Automatic Detection of Dry bean classification using machine learning

Abstract

This report focuses on developing a machine-learning model for classifying different registered types of dry beans. With the goal of obtaining uniform seed varieties, we are utilizing the dry bean dataset consisting of images of 13,611 grains from seven different dry bean varieties. We used XGBoost Classifier and Decision Tree and tuned its hyperparameters for better accuracy and were evaluated using 10-fold cross-validation. The XGBoost classifier achieved the highest accuracy, accurately classifying the Dermason, Barbunya, Bombay, Dermason ,Cali, Horoz, Seker, and Sira bean varieties with percentages ranging from 86.84% to 100.00%. These findings demonstrate the potential of the machine learning models in meeting the demands of producers and customers for obtaining uniform bean varieties.

1. Introduction

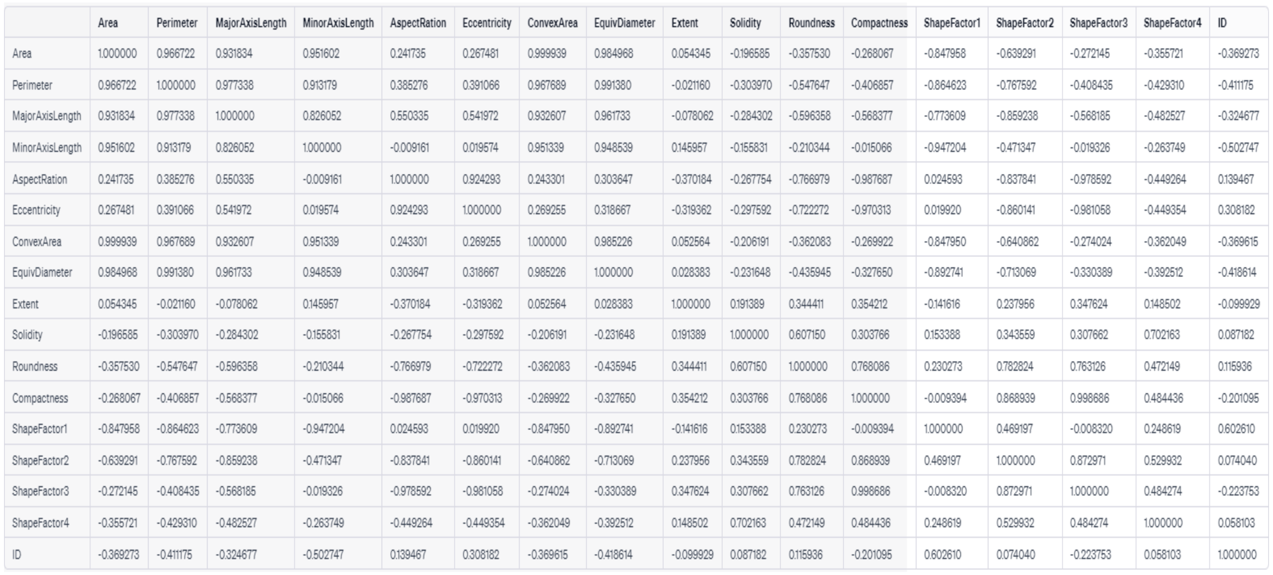
Dry bean is a globally significant pulse crop and plays a crucial role in Turkish agriculture. However, its cultivation is susceptible to the effects of climate change. Planting new seeds with increased resistance to stress factors and determining seed characteristics are vital for successful plant growth. Seed quality is a major concern for dry bean producers and markets, as lower-quality seeds can lead to reduced yields even under optimal conditions. Manual seed classification is challenging, time-consuming, and inefficient, necessitating the need for automatic grading and a system for categorisation. In Certain places, dry beans are classified into varieties based on botanical characteristics and market conditions. Achieving accurate classification of dry bean genotypes is crucial for crop production and market value. Recent research has used image processing techniques and computer vision systems to categorise dry bean seeds based on factors such as quality, colour, and size. The purpose of this paper is to create a machine-learning model based on a computer vision system for classifying dry beans based on morphologically comparable characteristics. Using 10-fold cross-validation, the XGBoost Classifier and Decision Tree were developed and tested. These models' performance criteria were evaluated to determine their usefulness in dry bean categorization.

2. Description and exploration of the dataset

The dataset under consideration contains 13,611 samples, each with 16 geometric attributes and a label specifying the bean species. Dermason, Barbunya, Bombay, Seker, Cali, Horoz, and Sira are the bean species represented in the dataset. Area, Perimeter, MajorAxisLength, MinorAxisLength, AspectRatio, Eccentricity, ConvexArea, EquivDiameter, Extent, Solidity, Roundness, Compactness, ShapeFactor1, ShapeFactor2, ShapeFactor3, and ShapeFactor4 are the 16 geometric properties.

It is important to note that the geometric data does not provide any information about the colour of the beans. While this may be inconvenient from a practical standpoint, as different dry bean species typically exhibit colour variations, it does not significantly affect the task of classifying dry beans using machine learning models.

After conducting a correlation analysis, it has been observed that many of the features exhibit strong positive or negative correlations. This is primarily attributed to the nature of these features, which predominantly represent its geometric measurements.



	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	Roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3	ShapeFactor4	ID
Area	1.000000	0.966722	0.931834	0.951602	0.241735	0.267481	0.999939	0.984968	0.054345	-0.196585	-0.357530	-0.268067	-0.847958	-0.639291	-0.272145	-0.355721	-0.369273
Perimeter	0.966722	1.000000	0.977338	0.913179	0.385276	0.391066	0.967689	0.991380	-0.021160	-0.303970	-0.547647	-0.406857	-0.864623	-0.767592	-0.408435	-0.429310	-0.411175
MajorAxisLength	0.931834	0.977338	1.000000	0.826052	0.550335	0.541972	0.932607	0.961733	-0.078062	-0.284302	-0.596358	-0.568377	-0.773609	-0.859238	-0.568185	-0.482527	-0.324677
MinorAxisLength	0.951602	0.913179	0.826052	1.000000	-0.009161	0.019574	0.951339	0.948539	0.145957	-0.155831	-0.210344	-0.015066	-0.947204	-0.471347	-0.019326	-0.263749	-0.502747
AspectRation	0.241735	0.385276	0.550335	-0.009161	1.000000	0.924293	0.243301	0.303647	-0.370184	-0.267754	-0.766979	-0.987687	0.024593	-0.837841	-0.978592	-0.449264	0.139467
Eccentricity	0.267481	0.391066	0.541972	0.019574	0.924293	1.000000	0.269255	0.318667	-0.319362	-0.297592	-0.722272	-0.970313	0.019920	-0.860141	-0.981058	-0.449354	0.308182
ConvexArea	0.999939	0.967689	0.932607	0.951339	0.243301	0.269255	1.000000	0.985226	0.052564	-0.206191	-0.362083	-0.269922	-0.847950	-0.640862	-0.274024	-0.362049	-0.369615
EquivDiameter	0.984968	0.991380	0.961733	0.948539	0.303647	0.318667	0.985226	1.000000	0.028383	-0.231648	-0.435945	-0.327650	-0.892741	-0.713069	-0.330389	-0.392512	-0.418614
Extent	0.054345	-0.021160	-0.078062	0.145957	-0.370184	-0.319362	0.052564	0.028383	1.000000	0.191389	0.344411	0.354212	-0.141616	0.237956	0.347624	0.148502	-0.099929
Solidity	-0.196585	-0.303970	-0.284302	-0.155831	-0.267754	-0.297592	-0.206191	-0.231648	0.191389	1.000000	0.607150	0.303766	0.153388	0.343559	0.307662	0.702163	0.087182
Roundness	-0.357530	-0.547647	-0.596358	-0.210344	-0.766979	-0.722272	-0.362083	-0.435945	0.344411	0.607150	1.000000	0.768086	0.230273	0.782824	0.763126	0.472149	0.115936
Compactness	-0.268067	-0.406857	-0.568377	-0.015066	-0.987687	-0.970313	-0.269922	-0.327650	0.354212	0.303766	0.768086	1.000000	-0.009394	0.868939	0.998686	0.484436	-0.201095
ShapeFactor1	-0.847958	-0.864623	-0.773609	-0.947204	0.024593	0.019920	-0.847950	-0.892741	-0.141616	0.153388	0.230273	-0.009394	1.000000	0.469197	-0.008320	0.248619	0.602610
ShapeFactor2	-0.639291	-0.767592	-0.859238	-0.471347	-0.837841	-0.860141	-0.640862	-0.713069	0.237956	0.343559	0.782824	0.868939	0.469197	1.000000	0.872971	0.529932	0.074040
ShapeFactor3	-0.272145	-0.408435	-0.568185	-0.019326	-0.978592	-0.981058	-0.274024	-0.330389	0.347624	0.307662	0.763126	0.998686	-0.008320	0.872971	1.000000	0.484274	-0.223753
ShapeFactor4	-0.355721	-0.429310	-0.482527	-0.263749	-0.449264	-0.449354	-0.362049	-0.392512	0.148502	0.702163	0.472149	0.484436	0.248619	0.529932	0.484274	1.000000	0.058103
ID	-0.369273	-0.411175	-0.324677	-0.502747	0.139467	0.308182	-0.369615	-0.418614	-0.099929	0.087182	0.115936	-0.201095	0.602610	0.074040	-0.223753	0.058103	1.000000

Pearson Correlation coefficient between the features

3. Machine Learning Technique used and Tuning

3.1. Data Preprocessing

This is the process of cleaning and converting raw data into a format that machine learning algorithms can interpret. Real-world data is frequently incomplete, inconsistent, and riddled with inaccuracies.

Convert Categorical Target to Numerical: Machine learning models require inputs to be numerical. If the target variable ('Class') is categorical, we need to convert it into a numerical form. Encoding techniques such as label encoding, or one-hot encoding can be used for this.

Splitting the data into features and targets: This is where we separate our independent variables (X2) from our dependent variable (y2). Independent variables are used to predict the value of the dependent variable.

Splitting the data into training and testing sets: It is a regular practice to divide your dataset into two parts: training and testing. This stage is critical for determining the accuracy of identifying new or previously unknown data that were not used during training. Otherwise, there is a significant possibility of the model becoming overfit.

The implemented approach used for this classification problem is the **XGBoost classifier**. The XGBoost classifier is a popular choice for classification tasks due to its strong performance, scalability, and ability to handle complex datasets. It often achieves state-of-the-art results and can handle a variety of feature types.

3.2. Model Development and Training

XGBoost is an abbreviation for eXtreme Gradient Boosting. It is a gradient-boosting machine solution that is intended to be very efficient, adaptable, and portable. In basic words, it constructs the model in stages, like other boosting approaches, then generalises them by enabling optimisation of any differentiable loss function.

The mathematical expression for XGBoost can be represented as follows:

For a given dataset with N samples and M features, the objective of XGBoost is to find an ensemble model F(x) that minimizes the following regularized objective function:

$$\mathrm{Obj}(F) = L\,(y, F) + \Omega(F),$$

where L (y, F) is the loss function that measures the difference between the true labels y and the predicted labels F(x), and Ω(F) is a regularization term that penalizes complex models to avoid overfitting.

Hyperparameter Tuning: Hyperparameters are the parameters that are not learned from the model and are set before the learning process. GridSearchCV exhaustively considers all parameter combinations, which can be computationally expensive but ensures finding the optimal values. In this report, parameters for max_depth, n_estimators, and learning_rate are tuned.

max_depth: The depth of each tree is represented by max_depth, which is the highest number of individual characteristics utilised in each tree. I chose [3, 5, 7, 10] as potential values because they are common for many classification tasks.

n_estimators: The number of trees in the forest is represented by n_estimators. Generally, the more trees there are, the better the data can be learned. However, adding too many trees will slow down the training process; hence, selecting the proper amount is critical. For this task, I chose [50, 100, 150, 200].

learning_rate: or ETA is the step size used to prevent overfitting. It's usually between 0.01 and 0.3.

After the best parameters are found, they are used to predict the test set. This Fine-tuning helps in increasing the model's efficiency.

3.3. Limitations

XGBoost: Although XGBoost is known for its high performance, it is susceptible to noise and outliers, and might overfit if the data is not tuned properly.

GridSearchCV: This method can be very time-consuming and computationally expensive, especially if you have a large number of hyperparameters to tune, or if the dataset is large.

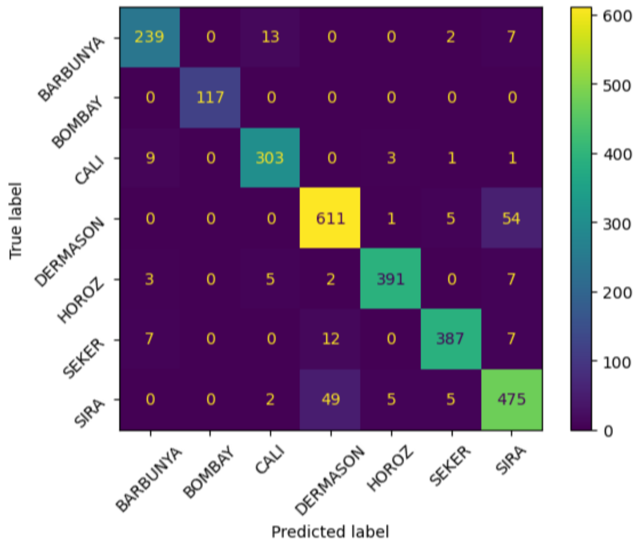
Alternative methods to XGBoost can be other ensemble techniques such as Random Forest or Supprt Vector Machines. For hyperparameter tuning, methods like Random Search or Bayesian Optimization can be more efficient.

4. Model Evaluation and results obtained

The model's performance is assessed using a k-fold evaluation strategy, which is chosen because it allows for the utilization of all available data for testing. Unlike the traditional 80:20 split, k-fold validation provides a more comprehensive evaluation by generating multiple metrics. Aggregating the mean of 10 average accuracies offers a more balanced assessment compared to relying on a single accuracy value from an 80:20 split. This approach instils greater confidence in the model. From the results, it is evident that the model after tuning excels in handling the data with values below.

Accuracy	**0.926551597502754**
Precision value	**0.9397191771330778**
Recall value	**0.937671337784139**
F1 Value	**0.9386062080753776**

A widely used method to visualize the performance of a multiclass classification model is through a confusion matrix. In the below figure, the confusion matrix illustrates the model's performance.



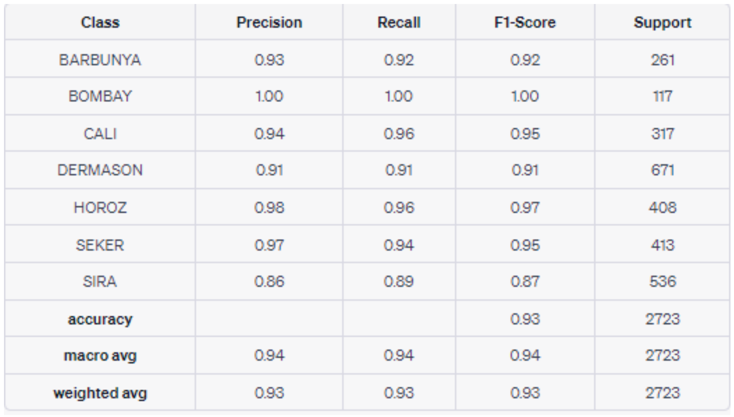
Confusion matrix for Xgboost classifier with tuned hyperparameters

The Confusion matrix shows the prediction for each class:

- Class 0 (BARBUNYA): 239 were correctly predicted (True positives), 22 were misclassified (False negatives).

- Class 1 (BOMBAY): 117 were correctly predicted, and none were misclassified.
- Class 2 (CALI): 303 were correctly predicted, 14 were misclassified.
- Class 3 (DERMASON): 611 were correctly predicted, 60 were misclassified.
- Class 4 (HOROZ): 391 were correctly predicted, 17 were misclassified.
- Class 5 (SEKER): 387 were correctly predicted, 26 were misclassified.
- Class 6 (SIRA): 475 were correctly predicted, 61 were misclassified.

The below table shows the precision, recall and f1 value for each class:



Class	Precision	Recall	F1-Score	Support
BARBUNYA	0.93	0.92	0.92	261
BOMBAY	1.00	1.00	1.00	117
CALI	0.94	0.96	0.95	317
DERMASON	0.91	0.91	0.91	671
HOROZ	0.98	0.96	0.97	408
SEKER	0.97	0.94	0.95	413
SIRA	0.86	0.89	0.87	536
accuracy			0.93	2723
macro avg	0.94	0.94	0.94	2723
weighted avg	0.93	0.93	0.93	2723

5. Decision Tree:

Decision trees are a type of supervised learning algorithm that is commonly used for classification tasks. They design a flowchart-like model, with each internal node representing a feature, each branch representing a decision rule, and each leaf node representing a class label. One of the main benefits of this is their inherent simplicity and ease of interpretation and they can also handle both numerical and categorical data. However, due to their unlimited degrees of freedom, decision trees are prone to overfitting if not properly monitored and adjusted. Additionally, since decision trees make splits based on a single feature at a time, and these splits are always orthogonal (at right angles), they are highly susceptible to changes in data orientation. We have tuned this model with a few hyperparameters.

The hyperparameters considered for the grid search are:

- criterion: it is the measure of equality of the split. Two options are considered: 'gini' and 'entropy'.
- max_depth: it is the maximum depth of the given decision tree. Four options are considered: 5, 10, 15, and 20.
- min_samples_split: It is the minimum sample required to split the internal node. Three options are considered: 2, 5, and 10.
- min_samples_leaf: It is the minimum sample required at leaf node. Three options are considered: 1, 2, and 5.

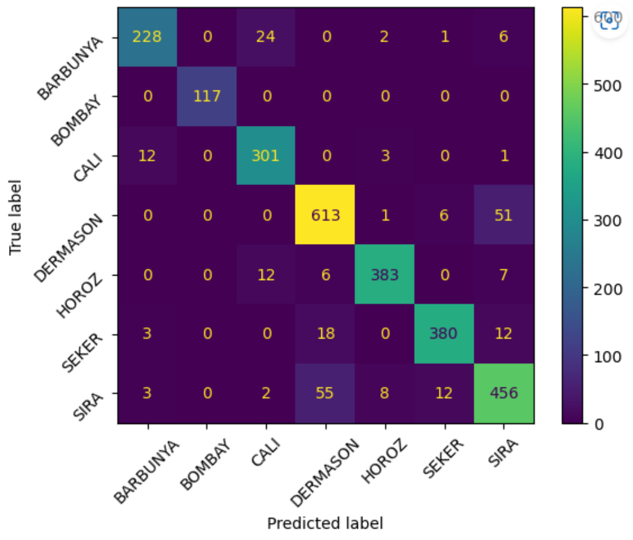
The grid search evaluates all combinations of these hyperparameters using cross-validation and selects the combination that achieves the best accuracy score.

The model is evaluated on the test set by making predictions using the best parameters obtained from the grid search.

5.1. Model Evaluation and results obtained:

The decision tree model achieved an accuracy of 0.910, indicating that it classified the majority of instances correctly. The precision rating of 0.925 indicates that the model correctly predicted a certain class 92.5% of the time. The recall value of 0.921 indicates that the model correctly classified 92.1% of the occurrences. The F1 value of 0.922 represents a good mix of precision and recall.

The confusion matrix gives additional information about the model's performance. It is a 7x7 matrix with the true class represented by each row and the predicted class represented by each column. Looking at the values in the below matrix, we can observe the following:



The model performed well in correctly classifying instances belonging to most classes, as indicated by the high values on the diagonal. For example, class 0 (the first row) has a high true positive count (228) and a relatively low number of false negatives (sum of values in the row except the diagonal). Some classes, such as class 1 (the second row), were predicted correctly in all instances, resulting in a precision of 1.0 for that class.

There are some misclassifications, as indicated by the non-zero values outside the diagonal. For example, class 0 has false positives in classes 2, 4, 5, and 6, and class 2 has false positives in class 4. These misclassifications contribute to the lower precision and recall values for these classes compared to most classes. The decision tree model shows good performance with high accuracy and relatively high precision, recall, and F1 values. However, there are some misclassifications, particularly between certain classes, which could be further investigated and addressed to improve the model's performance.

Overall, the XGBoost classifier performs well in this dataset with an accuracy of around 92 per cent, whereas the decision tree gives an accuracy of around 90 per cent but the XGBoost is computationally expensive and time-consuming compared to the Decision tree.

6. Conclusion:

The comprehensive analysis of the dry beans dataset in this report examined two classification approaches. The results indicated that the Decision tree achieved an accuracy of 91% while XGBoostClassifier demonstrated superior accuracy at 92.65% with a longer training time. If computational efficiency is a priority, selecting a Decision tree would be a suitable choice. On the other hand, applications that prioritize accuracy would benefit from selecting XGBoostClassifier. The confusion matrix, for both the models revealed common misclassifications between in few classes, whereas Bombay beans were classified perfectly due to their distinctive larger size. To further enhance accuracy, the utilization of hyperparameter tuning can be considered, although it may come at the expense of longer training times.

References:

Koklu, M. and Ozkan, I.A., 2020. Multiclass classification of dry beans using computer vision and machine learning techniques. Computers and Electronics in Agriculture, 174, https://doi.org/10.1016/j.compag.2020.105507

XGBoost documentation, XGBoost classifier, Available at: https://xgboost.readthedocs.io/en/stable/python/python_intro.html#scikit-learn-interface.

Scikit learn Documentation, Decision Trees, Available at: https://scikit-learn.org/stable/modules/tree.html

G Słowiński · 2021, Dry Beans Classification Using Machine Learning, Available at: https://ceur-ws.org/Vol-2951/paper3.pdf

Scikit learn Documentation, Decision tree Classifier. Available at: https://scikit-learn.org/stable/modules/tree.html

Murat Koklu, Ilker Ali Ozkan, Multiclass classification of dry beans using computer vision and machine learning techniques, Computers and Electronics in Agriculture 174 (2020) 105507.