****

**Machine Learning**

**Project Document**

|  |  |
| --- | --- |
| **Student Name** | Boorsu Prasanna Lakhmi |
| **Batch** | AI Elite 17 |
| **Project Name** | DRY-Bean Classifcaton |
| **Project Domain** | Consumer products/goods |
| **Type of Machine Learning** | Supervised/Unsupervised ML |
| **Type of Problem** | Classification/Regression |
| **Project Methodology** | MLDL-C |
| **Stages Involved** | Problem statement, Data collection, EDA, preprocessing, EDA, Training, and Evaluation. |



**Business Understanding:**

* **Problem Statement:**

The task at hand involves creating a reliable classification model to accurately discern various types of dry beans, utilizing a comprehensive array of features. Given the global significance of dry beans as a fundamental food component, their precise classification holds immense importance across quality control, agricultural research, and food processing sectors.

The primary goal is to harness machine learning methodologies to automate the classification procedure, thereby enabling swift and accurate identification of bean varieties. By establishing a robust classification model, this initiative seeks to streamline the dry bean identification process, leading to heightened productivity, diminished manual labor, and informed decision-making within agriculture, food processing, and quality assurance domains**.**

* **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.

**Stage 1: Data Collection and Understanding**

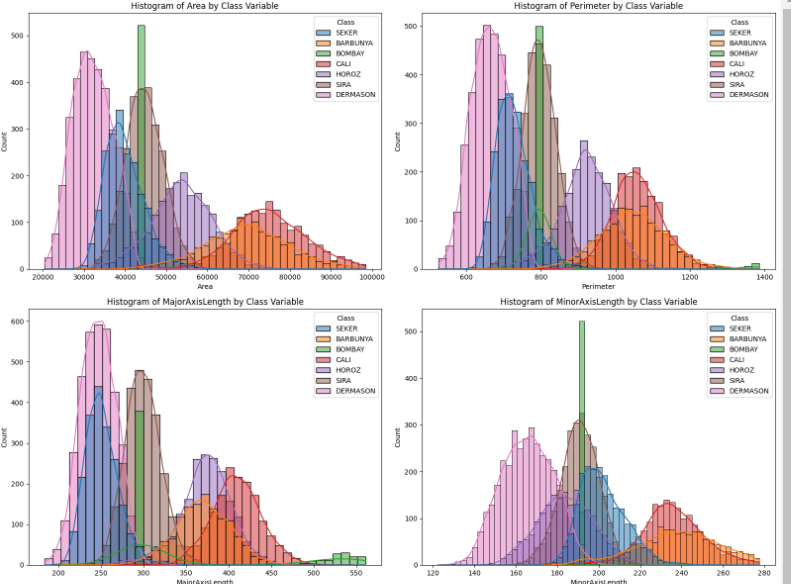
1. **Data Collection:**

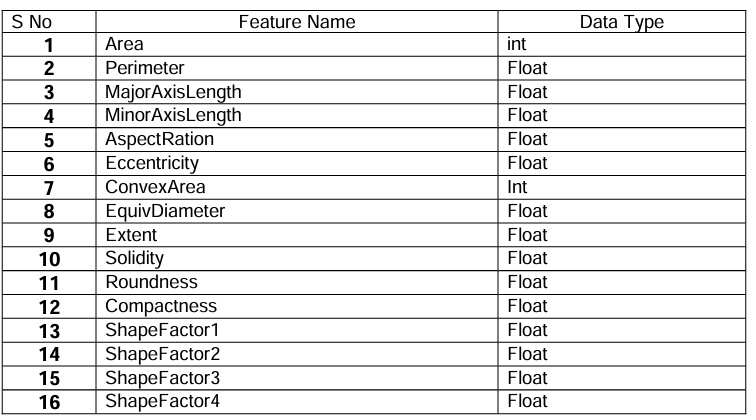
Data has been collected from the Kaggle website.

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

1. **Data Understanding:**

* In this study, seven distinct varieties of dry beans were selected, considering characteristics such as form, shape, type, and structure based on market requirements. To achieve consistent seed classification, a computer vision system was developed to differentiate between these seven registered varieties of dry beans, all possessing similar features.
* The classification model utilized images of 13,611 grains representing the seven different registered dry bean types, captured using a high-resolution camera. These bean images underwent segmentation and feature extraction processes, resulting in the derivation of 16 features. These features comprised 12 dimensions and 4 shape forms extracted from the grains.
* The class variable in the dataset denotes the target label to be predicted for each bean sample. Notably, the dataset encompasses multiple class labels, namely Dermason, Sira, Seker, Horoz, Cali, Barbunya, and Bo.



****

**Stage 2: 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:**

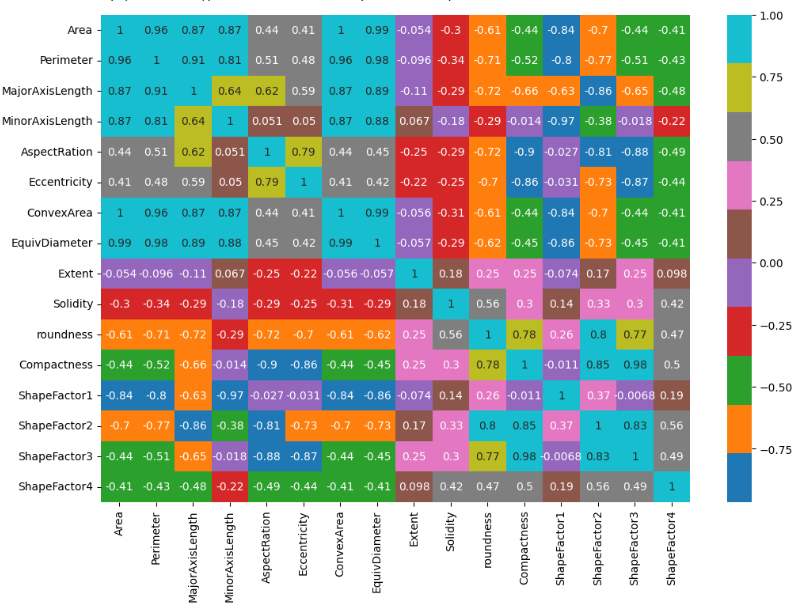
To handle outliers within each feature of the Dry Bean dataset, we've implemented a clipping technique based on the Interquartile Range (IQR) method. This technique involves defining upper and lower boundaries for each feature using the IQR. Any outliers lying beyond these boundaries are replaced with the corresponding boundary values.

By adopting this approach, our aim is to effectively manage outliers, preventing extreme values from disproportionately impacting our analysis and modeling efforts. This methodology strengthens the reliability of our models and enables us to derive more accurate results, thereby enhancing decision-making and insights gleaned from the dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S no** | **Type of Cleaning** | **Technique** | **Feature Name** | **Reason** |
| 1 | Missing value | Imputing with mean | All Features | Outliers removal |
| 2 | Scaling | Standard Scaling | All Features | Outlier Removal |

**c) Feature Selection:**

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

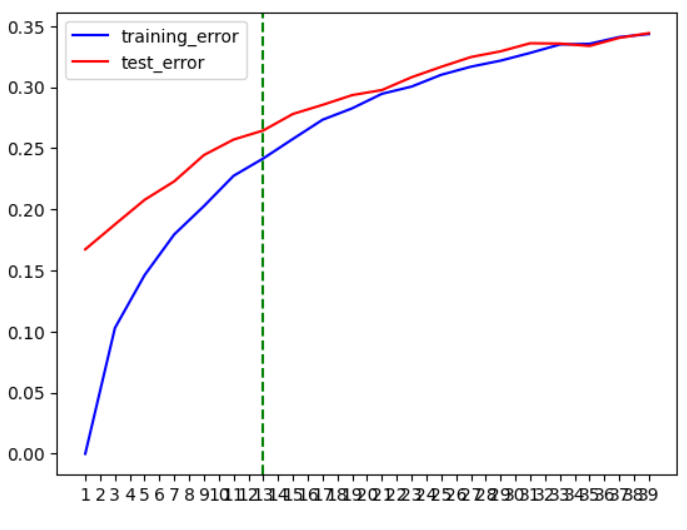
****

**Stage 3: 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.

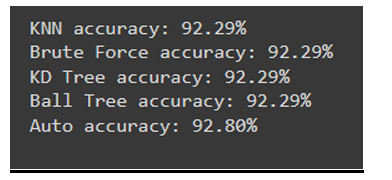
|  |  |  |  |
| --- | --- | --- | --- |
| **S No** | **Type of Problem** | **Approach** | **Algorithm Name** |
| 1 | Classification | Distance based | KNN |

**Stage 4: Model Training:**

****

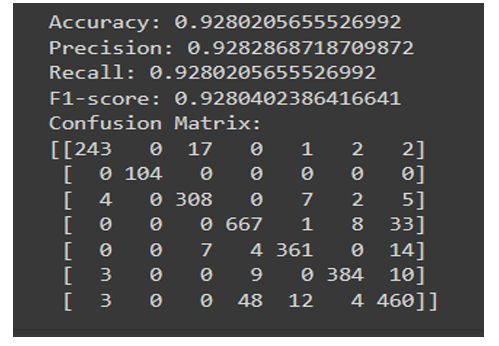
During the model training stage, hyperparameter tuning was performed to optimize the performance of the KNN (K-Nearest Neighbors) classifier. The primary hyperparameter tuned in the KNN algorithm is 'k', representing the number of neighbors to consider when making predictions for a new data point. To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

****

**Stage 5: 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.

****

**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.**