UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

MSc. in Machine Learning and Data Mining (MLDM)

Machine Learning - Fundamentals and algorithms

Support Vector Machines Applied on Dry Beans Dataset

	Contribution	
	Part 1 (Practice)	Part 2 (Go wild)
Fatemeh Gol Pour	31%	31%
Kawtar Ezzati	31%	31%
Marieme Asselman Tafristan	31%	31%
GenerativeAI* (ChatGPT)	7%	7%
Total =	100%	100%

Table 2: Team members contributions

Supervised by:

Sri Kalidindi
Richard Serrano

* : ChatGPT was mainly used as a search engine (to understand the role of the hyperparameters, the difference between normalization and scaling, the data preprocessing techniques, etc.), and as a grammar correction tool.

March 18, 2024

Acknowledgments

At the end of this work, we would like to thank all those who have participated in any way in the realization of this project, and who have contributed to making it a rewarding experience.

In particular, we would like to express our sincere gratitude to Mr. **Sri Kalidindi** for his invaluable support and guidance throughout the process of completing this project.

We would also like to thank Mr. **Rémi Eyraud** for all of his time and effort in assisting us in learning the fundamentals of Machine Learning.

Last but not least, we would like to thank Mr. **Richard Serrano** for his nice and gentle reminders.

List of Tables

List of Figures

Contents

Introduction

In the field of data science and machine learning, the utilization of Support Vector Machines (SVMs) stands out as a powerful tool for classification tasks. SVMs excel in separating data points into distinct classes by finding the optimal hyperplane that maximizes the margin between classes. However, the effectiveness of SVMs heavily relies on various factors, including the quality of the dataset and the appropriate selection of hyperparameters.

This project focuses on enhancing the performance of SVMs through a systematic approach involving data normalization, hyperparameter tuning, and rigorous testing. The project begins with acquiring a dataset relevant to the classification task at hand. Often, real-world datasets contain inconsistencies, outliers, or variations in scale among features, which can interfere with the performance of machine learning algorithms like SVMs.

To mitigate these issues and ensure optimal performance, the dataset undergoes a preprocessing step where normalization techniques are applied. Normalization transforms the data into a standardized format, ensuring that all features contribute equally to the learning process and preventing any one feature from dominating due to its scale.

Following data normalization, the project proceeds to construct an SVM model. SVMs are known for their versatility in handling linear and non-linear classification tasks. By employing kernel functions, SVMs can efficiently map data points into higher-dimensional spaces, enabling complex decision boundaries to be constructed.

However, the success of SVMs depends on the appropriate selection of hyperparameters such as the choice of kernel, regularization parameter (C), and kernel parameters (e.g., gamma for RBF kernel). In this project, grid search technique is employed to tune these hyperparameters effectively.

Once the SVM model is constructed and hyperparameters are tuned, it undergoes rigorous testing using evaluation metrics such as accuracy, precision, recall, and F1-score. Testing helps assess the generalization performance of the model on unseen data and ensures that it can effectively classify instances from different classes.

Overall, this project aims to demonstrate the effectiveness of SVMs in classification tasks through a systematic approach that emphasizes data normalization, hyperparameter tuning, and thorough testing. By optimizing the SVM model, this project seeks to provide insights into best practices for enhancing classification performance in various domains.

Chapter 1

Part 2 (Go Wild!)

1.1 Presentation of the dataset Dry Beans

1.1.1 Data source and overview

The dataset can be downloaded at [1]. Seven different types of dry beans were used to collect this data, taking into account the features such as form, shape, type, and structure by the market situation. A computer vision system was developed to distinguish seven different registered varieties of dry beans with similar features in order to obtain uniform seed classification. For the classification model, images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera. Bean images obtained by computer vision system were subjected to segmentation and feature extraction stages, and a total of 16 features (12 dimensions and 4 shape forms) were obtained from the grains. The dataset contains 13,611 instances, 16 features, and one categorical target representing the type of grains.

Figure 1.1 shows the types of dry beans that were part of the study [2]. To be more specific, 7 different classes were used:

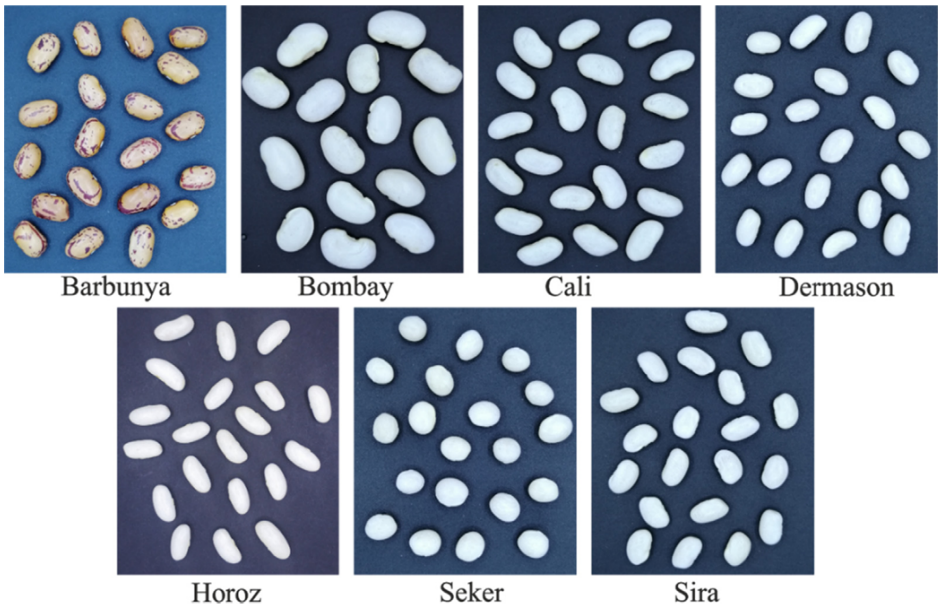

Figure 1.1: Types of dry beans used in the study

1.1.2 Data distribution

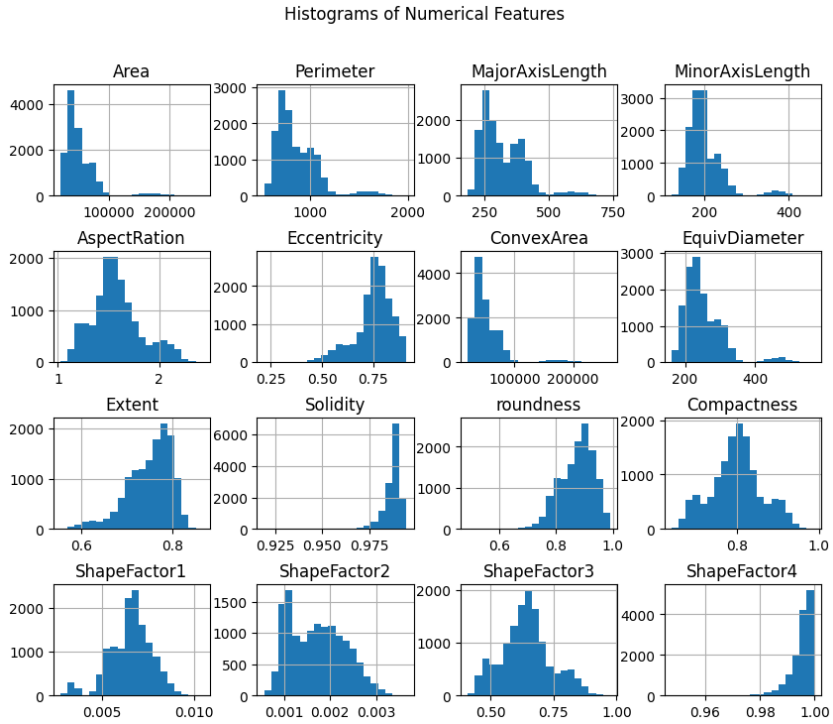

Figure 1.2: Histograms showing the distribution of numerical features

1.1.3 Correlation

Figure 1.3 shows the correlation heatmap. Area is strongly correlated with ConvexArea, EquivDiameter, Perimeter, MajorAxisLength, and MinorAxisLength and negatively correlated with ShapeFactor1. Eccentricity is strongly correlated with AspectRation and negatively correlated with Compactness, ShapeFactor3, and ShapeFactor2. Extent is not strongly correlated with any of the features.

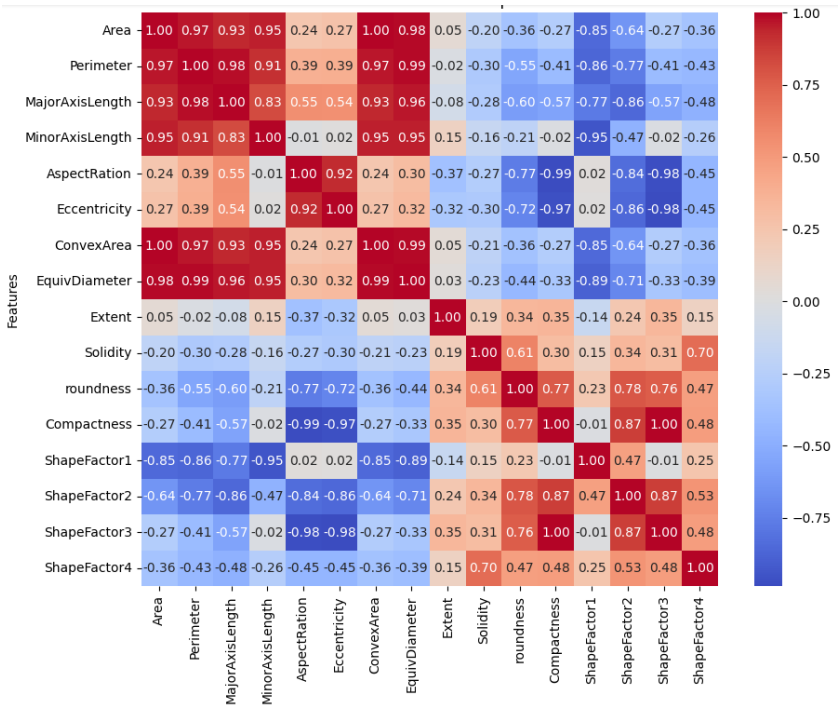

Figure 1.3: Correlation heatmap

1.1.4 Imbalance

Figure 1.4 shows the slight imbalance in the dry beans dataset. 26.1% of dry beans are from the class DERMASON, meanwhile, less than 4% are from the class BOMBAY.

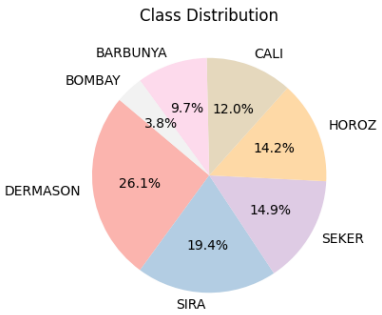

Figure 1.4: Dataset Imbalance

1.1.5 Outliers

Boxplots were used in figure 1.5 to visualize the distribution of each feature. Individual points that are far away from the main cluster of data points within the box (fall outside the whiskers of the boxplot) are likely to be outliers.

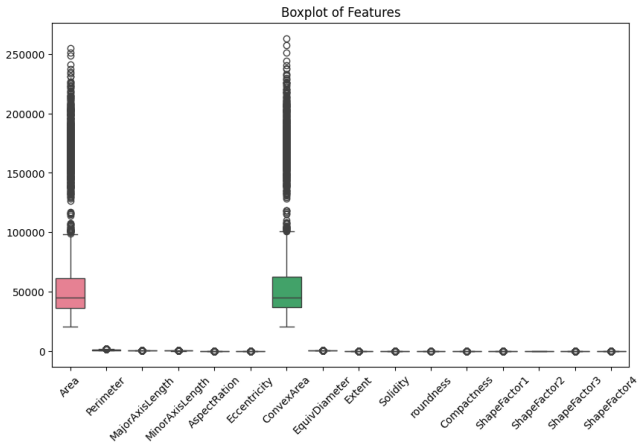

Figure 1.5: Outliers

1.2 SVM Principles

1.2.1 SVM Mechanisms

Support Vector Machine (SVM) is a powerful and versatile supervised learning algorithm used for both classification and regression tasks. The primary goal of SVM is to find the hyperplane that best separates data points into different classes while maximizing the margin, which is the distance between the hyperplane and the nearest data points from each class. This optimal hyperplane is the one that achieves the maximum margin and ensures the best generalization to unseen data [3].

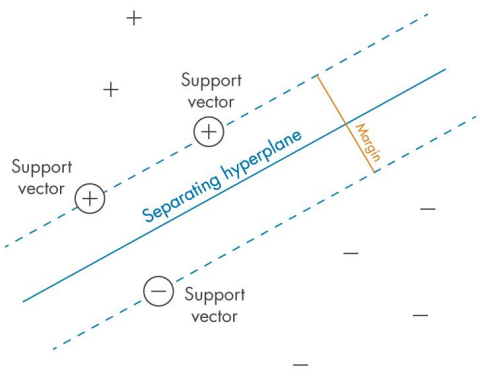

Figure 1.6: SVM illustration

The margin is defined as the distance between the hyperplane and the closest data point from each class. SVM aims to find the hyperplane that maximizes this margin, as it provides a measure of robustness to the model and improves its ability to generalize to new data points.

Support Vectors are the data points that lie closest to the hyperplane and are crucial in defining its position and orientation. These points are the ones that determine the margin and influence the decision boundary of the classifier.

1.2.2 SVM hyperparameters

Kernel Trick: SVM can efficiently handle non-linear decision boundaries by using the kernel trick. This technique allows SVM to implicitly map the input features into a higher-dimensional space where the data becomes linearly separable. Common kernel functions include **linear** (creates a linear decision boundary), **polynomial** (allows the model to learn polynomial decision boundaries of varying degrees), radial basis function **RBF** (a Gaussian kernel, which can capture non-linear decision boundaries), and **sigmoid** (which can capture non-linear decision boundaries similar to the logistic function).

Regularization Parameter (C): This hyperparameter controls the trade-off between maximizing the margin and minimizing the classification error. A small value of C leads to a wider margin but may result in misclassification of some data points, while a large value of C leads to a narrower margin but may overfit the training data.

Kernel Parameters: Depending on the chosen kernel function (e.g., RBF kernel), SVM may have additional hyperparameters such as gamma for RBF kernel, which controls the influence of individual training samples on the decision boundary [4].

1.3 Hyperparameters Tuning

To find the best combination of hyperparameters, we use **Grid Search**, which is a brute-force approach used in machine learning to find the optimal set of hyperparameters for a model. It works by exhaustively searching through a specified subset of hyperparameter combinations, evaluating each combination using cross-validation (training the model on different subsets of the training data and evaluating its performance on a validation set), and selecting the combination that yields the best performance metric [5].

Gamma	C	Kernel			
		Linear	Poly	RBF	Sigmoid
Auto	0.1	0.927	0.864	0.923	0.816
	1	0.926	0.906	0.929	0.729
	10	0.928	0.923	0.930	0.720
	100	0.927	0.927	0.930	0.718
Scale	0.1	0.927	0.864	0.923	0.816
	1	0.926	0.907	0.929	0.730
	10	0.928	0.923	0.931	0.719
	100	0.927	0.927	0.930	0.718

Table 1.1: Accuracy values for different hyperparameter combinations

As table 1.1 shows, the model performs the best when considering the combination *{'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}*.

1.4 Model evaluation

Evaluation in the context of machine learning refers to the process of assessing the performance and effectiveness of a trained model on a dataset. It involves using various metrics and techniques to analyze how well the model generalizes to unseen data and accomplishes the intended task. The goal of evaluation is to understand the model's strengths and weaknesses, identify areas for improvement, and make informed decisions about model selection, tuning, and deployment.

To evaluate a model, it's important to calculate various performance metrics:
Precision: is an important metric used to evaluate the performance of a binary classification model. It measures the proportion of correctly predicted positive instances out of all instances predicted as positive by the model.

Recall: also known as sensitivity or true positive rate (TPR), is a metric used to evaluate the performance of a binary classification model. It measures the proportion of actual positive instances that are correctly identified by the model.

The F1-score: The F1-score, also known as the F1 measure, is a metric used to assess the accuracy of a binary classification model. It is the harmonic mean of precision and recall, providing a single score that balances both metrics.
These are the results of our metrics after evaluating the SVM model:

Metric	Value
Accuracy	0.9346
Precision	0.9351
Recall	0.9346
F1-score	0.9348

Table 1.2: Model Evaluation Metrics

1.5 Influence of outliers and Imbalance on model's performance

1.5.1 Influence of outliers

To check how outliers are affecting the performance of the model, we first train the model on the original dataset, we evaluate then the model's performance metrics, and next, we remove the outliers from the initial dataset, we retrain the model on the outlier-removed dataset, and

finally we evaluate the new model's performance metrics on the testing dataset. We compare the performance metrics before and after removing outliers to assess the impact of outliers on model performance.

Performance Metric	Original Dataset	Outliers Removed Dataset
Accuracy	0.9346	0.8777
Precision	0.9351	0.8484
Recall	0.9346	0.8777
F1-score	0.9348	0.86062

Table 1.3: Performance Metrics Comparison (before and after removing outliers)

Accuracy: The accuracy on the original dataset (93.46%) is higher than the accuracy on the dataset with outliers removed (87.77%). This indicates that the presence of outliers may have a positive impact on overall accuracy, possibly due to the inclusion of important information from those outliers.

Precision: Precision measures the proportion of true positive predictions among all positive predictions. In this case, precision is higher on the original dataset (93.51%) compared to the dataset with outliers removed (84.84%). This suggests that the model on the original dataset is better at correctly identifying positive instances, with fewer false positives.

Recall: Recall measures the proportion of true positive predictions among all actual positive instances. In this case, recall is higher on the original dataset (93.46%) compared to the dataset with outliers removed (87.77%). This suggests that the model on the original dataset is better at capture actual positive instances.

F1-score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance. The F1-score is higher on the original dataset (93.48%) compared to the dataset with outliers removed (86.06%). This further confirms that the model performs better overall on the original dataset.

1.5.2 Influence of imbalance

1.5.2.1 Using class weights

Assigning different weights to classes can help alleviate the impact of class imbalance. We used the class_weight parameter in SVM to assign higher weights to minority classes. This allows the model to pay more attention to the minority class during training. Using 'balanced' mode automatically adjusts class weights inversely proportional to class frequencies.

Performance Metric	Original Model	Weighted Model
Accuracy	0.9346	0.9276
Precision	0.9351	0.9295
Recall	0.9346	0.9276
F1-score	0.9348	0.9280

Table 1.4: Performance Metrics Comparison (before and after using class weights)

1.5.2.2 Using SMOTE

we used SMOTE which is a technique used to address class imbalance by generating synthetic samples of the minority class, thereby balancing the dataset and improving the performance of machine learning models, especially in scenarios where class imbalance is prevalent.

Performance Metric	Original Dataset	Balanced Dataset
Accuracy	0.9346	0.9324
Precision	0.9351	0.9332
Recall	0.9346	0.9324
F1-score	0.9348	0.9326

Table 1.5: Performance Metrics Comparison (before and after using SMOTE)

After evaluating the model's performance on both the original dataset and the balanced dataset, we observe that the differences in performance metrics between the two datasets are minimal. Despite the slight decrease in accuracy, precision, recall, and F1-score on the balanced dataset compared to the original dataset, the performance remains consistently high. This suggests that balancing the dataset using techniques such as SMOTE and weighted classes does not significantly impact the model's overall performance. Therefore, when dealing with imbalanced datasets, applying these techniques can effectively address class imbalance without compromising the model's predictive ability.

1.5.3 Influence of noise

We tried feature engineering methods such as creating interaction features (polynomial features) and dimensionality reduction (PCA) to handle the noise.

1.5.3.1 Using PCA

We tried to improve the results by implementing PCA on the dataset.

Principal Component Analysis (PCA) is used to denoise and reduce the dimensionality of the dataset. It does not eliminate the noise but can reduce it. We performed a grid search on PCA to tune the number of components to be considered, and the best number of components we received was 15, which in comparison to the overall number of our features(16), indicates that there will not be any significant change in the results after performing PCA with these many components.

Performance Metric	Original Dataset	Reduced Dataset
Accuracy	0.9346	0.9346
Precision	0.9351	0.9351
Recall	0.9346	0.9346
F1-score	0.9348	0.9348

Table 1.6: Performance Metrics Comparison (before and after using PCA)

After evaluating the model's performance on both the original dataset and the reduced dataset, we observe that there is no difference in their performances. Reducing the dimensionality of the dataset with PCA did not impact the model's overall performance.

1.5.3.2 Using polynomial features

By generating polynomial features from existing features we tried to capture nonlinear relationships in the data. Feature engineering can help improve the discriminatory power of the model and make it more robust to noise.

Performance Metric	Original Dataset	Feature Engineered Dataset
Accuracy	0.9346	0.9305
Precision	0.9351	0.9311
Recall	0.9346	0.9305
F1-score	0.9348	0.9308

Table 1.7: Performance Metrics Comparison (before and after using polynomial features)

After evaluating the model's performance on both the original dataset and the dataset with additional polynomial features, we observe that the differences in performance metrics between the two datasets are minimal. Despite the slight decrease in accuracy, precision, recall, and F1-score on the new dataset compared to the original, the performance remains consistently high. This suggests that handling noise in the dry bean dataset using feature engineering methods such as dimensionality reduction or creating interaction features does not significantly impact the model's overall performance.

Conclusion

The objective of this project was to select a dataset meeting specific criteria, perform data cleaning procedures, develop an SVM model, apply it to the dataset, assess the model's performance, and employ various machine learning strategies to address limitations within the dataset in order to achieve better outcomes.

For this purpose, we chose a dataset called Dry Beans with 13,611 instances, 16 features, and one categorical target representing the type of beans. In our dataset, we had some imbalance between the seven different classes, two of the features had outliers, and there were also signs of noise.

At first, we defined an SVM model, trained it, and evaluated its performance, then tuned the major hyperparameters (C, gamma and the kernel) using Grid Search to find the best model. After that, we tried to improve the performance of this model, by monitoring the influence of each of the constraints in our database on the performance of our model.

We tried various approaches to address each constraint in our dataset. Initially, we focused on removing outliers from the data to see if it would improve our model's performance. However, even after training our model on the cleaned dataset and comparing the results with the original dataset, we did not observe any improvement.

Next, we tackled the class imbalance issue using two different methods: adjusting the class_weight parameter in the SVM model and employing the Synthetic Minority Over-sampling Technique (SMOTE) to balance the classes. Despite evaluating our model on the balanced datasets generated from these methods, we found that they did not lead to any significant improvement in our results.

Ultimately, we aimed to reduce the impact of noise in our dataset by implementing two feature engineering techniques: dimensionality reduction using Principal Component Analysis (PCA) and creating interaction features through polynomial features. Unfortunately, neither of these methods resulted in a noticeable enhancement in our model's performance.

Bibliography

[1] "Dry Bean Dataset." UCI Machine Learning Repository, 2020. DOI: https://doi.org/10.24432/C50S4B.

[2] 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.

[3] "What are support vector machines (SVMs)?." Available at https://www.ibm.com/topics/support-vector-machine.

[4] "Visualizing the effect of hyperparameters on support vector machines." Available at https://towardsdatascience.com/visualizing-the-effect-of-hyperparameters-on-support-vector-machines-b9eef6f7357b.

[5] "Grid search for model tuning." Available at https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e.