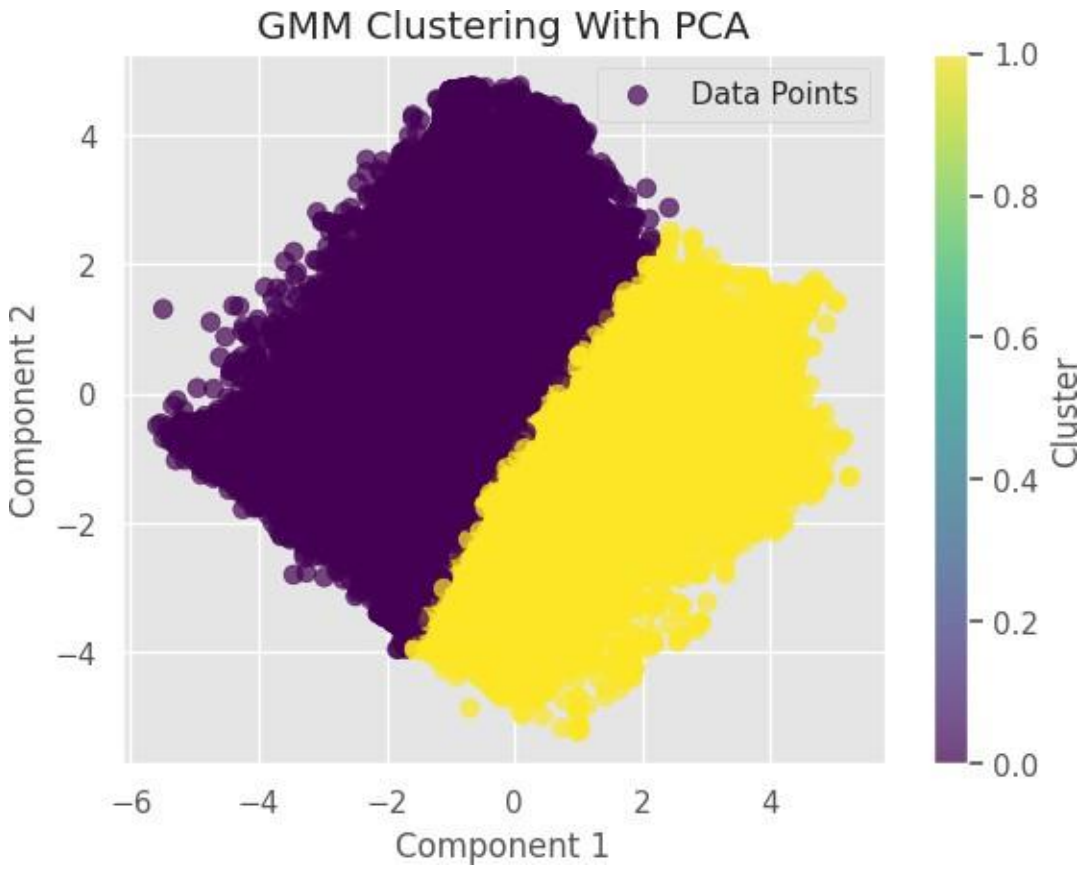


Cluster -1 has 2 datapoints with label 0 and 0 datapoints with label 1.

Cluster 0 has 25673 datapoints with label 0 and 24074 datapoints with label 1.

13.4.1.4 Gaussian Mixture Model Clustering*:*

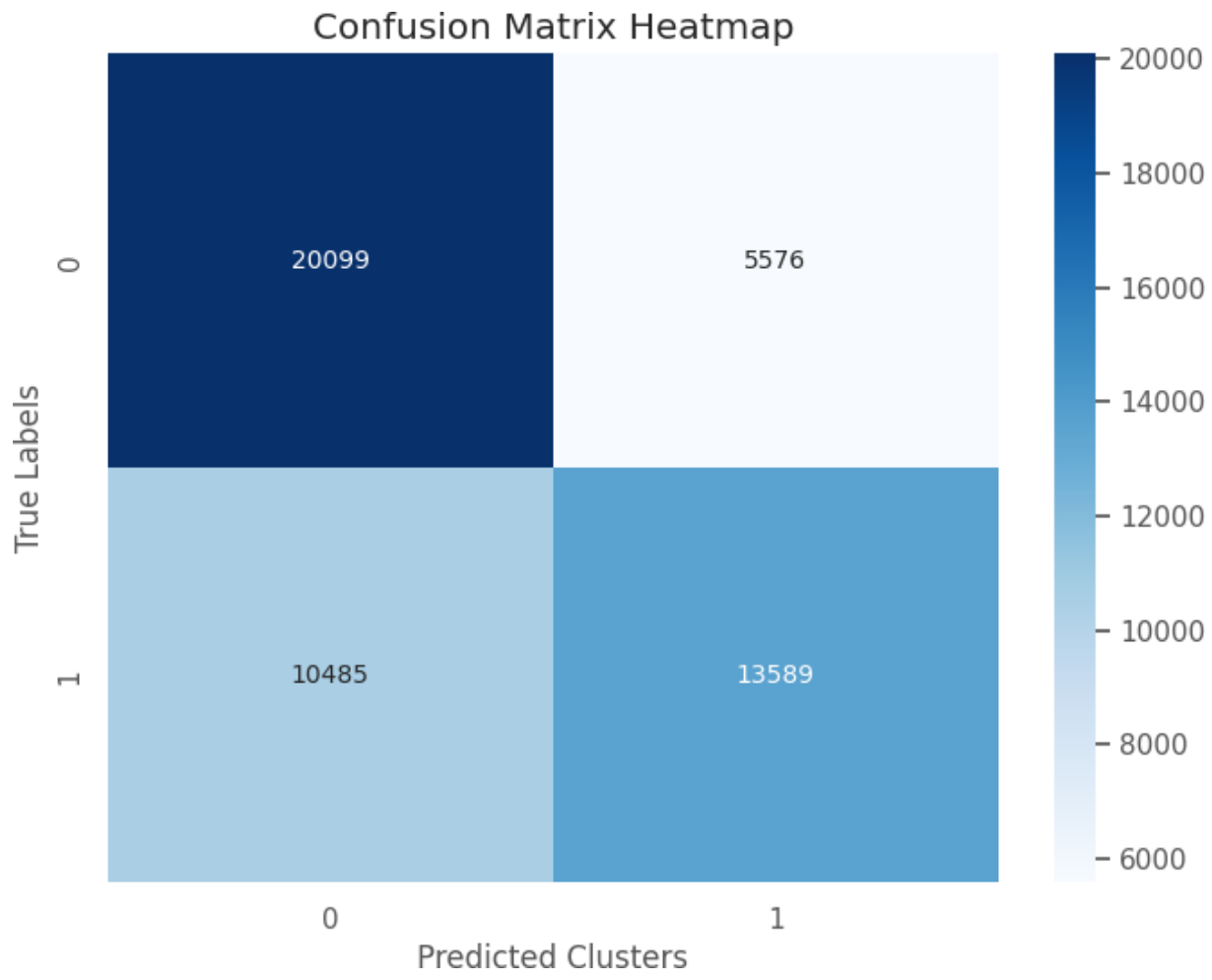
Visualization:



Evaluation Metrics:

Cluster Purity: 0.6771593398862289

Confusion matrix:

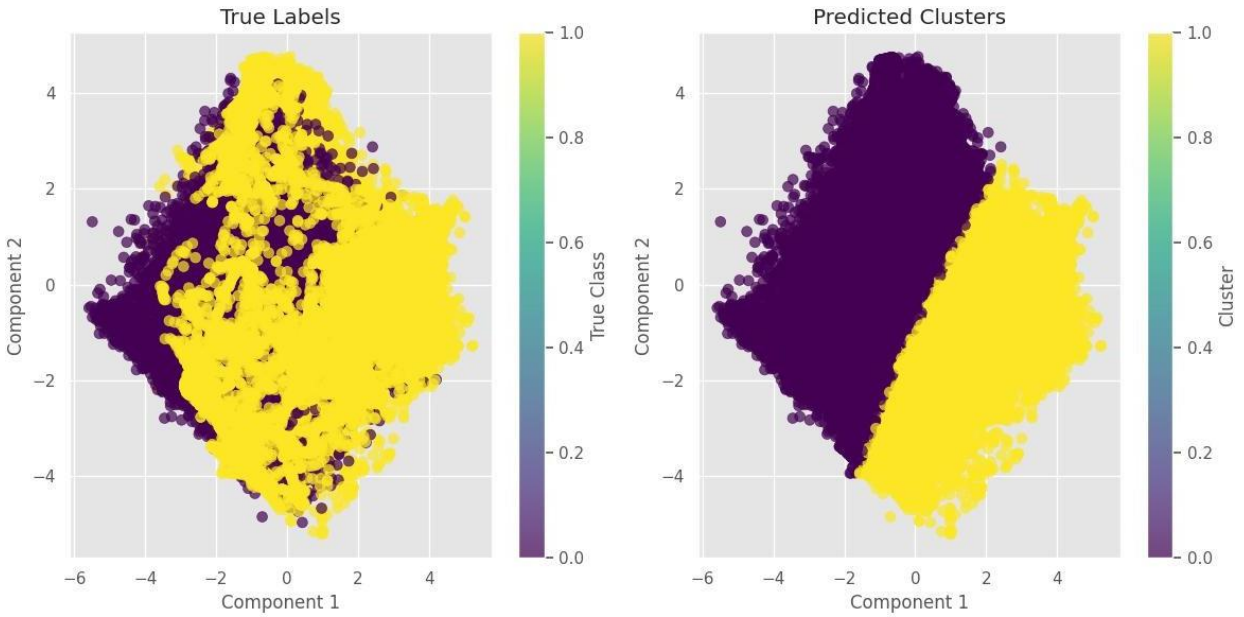


Other Metrics:

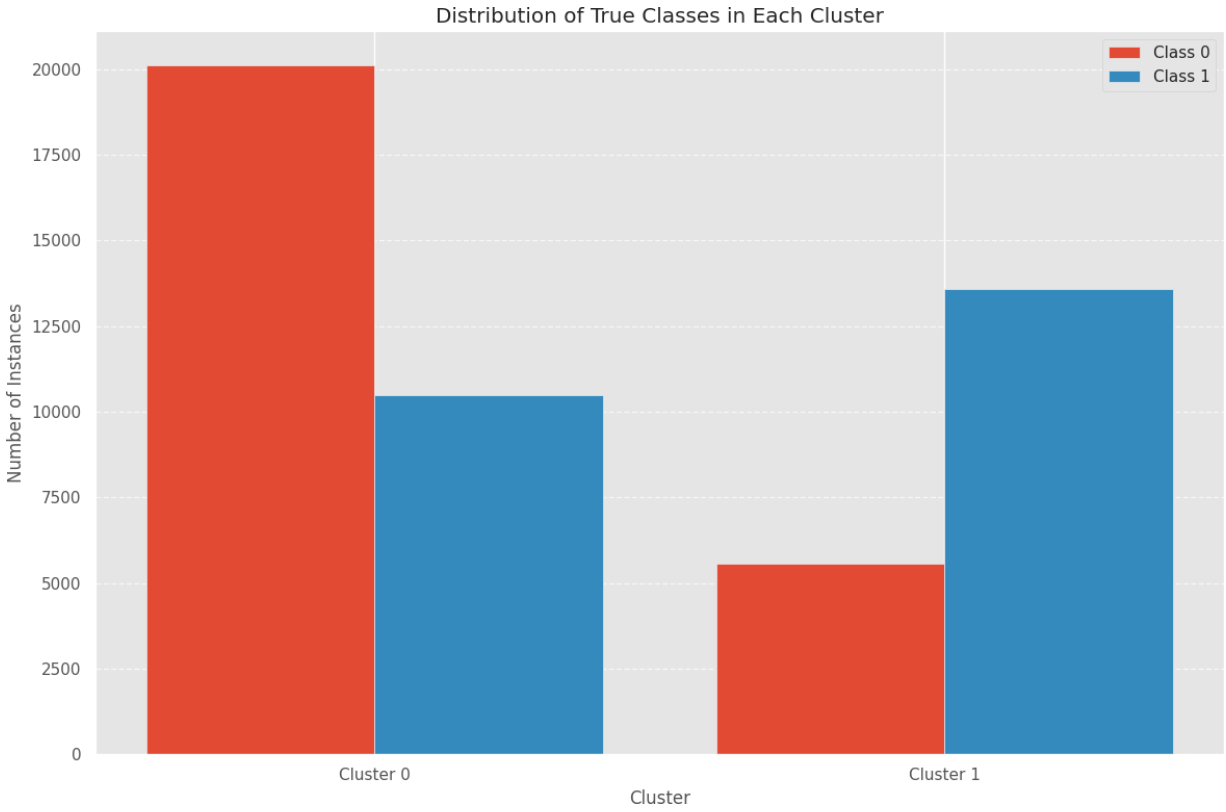
Normalized Mutual Information (NMI 0.09580028060315461

silhouette_score: 0.3689387073613973

Comparison with true labels:



Class Distribution among clusters:



Cluster 0 has 20099 datapoints with label 0 and 10485 datapoints with label 1.

Cluster 1 has 5576 datapoints with label 0 and 13589 datapoints with label 1.

13.4.1.5 Model Comparison:

Clustering Models: Performance Analysis

This comparison evaluates the clustering models applied to the dataset processed with PCA and SMOTE. Various methods were tested, and their performance metrics, including Cluster Purity, Normalized Mutual Information (NMI), Silhouette Score, and Class Distribution, were analyzed in detail.

1. K-Means Clustering

- **Elbow Method**:
  - Suggested k=3k=3, but due to the binary nature of the dataset, k=2k=2 was chosen.
- **Evaluation Metrics**:
  - **Cluster Purity**: 0.6757
  - **NMI**: 0.0930
  - **Silhouette Score**: 0.3694
- **Class Distribution**:
  - **Cluster 0**: 6002 (label 0), 14021 (label 1)
  - **Cluster 1**: 19673 (label 0), 10053 (label 1)
- **Observation**:
  K-Means demonstrated moderate clustering quality, with a fair silhouette score suggesting the clusters are distinguishable to some extent.

2. Hierarchical Clustering

a. Ward Linkage

- **Evaluation Metrics**:
  - **Cluster Purity**: 0.6062
  - **NMI**: 0.0360
  - **Silhouette Score**: 0.3241
- **Class Distribution**:
  - **Cluster 1**: 13390 (label 0), 7307 (label 1)
  - **Cluster 2**: 12285 (label 0), 16767 (label 1)

- **Observation**:
  Ward linkage showed poor performance in terms of NMI, but the silhouette score indicates moderate cluster separation.

b. Average Linkage

- **Evaluation Metrics**:
  - **Cluster Purity**: 0.6757
  - **NMI**: 0.1350
  - **Silhouette Score**: 0.3417
- **Class Distribution**:
  - **Cluster 1**: 1848 (label 0), 9788 (label 1)
  - **Cluster 2**: 23827 (label 0), 14286 (label 1)
- **Observation**:
  Slight improvement in NMI and silhouette score compared to Ward linkage. It demonstrates better separation of clusters.

c. Complete Linkage

- **Evaluation Metrics**:
  - **Cluster Purity**: 0.6942
  - **NMI**: 0.1184
  - **Silhouette Score**: 0.3643
- **Class Distribution**:
  - **Cluster 1**: 20920 (label 0), 10460 (label 1)
  - **Cluster 2**: 4755 (label 0), 13614 (label 1)
- **Observation**:
  The best among hierarchical methods, with a balance of cluster purity and silhouette score.

3. DBScan Clustering

- **Elbow Method**:
  - Suggested ϵ=0.1\epsilon = 0.1 with a minimum sample size of 4. Various configurations of ϵ\epsilon and minpts were tested.
- **Evaluation Metrics**:
  - **Cluster Purity**: 0.5161
  - **NMI**: 0.0037
  - **Silhouette Score**: -0.0534
- **Class Distribution**:
  - **Cluster -1**: 2 (label 0), 0 (label 1)
  - **Cluster 0**: 25673 (label 0), 24074 (label 1)

- **Observation**:
  DBScan performed poorly with a negative silhouette score, indicating overlapping or poorly separated clusters.

Summary Table

Clustering Method	Cluster Purity	NMI	Silhouette Score	Best Cluster (Label 1)	Best Cluster (Label 0)
K-Means	0.6757	0.0930	0.3694	14021	19673
Ward Linkage	0.6062	0.0360	0.3241	16767	13390
Average Linkage	0.6757	0.1350	0.3417	14286	23827
Complete Linkage	0.6942	0.1184	0.3643	13614	20920
DBScan	0.5161	0.0037	-0.0534	24074	25673

Key Insights

1. **Best Model**:
   - Complete Linkage Hierarchical Clustering achieved the highest cluster purity and a decent silhouette score.
2. **Worst Model**:
   - DBScan struggled to identify meaningful clusters due to its reliance on density parameters.
3. **General Performance**:
   - K-Means and Hierarchical methods (Average and Complete Linkage) outperformed DBScan, particularly in terms of silhouette score and NMI.

Conclusion: Unsupervised Learning

Unsupervised learning is a powerful tool for analyzing and understanding datasets without explicit labels. Through clustering techniques like K-Means, Hierarchical Clustering, DBScan, and Gaussian Mixture Models, we can uncover patterns and groupings that provide valuable insights into the data structure.

Key evaluation metrics, such as **Cluster Purity**, **Normalized Mutual Information (NMI)**, and **Silhouette Score**, help assess the quality of the clusters, each offering a unique perspective. While **Cluster Purity** focuses on the homogeneity of clusters with respect to true labels, **NMI**

evaluates the overlap between clusters and actual classes, and **Silhouette Score** measures the separation and cohesion of clusters.

However, the effectiveness of these methods depends on the nature of the data, parameter tuning, and the assumptions inherent in each algorithm. For instance:

- K-Means works well with spherical clusters but struggles with non-linear shapes.
- Hierarchical Clustering offers flexibility but can be computationally intensive for large datasets.
- DBScan excels at detecting arbitrarily shaped clusters but requires careful parameter selection.
- Gaussian Mixture Models provide probabilistic cluster assignments but assume Gaussian distributions.

The unsupervised learning approach is indispensable when labeled data is scarce or unavailable. While it cannot directly classify or predict outcomes, it lays the foundation for exploratory data analysis, feature engineering, and even supervised learning pipelines. By leveraging unsupervised methods, we can gain deeper insights into complex data and improve decision-making processes across various domains.