SVM Assignment

Yeshi, Turab and Osman

April 14, 2024

		CONTRIBUTION	
		Part 1 (Practice)	Part 2 (Go wild!)
student1	Yeshi	33%	30%
student2	Turab	33%	30%
student3	Osman	33%	30%
generativeAI	(ChatGPT)	0%	10%
total		100%	100%

GenerativeAI was used for commenting, latex code, formatting, and documentation.

1 Dataset

For our project we have chosen the Dry Beans Classification dataset. This dataset was made from images of 13611 grains of 7 different registered dry beans. This results in a total of 16 features, 12 dimensions and 4 shape forms:

- 13611 datapoints
- 16 features
  - Area, Perimeter, MajorAxisLength, MinorAxisLength, AspectRatio, Eccentricity, ConvexArea, EquivDiameter, Extent, Solidity, Roundness, Compactness, ShapeFactor1, ShapeFactor2, ShapeFactor3, ShapeFactor4
- 7 classes
  - SEKER, BARBUNYA, BOMBAY, CALI, HOROZ, SIRA, DERMASON

$$\text{ShapeFactor1 (SF1)} = \frac{L}{A} \tag{1}$$

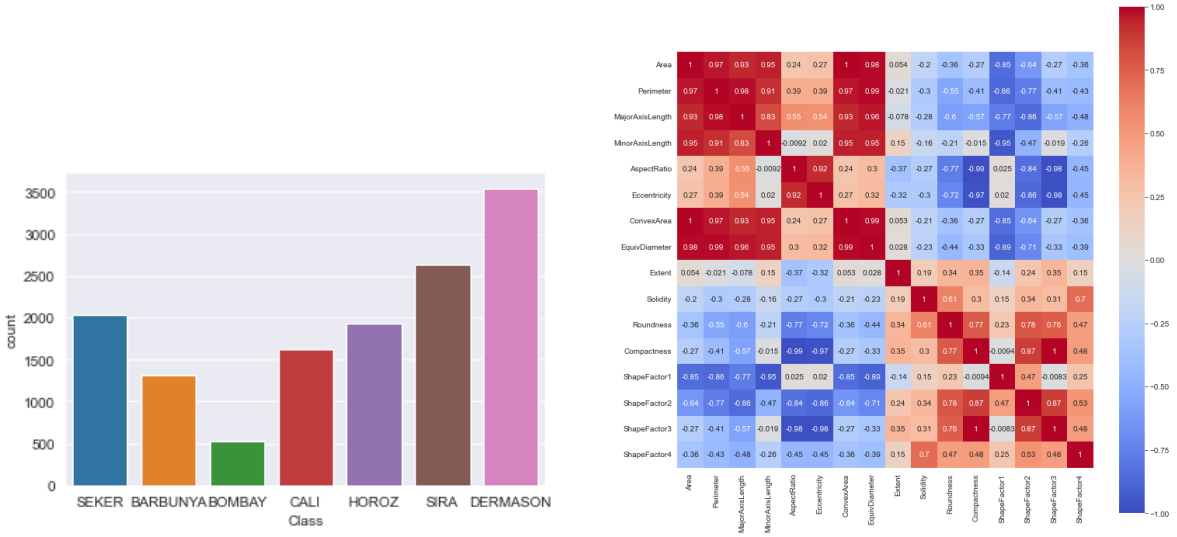
$$\text{ShapeFactor2 (SF2)} = \frac{l}{A} \tag{2}$$

$$\text{ShapeFactor3 (SF3)} = \frac{A}{\frac{L}{2} \times \frac{l}{2} \times \pi} \tag{3}$$

$$\text{ShapeFactor4 (SF4)} = \frac{A}{\frac{L}{2} \times \frac{l}{2} \times \pi} \tag{4}$$

where A is area, L is Major axis length and l is Minor axis length.

We have no missing values in the dataset and we have 68 duplicate datapoints.



(a) Count and distribution of all beans categories

(b) Pearson linear correlation

Figure 1: Visualizations of dry beans data analysis

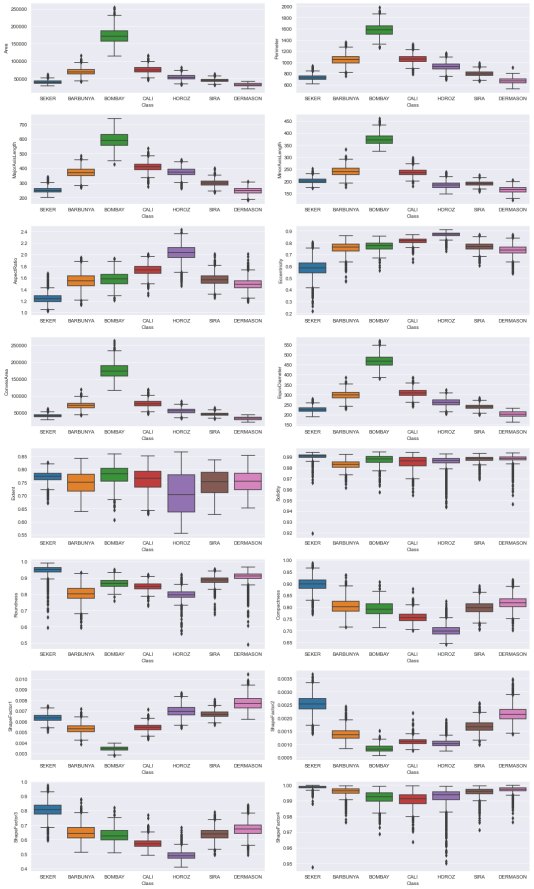


Figure 2: Boxplot of numerical features for each type of bean

As we can see in Figure 1a we have a class imbalance. From our original dataset, we have far more of the class 'DERMASON' than the class 'BOMBAY'. Class imbalance can lead to a model bias towards the majority class, affecting accuracy and reliability for minority classes. To mitigate this issue, we have implemented two oversampling methods in our data pre-processing: the Synthetic Minority Over-sampling Technique (SMOTE) and Adaptive Synthetic Sampling (ADASYN). Both methods generate 'synthetic' examples of the minority class instead of simple oversampling with replacement. This approach helps to prevent model bias towards the majority class and enhances overall model generalization."

Variables that have a strong positive correlation change in the same direction. As seen in Figure 1b, the features 'Area' and 'Perimeter' have a correlation coefficient of 0.97, indicating that when the area increases, the perimeter tends to increase as well. 'ShapeFactor1' exhibits a negative correlation with 'Area', 'Perimeter', 'MajorAxisLength', 'MinorAxisLength', and 'ConvexArea', suggesting an inverse relationship where increases in this shape factor correspond to decreases in the other attributes. 'Extend' shows little to no linear correlation with variables like 'Area' or 'Perimeter'. High correlations among 'Area', 'Perimeter', 'MajorAxisLength', 'MinorAxisLength', 'ConvexArea', and 'EquivDiameter' may indicate redundancy, as these geometrical measurements naturally scale with the size of the objects.

According to the Boxplot, Figure 2, the 'Bombay' class is distinct from the other classes, particularly in 'Area', 'perimiter', 'MajorAxisLength', 'MinorAxisLength', 'ConvexArea' and 'EquivDiameter'.

We use 'MinMaxScaler' to normalize the features values of the training and validation datasets. We then use 'StandardScaler' to ensure that the mean is 0 and scaling the features to unit variance to mitigate bias.

2 Our SVM algorithm

The Support Vector Machine (SVM) algorithm is a supervised machine learning method. We used it for classification for this dataset, however it can be used for regression as well. The goal of the SVM algorithm is to seek the best boundary that separates classes in the feature space. The key components of the SVM algorithm:

- **Hyperplane**: A decision boundary that separates different classes with maximum margin.
- **Support Vectors**: Data points that are closest to the hyperplane and influence its position and orientation.
- **Margin**: The distance between the hyperplane and the nearest data points from each class.
- **Kernel Trick**: A technique that transforms data into a higher dimension to make a non-linear separation possible.
- **Regularization (C parameter)**: A parameter that controls the trade-off between achieving a low error on the training data and minimizing model complexity for better generalization.

We train an SVM model with the RBF kernel, and evaluate its performance on training and validation datasets. We evaluate the model using the F1 score and accuracy. We also plot a confusion matrix which shows the number of correct and incorrect predictions made by the classifier. This helps us understand the classification performance on each class.

3 Hyper-parameter tuning

With the aim of optimizing our SVM classifier, we utilize Grid Search for hyper-parameter tuning to pinpoint the ideal values for 'C' and 'gamma'. We pick these values that achieve the highest F1 Score (Micro) during a 5-fold cross-validation process.

4 Results

Technique	Train Accuracy	Validation Accuracy	Train F1-Score	Validation F1-Score
With StandardScalar	93.46%	90.74%	93.46%	90.74%
With SMOTE Only	64.44%	63.92%	64.44%	63.92%
With ADASYN Only	59.47%	59.13%	59.47%	59.13%
SMOTE & StandardScalar	94.28%	92.58%	94.28%	92.58%
SMOTE, SScalar & MMScalar	94.28%	92.58%	94.28%	92.58%
Hyperparameter Tuning	95.48%	92.51%	95.48%	92.51%

Table 1: Performance Metrics for Different Techniques.
Note: Here SScalar: StandardScalar, MMScalar: MinMaxScalar

5 Challenges

During the course of our project, we faced several challenges, the most critical of which was managing the class imbalance inherent in our dataset. As observed from our initial analysis (see Figure 1a), the 'DERMASON' class significantly outnumbered others like 'BOMBAY', which can lead to a bias in

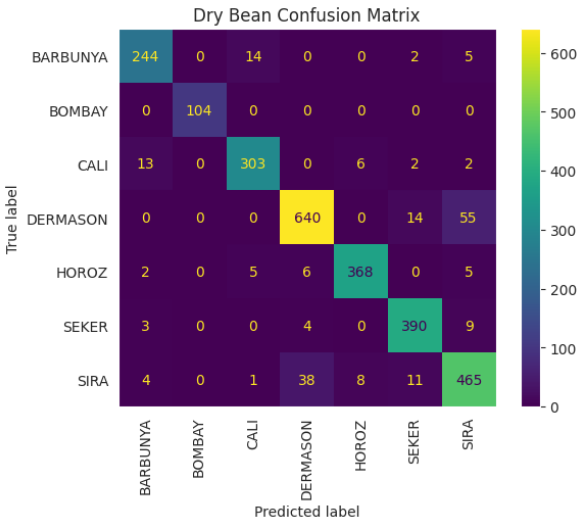


Figure 3: Confusion matrix

predictive modeling. To address this, we employed the oversampling technique SMOTE, which helped in synthesizing new examples in the minority class to achieve a balance.

The different ranges of numerical features presented a challenge for normalization. Without proper scaling, features with larger ranges could dominate the learning algorithm, leading to biased results. We used 'MinMaxScaler' to normalize features ensuring that each feature contributed equally to the model's learning process. Additionally, the use of 'StandardScaler' helped to standardize features by removing the mean and scaling to unit variance, which is crucial for models like SVM that are sensitive to the scale of input data [Sot17].

6 Model evaluation and fairness considerations

The performance of our dry bean classification model has been assessed using the F1 Score and Accuracy. We achieved a F1 Score of 0.9258 on our training set. We got a similar F1 Score on our validations data, which indicates that our model performs consistently with unseen data. The training accuracy is also high, which confirms the model's effective learning from the training data.

Our implementation using Synthetic Minority Over-sampling Technique (SMOTE) addresses the fairness aspect by ensuring the model is trained on a balanced dataset. This helps against the model favouring one class over others.

By observing the confusion matrix in Figure 3 we can see that the highest numbers are on the diagonal which indicates that our model performs well. However we can also see that the model confuses the classes "SIRA" and "BARBUNYA". This makes sense as these beans look similar, which can be seen in Figure 4. These findings prompt further investigation into feature distinctions and potential model adjustments to enhance discriminatory power between these classes.

References

[KO20] Murat Koklu and Ilker Ali Ozkan. Multiclass classification of dry beans using computer vision and machine learning techniques. *Computers and Electronics in Agriculture*, 2020.

[Sot17] Dave Sotelo. Effect of feature standardization on linear support vector machines, 2017. Accessed: April 14, 2024.

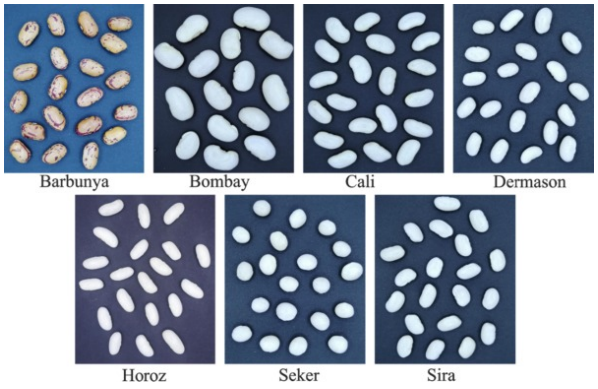


Figure 4: Images of beans [KO20]