Stockholms universitet

Kernel Methods in Credit Risk Prediction - Using Support Vector Machines for Credit Analysis

Rebecka Algervik

Kandidatuppsats i matematisk statistik
Bachelor Thesis in Mathematical Statistics

Kandidatuppsats 2024:11
Matematisk statistik
Augusti 2024

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Stockholm University

Mathematical Statistics
Stockholm University
Bachelor Thesis **2024:11**
http://www.math.su.se

Kernel Methods in Credit Risk Prediction - Using Support Vector Machines for Credit Analysis

Rebecka Algervik*

August 2024

Abstract

The credit risk of housing loans is undergoing dynamic changes in the global market- ing. With economic and market uncertainties, lenders are carefully assessing credit risk associated with housing loans. Factors such as borrower creditworthiness, em- ployment stability, and property market trends play pivotal roles in determining the level of credit risk. As financial institutions navigate these challenges, they employ advanced analytics and risk management strategies to reduce potential defaults and ensure the stability of their housing loan portfolios. This study analyzed the HMEQ dataset, comprising 3,364 observations and 12 in- put variables, to evaluate the performance of four types of Support Vector Machine (SVM) models: linear, polynomial, radial basis function (RBF), and sigmoid kernels. The data preprocessing steps included handling missing values, converting categor- ical variables into dummy variables, performing feature selection, addressing class imbalance by applying class weight balancing, and feature scaling. The dataset was split into training (70%) and testing (30%) sets. Model performance was assessed using metrics such as accuracy, precision, recall, false positive rate, F1 score, specificity, AUC values, and cross-validation. The results indicated that the Polynomial kernel achieved the highest accuracy and cross-validation scores, demonstrating its effectiveness in this context. However, considering the AUC values, the RBF model emerges as the most suitable model for this dataset. Additionally, balancing class weights effectively addressed the issue of data imbalance.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: rebecka.algervik@gmail.com. Supervisor: Taras Bodnar and Jan-Olov Persson.

Acknowledgement

I would like to express my gratitude to my supervisors, Taras Bodnar and Jan Olov Persson, for their valuable feedback during the writing process.

I would also like to express my gratitude to ChatGPT for providing valuable suggestions on the construction of the abstract and introduction, as well as for offering advice to enhance the grammar throughout the entire thesis.

Contents

1 Introduction

The housing loan market is a crucial part of the financial system and significantly influences economic development. However, predicting loan default risk remains a challenging task. Housing loan datasets typically include a large number of variables, both numerical and categorical. These datasets often exhibit strong nonlinearity and high-dimensional characteristics, making data modeling more complex.

One of the most well-known methods in machine learning is the support vector machine (SVM). SVM can capture complex patterns and relationships that may not be evident in traditional linear models. Since their introduction by Vapnik in the 1990s [2], SVMs have been widely studied. Recent studies have shown diverse applications of SVMs: Alshawi (2024) addressed the issue of imbalanced datasets by employing SVM kernels in conjunction with Generative Adversarial Networks (GANs) algorithms to generate synthetic data [1]. Sonmez, Sabanci, and Aydin (2024) explored a convolutional neural network-support vector machine-based approach for identifying wheat hybrids [9]. Han, Chen, and Zhou (2024) proposed a hybrid machine learning model combining a deep residual auto-encoder and SVM for classifying musical genres [3]. Khalifa (2024) integrated Independent Component Analysis (ICA) with SVM to improve feature extraction and diagnosis for heart disease, addressing current literature limitations and advancing prediction models [6].

While there are studies focusing on the application of SVMs to housing loan data, this field still has room for further exploration and development. This paper aims to contribute to this growing body of research by exploring effective handling of nonlinear classification issues, challenges posed by high-dimensional features, and the problem of imbalanced data. Specifically, we analyse a real dataset using the SVM algorithm to classify bank applicants into good or bad credit categories.

In this study, we analyzed the HMEQ dataset, which comprises 3,364 observations and 12 input variables after removing the missing values. The dataset was divided into a training set (70%) and a testing set (30%). The data preprocessing steps included deleting missing values, converting categorical variables into dummy variables, performing feature selection, addressing class imbalance by balancing class weights, and applying feature scaling.

We then trained four types of SVM models using the training data: linear, polynomial, radial basis function, and sigmoid kernels. The performance of each model was

evaluated on the testing data using metrics such as accuracy, precision, recall, false positive rate, F1 score, specificity, AUC values, and cross-validation.

The structure of this paper is as follows: Section 2 introduces the theoretical foundation. Section 3 provides a detailed description of the data preprocessing. Section 4 presents the results of the model evaluation. Section 5 discusses the parameter tuning results. Finally, Section 6 summarizes the research conclusions and proposes future research directions.

2 Theory

2.1 Kernel methods

Kernel methods are powerful, impacting many machine learning techniques, especially when dealing with non-linear relationships and complex data structures.

The fundamental concept behind kernel methods is to elevate data into a high-dimensional feature space, thereby making non-linear relationships linearly separable within that domain.

2.1.1 General introduction to kernel methods

All the theories are taken from [4], [7] and [8], unless stated otherwise.

We consider an instance of a binary classification problem mapped into feature space H, a Hilbert space with higher dimensionality, as shown in Figure 1. Let us assume that the true decision boundary forms an ellipse on the left; our objective is to identify this boundary after the mapping, now in a three-dimensional space on the right.

We perform a mapping of data points x_1 and x_2 from the original space X to a higher-dimensional feature space H, denoted as (z_1, z_2, z_3), through a mapping function ϕ. This mapping is defined as:

$$\phi(x_1,x_2) = (z_1,z_2,z_3) = (x_1^2,\sqrt{2}x_1x_2,x_2^2). \tag{1}$$

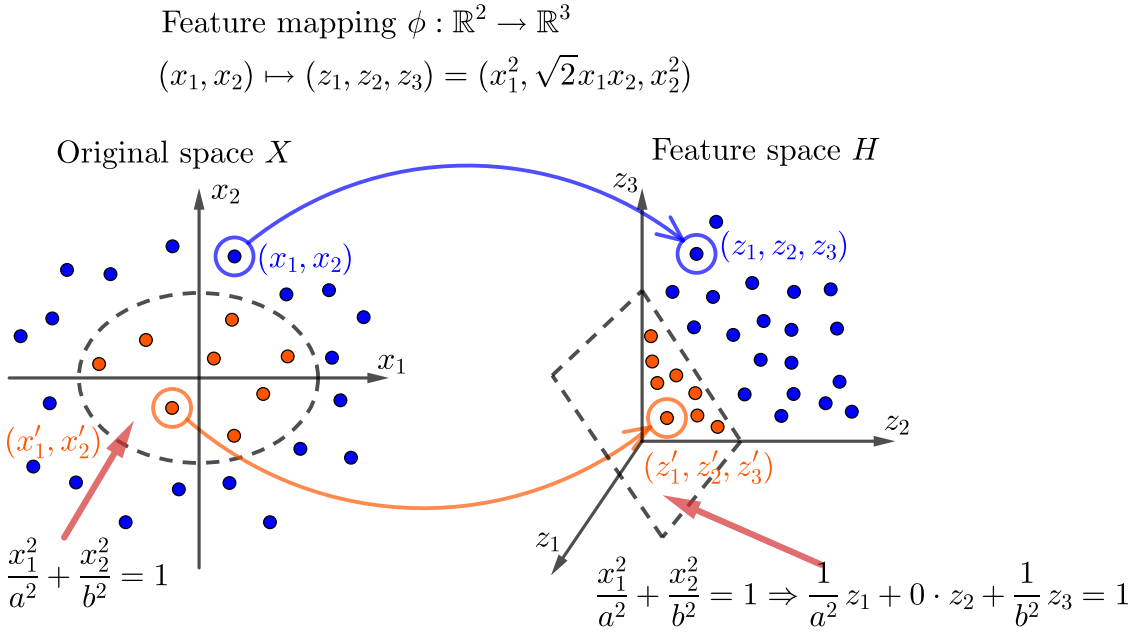

Figure 1:Mapping of Binary Classification to Higher-Dimensional Feature Space

Here, ϕ represents the transformation function that elevates each data point from X to the corresponding point in H. The resulting coordinates in H are expressed in terms of the original coordinates (x_1, x_2). This mapping is essential in the context of kernel methods, where nonlinear relationships in the original space can be linearly modeled in the higher-dimensional feature space.

2.1.2 Kernel function

Let us conduct a mapping of two data points, denoted as $x = (x_1, x_2)$ and $x' = (x_1', x_2')$, originating from the original space X to a higher-dimensional feature space H. In this elevated space, the data points are represented as (z_1, z_2, z_3) and (z_1', z_2', z_3') respectively. The entire mapping process can be elegantly expressed through inner products within the two spaces.

The mapping function, denoted as ϕ, can be precisely defined as follows:

$$
\begin{aligned}
< \phi(x_1,x_2),\phi(x_1',x_2') > &=< (z_1,z_2,z_3),(z_1',z_2',z_3') > \\
&=< (x_1^2, \sqrt{2}x_1x_2,x_2^2),(x_1'^2, \sqrt{2}x_1'x_2',x_2'^2) > \\
&= (< x,x' >^2) \\
&:= \mathcal{K}(x,x').
\end{aligned} \tag{2}
$$

Here the symbol $\mathcal{K}(x,x')$ represents the kernel function, a fundamental mathematical tool extensively employed in machine learning and statistical modelling.

With the aid of the kernel function, we can compute inner products within feature spaces. These inner products provide us with the geometric properties of high-dimensional spaces, including both distances and angles.

Distance in the feature space is:

$$
\begin{aligned}
||\phi(x) - \phi(x')||^2 &= (\phi(x) - \phi(x'))^T(\phi(x) - \phi(x')) \\
&= \mathcal{K}(x,x) - 2\mathcal{K}(x,x') + \mathcal{K}(x',x').
\end{aligned} \tag{3}
$$

And angle in the feature space is :

$$
\begin{aligned}
< \phi(x),\phi(x') >&= \|\phi(x)\| \cdot \|\phi(x')\| \cos\theta \\
\Rightarrow \cos\theta = \frac{< \phi(x),\phi(x') >}{\|\phi(x)\| \cdot \|\phi(x')\|} &= \frac{\mathcal{K}(x,x')}{\sqrt{\mathcal{K}(x,x')}\sqrt{\mathcal{K}(x',x')}}.
\end{aligned} \tag{4}
$$

To make better use of these inner product, we introduce the inner product matrix, also known as the Gram matrix or kernel matrix. The elements of this matrix are the inner products of input samples in the feature space, providing us with a means to measure the similarity and dissimilarity between samples.

Inner Product Matrix is :

$$
\begin{aligned}
\mathcal{K} &= \begin{bmatrix} < \phi(x_1),\phi(x_1) > & \cdots & < \phi(x_1),\phi(x_N) > \\ \vdots & \ddots & \vdots \\ < \phi(x_N),\phi(x_1) > & \cdots & < \phi(x_N),\phi(x_N) > \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{K}(x_1,x_1) & \cdots & \mathcal{K}(x_1,x_N) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(x_N,x_1) & \cdots & \mathcal{K}(x_N,x_N) \end{bmatrix}.
\end{aligned} \tag{5}
$$

2.1.3 Characterisation of Kernels

With the geometric properties of kernel functions in high-dimensional spaces, the theorem titled "Characterisation of Kernels" on page 61 of [7] provides a comprehensive understanding of their behavior and implications in machine learning and mathematical analysis.

Theorem 2.1 *Let X be a vector space. A function*

$$\mathcal{K} : X \times X \to \mathbb{R},$$

which is either continuous or has a finite domain, can be decomposed

$$\mathcal{K}(x,z) =< \phi(x), \phi(z) > \tag{6}$$

into a feature map ϕ into a Hilbert space $\mathcal{H}$ applied to both its arguments followed by the evaluation of the inner product in $\mathcal{H}$, if and only if it satisfies the finitely positive semi-definite property.

The "finitely positive semi-definite property" describes a property of the function $\mathcal{K}$ used to define a kernel function. This property ensures that for any finite dataset $x_1, x_2, \ldots, x_n$, the corresponding kernel matrix is positive semi-definite.

For any set of n data points, the kernel matrix $\mathcal{K}$ satisfies the following condition:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} v_i v_j \mathcal{K}(x_i,x_j) \geq 0.$$

where v_i are arbitrary real numbers, and x_i and x_j represent data points.

This theorem establishes a connection between kernel functions $\mathcal{K}$ and feature maps ϕ. Given a kernel function $\mathcal{K}$ that satisfies the finitley positive semi-definite property, we can find a corresponding feature map ϕ. Conversely, given a feature map ϕ, we can construct a kernel function $\mathcal{K}$ using the inner product in the Hilbert space. Working directly in the input space may be computationally expensive, but by using the kernel trick, we can implicitly work in a higher-dimensional space.

2.1.4 Dual representation

Having established the connection between kernel functions and feature maps, let's explore their dual representation further. The concept of dual representation lies in its capacity to operate implicitly in a high-dimensional feature space, facilitated by kernel methods. This capability allows us to effectively address non-linear relationships and complex data structures without explicit knowledge of the feature space's form or dimensionality.

We describe a simple recognition algorithm, as shown in Figure 2. We assume that our data are mapped into an inner product space $\mathcal{H}$. Samples y belong to binary classification, $y \in \{1, -1\}$. For example, training samples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ are mapped into a feature space $\mathcal{H}$ with points $\{(\phi(x_1), y_1), \ldots, (\phi(x_n), y_n)\}$. Using the inner product, we can measure the distance in the space $\mathcal{H}$.

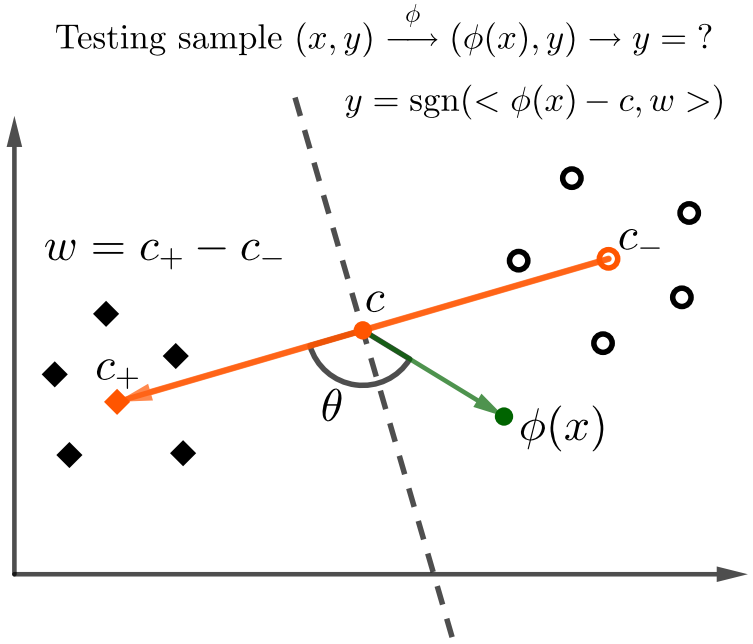

Figure 2: Illustration of a Simple Recognition Algorithm in Feature Space $\mathcal{H}$

Beginning by computing the means of the two classes in the feature space, we obtain:

$$c_+ = \frac{1}{m_+} \sum_{i|y_i=+1} \phi(x_i), \quad c_- = \frac{1}{m_-} \sum_{i|y_i=-1} \phi(x_i). \tag{7}$$

Here m_+ and m_- represent the counts with positive and negative labels, respectively. The difference between two vectors becomes: $w = c_+ - c_-$. To separate the two classes, we draw a line orthogonal to w passing through the midpoint of w. The midpoint is defined as $c = (c_+ + c_-)/2$, accurately portraying the decision boundary hyperplane (depicted as the dotted line).

To evaluate a new testing sample, we calculate the angle between the vector $\phi(x) - c$ and w to check whether it is smaller than $\pi/2$ or not . This lead to

$$y = \text{sgn}(\langle \phi(x) - c, w \rangle). \tag{8}$$

To compute y, we need to express the vectors c_i and w in terms of $x_1, \ldots, x_n$. Substitute equation (9) with inner product to obtain the decision function:

$$\langle \phi(x) - c, w \rangle = \Big(\frac{1}{m_+} \sum_{i|y_i=+1} \mathcal{K}(x, x_i) - \frac{1}{m_-} \sum_{i|y_i=-1} \mathcal{K}(x, x_i) \Big) + b. \tag{9}$$

Here, we have defined b :

$$b = \frac{1}{2}((\|c_-\|)^2 - (\|c_+\|)^2). \tag{10}$$

Similarly, b becomes

$$b = \frac{1}{2}\Big(\frac{1}{m_-^2} \sum_{(i,j)|y_i=y_j=-1} \mathcal{K}(x_i, x_j) - \frac{1}{m_+^2} \sum_{(i,j)|y_i=y_j=+1} \mathcal{K}(x_i, x_j)\Big). \tag{11}$$

To facilitate our analysis, we introduce Wahba's Representer Theorem (see page 90 of [8]). It loosely states that the solutions to certain risk minimisation problems, which involve an empirical risk term and a quadratic regulariser, can be expressed as expansions in terms of the training samples:

$$w = \sum_{i=1}^{N} \alpha_i \phi(x_i). \tag{12}$$

With the help of equation (12), the normal vectors w of decision hyperplanes can be written as the general linear combinations of training samples.

$$w = \frac{1}{m_+} \sum_{i|y_i=+1} \phi(x_i) - \frac{1}{m_-} \sum_{i|y_i=-1} \phi(x_i) = \sum_{i=1}^{N} \alpha_i \phi(x_i). \tag{13}$$

We consider the following linear function in the feature space $\mathcal{H}$,

$$f(x) = W^T \phi(x) + b$$

Let $\langle \phi(x) - \mathbf{c}, \mathbf{w} \rangle = f(x)$, then the output of y becomes:

$$\begin{aligned} y &= sgn(\langle \phi(x) - \mathbf{c}, \mathbf{w} \rangle) \\ &= sgn(f(x)) \\ &= sgn(w^T \phi(x) + b) \\ &= sgn(\Big(\sum_{i=1}^{N} \alpha_i \phi(x_i) \Big)^T \phi(x) + b) \\ &= sgn(\Big(\sum_{i=1}^{N} \alpha_i \phi(x_i)^T \phi(x) \Big) + b) \\ &= sgn(\sum_{i=1}^{N} \alpha_i \mathcal{K}(x_i, x) + b) \end{aligned} \tag{14}$$

where b is the same in (11).

Here the α_i can be viewed as a dual form of the normal vector for the hyperplane. Finding w is equivalent to finding α_i. The length of w corresponds to the dimension of the feature space $\mathcal{H}$, while the number of α_i corresponds to the number of training samples.

Through the dual representation, we can utilize kernel methods to implicitly map the raw data to a high-dimensional feature space and perform computations in this space. This enables us to handle data more effectively, achieving more precise classification or recognition.

2.2 Support vector machine (SVM)

SVM is a supervised learning algorithm designed to address both classification and regression problems. SVM aims to find an optimal decision boundary that separates different classes. One of the notable strengths of SVM lies in its ability to handle high-dimensional data effectively.

Traditional SVM employs a linear decision boundary, but to handle non-linear patterns, kernel methods come into play. Kernel methods play a pivotal role in SVM by mapping input data into a higher-dimensional feature space and capture non-linear decision boundaries.

SVM can be categorized into two main types: Hard-margin SVM and Soft-margin SVM. Hard-margin SVM aims for perfect separation of classes with no misclassifications, while soft-margin SVM allows for some misclassifications to achieve a more robust and generalized solution.

2.2.1 Hard-Margin SVM

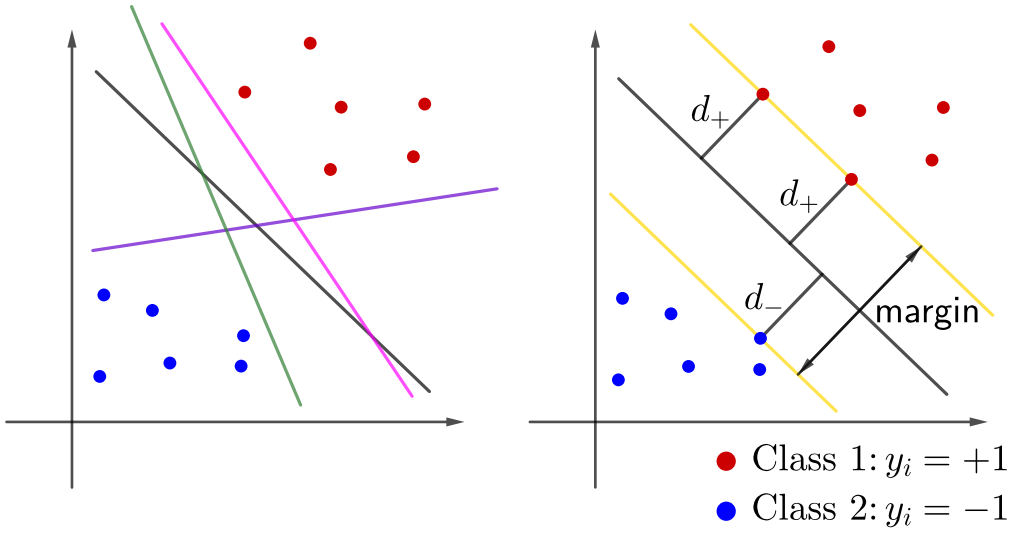

Figure 3: Illustration of Hard-Margin SVM with Linear Decision Boundary

Let us consider a linearly separable two-classification problem, with training data

$(x_i, y_i), i \in \{1, \ldots, n\}, x_i \in \mathbb{R}^n$, and y_i belongs to two-class with labels as shown in Figure 3 on the left:

$$y_i = \begin{cases} +1, & \text{if } x_i \in \text{class 1} \\ -1, & \text{if } x_i \in \text{class 2.} \end{cases} \tag{15}$$

In geometry, the Decision Hyperplane is defined as $w^T x + b = 0$, where w represents the normal vector orthogonal to the hyperplane, and b represents the intercept.

When the normal vector w moves towards class 1, a Support Vector Hyperplane is defined as $w^T x + b = 1$ by the first encountered point belonging to class 1. Conversely, when the normal vector w is oriented towards class 2, another Support Vector hyperplane is defines as $w^T x + b = -1$ by the first encountered point belonging to class 2.

With $w^T x_i + b \geq 1$ if $y_i = +1$, and $w^T x_i + b \leq 1$ if $y_i = -1$, we combine them into one constraint : $y_i(w^T x_i + b) \geq 1$.

With the aim of measuring the distance between two hyperplanes, specifically from both $w^T x + b = 1$ and $w^T x + b = -1$ to $w^T x + b = 0$,we obtain d_+ and d_- respectively:

$$d_+ = \frac{1}{||w||}, d_- = \frac{1}{||w||}.$$

.

Our goal is to finding an optimal decision boundary that separates two labels, which could be achieved by maximising the margin (see Figure 3).

$$\text{margin} = (d_+) + (d_-) = \frac{1}{||w||} + \frac{1}{||w||} = \frac{2}{||w||} = \frac{2}{\sqrt{w^T w}}$$

In order to maximize the margin, we minimize $\sqrt{w^T w}$.

The problem of maximizing the margin $(d_+) + (d_-)$ can be formulated as follows:

$$\begin{aligned} \text{Minimize} \quad & \Phi(w) = \frac{1}{2} w^T w, \\ \text{subject to} \quad & y_i(w^T x_i) + b \geq 1 \quad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{16}$$

This formulation represents the optimisation problem for the Hard-Margin SVM, where $\Phi(w)$ is the objective function to be minimized, and the constraints ensure

that each training point is correctly classified with a margin of at least 1. The solution to this problem provides the optimal hyperplane for linearly separating the two classes.

This convex minimization problem can be solved effectively using the Lagrange multipliers method:

$$\begin{aligned} \text{Minimize} \quad & L(w,b,\alpha) = \frac{1}{2}w^T w - \sum_{i=1}^{l} \alpha_i [y_i(w^T x_i + b) - 1], \\ \text{subject to} \quad & \alpha_i \geq 0 \quad \forall i \in \{1, \ldots, n\}, \end{aligned} \tag{17}$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_l)^T$.

By computing partial derivatives of the function L from (17) with respect to w and b, and setting them equal to 0, we obtain

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i, \tag{18}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0. \tag{19}$$

Suppose ϕ represents a feature mapping associated with a kernel function $\mathcal{K}(x,z)$, in this case, we obtain the following in the feature space

$$\begin{aligned} \text{Minimize} \quad & L(w,b,\alpha) = \frac{1}{2}w^T w - \sum_{i=1}^{N} \alpha_i [y_i(w^T \phi(x_i) + b) - 1], \\ \text{subject to} \quad & \alpha_i \geq 0 \quad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{20}$$

And

$$w = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i), \tag{21}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0. \tag{22}$$

Substituting equations (21) and (22) into the function (17), we obtain

$$w^T w = \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j y_i y_j \mathcal{K}(x_i,x_j), \tag{23}$$

and

$$w^T \phi(x_i) = \sum_{j=1}^{N} a_j y_j \mathcal{K}(x_i,x_j). \tag{24}$$

With results (23) and (24) substituted into the function w, we obtain

$$w = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathcal{K}(x_i,x_j). \tag{25}$$

Under the following two constraints, we could find the maximisation of w:

$$\begin{aligned} \text{Minimize} \quad & w = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathcal{K}(x_i,x_j), \\ \text{subject to} \quad & (1) \quad \sum_{i=1}^{N} \alpha_i y_i = 0, \\ & (2) \quad \alpha_i \geq 0, \quad \forall i \in \{1,\ldots,n\}. \end{aligned} \tag{26}$$

If α^*, w^*, b^* are the optimal solutions, then

$$\alpha_i^* (y_i (w^{*T} \phi(x_i) + b^*) - 1) = 0, \text{with} \quad i = 1,\ldots,n. \tag{27}$$

By the Karush-Kuhn-Tucker (KKT) conditions on the page 199 of [7] , we obtain that if $\alpha_i^* > 0$, then

$$y_i (w^{*T} \phi(x_i) + b^*) = 1,$$

then $\phi(x_i)$ is on the margin and is one of the support vectors. Otherwise, when $\alpha_i^* = 0$, all the corresponding data lies outside of margin.

Since most of α_i^* are zero, the decision boundary is only determined by the support vectors, then

$$w = \sum_{i=1}^{N} \alpha_i^* y_i \phi(x_i) = \sum_{\alpha_i^* \neq 0} \alpha_i^* y_i \phi(x_i). \tag{28}$$

Again, using the KKT conditions, we get

$$\alpha_t^*(y_t(w^{*T}\phi(x_t)+b^*)-1)=0, \quad \text{when} \quad \alpha_t^*>0. \tag{29}$$

Recall that $y_t = \pm 1$, so by the equation (29), $\phi(x_t)$ is on the margin, that is $w^{*T}\phi(x_t)+b^*=\pm 1$ and we get

$$w^{*T}\phi(x_t)+b^*=y_t. \tag{30}$$

Hence, the intercept b can be computed with

$$b^*=y_t-\sum_{i=1}^{N}\alpha_i^* y_i \mathcal{K}(x_i,x_t). \tag{31}$$

To classify a new testing sample, we can plug in new data with

$$w^T\phi(x_{new})+b=\sum_{i=1}^{N}\alpha_i y_i \mathcal{K}(x_i,x_{new})+b. \tag{32}$$

If the result is positive, classify x_{new} as belonging to class 1; otherwise, classify it as belonging to class 2. That is

$$y_{new}=\text{sgn}(\sum_{i=1}^{N}\alpha_i y_i \mathcal{K}(x_i,x_{new})+b). \tag{33}$$

2.2.2 Soft-Margin SVM

If there is overlap between two classes in a two-classification problem or if linear separability is not achievable as shown in Figure 4, we can allow for some degree of tolerance towards classification errors, meaning that certain points may fall within the margin. This concept is known as soft-margin SVM .

The problem of maximising the margin $(d_+)+(d_-)$ can be formulated as follows by introducing slack variables ξ_i:

$$\begin{aligned} &\text{Minimize} \qquad \Phi(w,\xi)=\frac{1}{2}w^Tw+C\sum_{i=1}^{N}\xi_i, \quad C>0, \\ &\text{subject to} \qquad \xi_i\geq 0, \quad y_i(w^Tx_i)+b\geq 1-\xi_i \quad \forall i\in\{1,\ldots,n\}. \end{aligned} \tag{34}$$

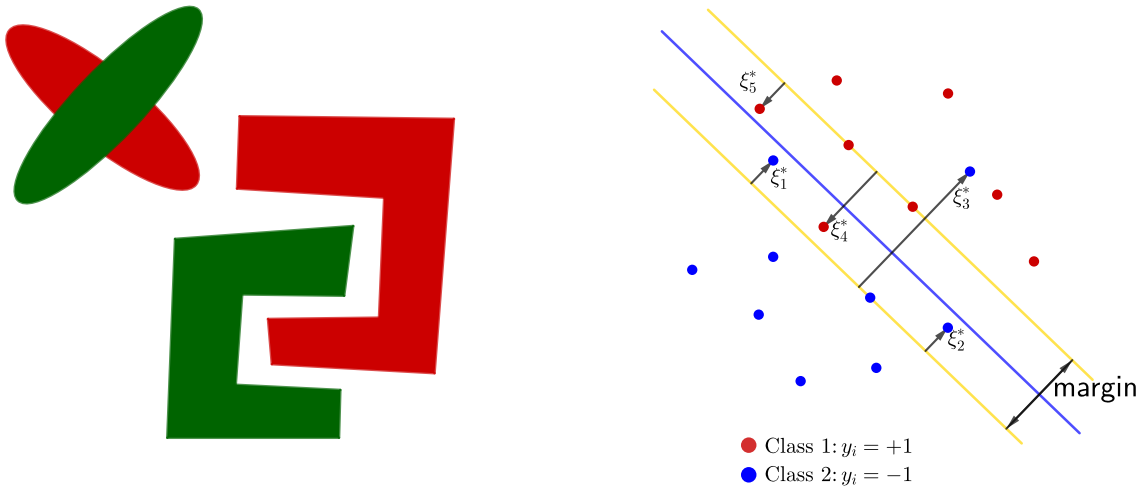

Figure 4:Illustration of Soft-Margin SVM with Tolerance for Classification Errors

The parameter C is a regularisation parameter that controls the trade-off between achieving a low training error and a low testing error. A smaller C value results in a wider margin, allowing for more training points to be misclassified. A larger C value enforces a stricter margin, imposing more substantial penalties for misclassification.

This convex minimisation problem can be also solved effectively using the Lagrange multipliers method:

Minimize

$$L(w,b,\xi,\alpha,\beta) = \frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(w^T x_i + b) - 1 + \xi_i] - \sum_{i=1}^{N}\beta_i\xi_i, \tag{35}$$

subject to $\alpha_i \geq 0, \beta \geq 0, \forall i \in \{1,\ldots,n\}$.

By computing partial derivatives of the function L from (35) with respect to w,b and ξ, and setting them equal to 0, we obtain

$$w = \sum_{i=1}^{N}\alpha_i y_i x_i, \tag{36}$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0, \tag{37}$$

$$\text{and} \quad C - \alpha_i - \beta_i = 0 \Rightarrow \begin{cases} \alpha_i = C - \beta_i \\ \beta_i = C - \alpha_i \end{cases} \Rightarrow 0 \leq \alpha_i, \beta_i \leq C. \tag{38}$$

Suppose ϕ is a feature mapping associated with a kernel function $\mathcal{K}(x,z)$, we replace x_i by $\phi(x_i)$ in (35), then we obtain the following in the feature space

Minimize

$$L(w,b,\xi,\alpha,\beta) = \frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(w^T\phi(x_i)+b) - 1 + \xi_i] - \sum_{i=1}^{N}\beta_i\xi_i,$$

subject to $\quad \alpha_i \geq 0, \beta \geq 0, \forall i \in \{1,\ldots,n\}.$

(39)

And

$$w = \sum_{i=1}^{N}\alpha_i y_i \phi(x_i) = X^T Y \alpha, \tag{40}$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0 = y^T\alpha, \tag{41}$$

$$C - \alpha_i - \beta_i = 0 \Rightarrow \begin{cases} \alpha_i = C - \beta_i \\ \beta_i = C - \alpha_i \end{cases} \Rightarrow 0 \leq \alpha_i, \beta_i \leq C, \tag{42}$$

where

$$\begin{aligned} X^T &= [\phi(x_1), \ldots, \phi(x_n)], \\ Y &= \text{diag}(y_1, \ldots, y_n), \\ y^T &= [y_1, \ldots, y_n], \\ \alpha^T &= [\alpha_1, \ldots, \alpha_n]. \end{aligned}$$

Substituting equations (40), (41), and (42) into the function (35), we obtain

$$w^T w = \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathcal{K}(x_i,x_j), \tag{43}$$

and

$$w^T\phi(x_i) = \sum_{j=1}^{N} a_j y_j \mathcal{K}(x_i,x_j). \tag{44}$$

With the results of (43) and (44) substitued into the function w from (40), we obtain

$$w = \alpha^T 1_{n\times 1} - \frac{1}{2}\alpha^T Y K Y \alpha, \tag{45}$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$, $1_{n\times 1} = (1, \ldots, 1)^T \in \mathbb{R}^n$.

Under the following two constraints, we could find the maximization of w

$$\begin{aligned} \text{Minimize} \quad & w = \alpha^T 1_{n\times 1} - \frac{1}{2}\alpha^T Y K Y \alpha, \\ \text{subject to} \quad & (1) \quad \alpha^T y = 0, \\ & (2) \quad 0 \le \alpha_i \le C, \quad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{46}$$

If $\alpha^*, w^*, b^*, \xi^*$ are the optimal solutions, then

$$\begin{cases} \alpha_i^*(y_i(w^{*T}\phi(x_i) + b^*) - 1 + \xi_i^*) = 0, \\ \beta_i^* \xi_i^* = 0, \end{cases} \qquad \forall i \in \{1, \ldots, n\}. \tag{47}$$

We use the KKT conditions and the first equation of (47), then we get the following results

$$\begin{cases} \alpha_i^* = 0 \Rightarrow y_i(w^{*T}\phi(x_i) + b^*) \ge 1, \\ 0 < \alpha_i^* < C \Rightarrow y_i(w^{*T}\phi(x_i) + b^*) = 1, \\ \alpha_i^* = C \Rightarrow y_i(w^{*T}\phi(x_i) + b^*) \le 1. \end{cases} \tag{48}$$

If $0 < \alpha_i^* < C$, and $y_i(w^{*T}\phi(x_i) + b^*) = 1$, then $\phi(x_i)$ is positioned on the margin and qualifies as one of the support vectors. When $\alpha_i^* = 0$, all the corresponding data lie outside of margin. If $\alpha_i^* = C$, all the data points lie either inside or on the margin (see Figure 4).

Again, using the KKT conditions, we get

$$\alpha_t^*(y_t(w^{*T}\phi(x_t) + b^*) - 1) = 0 \quad \text{when} \quad 0 < \alpha_t^* < C.$$

So $\phi(x_t)$ is on the margin $w^{*T}\phi(x_t) + b^* = \pm 1$, and

$$w^{*T}\phi(x_t) + b^* = y_t. \tag{49}$$

Hence, the intercept b can be computed with (44) and (49), then we get

$$b^* = y_t - \sum_{i=1}^{N} \alpha_i^* y_i \mathcal{K}(x_i, x_t). \tag{50}$$

To classify a new testing sample, we can plug in new data with

$$w^T\phi(x_{new}) + b = \sum_{i=1}^{N} \alpha_i y_i \mathcal{K}(x_i, x_{new}) + b. \tag{51}$$

If the result is positive, classify x_{new} as belonging to class 1; otherwise, classify it as belonging to class 2. That is

$$y_{new} = \text{sgn}(\sum_{i=1}^{N} \alpha_i y_i \mathcal{K}(x_i, x_{new}) + b). \tag{52}$$

The optimisation problem for the soft margin SVM in equation (34) can be written as

$$\min_{w,b} \sum_{i=1}^{N} [1 - y_i(w^T x_i + b)]_+ + \frac{\lambda}{2}||w||^2. \tag{53}$$

The expression $\sum_{i=1}^{N}[1 - y_i(w^T x_i + b)]_+$, where the subscript + denotes the positive part, is known as the hinge loss.

2.2.3 Kernel functions

The choice of kernel functions for our SVM model is crucial as it directly affects its ability to capture complex data patterns. The four kernel functions introduced in Table 1 form the basis of our SVM model, enabling us to investigate different approaches for separating and classifying data points effectively. In the following sections, we'll analyze the performance of these kernel functions and their impact on our SVM model's effectiveness.

Table 1: SVM Models with Four Kernels

SVM model	Solving problem	Kernel function
"linear"	Linear	$\mathcal{K}(x,z) = x^T \cdot z = Xz$
"poly"	Partially linear	$\mathcal{K}(x,z) = (\gamma(x \cdot z) + r)^d$
"rbf"	Partially nonlinear	$\mathcal{K}(x,z) = \exp(-\gamma\|\|x - z\|\|^2), \gamma > 0$
"sigmoid"	Nonlinear	$\mathcal{K}(x,z) = \tanh(\gamma(x \cdot z) + r), \gamma > 0$

Here, "poly" refers to polynomial functions, "rbf" refers to radial basis functions, and "sigmoid" refers to the sigmoid function.

2.3 Model evaluation

To assess the effectiveness of the SVM model, we introduce specific indices for evaluation. The model evaluation metrics come from [10].

2.3.1 Confusion matrix

A confusion matrix is a specialized form of a contingency table, with two dimensions labeled as "actual" and "predicted". It includes identical sets of "classes" in both dimensions, with each combination representing a variable in the contingency table.

The binary confusion matrix template incorporates four key result categories: true positives, false negatives, false positives, and true negatives, in addition to positive and negative classifications. These outcomes can be organised into a 2×2 confusion matrix as follows:

Table 2: A confusion matrix

	Predicted condition	
Total population = P+N	Positive	Negative
Positive (P)	True positive	False negative
Negative (N)	False positive	True negative

From the confusion matrix, we will describe some metrics that offer a comprehensive evaluation of a classification model's performance.

1. Accuracy: the overall correctness of the model, measuring the ratio of correctly predicted instances to the total instances

$$\text{ACC} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}}.$$

2. Precision(Positive Predictive Value): the proportion of correctly predicted positive

observations out of the total predicted positives

$$\text{PPV} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}.$$

3. Recall (Sensitivity, True Positive Rate): recall measures the ability of the model to capture all the positive instances, indicating the proportion of correctly predicted positives out of the total actual positives

$$\text{TPR} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}.$$

4. False Positive Rate: it represents the proportion of negative instances that were incorrectly predicted as positive out of the total actual negatives

$$\text{FPR} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}.$$

5. F1 Score: the F1 score is the harmonic mean of precision and recall, providing a balanced measure that considers both false positives and false negatives

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

6. Specificity (True Negative Rate): specificity measures the ability of the model to correctly identify negative instances out of the total actual negatives

$$\text{TNR} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}.$$

2.3.2 AUC and ROC

ROC and AUC provide a valuable means to assess and compare the discriminatory power of classification models, particularly in binary classification tasks.

ROC (Receiver Operating Characteristic): ROC is a visual tool that illustrates how well a classification model performs across different discrimination thresholds. It represents the trade-off between the true positive rate (recall) and the false positive rate (1 - specificity) as we adjust the threshold for classifying instances.

In our approach, we first measure the distance of each testing sample to the decision boundary of our classification model. This distance serves as an indicator of how confidently a sample is classified. By measuring the distance between samples and each class, we can better understand the model's predictions.

To convert these distances into meaningful probabilities, we employ Platt scaling and the sigmoid function. Platt scaling transforms the raw SVM decision function values into probability estimates, providing a probabilistic interpretation of the classification.

If a sample is far from the decision boundary, it is assigned a higher probability of belonging to a particular class, and vice versa. The probability $P(y = 1|x)$ of a sample point belonging to class 1 is calculated using the formula:

$$P(y = 1|x) = \frac{1}{1 + \exp(Af(x) + B)}$$

where:

- $P(y = 1|x)$: The probability of a sample point belonging to class 1
- $f(x)$: The original SVM output (the decision function value)
- A and B: Parameters obtained through cross-validation fitting

By using this method, we ensure that the distances to the decision boundary are effectively translated into probabilistic terms, enhancing the interpretability and reliability of our classification model.

By setting various probability thresholds, we create confusion matrices for each sample. These matrices allow us to compute the true positive rate (TPR) and false positive rate (FPR) at each threshold setting. TPR, also known as recall, measures the proportion of positive instances that are correctly classified, while FPR quantifies the proportion of negative instances that are incorrectly classified as positive.

By plotting TPR against FPR across different threshold settings, we can generate the ROC curve, providing valuable insights into the classification model's performance and its ability to distinguish between the positive and negative classes.

AUC (Area Under the Curve): AUC is the area under the ROC curve. It quantifies the overall performance of a classification model by measuring the area between the ROC curve and the baseline. AUC values range from 0 to 1, where a higher AUC indicates better discrimination ability. An AUC of 0.5 suggests a classifier performing no better than random chance, while an AUC of 1.0 represents a perfect classifier.

2.3.3 Cross-validation

Cross-validation is a statistical technique employed to assess the performance of machine learning models by segmenting the dataset into subsets for both training and validation purposes. It entails iteratively employing different data subsets for training and validation to accurately measure the model's generalization ability while mitigating bias stemming from data partitioning.

In our case, after dividing the dataset into a training set (70%) and a testing set (30%), we use the training set to perform cross-validation. With k-Fold Cross-Validation, our training set is divided into k equally sized subsets. During each iteration, $k-1$ subsets are utilized for training, while the remaining subset serves as the validation set. This process is reiterated k times, with a different validation set chosen each time. Ultimately, the model's performance is evaluated by averaging the results from the k validation iterations.

3 Data

3.1 Data description

The data we will analyse comes from a bank in the USA (downloaded from `http://www.creditriskanalytics.net/datasets-private2.html`), the name of the bank was not disclosed for privacy reasons. It contains information on 5960 equity loans and 12 features. A home equity loan is a financial arrangement in which the borrower utilises the equity in their home as collateral for the loan.

All the variables are presented in the Appendix Table 1. We have 2 categorical variables, "Reason" and "Job", 5 continuous variables, "Loan", "Mortdue", "Value",

"Clage", and "Debtinc", and 5 discrete variables, "Yoj", "Derog", "Delinq", "Ninq", and "Clno" in the dataset.

3.2 Data preprocessing

3.2.1 Missing value

The SVM algorithm involves distance measurement, and the support vectors determine the position of the hyperplane. If there are missing values in the support vectors, it will affect the position of the hyperplane, potentially resulting in underfitting or overfitting of the trained model. Therefore, SVM requires careful preprocessing to handle missing values. Common strategies include deletion and imputation, such as filling in missing values based on models using regression or neural networks.

Table 3: The number of missing values

Name	Number
LOAN	0
MORTDUE	518
VALUE	112
REASON	252
JOB	279
YOJ	515
DEROG	708
DELINQ	580
CLAGE	308
NINQ	510
CLNO	222
DEBTINC	1267

To illustrate the importance of handling missing values, Table 3 shows the number of missing values. Since the number of missing values for each observation is not significant, we chose to drop all observations with missing values. After removing all the missing values, we retained 3364 observations and 12 variables.

Next, we examined the data distribution after deleting the missing values. Figure 5 shows the data distribution using the first two variables after deleting the

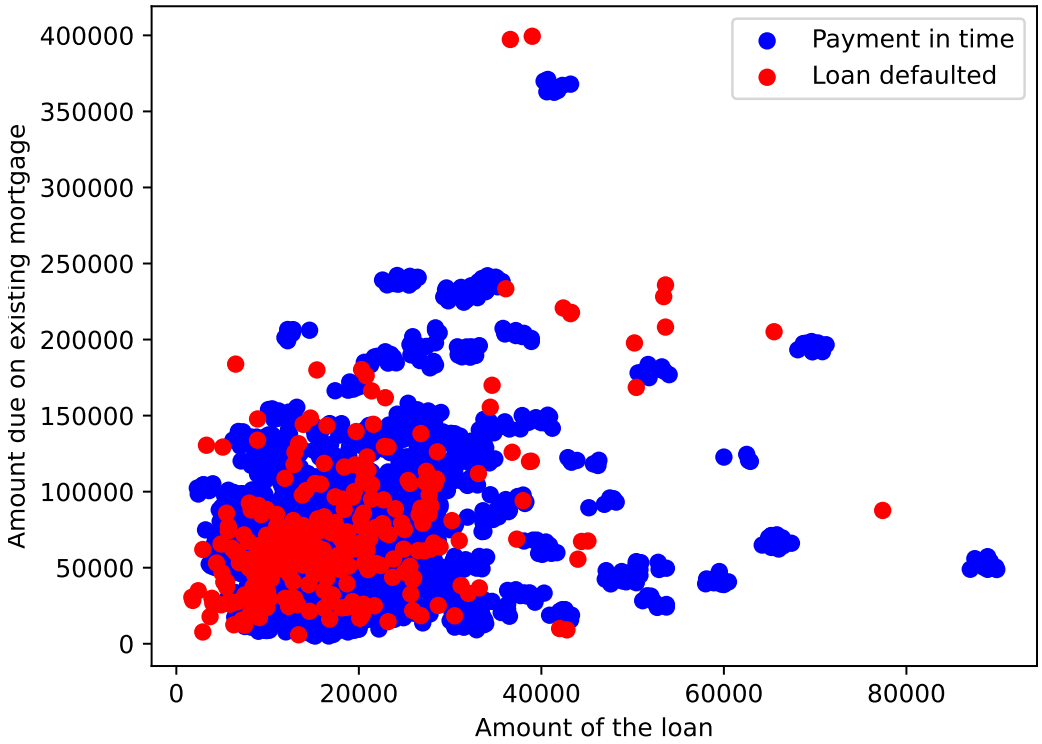

Figure 5: The distribution of the data with first and second variables.

missing values, revealing a significant overlap between the two classes. We selected these two variables because a two-dimensional plane can clearly show relationships and distributions among data points, making it easier for people to understand the structure and potential patterns in the data. Compared to high-dimensional data, two-dimensional data is simpler to visualize and interpret. Although the dataset contains multiple variables, it's common to initially choose a few features for preliminary exploration, such as the amount of the loan and the amount due on existing mortgages in our dataset. These two features are considered important in the banking data domain for initial exploration.

However, this initial exploration revealed a challenge. It is evident that a single linear boundary is insufficient for effectively separating the classes in this visualization. Consequently, a linear kernel model may not achieve optimal performance on this

dataset. Furthermore, the dataset exhibits a high degree of imbalance; specifically, the proportions of good and bad customers are unequal, with a ratio of 0.91 to 0.09, respectively. Dealing with imbalanced data poses a challenge for SVM, as it may bias the model towards the majority class and hinder its ability to accurately classify instances from the minority class. We will address this issue later.

3.2.2 Dummy variable

In order to make our SVM model suitable, we convert all categorical variables, such as "Job" and "Reason", into dummy variables. This ensures that categorical data is correctly represented without implying any ordinal relationship.

After converting the categorical variables into dummy variables, and including our 10 numerical variables, the total number of variables in the dataset becomes 18.

3.2.3 Feature selection

Feature selection plays a crucial role in machine learning by improving model performance and interpretability. By selecting the most relevant features, we can reduce overfitting, decrease training times, and enhance the model's ability to generalize to unseen data.

Before proceeding with feature selection, it is essential to partition the dataset into training and testing sets. We use a 70:30 ratio for this split. This ratio strikes a balance by providing enough data for the model to learn effectively (70% for training) while ensuring a sufficient amount of unseen data for reliable performance evaluation (30% for testing). This strategy helps mitigate the risk of information leakage from the training set to the testing set, thereby reducing the potential for model overfitting.

In the feature selection process for numerical variables, we employ the variance threshold technique as a critical step in our data preprocessing. This technique involves removing features with low variance, which indicates limited variation within the dataset. Specifically, we calculate the variance for each feature and eliminate those whose variance falls below a predefined threshold. The rationale behind this technique is that features with low variance contribute little to the dataset's diversity and are less likely to be useful for the model. For example, a feature with a vari-

ance close to zero indicates that nearly all its values are the same, offering minimal contribution to the model.

While SVM classification tasks are less impacted by multicollinearity due to their use of kernel functions to map data to higher-dimensional spaces, strong linear relationships between features in small datasets can still affect SVM performance. Such relationships may cause the model to fit noise rather than capturing true patterns. To address this, we use feature selection or dimensionality reduction techniques. We utilized the Variance Inflation Factor (VIF) to identify multicollinearity within the dataset. Notably, both MORTDUE and VALUE exhibited VIF values exceeding 20, as shown in Figure 6, indicating a strong presence of multicollinearity between them. As anticipated, a higher housing value is associated with a higher mortgage amount. Multicollinearity arises when two or more variables in a model are highly correlated, leading to redundancy in the information they provide. To resolve this issue, we will introduce feature scaling in the subsequent steps.

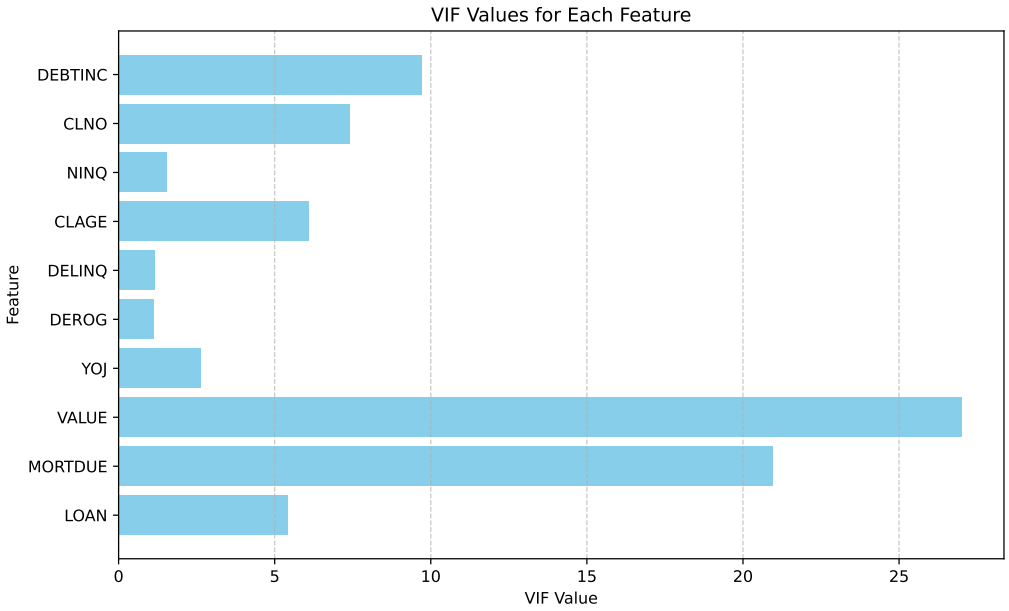

Figure 6: Variance Inflation Factor (VIF) Values for Features

For categorical variables, our feature selection strategy involves applying the chi-squared statistic. This statistical test assesses the independence between categor-

ical variables by comparing the observed frequency distribution with the expected frequency distribution under the assumption of independence. Higher chi-squared values indicate stronger associations between variables, suggesting their potential relevance as features in the model.

To implement this strategy, we compute the chi-squared statistic for each categorical variable, such as "Job" and "Reason," using dummy variables in relation to the target variable "Bad". We select the ones with p-values below a specified threshold (e.g., 0.05). This ensures that we are choosing features that have a statistically significant association with the target variable. In this case, we select the two features with p-values below 0.05: "JOB_ProfExe" and "JOB_Sales". These features are statistically significant in their association with the target variable "Bad".

After testing various combinations of numerical and categorical variables—such as 7 numerical and 2 categorical, 8 numerical and 2 categorical, and 10 numerical and 2 categorical—we have decided to proceed with a combination of 10 numerical and 2 categorical variables. This combination was chosen because the SVM models performed the best with it. With a total of 12 features, we will use them to evaluate our SVM models.

3.2.4 Feature scaling

When examining our dataset, we discovered that our numerical variables exhibit varying ranges. For instance, one feature ranges from 0 to 10, while another spans from 1,000 to 90,000. Such differences can potentially lead to poor performance of machine learning algorithms. Appendix Table 2 provides the descriptive statistics for all quantitative variables in our dataset.

To address this issue, we standardized our numerical variables. This process involved performing z-score standardization to ensure comparability and establish a consistent scale across all variables. Specifically, each variable was transformed to have a mean of 0 and a standard deviation of 1. While this step helped in making the variables more comparable, it also contributed to reducing potential multicollinearity issues among continuous variables, as illustrated in Figure 7.

Standardizing the data resolved multicollinearity among continuous variables by ensuring that each variable contributes equally to the model's learning process, regardless of its initial scale. This transformation removed scale differences between

variables, allowing the algorithm to focus on the relative importance of each variable rather than its absolute magnitude.

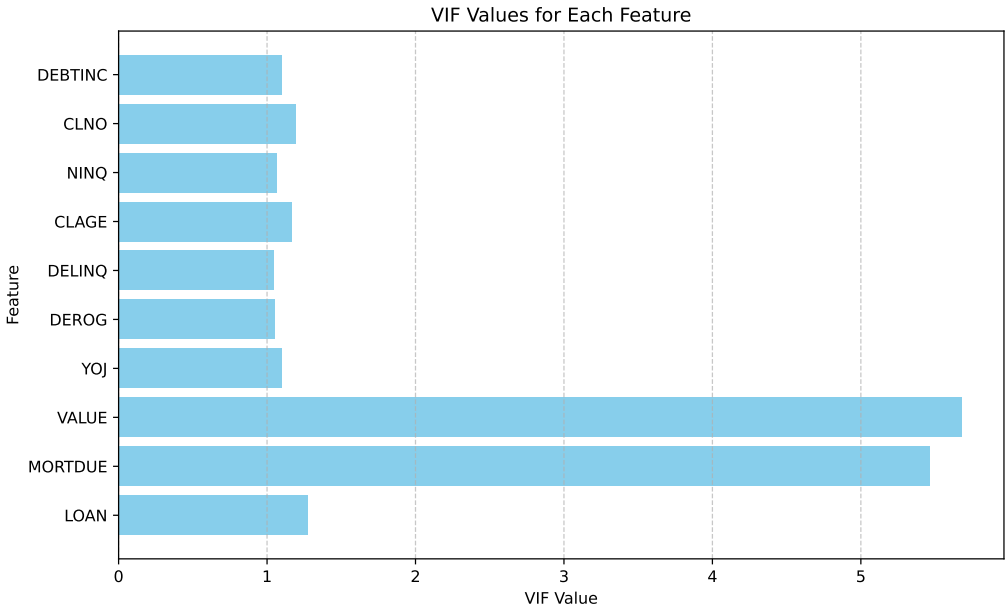

Figure 7: Effect of Feature Scaling on Variance Inflation Factor (VIF) Values

3.3 Model Optimization with Balanced Class Weights

Handling imbalanced datasets can lead to model bias towards the majority class, which can adversely affect the performance and accuracy of the model. To address this issue, we utilized a class weight adjustment strategy known as the "balanced" mode.

The "Balanced" Class Weight Mode is an adaptive weight allocation strategy that adjusts the weights of classes based on the number of samples in each class within the dataset. This mode allows the model to place more emphasis on the minority class, thereby improving performance on imbalanced datasets.

In our case, implementing the "balanced" mode involves specifying the parameter $class\_weight = "balanced"$ when configuring the machine learning model. This

parameter automatically adjusts the class weights based on the distribution of classes in the training data. Specifically, the class weight is given by

$$w_i = \frac{n\_samples}{n\_classes * n\_i}$$

where:

- w_i: Weight of class i
- $n\_samples$: Total number of samples in the dataset
- $n\_classes$: Number of classes
- n_i: Number of samples in class i

In SVM, adjusting class weights modifies the model's sensitivity to different classes by incorporating these weights during the optimization process. This helps prevent the model from favoring the majority class excessively, thereby improving overall performance.

By utilizing the "balanced" class weight mode, the performance of the SVM on imbalanced datasets is enhanced, ensuring fair treatment for all classes.

4 Modelling and Results

After completing data preprocessing, which included deleting missing values, converting categorical variables into dummy variables, feature selection, addressing class imbalance by balancing class weights, and feature scaling, we utilized the scikit-learn package in Python to fit Support Vector Machine (SVM) models (see [5]). These models were trained with four different kernels: the linear kernel, the polynomial kernel, the radial basis function (RBF) kernel, and the sigmoid kernel.

4.1 Confusion matrix

Table 4 displays the confusion matrix for SVM models employing four different kernels on the testing data, which consists of 1,010 observations and 12 variables. This

visualization presents the evaluation metrics for each SVM model, showing their ability to capture True Positives (TP), False Negatives (FN), False Positives (FP), and True Negatives (TN) within the dataset.

Table 4: Confusion Matrices for SVM Models with Four Different Kernels

Linear kernel

	Predicted 1	Predicted 0
Actual 1	44	39
Actual 0	160	767

Polynomial kernel

	Predicted 1	Predicted 0
Actual 1	36	47
Actual 0	56	871

RBF kernel

	Predicted 1	Predicted 0
Actual 1	53	30
Actual 0	92	835

Sigmoid kernel

	Predicted 1	Predicted 0
Actual 1	50	33
Actual 0	379	548

Table 5 presents the performance metrics of four SVM models, including accuracy, precision, recall, false positive rate (FPR), F1 score, and specificity. Among these, accuracy, FPR, and specificity are computed using the formulas from Section 2.3.1. However, precision, recall, and F1 score are calculated with average='weighted', meaning they are averaged and weighted by the number of true instances for each label (support). More details can be found in the Appendix: Derivation of Performance Metrics for the RBF Kernel.

Table 5: The evaluations values from SVM models for testing set

SVM model	ACC	PPV	TPR	FPR	F1-Score	TNR
"Linear"	0.8030	0.8911	0.8030	0.1726	0.8376	0.8274
"Poly"	**0.8980**	0.9029	**0.8980**	**0.0604**	**0.9004**	**0.9396**
"Rbf"	0.8792	**0.9160**	0.8792	0.0992	0.8935	0.9008
"Sigmoid"	0.5920	0.8752	0.5920	0.4088	0.6831	0.5912

The models achieved accuracies of 0.8030, 0.8980, 0.8792, and 0.5920. Notably, the Polynomial and RBF models exhibit commendable accuracy, while the Sigmoid model falls below 0.6. Among these, the SVM with the Polynomial kernel emerges as the best-performing model when considering accuracy, recall, false positive rate (FPR), F1 score, and specificity(TNR). The RBF model, however, demonstrates the best precision. Overall, the Polynomial model shows excellent performance in binary classification, making it a strong choice due to its distinct strengths.

4.2 AUC and ROC

AUC (Area Under the Curve) is another evaluation metric used to assess the performance of a binary classification model. A higher AUC indicates better discrimination and a more effective separation between positive (the minority class) and negative (the majority class) instances.

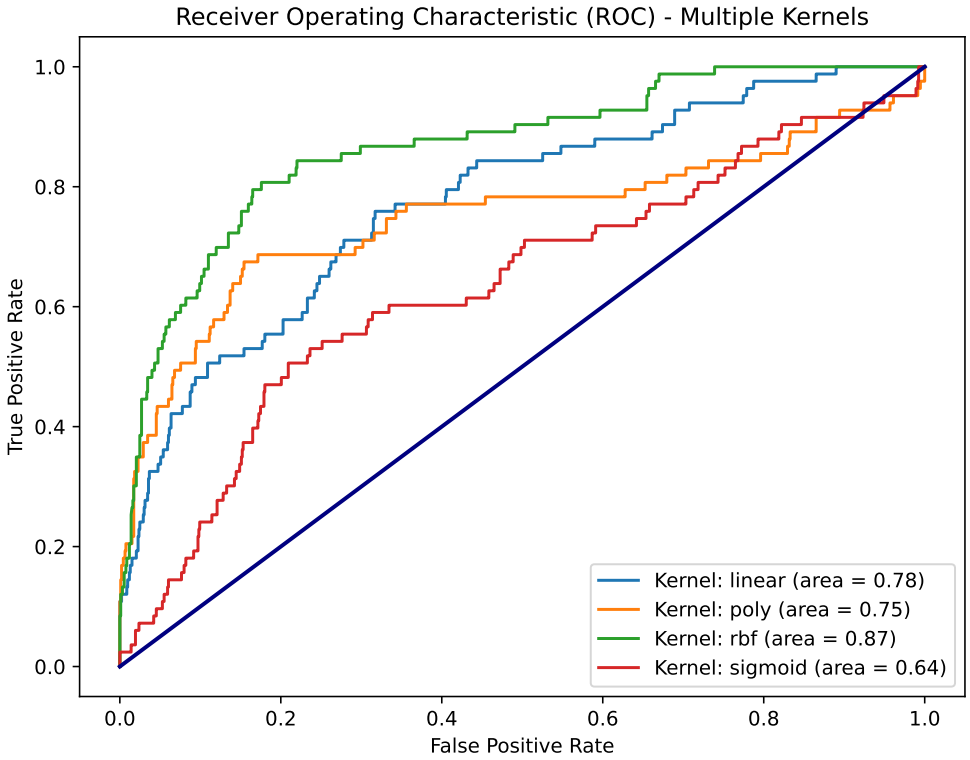

Figure 8: Comparison of AUC Values for SVM Models Using Four Different Kernels

Observing the AUC values in Figure 8, we can draw several inferences based on different ranges. The Sigmoid Model, with an AUC of 0.64, suggests moderate discriminatory power. Although it has some ability to distinguish between classes, its overall performance is limited. In contrast, the Linear Model and the Polynomial Model, with AUC values of 0.78 and 0.75 respectively, exhibit acceptable discrimina-

tory power and can effectively differentiate between positive and negative instances. The RBF Model, with an AUC of 0.87, demonstrates strong discriminatory power and can distinguish between classes effectively, reflecting robust predictive performance. Considering the AUC values, the RBF Model emerges as the most suitable model for this dataset.

4.3 Cross-validation

In our case, after dividing the dataset into a training set (70%) and a testing set (30%), we use the training set to perform cross-validation. With 5-Fold Cross-Validation, the training set is divided into 5 equally sized subsets. The process involves iteratively training the model on 4 of these subsets while using the remaining subset for validation. This process is repeated 5 times, with each subset taking turns as the validation set. The model's performance is then evaluated by averaging the results across the 5 validation iterations. Figure 9 presents a comprehensive summary of the cross-validation outcomes.

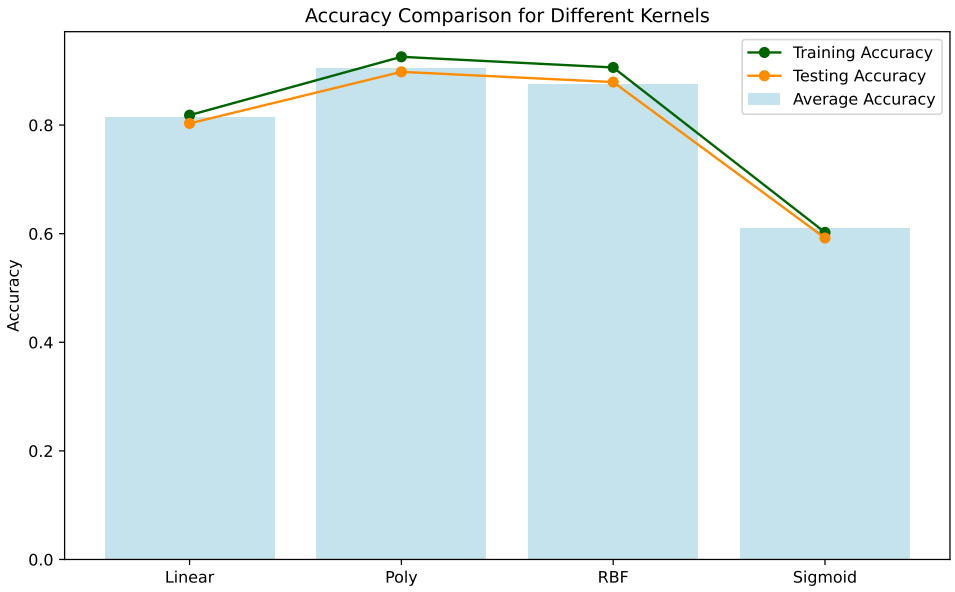

Figure 9: Cross-Validation Results for SVM Models

Notably, the small difference between the training and testing accuracies indicates good model consistency, with no significant overfitting or underfitting observed. The SVM models with Polynomial kernel achieves the highest average accuracy in cross-validation, while the Sigmoid kernel performs the worst.

5 Parameter tuning and conclusion

When evaluating model performance, it's essential to strike a balance between recall and accuracy. This objective aims to achieve a harmonious blend of correctly identifying cases from both classes while maintaining overall prediction accuracy. For a bank, this balance is crucial to increase profitability by issuing more loans and avoiding the misclassification of good customers as bad. Pursuing this balance ensures that the bank can accurately identify both good and bad customers, optimizing both financial gains and risk management.

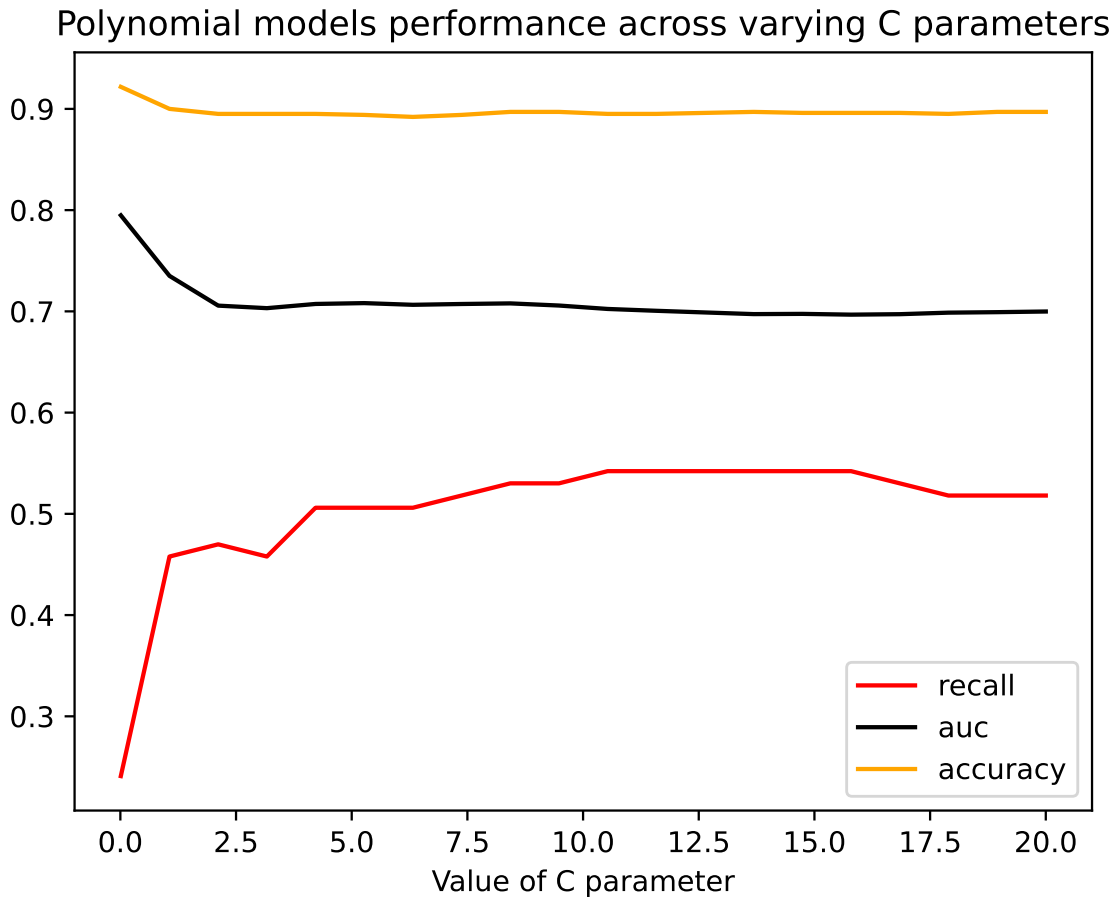

Figure 10: Learning Curve of Polynomial Model with Varied C Parameter Values

In our pursuit of achieving improved results in both accuracy and recall, we explored the impact of adjusting the C parameter in the Polynomial model (see Section 2.2.2 above). Figure 10 illustrates the model's performance across varying C parameters ranging from 0.01 to 20. Notably, the highest AUC value reaching 0.79, was observed at C = 0.01, with a testing accuracy of 0.9217 and a recall of 0.2410. Although the testing accuracy improved, the significant drop in recall indicates that the model became less effective at identifying the minority class. This suggests that fine-tuning the C parameter may not significantly enhance the overall performance of the Polynomial model, as it risks compromising recall in favor of accuracy, thereby making it challenging to strike a favorable balance between the two metrics.

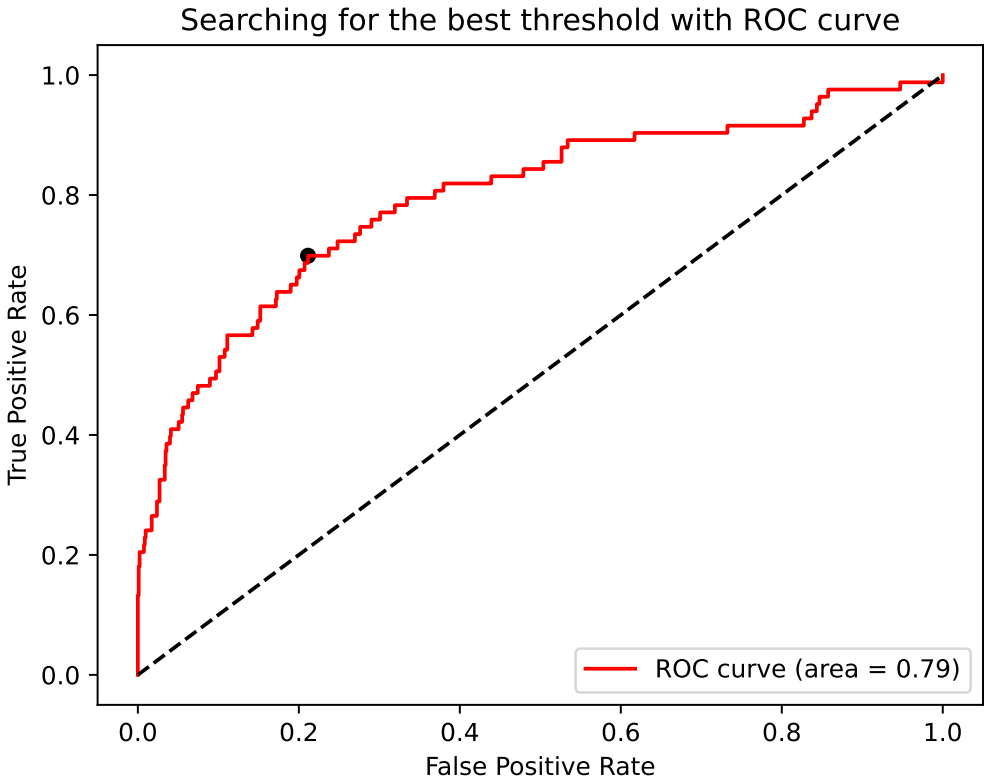

Figure 11: Searching for the best threshold with ROC curve

Adjusting only the parameters of the SVM itself is no longer sufficient to meet our requirements. We need to investigate whether improvements can be achieved by tuning the ROC curve threshold for the model. By adjusting the thresholds on the

ROC curve (see Figure 11), we aim to find the optimal balance between recall and the false positive rate, maximizing the distance between recall and the false positive rate. This means we aim for high recall while keeping the false positive rate low, demonstrating effective detection of positive samples while minimizing incorrectly identifying negative samples as positives. Typically, finding a balanced point in the classification of positive and negative classes helps achieve a balance in model performance between accuracy and recall. Our final model testing accuracy is 0.7812, and recall is 0.6988. This result maintains a balance between accuracy and recall.

6 Discussion

6.1 Predictive power

Section 4.1 presents the performance metrics of four SVM models, including accuracy, precision, recall, false positive rate (FPR), F1 score, and specificity.

The accuracies of the four SVM models were 0.8030, 0.8980, 0.8792, and 0.5920, respectively. Notably, the Polynomial SVM model achieved the highest accuracy at 89.80%, indicating its superior performance in classifying samples within the dataset, followed by the RBF model. This observation aligns with our prediction from the two-dimensional visualizations in Figure 5, which suggested the presence of nonlinear data relationships where linear kernel models might not perform optimally. The Sigmoid SVM model, however, recorded the lowest accuracy at 59.20%.

To further understand the classification capabilities of each model, we analyzed additional performance metrics such as False Positive Rate (FPR) and True Negative Rate (TNR).

For the linear kernel: The model has a False Positive Rate (FPR) of 0.1726 and a True Negative Rate (TNR) of 0.8274. Although the FPR is relatively high, indicating some misclassification of negative samples, the TNR reflects that the model still performs reasonably well in identifying negative samples.

For the polynomial kernel: The model has an FPR of 0.0604, which is lower than that of the linear kernel, indicating fewer misclassifications of negative samples. The TNR is high at 0.9396, demonstrating excellent accuracy in identifying negative samples.

This suggests that the polynomial kernel model performs best in handling negative samples.

For the radial basis function (RBF) kernel: The model shows an FPR of 0.0992 and a TNR of 0.9008, indicating robust classification performance with high accuracy for both positive and negative samples. While the RBF kernel performs well overall, the polynomial kernel still shows superior performance in terms of TNR.

For the Sigmoid kernel: The model's high FPR of 0.4088 and low TNR of 0.5912 indicate poor performance in classifying negative samples, highlighting its weaker classification capabilities.

In summary: Based on the analysis, the polynomial kernel SVM model exhibits the best overall performance, particularly excelling in the classification of negative samples. It is well-suited for datasets with complex, nonlinear relationships. In contrast, the Sigmoid kernel SVM model's higher FPR and lower TNR indicate less reliable classification performance, making it less favorable for accurate classification.

6.2 Generalization capability

Generalization capability refers to the performance of a machine learning model on unseen data, specifically its predictive ability on data outside the training set. Models with strong generalization capability typically exhibit stable and accurate performance across various evaluation metrics, whereas models with weaker generalization capability may perform poorly on certain metrics, especially on unseen data. Table 6 shows the results of cross-validation from four SVM models.

Considering the differences between training and testing accuracy, as well as the average accuracy, we draw the following conclusions:

Polynomial Kernel: The Polynomial kernel demonstrates the best generalization ability with the small difference between training and testing accuracy (0.0277). It achieves the highest average accuracy (0.9048) and is ranked first in testing accuracy (0.8980), indicating its superior performance.

RBF Kernel: The RBF kernel shows good generalization ability with a slightly larger difference between training and testing accuracy (0.0269). Its testing accuracy (0.8792) is the second highest, and its average accuracy (0.8760) is also competitive.

Table 6: Cross-validation from SVM models

	"Linear"	"Poly"	"Rbf"	"Sigmoid"
1 fold score	0.8565	0.9108	0.8811	0.6242
2 fold score	0.8132	0.9002	0.8917	0.6051
3 fold score	0.7919	0.8854	0.8471	0.6200
4 fold score	0.8301	0.9108	0.8896	0.6030
5 fold score	0.8000	0.9170	0.8702	0.5957
Average accuracy	0.8144	0.9048	0.8760	0.6096
Training accuracy	0.8182	0.9257	0.9061	0.6024
Testing accuracy	0.8030	0.8980	0.8792	0.5920

However, it does not surpass the Polynomial kernel in these metrics.

Linear Kernel: The Linear kernel exhibits a moderate generalization ability with a difference of 0.0152 between training and testing accuracy. Its testing accuracy (0.8030) is the lower among the models, and its average accuracy (0.8144) is also lower compared to the Polynomial and RBF kernels.

Sigmoid Kernel: The Sigmoid kernel has the weakest generalization ability, with the difference between training and testing accuracy (0.0104). Both its testing accuracy (0.5920) and average accuracy (0.6096) are the lowest among the models, indicating poorer performance.

In summary: The Polynomial kernel achieves the highest average accuracy and shows the best balance between training and testing accuracy, making it the most suitable choice for achieving strong generalization. Therefore, we recommend using the Polynomial kernel SVM model for better overall performance.

6.3 Improvements

6.3.1 Feature Creation

To enhance model performance, we can leverage feature engineering, which involves creating new features from existing ones to improve the model's effectiveness.

We generated new features by combining existing ones to capture ratios or propor-

tional relationships. The following new feature was created:

Mortdue-to-Value Ratio: Derived by dividing "Mortdue" (mortgage due) by "Value" (property value):

$$\text{Mortdue-to-Value Ratio} = \frac{\text{Mortdue}}{\text{Value}}$$

Introducing this new features into SVM models resulted in varying performance outcomes. Detailed comparisons can be found in Table 7.

Table 7: Models Performance with new feature: Mortdue-to-Value Ratio

Kernel Function	Original accuracy	New Features Accuracy
Linear	0.8030	0.8129
Polynomial	0.8980	0.9059
RBF	0.8792	0.8802
Sigmoid	0.5920	0.6010

Introducing the "Mortdue-to-Value Ratio" improved the performance of all kernel models, as reflected in the increased accuracy across each model.

6.3.2 Choosing Neural Networks Instead of SVM

Since SVM models require manual feature selection, which involves significant human intervention, opting for neural networks offers a more efficient and potentially more effective alternative.

Neural networks can automatically learn and extract features from raw data, thereby reducing the need for labor-intensive feature engineering. Additionally, neural networks are known for their flexibility in handling complex data patterns and can potentially achieve superior performance compared to traditional SVM models. This is especially true in scenarios involving high-dimensional or unstructured data.

6.4 Conclusion

In this study, we compared four kernel methods for binary classification using SVM models: linear, polynomial, radial basis function (RBF), and sigmoid kernels. The

accuracy values for these models were 0.8030, 0.8980, 0.8792, and 0.5920, respectively, while the AUC values were 0.78, 0.75, 0.87, and 0.64. These results demonstrate the effectiveness of the SVM models in classifying binary data. The Polynomial model stood out as the best performer, achieving the highest accuracy and the most robust cross-validation results. However, when considering AUC values, the RBF model appears to be the most suitable choice for this dataset. Additionally, cross-validation results indicated that none of the models showed signs of overfitting or underfitting.

Furthermore, in the pursuit of an optimal balance between accuracy and recall, the Polynomial model achieved accuracy and recall values of 0.7812 and 0.6988, respectively, underscoring its robustness in handling the dataset.

Additionally, to address class imbalance, we applied class weight balancing, which successfully adjusted the weights between different classes, enhancing the model's performance.

7 Appendix

Table 1: The variables of the dataset

Name	**Type**	**Description**
Bad	Category	Target: 1 = applicant defaulted on loan or seriously delinquent; 0 = applicant paid loan
Loan	Continuous	Amount of the loan request: value ranges from $1100 to $89900
Mortdue	Continuous	Amount due on existing mortgage: value ranges from $2063 to $399550
Value	Continuous	Value of current property: value ranges from $8000 to $855909
Reason	Category	DebtCon = debt consolidation; HomeImp = home improvement
Job	Category	Occupational categories: Office: office work ProfExe: professional executive Mgr: manager Self: self-employed Other: other Sales: sales
Yoj	Discrete	Years at present job: value ranges from 0 year to 41 years
Derog	Discrete	Number of major derogatory reports: value ranges from 0 to 10 times
Delinq	Discrete	Number of delinquent credit lines: value ranges from 0 to 15 times
Clage	Continuous	Age of oldest credit line in months: value ranges from 0 to 1168 months
Ninq	Discrete	Number of recent credit inquiries : value ranges from 0 to 17 times
Clno	Discrete	Number of credit lines: value ranges from 0 to 71 times
Debtinc	Continuous	Debt-to-income ratio: value ranges from 0.52 to 203

Table 2: Descriptiv statistics for the quantitative variables in the dataset

	count	mean	std	min	25%	50%	75%	max
LOAN	3364.0	19154.4	10875.4	1700.0	12000.0	17000.0	23825.0	89900.0
MORTDUE	3364.0	76249.6	45095.4	5076.0	49351.3	67278.5	92986.8	399412.0
VALUE	3364.0	107501.4	54728.2	21144.0	71235.0	94453.5	122339.3	512650.0
YOJ	3364.0	9.1	7.6	0.0	3.0	7.0	13.0	41.0
DEROG	3364.0	0.1	0.6	0.0	0.0	0.0	0.0	10.0
DELINQ	3364.0	0.3	0.8	0.0	0.0	0.0	0.0	10.0
CLAGE	3364.0	181.0	82.8	0.5	118.7	176.7	230.4	1168.2
NINQ	3364.0	1.0	1.5	0.0	0.0	1.0	2.0	13.0
CLNO	3364.0	22.1	9.4	0.0	16.0	21.0	27.0	64.0
DEBTINC	3364.0	34.1	8.0	0.8	29.4	35.1	39.1	144.2

Derivation of Performance Metrics for RBF Kernel

We use the RBF kernel as an example, utilizing the RBF kernel confusion matrix from Section 4.1 and the confusion matrix from Section 2.3.1 to compute Precision, Recall, and F1-Score. These metrics are averaged and weighted according to the number of true instances for each label (support).

Precision

The precision for each class can be computed as follows:

$$\text{Precision\_positive} = \frac{\text{True\_positive}}{\text{True\_positive} + \text{False\_positive}} = \frac{53}{53 + 30} \approx 0.366$$

$$\text{Precision\_negative} = \frac{\text{True\_negative}}{\text{True\_negative} + \text{False\_negative}} = \frac{835}{835 + 30} \approx 0.965$$

To compute the weighted precision:

$$\text{Precision\_weighted} =$$

$$= \frac{\text{Numbers\_Positive}}{\text{Total\_population}} \times \text{Precision\_positive} + \frac{\text{Numbers\_Negative}}{\text{Total\_population}} \times \text{Precision\_negative}$$

$$= \frac{(53+30) \times 0.366 + (835+92) \times 0.965}{1010} \approx 0.9160$$

Recall

The recall for each class can be computed as follows:

$$\text{Recall\_positive} = \frac{\text{True\_positive}}{\text{True\_positive} + \text{False\_negative}} = \frac{53}{53+30} \approx 0.639$$
$$\text{Recall\_negative} = \frac{\text{True\_negative}}{\text{True\_negative} + \text{False\_positive}} = \frac{835}{835+92} = 0.901$$

To compute the weighted recall:

$$\text{Recall\_weighted} =$$

$$= \frac{\text{Numbers\_Positive}}{\text{Total\_population}} \times \text{Recall\_positive} + \frac{\text{Numbers\_Negative}}{\text{Total\_population}} \times \text{Recall\_negative}$$
$$= \frac{(53+30) \times 0.639 + (835+92) \times 0.901}{1010} \approx 0.8792$$

F1-Score

The F1-Score for each class can be computed as follows:

$$\text{F1-score\_positive} = \frac{2 \times \text{Precision\_positive} \times \text{Recall\_positive}}{\text{Precision\_positive} + \text{Recall\_positive}} = \frac{2 \times 0.366 \times 0.639}{0.366 + 0.639} \approx 0.465$$
$$\text{F1-score\_negative} = \frac{2 \times \text{Precision\_negative} \times \text{Recall\_negative}}{\text{Precision\_negative} + \text{Recall\_negative}} = \frac{2 \times 0.965 \times 0.901}{0.965 + 0.901} \approx 0.932$$

To compute the weighted F1-score:

$$\text{F1-score\_weighted} =$$

$$= \frac{\text{Numbers\_Positive}}{\text{Total\_population}} \times \text{F1-score\_positive} + \frac{\text{Numbers\_Negative}}{\text{Total\_population}} \times \text{F1-score\_negative}$$
$$= \frac{(53+30) \times 0.465 + (835+92) \times 0.932}{1010} \approx 0.8935$$

References

[1] Alshawi, B. (2024). Comparison of svm kernels in credit card fraud detection using gans. *International Journal of Advanced Computer Science & Applications*, 15(1).

[2] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20:273–297.

[3] Han, X., Chen, W., and Zhou, C. (2024). Musical genre classification based on deep residual auto-encoder and support vector machine. *Journal of Information Processing Systems*, 20(1):13–23.

[4] Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

[5] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

[6] Khalifa, A. N. (2024). Optimization heart disease prediction using independent component analysis and support vector machine. *International Journal of Current Innovations in Advanced Research*, pages 14–22.

[7] Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press.

[8] Smola, A. J. and Schölkopf, B. (1998). *Learning with kernels*, volume 4. Citeseer.

[9] Sonmez, M., Sabanci, K., and Aydin, N. (2024). Convolutional neural network-support vector machine-based approach for identification of wheat hybrids. *Eur Food Res Technol*, 250:1353–1362.

[10] Tan, P.-N., Steinbach, M., and Kumar, V. (2016). *Introduction to data mining*. Pearson Education India.