CLASSIFYING DRY BEANS USING COMPUTER VISION SYSTEM (CVS) DATA

Kshitij Pathak, MSc. Data Science, University of Sussex
kp427@sussex.ac.uk, Reg no: 260873

I. Abstract

Machine learning is widely used in various agriculture-related fields such as optimizing crop cycles by utilizing weather models and identifying crop diseases using ML techniques, etc. This research investigates the dry beans data set produced by Koklu et.al [1] and attempts to present a classification model which can discriminate among beans for the purpose of segregation in agricultural applications like separating seeds before sowing to maintain crop purity.

II. Introduction

Image processing and machine learning methods find several applications in the agriculture domain, for example in crop management, disease detection, yield prediction, and soil analysis, among others. This paper discusses machine learning algorithms that can classify the type of dry beans using the beans dataset prepared by Koklu et.al [1] at UC Irvine, for their research on the same topic. There are a total of 13,611 samples that describe 16 distinct features belonging to 7 varieties of beans. The image data were collected using a computer vision system (CVS) and subsequently transformed into 16 numerical features which describe the physical dimensions and shape of the beans. Segregating beans is an important activity in farming because different types of beans have different characteristics, such as size, shape, color, and texture, which can affect their quality and suitability for different uses. For example, some types of beans may be better suited for human consumption, while others may be more suitable for animal feed or use in producing other products. Also, each variety of beans may require different growing conditions therefore it is necessary to segregate beans before sowing to ensure better quality and yield. The proposed model can be adapted to address these issues by mechanizing the process while ensuring near-human accuracy.

The further sections outline the approach in detail, firstly describing the data in an elaborated fashion, identifying any shortcomings in the dataset, exploring the features to identify relevant candidates for modeling, addressing data sanity issues like outlier detection and treatment, and transforming features, etc. We will try to analyse the features with the aim of identifying the best features for modeling and appropriate techniques suitable for the given data. Finally, the document compares the implemented classification algorithms and proposes the most suitable technique for the given problem while contrasting other available techniques and comparing them with existing work while providing suggestions for further improvement.

III. Dataset Description and Exploration

The dataset contains 13,611 samples with 17 columns out of which 16 are feature columns and one represents the class. 12 out of 16 features are dimensional in nature like area, perimeter, and length, etc. the remaining 4 determine the shape features [1]:

1. **Area (A):** The 2D area of a bean zone and the number of pixels within boundaries.
2. **Perimeter (P):** Bean circumference.
3. **Major axis length (L):** Length of the longest vertical line that can be drawn within the bean area.
4. **Minor axis length (l):** Longest line that can be drawn while standing perpendicular to the main axis.
5. **Aspect ratio (K):** the relationship between L and l.
6. **Eccentricity (Ec):** Eccentricity of the ellipse having the same moments as the bean region.
7. **Convex area (C):** Number of pixels in the smallest convex polygon that can contain the area of a bean seed.
8. **Equivalent diameter (Ed):** The diameter of a circle that has the same area as the bean zone.
9. **Extent (Ex):** The ratio of the pixels in the bounding box to the bean area.
10. **Solidity (S):** Also known as convexity. The ratio of the pixels in the convex shell to those found in beans.

11. **Roundness (R):** Calculated with the following formula: (4piA)/(P^2)
12. **Compactness (CO):** Measures the roundness of an object: Ed/L
13. **ShapeFactor1 (SF1)**
14. **ShapeFactor2 (SF2)**
15. **ShapeFactor3 (SF3)**
16. **ShapeFactor4 (SF4)**
17. **Class** (Seker, Barbunya, Bombay, Cali, Dermosan, Horoz and Sira)

The dataset is limited in terms of capturing the types of features. More features that can be informative in model development can be included to improve performance and robustness like the color, weight, 3-dimensional features of the beans, etc.

There are a sufficient number of observations to build a classification model however we need to check the distribution of data by class. Below pie-chart shows the same:

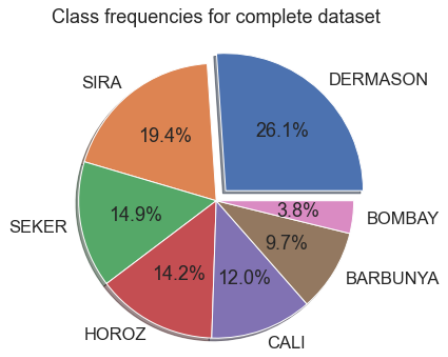


Figure 1: Class frequency distribution

The graphic shows that the data is not uniformly distributed among the classes. The highest proportion of data belongs to the "Dermason" class at 26.10% whereas the "Bombay" variety has just 3.80% representation. This may lead to poor performance (low accuracies and high error rates) for the underrepresented class.

As preliminary data analysis, checks were conducted to determine the data quality. A quick glance at the dataset columns shows that the overall quality of data is satisfactory. There is no missing data, therefore, no need for data imputation. No data point seems to be faulty or misleading therefore there is no need for any cleaning at this stage. The descriptive statistics illustrate that the features belong to different scales and therefore will be normalised for use in further steps, otherwise, it can cause issues such as the dominance of the feature with a higher scale, and overall poor performance of the model. Also, scaling the data helps the algorithm to converge faster.

Next, as a part of the feature selection exercise, correlation analysis was performed to drop highly correlated features with redundant information that might lead to multicollinearity and model performance.

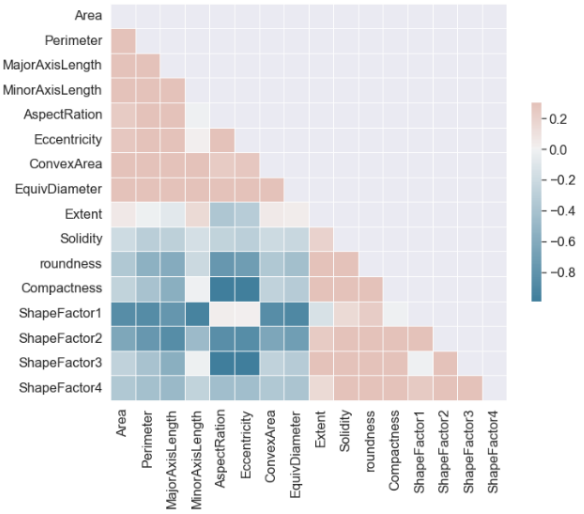


Figure 2: Correlation Analysis

Seven features (namely: Compactness, Shape Factor 1, Shape Factor 2, Shape Factor 3, Area, Aspect Ratio, and Eccentricity) with correlation in access to ~50% were dropped from further analysis, keeping only 9 features.

After shortlisting the features, boxplots were created to visualize the distribution of the dataset and to identify outliers if any:

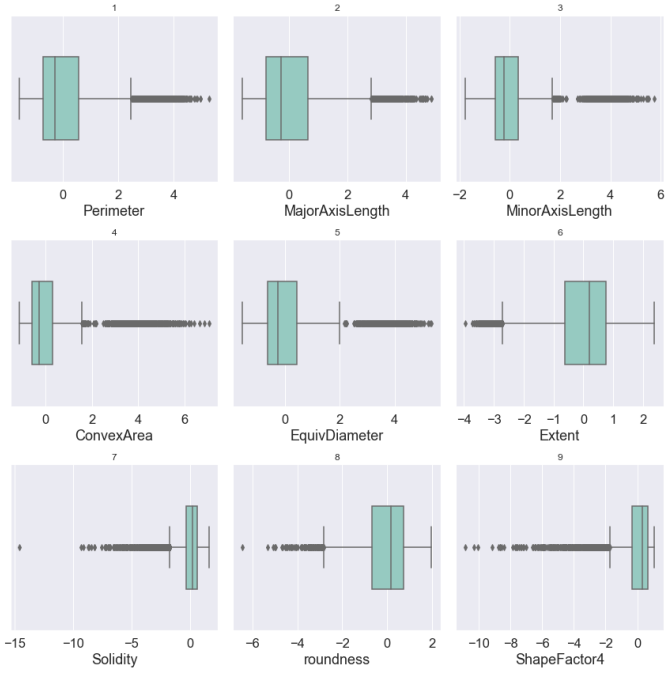


Figure 3: Feature Boxplots

It can be observed that outliers (values lying outside the 1.5 * interquartile range) are present in all the features. If we choose to strictly remove all the outliers, we may be left with too few instances and perhaps even fewer for

classes with low representation. For this reason, we will skip outlier elimination for now and may come back to this step if models perform poorly. Next, a Pairplot was used to identify relationships among variables and the features which can help in better classification:

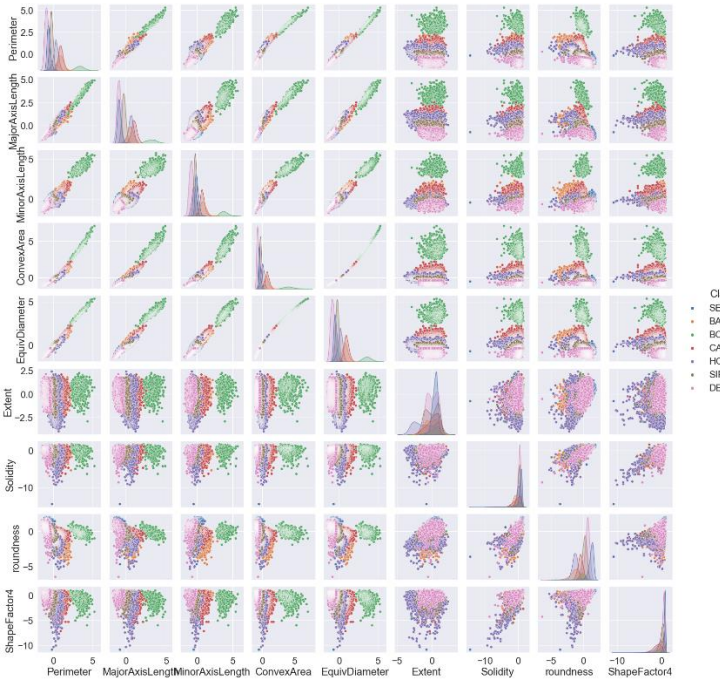


Figure 4: Feature Pairplot

"Bombay" seeds (in green) seem to be more distinguishable as compared to other classes. Furthermore, most features seem to be normally distributed. Shapiro-Wilk normality test confirms that most features are normally distributed across each class therefore we can assume normality and use Gaussian Naïve Bayes algorithms to create the first classification model which can act as a benchmark for further development. The next sections discuss the models employed, the rationale behind using each model, and classification results.

IV. Modeling Methodology

Before beginning with the modeling task, the data set is divided into two samples the training set, which will be used to train the model and the testing set which will be used to determine the performance of the model. The train-test split proportion is kept at 80:20 as we have sufficient data, and this is the commonly used ratio.

I. Naïve Bayes Classification

Naive Bayes is a probabilistic classification algorithm that is based on Bayes' theorem. Each feature is considered independent of all other features when predicting the class. This is known as the "naive" assumption. This is one of the shortcomings of the model but also makes it very easy to implement.

The mathematical formula for Naive Bayes:

> ***P(y|x1, x2, ..., xn) = P(y) * P(x1|y) * P(x2|y) * ... * P(xn|y) / P(x1, x2, ..., xn)***
>
> *where:*
> ***P(y|x1, x2, ..., xn)*** *is the probability of class y given the features x1, x2, ..., xn.*
> ***P(y)*** *is the prior probability of class y.*
> ***P(xi|y)*** *is the conditional probability of feature i given class y.*
> ***P(x1, x2, ..., xn)*** *is the probability of observing features x1, x2, ..., xn.*

Naïve Bayes classification model gives satisfactory output with **86.67%** accuracy which is close to human accuracy in segregation. Below is the confusion matrix for the same:

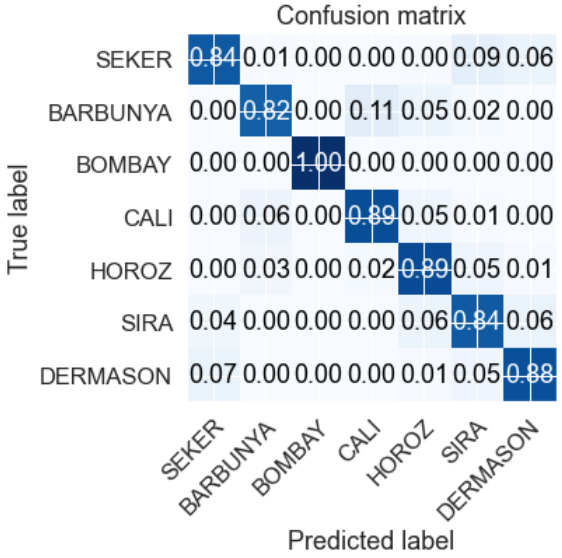


Figure 5: Naive Bayes- Confusion Matrix

The highest classification accuracy is obtained for Bombay followed by Cali, Horoz, and the remaining.

II. Random Forest Classification

Random forest is an ensemble learning method for classification. The random forest classifier builds multiple decision trees, each based on a randomly sampled subset of the training data and a randomly sampled subset of the features, and finally combines the predictions of all trees to obtain a final prediction. With the number of trees set at 100, this approach gives a classification accuracy of 92.91% which is impressive and close to human accuracy. This method can be understood intuitively and is easy to implement as there are not many hyperparameters to be tuned. One limitation of using this approach here is that this methodology is known to work better with categorical features rather than

continuous features which we have. Also, Random forests can perform poorly when the data is imbalanced, which is the case here.

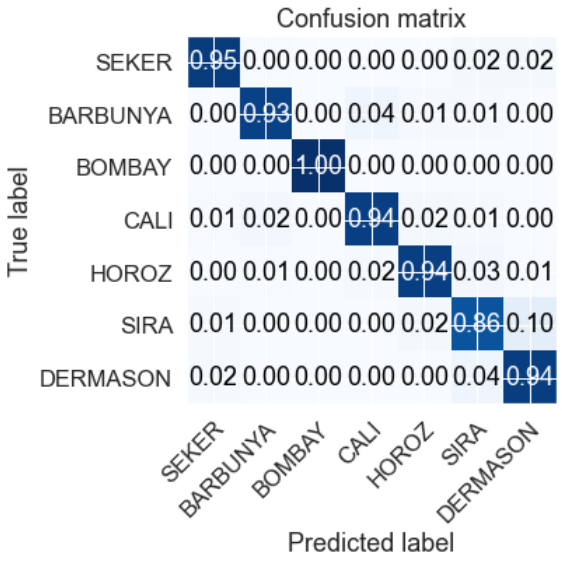


Figure 6: Random Forest- Confusion Matrix

III. Support Vector Machines (Additional Model / Challenger model)

The SVM classifier is introduced as the challenger model to the random forest model. The SVM algorithm uses hyperplanes to separate data points belonging to different classes. These hyperplanes are equidistant from the support vectors of the classes. SVM works well for high-dimensional datasets and can handle non-linearly separable data using kernel functions. The choice of kernel functions therefore can affect the learning outcome. SVM may perform better when working with continuous data in contrast to random forest algorithm.

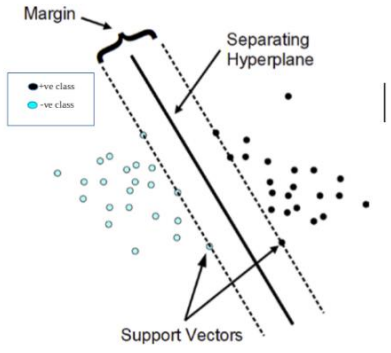


Figure 7: SVM Hyperplane visualisation

Two kernels were implemented for training the SVM model. The linear kernel gives classification accuracy closer to the random forest at 92.8%. However, the radial basis kernel function outperforms both with an accuracy of 93.08%, which is the highest among all the classification algorithms applied. The following confusion matrix illustrates the classification accuracies for the SVM classifier with RBF kernel.

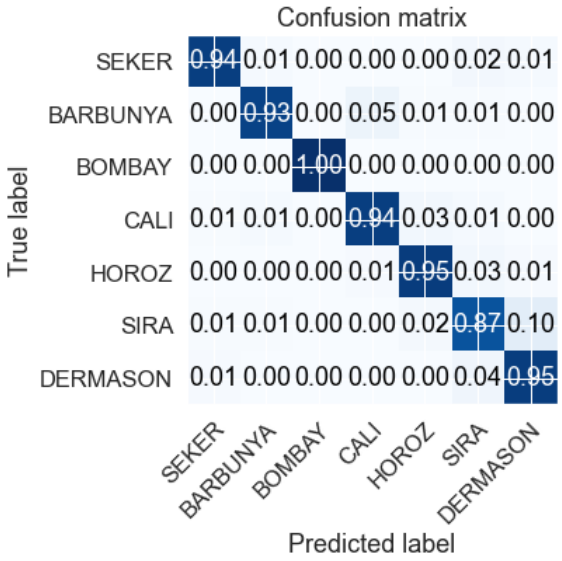


Figure 8: SVM- Confusion Matrix

V. Results and Discussion

The following evaluation metrics were used to judge the performance of each of the models:

1. **Precision**: The ratio of true positives (TP) out of the total predicted positives (TP + FP). It measures the accuracy of the positive predictions. [4]
2. **Recall**: The fraction of true positives (TP) to total actual positives (TP + FN). It measures the ability of the model to identify all positive instances. [4]
3. **F1-Score**: The harmonic mean of precision and recall. It provides a single score that balances both precision and recall and ranges from 0 to 1. The F1-score is a measure of the overall accuracy of the model. [5]
4. **Support**: Support refers to the actual number of samples in each class. [6]

The below tables compare the performance of the three models described earlier:

Class	Precision	Recall	F1-Score	Support
SEKER	0.84	0.84	0.84	406
BARBUNYA	0.86	0.82	0.84	265
BOMBAY	0.99	1.0	1.00	104
CALI	0.89	0.89	0.89	326
HOROZ	0.83	0.89	0.86	386
SIRA	0.83	0.84	0.83	527
DERMASON	0.91	0.88	0.89	709

Figure 9: Naive Bayes- Performance Metrics

The Gaussian Naïve Bayes algorithm performs much better than a random model which would give an accuracy

of (1/7) * 100 = 14.29%. It gives an average accuracy score of 86.67% which is much higher than a random model. Moreover, it performs quite well for Bombay beans with an F-1 score of 1.

Class	Precision	Recall	F1-Score	Support
SEKER	0.95	0.95	0.95	406
BARBUNYA	0.94	0.93	0.94	265
BOMBAY	0.99	1.0	1.00	104
CALI	0.94	0.94	0.94	326
HOROZ	0.95	0.94	0.94	386
SIRA	0.9	0.86	0.88	527
DERMASON	0.91	0.94	0.93	709

Figure 10: Random Forest- Performance Metrics

Class	Precision	Recall	F1-Score	Support
SEKER	0.96	0.94	0.95	406
BARBUNYA	0.95	0.93	0.94	265
BOMBAY	1.0	1.0	1.00	104
CALI	0.94	0.94	0.94	326
HOROZ	0.94	0.95	0.95	386
SIRA	0.89	0.87	0.88	527
DERMASON	0.91	0.95	0.93	709

Figure 11: SVM- Performance Metrics

The above two tables compare the performance of Random Forest and SVM classification (with RBF kernel) algorithms. Both models perform well for all classes. However, SVM scores are marginally higher for a few classes, so the average score is higher than Random Forest. Here we can conclude that using SVM is a slightly better option because it is a powerful algorithm that can handle non-linear relationships between the features and the outcome variable. Moreover, the SVM works better with continuous variables whereas Random Forest is more robust with categorical variables.

The previous research by Koklu et.al [1] also produced the highest accuracy using the SVM algorithm however they did not drop any variables due to high correlation. This can lead to the dominance of one kind of information, which is not good for generalization. Moreover, the accuracy achieved by Koklu et.al [1] is only marginally higher than what we produced by using 9 features. Reducing the number of features means lower computational requirements, therefore, is more suitable for real-life applications.

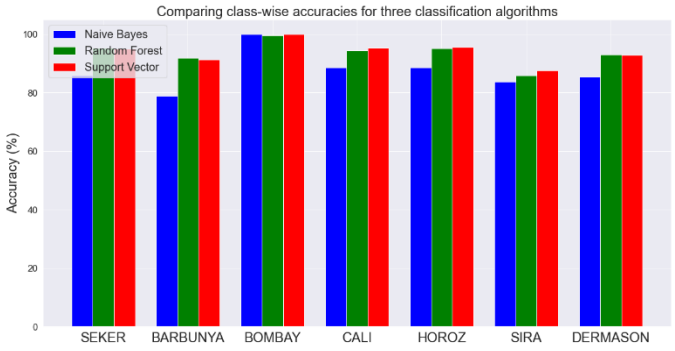
K-fold validation

The validation exercise is conducted to test the robustness of the models. We used the same validation methodology as in the original research paper. In this technique, the dataset is split into K smaller sets (called "folds"), then use K-1 of those folds as training data, and the remaining fold as validation data. This technique helps in reducing the variance of the evaluation metric by using multiple rounds of training and testing on different subsets of the data. It also ensures that every data point is used for both training and validation at some point during the modeling process.

The average accuracies for 10-fold validation are mentioned in the below table:

Classification Algorithm	Average Accuracy
Naïve Bayes	85.87%
Random Forest	92.53%
SVM	92.87%

SVM has the highest accuracy among all algorithms and as suggested earlier it is more suitable for this kind of dataset. This is near human accuracy. The below graph shows the class-wise accuracies of different classification algorithms:



It can be observed from the above bar chart that Bombay has the highest accuracy among the classes and SVM marginally outperforms Random Forest. Due to better suitability and accuracies, this paper recommends using SVM as the final automatic detection model.

Finally, we trained the SVM model on the full dataset and achieved an accuracy of 93.17%. We then used this as our final model to generate the automatic detection output file.

Limitations:

1. The performance of SVMs can be affected by non-uniform class distributions in the dataset, where the number of instances in one class is much larger or smaller than the others.
2. Also, SVMs can be sensitive to the choice of kernel function and its parameters. Which affects the performance of the model.

Future work:

1. Exploring different kinds of kernel functions and their parameters to find the optimal combination for the given dataset.
2. To develop strategies for addressing the issue of imbalanced class distributions, such as using oversampling or partial-sampling techniques or adjusting the class weights.

Model Risks:

1. Feature selection bias: The choice of features used to train the SVM model might be biased towards certain properties of the beans, ignoring other important features that could improve the model's performance.
2. Model overfitting: The model might be overfitting the training data i.e., it might learn the noise in the data instead of the underlying patterns, which may lead to poor performance on new, unseen data.

VI. Conclusion

The study explored the application of machine learning techniques to classify different types of dry beans using computer vision system (CVS) data. The aim of the study was to develop a model that can effectively segregate beans based on their physical characteristics, such as area, perimeter, shape, etc. These features characterise the quality and suitability for various uses. The study analysed the dataset, explored the features to identify relevant candidates for modeling, addressed data sanity issues, and subsequently transformed features. The study compares various classification algorithms and proposes the most suitable technique for the given problem while contrasting other available techniques and providing suggestions for further improvement. The study concludes that the proposed model can be used to segregate beans, automate the process, and save a lot of resources, meanwhile ensuring near-human accuracy, thus improving productivity and efficiency in the agriculture industry. However, the study suggests that more informative features, such as the color and weight of the beans and 3-dimensional features, can be included to improve the performance and robustness of the model.

VII. References

1) M. Koklu and I. A. Ozkan, “Multiclass classification of dry beans using computer vision and machine learning techniques,” Computers and Electronics in Agriculture, vol. 174, p. 105507, 2020.
2) Bhakthi Shetty, Disha Yeshwant, Kritika. R. Kotian, Anand. S. Uppar, “Coffee Bean Detection and Segregation”, Volume 9 Issue No.4, p. 21636, 2019.
3) https://www.analyticsvidhya.com/blog/2021/04/insight-into-svm-support-vector-machine-along-with-code/
4) Precision and Recall: Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.
5) F1-Score: Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427–437.
6) Support: Scikit-learn documentation. (n.d.). Glossary of common terms and API elements. Retrieved May 11, 2023, from https://scikit-learn.org/stable/glossary.html
7) https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python
8) https://www.sciencedirect.com/science/article/pii/S2666389921001847
9) https://www.youtube.com/watch?v=i_LwzRVP7bg&t=145
10) https://archive-beta.ics.uci.edu/dataset/602/dry+bean+dataset
11) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
12) https://scikit-learn.org/stable/modules/naive_bayes.html
13) M Deisenroth, A Faisal, CS Ong, Mathematics for Machine Learning. Available at: https://mml-book.github.io/book/mml-book.pdf
14) M. M. Hasan, M. U. Islam and M. J. Sadeq, "A Deep Neural Network for Multi-class Dry Beans Classification," 2021 24th International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 2021, pp. 1-5, doi: 10.1109/ICCIT54785.2021.9689905.
15) Grzegorz Słowiński, “Dry Beans Classification Using Machine Learning”, University of Technology and Economics, ul. Jagiellońska 82f, 03-301 Warsaw, Poland. Available at: https://ceur-ws.org/Vol-2951/paper3.pdf