Comparison of Dry Beans Classification using various Machine Learning Methods

Candidate Number: 260072
Informatics and Engineering
University of Sussex
Sussex, United Kingdom

***Abstract*— This report analysed two different machine learning techniques on their effectiveness in classifying Turkish dry beans. The study found that an approach with a multilayer perceptron and boosting algorithms (i.e. XGBoost) could achieve high classification scores. The paper uses these results to contrast the two methods and highlight their performance in the context of other publications on the topic as well as their viability in industrial applications. Both methods outperformed a naive baseline model, but the boosting algorithm showed better efficiency and faster runtimes, presenting itself as the overall better choice.**

***Index terms*—Machine Learning, Classification, MLP, Boosting Algorithms**

I. Introduction

This report analyses the effectiveness of two machine learning methods for classifying dry beans. Koklu and Ozkan provided the underlying dataset. [1]

A. *Reference paper overview*

The authors aimed to classify seven different species of dry beans cultivated in Turkey using machine learning techniques. A computer vision system (CVS) was developed to classify dry bean varieties according to their features. A good classifier for these dry beans is a valuable tool in industrial contexts. A well-working classification model could be used in factories to automate the selection and sorting processes that would be arduous for human workers.

B. *Features and Classification Method*

The system uses a camera with a CMOS-type sensor and extracts 16-dimensional and shape features for each dry bean. These features include area, perimeter, major and minor axis length, aspect ratio, roundness, compactness, and solidity. It then used image pre-processing and segmentation processes to obtain 13,611 bean sample images for feature extraction. The extracted features were used to create a multiclass dry bean dataset for classification. Different machine learning techniques, including Multi-Layer Perceptron (MLP), Support Vector Machines (SVM), Decision Trees (DT), and k-Nearest Neighborhood (kNN), were used to classify the collected images of the dry beans. The accuracy, error rate, precision, specificity, recall, and F1-score were used to evaluate the classification performance metrics of the models. Cross-validation was used to increase the robustness of the classification results [1].

II. Methods

The following section describes the machine learning pipeline deployed in this report. It is segmented into the following parts:

1) *General Preprocessing and Baseline Model:* Summary of the data preprocessing that has been done in this report to select the best features for the best classification results
2) *Classification:* Analysis using a Multi-layer Perceptron (MLP): Description of the Multilayer Perceptron Architecture used to classify the dataset
3) *Classification Analysis using Boosting Algorithm (XGBoost):* Description of an Additional Machine Learning Architecture (i.e. Boosting Algorithm) to enable a comparison between solutions
4) *Results:* Descriptions of the Results for both machine learning methods
5) *Discussion:* Discussion and contrast the two approaches taken in the publication and the broader implication of the results in the context of the current literature.

B. *Data Preprocessing*

The dataset consisted of 13611 samples and 16 features for each sample. The 16 features are: ‘Area’, ‘Perimeter’, ‘MajorAxisLength’, ‘MinorAxisLength’, ‘AspectRation’, ‘Eccentricity’, ‘ConvexArea’, ‘EquivDiameter’, ‘Extent’, ‘Solidity’, ‘roundness’, ‘Compactness’, ‘ShapeFactor1’, ‘ShapeFactor2’, ‘ShapeFactor3’, ‘ShapeFactor4’. Initially, the dataset was checked for duplicates and missing values. No missing values were detected, but 67 samples were duplicates and removed. This led to a dataset with 13547 entries. These steps were conducted to have a complete dataset available for analysis. Further, the correlations between all the variables were calculated to identify whether additional pre-processing steps need to be undertaken.

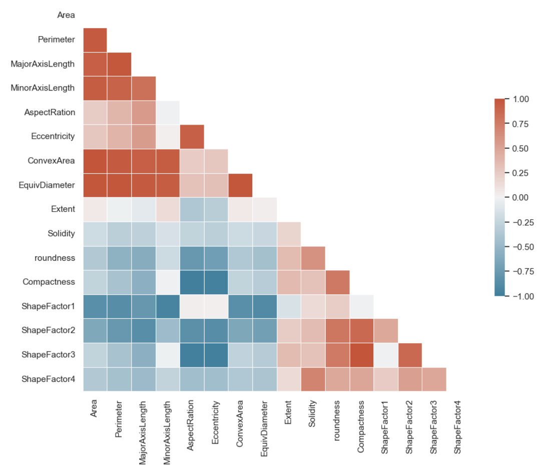

Figure 1: Correlation Matrix for all 16 features in the dataset.

In addition to the correlations of the variables, the distribution of the different bean types in the dataset was also assessed. The data indicated that many of the variables in the dataset were highly correlated. This suggests that more preprocessing steps are necessary to reduce unnecessary training times and prevent the deployed models from overfitting.

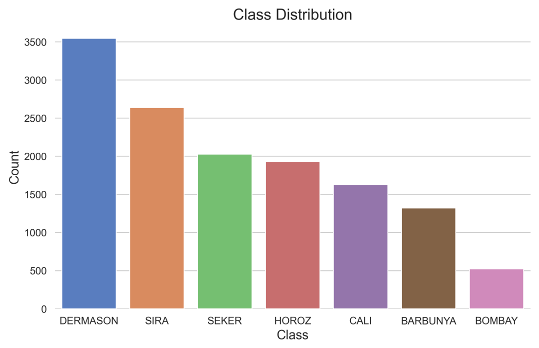

Figure 2: Distribution of the beans frequency within the dataset.

C. *'Dumb' Baseline Model*

To understand the effectiveness of any machine learning model, it is necessary to establish a baseline model that makes predictions using elementary rules. In the case of this report, the model selected is a ZeroR model. The most significant dry bean variant represented in the dataset is *Dermason* dry beans. 3546 out of 13543 samples in the dataset were Dermason dry beans. Hence the baseline model performance is the proportion of dermason beans in the entire dataset.

$$\text{Baseline}_{\text{Accuracy}} = \frac{N_{\text{Dermason}}}{N_{\text{total}}} \approx 0.26 \tag{1}$$

It follows from the baseline model that any more complicated model has to result in a prediction accuracy higher than 26 %. The following sections of the report provide a detailed analysis of the performance of a multilayer perceptron and a boosting algorithm for the classification task.

D. *Multilayer Perceptron (MLP)*

1) *Model Introduction:* A multilayer perceptron (MLP) is an artificial neural network consisting of multiple layers of interconnected neurons or nodes, primarily used for pattern recognition and classification tasks. It is an extension of the single-layer perceptron, which has limited capabilities. An MLP comprises an input layer, one or more hidden layers, and an output layer, each containing a set of neurons with activation functions. The primary purpose of an MLP is to learn and model complex, non-linear relationships in data. This is achieved through forward propagation, where input data is passed through the network to produce an output. The weights and biases associated with each neuron are adjusted during training using backpropagation to minimise the error between the predicted outcome and the actual target.MLPs are widely used in supervised learning tasks, such as image recognition, natural language processing, and speech recognition. Despite their capabilities, they suffer from certain drawbacks, including overfitting, and the vanishing gradient problem, which can be addressed using regularisation techniques and advanced architectures like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [2]. Alternatives to MLP use other algorithms, such as tree-based algorithms or logistic regression. However, MLP is, at its core, a universal function approximator and should hence be able to perform well at any task given enough data. The dataset presented here has enough entries for the MLP to learn the underlying relationships; therefore, it was chosen for this report.

2) *Feature Selection:* As demonstrated in a previous paragraph, multiple variables in the dataset are highly correlated. To increase the performance of the models and reduce the training time, the input features of the multilayer perceptron have been reduced. All features in the dataset are continuous variables; hence, the data is suitable for dimensionality reduction with a principle component analysis (PCA). A PCA is a statistical technique that simplifies complex datasets by reducing dimensionality, enabling the identification of dominant patterns and trends [3]. A PCA achieves this by transforming original variables into uncorrelated principal components, which capture the maximum variation in data. In the case of the dry beans dataset analysed in this report, the PCA was set to capture 99% of the variance in the original dataset. Below, the cumulative explained variance is shown, indicating that six main components can explain 99% of the variance.

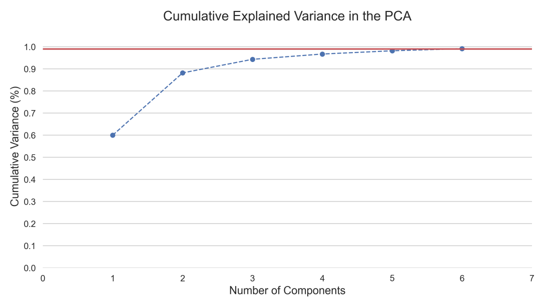

Figure 3: Cumulative explained variance for the PCA performed on the dataset.

After the feature selection, the dataset was split into training, validation, and testing data, with a 90% training, 5% validation, and 5% testing distribution.

3) *Architecture:* For this classification task, a simple neural network architecture was selected. Following the results of the PCA, the dataset with reduced dimensions had six standardised features. Hence the neural network had six input nodes. Two hidden layers were deployed with 16 and 8 nodes, respectively, and the output layer had seven output neurons—one for each class in the dataset. In line with modern neural network architectures, each layer used the Rectified Linear Unit (ReLU) function for the activation. The network output was constrained between 0 and 1 with a softmax layer. This was done so that the network outputs interpretable percentage values of the classification confidence [2]. A schematic overview of the network architecture is displayed below.

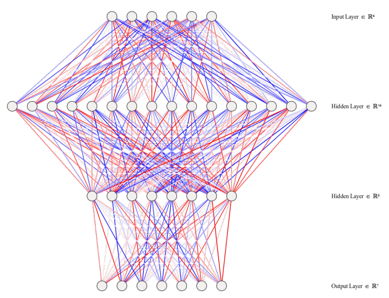
Input Layer ∈ $\mathbb{R}^6$
Hidden Layer ∈ $\mathbb{R}^{16}$
Hidden Layer ∈ $\mathbb{R}^8$
Output Layer ∈ $\mathbb{R}^7$

Figure 4: Neural Network Architecture for the classification of the dry beans.

4) *Loss Function and Optimization:* The neural network used in this classification used the categorical cross-entropy loss to calculate the forward pass, thereby calculating the predictions the model makes on the training data.

$$L_{\mathrm{CE}}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log\left(\hat{y}_i\right) \tag{2}$$

In the formula above, y_i is the true label for training example i, $\hat{y}_i$ is the prediction of the model for training example i, and C is the number of classes. For the optimisation of the model, Adam was used with a weight decay value of 10^{-8}.

$$m_t = \beta_1 = m_{t-1} + \left(1 - \beta_1\right) g_t \tag{3}$$

$$v_t = \beta_2 v_{t-1} + \left(1 - \beta_2\right) g_t^2 \tag{4}$$

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{5}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{6}$$

$$\theta_t = \theta_{t-1} - \alpha \frac{\widehat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{7}$$

The Adam optimiser employs a two-fold strategy to adjust the weights in a neural network during training [4]. Firstly, it calculates the gradients of the loss function with respect to the parameters and updates biased estimates of the gradient's mean (first moment) and variance (second moment). This is influenced by both the current gradient and prior estimates, creating a momentum-like effect. Secondly, these biased estimates are corrected based on the number of performed updates, which counteracts the initial bias towards zero. Finally, the model parameters are updated by subtracting a scaled version of the bias-corrected mean from the parameters, normalised by the square root of the bias-corrected variance (plus a small constant for numerical stability). This combination of Momentum optimisation and RMSProp techniques makes Adam popular for training neural networks.

5) *Hyperparameter:* Using a neural network for classification requires some search for the optimal hyperparameter of the network. Hyperparameters are the network parameters set before the model's training and determine key aspects of the learning process. This paper used a grid search to find the optimal values for the following parameter: learning rate and batch size. The table below lists this experiment's learning rates and batch sizes. Each learning rate was tested with each batch size.

Learning Rate	**Batch Size**
1e-6 / 0.000001	16
1e-5 / 0.00001	32
5e-5 / 0.00005	64
1e-4 / 0.0001	256

The parameters above have been selected following the recommendation in the literature [5]. Lower learning rates should, in general, lead to slower convergence. However, too large learning rates might lead to an unstable model and no convergence.

In addition, to the learning rate, two more hyperparameters were set in advance. The training loop was restricted to 50 epochs, and early stopping was implemented with a patience variable of 5. Early stopping is a technique that efficiently prevents overfitting to the training dataset. At each iteration of the training process, the training loss is evaluated, and if an improvement has been made, the new parameters are stored. Otherwise, the old parameters are kept in memory. If the model starts to overfit to the training data - measured in an increase in the validation set loss - the training loop is exited, and the model is restored with the parameters that lead to the lowest loss. In conclusion, early stopping protects the model from overfitting and reduces the dependence of the model on a well-selected number of training epochs.

6) *Success Metrics:* The model performance was assessed with multiple metrics. The accuracy, recall, precision and f1 score were used to determine the model's strength. Accuracy, is defined as the proportion of correct predictions over all predictions.

$$\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{TN}+\mathrm{FN}} \tag{8}$$

Here TP stands for true positive, TN true negative, FP false positive and FN false negative. Second, Recall, or Sensitivity, quantifying the model's proficiency in identifying all pertinent instances, is computed as the ratio of True Positives to the sum of True Positives and False Negatives. Third, Precision, a measure of the fraction of valid positive instances among those declared as positive, is determined by the formula

$$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{9}$$

Lastly, the F1 score, representing the harmonic mean of Precision and Recall, serves as a balanced measure incorporating both False Positives and False Negatives into its computation and is particularly useful for imbalanced datasets, like the dry beans dataset [6].

$$\mathrm{F1} = 2 * \frac{\mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{10}$$

E. Boosting Algorithm XGBoost

1) *Model Introduction:*

Boosting algorithms constitute a class of ensemble learning methods employed in classification tasks to enhance the performance of weak classifiers by combining them into a more robust classifier. These algorithms iteratively train a sequence of weak classifiers while each subsequent classifier focuses on correcting the misclassified instances from the previous iteration.

Extreme Gradient Boosting (XGBoost) is an advanced and highly efficient implementation of gradient boosting that has gained widespread recognition in machine learning competitions and applications [7]. XGBoost employs a regularised learning framework, incorporating both L1 and L2 regularisation terms to control model complexity and mitigate overfitting. This framework distinguishes XGBoost from other boosting algorithms and contributes to its exceptional performance.

2) *Data Preprocessing:* In contrast to the Multilayer perceptron, only light preprocessing was done for the boosting algorithm. The duplicates were removed, but no other preprocessing was conducted on the data. Standardisation becomes essential when the input data set's features exhibit significant discrepancies in their ranges or when they're quantified in different units, like pounds, meters or miles. Tree-based methods, including decision trees, random forests, and gradient boosting, aren't influenced by the scale of variables [7]. Hence, it's unnecessary to standardise variables before employing these models, which were omitted in this experimental setup. For the boosting algorithm, the dataset was split into training and testing data with an 85/15 split ratio.

3) *Hyperparameters:* Grid search was used to find the optimal parameter for the boosting algorithm. The parameters under consideration were the number of estimators, the number of tree depths, the regularisation parameter gamma and the learning rate. To find the optimal parameter, a grid search was conducted with the following parameter values:

Parameters	**Values**
Estimators	100,200,500
Tree Depth	3,6,9
Gamma	0.01, 0.1
Learning Rate	0.001,0.01,0.1,1

The model then used the optimal parameters to classify the beans in the dataset.

4) *Success Metrics:* The model performance was assessed with the same metrics as the multilayer perceptron.

III. Results

The following section describes the results of the two machine learning techniques on the dataset.

A. Results of the Multiplayer perceptron

First, the results for the hyperparameter search are presented in the following. The classification results for the best parameters are presented. The hyperparameter search indicated only marginal differences between batch sizes and learning rates. The following table shows the accuracy for the various combinations of learning rates and batch sizes.

LR/BZ	**16**	**32**	**64**	**256**
1e-6	0.9167	0.9304	0.9187	0.9108
1e-5	0.9236	0.9167	0.9147	0.9098
5e-5	0.9147	0.9177	0.9128	0.9138

1e-4	0.9255	0.9255	0.9167	0.9177
1e-2	0.9255	0.9275	0.9059	0.9128

Looking at the data, it appears that smaller batch sizes lead to marginally better results. In addition, the data above shows that the optimal parameter values for the learning rate and batch size were $1e^{-6}$ and 32, respectively. The model quickly reached very high accuracy levels in the training process, as shown in Figure 5.

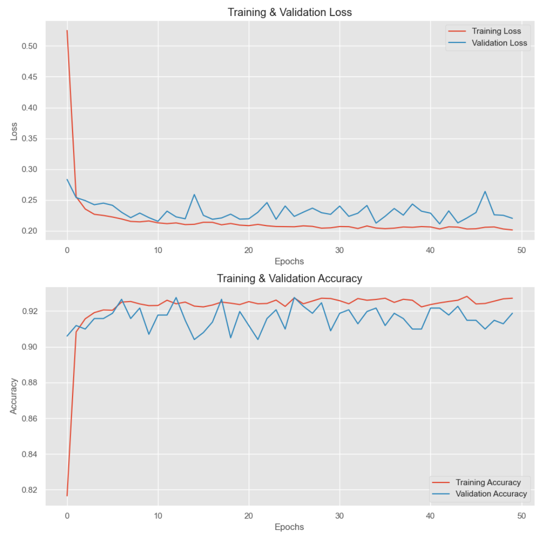

Figure 5: Training and Validation loss and accuracy. The model was able to learn the underlying relationships rapidly.

For the performance metrics, a confusion matrix was constructedA confusion matrix was constructed, and an overview report for accuracy, precision, recall and the f1 score was created.

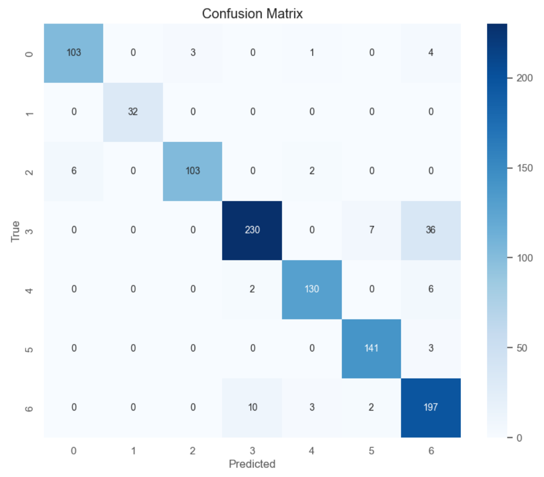

Figure 6: Confusion Matrix for the multilayer perceptron.

Metrics	**Results**
Accuracy	0.9304
Recall	0.92
Precision	0.92
F1 Score	0.92

The results indicate that the model was generally very good at classification. However, especially class 6 and class 3 showed some heightened classification error rates. In the next section, the results for the boosting algorithm XGBoost are presented.

B. *Results for the Boosting Algorithm*

For the xgboost model, a grid search for the best parameter yielded the following best parameter results.

Parameter	**Values**
n estimators	200
maximum depth	9
gamma	0.1
learning rate	0.1

The performance metrics and confusion matrix for the boosting algorithm are displayed below.

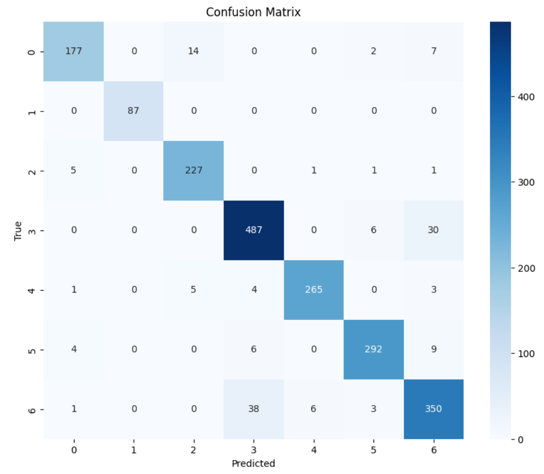

Figure 7: Confusion Matrix for the XGBoost boosting algorithm.

Metrics	**Results**
Accuracy	0.9277
Recall	0.94
Precision	0.94
F1 Score	0.94

Similarly to the MLP, the boosting algorithm was very good at classifying the dataset, reaching an accuracy of approximately 93%. In the same fashion as the MLP, the algorithm struggled the most with classifying the data in class 6 and class 3.

IV. Discussion

This work looked at two different machine learning techniques to classify dry beans concerning other aspects of their appearance. In the following discussion, the results are summarised again and then discussed.

A. Approach Comparison

The performance of both methods is compared, and context is provided for the position of this report in the broader literature [1, 8, 9]. The classification with xgboost and the multilayer perceptron yielded exceptional results that fit other developments in the literature. In both cases, the peak accuracy was around 93%. This is much higher than the results of the naive zeroR model; hence, both techniques added value to the classification task. However, looking at both processes, the author believes boosts should be preferred over the neural network approach. The neural network required much more data preprocessing steps and fine-tuning of the network's hyperparameter. In the case of the neural network, any implementation would need to consider the architecture of the network, batch size, learning rate, patience and epoch. While some of these hyperparameters are less impactful if early stopping is implemented (i.e. the number of epochs), they still require time to implement and fine-tune. In contrast, xgboost requires fewer parameters and fewer data preprocessing, as normalisation and standardisation are no concerns using this technique, and the ensemble nature of the algorithm allows it to find suitable solutions with greater ease.

In addition, the training time for the xgboost algorithm was much faster, with the average runtime for the xgboost algorithm lying around a minute and the MLP running for 2-3 minutes. With both methods providing equally good performance on the classification task, any application within an industry context should therefore be expected to prefer the boosting algorithm over the neural network approach. However, other methods might be advantageous over both presented methods, and XGBoost should not be seen as a panacea for all poisons.

B. Wider Literature

Searching for other publications using the underlying dataset [1], only one publication could be identified [8]. Similarly to this publication, the authors used a deep learning model to classify the dataset provided by Koklu and colleagues. Their performance metrics were very similar to the performance in this report. However, they used a different activation function (i.e. sigmoid) instead of the ReLU used in this publication. In addition, they used dropout layers instead of early stopping to prevent overfitting. The differences show that both approaches result in very high classification accuracy. However, the results from this study highlight the advantages of tree-based boosting algorithms for simple classification tasks. It appears, therefore, that overengineering the problem is - as it is usually - not the best approach and that established methods with the lowest complexity are usually the best way to go forward.

C. Future Research

While the two machine learning approaches discussed in this paper have yielded promising results in the classification task, other techniques could also lead to good results. Logistic regression and different clustering algorithms should also perform well in the classification of the dry beans dataset. These solutions have fewer parameters and could be even more efficient than boosting algorithms and neural networks. In addition, this publication did not stratify the dataset to account for imbalances in the data distribution. Future work could use different sampling techniques for a more balanced data distribution. This could allow for better results for the MLP as it showed the most significant error rate for class 6 (Bombay), which was underrepresented in the dataset.

V. Conclusion

Both approaches discussed and analysed in this report performed much better than the baseline model. While the performance was comparable between the multilayer perceptron and the boosting algorithm, the boosting algorithm required less work to implement. It was deemed the better solution for real-world classification applications.

References

[1] M. Koklu, and I. A. Ozkan, "Multiclass classification of dry beans using computer vision and machine learning techniques," *Comput. Electronics Agriculture*, vol. 174, p. 105507, Jul. 2020, doi: 10.1016/j.compag.2020.105507. Accessed: Apr. 18, 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0168169919311573

[2] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," 2023.

[3] I. Jolliffe, and J. Cadima, "Principal component analysis: a review and recent developments," vol. 374, 2016, p. 20150202.

[4] S. Ruder, "An overview of gradient descent optimization algorithms," 2017.

[5] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," 2012.

[6] G. Forman, and M. Scholz, "Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement abstract," *SIGKDD Explorations*, vol. 12, pp. 49–57, 2010.

[7] T. Chen, and C. Guestrin, "XGBoost," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, doi: 10.1145/2939672.2939785. [Online]. Available: https://doi.org/10.1145%2F2939672.2939785

[8] V. Gautam, N. K. Trivedi, A. Anand, and J. Rani, "Feature selection based dry-beans multiclass classification with optimized deep neural network," in *2022 10th Int. Conf. Reliability, Infocom Technologies Optim. (Trends Future Directions) (Icrito)*, vol. 0, 2022, pp. 1–5, doi: 10.1109/ICRITO56286.2022.9964754.

[9] M. Hasan, M. U. Islam, and M. Sadeq, “A deep neural network for multi-class dry beans classification,” 2021, pp. 1–5, doi: 10.1109/ICCIT54785.2021.9689905.