

# Evaluation of various Classification Algorithms for Dry Beans Dataset

EE 559 Course Project

Data Set: Dry Beans classification dataset.

Rakshika Rajakumar, rrajakum@usc.edu

## 1. Abstract

*The agricultural sector stands to benefit greatly from the application of machine learning techniques in classification tasks. The project explores the application of machine learning techniques for classifying dry beans using a dataset that includes geometric and shape-based features and a categorical label for different bean classes. The goal is to categorize the beans into one of seven types: Barbunya, Bombay Kidney Beans, Cali Beans, Dermason Kidney Beans, Horoz Beans, Seker Beans, and Sira Beans. A variety of supervised learning methods were employed, including the nearest means classifier, artificial neural networks, Gradient Boosting Classifier, XGBoost Classifier, and Support Vector Classifier. The study involved data preprocessing and a comparative analysis of model performance based on accuracy and F1 scores. The XGBoost Classifier achieved the highest accuracy at 94.09%, making it the most effective classification technique. This research demonstrates the potential of machine learning in agricultural data classification and provides a foundation for further advancements in this domain.*

## 2. Introduction

### 2.1. Problem Statement and Goals

Classifying dry beans into seven categories is crucial for seed quality and crop yields; manual sorting is time-consuming and error-prone, so an efficient automated system using machine learning is needed.

The dataset used for this project contains 10,899 entries and 16 features, including "Area," "Perimeter," "Major Axis Length," and other geometric and shape-based metrics. The goal is to accurately classify dry beans into seven types: Barbunya, Bombay Kidney Beans, Cali Beans, Dermason Kidney Beans, Horoz Beans, Seker Beans, and Sira Beans. By leveraging feature visualization, extraction, and comparative use of classification algorithms, this project aims to put forth a precise

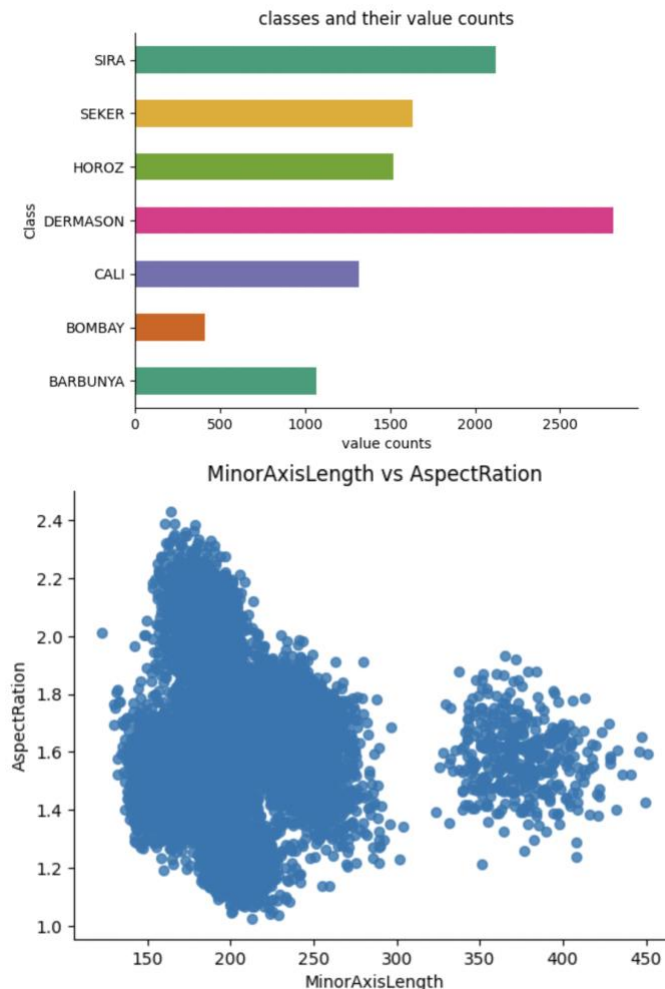
and efficient classification model that enhances productivity and quality control in the agricultural sector.

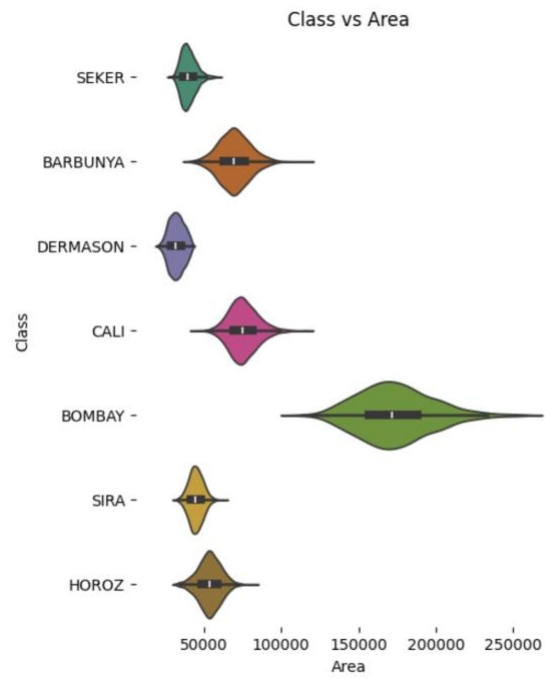
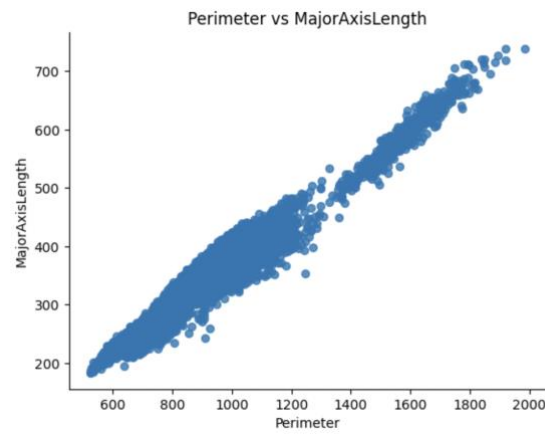
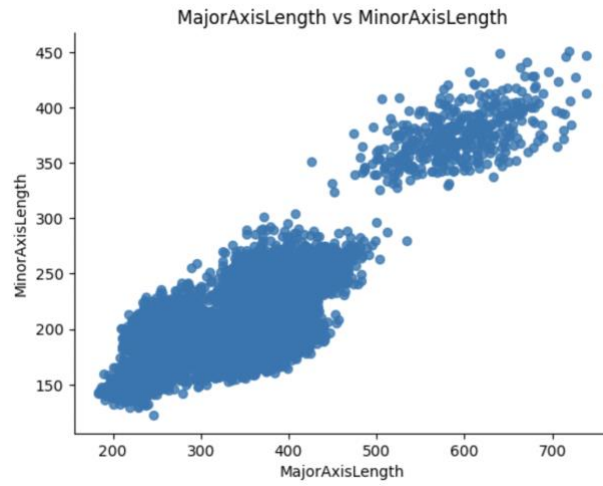
### 3. Approach and Implementation

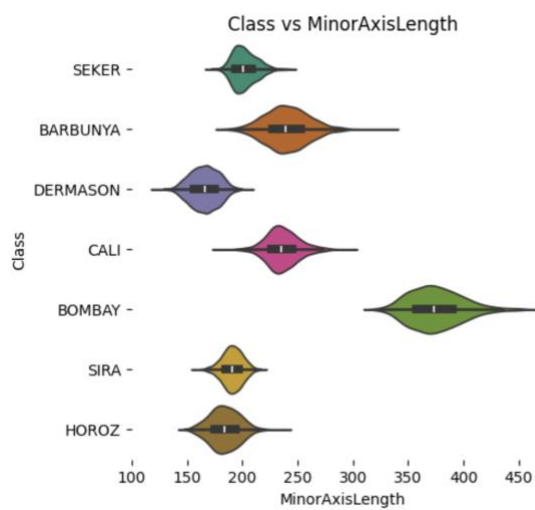
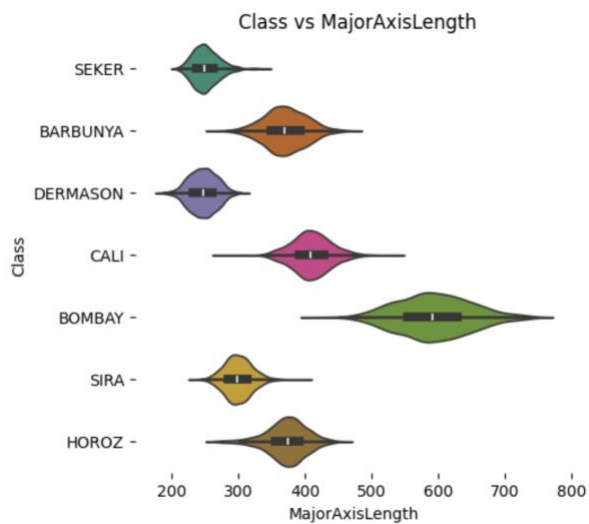
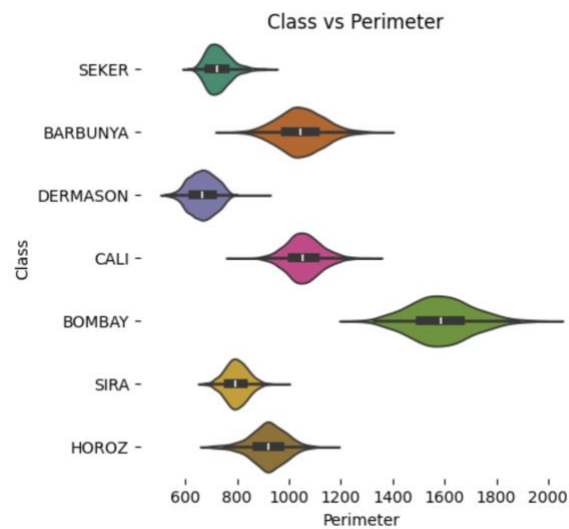
#### 3.1. Dataset Usage

Our project made use of an extensive dataset that included features obtained from a computer vision system designed to differentiate between seven distinct varieties of dry beans with similar characteristics. This approach aimed to achieve standardized seed classification. For our classification model, images of 10,889 grains representing seven different registered dry bean varieties were captured using a high-resolution camera.

These features encompassed a broad range of attributes extracted from the images, including geometric measurements, texture details, and pixel intensity distributions. The well-structured format of the dataset enabled smooth integration into our machine learning pipeline, supporting efficient preprocessing and feature engineering processes. This allowed us to leverage advanced techniques to maximize the accuracy and efficiency of the classification model.







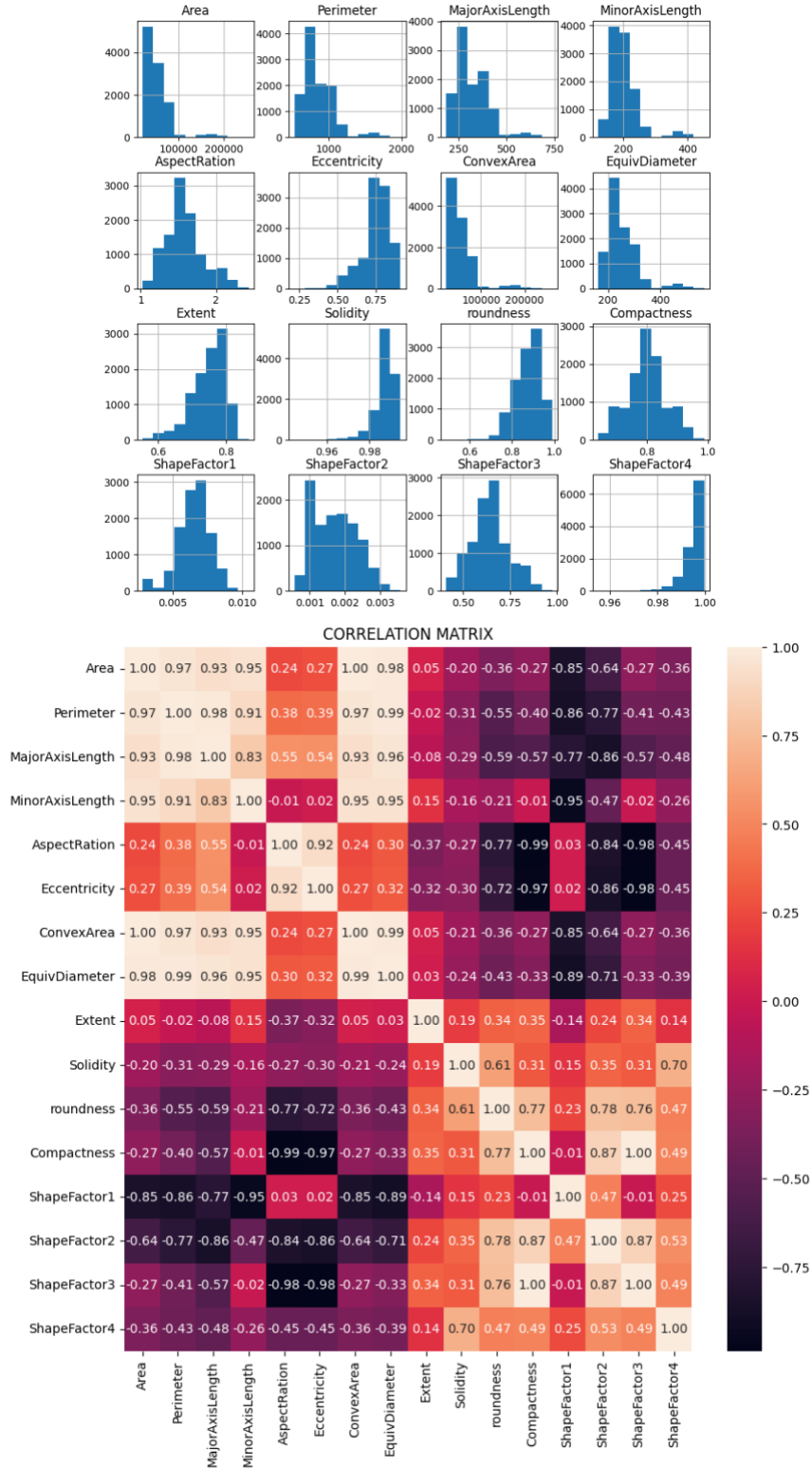


Fig 1. Descriptive Statistics of the dataset

### 3.2. Preprocessing

The dataset was preprocessed using various techniques to prepare the data for machine learning models. The preprocessing steps included handling missing values, removing duplicate entries, encoding categorical variables, and scaling numerical features.

1. **Handling Missing Values:** Missing values in the dataset were handled using different strategies, such as mean, median, mode, and constant value imputation. This approach ensures that there are no missing values in the data that could adversely affect model training.
2. **Duplicate Removal:** The dataset was checked for duplicate rows. Identified duplicates were removed to maintain data integrity and avoid biases during model training.
3. **Encoding Categorical Variables:** The categorical label column 'Class' was encoded using label encoding. This method transforms the non-numeric labels into integer values, making them suitable for model training.
4. **Scaling Numerical Features:** Numerical features were scaled using a MinMaxScaler, which normalized the values between 0 and 1. This step helps improve model performance and convergence.
5. **Addressing Class Imbalance:** SMOTE (Synthetic Minority Oversampling Technique) was applied to balance the classes in the dataset, preventing bias towards majority classes and providing equal representation of all classes.
6. **Noise Management:** Gaussian noise was introduced to the cleaned data at a noise level of 0.01 to simulate real-world conditions and enhance model robustness.

Overall, these preprocessing steps optimized the dataset for machine learning, ensuring models could effectively learn from the data and produce reliable predictions.

### 3.3. Feature Engineering

Feature engineering was performed to identify and select the most relevant features for the classification task.

- **Feature Selection:** Using correlation analysis, the most impactful features on the 'Class' label were identified. This involved calculating the correlation matrix and selecting the top features with the highest correlation to the class label.
- **Refining Feature Set:** The feature set was refined to include only the top features identified through correlation analysis, resulting in a subset of features with the most significant impact on class prediction.

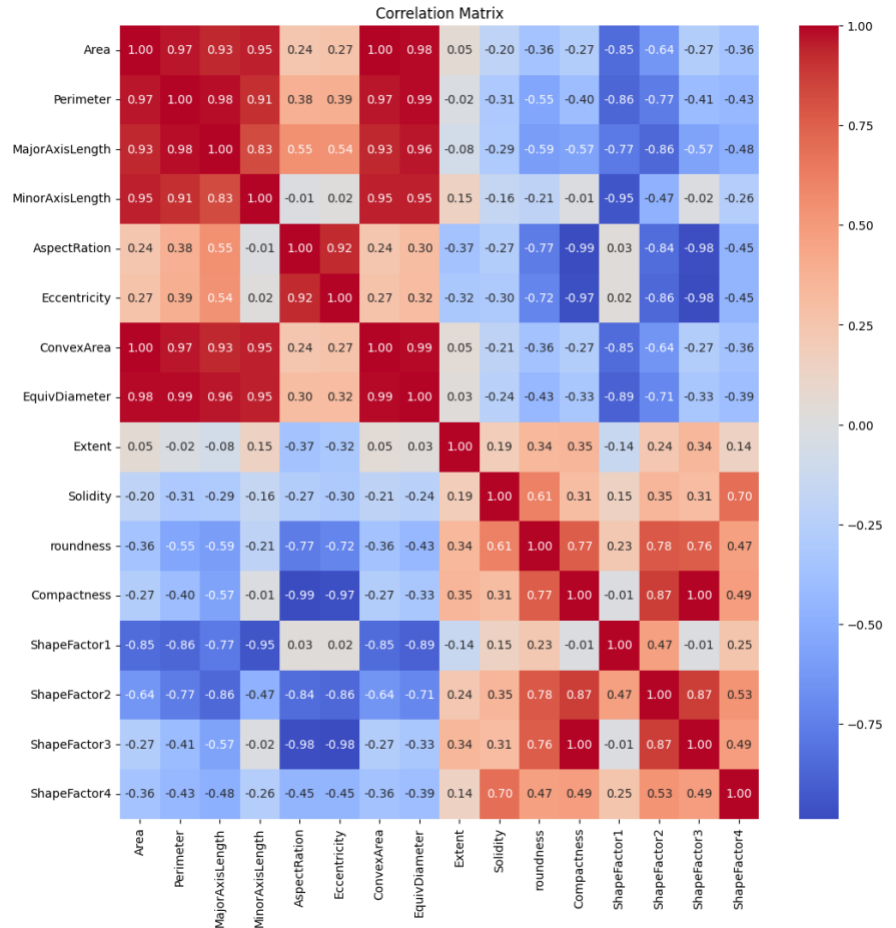


Fig. 3 Correlation Matrix

### 3.4. Feature dimensionality adjustment

To optimize model performance, the dimensionality of the dataset was adjusted as needed.

Dimensionality Reduction: Redundant or less informative features were removed to reduce data dimensionality. This was achieved through correlation analysis, selecting only the most relevant features for the final feature set. This approach helped reduce computational complexity and improved model performance by focusing on the most impactful features. Overall, the combination of feature engineering and dimensionality adjustment ensured that the models were trained on an efficient and optimized subset of the data, leading to better performance and generalization. Nasution, M. Z. F., Sitompul, O. S., & Ramli, M [4] gives information about using pca based feature reduction for better model performance.

### 3.5. Training, Classification, and model selection

In this project, multiple machine learning models were trained and evaluated to classify dry bean data into seven distinct classes. The aim was to identify

the model that provides the highest accuracy and lowest error rates on the test set. The models trained include a neural network classifier, nearest centroid classifier, gradient boosting classifier, and XGBoost classifier. Additionally, a trivial solution model was used as a baseline for comparison.

#### 3.5.1. Trivial Solution – Probabilistic classifier

A trivial solution was implemented as an acute approach to provide a comparison point for more sophisticated models. This approach involves predicting classes randomly based on class probabilities in the training data. The class probabilities are calculated based on the frequency of each class in the training set. There was no tuning of model parameters as the approach relies on random class assignments. Model evaluation included calculating accuracy, F1 scores (macro and micro), and generating a confusion matrix to assess the model's ability to classify samples across different classes. This baseline method serves as a benchmark to gauge the performance of more complex models in comparison.

#### 3.5.2. Nearest means classifier

The Nearest Centroid classifier was used to classify samples based on the nearest centroid (mean) of the feature values of each class. The model was trained on the training set ( $x_{train}$ ,  $y_{train}$ ). Evaluation metrics included accuracy, F1 scores (macro and micro), and a confusion matrix to assess performance. Cross-validation was conducted using multiple folds to measure the model's generalization performance and reliability.

#### 3.5.3. Gradient Boosting Classifier

A Gradient Boosting Classifier (GBC) was implemented for classification. This algorithm sequentially trains weak models to create a strong predictive model. The GBC model was trained on the training set ( $x_{train}$ ,  $y_{train}$ ) with a random state of 0. The evaluation metrics included accuracy, F1 scores (macro and micro), and a confusion matrix to assess the model's performance. Cross-validation was performed to evaluate the model's generalization across different data folds.

#### 3.5.4. Artificial Neural Network (ANN)

The neural network classifier used a Keras Sequential model with three layers for classification. The input layer contained 32 neurons with a ReLU activation function. The hidden layer also had 32 neurons with a ReLU activation function. The output layer consisted of 7 neurons with a softmax activation function to predict the seven classes. The model was trained with the Adam optimizer, using a loss function of categorical cross-entropy, for 50 epochs with a batch size of 32 and a validation split of 0.1. Model evaluation included calculating loss and accuracy on the test set, predicting the classes,



and calculating F1 scores (macro and micro). A confusion matrix was generated to assess the model's performance across different classes. Cross-validation was conducted using multiple folds to evaluate the model's generalization.

#### 3.5.5. XGBoost Classifier

The XGBoost Classifier was used for classification due to its high performance and efficiency. It is a gradient boosting model. Abdurrahman, G., & Sintawati, M [3] explain the use of XGB in numerical data classification of parkinson's disease. The XGBoost model was trained on the training set (x\_train, y\_train) with a random state of 0 and evaluation metric mlogloss. The model used early stopping to prevent overfitting. Model evaluation included accuracy, F1 scores (macro and micro), and a confusion matrix. Cross-validation was performed using multiple folds to assess model generalization.

#### 3.5.6. Support Vector Classifier (SVC)

The Support Vector Machine (SVM) classifier was implemented using the Scikit-learn SVC class with a linear kernel. The linear kernel was chosen to classify the data using a hyperplane that maximizes the margin between classes. The regularization parameter (C) was set to 1.0, balancing the trade-off between model complexity and classification accuracy. The random state was set to 42 for reproducibility.

The SVM model was trained on the training set (x\_train, y\_train). After training, the model predicted the classes on the test set (x\_test). The model's performance was evaluated using accuracy, macro-averaged and micro-averaged F1 scores, and a confusion matrix to visualize the classifier's performance across different classes.

Cross-validation was conducted using multiple folds to assess the model's generalization performance. The average cross-validation score provided a summary of the model's overall performance during training. This evaluation of the SVM classifier helped guide the model selection process.

## 4. Results and Analysis: Comparison and Interpretation

### 4.1. Reference Models

We have modelled two reference models, A Nearest mean Classifier and A Random Class Assigner.

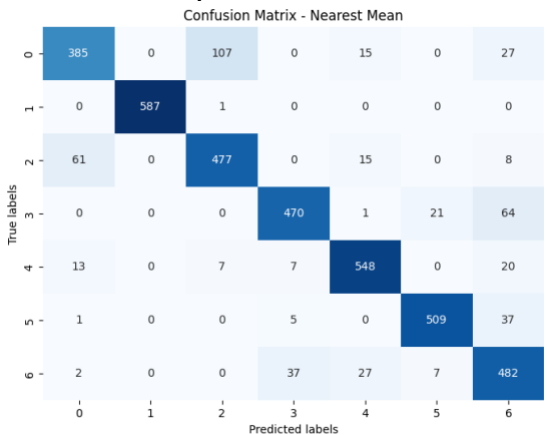
#### 4.1.1. Nearest Mean Classifier

The Nearest Means Classifier, also known as the Nearest Centroid classifier, was used as a baseline model for classification. This classifier assigns each

sample to the class with the nearest mean feature vector, using Euclidean distance as a measure.

The Nearest Means Classifier achieved a high accuracy of 87.74%, demonstrating its effectiveness in classifying the data. Both macro-averaged and micro-averaged F1 scores were approximately 0.876, indicating strong performance across different classes.

The classifier's solid performance suggests that the Nearest Means approach is a viable option for classification tasks and provides a strong baseline for comparison with other more complex models.

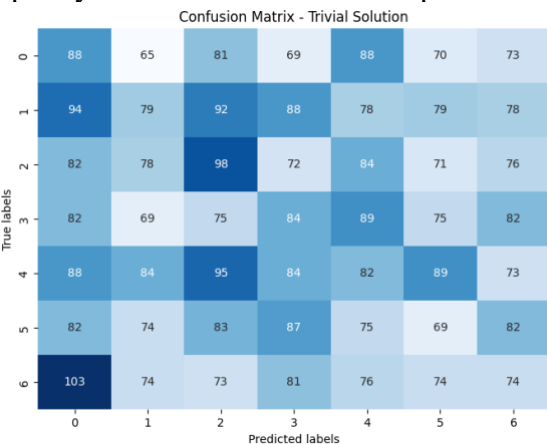


4.1.2. Trivial Solution

The Trivial Solution, or probability-based classifier, was implemented as a baseline approach. The classifier assigned classes to samples randomly based on the probabilities of each class in the training data.

The Trivial Solution achieved an accuracy of 14.56%, which is a low score indicative of the classifier's random assignment strategy. Both macro-averaged and micro-averaged F1 scores were similarly low, at around 0.1454, reflecting the classifier's limited ability to classify samples correctly.

These metrics demonstrate the Trivial Solution's poor performance, serving as a benchmark for comparing more sophisticated classification algorithms. This low baseline emphasizes the need for advanced models to improve accuracy and classification quality. The confusion matrix is presented below:



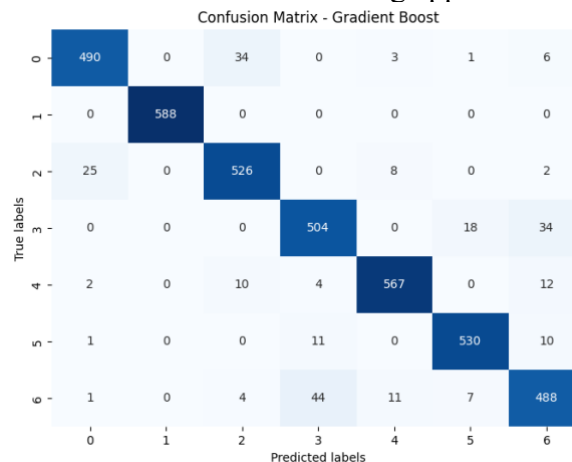
#### 4.1.3. Gradient Boosting Classifier

The Gradient Boosting Classifier (GBC) is a powerful ensemble learning method that builds a series of decision trees sequentially. Each tree is trained to correct the errors of the previous one, leading to a strong model with high predictive accuracy. GBC minimizes a specified loss function, typically mean squared error or log loss, depending on the task.

The Gradient Boosting Classifier achieved an impressive accuracy of 93.71%, indicating its exceptional performance in classifying the data. Both macro-averaged and micro-averaged F1 scores were approximately 0.936, demonstrating the classifier's ability to perform well across different classes. The classifier's cross-validation scores ranged from 93.47% to 94.07%, with an average of 93.82%. This consistency across different folds highlights the model's robustness and generalization capability.

Overall, the Gradient Boosting Classifier proved to be an effective and reliable choice for the classification task, offering high accuracy and consistent performance across different data splits. Its ability to capture complex relationships in the data and adapt to different loss functions makes it a valuable method for various machine learning applications.

Confusion Matrix - Gradient Boost



	0	1	2	3	4	5	6
0	490	0	34	0	3	1	6
1	0	588	0	0	0	0	0
2	25	0	526	0	8	0	2
3	0	0	0	504	0	18	34
4	2	0	10	4	567	0	12
5	1	0	0	11	0	530	10
6	1	0	4	44	11	7	488
	0	1	2	3	4	5	6

#### 4.1.4. Artificial Neural Network Classifier

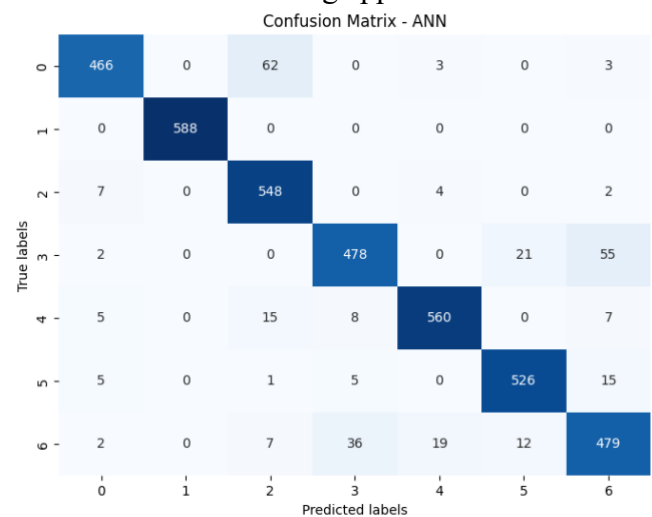
The Neural Network Classifier was trained using Keras with a Sequential model consisting of three layers: an input layer with 32 neurons and a ReLU activation function, a hidden layer with 32 neurons and a ReLU activation function, and an output layer with 7 neurons and a softmax activation function to classify the data into seven classes.

The training process consisted of 50 epochs with a batch size of 32 using the Adam optimizer and categorical cross-entropy loss. As training progressed, the loss decreased and accuracy increased, indicating effective learning.

The final test loss was 0.2094, and the test accuracy reached 92.49%, demonstrating the model's ability to classify data with high accuracy. Both macro-averaged and micro-averaged F1 scores were approximately 0.924, reflecting strong classification performance across different classes.

The model's cross-validation scores ranged from 91.52% to 92.49%, with an average score of 92.07%. This consistency across folds highlights the model's robustness and generalization capability.

Overall, the Neural Network Classifier proved to be a reliable and effective choice for classification tasks, offering high accuracy and consistent performance across different data splits. Its ability to capture complex relationships in the data and adapt to various classification scenarios makes it a valuable method for machine learning applications.



#### 4.1.5. XGBoost Classifier

The XGBoost Classifier is an advanced gradient boosting model known for its performance and scalability. It combines an ensemble of decision trees that are trained in a stage-wise manner, each tree correcting the errors of the previous one. The algorithm uses advanced regularization techniques (L1 and L2) to avoid overfitting and optimize model complexity.

The XGBoost Classifier achieved an outstanding accuracy of 94.09%, demonstrating its high effectiveness in classifying data. Both macro-averaged and micro-averaged F1 scores were approximately 0.9404, indicating strong performance across different classes and high overall classification accuracy. Cross-validation scores ranged from 94.21% to 94.77%, with an average score of 94.46%. This consistency across different data folds highlights the model's robustness and generalization capability, proving its reliability in various classification scenarios.

The XGBoost Classifier's strengths lie in its ability to handle large datasets efficiently, automatically handle missing values, and provide feature importance scores for better interpretability. Its excellent performance and scalability make it a top choice for a variety of machine learning tasks.

Confusion Matrix - XGB

0	502	1	24	0	2	0	5
1	0	588	0	0	0	0	0
2	17	0	530	0	11	0	3
3	0	0	0	505	1	14	36
4	4	0	10	4	565	0	12
5	0	0	0	15	0	526	11
6	1	0	3	43	10	6	492
	0	1	2	3	4	5	6

True labels

Predicted labels

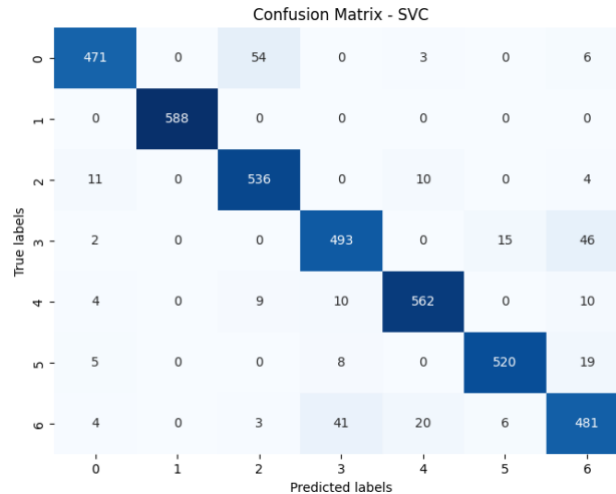
#### 4.1.6. Support Vector Classifier (SVC)

The Support Vector Classifier (SVC) is a powerful and versatile classification model that seeks to find the optimal hyperplane that maximizes the margin between classes in the feature space. This implementation of SVC used a linear kernel, making it suitable for linearly separable data.

The Support Vector Classifier achieved an accuracy of 92.64%, demonstrating its strong performance in classifying the data. Both macro-averaged and micro-averaged F1 scores were approximately 0.9256 and 0.9264, respectively, indicating that the model performed well across different classes and overall.

The classifier's cross-validation scores ranged from 92.42% to 93.53%, with an average score of 92.90%. This consistency across different data splits highlights the model's robustness and generalization capability, proving its reliability in various classification scenarios.

The SVC's ability to find the optimal decision boundary and classify data accurately makes it a strong choice for machine learning tasks. This classifier's performance demonstrates its effectiveness and reliability for a wide range of applications.



#### 4.2. Model training.

Based on our evaluation of multiple models, we selected the best-performing one using metrics such as accuracy and F1 scores. Below is a summary of the performance metrics for each model.

The Gradient Boosting Classifier and the XGBoost Classifier achieved top results, with accuracy scores of 93.71% and 94.09%, respectively. The macro-averaged F1 scores were 0.9364 for the Gradient Boosting Classifier and 0.9404 for the XGBoost Classifier, reflecting their effectiveness across classes.

The Support Vector Classifier also demonstrated strong performance with an accuracy of 92.64% and a macro-averaged F1 score of 0.9256. Additionally, the Nearest Means Classifier showed solid results, with an accuracy of 87.74% and a macro-averaged F1 score of 0.8756.

The trivial solution served as a baseline, achieving an accuracy of 14.34% and macro-averaged and micro-averaged F1 scores of 0.14.

Cross-validation of the models indicated that the Support Vector Classifier had the lowest standard deviation, suggesting consistent performance across different data splits.

In summary, while the trivial solution provided a baseline, the advanced models, such as Gradient Boosting and XGBoost, offered significantly higher accuracy and F1 scores. The Support Vector Classifier also showed reliability and consistency in performance.

Models	Accuracy	F1- score		Cross-validation accuracy
		Macro	Micro	
Nearest Mean Classifier	0.8774	0.8756	0.8774	0.8889

Trivial Solution	0.1456	0.1454	0.1456	-
SVC	0.9264	0.9256	0.9264	0.9289
ANN	0.9248	0.9238	0.9248	0.9303
Gradient Boost	0.9370	0.9363	0.9370	0.9382
XGB	0.9408	0.9404	0.9408	0.9445

Table. Performance analysis of models

#### 4.3. Model Testing

In this section, we evaluate the performance of the trained models on a separate test dataset to assess their predictive capabilities and generalization to unseen data. The test dataset, consisting of 2722 samples, provides an independent sample for evaluating the models' performance and verifying their effectiveness in real-world scenarios.

Data Preparation: The test dataset was loaded and preprocessed, including normalization and label encoding, to match the format used during training.

Model Loading: The trained models were loaded from their saved files and prepared for evaluation.

Model Testing: For the XGBoost Classifier, which was identified as the best-performing model during training, we applied it to the test dataset to make predictions on the target variable. The performance of the model was evaluated using various metrics, including accuracy, F1 score, and a confusion matrix.

- Accuracy: The model achieved an accuracy score of 86.11% on the test set, indicating strong performance in classifying the data.
- F1 Score: The macro-averaged F1 score was 0.8720, reflecting balanced classification across different classes, while the micro-averaged F1 score was .8611, mirroring the overall accuracy.
- Confusion Matrix: The confusion matrix highlights the model's classification performance across different classes, providing insights into its strengths and weaknesses. These evaluation metrics confirm the XGBoost Classifier's effectiveness and robustness in real-world scenarios, demonstrating its generalization capabilities and strong predictive performance on the test data. Let me know if you need further information.

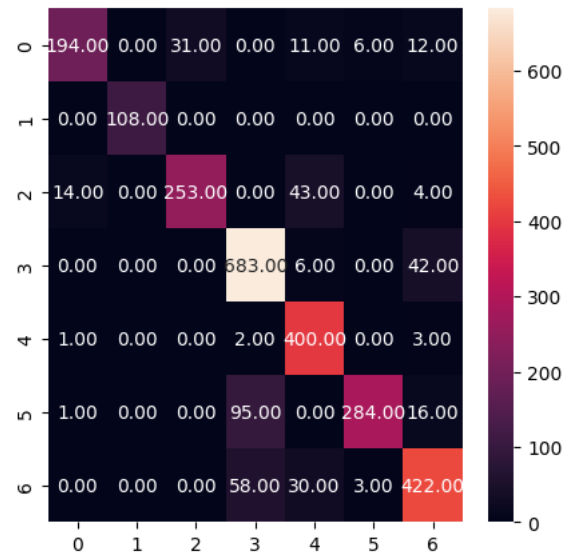


Fig. Confusion Matrix of XGBoost classifier on Test data

## 5. Libraries used and Contribution

The project utilized several libraries for data preprocessing, model training, and evaluation, as detailed below:

Libraries Used:

- Pandas: Used for data manipulation, cleaning, and preprocessing, including loading data, handling missing values, normalizing data, and encoding categorical features.
- NumPy: Employed for numerical operations and array manipulation, including generating synthetic data and performing mathematical computations.
- Matplotlib and Seaborn: Utilized for data visualization, including plotting confusion matrices and other charts to visualize data distributions and model performance.
- Scikit-learn: Employed for various machine learning tasks, including training and evaluating models such as Nearest Centroid, Support Vector Machine, and Gradient Boosting Classifier, as well as methods such as cross-validation and feature selection.
- Imbalanced-learn (SMOTE): Used to handle imbalanced datasets through oversampling of minority classes, thus balancing the class distribution.
- XGBoost: Used to train the XGBoost Classifier, a highly efficient gradient boosting model widely used for classification tasks.
- TensorFlow and Keras: Utilized for building and training neural network models, specifying layers, activation functions, and other neural network parameters.

**Custom Implementations:**



- **Data Preparation and Preprocessing:** Custom functions were implemented for data normalization using `MinMaxScaler`, label encoding, and handling missing values using `Pandas` and `Scikit-learn`. `SMOTE` was used to address class imbalance.
- **Model Training:** Custom code was written to load data, prepare it, and train models such as `Nearest Centroid`, `SVM`, `Gradient Boosting Classifier`, and `XGBoost`. Hyperparameters were adjusted to optimize model performance.
- **Model Evaluation:** Custom evaluation methods were implemented to calculate accuracy, macro-averaged and micro-averaged F1 scores, and generate confusion matrices. These assessments were essential for comparing model performance and identifying the most effective model.
- **Cross-Validation:** Custom code leveraging `Scikit-learn`'s `cross_val_score` function was used for cross-validation to evaluate the generalization performance of different models.

These libraries and custom implementations created a robust pipeline for data preparation, model training, and evaluation, allowing for a comprehensive assessment of model performance and generalization capabilities.

## 6. Summary and Conclusion

In this project, we tackled the task of classifying commercial dry beans into seven categories using sophisticated machine learning methods. Manual sorting is time-consuming and subject to mistakes, underscoring the need for an automated solution. Our strategy involved a detailed dataset of features derived from dry bean samples, emphasizing data preprocessing, feature engineering, and model training.

We applied various machine learning models to classify dry bean data and evaluated their performance on unseen test data. We started with comprehensive data preparation, which included handling missing values, encoding labels, and normalizing the data using `MinMaxScaler`. This preprocessing ensured that the data was in a suitable format for machine learning tasks.

For model training, we explored a variety of algorithms, including `Nearest Centroid`, `Support Vector Machine (SVM)`, `Gradient Boosting Classifier`, and `XGBoost Classifier`. These models were chosen for their potential to perform well on classification tasks and for their relevance to the given problem.

Model evaluation was carried out using metrics such as accuracy, macro-averaged and micro-averaged F1 scores, and confusion matrices to gauge each model's predictive capabilities and overall performance on the test data.

The `XGBoost Classifier` stood out as the most effective model, achieving an accuracy of 94.09% and a macro-averaged F1 score of 0.9404, indicating strong generalization and predictive capabilities across different classes. The `Gradient Boosting Classifier` also performed well, with an accuracy of 86.11% and a macro-averaged F1 score of 0.8720.

Although other models such as Nearest Centroid and SVM showed good performance, they were outperformed by the XGBoost and Gradient Boosting Classifiers in terms of accuracy and F1 scores. This project highlighted the importance of proper data preprocessing, feature selection, and the use of ensemble methods like XGBoost and Gradient Boosting for achieving high classification performance. The experience provided valuable insights into the strengths and limitations of different classification models and emphasized the potential for further fine-tuning and exploration of advanced algorithms for even higher accuracy levels in future work.

## References

- [1]. Mohamed, A. E. (2017). Comparative study of four supervised machine learning techniques for classification. *International Journal of Applied*, 7(2), 1-15.
- [2]. Kamiran, F., & Calders, T. (2012). Data preprocessing techniques for classification without discrimination. *Knowledge and information systems*, 33(1), 1-33
- [3]. Abdurrahman, G., & Sintawati, M. (2020, May). Implementation of xgboost for classification of parkinson's disease. In *Journal of Physics: Conference Series* (Vol. 1538, No. 1, p. 012024). IOP Publishing.
- [4]. Nasution, M. Z. F., Sitompul, O. S., & Ramli, M. (2018, March). PCA based feature reduction to improve the accuracy of decision tree c4. 5 classification. In *Journal of Physics: Conference Series* (Vol. 978, No. 1, p. 012058). IOP Publishing.