CS361: Dry Bean Classification

Abinash Sonowal 210101004, Anoop Singh 210101016, Kundan Meena 210101060, Ravi Lahare 210101086

Abstract

This project investigates the application of machine learning for classifying dry bean varieties based on a dataset containing seven dimensions and four shape parameters. Support Vector Machines, Random Forest and Decision Tree are implemented from scratch and compared for classification accuracy. The study emphasizes the significance of multidimensional shape attributes in effectively categorizing dry bean in different varieties. The findings contribute to streamlining agricultural processes, offering a practical approach for quality control and classification.

1. Introduction

Dry beans, integral to diets globally, present a challenge in efficient classification due to their diverse varieties. The motivation behind this project stems from the need to streamline the sorting process for seven specific dry bean classes: Seker, Barbunya, Bombay, Cali, Horoz, Sira, and Dermason. The current manual sorting methods are time-consuming and prone to errors, impacting both efficiency and quality control in the agricultural industry.

The primary objective of this project is to leverage machine learning techniques to create a classification model based on user-provided features. By utilizing 12-dimensional and 4 shape features for each bean, we aim to develop a robust system capable of accurately categorizing dry beans into their respective classes. The ultimate goal is to provide a reliable and automated solution that not only enhances sorting accuracy but also contributes to the overall efficiency and sustainability of the dry bean production and distribution chain. This project holds promise for revolutionizing agricultural practices, ensuring consistent quality, and supporting the industry's progression towards more advanced, automated methodologies.

2. Methods

2.1. Data Collection and Preprocessing

1.Dataset is collected from Kaggle ,contains 13,611 instances ,each instanse having a total of 16 features; 12 dimensions and 4 shape forms. .
2.Outline the preprocessing steps you will take to clean and prepare the data for model training. This may include handling missing values, normalization, feature scaling, and encoding categorical variables

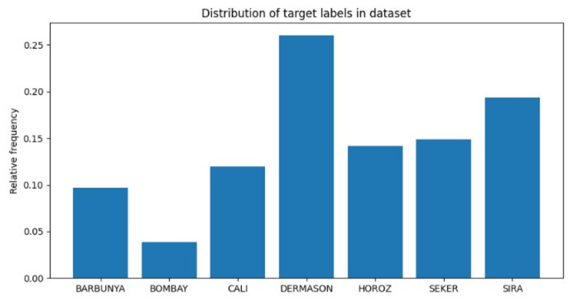


Figure 1. data distribution

2.2. Model Selection

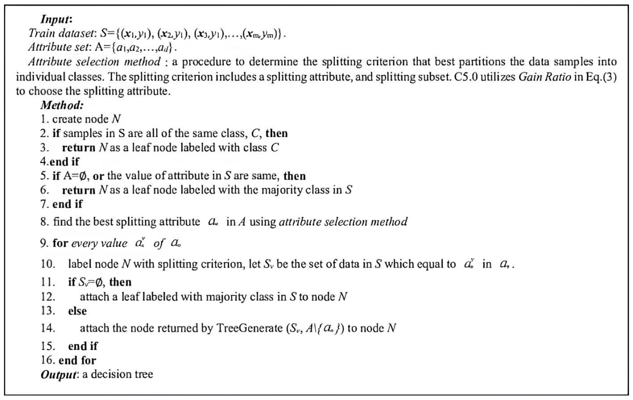
Choose appropriate machine learning algorithms for classification tasks, such as decision trees, random forests, support vector machines, or neural networks. Experiment with different algorithms to determine which ones perform best on the dataset.

- Explore a variety of machine learning algorithms suitable for classification tasks, such as Random Forest, Support Vector Machines (SVM).
- Compare the performance of these models using cross-validation and select the best-performing one(s).

2.3. Algorithm

Following are the three main algorithm to be used

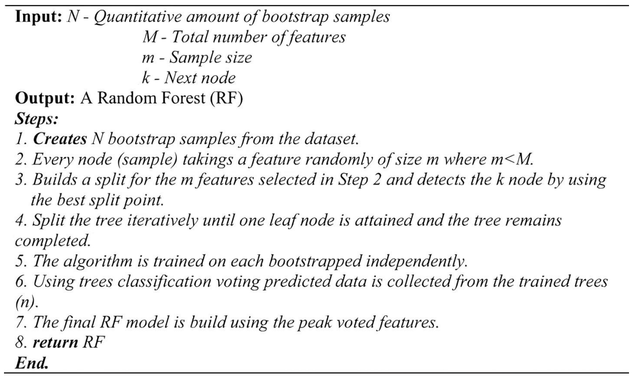
- 1.Decision tree: Decision trees can effectively handle both numerical and categorical data and are interpretable, making them suitable for classification tasks



```
Input:
Train dataset: $S=\{(x_1,y_1), (x_2,y_1), (x_3,y_1),\ldots,(x_m,y_m)\}$.
Attribute set: $A=\{a_1,a_2,\ldots,a_d\}$.
Attribute selection method : a procedure to determine the splitting criterion that best partitions the data samples into individual classes. The splitting criterion includes a splitting attribute, and splitting subset. C5.0 utilizes Gain Ratio in Eq.(3) to choose the splitting attribute.
Method:
1. create node $N$
2. if samples in S are all of the same class, $C$, then
3.    return $N$ as a leaf node labeled with class $C$
4.end if
5. if $A=\emptyset$, or the value of attribute in $S$ are same, then
6.    return $N$ as a leaf node labeled with the majority class in $S$
7. end if
8. find the best splitting attribute $a_*$ in $A$ using attribute selection method
9. for every value $a_*^v$ of $a_*$
10.    label node $N$ with splitting criterion, let $S_v$ be the set of data in $S$ which equal to $a_*^v$ in $a_*$.
11.    if $S_v=\emptyset$, then
12.       attach a leaf labeled with majority class in $S$ to node $N$
13.    else
14.       attach the node returned by TreeGenerate ($S_v$, $A\setminus\{a_*\}$) to node $N$
15.    end if
16. end for
Output: a decision tree
```

Figure 2. pseudo code of Decision tree

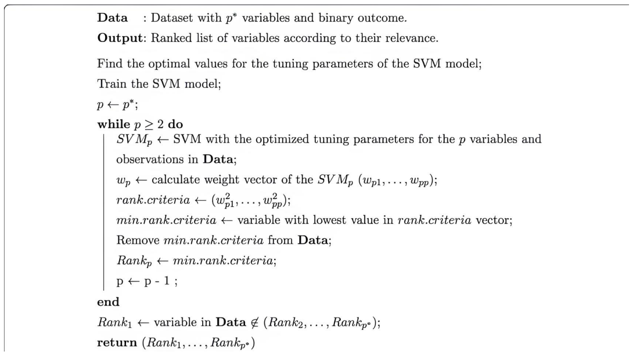
- 2.Random Forest: Random forests can provide higher accuracy by aggregating multiple decision trees and reducing overfitting.



```
Input: N - Quantitative amount of bootstrap samples
       M - Total number of features
       m - Sample size
       k - Next node
Output: A Random Forest (RF)
Steps:
1. Creates N bootstrap samples from the dataset.
2. Every node (sample) takings a feature randomly of size m where m<M.
3. Builds a split for the m features selected in Step 2 and detects the k node by using
   the best split point.
4. Split the tree iteratively until one leaf node is attained and the tree remains
completed.
5. The algorithm is trained on each bootstrapped independently.
6. Using trees classification voting predicted data is collected from the trained trees
(n).
7. The final RF model is build using the peak voted features.
8. return RF
End.
```

Figure 3. pseudo code of Random Forest

- 3.Support Vector Machines (SVM): SVMs work well for both linear and nonlinear classification tasks and can handle high-dimensional data effectively.



```
Data   : Dataset with $p^*$ variables and binary outcome.
Output: Ranked list of variables according to their relevance.
Find the optimal values for the tuning parameters of the SVM model;
Train the SVM model;
$p \leftarrow p^*$;
while $p \geq 2$ do
    $SVM_p \leftarrow$ SVM with the optimized tuning parameters for the $p$ variables and
    observations in Data;
    $w_p \leftarrow$ calculate weight vector of the $SVM_p$ $(w_{p1},\ldots,w_{pp})$;
    $rank.criteria \leftarrow (w_{p1}^2,\ldots,w_{pp}^2)$;
    $min.rank.criteria \leftarrow$ variable with lowest value in $rank.criteria$ vector;
    Remove $min.rank.criteria$ from Data;
    $Rank_p \leftarrow min.rank.criteria$;
    p ← p - 1 ;
end
$Rank_1 \leftarrow$ variable in Data $\notin (Rank_2,\ldots,Rank_{p^*})$;
return $(Rank_1,\ldots,Rank_{p^*})$
```

Figure 4. pseudo code of Support Vector Machines (SVM)

3. References

Kaggle. Dry Bean Dataset Classification. Retrieved from [https://www.kaggle.com/datasets/nimapourmoradi/dry-bean-dataset-classification/data]

Kaya, Y., & Karaköy, T. (2019). Automatic Recognition of Seven Dry Beans Using Computer Vision and Machine Learning Techniques. Computers and Electronics in Agriculture, 161, 280-290.

Akpınar, E. K., & Karaköy, T. (2020). A Comparative Analysis of Machine Learning Algorithms for Dry Bean Classification. Turkish Journal of Electrical Engineering & Computer Sciences, 28(4), 2431-2445.

Alotaibi, B. S., Tola, E., & Alharbi, E. (2021). Deep Learning Models for Dry Bean Classification: A Comparative Study. Computers and Electronics in Agriculture, 183, 106005.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). Pattern Classification (2nd edition). John Wiley and Sons.