Machine Learning
Project Document

Student Name	Mohd Arshad
Batch	AI Elite 17
Project Name	Bean Type Classifier
Project Domain	Consumer products/goods
Type of Machine Learning	Supervised
Type of Problem	Classification
Project Methodology	MLDL-C
Stages Involved	Problem statement, Data collection, EDA, preprocessing, EDA, Training, and Evaluation.

Business Understanding:

➢ Problem Statement:

The problem aims to develop a robust classification model capable of accurately identifying different classes of dry beans based on a comprehensive set of features. Dry beans are a staple food ingredient consumed worldwide, and their classification plays a crucial role in quality control, agricultural research, and food processing industries. Therefore, the objective is to leverage machine learning techniques to automate the classification process, enabling efficient and accurate identification of bean types.

By developing a robust classification model, this project aims to streamline the process of identifying different classes of dry beans, thereby enhancing productivity, reducing manual effort, and facilitating better decision-making in industries related to agriculture, food processing, and quality assurance.

➢ Business constraints:

Importance of Accurate Classification: Accurate classification is crucial for maintaining product quality and customer satisfaction. By precisely identifying different classes of dry beans, producers can ensure consistency in product quality and meet the expectations of consumers. This accuracy contributes to building trust in the brand and fostering long-term relationships with customers.

Need for Computational Efficiency: The model should be computationally efficient to handle large datasets in real time. With the increasing volume of data generated in the agricultural and food processing industries, the classification model needs to process information swiftly and make predictions in real time. This efficiency enables timely decision-making and enhances operational agility in various applications.

Impact of Misclassification: Misclassification of dry beans could result in financial losses for both producers and consumers. Incorrect categorization of beans may lead to quality control issues, affecting product consistency and market competitiveness. Therefore, minimizing misclassification errors is paramount to safeguarding financial interests and maintaining consumer trust.

Data Collection and Understanding

a) Data Collection:

Data has been collected from the Kaggle website.

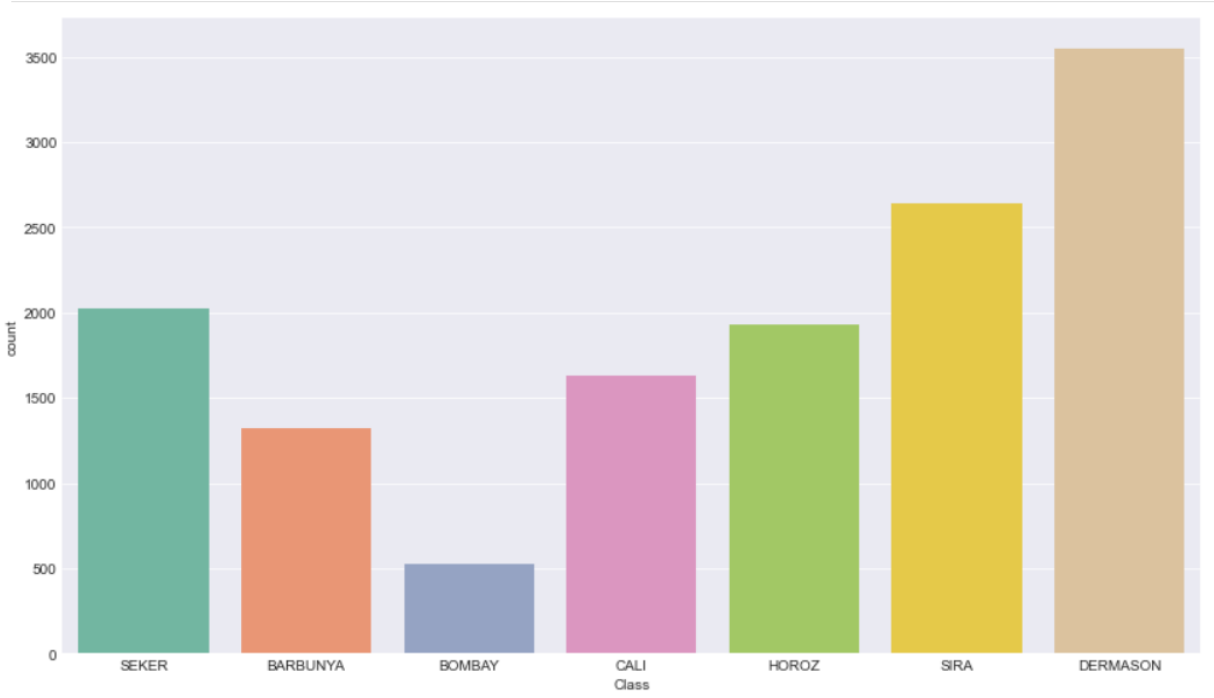
https://www.kaggle.com/datasets/muratkokludataset/dry-bean-dataset

b) Data Understanding:

- Seven different types of dry beans were used in this research, taking into account features such as form, shape, type, and structure by the market situation. A computer vision system was developed to distinguish seven different registered varieties of dry beans with similar features to obtain uniform seed classification.

- For the classification model, images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera. Bean images obtained by computer vision system were subjected to segmentation and feature extraction stages, and a total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains.

- The class variable represents the target label to be predicted for each bean sample. The dataset includes multiple class labels, including Dermason, Sira, Seker, Horoz, Cali, Barbunya, and Bombay. Each class label corresponds to a distinct type or variety of dry beans.



Number of instancs for each class , Dermason has the highest number.

S No	Feature Name	Data Type
1	Area	int
2	Perimeter	Float
3	MajorAxisLength	Float
4	MinorAxisLength	Float
5	AspectRation	Float
6	Eccentricity	Float
7	ConvexArea	Int
8	EquivDiameter	Float
9	Extent	Float
10	Solidity	Float
11	Roundness	Float
12	Compactness	Float
13	ShapeFactor1	Float
14	ShapeFactor2	Float
15	ShapeFactor3	Float
16	ShapeFactor4	Float

Data Preparation

a) Exploratory Data Analysis:

Missing Values: No missing values found.
Duplicates: No duplicate rows present.
Outliers: Identified and addressed using appropriate techniques.
Distributions: Analyzed distributions of each feature.

S No	**Type**	**Feature Names**	**Observation**
1	Missing Values	-	
2	Duplicates	-	
3	Outliers	All features	There are outliers in each feature.
4	Distributions	-	

b) Data Cleaning/wrangling:

To tackle outliers within each feature of the Dry Bean dataset, we've adopted a clipping technique leveraging the Interquartile Range (IQR) method. This strategy revolves around establishing upper and lower boundaries for each feature by utilizing the IQR. Outliers, that lie beyond these boundaries, are then substituted with the corresponding boundary values.

By utilizing this approach, we aim to effectively address outliers, ensuring that extreme values do not unduly influence our analysis and modeling processes. This methodology enhances the robustness of our models and facilitates more accurate and reliable results, ultimately contributing to improved decision-making and insights derived from the dataset.

c) Feature Selection:

S No	Removed Feature Name	Removed Feature Name	Test Performed
1	EquivDiameter	Redundant; derived from Area and Perimeter**.**	histplot
2	Compactness	Prevents multicollinearity; simplifies model while retaining relevance.	histplot
3	ConvexArea	Overlaps with Area information.	histplot

Model Building:

For the model-building stage, the KNN (K-Nearest Neighbors) classifier was selected as the algorithm of choice for solving the classification problem of identifying different types of dry beans based on their characteristics.

To start, the dataset was divided into two main subsets: the training set and the testing set. The training set, which comprised the majority of the data, was used to train the KNN classifier, while the testing set, representing a smaller portion of the data, was held back to evaluate the trained model's performance.

During the training phase, the KNN algorithm effectively learns the underlying patterns and relationships in the training data by memorizing the feature values and corresponding class labels of the training instances. This process allows the model to make predictions for unseen data points during the testing phase.

In terms of implementation, various techniques were employed to enhance the performance of the KNN classifier. These include:

Feature scaling: Before training the model, numeric features were standardized using standard scaling to ensure that all features contributed equally to the distance calculations and prevent features with larger scales from dominating the distance metrics.

Hyperparameter tuning: The hyperparameter 'k', representing the number of neighbors to consider, was optimized through techniques like cross-validation to find the optimal value that maximizes the model's performance on the validation set.

Evaluation metrics: Different evaluation metrics, such as accuracy, precision, recall, and F1-score, were used to assess the classifier's performance on the testing set and determine its effectiveness in accurately classifying different types of dry beans.

Overall, the KNN classifier proved to be a suitable choice for the classification problem, leveraging a distance-based approach to effectively classify different types of dry beans based on their characteristics, while incorporating various techniques to optimize its performance and generalization capabilities.

S No	Type of Problem	Approach	Algorithm Name
1	Classification	Distance-based	KNN

Model Training:

During the model training stage, hyperparameter tuning was performed to optimize the performance of the KNN (K-Nearest Neighbors) classifier. The primary hyperparameter tuned in the KNN algorithm is 'k', representing the number of neighbors to consider when making predictions for a new data point.

To find the optimal value of 'k', a systematic approach was employed, where the KNN classifier was trained and evaluated using different values of 'k'. Typically, a range of odd values for 'k' is considered to prevent ties when determining the class label based on the majority vote of the nearest neighbors.

In this specific case, the hyperparameter tuning process involved testing various values of 'k' within a predefined range. For example, 'k' values ranging from 1 to 10 with a step size of 2 (i.e., 1, 3, 5, 7, 9) might have been considered. Each value of 'k' was evaluated using a suitable evaluation metric, such as accuracy, which measures the model's overall correctness in predicting the class labels of the test data. In this case, the calculated log loss value was 1.824, indicating the average negative log likelihood of the classifier's predictions.

S No	Algorithm Name	Hyper-parameter tuning	Metric Used for Evaluation
1	KNN(brute force)	7-15	Accuracy and log loss

```
KNN accuracy: 92.29%
Brute Force accuracy: 92.29%
KD Tree accuracy: 92.29%
Ball Tree accuracy: 92.29%
Auto accuracy: 92.80%
```

Model Evaluation:

The final model demonstrates strong performance in classifying dry beans, with an accuracy of approximately 92.80%. It exhibits high precision (92.83%), recall (92.80%), and F1-score (92.80%), indicating effective classification across various classes. The confusion matrix provides detailed insights into the model's predictions for each class. Overall, the model shows promising results, indicating its potential for accurate and reliable classification of dry beans.

```
Accuracy: 0.9280205655526992
Precision: 0.9282868718709872
Recall: 0.9280205655526992
F1-score: 0.9280402386416641
Confusion Matrix:
[[243   0  17   0   1   2   2]
 [  0 104   0   0   0   0   0]
 [  4   0 308   0   7   2   5]
 [  0   0   0 667   1   8  33]
 [  0   0   7   4 361   0  14]
 [  3   0   0   9   0 384  10]
 [  3   0   0  48  12   4 460]]
```

Conclusion:

After analyzing the performance of the K-nearest neighbors (KNN) algorithm with different values of k after feature scaling, we've determined that k=7 provides the best accuracy. At this point, the accuracy is high, and there's minimal change in accuracy beyond this value. Additionally, the error is relatively low when k=7.

After conducting an extensive evaluation of various algorithms, including KDTree, BallTree, and Brute-Force, we've observed that they all exhibit a similar level of performance in terms of accuracy. Across the board, regardless of the algorithm employed, the general accuracy rate remains consistently at 92%.

This finding suggests that the choice of algorithm doesn't significantly impact the predictive capability of the model. Despite their differences in implementation and computational complexity, these algorithms yield comparable results in terms of accuracy.