Evaluating Multilayer Perceptrons and Support Vector Machines in Dry Bean Classification

Andreas Ioannides
Andreas.Ioannides@city.ac.uk

Abstract

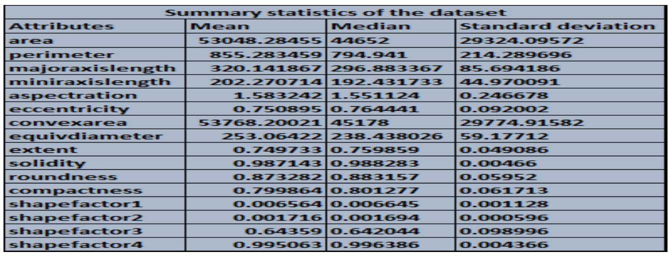
The research paper explores the classification of dry beans using support vector machines and multilayer perceptrons. The study aims to construct an optimal model for both algorithms, to classify dry beans in their respective classes. These techniques can help agricultural specialists such as farmers meet market demands, and strive for better quality, and more efficient productivity of agricultural products, ultimately contributing to food security and supply chains.

1. Introduction

A diversified understanding of agricultural products is vital for ensuring food security and optimizing supply chains. Dry beans are a common meal globally, emphasizing the unique food culture and identity of many countries. Beans allow individuals to maintain a healthy and balanced diet since they are rich in protein, and fiber. [1] They are also considered antioxidants and anti-inflammatory leading to a lot of health benefits.[1] Beans can significantly lower the risk of cancer and diabetes that many people suffer daily during their lifetimes.[2] However, a plethora of factors such as genetic diversity, and farming techniques can greatly impact the quality and health benefits of dry beans, leading to lower market value and consumption. The purpose of this research paper is to analyze and evaluate two neural networks, specifically multilayer perceptrons and support vector machines, aiming to develop an optimal model for each algorithm, to accurately classify beans in their respective classes. We hope to bridge the gap between traditional agricultural techniques and current technological advancements, to ensure a more efficient production of dry beans and a better understanding of crop characteristics.

2. Dataset

The dataset is publicly available on Kaggle and consists of 13611 samples of dry beans and includes 17 columns, each representing a different attribute. The dataset has no missing values and Dry beans are classified as Seker, Barbunya, Bombay, Cali, Horoz, Sira, and Dermason. From Table 1 we observe that while there is considerable variability in bean sizes, shape-related attributes are a lot more consistent. This demonstrates that even though bean dimensions vary, the proportional measures remain stable.

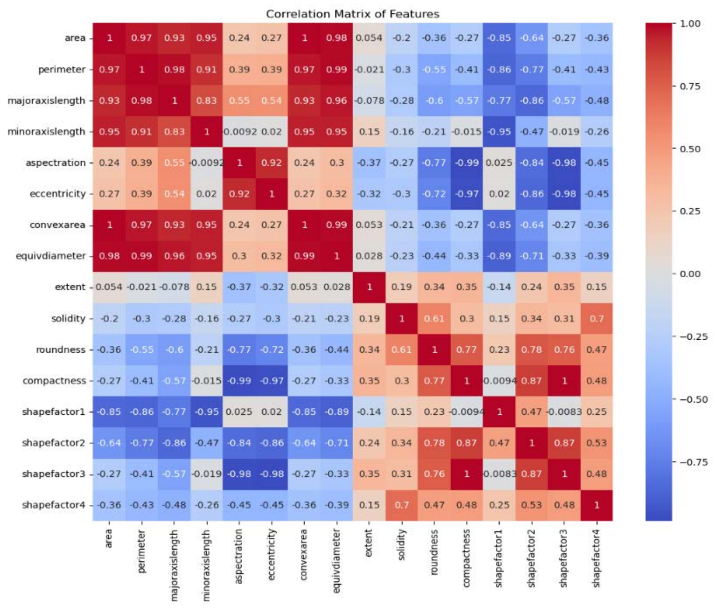


Summary statistics of the dataset			
Attributes	Mean	Median	Standard deviation
area	53048.28455	44652	29324.09572
perimeter	855.283459	794.941	214.289696
majoraxislength	320.141867	296.883367	85.694186
miniraxislength	202.270714	192.431733	44.970091
aspectration	1.583242	1.551124	0.246678
eccentricity	0.750895	0.764441	0.092002
convexarea	53768.20021	45178	29774.91582
equivdiameter	253.06422	238.438026	59.17712
extent	0.749733	0.759859	0.049086
solidity	0.987143	0.988283	0.00466
roundness	0.873282	0.883157	0.05952
compactness	0.799864	0.801277	0.061713
shapefactor1	0.006564	0.006645	0.001128
shapefactor2	0.001716	0.001694	0.000596
shapefactor3	0.64359	0.642044	0.098996
shapefactor4	0.995063	0.996386	0.004366

Table 1. Summary statistics of the dataset

2.1. Initial Data Analysis

As part of the initial data analysis process, we generate a correlation matrix for all the attributes, as shown in Figure 1. A correlation matrix helps visualize the relationship between variables. We can see a very high positive correlation in features like area, perimeter, axis length, and equivalent diameter, showing that more elongated beans are less compact. On the other hand, compactness and roundness have a strong negative correlation with aspect ratio and eccentricity indicating that more circular beans have a smaller value of these two metrics. Finally, shape factors also have a strong negative correlation, indicating a lack of information on these attributes. A bar chart showing the different frequency distributions of bean classes was also constructed, labeled as Figure 2. The Bombay class has significantly fewer samples than the other classes, indicating a class imbalance. Dermason is the most common bean class, and Bombay is the least common. Min-Max scaling was implemented when carrying out the MLP, however, SMOTE or data generation could have been performed to address the issue of class imbalance.



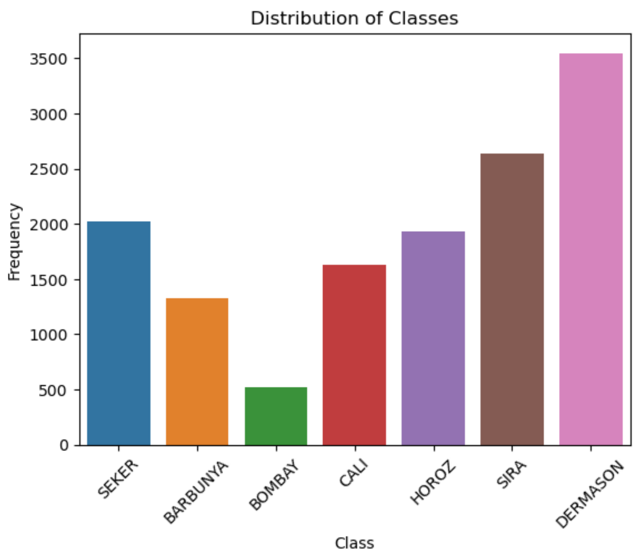


Fig 1. Correlation matrix of features

Fig 2. Bar chart of class distribution

3. Summary of Methods

3.1 Multilayer Perceptron

A multilayer perceptron is a supervised learning neural network consisting of at least 3 layers of nodes, an input layer, an output layer, and at least one hidden layer.[3] This makes the multilayer perceptron able to model nonlinear relationships and complex data. The input layer takes input features, the hidden layer enables the MLP to capture patterns in the data, and the output layer provides the classification output. The learning process involves forward propagation and backpropagation. Weight decay can also be included to improve generalization and reduce overfitting. The MLP is ideal for the dry beans dataset since it performs better when handling classification tasks and complex data with non-linear patterns.

Advantages
Multilayer perceptrons are flexible neural networks since they are very good at learning non-linear relationships and perform best during classification and regression tasks. The multilayer perceptron can learn and implement any function no matter how difficult or complex it is, because enough data must be present, alongside an appropriate network structure. The hidden layers of the MLP allow them to extract meaningful and hidden patterns in data that other algorithms might miss.

Disadvantages
Multilayer perceptrons can easily become complex once the number of hidden layers and nodes increases, which can lead to overfitting. Also, training the MLP usually takes a lot of time because a variety of hyperparameters have to be tuned and assigned values. Therefore, the MLP is not ideal if we are pressured by time since they become computationally expensive. MLP is not useful when we have temporal or sequential data. MLPs only perform to the best of their ability when trained perfectly, hence why they need expertise in the field to handle them.

3.2 Support Vector Machines

Support vector machines are a supervised learning algorithm, that can handle classification and regression tasks. SVM aims to find the ideal hyperplane that separates class labels[4]. Linear SVM does this by finding the maximum distance between data points of classes. Non-linear SVM uses the kernel trick to transform high-dimensional space without calculating data points. Instead, they calculate the products between data points. Some kernel examples are RBF and Sigmoid. Since our dataset includes geometrical and statistical attributes, SVM is effective in complex patterns and tasks. SVM is effective for the dry beans dataset since it involves many features.

Advantages
Support vector machines are very effective in high-dimensional spaces even when the dimensional space is higher than the sample[5]. SVM also performs well in small datasets. They can store memory and are computationally effective once trained[5].

Disadvantages
Support vector machines are not very effective when used in large datasets. They require expertise and careful parameter tuning to perform to the best of their ability, making them expensive and inefficient to train. SVM tends to underperform when dealing with noisy data.

4. Hypothesis Statement

In this study, we evaluate the effectiveness of multilayer perceptrons and support vector machines on a dry bean classification dataset. Since the SVM performs better when managing high-dimensional space and complex data, it should outperform the MLP, in evaluation metrics. However, the MLP can also capture complex data patterns due to its architecture, and provide reliable results, since many of its hyperparameters can be tuned.

Overall, we hope the advanced and computational models can set a new benchmark for bean classification since both models can give a variety of evaluation metrics.

5. Methodology

Initial data pre-processing involved checking for null values, which fortunately were none, and then continued with plotting box plots to identify outliers, pairwise plots to see the data distribution and a correlation matrix of the attributes. For both algorithms, we used the train test split function in Python, and the holdout method, to assign 80% of the data to the training set, and leave the other 20% to the test set. Starting with the SVM, we performed a grid search analysis to optimize critical parameters such as the regularization parameter C, and the kernel coefficient γ, and inspected both linear and non-linear support vector machines. For the MLP, we first standardized our variables using the min-max scaler. In terms of the architecture of the network, we used different combinations of hidden layers, learning rates, and constant epochs. Finally, to keep our comparison fair and consistent, we used the same evaluation methods. Examples include performance metrics, learning curves, confusion matrix, and ROC curve, calculating correct and incorrect classifications, and performing cross-validation to ensure that both models perform well across other subsets of the data.

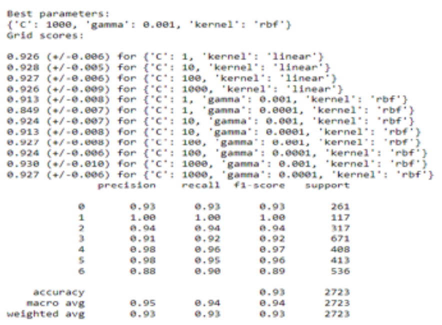
6. Choice of Parameters and Experimental Results

Starting with the SVM, we tested both linear and nonlinear SVMs. We first explored by setting the regularization parameter C, to 1, 10, 100, and 1000, to have a range of low-complexity and high-complexity models. We kept the kernel linear so the boundary between the classes remains a straight line. Then we kept the same regularization parameters but added a γ parameter consisting of 0.001, 0.0001 in order to test how tightly the SVM model fits to the training data. We specified the kernel to be RBF so that it can handle nonlinear data. In contrast, for the MLP, we constructed the network by setting a combination of hidden sizes to be [10,5]. [20,10], [30,20,10], set learning rates to be [0.1, 0.01, 0.001] and kept a constant epoch number of 1000. We also used the ReLU activation function to introduce non-linearity, the Adam Optimizer for its learning rate capabilities, and the Cross-Entropy loss which is effective for classification tasks since it calculates the difference between predicted probabilities and actual labels. Concerning experimental results, the nonlinear SVM provided high classification accuracy and better overall performance metric results. On the other hand, the MLP led to better performance since it showcased improved evaluation metrics across different class predictions.

7. Analysis and Critical Evaluation of Results

7.1 Model Selection

Figure 3 shows the best SVM, which is nonlinear, and has a regularization parameter of 1000 and a kernel coefficient of 0.001 alongside the optimal evaluation metrics. Similarly, Figure 4 shows the best MLP which consists of a learning rate of 0.001 and hidden sizes of 30, 20, and 10 respectively. We can also see the optimal evaluation metrics and the training time.

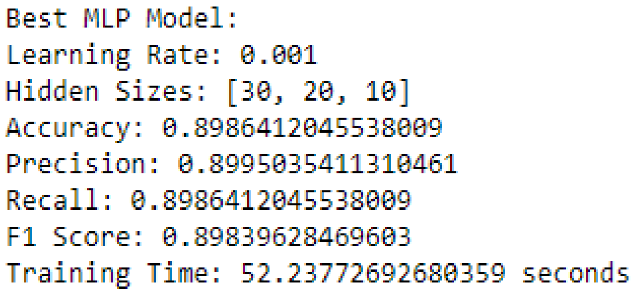


```
Best parameters:
{'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
Grid scores:

0.926 (+/-0.006) for {'C': 1, 'kernel': 'linear'}
0.928 (+/-0.005) for {'C': 10, 'kernel': 'linear'}
0.927 (+/-0.006) for {'C': 100, 'kernel': 'linear'}
0.926 (+/-0.009) for {'C': 1000, 'kernel': 'linear'}
0.913 (+/-0.008) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.849 (+/-0.007) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
0.924 (+/-0.007) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
0.913 (+/-0.008) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
0.927 (+/-0.008) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
0.924 (+/-0.006) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
0.930 (+/-0.010) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
0.927 (+/-0.006) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
              precision    recall  f1-score   support

           0       0.93      0.93      0.93       261
           1       1.00      1.00      1.00       117
           2       0.94      0.94      0.94       317
           3       0.91      0.92      0.92       671
           4       0.98      0.96      0.97       408
           5       0.98      0.95      0.96       413
           6       0.88      0.90      0.89       536

    accuracy                           0.93      2723
   macro avg       0.95      0.94      0.94      2723
weighted avg       0.93      0.93      0.93      2723
```



```
Best MLP Model:
Learning Rate: 0.001
Hidden Sizes: [30, 20, 10]
Accuracy: 0.8986412045538009
Precision: 0.8995035411310461
Recall: 0.8986412045538009
F1 Score: 0.89839628469603
Training Time: 52.23772692680359 seconds
```

Fig 3. Optimal SVM model

Fig 4. Optimal MLP model

7.2 Algorithm Comparison

To begin with, Figures 5 and 6 show the confusion matrix for the SVM and MLP respectively. We can see that the SVM has a higher overall accuracy due to a higher count in diagonal cells. The MLP seems to misclassify class attributes more often indicating issues with model handling or model capacity. The SVM performs better in the Bombay and Cali classes, whereas the MLP performs better in the Horoz class. Looking at the ROC curves in Figures 7 and 8, the SVM model indicates a significantly better performance in all classes as shown by the area under the curve. AUC values are at least 0.87 highlighting excellent model performance and model reliability. On the other hand, the MLP indicates poor performance with most classes below 0.5 and some even close to 0, indicating that the MLP performs no better than random chance. Overall, the SVM consistently calculates classes better than the MLP. From Figures 9, 10, and 11 we can conclude that for the SVM, the learning curves show good performance on the training data by consistently having a high training score and cross-validation accuracy, indicating a good generalization of the model. In contrast, for the MLP, we conclude that after a slow start, the model quickly adjusts to the training data across all learning rates. This highlights that the MLP requires careful hyperparameter tuning as its performance can be significantly influenced by the choice of learning rates. It also shows differences in loss, depending on the learning rate, with the lowest learning rate leading to the lowest loss. Figures 12 and 13 show correct and incorrect classifications of the SVM and MLP respectively. Considering the SVM model, it showcased a consistently higher number of correct classifications compared to incorrect classifications, with the highest number of correct classifications being in Cali. Both models had 0 incorrect classifications for Barbunyha showing good generalization capabilities since they learned effectively the key patterns and features of this class. Generally, the MLP also showcases a higher number of correct classifications. However, for classes such as Sira and Seker, the MLP has more incorrect classifications, making the SVM stronger in classifying dry beans.

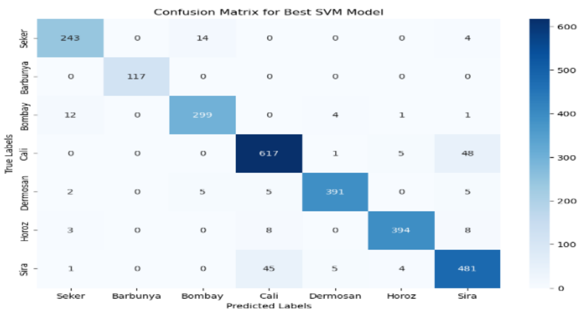


Fig 5. Confusion matrix of MLP

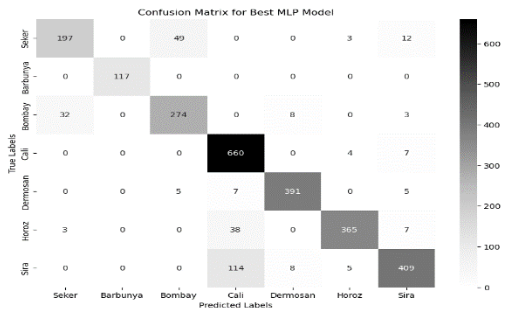
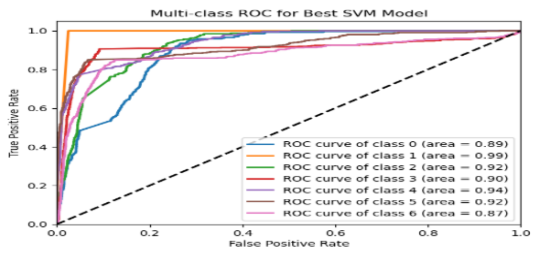


Fig 6. Confusion matrix of SVM



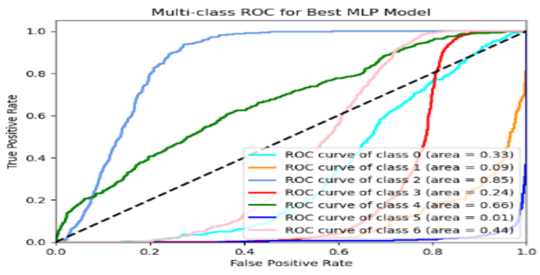
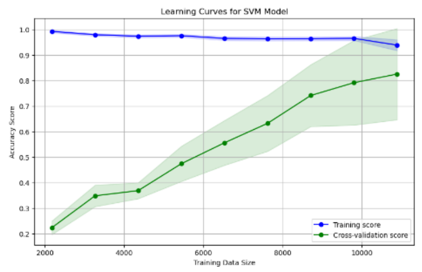


Fig 7. ROC curve for SVM

Fig 8. ROC curve for MLP



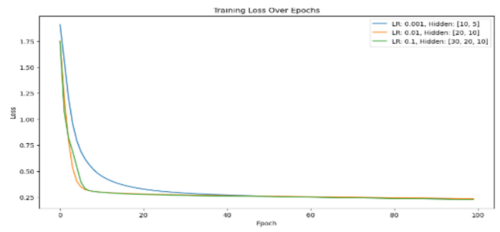


Fig 9. Learning Curves for SVM

Fig 10. Training Loss MLP

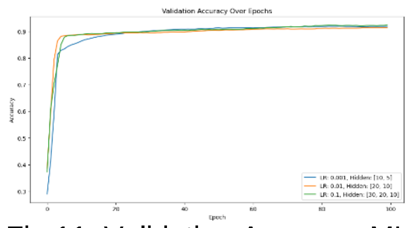


Fig 11. Validation Accuracy MLP

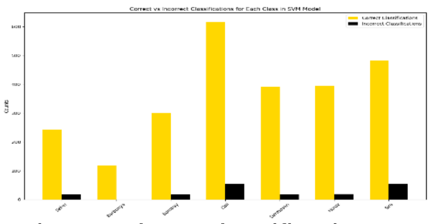


Fig 12. Class Classification SVM

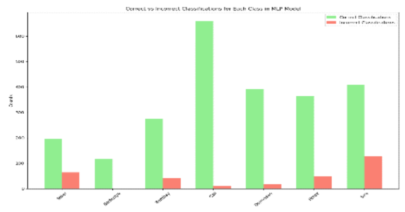


Fig 13 Class Classification MLP

8. Conclusion, Lessons Learned and Future Work

The study on dry bean classification concludes that the nonlinear SVM had better accuracy and consistently performed well across all bean classes which is supported by the variety of evaluation metrics. The SVM supported its advantage of handling high-dimensional data and complex patterns. The flexible architecture of the MLP captured nonlinear relationships and was promising in evaluation metrics, however, it was outperformed by the SVM as it was more accurate and precise. We understood the importance of rigorous hyperparameter tuning on model evaluations and comprehensive metrics. We learned that the MLP is sensitive to class imbalance and requires careful hyperparameter tuning and validation to achieve optimal results. For future work, implementing SMOTE could tackle class imbalance and potentially lead to better performance for the MLP. Additionally, introducing ensemble methods such as combining algorithm predictions, or extending the number of hyperparameters for the MLP may achieve higher accuracy and robustness in dry bean classification.

9. References

[1] "9 health benefits of beans." Accessed: Apr. 18, 2024. [Online]. Available: https://www.medicalnewstoday.com/articles/320192#benefits

[2] M. Jessimy, "13 Impressive Health Benefits of Beans," Natural Food Series. Accessed: Apr. 18, 2024. [Online]. Available: https://naturalfoodseries.com/13-benefits-beans/

[3] "Multilayer perceptron," *Wikipedia*. Mar. 29, 2024. Accessed: Apr. 18, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Multilayer_perceptron&oldid=1216115889

[4] "What Is Support Vector Machine? | IBM." Accessed: Apr. 18, 2024. [Online]. Available: https://www.ibm.com/topics/support-vector-machine

[5] "Support vector machine in Machine Learning," GeeksforGeeks. Accessed: Apr. 18, 2024. [Online]. Available: https://www.geeksforgeeks.org/support-vector-machine-in-machine-learning/

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

Glossary

Multilayer perceptron: A supervised neural network mainly used to learn and model nonlinear relationships. It has different activation functions and learning rates that can be manipulated.

Support vector machine: A supervised learning algorithm that can be used in regression and classification. It finds the ideal hyperplane by calculating the maximum distance of data points.

ROC curve: An evaluation metric where the graph consists of true positive values on the x-axis and false positives on the y-axis. It is used to illustrate the effectiveness of binary classifiers. The closer the area under the curve is to one, the more effective the classifier.

Learning rate: An example of a hyperparameter of the multilayer perceptron. It controls how much the model needs to be changed in response to the estimated error function.

Intermediate results including negative results

The SVM generally outperformed the MLP in overall accuracy as well as most other evaluation metrics in terms of classifying dry beans. When increasing the learning rate for the MLP, its performance had a decline, and even showed effects of overfitting in some configurations. The Bombay bean class had a significant amount of fewer samples, in comparison to the rest of the classifiers, indicating a class imbalance. This class imbalance harmed the performance of the MLP. Furthermore, whilst both models performed relatively well, it is clear that they need careful and extensive hyperparameter tuning for them to achieve good results.

Implementation details including main implementation choices

Both models used the train and test split method using the Python programming language, assigning 80% of the data to the training set, and the remaining 20% of the data to the test set. Both models had the same evaluation methods which consisted of a confusion matrix, performance measures, ROC curve, learning rates, loss and accuracy graphs, and class imbalance bar charts. This allowed for a fair comparison of both models. For the SVM, we used a grid search to optimize hyperparameters such as the regularization parameter, kernel coefficient, and kernel type which included both linear and nonlinear SVM. For the MLP we used min-max scaling to try and maximize model performance, and then manipulated learning rates and hidden size options We also utilized the Adam's Optimizer and the ReLU activation function to find the optimal model.

Extra graphs and visualizations

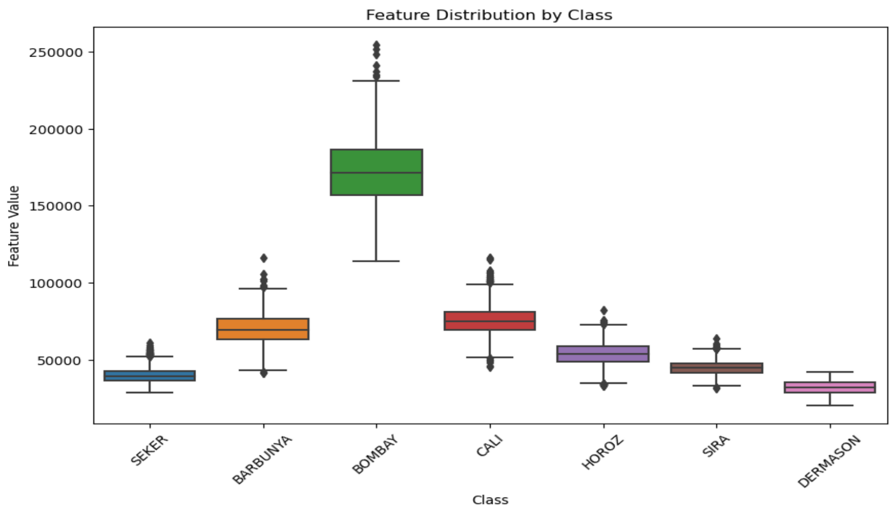


Fig 14 Box Plot of bean distribution across classes

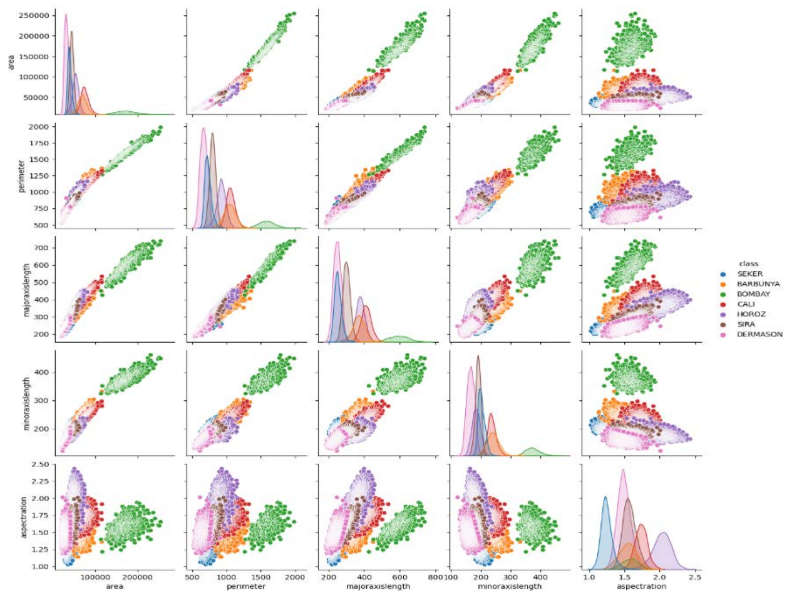


Fig 15 Pairwise Relationships and Feature Distributions of bean classes