Multi-class Classification of Dry Beans using Machine Learning

May 2023

1. Introduction:

The application of Machine Learning techniques to perform tasks such as classification, regression, clustering is extremely popular and in widespread use. ML techniques allow us to build extremely efficient and accurate models to model data. This report will discuss the application of different supervised Machine Learning techniques to classify 7 different varieties of Dry Beans into their respective classes. This report is written based on results obtained by building and evaluating the models discussed in further discussions on the Dry Beans dataset, which is made available to the public using a Creative Commons License.

2. The Dataset: Dry Beans

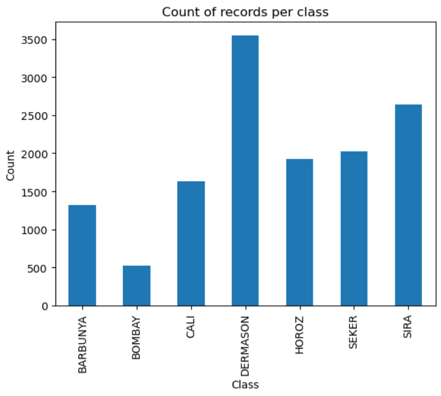

Figure 1. Class Distribution

The Dry Beans dataset contains attributes of seven different types of dry beans, which have been gathered from high resolution images of 13,611 grains of the 7 different registered dry beans. These images were subjected to segmentation and feature extraction stages, and a total of 16 features, including 14 dimensions and 4 shape forms were obtained from the grains (UC Irvine, 2020). However, the dataset is extremely unbalanced as the records for each label is unevenly distributed, as seen in Figure 1.

When training machine learning models, it is better to have a balanced dataset so that the model does not develop a bias to a certain label. However, the dataset requires minimal cleaning operations, as all the feature columns are in numeric format with no null values in any of the fields.

3. Exploratory Data Analysis Findings

EDA was performed on the dataset that identified strong positive correlation among certain dataset features. The fields - 'Area', 'Perimeter', MajorAxisLength', 'MinorAxisLength', 'ConvexArea', 'EquivDiameter' are strongly correlated to each other indicated by their spearman coefficients (see Figure 2). The strong correlation between these columns can also be indicative of redundancy or similiarity.

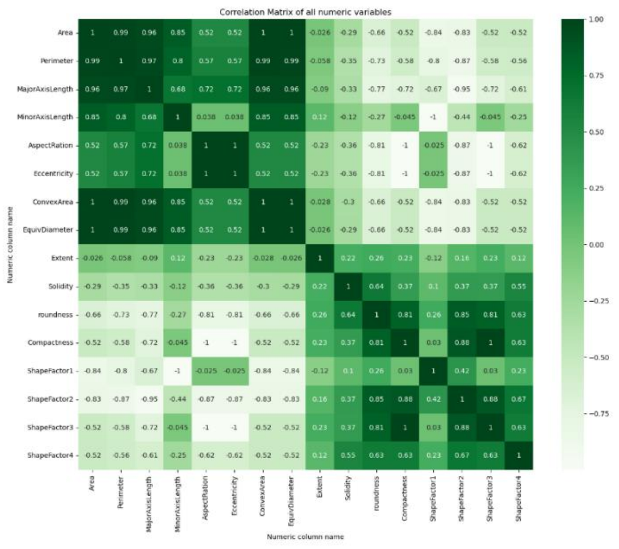

Figure 2. Correlation Matrix

The boxplots of the numeric columns also indicated that a large set of values lie outside the whiskers which indicate outliers in the data. This can potentially throw our models off. Moreover, the numeric values in the dataset are in different ranges, indicating the need for data normalisation.

Thus, a strong need to perform data scaling, oversampling the data to reduce the class imbalance and dimensionality reduction using Principal Component Analysis (PCA) was identified.

4. Data Transformations

Before the model training begins, it is necessary to perform transformations on the dataset that will make the training process faster, more efficient and yield accurate results.

4.1. Data normalisation

The first step was to perform data normalisation, i.e., to bring all the numeric features to a common scale. This effectively prevents features with large values dominating those with small values.

The data was scaled/normalised using the RobustScaler which scales the data to have a median of 0 and a range between the 1st and 3rd quartile (Scikit Learn, n.d.). The RobustScaler was chosen over the StandardScaler because of the presence of a lot of outliers in the dataset. The StandardScaler scales the data to have a mean of 0 and a standard deviation of 1 (Scikit Learn, n.d.). Outliers can sway the mean and standard deviation of a dataset and as such the StandardScaler becomes sensitive to outliers in the data.

4.2. Creation of Training and Test sets

The dry beans dataset has to be split into separate sets for training and testing the models. This is to ensure that the model is evaluated on data that it has not seen during the training phase and avoid overfitting the data. A good model should generalise well to new data and should not have high bias. The dry beans dataset is split into training and test sets in a 70:30 ratio respectively.

4.3. Overcoming the class imbalance

There are several methods to overcome class imbalance in a dataset, such as oversampling, undersampling etc. But oversampling the minority classes can lead to too many identical instances of the class that can lead to overfitting. While undersampling the majority class can lead the model to be trained on a training set with very few training examples. To avoid these ill effects, the SMOTE (Synthetic Minority Over-sampling Technique) was chosen. This will generate

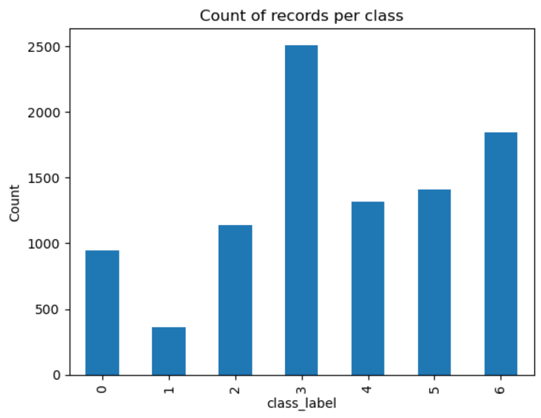

Figure 4. Class Distribution before SMOTE (Training Set)

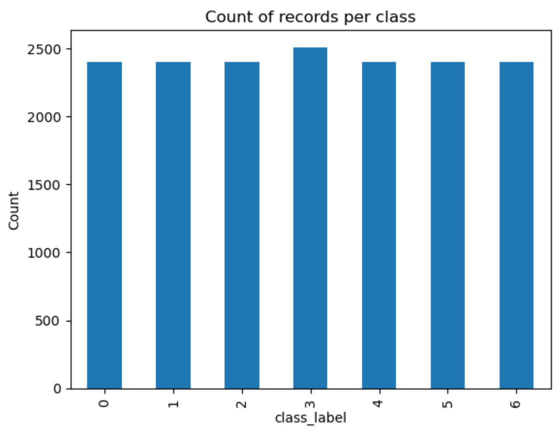

Figure 3 Class Distribution after SMOTE (Training Set)

synthetic/artificial data points for the minority class by interpolating new data points between existing data points (Korstanje, 2021). Since these are effectively new samples, this method helps to improve the diversity of the minority classes while preserving existing data. It also gets rid of the risk of overfitting the model.

It is to be noted that the SMOTE oversampling method is applied only on the training set and the test set is left untouched since applying SMOTE to the test set will change the data distribution of the real-world data.

4.4. Principal Component Analysis

PCA is a popular technique to reduce the dimensionality of a dataset and enables to analyse large datasets containing a high number of dimensions/features per instance, increasing the interpretability of data while preserving the maximum amount of information (Wikipedia, n.d.). There are 16 features in the dry beans dataset. During data exploration, it has been noted that some of these features may be redundant. Therefore, we have to investigate the possibility of reducing the dimensionality of the dataset using PCA, as it will help the models train better, can reduce overfitting to the data and reduce training time.

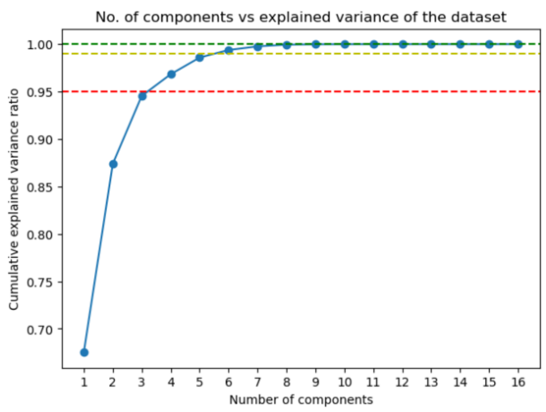

Figure 5 No. of components vs explained variance

During the experiments, it was found that 3 components can explain 95% variance of the dataset. Increasing the number of components to 6, we are able to capture 99% variance of the dataset and by increasing it 8 or more explains 100% of the variance. **In this experiment, the number of components will be set to 6.** This is because there is no need to capture 100% of the variance in the dataset as this can increase the probability of overfitting the model to the training data.

A scatter plot of the dataset was also plotted by reducing the number of dimensions to 2. This was done to find out if the dataset is linearly separable. However, the graph shows that it is not easy to linearly separate the classes into the 7 different classes.

5. Model Training

Since the dataset is not easily linearly separable, has multiple class labels for classification, and features with complex relationships between each other, it was decided to use a more complex supervised machine learning model. Logistic regression, Decision Tree, k-nearest neighbours was ruled out. The Machine Learning model used for performing multi-class classification of dry beans is:

1) Random Forest Algorithm
2) Support Vector Machines (SVM).

5.1. Random Forest

Random Forest is an ensemble learning method for performing classification, regressions tasks. Ensemble learning is a technique in which multiple models are trained and combined to improve the accuracy and robustness of the predictions. In the case of Random Forests. decision trees are used as base models. It works by constructing a large number of decision trees and aggregating their predictions to make the final prediction. Each decision tree is trained on a random subset of the training features. This randomness also helps to reduce overfitting and improve the accuracy of the model.

The benefit of random forest model is that they can handle large datasets with high dimensionality. They are also extremely robust to noisy data and less prone to overfitting than the base decision tree models (Andy Liaw, 2002). Hence, the random forest model was used to build the multi-class classifier.

Hyperparameters:

1) n_estimators: This defines the number of decision trees that will make up the forest. More number of estimators can lead to better accuracy at the cost of increased compute power.
2) max_depth: This is the maximum permissible depth of each decision tree in the forest. Increasing this limit can allow the model to capture complex relationships in the data, but increases the chance of overfitting.

Hyperparameter Optimisation/Tuning:

To get the best model, it is necessary to find the optimum values for the hyperparameters. For this, the model needs to be initialised with a specific setting and iterated of multiple settings and combinations and its performance needs to be evaluated on a test of validation set. This process is exhaustive.

In this experiment, hyperparameter tuning is carried out by implementing grid search cross-validation (GridSearchCV). It works by searching over a specified hyperparameter grid, which is basically a list of values for each hyperparameter to be tried. It evaluates the model for each combination of these hyperparameters by using cross-validation i.e. splitting the original dataset into multiple subsets and iteratively training the model on one subset and testing it on the remaining subsets. In this experiment, the performance metric used to identify the best model is the default mean-square error.

Results:

The hyperparameter grid contains a list of values for max_depth and the ‘n_estimators’ parameters, which contains a spectrum of values ranging from the low end to the high end.

Upon performing GridSearchCV on the model, the best hyperparameter values that were obtained are:

max_depth = 20 and **n_estimators = 200**

The model was retrained on the training dataset with these settings, and returned an **accuracy of 92.33%, precision of 0.9347, recall of 0.9334** and **f1-score of 0.9340.**

5.2. Support Vector Machines (SVM) Classifier

The SVM model performs classification/regression by creating a hyperplane or a boundary that separates different classes by maximising the distance between the closest points of each class. In a case where the data is binary, a hyperplane is a line that separates the two classes.

The hyperplane is defined by the equation:

w*x - b = 0

where **w -> weight vector that represents direction of hyperplane,**

x -> input vector

b -> bias term. This shifts the hyperplane (SVM Tutorial, n.d.)**.**

However, SVM can also model multi-class problems. It does so by using a kernel function to map the data to a higher dimension and find the hyperplanes in this higher dimensional space.

The SVM model is highly-effective in high dimensional spaces. It can even work well when the number of features is much larger than the number of samples. SVM is memory efficient by using only a subset of the training data points to define the decision boundary. It also has the added ability to handle non-linearly separable data and also generalises well to unseen data. These factors make SVM an ideal choice for performing multi-class classification on the dataset which we've already observed as linearly separable, multi-dimensional and complex.

Hyperparameters:

C: This is the regularisation parameter. The strength of regularisation is inversely proportional to C (Scikit Learn, n.d.). Basically, it controls the trade-off between achieving a low training error and a low testing error.

Kernel: This determines the transformation of the original data into a higher-dimensional feature space. It is set to 'rbf' (radial basis function) as the data is not linear.

Degree: controls the degree of the polynomial function. A higher degree helps to learn more complex features. Only valid for 'poly' kernel function.

Hyperparameter Optimisation/Tuning:

GridSearchCV is used here as well to find the best hyperparameter values for the model. Cross-validation is also applied with the cross-validation parameter set to 10, i.e. use 10 subsets for cross-validation.

The best hyperparameters obtained are:

C = 15.0 , **kernel = 'rbf'** and **degree = 3**

Results:

The model was retrained on the training dataset with these settings, and returned an **accuracy of 92.62%, precision of 0.9386, recall of 0.9378%** and **f1-score of 0.9381**

Confusion Matrix:

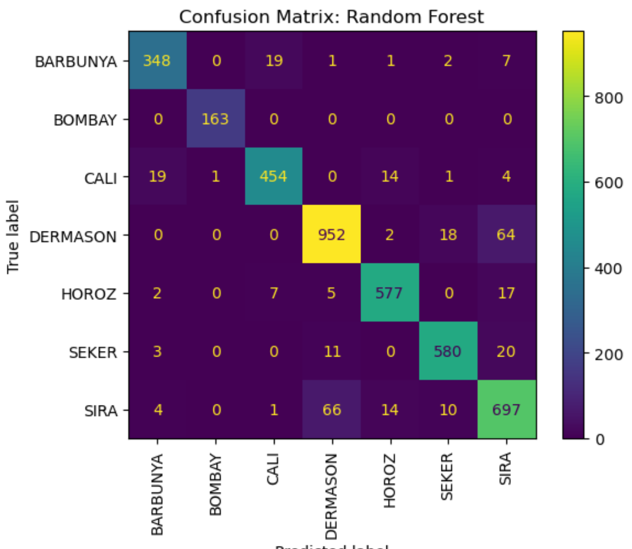

Figure 7 Confusion Matrix of RandomForest

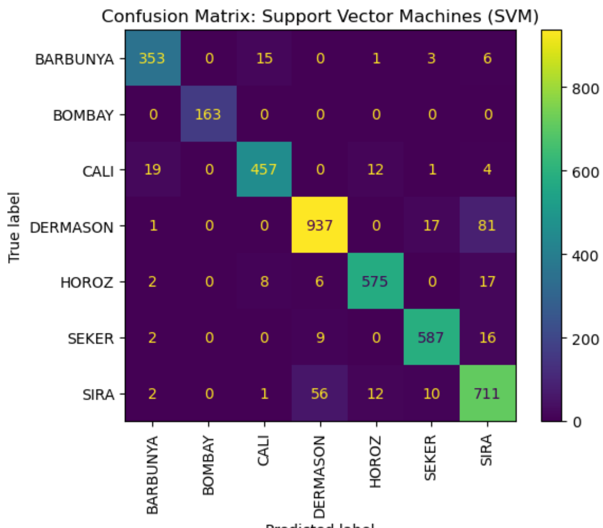

Figure 6 Confusion Matrix of SVM

Results Table

Model	Accuracy	F1-Score	Precision	Recall
RandomForest	92.33%	0.9340	0.9347	0.9334
SVM	92.62%	0.9381	0.9386	0.9378

Table 1 Performance Metrics of RF and SVM models

It is evident that the SVM has scored higher on all metrics. Looking at the confusion matrix, the SVM has higher number of correct classifications and lowest number of misclassifications among the two models. Hence, SVM is chosen as my primary model.

5.3. Secondary Model (Three Layer MLP / 3 Layer Neural Network):

Neural networks have gained popularity over the years due to their ability to learn complex relationships in the data. Given sufficient compute power and enough number of layers and neurons in them, they have the ability to learn almost any complex pattern or relationships in the data. And as such, it makes an effectively good choice to build a multi-class classifier by building a multi-layer perceptron network (also called neural network) for this purpose. As a secondary model, a 3-layer MLP was built to classify the dry beans dataset.

5.3.1. 3-Layer MLP Architecture:

The MLP consists of 3 different linear layers with a LeakyReLU activation function and a dropout layer. The linear layers are defined below:

1) **1st Layer**: This layer has an input size of 6 and output size of 100.
2) **2nd Layer**: The second layer is also a linear layer with 100 as input size and output size
3) **3rd Layer:** The third is also a linear layer with 100 as input size and 7 output size, since there are 7 different class labels to predict

Activation function:

The linear layers can only capture linear relationships between the data. Non-linearity is introduced within the neural network with the help of a LeakyReLU activation function. The original ReLU activation function outputs 0 until the output of layer is positive. This can lead to a a problem called "dying ReLUs" where neurons stop responding to any input because its activation is zero. The LeakyReLU can lead to better performance in these cases. The LeakyReLU activation is applied after the second linear network.

Dropout Layer:

A dropout layer is also added to the network which switches off a given percentage of neurons randomly. This leads the network generalise and capture a wide variety of features and also helps to generalise well to unseen data during the evaluation phase and reduces the probability of overfitting. At any time, 20% of the neurons may randomly shut off. Therefore, network does not rely on specific features and gains low bias. The dropout is applied to the output of the first linear layer.

Hyperparameters:

Epochs: Epochs are the number of iterations a model is trained for.

Learning Rate: This is the rate at which the weights of a model are updated during each epoch. A high learning rate can lead to unstable training, where the weights oscillate and the loss function fails to converge.

Batch Size: This determines the number of training samples used in each iteration of the training process.

Model Training and Hyperparameter Optimisation:

The model is trained on a training set and the performance is evaluated on the validation set to find the best hyperparameter settings. The best hyperparameters are identified by exhaustively evaluating the performance of a model on all combinations of values for the three different hyperparameters. This is similar to the GridSearchCV hyperparameter optimisation strategy, however, no cross-validation is performed on the dataset.

The best hyperparameter settings obtained by the above strategy are:

Epoch = 50, Learning Rate = 0.01 and **Batch Size = 10**

The 3 layer MLP was built, trained and evaluated using the train, validation and test datasets and obtained an accuracy of 92.72% and F1-Score of 93.81.

Results:

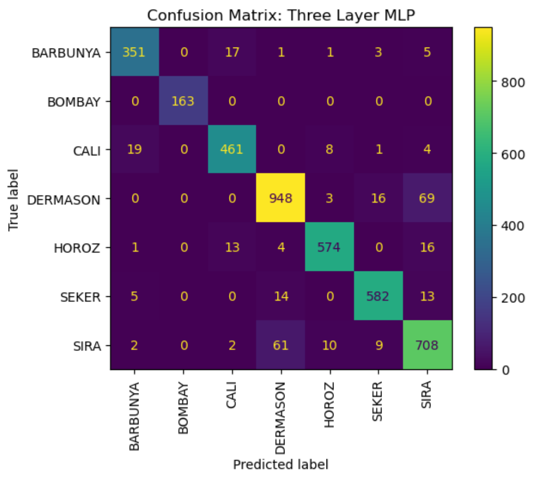

Figure 8. Confusion Matrix of 3-Layer MLP

Model	Accuracy	F1-Score
3-Layer MLP	92.72%	0.938157
SVM	92.62%	0.938109

Table 2 Performance Metrics of 3-Layer MLP and SVM Models

The obtained 3-layer MLP has better F1-Score (very slightly) and accuracy over the SVM model. However, I am still choosing the SVM model as my primary model. This is because the choice of a model depends on a lot of factors other than the performance metrics alone. The 3-layer MLP does have better performance, but requires more compute power and time for hyperparameter tuning and training. It is harder to scale the MLP into a greater number of layers as that increases the compute resource required exponentially. Though this can be brought down using a GPU, it means the MLP requires the use of specialised hardware. The MLP is also more prone to overfitting compared to the SVM model.

The SVM model has lower complexity compared to the 3-layer. It is able to produce results the same as the MLP, using fewer compute resources and in a shorter time duration. Though the CPU/GPU utilisation and time taken have not been recorded as part of this experiment, it is generally observed that SVM gives results in a quicky time than the MLP which require 50 epochs to get to this accuracy level.

6. Conclusion:

3 models were trained and evaluated on the Dry Beans dataset. All 3 of them performed well and scored almost similar scores across different performance metrics such as accuracy, f1-score, precision and recall. However, SVM was chosen as the primary model due to its scalability, requirement of comparatively less compute resource, time taken for hyperparameter tuning and model training. The 3-layer MLP was chosen as the secondary model on account of its performance, beating both the other models in accuracy and f1-score.

It should also be noted that good performance for multi-class classification could be obtained due to performing data pre-processing steps such as SMOTE, normalising the features and dimensionality reduction using PCA. This underlines the importance of performing sufficient pre-training steps.

7. References

Andy Liaw, M. W., 2002. Classification and regression by RandomForest.. pp. 18-22.

Korstanje, J., 2021. *SMOTE.* [Online]
Available at: https://towardsdatascience.com/smote-fdce2f605729

Scikit Learn, n.d. *RobustScaler.* [Online]
Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

Scikit Learn, n.d. *StandardScaler.* [Online]
Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

Scikit Learn, n.d. *SVM.* [Online]
Available at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

SVM Tutorial, n.d. *SVM - Understanding the Math.* [Online]
Available at: https://www.svm-tutorial.com/2015/06/svm-understanding-math-part-3/

UC Irvine, 2020. *Dry Bean Dataset.* [Online]
Available at: https://archive-beta.ics.uci.edu/dataset/602/dry+bean+dataset
[Accessed 2023].

Wikipedia, n.d. *Principal Component Analysis.* [Online]
Available at: https://en.wikipedia.org/wiki/Principal_component_analysis

8. Code Appendix

See ‘MachineLearningCourseworkFinal.ipynb’ for all the code used in the experiment

9. Automatic Detection Output

See ‘automatic_detection_output.csv’ for the output.