Introduction

Using image processing techniques to extract features from video recordings and applying machine learning algorithms to classify objects is a subject of great interest in many business areas. It automates tasks and reduces costs.

Publicly available Dry Bean dataset (Koklu, 2020), used in this experiment, was created from images of 13,611 grains of 7 basic dry bean varieties captured by a specific computer vision system (Koklu and Ozkan, 2020). These images were further processed to extract 16 features - 12 dimensions and 4 shape forms. Five classical supervised machine learning algorithms (Support Vector Machine (SVM), Decision Tree (DT), k-Nearest Neighbors (KNN), Random Forest (RF) and Multilayer Perceptron (MLP)) were employed to classify beans. Subsequently, the three models with the highest average weighted F1-scores were chosen. We then tuned the hyperparameters of these models to identify the one that achieves the best averaged weighted F1 score, optimizing it for this specific multi-class classification task.

Exploratory Data Analysis

The dataset contains 13,611 instances, 16 numerical features, and a label "Class" (Fig. 1). The data is of good quality with no missing or zero values. The distribution of data for the seven bean varieties is uneven across the dataset (Tab. 1). Therefore, stratified sampling will

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13611 entries, 0 to 13610
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Area             13611 non-null  float64
 1   Perimeter        13611 non-null  float64
 2   MajorAxisLength  13611 non-null  float64
 3   MinorAxisLength  13611 non-null  float64
 4   AspectRation     13611 non-null  float64
 5   Eccentricity     13611 non-null  float64
 6   ConvexArea       13611 non-null  float64
 7   EquivDiameter    13611 non-null  float64
 8   Extent           13611 non-null  float64
 9   Solidity         13611 non-null  float64
 10  roundness        13611 non-null  float64
 11  Compactness      13611 non-null  float64
 12  ShapeFactor1     13611 non-null  float64
 13  ShapeFactor2     13611 non-null  float64
 14  ShapeFactor3     13611 non-null  float64
 15  ShapeFactor4     13611 non-null  float64
 16  Class            13611 non-null  object
dtypes: float64(16), object(1)
memory usage: 1.8+ MB
```

Fig. < 1 > .info() output.

Table < 1 > Distribution of labels.

Label	Count	%
DERMASON	3546	26.1
SIRA	2636	19.4
SEKER	2027	14.9
HOROZ	1928	14.2
CALI	1630	12.0
BARBUNYA	1322	9.7
BOMBAY	522	3.8

be used to divide the dataset into test and train sets. Analysis of the descriptive statistics revealed that the data have a large spread in values and need to be standardized. Histograms (Fig. 2) showed that several features are multimodal. The dataset also includes right-skewed, left-skewed, and nearly normally distributed features.

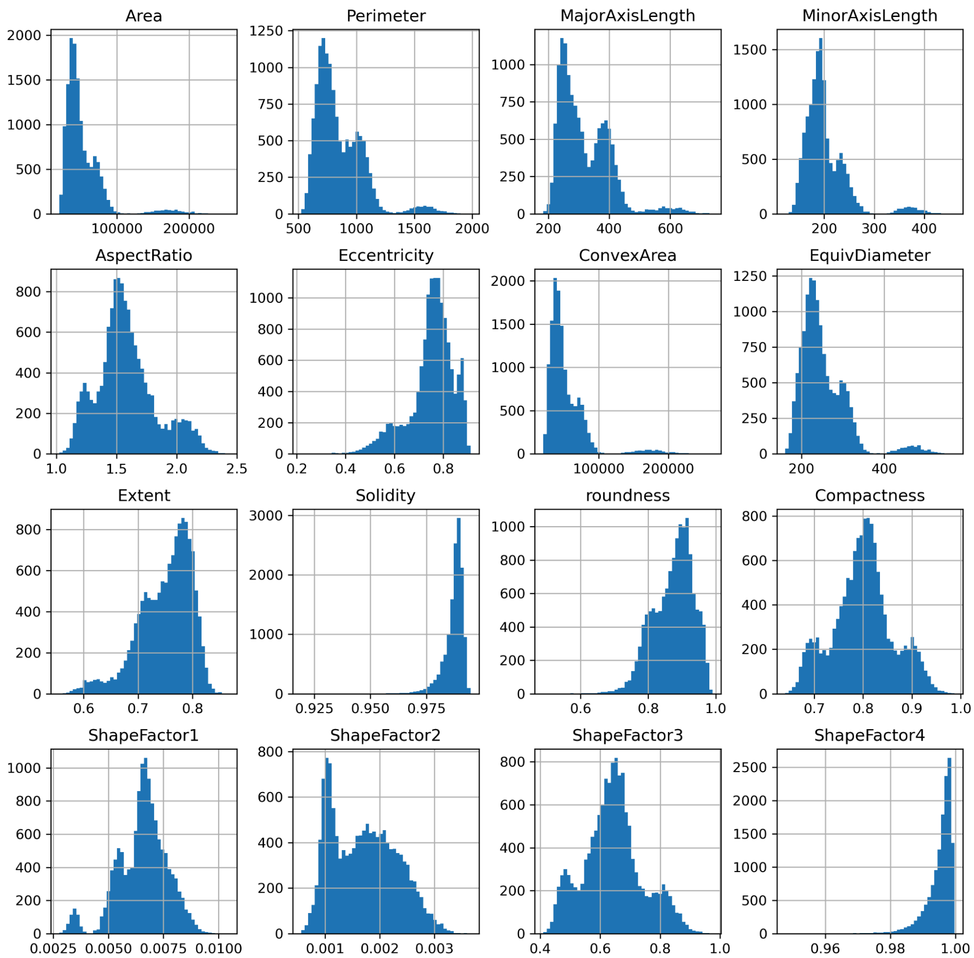

Fig. < 2 > Histograms of all numerical features in the dataset.

Density plots (Fig. 3) were created for all variables to better understand the slight rise on the right side of the distributions observed in the histograms of the variables "Area", "Perimeter", "MajorAxisLength", "MinorAxisLength", "ConvexArea", and "Compactness".

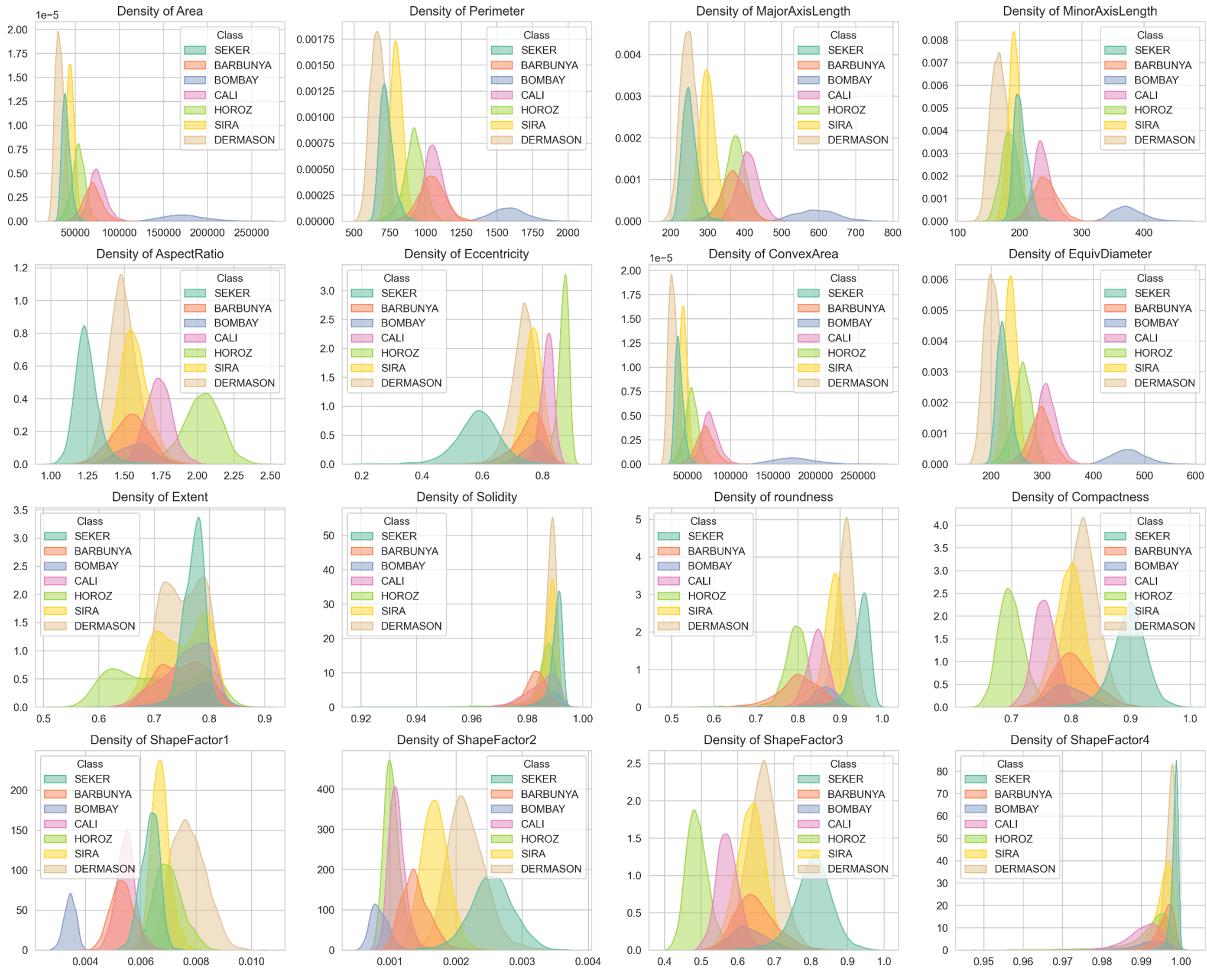

Fig. < 3 > Density plots of all numerical features in the dataset.

The “BOMBAY” class shows strong right-skewness in size-related features, indicating that its grains are significantly larger. This distinct size difference could make it easier to classify this variety in future experiments. As the given data is very specific and was collected in the special conditions we assume that there are now outliers in the data, although the box plots showed their presence.

Since DT and RF handle skewness well, but KNN, SVM, and MLP are sensitive to it (Géron, 2023), we will use logarithmic transformations for right-skewed features and Box-Cox transformations for left-skewed features.

Methodology

Experiments were conducted using Jupyter Notebook on VSCode, executed on an Apple MacBook Air equipped with an M2 chip, 8GB RAM, 256GB SSD, and running the Sequoia 15.2 operating system.

To ensure equal representation of each class in the training, validation, and testing sets, stratified sampling was employed. The initial split was 20% for testing and 80% for training. The training set was then further divided into 10% for validation and 90% for training. A data transformation pipeline was created and utilized to first apply logarithmic transformation to right-skewed features and Box-Cox transformation to left-skewed features (Fig. 4). In the second step, all features were standardized using StandardScaler().

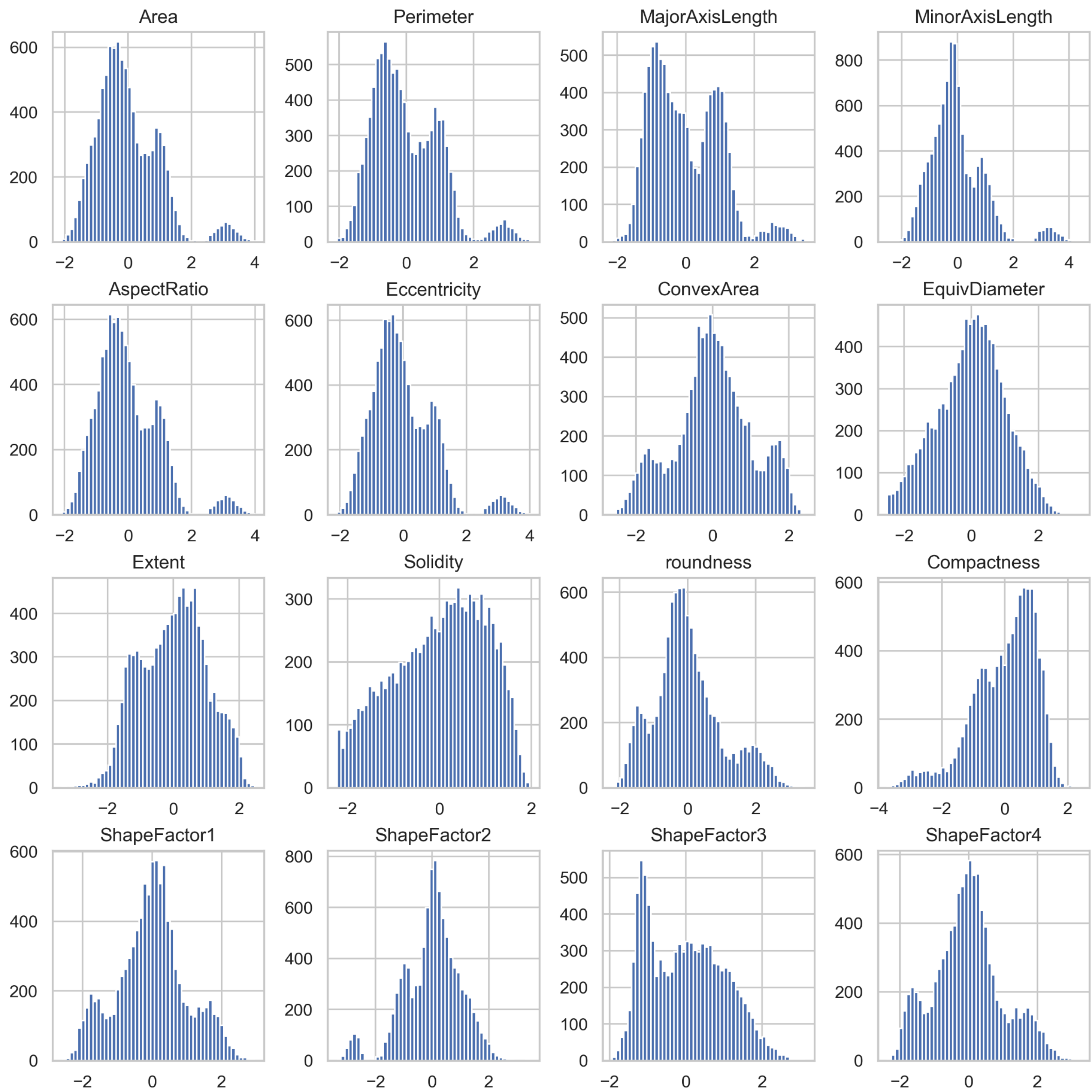

Fig. < 4 > Histograms of all transformed and standartized numerical features in the dataset.

SVM and KNN classifiers with default settings, DT and RF with parameter *random_state* changed to 42 and MLP with *max_iter* parameter changed to 500 and *random_state* changed to 42 were trained on the training set and validated on the validation set. Setting the *random_state* parameter to 42 for DT, RF, and MLP classifiers ensures that the randomness built into their processes (like data splitting and initial weight settings) is consistent across runs, making the results reproducible and comparable.

The weighted average F1-score from the sklearn *classification_report()* was used to evaluate trained models because it provides a single metric that balances both precision and recall, and it adjusts for class imbalance by weighting the F1-score of each class by its presence in the dataset, ensuring a fair representation of model performance across all classes.

Principal Component Analysis (PCA) was also conducted on the training (80%) and test (20%) sets to explore the impact of dimensionality reduction. Before applying PCA, the data was standardized using *StandardScaler()* to ensure that each feature contributed equally, avoiding bias toward variables with larger scales. This analysis aimed to retain Principal Components that described 95% of the data variance, to determine if this approach could enhance the weighted average F1-score of the trained models. The same models with the same parameters as mentioned above were used after applying PCA.

Code is available on GitHub: https://github.com/MaxNoLV/ml_pa_project

Preliminary modeling outcomes

Table 2 displays the performance of each model using the initial parameters described above, focusing on the F1-score metric for each class. At this stage, the three models—RF, MLP, and SVM—with the highest average weighted F1-scores were selected for further performance enhancement through hyperparameter tuning. Detailed tables showing precision, recall, and accuracy metrics are included in Appendix 1.

Proportions of variance of the four Principal Components, obtained using sklearn's *PCA(n_components=0.95)* that describe 95% of the data variance, are shown in Table 3.

A comparison of the average weighted F1-scores of models trained on the initial transformed and standartized data, evaluated on the validation set, and on these four Principal Components (Table 4), indicates that dimensionality reduction negatively impacted the models. Although we reduced the number of features from 16 to 4, it did not result in better-performing models.

Table < 2 > F1-scores and Average weighted F1-scores for each class and trained model on the validation set.

Class ↓ / Model →	SVM	DT	KNN	RF	MLP
BARBUNYA	0.9192	0.8844	0.9200	0.9254	0.9353
BOMBAY	1.0000	1.0000	1.0000	1.0000	1.0000
CALI	0.9283	0.8947	0.9283	0.9430	0.9506
DERMASON	0.9190	0.8826	0.9120	0.9364	0.9101
HOROZ	0.9618	0.9460	0.9587	0.9587	0.9587
SEKER	0.9649	0.9060	0.9615	0.9811	0.9515
SIRA	0.8716	0.8194	0.8756	0.8910	0.8793
Avg. weighted F1-score	**0.9269**	0.8890	0.9250	**0.9396**	**0.9279**

Table < 3 > Proportions of variance explained by Principal Components.

Principal Component 1	55.38%
Principal Component 2	26.47%
Principal Component 3	8.09%
Principal Component 4	5.10%

Table < 4 > Comparison of Average weighted F1-scores for models trained on transformed and standardized data versus Principal Components.

Data ↓ / Model →	SVM	DT	KNN	RF	MLP
Principal Components	0.8949	0.8536	0.8917	0.8955	0.9001
Transformed and standartized initial	0.9269	0.8890	0.9250	0.9396	0.9279

Hyperparameter tuning

The 5-fold cross-validation GridSearch method with parameter *scoring* set to *f1_weighted*, used for tuning the hyperparameters of the SVM, RF, and MLP models, was applied to the 80% training data (obtained in initial split) to ensure that the test data remained unseen by the models during the tuning process. Figure 5 outlines the specific parameters and their respective ranges that were explored during the hyperparameter tuning process for the RF, MLP, and SVM models.

RF		MLP		SVM	
Factors	***Levels***	***Factors***	***Levels***	***Factors***	***Levels***
n_estimators	100, 200, 300	hidden_layer_sizes	100, (100, 100), (100, 100, 100)	C	0.1, 1, 10, 100
max_features	sqrt, log2	activation	relu, tanh	gamma	0.001, 0.01, 0.1, 1, scale
max_depth	None, 10, 20, 30	solver	sgd, adam	kernel	linear, rbf, poly
min_samples_split	2, 5, 10	alpha	0.0001, 0.001, 0.01		
min_samples_leaf	1, 2, 4	learning_rate_init	0.001, 0.01		

Fig. < 5 > Hyperparameter ranges for RF, MLP, and SVM Models.

Figure 6 presents the combinations of parameters for the models with the highest average weighted F1-scores, along with the time required for tuning each model.

RF	
Factors	***Levels***
n_estimators	200
max_features	sqrt
max_depth	None
min_samples_split	2
min_samples_leaf	1

Time: 1 hour 6 minutes

MLP	
Factors	***Levels***
hidden_layer_sizes	100
activation	relu
solver	sgd
alpha	0.01
learning_rate_init	0.01

Time: 2 hours 31 minutes

SVM	
Factors	***Levels***
C	10
gamma	scale
kernel	rbf

Time: 11 minutes

Fig. < 6 > Parameter combinations and tuning times.

The final averaged weighted F1-scores on the test data reveal that MLP achieved the highest score of 0.9143, closely followed by SVM at 0.9130 and RF at 0.9121.

Finally, Table 5 provides a comparison of the average weighted F1-scores on the test data for the tuned RF, SVM, and MLP models against the results of the same models with default (minor adjustments) parameters. Detailed tables showing precision, recall, and accuracy metrics are included in Appendix 1.

Table < 5 > Proportions of variance explained by Principal Components.

Parameters ↓ / Model →	**SVM**	**RF**	**MLP**
Default (test set)	0.9156	0.9121	0.9077
Tuned (test set)	0.9130	0.9121	0.9143

The confusion matrix (Figure 7) displays that the SVM model (has best average weighted F1-score) with default parameters performs well in classifying bean varieties. The "BOMBAY" class has perfect classification with no misclassifications observed. However, notable misclassifications were observed between "DERMASON" and "SIRA": 38 instances of "DERMASON" were incorrectly classified as "SIRA", and 65 instances of "SIRA" were misclassified as "DERMASON". This suggests an overlap in the feature characteristics of these two bean varieties, leading to confusion for the SVM model. It means that while the SVM model is effective for most classes, there is room for improvement in distinguishing between a few specific classes where similar features may lead to confusion.

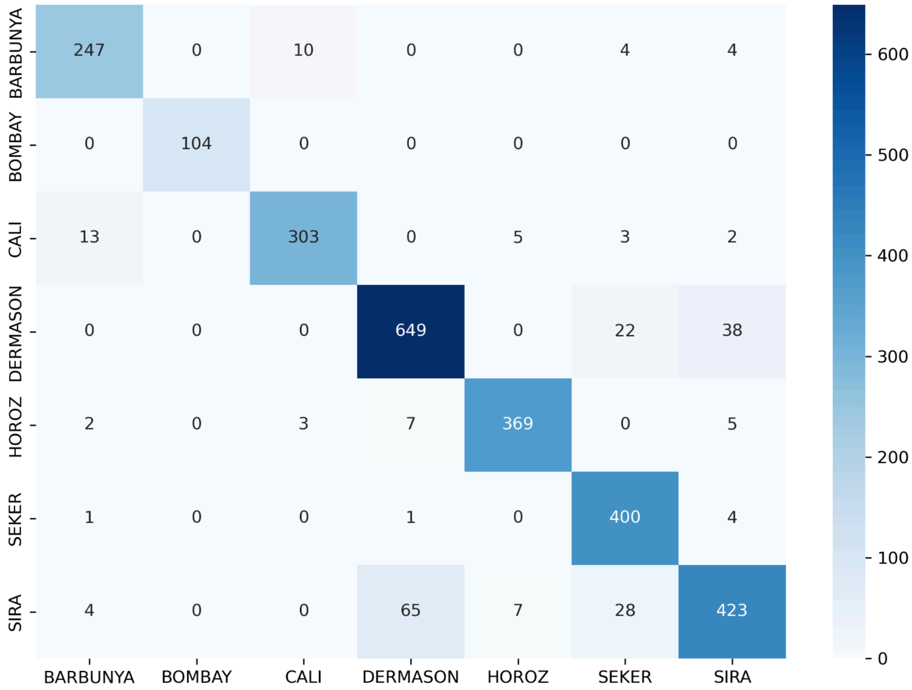

Fig. < 7 > Confusion matrix for the SVM model performance on test data.

Discussion

The comparison of average weighted F1-scores for SVM, RF, and MLP models with both default and tuned parameters reveals some interesting results. With default settings, the SVM model outperformed both the RF and MLP models with tuned and default parameters, achieving an F1-score of 0.9156, suggesting that SVM's capabilities and hyperparameters are well-suited to the characteristics of this dataset. Interestingly, the RF model showed no improvement in performance after tuning, maintaining an F1-score of 0.9121, which might indicate that the default settings are already near-optimal for this specific task. In contrast, the MLP model showed an improvement from 0.9077 to 0.9143 when tuned, indicating that MLP's performance is dependent on the right parameter settings.

An unexpected finding was that despite tuning, the SVM's performance slightly decreased from 0.9156 to 0.9130. This result challenges the opinion that tuning always leads to better performance and underscores the importance of understanding the data and algorithm interaction.

It was also interesting to observe that the RF model's performance did not change after tuning, suggesting a robustness to the choice of hyperparameters for this type of data.

Moreover, the selection of parameters for GridSearch may not have been optimal, reflecting my initial experience with hyperparameter tuning, which could have further influenced the effectiveness of the model improvements.

The decline in model performance following the application of PCA was also unexpected, indicating that some critical information useful for classification might have been lost during dimensionality reduction. This outcome highlights the complexity of feature selection and the need for careful consideration when applying methods like PCA. Method might simplify the model at the cost of losing essential data.

Conclusion

The SVM algorithm proved to be the best choice for this task. It worked well right away with default settings.

In this project, I began by evaluating five supervised machine learning algorithms: SVM, DT, KNN, RF, and MLP, using default settings with minor adjustments. After the initial assessment, SVM, RF, and MLP, which had the highest average weighted F1-scores, were selected for further tuning. My findings showed that SVM had the best performance with default settings and MLP benefited from hyperparameter tuning. In contrast, RF's performance remained stable, unaffected by tuning. Additionally, applying PCA to reduce dimensions from 16 to 4 led to a decrease in model performance, indicating a loss of important information.

This analysis improved my understanding of the model's behavior with different configurations and showed the importance of a detailed approach to machine learning. These insights will help with future projects, especially in selecting and setting up models more effectively.

As next steps, I could explore using different feature sets to see if they affect model performance. Trying ensemble methods or more advanced techniques like deep learning could also bring improvements. Additionally, gaining more experience in selecting hyperparameters for tuning could enhance model effectiveness.

References

Géron, A. (2023)‘Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems’ Data science / machine learning. Third edition. Beijing Boston Farnham Sebastopol Tokyo: O’Reilly.

Koklu, M. (2020) *Dry Bean* [online]. Available from: https://archive.ics.uci.edu/dataset/602 [Accessed 21 December 2024].

Koklu, M. and Ozkan, I.A. (2020) Multiclass classification of dry beans using computer vision and machine learning techniques. *Computers and Electronics in Agriculture* [online]. 174, p. 105507.

Appendix 1

Classification Report for SVM:

	precision	recall	f1-score
BARBUNYA	0.9891	0.8585	0.9192
BOMBAY	1.0000	1.0000	1.0000
CALI	0.9111	0.9462	0.9283
DERMASON	0.9190	0.9190	0.9190
HOROZ	0.9437	0.9805	0.9618
SEKER	1.0000	0.9321	0.9649
SIRA	0.8444	0.9005	0.8716
accuracy			0.9265
macro avg	0.9439	0.9338	0.9378
weighted avg	0.9291	0.9265	0.9269

Fig < 1 > Classification report for SVM using default parameters (validation set)

Classification Report for Decision Tree:

	precision	recall	f1-score
BARBUNYA	0.9462	0.8302	0.8844
BOMBAY	1.0000	1.0000	1.0000
CALI	0.8750	0.9154	0.8947
DERMASON	0.8921	0.8732	0.8826
HOROZ	0.9255	0.9675	0.9460
SEKER	0.9926	0.8333	0.9060
SIRA	0.7654	0.8815	0.8194
accuracy			0.8880
macro avg	0.9138	0.9002	0.9047
weighted avg	0.8946	0.8880	0.8890

Fig < 2 > Classification report for DT using default parameters (validation set)

Classification Report for KNN:

	precision	recall	f1-score
BARBUNYA	0.9787	0.8679	0.9200
BOMBAY	1.0000	1.0000	1.0000
CALI	0.9111	0.9462	0.9283
DERMASON	0.9120	0.9120	0.9120
HOROZ	0.9379	0.9805	0.9587
SEKER	1.0000	0.9259	0.9615
SIRA	0.8520	0.9005	0.8756
accuracy			0.9247
macro avg	0.9417	0.9333	0.9366
weighted avg	0.9269	0.9247	0.9250

Fig < 3 > Classification report for KNN using default parameters (validation set)

Classification Report for Random Forest:

	precision	recall	f1-score
BARBUNYA	0.9789	0.8774	0.9254
BOMBAY	1.0000	1.0000	1.0000
CALI	0.9323	0.9538	0.9430
DERMASON	0.9397	0.9331	0.9364
HOROZ	0.9379	0.9805	0.9587
SEKER	1.0000	0.9630	0.9811
SIRA	0.8727	0.9100	0.8910
accuracy			0.9394
macro avg	0.9517	0.9454	0.9479
weighted avg	0.9407	0.9394	0.9396

Fig < 4 > Classification report for RF using default parameters (validation set)

Classification Report for MLP Neural Network:

	precision	recall	f1-score
BARBUNYA	0.9895	0.8868	0.9353
BOMBAY	1.0000	1.0000	1.0000
CALI	0.9398	0.9615	0.9506
DERMASON	0.9117	0.9085	0.9101
HOROZ	0.9379	0.9805	0.9587
SEKER	1.0000	0.9074	0.9515
SIRA	0.8465	0.9147	0.8793
accuracy			0.9275
macro avg	0.9465	0.9371	0.9408
weighted avg	0.9302	0.9275	0.9279

Fig < 5 > Classification report for MLP using default parameters (validation set)

Classification Report for RF:

	precision	recall	f1-score
BARBUNYA	0.9228	0.9019	0.9122
BOMBAY	1.0000	1.0000	1.0000
CALI	0.9437	0.9264	0.9350
DERMASON	0.8884	0.9210	0.9044
HOROZ	0.9608	0.9534	0.9571
SEKER	0.9097	0.9680	0.9379
SIRA	0.8694	0.8083	0.8378
accuracy			0.9126
macro avg	0.9278	0.9256	0.9263
weighted avg	0.9124	0.9126	0.9121

Fig < 6 > Classification report for RF after parameters tuning (test set)

Classification Report for MLP:

	precision	recall	f1-score
BARBUNYA	0.9182	0.9321	0.9251
BOMBAY	1.0000	1.0000	1.0000
CALI	0.9556	0.9233	0.9392
DERMASON	0.9099	0.9111	0.9105
HOROZ	0.9682	0.9456	0.9567
SEKER	0.8639	0.9852	0.9206
SIRA	0.8825	0.8121	0.8458
accuracy			0.9148
macro avg	0.9283	0.9299	0.9283
weighted avg	0.9157	0.9148	0.9143

Fig < 7 > Classification report for MLP after parameters tuning (test set)

Classification Report for SVM:

	precision	recall	f1-score
BARBUNYA	0.9182	0.9321	0.9251
BOMBAY	1.0000	1.0000	1.0000
CALI	0.9558	0.9294	0.9425
DERMASON	0.9034	0.9097	0.9065
HOROZ	0.9684	0.9534	0.9608
SEKER	0.8602	0.9852	0.9185
SIRA	0.8882	0.7989	0.8412
accuracy			0.9137
macro avg	0.9277	0.9298	0.9278
weighted avg	0.9146	0.9137	0.9130

Fig < 8 > Classification report for SVM after parameters tuning (test set)

Classification Report for SVM:

	precision	recall	f1-score
BARBUNYA	0.9251	0.9321	0.9286
BOMBAY	1.0000	1.0000	1.0000
CALI	0.9589	0.9294	0.9439
DERMASON	0.8989	0.9154	0.9071
HOROZ	0.9685	0.9560	0.9622
SEKER	0.8753	0.9852	0.9270
SIRA	0.8887	0.8027	0.8435
accuracy			0.9163
macro avg	0.9308	0.9315	0.9303
weighted avg	0.9168	0.9163	0.9156

Fig < 9 > Classification report for SVM using default parameters (test set)

Classification Report for Random Forest Accuracy:

	precision	recall	f1-score
BARBUNYA	0.9195	0.9057	0.9125
BOMBAY	1.0000	1.0000	1.0000
CALI	0.9467	0.9264	0.9364
DERMASON	0.8930	0.9182	0.9054
HOROZ	0.9633	0.9508	0.9570
SEKER	0.8993	0.9680	0.9324
SIRA	0.8699	0.8121	0.8400
accuracy			0.9126
macro avg	0.9274	0.9259	0.9263
weighted avg	0.9125	0.9126	0.9121

Fig < 10 > Classification report for RF using default parameters (test set)

Classification Report for MLP:

	precision	recall	f1-score
BARBUNYA	0.9313	0.9208	0.9260
BOMBAY	1.0000	1.0000	1.0000
CALI	0.9479	0.9479	0.9479
DERMASON	0.8934	0.9224	0.9077
HOROZ	0.9629	0.9404	0.9515
SEKER	0.8407	0.9877	0.9083
SIRA	0.8989	0.7590	0.8230
accuracy			0.9089
macro avg	0.9250	0.9254	0.9235
weighted avg	0.9107	0.9089	0.9077

Fig < 11 > Classification report for MLP using default parameters (test set)