COLLEGE OF SCIENCE AND TECHNOLOGY

SCHOOL OF SCIENCE

DEPARTMENT OF MATHEMATICS

Application of Nonlinear Principal Component Analysis in Industrial Product Quality Assessment

By

Eric SIBOMANA

Student Number: 221025221

Thesis Submitted in Partial Fulfillment of the Academic Degree of

Master of Science

in

Applied Mathematics

(Statistical modeling and actuarial sciences)

Supervisor: Dr. UWIMBABAZI RUGANZU Léon Fidele

August 30, 2024

Declaration

I, Eric SIBOMANA, declare that the research outlined in this dissertation is entirely my original work. It has not been presented or submitted for any comparable academic recognition before. Unless expressly indicated otherwise through citation or acknowledgment, all the research presented here is my own intellectual contribution.

Student: Eric SIBOMANA

Signature:

Date: 30/08/2024

Certification

I, undersigned, certify that this projected titled Application of Nonlinear Principal Component Analysis in Industrial Product Quality Assessment was conducted under my supervision and has been submitted under my approval.
Supervisor: Dr. UWIMBABAZI RUGANZU Léon Fidele

Signature:

Date: 30/08/2024

Approved by:
Head of Department: Dr.Célestin KURUJYIBWAMI

Signature:

Date: 30/08/2024

Dedication

This research is devoted to my family, friends, coworkers, lecturers, colleagues, and students.

Acknowledgements

I am deeply thankful to God for the guidance, protection, and blessings received throughout my research. I extend my sincere gratitude to Dr. Uwimbabazi Ruganzu Leon Fidele for his invaluable supervision, patience, and dedication, which were instrumental in the completion of this project. I also appreciate the support from the University of Rwanda's Department of Mathematics and the significant assistance from ISP with the EAUMP/UR-CST node.

I would like to express my heartfelt thanks to Father Wellars, Father Jean Paul, and Father Jean Claude for their encouragement and for granting me permission to continue my studies on behalf of College St. Ignace Mugina. I also extend my gratitude to my coworkers for their unwavering support. Your generosity and inspiration have been invaluable to me. May you be richly blessed.

Finally, I would like to express my heartfelt gratitude to my mother, siblings, uncles, aunts, colleagues, and friends for their unwavering support throughout this journey, including their prayers, guidance, and technical assistance.

Eric SIBOMANA

Abstract

This thesis examines the use of nonlinear PCA in data analysis, aiming to reduce information loss during dimensionality reduction while preserving essential data patterns. The study utilizes the wine dataset, selected for its relevance to customer segmentation, containing 14 variables that present a challenging test case for evaluating nonlinear PCA techniques on high-dimensional data. The dataset's complexity and practical significance in the wine industry add considerable value to the research outcomes.

Data visualization using Python revealed that traditional PCA produced mixed clusters, whereas Kernel PCA generated distinct clusters, highlighting its superior performance. The dataset matrix exhibited a high condition number of 16251.1265, indicating poor conditioning. Standard PCA resulted in a condition number of 1.502, pointing to sensitivity issues. Conversely, Kernel PCA achieved a lower condition number of 1.42, reflecting its better handling of poorly conditioned datasets and reduced information loss.

The research highlights Kernel PCA's potential to enhance data analysis and quality assessment in industrial settings. The thesis concludes by suggesting further research to explore the practical applications of nonlinear PCA across various industrial domains, contributing to the fields of applied mathematics and industrial engineering.

Enumeration of acronyms

The matrices, vectors, scalars, and elements in matrices and vectors containing lowercase characters shall all be denoted with bold font uppercase letters throughout this thesis.

$\mathbf{A}^T$: matrix transposed

$var(x)$: Variance

Σ: The matrix of covariance

ASM Active Shape Model

EVD: Eigenvalue Decomposition

BD : Big Data

DR: Dimension Reduction

I: Identity matrix

PCA: Principal Component Analysis

NLPCA :Nonlinear Principal Component Analysis

KPCA: Kernel Principal Component Analysis

SVD : Singular value decomposition

SVM : Support Vector Machine

RBF: Radial basis function

MDs: Multidimensional signals

LLE : Locally linear embedding

LLS: Linear Least Square

FP: False Positive

FN: False Negative

TP: True Positive

TN: True Negative

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

PCA is the primary and most commonly employed approach for analyzing multivariate data. In 1901, Pearson made its initial presentation of it, and in 1933, Hotelling built on it independently. Similar to numerous other multivariate techniques, its acceptance and utilization were limited until the introduction of electronic computers. However, at present, nearly all statistical software suites incorporate it extensively. The efficient technique of PCA can be employed to uncover patterns within datasets that may possess high dimensionality. The majority of the structure is frequently explained by a limited number of principle components [1].

Even though carrying four or more variables makes the calculations more difficult, the case is nevertheless made that they are perfectly doable. Performing PCA manually is only feasible with four variables or fewer. However, PCA typically yields better results when applied to datasets with more than four variables. To effectively analyze the data using PCA, we need a strong understanding of both matrix algebra and statistics. PCA's main goal is to find patterns and correlations in a dataset with higher dimensions so that it can be drastically reduced in dimension without losing any important information [2].

To address this problem, dimensionality reduction methods are pivotal, as they decrease the number of dimensions while retaining vital data. PCA stands out as a commonly used technique for linear dimension reduction, applicable across diverse fields. However, in cases where data relationships are nonlinear, PCA's reliance on linearity might be inappropriate, as it assumes linear connections between variables.

As a logical progression of PCA, NPCA looks for a limited number of underlying elements or sources to create a multivariate data set that may most effectively explain the variance-covariance structure that has been seen. From related nonlinear structural analysis techniques that have been developed during the last few decades, nonlinear PCA is a subset [3].
Nonlinear PCA expands upon traditional PCA to accommodate nonlinear datasets through the utilization of nonlinear transformations, such as kernel functions, neural networks, or manifold learning techniques. NLPCA can map the data to a higher-dimensional feature space where it becomes linearly separable, and then apply PCA to find the nonlinear principal components. NLPCA can potentially reduce the loss of information by capturing the nonlinear structure and variability of the data [4].
In the realm of machine learning, there exists a technique known as kernel PCA, which serves to decrease nonlinear dimensionality. Unlike the conventional PCA, which focuses on linear relationships to identify key features or components within a dataset, KPCA extends this methodology. It achieves this by applying a nonlinear mapping function to the dataset before conducting PCA, thereby enabling the capture of intricate and nonlinear connections between data points.

In KPCA, the original data undergoes transformation into a high-dimensional feature space through a kernel function. This helps to capture the nonlinear interactions between the data points in an effective manner using linear methods like PCA. Next, the altered data's principal components are generated, which can be applied to data visualization, clustering, or classification tasks [5].
It is suggested that this study concentrate on datasets that cannot be separated

linearly and instead use polynomial kernel functions to do it. A technique for dimension reduction and examining the dependence structure of multivariate data is KPCA. It presents an efficient approach to reduce dimensions and group data, especially in cases where there exist nonlinear relationships among data points. To capture intricate nonlinear relationships, KPCA initially employs a kernel function to nonlinearly transform the data. The modified data is then subjected to PCA in order to identify the major components. Through supervised learning or by taking use of the nonlinear transformation's invertibility in specific subdomains, the original input signal from KPCA can be reconstructed. Numerous applications, such as denoising tasks and defect detection in industrial systems, have demonstrated the effectiveness of KPCA [6, 7, 8, 9, 10].

1.2 Problem statement

Traditional linear PCA operates under the assumption of linearity in data, which can result in significant information loss when applied to complex, high-dimensional datasets. This study seeks to address the limitations of linear PCA by exploring nonlinear dimensionality reduction techniques. This study evaluates the effectiveness of nonlinear PCA methods, including autoencoders and Kernel PCA, in datasets with prominent nonlinearity, such as those with complex patterns or variable interactions. The goal is to determine if these advanced techniques outperform linear PCA in preserving data integrity and enhancing analytical results.

1.3 Objectives and Contributions

1.3.1 The Main Objective

The aim of this study is to evaluate the benefits of utilizing NPCA methods for improving the accuracy and efficiency of data classification, particularly in the context of logistic regression.

1.3.2 The Specific Objectives

This research aims to achieve the following precise objectives:

1. Determine whether kernel PCA, locally linear embedding, autoencoders, and other nonlinear PCA techniques are effective at preserving information in large datasets by evaluating metrics such as reconstruction error, explained variance, and computational efficiency. This will also involve assessing how well these techniques improve the classification accuracy of logistic regression models by retaining critical features in the reduced dimensionality space.

2. Examine how nonlinear PCA performance is influenced by various kernel functions by evaluating metrics such as reconstruction error, explained variance ratio, classification accuracy, computational efficiency, and result stability. This will provide a comprehensive assessment of how hyperparameters, including those of kernel functions, affect the effectiveness of nonlinear PCA techniques in enhancing the accuracy of logistic regression models.

3. Assess the effectiveness of nonlinear PCA methods using datasets from various industries by measuring dimensionality reduction performance, reconstruction error, classification accuracy, and computational efficiency. This will involve evaluating the generalizability of these methods across different industry datasets and determining their applicability in improving the performance of logistic regression for data classification tasks.

These specific objectives aim to provide insights into how nonlinear PCA methods can effectively minimize information loss during dimensionality reduction and their applicability in real-world scenarios.

1.4 Methodology

In this section, the methodology utilized to perform nonlinear PCA on the data is elucidated, in addition to an evaluation of its influence on mitigating the loss of information via dimensionality reduction. This data is secondary data, and it encompasses 14 variables. Nonlinear PCA is a method that expands upon the traditional PCA to effectively handle qualitative and nonlinear data, accomplished through the utilization of optimal quantification and neural networks.
Nonlinear techniques for reducing dimensionality, such as kernel PCA and autoencoders, aim to minimize information loss during the process. Nevertheless, these methods often encounter challenges in terms of their interpretability. To overcome this limitation, a solution has been proposed: the kernel method for PCA. This approach enhances interpretability by employing a nonlinear transformation on the data using a kernel function. This makes it possible for the technique to capture complex nonlinear interactions. The modified data is then subjected to PCA in order to determine the major components.

To maintain data quality for PCA, the study implemented essential preprocessing techniques. Normalization was performed to standardize variables, ensuring consistent scales for precise PCA analysis. Handling missing values was addressed through imputation or alternative methods to preserve data integrity and prevent bias. These preprocessing measures were vital before applying dimensionality reduction techniques such as PCA, KPCA, and LLE. Effective preprocessing guaranteed that the dimensionality reduction methods could be accurately and reliably evaluated. This underscores the critical role of data preparation in obtaining significant results from PCA techniques [11].
In this thesis, we compare various nonlinear PCA techniques to offer valuable insights into the strengths and limitations of each method. This comparative analysis can assist researchers and practitioners in selecting the most appropriate approach for different types of data.

Nonlinear PCA is particularly beneficial for high-dimensional datasets, where linear methods may struggle to reduce dimensionality while retaining crucial information. By employing nonlinear techniques, data can be represented more concisely and meaningfully, extracting more significant features.
This section explores two approaches to dimensionality reduction, with the most widely recognized being the linear method, which is discussed in detail in Subsection 2.3 (PCA).
The goal of employing nonlinear PCA for dimensionality reduction is to decrease the dataset's dimensions while preserving as much information as possible. This involves employing techniques such as kernel PCA to grasp the intricate patterns and configurations existing within the data.

Nonlinear PCA operates by employing a nonlinear mapping function to transform the data into a higher-dimensional feature space, followed by the application of PCA within this space to identify the principal components.

Initially, we define the data matrix x, which comprises m variables and n observations:

$$x = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \tag{1.1}$$

Next, generate the mean vector $\bar{\mu}$ by summing each column of x and dividing the result by the total number of observations, n.
$\bar{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}(x_1 + x_2 + ... + x_n)$; where x_i represents a vector containing the m features of observation i.
A square matrix called the Covariance Matrix represents the correlation between various variables. The covariance matrix for the m-dimensional dataset has a size of $m \times m$, with each item representing the covariance between two distinct dimensions. In the context of an m-dimensional dataset, the covariance matrix delineates both

the extent (variance) and alignment (covariance) of our data. Features of the matrix of covariance:

1. As $cov(x, y) = cov(y, x)$, the covariance matrix is implied to be symmetric.
2. The covariance matrix is a square matrix, meaning it has the same number of rows and columns.
3. $A\Sigma A^T$ is the positive semidefinite matrix that represents the covariance matrix of the data in a transformed space, where A is a transformation matrix (such as from PCA) and Σ is the diagonal matrix of eigenvalues or singular values.

The covariance value between one dimension and itself is known as a diagonal element. Variance is expressed as $cov(x, x) = var(x)$ on diagonal elements, in place of covariance.

1.5 Scope and Depth

PCA is widely used but has limitations. It assumes linear relationships, which can be inadequate for complex datasets with nonlinear relationships. PCA is sensitive to data scaling, can lose important information during dimensionality reduction, and its components can be difficult to interpret. Additionally, PCA is sensitive to outliers, assumes constant variance across variables, and can be computationally expensive for large datasets.
KPCA highlights its novel approach to nonlinear PCA, emphasizing its advantages in feature extraction. However, it identifies significant challenges, including high computational complexity, difficulty in selecting the appropriate kernel, non equivalence to explicit nonlinear mapping, and dependence on sample size. These limitations suggest areas for further research to enhance the effectiveness of KPCA in nonlinear dimensionality reduction [3].

1.6 Outline of the thesis

The five chapters that follow make up this thesis. Introduction is covered in Chapter 1. In-depth reviews of previous studies on PCA, nonlinear techniques, and their uses in dimensional reduction are given in Chapter 2. The methodology classical and nonlinear PCA is described in detail in Chapter 3. The results and a discussion of the findings in light of the study's objectives are provided in Chapter 4. The thesis is concluded in Chapter 5, which highlights its conclusions, highlights its recommendations, and suggests possible lines of inquiry for further research.

Chapter 2

Literature review

An overview of the fundamental ideas of dimensional reduction using linear algebra is given in this section. A multitude of academic works have been generated exploring this subject matter, encompassing investigations into nonlinear approaches to dimensionality reduction. These strategies, which aim to minimize information loss during the dimensionality reduction process, have been developed with the purpose of addressing the limitations associated with linear techniques like PCA. Examples of these methodologies include kernel PCA and autoencoders [12]. Instead of projecting data linearly onto a hyperplane with fewer dimensions, these nonlinear techniques aim to maintain the configuration of data with many dimensions [13].

Path lasso penalized autoencoders are one such technique that improves interpretability by penalizing every path that an input takes through the encoder to become a latent variable. This limits the number of input variables that may be represented in each latent dimension [14]. These methods have demonstrated potential in reducing reconstruction errors and better maintaining the relative distances between objects in both the original and reconstructed spaces [15]. Through the utilization of nonlinear dimensionality reduction methods such as these, it becomes feasible to reduce information loss during dimensionality reduction, thereby enhancing the effectiveness of data analysis and modeling [16]. Dimensionality reduction is a technique used to address the complexities associated

with handling data that has a large number of dimensions. Currently, diverse methods are employed to identify the essential dimensions within a dataset, resulting in a notable reduction in overall dimensionality [17]. Projection Matrix Optimization involves utilizing SVD to enhance the efficacy of Compressive Sensing within Sparse signal sensing and reconstruction scenarios. Compressive sensing enables the sensing of Sparse signals using a reduced number of projections, followed by their reconstruction. In the traditional approach, a random projection matrix is employed. However, by designing a projection sensing matrix that is optimized for a specific class of signals, the accuracy of reconstruction can be further enhanced, or the number of measurement samples required can be reduced even further Sensing Systems [18].

A PCA Tutorial; We will give a clear explanation of the purpose of PCA starting with a basic example. In order to give an explicit answer, we shall proceed by applying further mathematical rigor and placing it within the context of linear algebra. We'll examine the close connection between SVD and PCA as a mathematical technique. This comprehension enable us to prescribe a practical application of PCA and gain an appreciation for the underlying presumptions [19], comparative analysis of dimensionality reduction in big data through PCA and SVD: A Case Study [20]; KPCA has been convincingly shown to be effective as an initial processing stage for classification methods, along with KPCA. Furthermore, KPCA represents an evolutionary advancement beyond linear PCA. It can accommodate more complex nonlinear variations and is superior at capturing nonlinear features [21] and an algorithm for extracting principal components online with a fixed point has been developed as an alternative to KPCA, which can be challenging or impractical for handling large-scale datasets. This necessity led to the creation of simpler iterative extraction algorithms, primarily focused on applications in image processing. While there have been recent attempts to introduce online KPCA extraction algorithms, many of them encounter issues with slow convergence rates [22].

2.1 Norm

2.1.1 Vector norm

All the vector norms we'll employ exemplify p-norms. These norms are defined for a positive integer p and an n-vector x. However, a more general formulation is possible:

$||x||_p = (\sum_{i=1}^{n} ||x_i||^p)^{1/p}$.

Important special cases are:

1-norm: $||x||_1 = \sum_{i=1}^{n} |x_i|$, when referring to the two-dimensional distance between two points expressed in "city blocks," it is frequently referred to as the Manhattan standard.

The 2-norm, denoted as $||x||_2 = (\sum_{i=1}^{n} ||x_i||^2)^{1/2}$, represents the square root of the sum of the squares of individual components of the vector x, and is commonly referred to as the Euclidean norm due to its association with the standard distance measure in Euclidean space.

The infinity norm, denoted as $||x||_\infty$, represents the maximum absolute value among all elements of vector x. It can be understood as a limit when p approaches infinity. While the qualitative outcomes of all these standards are comparable, there are situations where working with a specific norm may be more straightforward in terms of analysis or calculation. When evaluating how solutions of linear systems react to changes, the maximum norm, which is commonly expressed as the ∞-norm, or the 1-norm are both frequently used [23].

Example 2.1. *Vectors' Norms. With respect to* $x = [1.6, 1.2]^T$.

$||x||_1 = 2.8,\ ||x||_2 = 2.0,\ and\ ||x||_\infty = 1.6$

In general, for any n-vector $\mathbf{x}$, we have: $||x||_1 \geq ||x||_2 \geq ||x||_\infty$

For any p-norm of a vector, the following fundamental properties hold true, where x and y stand for any arbitrary vectors:

1. $||x|| > 0,\ when\ x \neq 0$

2. $||\gamma x|| = |\gamma|||x||$ for any scalar γ

3. $||x + y|| \leq ||x|| + ||y||$ (triangle inequality)

More broadly, a vector norm can be characterized as any function that assigns a real value to a vector while satisfying these three properties [23].

2.1.2 Matrix norms

Additionally, we require a means of quantifying the size or magnitude of matrices. Although there is room for a more expansive definition, the definition of each matrix norm that we shall employ is contingent upon an underlying vector norm. To be more precise, we define the matching matrix norm of a $m \times n$ given a vector norm matrix A created by
$||A|| = \max_{x \neq 0} \frac{||Ax||}{||x||}$, this particular matrix norm is defined in relation to or dependent on the vector norm. In essence, it gauges the maximum extent to which a matrix stretches any vector based on the specified vector norm in a straightforward manner.
Some matrix norms are much easier to compute than others:
The matrix norm corresponding to the vector 1-norm is simply the maximum absolute column sum of the matrix, $||A||_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$ The matrix norm associated with the vector infinity-norm is essentially the highest absolute sum of a row within the matrix, $||A||_\infty = \max_i \sum_{i=1}^{n} |a_{ij}|$ Calculating the matrix norm associated with the vector 2-norm is challenging.

Example 2.2. *Example Matrix norms:*

For the matrix $A = \begin{bmatrix} 2 & -1 & 1 \\ 1 & 0 & 1 \\ 3 & -1 & 4 \end{bmatrix}$, then $||A||_1 = 6$ *and* $||A||_\infty = 8$

The matrix norms that we have established meet the subsequent significant characteristics, in which A and B can be any matrices:

1. $||A|| > 0 \; if A \neq 0$

2. $||\gamma A|| = |\gamma|.||A||$ for any scalar γ.

3. $||A + B|| \leq ||A|| + ||B||$

4. $||AB|| \leq ||A||.||B||$

5. $||Ax|| \leq ||A||.||x||$ for any vector x.

Then, you can find a broader definition of a matrix norm in any real-valued function of a matrix that meets the first three of these criteria. Remember that the first two attributes taken together suggest that $||A|| = 0, if\ A = 0$ [23].

2.1.3 Matrix Condition Number

The condition number of a matrix A with respect to a particular norm $\|\cdot\|$ measures the extent of variation in the solution of a linear system $Ax = b$ when the coefficients of the matrix A are altered. It quantifies the sensitivity of the solution **x** to changes in the coefficients of A. Mathematically, the condition number $Cond(A)$ with respect to the norm $\|\cdot\|$ is defined as: $Cond(A) = \|A\| \cdot \|A^{-1}\|$.
This reflects how much the output value x can change for a small change in the input value b. Incorporating additional evaluation metrics such as reconstruction error, variance explained by components, and computational efficiency would provide a more comprehensive assessment of the techniques.

Definition 2.3. *The condition number of a square matrix A, which is not singular, can typically be calculated by multiplying the norm of A with the norm of its inverse, denoted as* $cond(A) = ||A||.||A^{-1}||$. *However, if A is singular, conventional techniques may not be applicable, resulting in* $cond(A) = \infty$ *[23].*

A greater condition number indicates heightened susceptibility to variations in the input data. With a large condition number, even slight alterations in the input data can result in significant changes in the output. On the flip side, a smaller condition number indicates that the matrix is less sensitive to changes. It's crucial to specify the norm, such as Euclidean or Frobenius, since different norms can influence the condition number of a matrix.

Example 2.4. *The condition number of a matrix can be determined by examining the result of matrix multiplication. By employing this approach, it becomes straightforward to compute the inverse of the matrix presented in Example 2.2.*

$$A^{-1} = \begin{bmatrix} 0.5 & 1.5 & -0.5 \\ -0.5 & 2.5 & -0.5 \\ -0.5 & -0.5 & 0.5 \end{bmatrix}.$$

Given that $||A^{-1}||1 = 4.5$ and $||A^{-1}||\infty = 3.5$, we can calculate the condition numbers as follows: $cond_1(A) = ||A||1.||A^{-1}||1 = 6 \times 4.5 = 27$ and $cond_\infty(A) = ||A||\infty.||A^{-1}||\infty = 8 \times 3.5 = 28$. To determine the condition number utilizing the 2-norm, consult Example 2.6.

2.1.4 Euclidean matrix norm

The Euclidean matrix norm, also known as the spectral norm or the induced 2-norm.

Definition 2.5. *The eucledian matrix norm is defined as the largest singular value of the matrix. For an $m \times n$ matrix A, it is denoted by $\|A\|_2$*

and calculated as follows $\|A\|_2 = \sigma_{\max}(A)$, where $\sigma_{\max}(A)$ represents the largest singular value of A.

In practical terms, the Euclidean matrix norm gauges the extent to which matrix A elongates vectors upon application.

This metric finds extensive utility across diverse fields such as numerical analysis, linear algebra, and control theory. The condition number of a matrix concerning the Euclidean norm is expressed as $cond_2(A) = ||A||_2.||A^{-1}||_2 = \frac{\sigma_{\max}}{\sigma_{\min}}$; where $\|A\|_2$ is the Euclidean norm of A and $\|A^{-1}\|_2$ is the Euclidean norm of its inverse.

Example 2.6. *The representation of the singular value decomposition (SVD) for matrix A in Examples 2.2 and 2.4 can be expressed as follows: $A = U\Sigma V^T$,*

$$\text{where } U = \begin{bmatrix} 0.392 & -0.920 & -0.021 \\ 0.240 & 0.081 & -0.967 \\ 0.888 & 0.384 & -0.253 \end{bmatrix},$$

$$\Sigma = \begin{bmatrix} 5.723 & 0 & 0 \\ 0 & 1.068 & 0 \\ 0 & 0 & 0.327 \end{bmatrix} \text{ and } V = \begin{bmatrix} 0.645 & -0.224 & 0.731 \\ -0.567 & 0.501 & 0.653 \\ 0.513 & 0.836 & -0.196 \end{bmatrix}$$

So that $||A||_2 = 5.723$ and $cond_2(A) = \frac{\sigma_{\max}}{\sigma_{\min}} = \frac{5.723}{0.327} = 17.5$.

So, a high value for the condition number $cond_2(A)$ implies that the matrix A is poorly conditioned, indicating that small changes in the input could lead to significant changes in the output. On the other hand, a low condition number suggests that the matrix is well-conditioned, meaning that small changes in the input would result in minor alterations in the output.

2.2 Linear Least Squares

Linear least squares is a technique utilized for determining the parameters of a linear regression model. Its aim is to minimize the overall squared discrepancy between the observed and forecasted values.

This method is frequently used when there's a requirement to establish a linear correlation between variables, especially in datasets that might include discrepancies or inaccuracies.

The general form of a linear regression model is

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon$, where y represents the dependent variable, while the parameters to be estimated are $\beta_0, \beta_1, \ldots, \beta_n$. The independent variables are denoted as $x_1, x_2, \ldots, x_n$ and ε denotes the error term.

2.2.1 Linear least squares problems

The matrix A often represents the coefficients in the system of linear equations in linear least squares scenarios. The objective is to identify the vector x, where b is the vector of observed values, that minimizes the residual sum of squares $\|Ax - b\|^2$. We express a linear least squares problem in the form of $Ax = b$. In the scenario where A is an A is an $m \times n$ matrix, x is an $n \times 1$ representing unknowns, and b is an $m \times 1$ vector, the least squares solution $\hat{x}$ is expressed as $\hat{x} = (A^T A)^{-1} A^T b$.

This approach aims to reduce the Euclidean distance of the difference vector $r = Ax - b$ to its minimum, which is represented as $||Ax - b||_2^2$. This interpretation implies that the approximation is made based on the least squares method in the context of the 2-norm.

Example 2.7. *An overdetermined system arises when there are more equations than unknowns, leading to redundancy. In this context, a surveyor aims to ascertain the elevations of three hills in relation to a specific reference point.*

Given that $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix}$, and $b = \begin{bmatrix} 1237 \\ 1941 \\ 2417 \\ 711 \\ 1177 \\ 475 \end{bmatrix}$, then

$$Ax = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \approx \begin{bmatrix} 1237 \\ 1941 \\ 2417 \\ 711 \\ 1177 \\ 475 \end{bmatrix} = b$$

Thus, $\hat{x} = (A^T A)^{-1} A^T b = [1236, 1943, 2416]^T$, where the measurements, differing slightly from the initial three height readings, represent a harmonized solution that effectively addresses discrepancies caused by measurement errors, thereby enhancing the alignment within the least squares method [23].

2.2.1.1 Existence and uniqueness

In linear least squares, the existence of a solution hinges on the continuity exhibited by the function $\phi(y) = ||b - y||_2$ within the space $\mathbb{R}^m$. This continuity guarantees that solutions remain feasible and accessible throughout the domain. Specifically, the input matrix, conventionally represented as A, achieves full column rank ($\text{rank}(A) = n$) when its columns are linearly independent. This criterion is vital as

it indicates the distinct singularity of solutions to an $m \times n$ least squares problem, often represented as $Ax \approx b$. Conversely, if the rank of A drops below n, indicated as rank$(A) < n$, the matrix is considered rank-deficient. In such instances, while solutions to the corresponding least squares problem persist, they lose their uniqueness. This emphasizes the pivotal role of column rank in determining the presence and exclusivity of solutions in linear least squares scenarios [23].

2.2.2 Sensitivity and conditioning

Let's now turn our attention to the sensitivity and conditioning of linear least squares problems. To begin with, we must extend the definition of the matrix condition number to include rectangular matrices. The definition of the condition number for a square matrix discussed in Section 2.1.3 is based on the matrix inverse. While a non square matrix A lacks a conventional inverse, we can establish a pseudoinverse, represented as A^+, which mimics an inverse in various aspects. Later on, we'll encounter a broader definition that pertains to any matrix. However, at this point, we're focusing solely on matrices A with complete column rank. In this scenario, where A^TA is nonsingular, we define the pseudoinverse of A as $A^+ = (A^TA)^{-1}A^T$.

We now define the condition number of an $m \times n$ matrix with $rank(A) = n$ to be $cond(A) = ||A||_2.||A^+||_2$.

By convention, $cond(A) = \infty$ if $rank(A) < n$. The condition number of a square matrix quantifies its proximity to singularity, whereas the condition number of a rectangular matrix quantifies its proximity to rank deficiency.

The stability of solving a least squares problem $Ax \approx b$ is influenced by both the matrix A and the vector b on the right-hand side. Yet, in the resolution of a square linear system $Ax = b$, the stability is solely influenced by the matrix A. Consequently, depending solely on the condition number cond(A) is insufficient for evaluating sensitivity. In particular, a small change in b cause only a slight alteration in $y = Pb$ if b is close to the subspace spanned by A.

If b is nearly perpendicular to the subspace spanned by A, then the resulting vector

$y = Pb$ be relatively small. Consequently, even a slight change in b can lead to a significant change in y, and subsequently in the least squares solution x perturbations [23].

2.3 Principal component analysis

Definition 2.8. *PCA is a statistical technique used for dimensionality reduction while preserving as much variance as possible in a dataset.*
It transforms the data into a new coordinate system where the axes, known as principal components, are orthogonal and ranked according to the amount of variance they capture from the data. The first principal component accounts for the greatest variance, the second principal component accounts for the second greatest variance, and so on.
The most used linear technique for reducing dimensionality is PCA. The principal component of the PCA, a statistical data analysis technique, are different sets of linear combinations created from the original set of input variables [17].

$$cov(x, y) = \begin{bmatrix} var(x_1, x_1) & \cdots & cov(x_1, x_d) \\ \vdots & \ddots & \vdots \\ cov(x_d, x_1) & \cdots & var(x_d, x_d) \end{bmatrix} \tag{2.1}$$

The dot product between two vectors x, and y is defined as: $x^T.y = \sum_{i=1}^{n} x_i.y_i$.
If two vectors are perpendicular, they are also orthogonal, which is mathematically expressed as $x^T y = 0$. PCA, commonly used in data analysis and machine learning, serves both feature extraction and dimensionality reduction. Although "perpendicular" and "orthogonal" are often used interchangeably in Euclidean space, orthogonality is a broader concept that applies to any inner product space. Perpendicularity specifically refers to a 90-degree angle between vectors in Euclidean space, while orthogonality encompasses a wider range of mathematical contexts [17].

2.4 Singular value decomposition

Let A be an $n \times n$ matrix. Before explaining what a singular value decomposition is, we first need to define the singular values of A.

Consider the matrix A^TA. This is a symmetric n n matrix, so its eigenvalues are real.

Lemma. If λ is an eigenvalue of A^TA, then $\lambda \geq 0$.

Proof. Let x be an eigenvector of A^TA with eigenvalue λ. We compute that $||Ax||^2 = (Ax)(Ax) = (Ax)^TAx = x^T(\lambda x) = \lambda x^Tx = \lambda||x||^2$.

Since $\lambda||x||^2 \geq 0$, it follows from the above equation that $\lambda||x||^2 \geq 0$.Since $||x||^2 > 0$ (as our convention is that eigenvectors are nonzero), we deduce that $\lambda \geq 0$.

Let $\lambda_1, ..., \lambda_n$ denote the eigenvalues of A^TA, with repetitions. Order these so that $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$.

Let $\sigma_i = \sqrt{\lambda_i}$ so that $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$.

Definition 2.9. *The numbers $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$ defined above are called the singular values of A.*

The number of nonzero singular values of A equals the rank of A.

Proof. The rank of any square matrix equals the number of nonzero eigen- values (with repetitions), so the number of nonzero singular values of A equals the rank of A^TA. By a previous homework problem, A^TA and A have the same kernel. It then follows from the rank-nullity" theorem that A^TA and A have the same rank.

□

2.4.1 How to find a SVD

Let A be an $m \times n$ matrix with singular values $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$, and let r denote the number of nonzero singular values. We now explain how to find a SVD of A.

Let $v_1, ..., v_n$ be an orthonomal basis of $\mathbb{R}^n$, where v_i is an eigenvector of A^TA with eigenvalue σ_i^2.

Lemma 2.10. *(a)* $||Av_i|| = \sigma_i$.

(b) If $i \neq j$ *then* Av_i *and* Av_j *are orthogonal.*

Proof. We compute
$(Av_i)(Av_j) = (Av_i)^T(Av_j) = v_i^T A^T A v_j = v_i^T \sigma_j^2 v_j = \sigma_j^2(v_i.v_j)$.
If $i = j$, then since $||v_i|| = 1$, this calculation tells us that $||Av_i||^2 = \sigma_j^2$ which proves (a).
If $i \neq j$, then since $v_i.v_j = 0$,this calculation shows that $(Av_i).(Av_j) = 0$.

□

Theorem 2.11. *Let* A *be an* $m \times n$ *matrix. Then* A *has a (not unique) singular value decomposition* $A = U\Sigma V^T$*, where* U *and* V *are as follows:*

1. *The columns of* V *are orthonormal eigenvectors* $v_1, \ldots, v_n$ *of* $A^T A$*, where* $A^T A v_i = \sigma_i^2 v_i$.
2. *If* $i \leq r$*, so that* $\sigma_i \neq 0$*, then the* i*th column of* U *is* $\sigma_i^{-1} A v_i$*. By Lemma 2.10, these columns are orthonormal, and the remaining columns of* U *are obtained by arbitrarily extending to an orthonormal basis for* $\mathbb{R}^m$ *[24].*

Proof. We just have to check that if U and V are defined as above, then $A = U\Sigma V^T$. If $x \in \mathbb{R}^n$, then the components of $V^T x$ are the dot products of the rows of V^T with x, so

$$V^T x = \begin{bmatrix} v_1.x \\ v_2.x \\ \vdots \\ v_n.x \end{bmatrix} \text{ and then } \Sigma V^T x = \begin{bmatrix} \sigma_1 v_1.x \\ \sigma_2 v_2.x \\ \vdots \\ \sigma_r v_r.x \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

□

When we multiply on the left by U, we get the sum of the columns of U, weighted by the components of the above vector, so that
$U\Sigma V^T x = (\sigma_1 v_1 \cdot x)\sigma_1^{-1} A v_1 + \cdots + (\sigma_r v_r \cdot x)\sigma_r^{-1} A v_r = (v_1 \cdot x)Av_1 + \cdots + (v_r \cdot x)Av_r$.

2.5 Relationship between PCA and SVD

The right singular vectors of A are associated with the eigenvectors of A^TA, while the left singular vectors correspond to the eigenvectors of AA^T. Let $A \in \mathbb{R}^{n\times d}$ be a matrix where n data points $a_1, a_2, \ldots, a_n$ in $\mathbb{R}^d$ are organized as its rows. Consider a random vector x uniformly distributed over the n data points. It follows that $E(AA^T) = \frac{1}{n}\sum_{i=1}^{n} a_i a_i^T = \frac{1}{n}A^TA$. As a result, the principal κ right singular vectors of matrix A align with the leading κ eigenvectors of the matrix $\frac{1}{n}A^TA$. Therefore, the subspace formed by the top κ right singular vectors of A corresponds to the rank-κ uncentered PCA, as stated in Theorem 2.1.

In this research, the dataset was carefully examined and categorized using various techniques for reducing dimensionality, including PCA and KPCA. A^TA and AA^T are both symmetric matrices and are positive semi-definite. It's clear that the eigenvalues of both A^TA and AA^T are non negative. Additionally, the non zero eigenvalues of the matrices A^TA and AA^T are identical and these matrices are symmetric and positive semi-definite.

Let λ denote an eigenvalue of A^TA along with its corresponding eigenvector v.
λ is a non zero eigenvalue of A^TA with a corresponding eigenvector of Av if $\lambda > 0$ and $Av = 0$ if $\lambda = 0$ [25].

2.6 Locally Linear Embedding

Locally Linear Embedding reduces dimensionality by preserving locally linear relationships between data points in a lower-dimensional space, mapping each point to a nearby linear patch. This is crucial for preprocessing multidimensional signals to find concise representations, as seen in speech spectrograms and facial images. PCA and MDs also reduce dimensionality, with PCA using the top eigenvectors of the data covariance matrix and MDS maintaining pairwise distances between points. However, LLE can be affected by noise, leading to the development of LLE with Additive Noise , which extracts noiseless data from noisy observations. Mathematical formulas

include:

1. **Locally Linear Embedding**:

$$\text{Minimize} \sum_i \left| \mathbf{x}_i - \sum_j w_{ij} \mathbf{x}_j \right|^2$$

In this context, w_{ij} represents the weight assigned to the data point $\mathbf{x}_j$ when reconstructing the data point $\mathbf{x}_i$.

2. **Principal Component Analysis**:

$$\text{Maximize } \mathbf{v}^T \mathbf{C} \mathbf{v} \quad \text{subject to } \mathbf{v}^T \mathbf{v} = 1$$

Here, $\mathbf{C}$ denotes the covariance matrix of the data.

3. **Multidimensional Scaling**:

$$\text{Minimize} \sum_{i<j} (d_{ij} - d'_{ij})^2$$

In this formula, d_{ij} represents the distance between data points i and j in the original space, while d'_{ij} denotes the distance between the corresponding points in the reduced space [26].

2.7 Autoencoders

An autoencoder is a neural network designed to compress and then accurately reconstruct input data. By reducing the dimensionality of the input data in the hidden layer, autoencoders create a concise representation of the input data at the output layer [27]. Applications of autoencoders include anomaly detection, data visualization, denoising, and semantic hashing [28]. The value of autoencoders lies in their ability to extract meaningful representations of data, not just in reproducing the input, and they can be used for dimensionality reduction, classification, and as components in other neural networks [29]. Autoencoders aim to capture a meaningful data representation through unsupervised learning, akin to mappings obtained from PCA [30].

Objective: $\min_{\theta} \|X - \hat{X}\|^2$ where $\hat{X} = f_\theta(g_\theta(X))$

Hidden Layer Representation: $h = g_\theta(X)$ where g_θ is the encoder function

Reconstruction: $\hat{X} = f_\theta(h)$ where f_θ is the decoder function

, where X represents the original input data, which the autoencoder compresses and reconstructs. θ are the learned parameters (weights and biases), and $\hat{X}$ is the reconstructed version of X, aiming to closely match the original data [30].

Chapter 3

Classical PCA and nonlinear PCA

There exist two primary forms of PCA:

3.1 Classical PCA

Linear Transformation: Conventional PCA works on the premise of a linear correlation between the original characteristics and the principal components. Its aim is to discover a linear transformation that maximizes the variability present in the dataset.
SVD is a crucial step in classical PCA. It centers on finding the eigenvalues and eigenvectors, which can be done by computing the eigenvalues of the covariance matrix or by applying SVD to the data matrix.
Orthogonality: It is ensured that the principal components can capture independent sources of variance in the data because they are mutually perpendicular.
Linearity: The initial features' linear combinations make up the principal components.

3.2 Nonlinear PCA

Nonlinear Relationships: Nonlinear PCA acknowledges that the underlying relationships in the data may be nonlinear. It seeks to identify nonlinear patterns

within the data that traditional PCA might overlook.
Kernel PCA: Kernel PCA is a popular nonlinear PCA technique. In this case, the data is implicitly transformed into a higher-dimensional space using a kernel function, which improves the efficiency of linear transformations. The nonlinear patterns that were present in the original feature space can be captured using this method.
Nonlinear PCA can be accomplished through manifold learning methods such as Isomap, Locally Linear Embedding , and t-Distributed Stochastic Neighbor Embedding . These approaches focus on preserving the local structure of the data by capturing its nonlinear characteristics.
Computational Complexity: Nonlinear PCA methods can be computationally more demanding than classical PCA, especially when dealing with large datasets or complex nonlinear relationships.

3.3 Classification using Support Vector Machine

3.3.1 Linearly separable and non separable data

If the data are linearly separable, then the traditional PCA approach talked about before is a useful linear projection technique. To decrease a dataset's dimensionality when dealing with linearly inseparable data, a nonlinear approach is needed.
Two dimensions make it simplest to see and comprehend the concept of linearly separable. Green and blue should be the colors used to symbolize the two grades. If a line can be drawn to divide the blue and green points from one another, the dataset is said to be linearly separable.
NPCA involves using techniques that capture nonlinear relationships in your data. One common approach for nonlinear dimensionality reduction is KPCA [31].

And the Figure 3.1 , A represents linearly separable data points and B represents nonlinear separable.

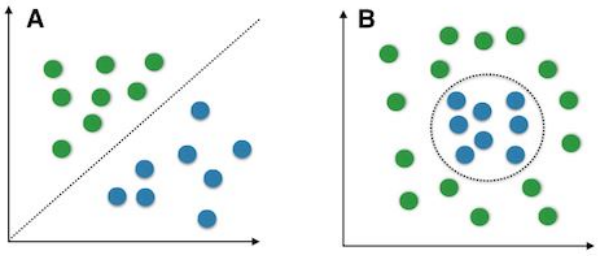

Figure 3.1: Linear vs Non linear problems

3.3.2 Classification using SVM

In a machine learning framework, classification is a supervised technique that involves two main steps: constructing a model by training it on a pre-categorized dataset with positive and negative labels, and then using the model to classify new, unseen documents. The preferred approach in this study is the SVM, which operates within a vector space over the real numbers ($\mathbb{R}$), where documents are represented as points based on selected features. Instead of fitting features, SVM optimizes the margins of data separation, making it suitable for large datasets. SVM training involves prepared, categorized text. The key components of SVMs are the real weight vector ($w \in \mathbb{R}^n$) and the bias term ($b \in \mathbb{R}$), which define the distance between the hyperplane and nearby points, denoted as ρ. The ultimate decision boundary for SVMs is the optimal hyperplane, found by maximizing the margin that separates data points.

Figure 3.2 illustrates the creation of an ideal hyperplane for two distinct datasets, using various shapes to represent each dataset [32].

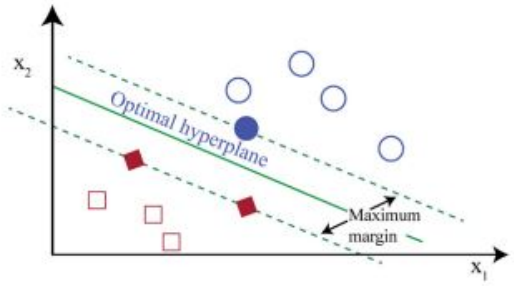

Figure 3.2: SVM using Hyperplane

.

3.4 Kernel functions and the kernel trick

Definition 3.1. *A **kernel function** is a mathematical formula that calculates the similarity or scalar product of two input vectors within a modified feature space.*

Kernel functions enable algorithms to operate implicitly in a higher-dimensional feature space without explicitly computing the transformation. In SVM, kernel functions are used to measure the resemblance between data points, facilitating the classification of data that isn't linearly separable. The fundamental strategy for addressing nonlinearly separable data involves mapping it into a higher-dimensional space where linear separation becomes feasible. This transformation is denoted by ϕ, known as the nonlinear mapping function. Consequently, the mapping of a sample $\mathbf{x}$ is expressed as $\mathbf{x} \to \phi(\mathbf{x})$, with the kernel function capturing this transformation. Currently, the term "kernel" refers to a function responsible for computing the dot product of the transformed versions of sample vectors under ϕ. Specifically, the kernel function can be expressed as $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)^T$.
PCA is fundamental for converting input data to diagonalize the estimated covariance matrix C. However, PCA, originally designed for linear data, is less effective for nonlinear data. To address this limitation, the Kernel PCA scheme was

proposed, which incorporates kernel methods into PCA. In Kernel PCA, the covariance matrix C^F in the feature space is given by $C^F = \frac{1}{n}\sum_{j=1}^{n}\phi(\mathbf{y}_j)\phi(\mathbf{y}_j)^T$. The kernel trick allows SVMs and other algorithms to operate in a higher-dimensional space without explicitly computing the transformed feature vectors, offering both computational and conceptual advantages when dealing with nonlinear relationships in data [31].

3.5 Kernel PCA

3.5.1 The motivation of kernel method

Traditional PCA faces challenges with complex and nonlinear data patterns, which led to the development of Kernel PCA. While traditional PCA is effective in identifying primary components and linear relationships within the original feature set, it struggles with nonlinear data structures or when detecting nonlinear patterns is crucial. Kernel PCA addresses these limitations by using the kernel trick to map data into a higher-dimensional space, thus handling nonlinear relationships more effectively.

The kernel method is significant in pattern analysis due to its ability to manage intricate and nonlinear data relationships. It is widely used in machine learning and data analysis for tasks such as handling nonlinearities, enhancing linear methods in high-dimensional contexts, and improving classification and regression performance. By implicitly converting data into higher-dimensional spaces, kernel methods enable effective analysis and modeling of complex patterns.

For instance, a kernel function κ is employed to understand geometric characteristics in the transformed space, aiding in the categorization of patterns. The kernel function can be exemplified as

$\phi(\mathbf{x})^T\phi(\mathbf{z}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)(z_1^2, \sqrt{2}z_1z_2, z_2^2) =$

$x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 + (x_1z_1 + x_2z_2)^2 = (\mathbf{x}^T\mathbf{z})^2 = \kappa(\mathbf{x}, \mathbf{z})$.

In contrast to PCA, which performs linear transformations, Kernel PCA leverages

these nonlinear kernel functions to capture complex data patterns.

3.6 Mathematical modeling

To address nonlinear relationships among input variables, we focus on characteristics that reflect these nonlinear connections rather than directly using PCs from the input space. Linear PCA is not suitable for such data due to its nonlinear nature, so we consider transforming the data into a higher-dimensional feature space $\Phi : \mathbb{R}^N \to F$, where $\mathbf{x} \to \Phi(\mathbf{x})$. PCA is then applied in this new feature space. The covariance matrix in the feature space F can be computed using traditional PCA method

$$\bar{C} = \frac{1}{M} \sum_{j=1}^{M} \Phi(x_j)\Phi(x_j)^T \tag{3.1}$$

$$\lambda \mathbf{V} = \bar{C}V. \tag{3.2}$$

However, performing eigenvalue decomposition in high-dimensional spaces is computationally expensive. Instead, the eigenvalue problem can be rephrased using dot products

$$\lambda(\Phi(x_k) \cdot V) = (\Phi(x_k) \cdot \bar{C}V) \tag{3.3}$$

for $k = 1, \ldots, M$. We express the eigenvectors V as linear combinations of feature vectors

$$V = \sum_{j=1}^{M} \alpha_i \Phi(x_i) \tag{3.4}$$

Focusing on the dot product of transformed feature vectors, we use kernel functions to simplify the calculations

$$\lambda \sum_{j=1}^{M} \alpha_i(\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_i)) = \frac{1}{M} \sum_{i=1}^{M} \alpha_i(\Phi(\mathbf{x}_k) \cdot \sum_{j=1}^{M} \Phi(\mathbf{x}_j))(\Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i)) \tag{3.5}$$

Here, the dot product in the feature space is represented by a kernel function $\kappa(x, y) = (\Phi(x) \cdot \Phi(y))$. This leads to a simplified eigenvalue problem

$$M\lambda\alpha = \kappa\alpha, \tag{3.6}$$

with the kernel matrix K specified as

$$\kappa_{ij} = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j). \tag{3.7}$$

Solving this equation yields the eigenvectors and eigenvalues. Kernel PCA uses an arbitrary function $\phi(\cdot)$, often without explicit calculation, to project data onto principal components:

$$(V^n \cdot \phi(\mathbf{x})) = \sum_{i=1}^{M} \alpha_i^n (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)) = \sum_{i=1}^{M} \alpha_i^n \kappa_{ij}, \tag{3.8}$$

where α_i are the eigenvectors of K and $\lambda_1, \ldots, \lambda_M$ are the eigenvalues of κ [5].

3.6.1 The standard PCA algorithm

Observations focused at the center: column vectors $x_i \in \mathbb{R}^N, i = 1, ..., M$ (centered indicating $\sum_{i=1}^{M} x_i = 0$).
PCA discovers the principal axes through the process of diagonalizing the covariance matrix $\bar{C} = \frac{1}{M} \sum_{j=1}^{M} \phi(x_j)\phi(x_j)^T$ and can be diagonalized with non negative eigenvalues.

3.6.2 Rewriting PCA in terms of dot product

We must keep in mind that the eigenvectors are contained within the linear combination of $x_1, ..., x_M$.
$\bar{C}V = \frac{1}{M} \sum_{j=1}^{M} \Phi(x_j)\Phi(x_j)^T V = \lambda V$.

Thus,

$$V = \frac{1}{M\lambda}\sum_{j=1}^{M} x_j x_j^T V = \frac{1}{M\lambda}\sum_{j=1}^{M}(x_j.V)x_j. \tag{3.9}$$

However, the fact that $(x_j.v)$ is merely a scalar implies that all solutions of v with $\lambda \neq 0$ can be expressed as $v = \sum_{i=1}^{n} \alpha_i x_i$

3.6.3 Algorithm

1. Compute $K_{ij} = (\kappa(x_i, x_j))_{ij}$

2. Solve $M\lambda\alpha = K\alpha$ by diagonal K, and normalizing expansion coefficients α^n by solving $1 = \lambda_n(\alpha^n.\alpha^n)$

3. Extract the principal components (corresponding to the kernel K) of a test point X: compute $(kPC)_n(x) = (V^n.\Phi(x)) = \sum_{i=1}^{M} \alpha_i^n(x_i, x)$ [5].

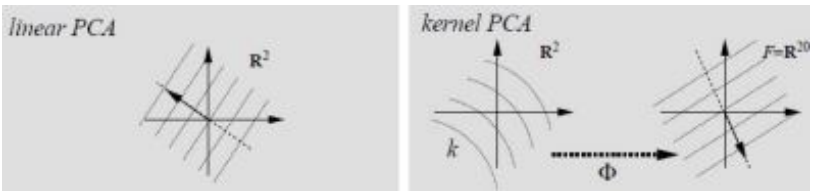

Figure 3.3: Kernel PCA vs Regular (linear) PCA

3.6.4 Kernel PCA's method

Kernel PCA represents an advanced iteration of the conventional PCA technique, incorporating nonlinearity. While PCA predominantly functions in a linear manner to diminish dimensionality, its objective persists in recognizing principal components as linear combinations of initial features that effectively capture the dataset's variance.

To be more precise, to calculate the nonlinear principal components y_i, we only require computing the elements of the matrix $\phi^T\phi$ and the elements of the vectors $\phi^T\phi = \phi^T\phi(x_j)$, we demonstrate that all these values can be derived from inner products like $\phi^T\phi$.

Definition 3.2. *The set comprising all functions that are integrable, defined as*

$$L^2(\mathbb{R}^D) = \{f : \mathbb{R}^D \to \mathbb{R}\} \text{ such that } \int f(x)^2 dx < \infty \tag{3.10}$$

Definition 3.3. *A function $\kappa : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is symmetric function if for all $x, y \in \mathbb{R}^D$, we have $\kappa(x, y) = \kappa(y, x)$. A symmetric function κ is positive semi definite if for all $f \in L^2(\mathbb{R}^D)$ and we have*

$$\int \int_{\mathbb{R}^D \times \mathbb{R}^D} f(x)\kappa(x, y)f(y)dxdy \geq 0.. \tag{3.11}$$

Definition 3.4. *Let $\phi : \mathbb{R}^D \to \mathbb{R}^M$ be an embedding function. The kernel function $\kappa : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ of two vectors $x, y \in \mathbb{R}^D$, where D is defined to be the inner product of their features*

$$\kappa(x, y) = \phi(x)^T \phi(y) \in \mathbb{R}. \tag{3.12}$$

It can be shown that κ is a symmetric function that remains positive semidefinite with respect to both x and y [33].

3.7 Support vector machine

In supervised learning, SVMs are methods used to find patterns in data analysis. They find applications primarily in regression analysis, novelty detection, and classification tasks. The SVM training process creates a binary linear classifier that categorizes new data points into one of two classes using a hyperplane. This hyperplane is determined by a given set of training data in a binary classification scenario. The SVM model represents observations as points in space and divides them into separate partitions based on their functional margin, which is the greatest distance between any two points. When new observations are introduced, they are placed onto these partitions to identify their class. In the classification process, the support vectors play a crucial role as they represent the data points closest to the dividing hyperplane [34, 35]

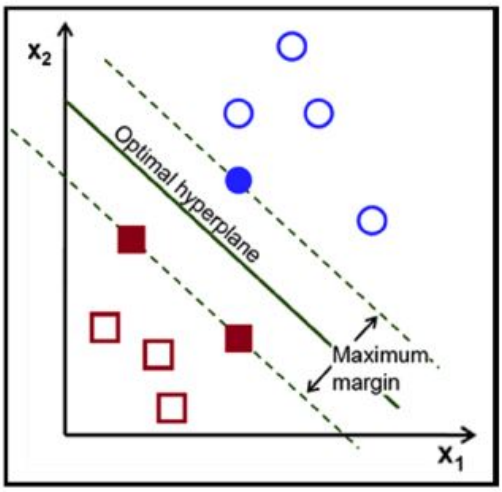

Figure 3.4: The SVM algorithm finds the hyperplane

In scenarios where both dimensionality reduction and classification are involved, SVMs have proven to be valuable assets.The effectiveness of SVM-driven dimensionality reduction is notably influenced by the selection of kernel function. In this study, we assess the performance of three frequently employed kernel functions Polynomial, Linear, and Gaussian (RBF) Kernel [35].

3.7.1 Kernel Functions

Kernels are an adaptable way to express your data samples so that you may compare them in a complex space.

3.7.1.1 Linear kernel

Example 3.5. *The Linear Kernel is utilized when the data demonstrates linear separability, indicating that it can be partitioned with a single straight line.*
The linear kernel assumes a linear relationship between input features. It is computationally efficient and suitable when the underlying data structure is linearly separable.
If we let $\phi(x) = x$, we derive the linear kernel by computing the dot product of the two vectors representing objects

$$K(x, y) = x^T y. \tag{3.13}$$

When using a linear KPCA, the method effectively reduces to PCA. In this scenario, the input space $\mathbb{X}$ and the feature space $\mathbb{B}$ are of the same size, as defined by $\Phi(x) = x$. A linear kernel is appropriate for data that can be separated by a single line, making it useful for datasets with a high number of features where a clear linear boundary exists. This kernel is commonly used in cases where data can be partitioned linearly. The linear kernel's effectiveness is demonstrated when a distinct linear separation within the input space is present. Overall, it is ideal for datasets with a clear linear boundary.
The linear kernel is advantageous for its simplicity and computational efficiency, but it is limited to linearly separable data. It is useful in dimensionality reduction by creating hyperplanes that optimally separate data points in the original feature space [35].

3.7.1.2 Polynomial Kernel

The polynomial kernel extends the linear kernel by introducing polynomial functions of higher degrees, allowing for the capture of more complex relationships [36]. Input data is mapped into a feature space over polynomials of the original variables using the polynomial kernel. One illustration of a non stationary kernel is this kernel function. The degree d, constant c, and slope α are all modifiable parameters.

$$K(x_i, x_j) = (\alpha {x_i}^T x_j + c)^d \tag{3.14}$$

Polynomial kernels are useful for handling nonlinear decision boundaries, with the parameter determining the polynomial's degree and being a constant. Higher polynomial degrees can model more complex relationships but may risk overfitting. The advantage of polynomial kernels lies in their ability to capture intricate, nonlinear relationships, while a major disadvantage is their susceptibility to overfitting and the critical importance of selecting the right polynomial degree. In dimensionality reduction, polynomial kernels offer a controlled approach to capturing nonlinear structures compared to radial basis function (RBF) kernels. They are particularly valuable when dealing with complex data patterns [36].

Example 3.6. *For the second-degree polynomial embedding, we obtain:*

$$\kappa(x, y) = [x_1^2, \sqrt{2}x_1x_2, x_2^2][y_1^2, \sqrt{2}y_1y_2, y_2^2]^T, \tag{3.15}$$

which can be computed directly in $\mathbb{R}^2$ without the necessity of computing the embedding into $\mathbb{R}^3$ [33].

3.7.1.3 Gaussian (RBF) Kernel

A higher-dimensional space is mapped onto input characteristics using the RBF kernel in order to capture nonlinear correlations. It manages complicated data structures well [37].The Gaussian kernel functions as the kernel for a radial basis function, altering input data into an infinite dimensional Hilbert space. The parameter that can be adjusted significantly influences the effectiveness of the

kernel. Overestimating the parameter can lead to a nearly linear response from the kernel, while underestimating it can result in insufficient regularization, making the function highly susceptible to noise.

$$K(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2}) \tag{3.16}$$

In kernel functions, $K(x, y)$ measures the similarity between two data points x and y in a high-dimensional space. For data points x_i and x_j, $\|x_i - x_j\|$ represents their Euclidean distance, while σ (the "bandwidth" or "width" parameter) controls the kernel's spread and affects the smoothness of the decision boundary.
The advantage of RBF kernels is their versatility in modeling complex relationships, but they are sensitive to the choice of the kernel width parameter (σ); they are useful in dimensionality reduction as they can capture intricate data patterns [37].

3.7.1.4 Exponential Kernel

The exponential kernel is another radial basis function kernel that bears resemblance to the Gaussian RBF. It computes the Euclidean norm directly rather than the squared Euclidean norm, in contrast to the Gaussian RBF.

$$K(x_i, x_j) = \exp(-\frac{||x_i - x_j||}{2\sigma^2}) \tag{3.17}$$

This kernel is not as standard as others. It might be used in cases where the relationship between data points exhibits exponential behavior.
An advantage is that it can capture exponential relationships in the data, but a disadvantage is that it may not perform well on datasets with different patterns.

3.7.1.5 Laplacian Kernel

Another example of a radial basis function kernel is the Laplacian Kernel, characterized by its diminished responsiveness to variations in the sigma parameter. Furthermore, it employs the Euclidean norm.

$$K(x_i, x_j) = \exp(-\frac{||x_i - x_j||}{\sigma}). \tag{3.18}$$

Like the Gaussian kernel, but shaped differently. It might work well with datasets that exhibit abrupt behavioral changes. The kernel's width is set by the σ parameter.
Advantage: Suitable for datasets with abrupt changes; disadvantage: similar sensitivity to the choice of the kernel width parameter.

3.7.1.6 Logarithmic Kernel

The logarithmic kernel is a form of kernel that exhibits conditional positive definiteness. Since its introduction in 2005, it has shown a great deal of success, especially in challenges involving SVM-based image identification. This kernel is a fascinating topic for more research because of its remarkable performance over traditional positive definite kernel alternatives

$$K(x_i, x_j) = -\log(||x_i - x_j||^d + 1). \tag{3.19}$$

Applicable when capturing logarithmic relationships between data points is important. Less common and might be used in specific situations [38].
The method can capture logarithmic relationships in data but is less commonly used and may not be suitable for all datasets.
Note:The choice of the kernel function in SVM-based dimensionality reduction should align with the underlying structure of the data. Linear kernels are efficient for linearly separable data, while RBF and Polynomial kernels offer flexibility for capturing nonlinear relationships. The decision should consider the trade-offs between computational efficiency and the ability to model complex data structures [34, 38] .
RBF kernels are extensively used in SVMs for classification and regression tasks, giving them an edge over other kernel types. Their capability to map the input space into a higher-dimensional feature space enables the separation of data points in a linear manner. This makes the RBF kernel the preferred option for handling nonlinearly separable data when compared to other alternatives. Alternatively known as the Gaussian kernel, the RBF kernel utilizes Gaussian distributions

around the data points to define the decision boundary.

The RBF kernel is a stationary kernel since it does not change when translations occur in the input space. The Gaussian kernel provides an adaptable technique for representing complex data relationships, and its value σ allows the smoothness of the decision border to be adjusted.
Following feature reduction, it's advisable to explore a different kernel if the resulting kernel matrix is diagonal, indicating redundancy in the feature space.
It should be noted that large amounts of memory and processing power are needed for the kernel matrix computation when using kernels to convert feature vectors for datasets that are linearly nonseparable.
After feature reduction, another kernel should be attempted if the kernel matrix turns out to be diagonal, which suggests that the feature space is redundant. It should be noted that large amounts of memory and processing power are needed for the kernel matrix computation when using kernels to convert feature vectors for datasets that are linearly nonseparable. The best value for σ often involves experimentation, cross-validation, and tuning to find the optimal setting for your specific problem.
For a given σ, the corresponding gamma in the RBF kernel is calculated as

$$gamma = \frac{1}{2\sigma^2}. \tag{3.20}$$

Therefore, a smaller σ corresponds to a larger gamma, and vice versa.

3.8 Impact of σ in the Gaussian (RBF) Kernel

Definition 3.7. *Gaussian kernels are summed for various σ values $K(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$ [39], where $K(x, y)$ represents the similarity between two data points x and y in a high-dimensional space, x_i and x_j are two data points for which you want to compute the kernel function value,*

and σ the parameter that influences how wide the kernel spreads is often referred to as the "bandwidth" or "width" parameter.
The parameter σ is pivotal in shaping how the Gaussian (RBF) kernel behaves, thereby impacting the effectiveness of the kernelized algorithm.

The range of σ values typically depends on the dataset's attributes and the application context. While σ can span from very small (e.g., 10^{-3}) to relatively large (e.g., 10^{3}) values, the suitable range should be determined through experimentation. Extreme values of σ should be avoided to prevent underfitting or overfitting. Employing cross-validation techniques aids in identifying the most fitting σ value within a predefined range for a given problem [40].

3.9 Prediction of data with confusion matrix

Using the data arranged in a matrix, a visual representation is created to assess the effectiveness of the classification method. This chart provides information about incorrect positive identifications, correct positive identifications, incorrect negative identifications, and correct negative identifications, facilitating a comparison of anticipated and real classifications. Specifically, the TP and FP information within the confusion matrix offers a more precise evaluation of the classifier's performance. In unsupervised learning, the confusion matrix is alternatively termed the matching matrix, while in disciplines beyond machine learning, it may be recognized as the error matrix or contingency matrix. Despite its various names, the confusion matrix serves to visually demonstrate the accuracies of both the predicted and actual classes.
A confusion matrix appears as:

Predicted/Actual class	Positive class	Negative class
Positive class	TP	FP
Negative class	FN	TN

Table 3.1: Confusion Matrix

Accuracy refers to the quantity of data points accurately categorized by the classification algorithm:

$$accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{3.21}$$

Typically, the minority class, often the positive class, holds greater significance to the designer. The count of data points accurately identified as part of the positive class is referred to as true positives (TP) or recall in certain sectors. False positives (FP) denote data points wrongly classified as positive but actually belong to the negative class. True negatives (TN) signify the count of data points accurately labeled as negative. False negatives (FN) denote the instances where data points expected to belong to the positive class are incorrectly classified as negative. Sensitivity, sometimes known as true positive rate (TPR) or recall rate (RR), measures a classification algorithm's ability to accurately identify data points [34]

$$Sensitivity(or\ recall) = \frac{TP}{TP + FN}, \tag{3.22}$$

$$Precision = \frac{TP}{TP + FP}. \tag{3.23}$$

Specificity, or the true negative rate, acts as a measure of how well the classification algorithm can correctly identify data points belonging to the negative class [34]

$$Specificity = \frac{TN}{TN + FP}\ and\ F1Score := \frac{2 * (Precision * Recall)}{(Precision + Recall)}. \tag{3.24}$$

Chapter 4

Application to data classifications

4.1 Dataset description

The study utilized secondary data, consisting of 14 variables such as Hue OD280, Proline, Alcohol,... and Customer Segment. Due to the limitations of data profiling methods in Python, the full dataset isn't easily visible, but PCA allows for viewing the data in a two-dimensional format and identifying its most significant components. Table 4.1 provides descriptive and quintile statistics for each variable. To assess the data effectively and achieve the study's objectives, Python commands and procedures were used to conduct exploratory data analysis, revealing how nonlinear PCA minimizes information loss through dimensionality reduction. Nonlinear PCA algorithms uncover complex relationships and patterns that linear PCA might miss, making them valuable for analyzing the dataset.
Table 4.1 displays the first three and last two datasets for each of the fourteen variables

Alcohol	Malic Acid	Ash	Ash Alcanity	Magnesium	Total Phenols	Flavonoids	Nonflavanoid Phenols	Proanthocyanins	Color Intensity	Hue	OD280	Proline	Customer Segment
14.23	1.71	2.43	15.6	127	2.8	3.06	0.28	2.29	5.64	1.04	3.92	1065	1
13.2	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.4	1050	1
13.16	2.36	2.67	18.6	101	2.8	3.24	0.3	2.81	5.68	1.03	3.17	1185	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.3	0.6	1.62	840	3
14.13	4.1	2.74	24.5	96	2.05	0.76	0.56	1.35	9.2	0.61	1.6	560	3

Table 4.1: Dataset with various chemical attributes

Before using a machine learning algorithm, PCA must be used.
In this data-set, we should apply it to the features rather than the goal value

"Customer Segment". Using sklearn's Standart Scaler, we must first normalize the variables before applying PCA to the data. Our data was converted into a numpy array using Standard Scaler, which also normalized all of the data's variables. Nonlinear methods like kernel PCA and autoencoders seek to preserve the complex arrangement of data in high-dimensional spaces without resorting to linear transformations for dimensionality reduction. Their goal is to minimize the loss of information while reducing dimensionality.

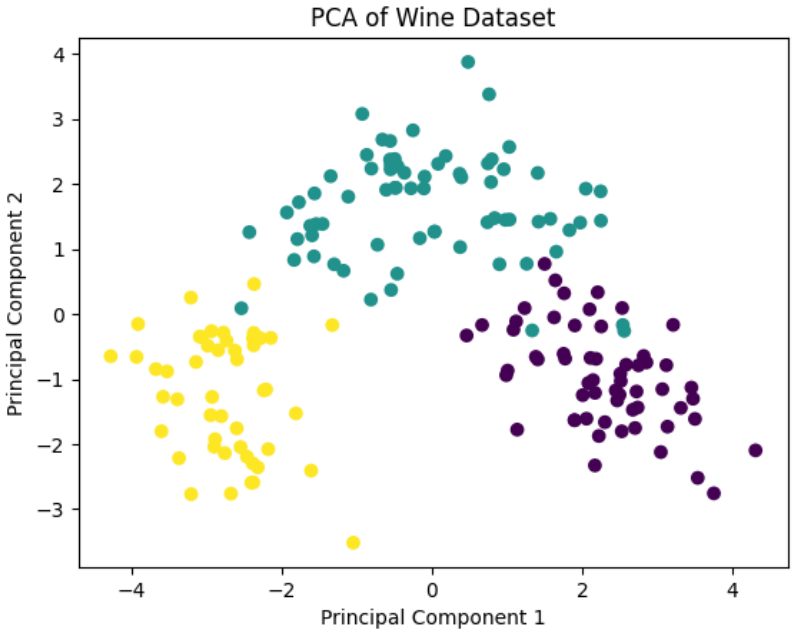

Figure 4.1: PCA of Wine Dataset

Patterns in Figure 4.1 above reveal the arrangement and dispersion of data following dimensionality reduction, potentially highlighting clusters or trends within the dataset. The "plasma" colormap, which is perceptually uniform, is used to color the points, helping to visualize variations in data density or patterns.

Using the formulas from Table 3.1 (Confusion Matrix) and equation (3.24), we derive the following results, presented in the Table 4.2.

	0	1	2	3	4
0	0.127532	-0.256237	-0.022362	-0.244281	0.147578
1	-0.492805	-0.190446	-0.316764	0.035139	-0.262172

Table 4.2: Data distribution

Each row in the data represents the principal components, while each column corresponds to the original features, which can be better visualized using Figure 4.2 (a heatmap). Transforming this data into a dataframe allows for the creation of a scatter plot that shows the distribution of points in the reduced-dimensional space, with the colormap and axis labels helping to interpret patterns and the nature of the principal components.
The key elements are actually the combinations of all these data attributes.

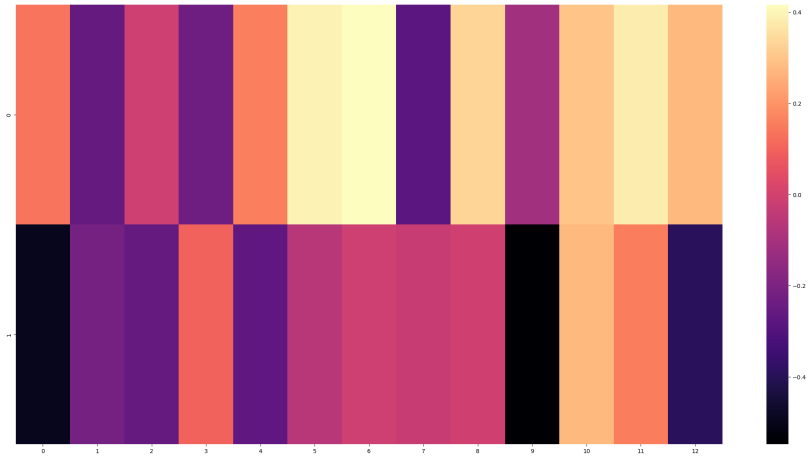

Figure 4.2: Heatmap

The Figure 4.2 (heatmap) displays the relationship between principal components and original features, where light colors represent strong correlations and dark colors indicate weaker or negative correlations. To construct matrix A, the data is organized with rows representing individual data points and columns for features,

including a final column for the target variable, 'Customer Segment', with each row corresponding to a wine sample and each column to a specific feature. The matrix A is shown below:

$$A = \begin{bmatrix} 14.23 & 1.71 & 2.43 & 15.6 & 127 & 2.8 & 3.06 & 0.28 & 2.29 & 5.64 & 1.04 & 3.92 & 1065 \\ 13.2 & 1.78 & 2.14 & 11.2 & 100 & 2.65 & 2.76 & 0.26 & 1.28 & 4.38 & 1.05 & 3.4 & 1050 \\ 13.16 & 2.36 & 2.67 & 18.6 & 101 & 2.8 & 3.24 & 0.3 & 2.81 & 5.68 & 1.03 & 3.17 & 1185 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 13.17 & 2.59 & 2.37 & 20.0 & 120 & 1.65 & 0.68 & 0.53 & 1.46 & 9.3 & 0.6 & 1.62 & 840 \\ 14.13 & 4.1 & 2.74 & 24.5 & 96 & 2.05 & 0.76 & 0.56 & 1.35 & 9.2 & 0.61 & 1.6 & 560 \end{bmatrix}$$

The condition number of matrix A can be expressed using this formula $cond(A) = ||A||_2.||A^+||_2$. The 2-norm of A is $||A||_2 = 9758.8154379277$.
$A^+ = (A^T A)^{-1} A^T$ yields the pseudo-inverse of matrix $A.A^T$, yielding the pseudo-inverse matrix A^+'s 2-norm, which is $||A^+||_2 = 1.6652765518260193$.
$Cond(A) = ||A||_2.||A^+||_2 = 16251.126522378765$ is the result, which is the condition number for A.

The condition number acts as an indicator of the sensitivity of matrix A. A higher condition number implies greater sensitivity, meaning that even minor changes in the input (matrix coefficients) could have a notable effect on the output (solution to a linear system), potentially resulting in an excessively large condition number. Evaluating the sensitivity of eigenvectors is intricate as it relies on both the sensitivity of the associated eigenvalue and the separation between eigenvalues. Eigenvalues that are well-conditioned and well-separated lead to well-conditioned eigenvectors, while poorly conditioned or closely clustered eigenvalues may produce eigenvectors of inferior quality [23].

Thus, the matrix is deemed highly sensitive to perturbations due to $cond(A) = 16251.126522378765 > 1$, implying that minor changes in its coefficients can disproportionately affect the linear system solution. Since our dataset exhibits ill conditioning, we intend to perform PCA to ascertain its linear separability. Should the data prove nonlinearly separable, Kernel PCA would be employed to linearize it.

4.2 The principal Component Analysis

PCA is a dimensionality reduction technique aimed at identifying principal components that capture the most variability in high-dimensional data. Given a dataset $X = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^{n \times d}$, where n represents the number of samples and d denotes the number of features, PCA projects the data into a lower-dimensional space while retaining most of its variability. The first principal component, or primary axis, shows the direction of the highest variability, while the second principal component, or secondary axis, captures the next highest variability orthogonal to the first. These principal components provide a clearer representation of the data, which can then be used in machine learning algorithms.

In this study, logistic regression is performed on the transformed data rather than on the original dataset. The model's performance is assessed using a confusion matrix, which details the types of classification errors and evaluates the model's overall effectiveness. Check out the following confusion matrix:

$$A = \begin{bmatrix} 17 & 2 & 0 \\ 0 & 21 & 0 \\ 0 & 0 & 14 \end{bmatrix}$$

The confusion matrix thus makes it easier to evaluate the F1-score, accuracy, precision, and recall of the model.
It's a valuable tool for assessing classification performance as there are shown in Table 4.3 by using the formulas from (3.24), (3.25) , (3.26) and (3.27).

	precision	recall	f1-score	support
1	1.00	0.89	0.94	19
2	0.91	1.00	0.95	21
3	1.00	1.00	1.00	14
accuracy			0.6	54
macro avg	0.97	0.96	0.97	54
weighted avg	0.97	0.96	0.96	54

Table 4.3: PCA classification report

We calculated the condition number of matrix A with respect to the 2-norm in Python using NumPy, resulting in $cond(A) = 1.508$. Support vector machines could be a viable alternative for analyzing this data. We use fundamental elements of the data for machine learning algorithms, focusing on simple, comprehensible inputs. Instead of using the entire dataset, logistic regression is performed with the adjusted data. Based on feedback from 178 consumers, logistic regression analysis reveals three distinct consumer clusters, with the customer segment as the dependent variable and the remaining 13 variables as independent variables.

Using only the two main components of the data can often yield a relatively good forecast compared to using the entire dataset. PCA is particularly useful for handling large datasets by reducing their dimensionality. A scatter plot visualizes the results of a logistic regression on a subset of data, divided into three distinct sections with colors green, black, and red each representing different classes. PCA is employed to plot the data points based on their PC scores, aiming to simplify and reduce the data's dimensions. The legend on the Figure 4.3 shows that the color coded areas correspond to correct classifications of data points, with decision boundaries separating each class into triangular sections.

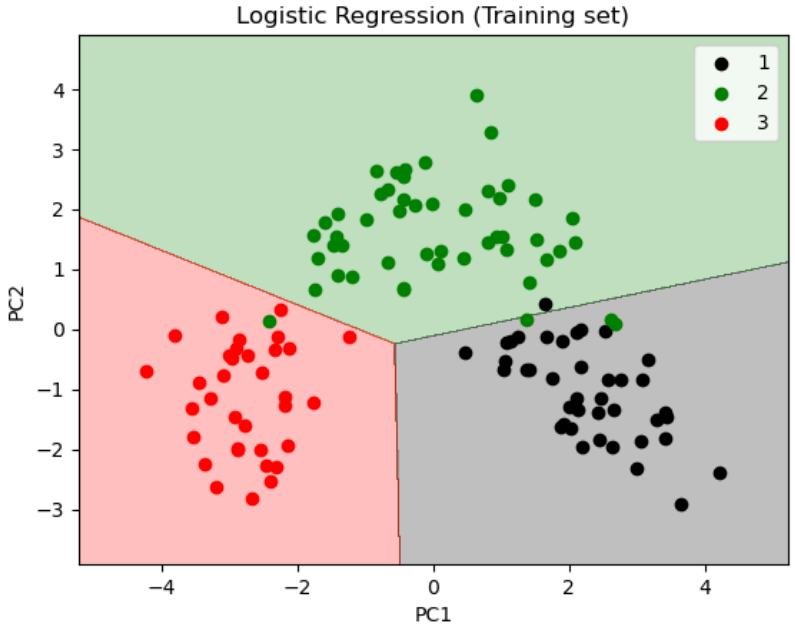

Figure 4.3: PCA of Training Set

The goal of PCA is to simplify the data by reducing its dimensionality. The legend on the right side of the Figure 4.4 clarifies that green, purple, and grey points correspond to different classes. The green section shows correctly classified data points in green, the purple section shows purple points, and the grey section shows correctly classified grey points. Decision boundaries, forming triangular sections, effectively separate these classes.

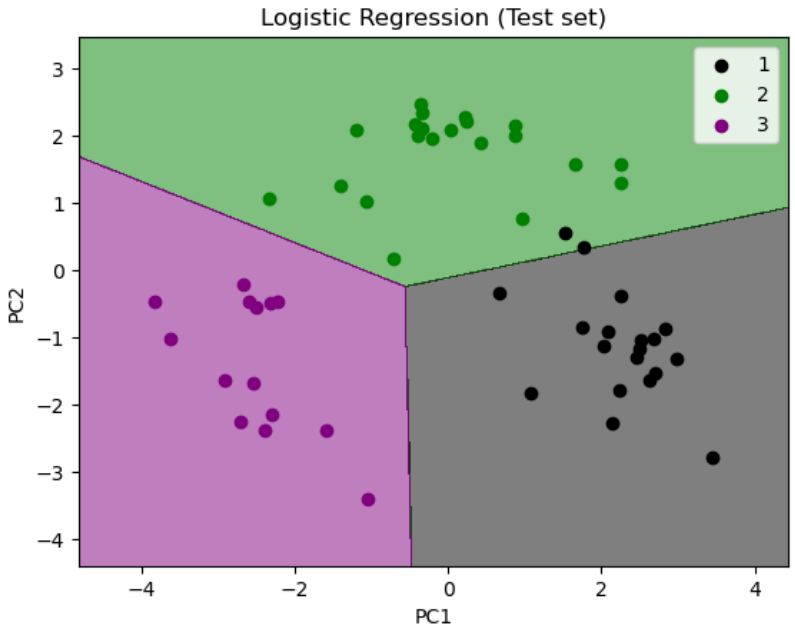

Figure 4.4: PCA of Wine Test Set

4.3 Linear Discriminant Analysis

LDA is a supervised dimensionality reduction technique focused on finding a feature space that optimizes class separation. Given a dataset $x = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^{n \times d}$ with class labels $y = [y_1, y_2, \ldots, y_n]^T$, where each y_i represents one of c classes, LDA aims to identify axes that maximize class separation. Unlike PCA, which prioritizes axes that maximize data variance without considering class labels, LDA specifically enhances class distinction. Both PCA and LDA are used for linear dimensionality reduction, but LDA is supervised and focuses on creating "linear discriminants"

that improve class separability. Visualizations of the training and test datasets show that the classifier's decision boundary is linear.

See the confusion matrix below:

$$B = \begin{bmatrix} 17 & 2 & 0 \\ 0 & 21 & 0 \\ 0 & 0 & 14 \end{bmatrix}$$

The Table 4.4 summarizing all:

	precision	recall	f1-score	support
1	1.00	0.89	0.94	19
2	0.91	1.00	0.95	21
3	1.00	1.00	1.00	14
accuracy			0.96	54
macro avg	0.97	0.96	0.97	54
weighted avg	0.97	0.96	0.96	54

Table 4.4: LDA Classification report

From (3.21) and we get the accuracy of 1.0.

We obtain $cond(B) = 1.5187$ by computing the condition number of matrix B with regard to the 2-norm in Python.

Figure 4.5 illustrates the contrast between PCA and LDA [41, 31].
Two frequently used techniques for reducing dimensionality using linear transformations include LDA and PCA. Since principle components, or the directions that maximize variance in a dataset, are "ignored," PCA can be referred to as a "unsupervised" technique.

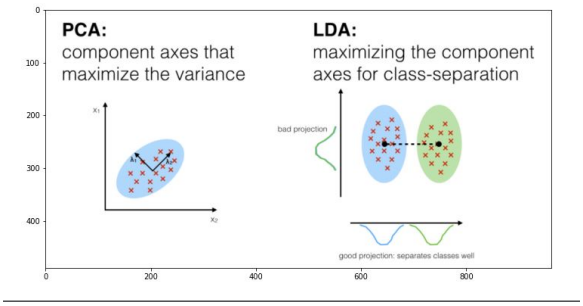

Figure 4.5: PCA vs LDA

Comparatively speaking to PCA, LDA is "supervised" and finds "linear discriminants," or the directions corresponding to the axes that optimize the degree of separation between several groups.

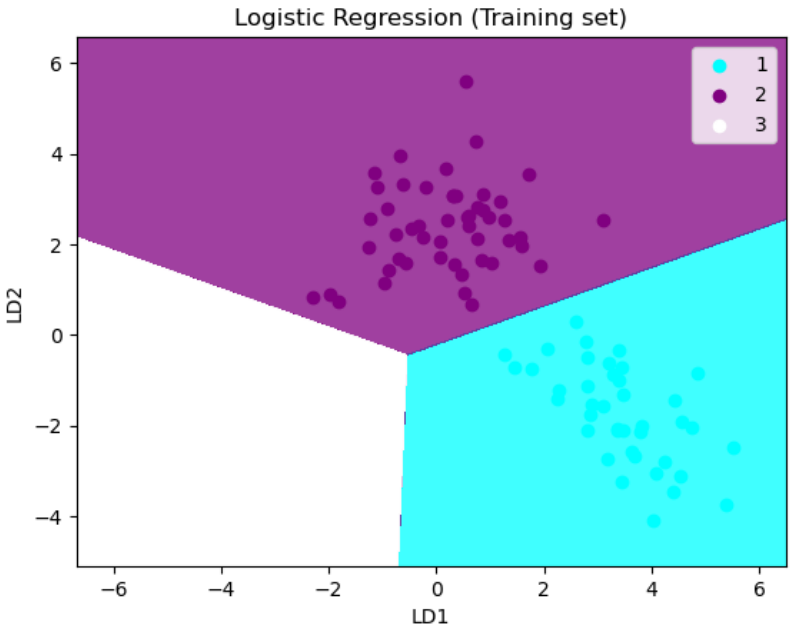

Figure 4.6: LDA of Wine Training Set

Logistic Regression (Training set) indicates that the Figure 4.6 represents a logistic regression model trained on a dataset. The graph features three distinct zones, each corresponding to a separate class (1, 2, or 3). Two features, LD1 and LD2, determine the positions of the plotted data points, which represent individual samples. Points are color-coded based on their class: blue for class 1, purple for class 2, and no color for the third class. In the Figure 4.6, blue data points are clustered in the top left corner, while purple data points are grouped at the bottom center. The legend in the bottom left confirms that blue corresponds to class 1 and purple to class 2, with no visible points for the third class. The plot's background is divided into three colored zones blue, purple, and white indicating the decision boundaries for each class.

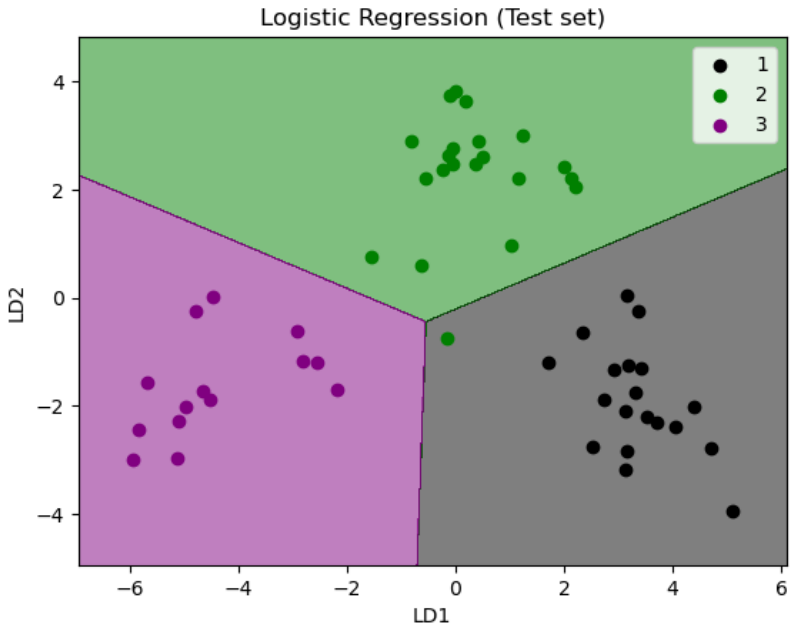

Figure 4.7: LDA of Wine Test Set

The outcomes of Logistic Regression on a test set are illustrated in Figure 4.7. Ultimately, LDA outperforms PCA in the training set, yet both exhibit similar accuracy in the test set. This implies that LDA excels in predicting the test set's outcome compared to PCA.

4.4 Kernal PCA

KPCA extends the principles of PCA by using the kernel method to map data into a higher-dimensional space, thereby enhancing linear separability. In this transformed space, PCA is then applied to identify principal components. Given a dataset $X = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^{n \times d}$, kernel methods aim to increase dimensionality, while SVD aims to decrease it. Comparison shows that KPCA produces eigenvectors with higher variance (eigenvalues) than PCA. This variance difference is attributed to KPCA's ability to project data into a circular shape, whereas PCA results in a linear shape, capturing more variation. Consequently, KPCA generates more principal components than PCA. While PCA focuses on identifying a lower-dimensional linear subspace, KPCA can detect nearly one-dimensional data and uncover nonlinear manifolds. KPCA, a nonlinear approach building on PCA principles, effectively handles complex relationships and nonlinear separability within datasets. Applying KPCA to a dataset with 14 wine quality variables resulted in effective dimensionality reduction. This demonstrates KPCA's utility in revealing intricate structures that traditional PCA might miss. See the confusion matrix below:

$$C = \begin{bmatrix} 19 & 0 & 0 \\ 0 & 21 & 0 \\ 0 & 0 & 14 \end{bmatrix}$$

	precision	recall	f1-score	support
1	1.00	1.00	1.00	19
2	1.00	1.00	1.00	21
3	1.00	1.00	1.00	14
accuracy			1.00	54
macro avg	1.00	1.00	1.00	54
weighted avg	1.00	1.00	1.00	54

Table 4.5: KPCA Classification report

An F1-score of 1 indicates perfect precision and recall, showing that the model's predictions perfectly match the actual outcomes. A condition number near 1 for matrix C implies robust solutions unaffected by small data fluctuations. Using Python and NumPy, the condition number of C relative to the 2-norm was calculated as $\text{cond}_2 C = 1.428$. This low condition number indicates that matrix C, used for KPCA, has the best conditioning and is less sensitive to input data changes compared to matrices A (PCA) and B (LDA) [31].

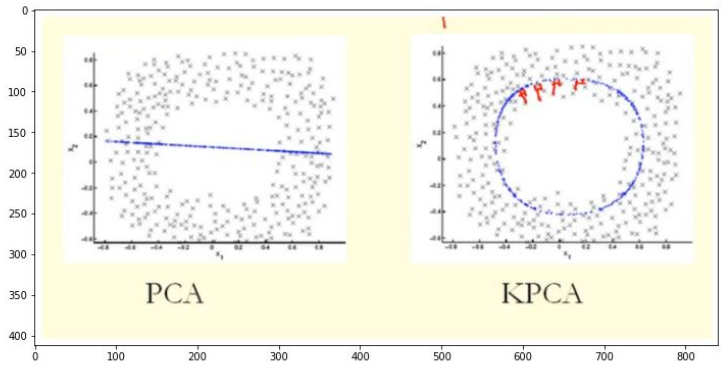

Figure 4.8: PCA vs Kernel PCA

Understanding KPCA involves recognizing that its principal components are combinations of original variables that capture most of the data's variance. These components are computed in a higher-dimensional space using a kernel function, allowing for the transformation of nonlinearly separable data into linearly separable data. We demonstrate this dimensionality reduction by presenting results from both training and test sets using the same dataset. To determine if the dataset is linearly separable, the initial step is to train a logistic regression model; effective performance would indicate near-linear separability. For RBF kernels, finding the optimal value of sigma (σ) is crucial, often achieved through trial and error or cross-validation. Cross-validation helps in identifying the best σ value, which can vary depending on the specific problem and dataset. This technique is useful not only for RBF interpolation but also for solving partial differential equations with RBF pseudo-spectral methods. Finally, it is noted that all dimensionality analysis

tools used in this study achieved 100% accuracy [42].

Utilizing the kernel trick (with the kernel function: $\kappa(x, y) = \phi(x)^T \phi(y) \in \mathbb{R}$), KPCA emerges as a method for nonlinear dimensionality reduction, achieved by mapping data into a higher-dimensional space.

This approach enables the discovery of complex relationships that linear techniques may fail to capture. The selection of the kernel function (in this example, the RBF) and its associated parameters can have a substantial impact on the outcome. Consequently, it may be necessary to adjust these parameters in accordance with the unique attributes of the dataset. Through the utilization of equation $(K(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2}))$, it becomes possible to obtain Figure 4.9.

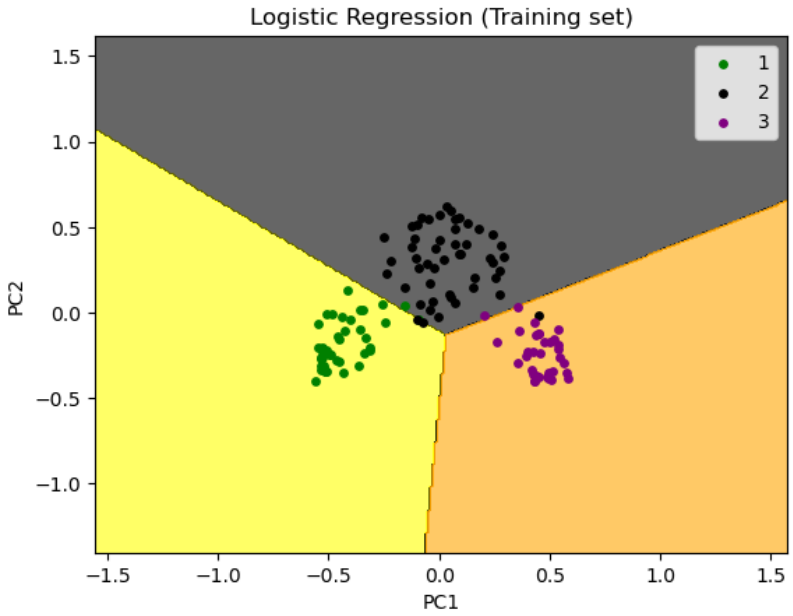

Figure 4.9: Kernel PCA of Wine Training Set

This picture shows a scatter plot of a logistic regression model, which predicts outcomes by analyzing data. The x-axis represents the PC1 and the y-axis represents the PC2, both derived from the original dataset. These principal components are linear combinations of the original variables, capturing the variance within the data. The Figure 4.9 includes data points in green, black, and purple, each representing a different class. The model predicts customer groupings based on

PC1 and PC2 values, derived from the preferences of 178 consumers who rated wines between 1 and 3.

Decision boundaries are depicted, separating the background into three zones, each corresponding to a class. These boundaries are linear functions of PC1 and PC2, indicating the highest probability of class membership. To evaluate the model's accuracy, predicted results on the test set are compared with observed outcomes. Performance is assessed using a confusion matrix and a classification report. This approach helps visualize the effectiveness of the logistic regression model in classifying data based on principal components.
In machine learning and statistics, a confusion matrix is a basic tool for assessing a classification algorithm's efficacy. It gives an overview of how well the actual class labels in a classification task match the predictions made by a model.
For binary or multi-class classification tasks, the confusion matrix is particularly helpful.

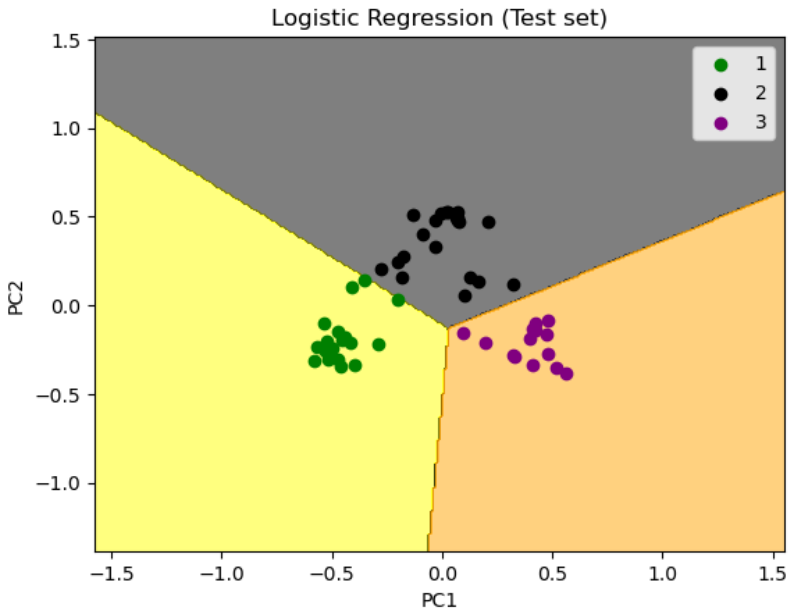

Figure 4.10: Kernel PCA of Wine Test Set

Grid search combined with cross-validation effectively identifies the optimal σ parameter for the RBF kernel when analyzing the 14-variable wine dataset. This

method enhances the model's performance on the validation set by selecting the most suitable sigma value. The improved classification performance in Figure 4.11 (Logistic test set) suggests better model alignment with the data. This improvement is likely due to factors such as refined feature selection, optimized hyperparameters, or the use of a more appropriate kernel function in kernel logistic regression.

The clear clustering in Figure 4.11 shows that the logistic regression model has successfully classified data points into the appropriate groups with a high degree of accuracy.
By analyzing Figures 4.4 and 4.5 from the literature on PCA, it becomes apparent that there are certain instances where data points form clusters that lack a shared data interpretation. This suggests that the PCA dataset is not accurately predicted. However, in the scenario of KPCA for linear regression (test set), each dataset with corresponding colors is effectively grouped together within a single cluster, indicating a successful prediction.

The comparison shows that KPCA achieves higher variance (eigenvalue) in its eigenvector than traditional PCA, as KPCA produces a circular projection, while PCA results in a straight line. KPCA identifies higher principal components by mapping data into a higher-dimensional space and then finding a lower-dimensional subspace within it. Unlike PCA, which finds a linear subspace, KPCA can uncover nonlinear structures, revealing that the data may be nearly one-dimensional. This approach increases dimensionality temporarily to better capture and reduce the data's complexity.

Chapter 5

Conclusion and Recommendation

This thesis has successfully demonstrated the effectiveness of NPCA techniques in enhancing data analysis across in various fields, including manufacturing, engineering, biomedical research, finance, and environmental science. By integrating kernel functions, the research reveals that nonlinear PCA can uncover complex relationships within datasets that traditional linear PCA may overlook. The ability of nonlinear PCA to manage nonlinearly separable data and identify intricate patterns is a significant advancement in the field of dimensionality reduction.
The findings indicate that nonlinear PCA not only preserves essential data relationships during the dimensionality reduction process but also improves the stability of the system, as evidenced by a condition number of 1.42 for the analyzed wine dataset. This condition number, being closer to 1, suggests a more stable and reliable model compared to traditional PCA and LDA. The classification of data points into three distinct clusters further illustrates the method's capability to effectively segment data based on the target variable.
While this research lays a solid foundation for understanding and applying NPCA, it also highlights the need for further exploration. Future studies should focus on refining the mathematical stability of nonlinear PCA techniques to achieve even lower condition numbers, thereby enhancing performance and accuracy.

References

[1] Roman Rosipal and Leonard J Trejo. Kernel partial least squares regression in reproducing kernel hilbert space. *Journal of machine learning research*, 2(Dec):97–123, 2001.

[2] Mohammad Ihsan, Mohammad Nisar, Nausheen Nazir, Muhammad Zahoor, Atif Ali Khan Khalil, Abdul Ghafoor, Arshad Khan, Ramzi A Mothana, Riaz Ullah, and Nisar Ahmad. Genetic diversity in nutritional composition of oat (avena sativa l.) germplasm reported from pakistan. *Saudi Journal of Biological Sciences*, 29(3):1487–1500, 2022.

[3] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.

[4] Matthias Scholz, Fatma Kaplan, Charles L Guy, Joachim Kopka, and Joachim Selbig. Non-linear pca: a missing data approach. *Bioinformatics*, 21(20):3887–3895, 2005.

[5] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[6] Daniel Gedon, Antônio H Ribeiro, Niklas Wahlström, and Thomas B Schön. Invertible kernel pca with random fourier features. *IEEE Signal Processing Letters*, 2023.

[7] Fredrik Hallgren. Kernel pca and the nyström method, 2022.

[8] Marco Avella-Medina, Richard A Davis, and Gennady Samorodnitsky. Kernel pca for multivariate extremes. *arXiv preprint arXiv:2211.13172*, 2022.

[9] Faisal Shahzad, Zhensheng Huang, and Waqar Hussain Memon. Process monitoring using kernel pca and kernel density estimation-based ssglr method for nonlinear fault detection. *Applied Sciences*, 12(6):2981, 2022.

[10] Yichuan Deng, Zhao Song, Zifan Wang, and Han Zhang. Streaming kernel pca algorithm with small space. *arXiv preprint arXiv:2303.04555*, 2023.

[11] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[12] Jirat Bhanpato, Ameya Behere, and Dimitri N Mavris. Non-linear dimensionality reduction techniques for model order reduction of aviation noise metrics. In *AIAA AVIATION 2023 Forum*, page 4061, 2023.

[13] Oskar Allerbo and Rebecka Jörnsten. Non-linear, sparse dimensionality reduction via path lasso penalized autoencoders. *The Journal of Machine Learning Research*, 22(1):12978–13005, 2021.

[14] Basna Mohammed Salih Hasan and Adnan Mohsin Abdulazeez. A review of principal component analysis algorithm for dimensionality reduction. *Journal of Soft Computing and Data Mining*, 2(1):20–30, 2021.

[15] Rou Zhong, Chunming Zhang, and Jingxiao Zhang. Nonlinear functional principal component analysis using neural networks. *arXiv preprint arXiv:2306.14388*, 2023.

[16] Shaojie Xu, Joel Vaughan, Jie Chen, Agus Sudjianto, and Vijayan Nair. Supervised linear dimension-reduction methods: Review, extensions, and comparisons. *arXiv preprint arXiv:2109.04244*, 2021.

[17] Omprakash Saini and Sumit Sharma. A review on dimension reduction techniques in data mining. *Comput. Eng. Intell. Syst*, 9(1):7–14, 2018.

[18] Qiuwei Li, Zhihui Zhu, Si Tang, Liping Chang, and Gang Li. Projection matrix optimization based on svd for compressive sensing systems. In *Proceedings of the 32nd Chinese Control Conference*, pages 4820–4825. IEEE, 2013.

[19] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[20] Sudeep Tanwar, Tilak Ramani, and Sudhanshu Tyagi. Dimensionality reduction using pca and svd in big data: A comparative case study. In *Future Internet Technologies and Trends: First International Conference, ICFITT 2017, Surat, India, August 31-September 2, 2017, Proceedings 1*, pages 116–125. Springer, 2018.

[21] Sami Romdhani, Shaogang Gong, Alexandra Psarrou, et al. A multi-view nonlinear active shape model using kernel pca. In *BMVC*, volume 10, pages 483–492, 1999.

[22] João BO Souza Filho and Paulo SR Diniz. A fixed-point online kernel principal component extraction algorithm. *IEEE Transactions on Signal Processing*, 65(23):6244–6259, 2017.

[23] Michael T Heath. *Scientific computing: an introductory survey, revised second edition*. SIAM, 2018.

[24] Gilbert Strang. *Introduction to linear algebra*. SIAM, 2022.

[25] Michael E Wall, Andreas Rechtsteiner, and Luis M Rocha. Singular value decomposition and principal component analysis. *A practical approach to microarray data analysis*, pages 91–109, 2003.

[26] Justin Wang, Raymond KW Wong, and Thomas CM Lee. Locally linear embedding with additive noise. *Pattern recognition letters*, 123:47–52, 2019.

[27] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pages 353–374, 2023.

[28] Seung Hwan Hong, Seongok Ryu, Jaechang Lim, and Woo Youn Kim. Molecular generative model based on an adversarially regularized autoencoder. *Journal of chemical information and modeling*, 60(1):29–36, 2019.

[29] David Charte, Francisco Charte, María J del Jesus, and Francisco Herrera. A showcase of the use of autoencoders in feature learning applications. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pages 412–421. Springer, 2019.

[30] Terry L Gillum, Robert H George, and Jack E Leitmeyer. An autoencoder for clinical and regulatory data processing. *Drug Information Journal*, 29(1):107–113, 1995.

[31] Sebastian Raschka and Locally-Linear Embedding LLE. Kernel tricks and nonlinear dimensionality reduction via rbf kernel pca.

[32] Xiaoyu Luo. Efficient english text classification using selected machine learning techniques. *Alexandria Engineering Journal*, 60(3):3401–3409, 2021.

[33] René Vidal, Yi Ma, S Shankar Sastry, René Vidal, Yi Ma, and S Shankar Sastry. Robust principal component analysis. *Generalized Principal Component Analysis*, pages 63–122, 2016.

[34] Mariette Awad and Rahul Khanna. *Efficient learning machines: theories, concepts, and applications for engineers and system designers*. Springer nature, 2015.

[35] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[36] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[37] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[38] Sayed Fadel, Said Ghoniemy, Mohamed Abdallah, Hussein Abu Sorra, Amira Ashour, and Asif Ansary. Investigating the effect of different kernel functions on the performance of svm for recognizing arabic characters. *Int. J. Adv. Comput. Sci. Appl*, 7(1):446–450, 2016.

[39] Nabil Benoudjit and Michel Verleysen. On the kernel widths in radial-basis function networks. *Neural Processing Letters*, 18:139–154, 2003.

[40] Olivier Chapelle, Patrick Haffner, and Vladimir N Vapnik. Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, 10(5):1055–1064, 1999.

[41] Alaa Tharwat, Tarek Gaber, Abdelhameed Ibrahim, and Aboul Ella Hassanien. Linear discriminant analysis: A detailed tutorial. *AI communications*, 30(2):169–190, 2017.

[42] Zuzana Majdisova and Vaclav Skala. Radial basis function approximations: comparison and applications. *Applied Mathematical Modelling*, 51:728–743, 2017.