Support Vector Machines

MGSC 695: Optimization for Data Science

Final Project

Master of Management Analytics

Desautels Faculty of Management

by

Oyundari Batbayar, Samia Belmadani, Wenya Cai, & Rodrigo Castro Canal

McGill University
March 5th, 2024

Contents

1 Introduction

1.1 Project Summary

This project focuses on Support Vector Machine (SVM) to classify grapes into Kecimen or Besni varieties, based on the data provided by a research paper written by Cinar, Koklu, and Tasdemir (2020). It further employs multiple SVM methodologies and benchmarks the resulting models against a black-box reference. The findings of this project have the potential to benefit business activities by enhancing efficiency in sorting and quality assessment processes, thereby contributing to improved product standards, as mentioned in Appendix 6.1.

2 Data Description

2.1 Introduction to the Dataset

The dataset is a comprehensive collection of 900 grape samples obtained through image processing techniques. It encompasses a variety of morphological features, including area, major and minor axis lengths, eccentricity, convex area, extent, and perimeter, alongside the classification label for each sample.

2.2 Target Variable

The target variable "Class" within the dataset categorizes each grape sample into one of two distinct varieties: Kecimen and Besni. Kecimen grapes are known for their specific shape, size, and color attributes that set them apart from Besni grapes, which have a different set of qualities in terms of sweetness, skin texture, and color.

2.3 Predictor Variables

Following are the morphological features of each grape, that serves as predictors:

- Area: This represents the size of the grape, measured in pixels. The area varies significantly across samples, indicating a range of grape sizes.
- MajorAxisLength: The length of the longest line that can be drawn through the grape, also measured in pixels. This feature helps in understanding the elongation of the grape.
- MinorAxisLength: The length of the shortest line perpendicular to the major axis, providing insight into the grape's width.
- Eccentricity: A measure of the deviation of the shape of the grape from a perfect circle, where 0 indicates a perfect circle and values closer to 1 indicate more elongated shapes.
- ConvexArea: The area of the smallest convex polygon that can enclose the grape's area, also measured in pixels.
- Extent: The ratio of the grape's area to the area of the bounding box that contains the grape. It measures how much of the bounding box is filled by the grape.
- Perimeter: The total length of the boundary of the grape, measured in pixels. This feature reflects the complexity of the grape's shape.

Note that Business Implications and Exploratory Data Analysis can be found in the Appendix 6.2.

3 Support Vector Machines

3.1 Intuition

Now, let's take a step back and start our initial examination of SVM with the crucial concept of margins. What do they actually do? SVMs endeavor to identify a hyperplane that most efficiently separates distinct classes within the feature space. The optimal hyperplane is characterized by its ability to maximize the margin, defined as the distance between the hyperplane and the nearest points of each class. The rationale behind prioritizing a larger margin is to enhance the classification robustness. To illustrate this principle, consider a scenario depicted by Figure 1 where positive training examples are marked by 'x' and negative training examples by 'o'. Alongside these, a decision boundary (the line) mathematically represented as $\beta^T x = 0$,[1] also known as a separating hyperplane is drawn.

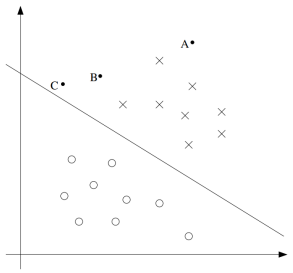

Figure 1: A Simple Illustration of a Margin

In Figure 1, note that point A is significantly distant from the decision boundary, which implies a strong confidence in the prediction of $y = 1$ for this point. In contrast, point C is situated near the decision boundary. Although it resides on the predictive side for $y = 1$, the proximity suggests that minor alterations to the boundary could result in a prediction shift to $y = 0$. Point B occupies a position between these two, underscoring the principle that the distance of a point from the separating hyperplane directly influences our prediction confidence. From here, we see that ideally a decision boundary that maximizes this distance for all training examples enhances both the accuracy and confidence of predictions, which will further be explained in our section about geometric margins.

3.2 Notation

To facilitate our discussion on SVMs, it is important to introduce specific notations for classification. We consider a linear classifier addressing a binary classification task, using labels y and features x. Here, y will be represented within the set {-1, 1}, diverging from the conventional {1, 0} for class labels. Instead of parameterizing the classifier with a vector β, we employ parameters w and b, defining the classifier as $h_{wb}(x) = g(w^T x + b)$, where $g(z) = 1$ for $z \geq 0$ and $g(z) = -1$ otherwise. This notation distinctly separates the intercept term b from the parameter vector w, moving away from the norm of augmenting the input feature vector with an additional coordinate $x_0 = 1$. Consequently, b assumes the role of intercept, previously denoted by β_0, with w representing the remaining parameters. This formulation allows the classifier to directly predict values of 1 or -1, eliminating the logistic regression's intermediary step of estimating $p(y = 1)$.[2]

[1] β represents a vector of coefficients and x is a feature vector.

[2] In contrast to the direct prediction approach employed by SVMs, logistic regression operates by estimating the probability that a given observation belongs to a particular class. Specifically, logistic regression calculates $p(y = 1|\mathbf{x})$, the probability that the class label y is 1 given the feature vector $\mathbf{x}$. This is achieved through the logistic function, $\sigma(z) = \frac{1}{1+e^{-z}}$, where $z = \mathbf{w}^T\mathbf{x} + b$. The output of the logistic function lies in the interval (0, 1), serving as a probabilistic prediction. A threshold, commonly 0.5, is then applied to this probability to classify observations into either class 1 ($p > 0.5$) or class 0 ($p \leq 0.5$). In the framework of SVMs, the strategy diverges significantly. The SVM model directly assigns observations to classes based on the sign of $g(z) = w^T\mathbf{x} + b$, bypassing the probabilistic estimation stage inherent to logistic regression. This direct assignment underlines SVM's classification mechanism, which relies on the geometric margins from the decision boundary rather than probabilistic thresholds. Hence, while logistic regression provides a probability that offers insight into the model's confidence in its predictions, SVMs focus on maximizing the margin between classes for classification, offering a different perspective on model confidence derived from the distance of data points from the decision boundary.

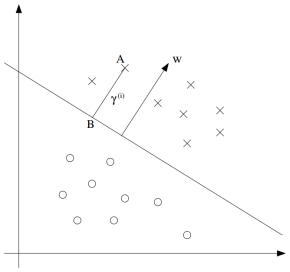

Figure 2: A Simple Illustration of Geometric Margins

3.3 Geometric Margins

In the framework of SVMs, geometric margins play a pivotal role in determining the optimal decision boundary. The geometric margin is essentially the minimum distance from the decision boundary to any training example. To understand its derivation and significance, let's delve deeper into its mathematical underpinnings.

In Figure 2, see the data point A characterized by feature vector x_i and associated class label $y_i = 1$.[3] The geometric margin, γ, delineates A's perpendicular distance to the decision boundary, defined by the hyperplane equation $w^T\mathbf{x} + b = 0$. This boundary discriminates between the binary classes under consideration.

To compute γ, we utilize the unit vector $\frac{\mathbf{w}}{\|\mathbf{w}\|}$, which points in the direction of $\mathbf{w}$ and represents the orientation of the decision boundary. The distance of any point x_i from the decision boundary can be geometrically projected as $\gamma_i = y_i \left(\frac{\mathbf{w}^T}{\|\mathbf{w}\|}\mathbf{x}_i + \frac{b}{\|\mathbf{w}\|} \right)$, where y_i adjusts the direction of the margin depending on the class label. The derivation of γ_i starts with the understanding that the shortest distance from a point to a plane is along the line perpendicular to the plane. Hence, for point B on the boundary:

$$\mathbf{w}^T \left(\mathbf{x}_i - \gamma_i \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} \right) + b = 0.$$

Solving this equation for γ_i provides the explicit formula for the geometric margin:

$$\gamma_i = \frac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|},$$

highlighting that γ_i is indeed the signed distance from the decision boundary to the point x_i.

The optimization objective in SVMs, therefore, focuses on finding w and b [4] that maximize the minimum geometric margin γ across all training examples, encapsulated as:

$$\gamma = \min_{i=1,\ldots,n} \gamma_i.$$

This optimization ensures that the decision boundary is as far away as possible from the nearest training examples, effectively creating a "buffer zone" around the boundary. Such a maximization of the geometric margin is what contributes to the distinctive robustness and efficacy of SVM classifiers.

[3] A pair x_i and y_i is called a training example.

[4] It is also worth noting that multiplying w and b by a scalar factor k does not alter the geometric margin. This property is elucidated by the equation:

$$\gamma_{i\,scaled} = \frac{(k\mathbf{w})^T\mathbf{x}_i + kb}{\|k\mathbf{w}\|} = \gamma_i,$$

demonstrating that the distance measurement is unaffected by the magnitude of the parameters but is instead contingent on their direction. This invariance signifies that the SVM's decision function, and thus its classification capability, remains unchanged under parameter scaling, underscoring the model's robustness to transformations. However, note that SVM performance is sensitive to input feature scaling due to its impact on Euclidean distances critical for margin calculation.

3.4 Optimization Problem

Let us consider the scenario where our training dataset is linearly separable. The crucial question then becomes: how do we identify the hyperplane that not only separates the positive and negative classes but also maximizes the geometric margin we defined above? To address this, we formulate an optimization problem that seeks to find the hyperplane offering the largest possible margin between the two classes. The problem is defined as follows:

$$\begin{aligned} \underset{w,b}{\text{maximize}} \quad & \gamma \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq \gamma, \ i = 1, \ldots, n, \\ & \|\mathbf{w}\| = 1. \end{aligned} \tag{1}$$

In this formulation, γ represents the geometric margin we aim to maximize, subject to the constraint that every data point $(\mathbf{x}_i, y_i)$ in the training set are correctly classified with a margin at least as large as γ, while normalizing the weight vector $\mathbf{w}$ to ensure the margin is defined in a consistent scale.

Addressing the challenge of finding the optimal hyperplane in SVMs involves transforming an initially non-convex problem into a convex one. The original formulation, constrained by $\|\mathbf{w}\| = 1$, is inherently non-convex, making direct application of standard optimization techniques infeasible. To navigate around this obstacle, an alternative formulation is proposed where the objective shifts towards maximizing $\hat{\gamma}/\|\mathbf{w}\|$, subject to ensuring that all data points maintain a minimum functional margin of $\hat{\gamma}$.[5] This approach, while eliminating the explicit normalization constraint on $\mathbf{w}$, introduces a non-convex objective function $\hat{\gamma}/\|\mathbf{w}\|$, still rendering the problem challenging for off-the-shelf optimization solutions.

$$\begin{aligned} \underset{\mathbf{w},b}{\text{maximize}} \quad & \frac{\hat{\gamma}}{\|\mathbf{w}\|} \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq \hat{\gamma}, \ i = 1, \ldots, n. \end{aligned} \tag{2}$$

The breakthrough then comes from applying a key insight: by leveraging the concept introduced in Footnote 3 about the parameters w and b being scaled arbitrarily without altering the classification decision, we set the functional margin $\hat{\gamma} = 1$, simplifying the maximization of $\hat{\gamma}/\|\mathbf{w}\|$ to minimizing $\|\mathbf{w}\|^2$. This yields a convex quadratic optimization problem with linear constraints:

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \ i = 1, \ldots, n. \end{aligned} \tag{3}$$

This primal formulation aligns with the requirements for quadratic programming, making it solvable with existing optimization software and effectively identifying the optimal margin classifier.

3.5 The Dual Form

We can reformulate the above primal function with Lagrange multipliers α_i, where $\alpha_i \geq 0$ for all i:

$$L = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{L} \alpha_i[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1] \tag{4}$$

We wish to find $\mathbf{w}$ and b which minimize this Lagrangian, and α which maximizes it, subject to $\alpha_i \geq 0$ for all i. This is done by differentiating L with respect to $\mathbf{w}$ and b, and setting the derivatives to zero:

[5] Since the geometric and functional margins are related by $\gamma = \hat{\gamma}/\|\mathbf{w}\|$, this will give us the answer we want. We also got rid of the constraint $||\mathbf{w}|| = 1$. The difference between functional and geometric margins is that the geometric margin is a scale-invariant version of the functional margin. It takes into account the normalization of the weight vector.

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \implies \mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \mathbf{x}_i \tag{5}$$

$$\frac{\partial L}{\partial b} = 0 \implies \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{6}$$

Substituting these back into the Lagrangian gives a new formulation which we aim to maximize:

$$L = \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{7}$$

This dual formulation, dependent on α, is maximized subject to $\alpha_i \geq 0$ for all i, and $\sum_{i=1}^{L} \alpha_i y_i = 0$. It is a convex quadratic optimization problem that returns α and from it $\mathbf{w}$. The remaining task is to calculate b.

A Support Vector $\mathbf{x}_s$ that lies closest to the separating hyperplane in SVM satisfies:

$$y_{(s)}(\mathbf{x}_s \cdot \mathbf{w} + b) = 1 \tag{8}$$

By averaging over all Support Vectors, we find b:

$$b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s) \tag{9}$$

where N_s is the number of support vectors, S is the set of indices of the support vectors, α_m are the Lagrange multipliers for the support vectors, and y_m and $\mathbf{x}_m$ are the labels and feature vectors of the support vectors, respectively. We now have $\mathbf{w}$ and b defining our SVM's separating hyperplane and are equipped to classify new data points efficiently.

3.6 Implementation

Given our use case, dataset, and an understanding of SVM, we employ different SVM optimization methods, including the standard dual form with linear separability assumption and with gradient descent, and SVM implementations with hinge, quadratic loss, and RBF kernel.

3.6.1 Linear SVM Classifier

The Dual Form SVM, assuming linear separability, optimizes the margin between classes by solving a Lagrangian dual problem. The objective is to maximize the function:

$$L(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \tag{10}$$

where α_i are the Lagrange multipliers, y_i are the class labels, and $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ is the kernel-induced inner product between the feature vectors. This formulation encapsulates the intuition that the optimal decision boundary is influenced most strongly by the data points nearest to it, known as support vectors. The constraints $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C$ ensure that the solution adheres to the margin maximization principle while allowing for some misclassifications, controlled by C. This problem is convex and quadratic, guaranteeing a global optimum solution.

3.6.2 Gradient Descent SVM

The SVM optimization via gradient descent aims to minimize the primal objective function, which incorporates both regularization for model complexity and a loss function for classification error. The primal objective function is defined as:

$$J(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\max(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)) \tag{11}$$

where C is the regularization parameter, $\mathbf{w}$ is the weight vector, b is the bias, and n is the number of training samples. Our Python function `svm_gradient_descent` is an implementation of a linear SVM classifier using gradient descent for optimization.[6]

Gradient Descent Update Rule

The general update rule for parameters using gradient descent is given by:

$$\theta := \theta - \eta\frac{\partial J}{\partial \theta} \tag{12}$$

where θ represents the parameters ($\mathbf{w}$ or b) being optimized, η is the learning rate, and $\frac{\partial J}{\partial \theta}$ is the gradient of the objective function with respect to θ.

Derivation for w

The gradient of the regularization term with respect to $\mathbf{w}$ is straightforward:

$$\frac{\partial}{\partial \mathbf{w}}\left(\frac{1}{2}\|\mathbf{w}\|^2\right) = \mathbf{w} \tag{13}$$

For the hinge loss component, the gradient with respect to $\mathbf{w}$ depends on the condition of the data point:

- If $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) < 1$, the gradient is $-y_i\mathbf{x}_i$.
- If $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$, the gradient is 0.

Thus, the update rule for $\mathbf{w}$ when the hinge loss is activated is:

$$\mathbf{w} := \mathbf{w} - \eta\left(\mathbf{w} - C\sum_{i=1}^{n} y_i\mathbf{x}_i\right) \tag{14}$$

Derivation for b

The bias b update rule is derived only from the hinge loss component:

- If $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) < 1$, the gradient with respect to b is $-y_i$.
- If $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$, the gradient is 0.

Hence, the update rule for b is:

$$b := b + \eta C\sum_{i=1}^{n} y_i \tag{15}$$

In our implementation, for points with $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) < 1$: The weight update rule derives from the gradient of the loss function with respect to $\mathbf{w}$, which includes both the regularization term and the loss due to margin violation. The bias update rule $b+ = \eta y_i$ adjusts the hyperplane to correct for these margin violations. For points correctly classified and outside the margin: The weight update is influenced

[6] However, it is important to clarify that this function does not explicitly compute a loss value at each iteration; instead, it directly updates the weights and bias based on the gradient of the SVM's loss function. This part implicitly uses the principles of the hinge loss (which will be explained later) by adjusting weights and bias based on whether samples are correctly classified and outside the margin or not. However, it does not explicitly calculate the hinge loss as part of the optimization; rather, it applies the concept through updates that would reduce the hinge loss.

only by the regularization term, as these points do not contribute to the hinge loss. There's no need to update the bias since there's no violation to correct for. For more, read Footnote.[7]

3.6.3 The Hinge/Quadratic Loss SVM

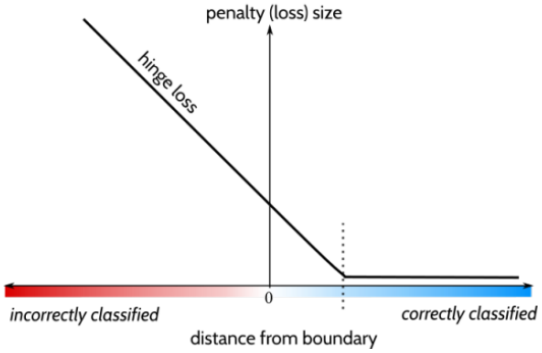

Figure 3: Hinge Loss Visualized

The hinge loss is a loss function used for training classifiers, most notably the SVM. When a data point's distance from the decision boundary is greater than or equal to 1, the loss size is 0. If the distance from the boundary is less than 1, we incur a loss. When it is on the boundary, then the loss size is 1. As illustrated in Figure 3 the correctly classified points will have a small loss size, while incorrectly classified instances will have a high loss size. High hinge loss indicates data points being on the wrong side of the boundary, and hence are misclassified; while a positive distance calls for low or zero hinge loss and correct classification.

Mathematical Description of Hinge Loss

For $t = \pm 1$, the true label of a data point, and y, a raw output of the classifier's decision function for a data point, the hinge loss of the prediction y is defined as [8]

$$L(y) = \max(0, 1 - t \cdot y)$$

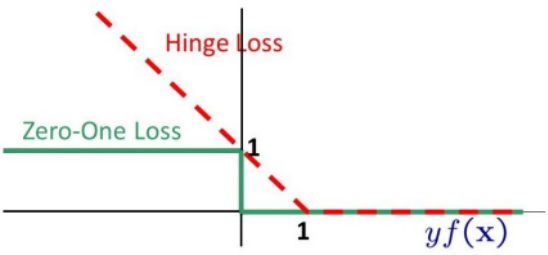

Figure 4: Hinge Loss vs Zero-one loss (Zero-one is for a fixed $t = 1$ and x-axis represents the prediction value.

[7] Adjusting w not only affects the direction and orientation of the decision boundary but also its margin. The regularization component ensures the model remains simple and generalizable, while the component related to the hinge loss ensures misclassified points or points within the margin move towards correct classification. Meanwhile, adjusting b shifts the decision boundary closer or further from the origin without changing its orientation. This helps in fine-tuning the position of the decision boundary to ensure correct classification of points, especially those near the margin. The combination of these updates allows the SVM to find a decision boundary that both separates the classes well and maintains a maximal margin, adhering to the principle of structural risk minimization.

[8] Note that y should be raw output of the classifier's decision function, not the predicted class label. For instance, in linear SVMs, $y = \mathbf{w}^T \cdot \mathbf{x} + b$, where $(\mathbf{w}, b)$ are the parameters of the hyperplane and $\mathbf{x}$ is the vector composed of input variables.

The plot shows that the Hinge loss penalizes predictions $y > 1$, corresponding to the notion of a margin in a SVM.

Soft Margin in SVM using Hinge Loss

The hinge loss function for soft margin SVM is defined by

$$L(y) = \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b)) \tag{4}$$

The goal of optimization is to minimize

$$\lambda\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b))$$

where the parameter $\lambda > 0$ determines the trade-off between increasing the margin size and ensuring that the $\mathbf{x}_i$ lie on the correct side of the margin. If the margin is high, then $\|\mathbf{w}\|$ is less and some misclassification is allowed for the second term.[9]

3.6.4 Radial Basis Function in SVM

The SVM with a Radial Basis Function (RBF) kernel is a powerful method for solving non-linear classification problems. The RBF kernel is defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2\right), \tag{16}$$

where $\gamma > 0$ is a parameter that determines the spread of the kernel, and $||\mathbf{x}_i - \mathbf{x}_j||^2$ is the squared Euclidean distance between the feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$.[10]

Objective Function and Constraints

1. Primal Problem:

First, we define the primal problem of the SVM which aims to maximize the margin between the classes by minimizing the following objective function:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i, \tag{17}$$

subject to the constraints:

$$y_i(\mathbf{w}^\top\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \forall i, \tag{18}$$

$$\xi_i \geq 0, \quad \forall i, \tag{19}$$

where $\phi(\mathbf{x}_i)$ is the function that maps the input features into a higher-dimensional space. The slack variable ξ_i for the i-th data point, is introduced to handle misclassifications and instances within the margin boundary. [11]

2. Introducing the RBF Kernel:

Since we're using an RBF kernel, we don't need to explicitly compute the mapping ϕ. Instead, we define the kernel function K, which computes the inner product in the feature space indirectly:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2\right). \tag{20}$$

[9] Read more here: https://www.niser.ac.in/ smishra/teach/cs460/23cs460/lectures/lec11.pdf

[10] Note the γ used here is different than the one used for functional/geometrical margins

[11] For more information and visualization on RBF Kernel, go to the Appendix 6.4.

3. Formulating the Lagrangian:

To transition from the primal to the dual problem, we introduce Lagrange multipliers α_i and μ_i for each constraint and construct the Lagrangian L:

$$L(\mathbf{w}, b, \xi, \alpha, \mu) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i(\mathbf{w}^\top\phi(\mathbf{x}_i) + b) - 1 + \xi_i\right] - \sum_{i=1}^{n}\mu_i\xi_i, \tag{21}$$

where $\alpha_i \geq 0$ and $\mu_i \geq 0$.

4. Deriving the Karush-Kuhn-Tucker (KKT) Conditions:

Taking partial derivatives of L with respect to $\mathbf{w}$, b, and ξ_i, and setting them to zero yields the KKT conditions which are necessary for optimality:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i \phi(\mathbf{x}_i) = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \phi(\mathbf{x}_i), \tag{22}$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n}\alpha_i y_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{n}\alpha_i y_i = 0, \tag{23}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \quad \Rightarrow \quad \alpha_i + \mu_i = C. \tag{24}$$

5. Constructing the Dual Problem:

Substituting the KKT conditions back into the Lagrangian, we eliminate $\mathbf{w}$, b, and ξ_i to get the dual problem:

$$\max_{\alpha}\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{25}$$

subject to the constraints:

$$0 \leq \alpha_i \leq C, \quad \forall i, \tag{26}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0. \tag{27}$$

The dual form seeks to maximize the margin between classes while penalizing instances that lie within the margin or are misclassified. The dual formulation is preferred because it only involves the dot products of input vectors, which are computed using the RBF kernel function. This approach allows us to handle the non-linearity of the data and find the optimal decision boundary in the transformed feature space.

4 Results

Table 1: Comparison of SVM Models and Implementation Accuracies

Model	Our Implementation	Black Box (Unscaled)	Black Box (Scaled)
Linear SVM (Unscaled)	0.82	0.85	—
Linear SVM (Scaled)	0.48	—	0.86
Gradient Descent SVM (Unscaled)	0.61	0.85	—
Gradient Descent SVM (Scaled)	0.87	—	0.86
Hinge Loss SVM	0.86	0.83	0.86
Quadratic Loss SVM	0.85	—	—
SVM with RBF Kernel	0.8667	—	0.8677

Linear SVM

On the unscaled data, our implementation achieved an accuracy of 0.82. Conversely, when applied to scaled data, the accuracy substantially declined to 0.48. This is very surprising. This severe difference suggests a significant impact of the scaling process on the model's efficacy, potentially highlighting a mismatch between the scaling method and the underlying data characteristics. Consequently, further investigation of the model parameters may be necessary to enhance both accuracy and stability.

Gradient Descent SVM

Our SVM here achieved an accuracy of 0.61. Conversely, when applied to scaled data, the accuracy substantially improved to 0.87. The accuracy of the black box SVM without scaling was found to be 0.85, while the accuracy with scaling improved slightly to 0.86 These results highlight the competitive performance of both the gradient descent SVM and the black box SVM, with scaling proving to be beneficial in enhancing classification accuracy.

Hinge Loss and Quadratic Loss Functions

Our hinge loss SVM implementation achieved an accuracy of 0.86 on the test data. For the black box approach, the LinearSVC class from scikit-learn is utilized. Initially, the model is trained on the unscaled training data, and its accuracy is assessed on the test set, yielding an accuracy of 0.83. Subsequently, the model is trained on scaled data, showcasing an improved accuracy of 0.86 on the test set. This comparison underscores the efficacy of both implementations, with the SVM model demonstrating comparable performance to the black box SVM, particularly again when applied to scaled data.

For the SVM model with quadratic loss function using the quadratic programming solver from the CVXOPT library, our trained SVM model is evaluated on a test dataset, yielding an accuracy of 0.85.

SVM with RBF Kernel

Our implementation here achieved classification accuracy of 0.8667 on the test set. Further, we compared the model with the black-box algorithm that had yielded a comparable accuracy of 0.8677 as well. The high accuracy of both models underscores the effectiveness of both models in accurately classifying instances. The RBF Kernel-based SVM model's ability to achieve similar accuracy to the black-box SVM further highlights its robustness and reliability, affirming its suitability for practical applications where performance is important.

5 Conclusion

In conclusion, while our current analysis has provided valuable insights into the performance of SVM models with different kernels, there are several avenues for further exploration and refinement. For now, we have chosen the RBF kernel with fixed values for C and gamma, yet future extensions of this work could involve conducting a grid search to identify optimal values for these hyperparameters. Additionally, our findings strongly suggest that the RBF kernel-based SVM model outperforms its linear counterpart, underscoring its effectiveness in capturing complex relationships within the dataset. We attribute this superiority to the RBF kernel's ability to flexibly model non-linear boundaries, which aligns well with the observed characteristics of our dataset during exploratory data analysis. Notably, the presence of normal distribution tendencies in our dataset indicates the presence of intricate relationships that are better captured by the RBF kernel. Furthermore, the poor performance of the initial linear SVM model hints at the dataset's non-linear separability. Moving forward, refining our model through hyperparameter tuning will be crucial for achieving even higher levels of predictive accuracy.

6 Appendix

6.1 Use Case: Business Application and Relevance

The project's business application and relevance are rooted in its potential to transform operational efficiency within the agricultural and food sectors. For instance, vineyards and food distributors can utilize the model to automate the sorting process, reducing the time and labor traditionally required to manually sort grape varieties, which can often be error-prone and inconsistent. This automation not only increases throughput but also ensures that only grapes that meet strict quality standards reach the market, reducing the risk of recalls and enhancing brand reputation.

On the other hand, the precise classification capabilities provided by the model could enable producers to target specific markets more effectively. For example, the Kecimen grapes, if they are sweeter, could be marketed for fresh consumption or desserts, while the Besni grapes could be better suited for wines or raisins, depending on their characteristics. By clearly differentiating their product offerings, producers can cater to niche markets and set higher prices. On a broader scale, such advancements demonstrate the practical benefits of integrating AI with traditional agricultural methods, showcasing a step forward in modernizing food systems and potentially leading to more sustainable and profitable industry practices. Additionally, the precision of SVM-based classification can assist grocery retailers in enhancing the customer shopping experience. By ensuring that only the best-quality grapes are on the shelves, retailers can build a reputation for offering premium products, which can increase customer satisfaction. For example, knowing that a store consistently provides high-quality Besni grapes, a customer who prefers them for their unique taste profile might choose this retailer over competitors. This level of quality control can also contribute to reducing food waste in stores, as higher quality produce tends to have a longer shelf life and less likelihood of being discarded due to spoilage. Such improvements in product quality and waste reduction are not only economically beneficial but also align with the growing consumer demand for sustainability in food sourcing.

6.2 Data Exploration

The Exploratory Data Analysis provides a visual and quantitative exploration of the dataset's features.

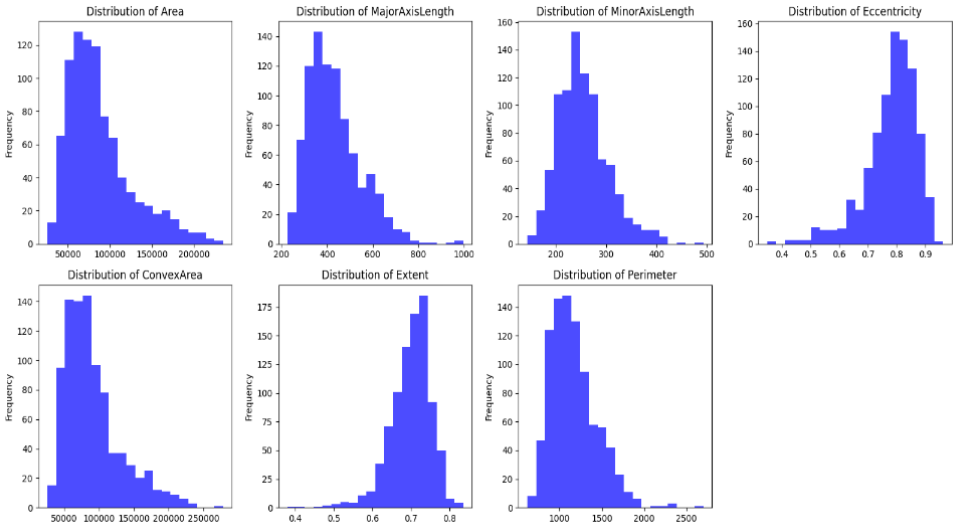

Figure 5: Histogram of Features

From the plotted histograms, we can see that most features have a unimodal distribution, with varying degrees of skewness. For instance, the 'Area' and 'ConvexArea' exhibit right-skewness, where a majority of the grapes have a smaller area.

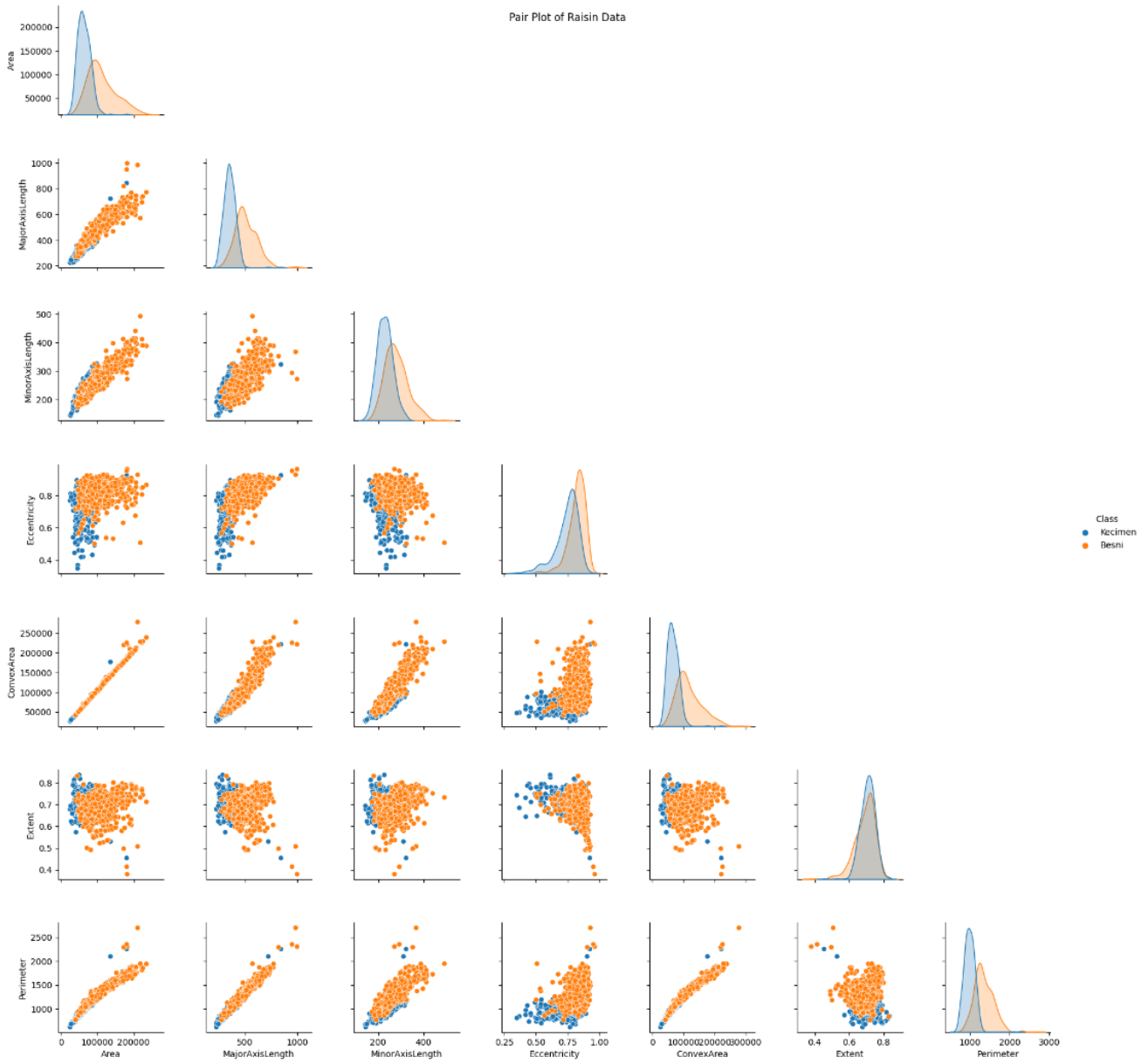

Figure 6: Pair Plot

The pair plot further elaborates on the relationship between each pair of features. The scatter plots combined with density plots allow us to see both the distribution of individual features and the bivariate relationships. From the scatter plots, we observe that some features, such as 'MajorAxisLength' and 'Area', show a strong positive correlation, indicating that as one increases, the other tends to increase as well.

Overall, the EDA aims to uncover underlying patterns, detect outliers, understand relationships, and identify features that are most indicative of the grape variety, which are crucial steps before applying machine learning models for classification.

6.3 Data Concerns

In this dataset, a primary concern is the potential for skewed distributions and outliers, as observed in the histograms in Appendix 6.2. Skewed distributions may impact the SVM model, which could distort the model's understanding of the data, leading to overfitting or poor generalization to new data samples. Addressing these could involve transformations, such as standardization.

Another concern involves the overlap between classes, as seen in the pair plot in Appendix 6.2, which could indicate that some features are not discriminative enough to distinguish between the classes. This could require hyperparameter tuning and play with multiple SVM approaches to improve the model's predictive ability.

6.4 Further Information on RBF Kernel

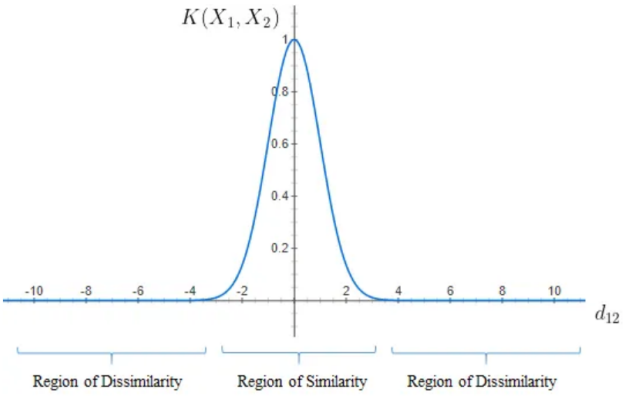

Figure 7: RBF Kernel with $\sigma = 1$

The graph illustrates the Radial Basis Function (RBF) kernel, a measure of similarity between two data points, X_1 and X_2, within the context of a Support Vector Machine (SVM). The RBF kernel is represented as:

$$K(X_1, X_2) = \exp(-\gamma \|X_1 - X_2\|^2)$$

where:

- The x-axis, denoted as d_{12}, represents the distance between the data points X_1 and X_2.
- The y-axis, $K(X_1, X_2)$, indicates the kernel value which quantifies the similarity between X_1 and X_2.
- The peak at $d_{12} = 0$ signifies the highest similarity score of 1 when X_1 and X_2 coincide or are very close.
- The similarity score diminishes as the distance d_{12} increases, converging to zero, which indicates that farther points are less similar.
- The Region of Similarity encompasses the peak's vicinity where the distance is minimal, and the similarity is maximal.
- The Region of Dissimilarity refers to the areas where d_{12} is significant, implying a low similarity score.

This function is critical for SVM's ability to classify data points in a transformed space where a hyperplane can linearly separate them, an invaluable feature for datasets that are not linearly separable in the original feature space.

The 'Region of Similarity' around the peak of the curve indicates that data points close to each other are considered similar (resulting in a high value of the kernel function), which contributes to the SVM model's decision in classifying them into the same category. Conversely, the 'Region of Dissimilarity' suggests that points farther apart are likely to be classified into different categories. The peak at $\mathbf{d}_{12} = 0$,

where the kernel value is 1, represents identical or extremely close points.

This graphical representation aids in understanding the influence of a point on the decision boundary in SVM: points close to the boundary (in the region of similarity) have more influence than those far away (in the region of dissimilarity), leading to a non-linear boundary that is flexible enough to separate classes that are not linearly separable.

7 References

Cinar, I., Koklu, M., & Tasdemir, S. (2020). “Classification of Raisin Grains Using Machine Vision and Artificial Intelligence Methods”.

UC Berkeley and Stanford (CS229) Computer Science Lectures on Support Vector Machines

https://www.niser.ac.in/ smishra/teach/cs460/23cs460/lectures/lec11.pdf

https://towardsdatascience.com/radial-basis-function-rbf-kernel-the-go-to-kernel-acf0d22c798a