Development of Novel Biclustering Algorithms and a Non-Linear Feature Relationship Index for Pattern Recognition Tasks

Thesis submitted by

Namita Jain

Doctor of Philosophy (Engineering)

Department of Computer Science and Engineering
Faculty Council of Engineering & Technology
Jadavpur University
Kolkata, India

2025

JADAVPUR UNIVERSITY
Kolkata-700032, India

INDEX-NO. 24/18/E

1. Title of the Thesis: **Development of Novel Biclustering Algorithms and a Non-Linear Feature Relationship Index for Pattern Recognition Tasks**

2. Name, Designation and Institution of the Supervisor:
Dr. Susmita Ghosh
Professor
Department of Computer Science & Engineering
Jadavpur University
Kolkata - 700 032

List of Publications

List of Journal papers

1. N. Jain, S. Ghosh, and A. Ghosh, "A parameter free relative density based biclustering method for identifying non-linear feature relations," *Heliyon*, vol. 10, p. e34736, 2024

2. N. Jain and C. A. Murthy, "Connectedness-based subspace clustering," *Knowledge and Information Systems*, vol. 58, no. 1, pp. 9–34, 2019

3. N. Jain and C. A. Murthy, "A new estimate of mutual information based measure of dependence between two variables: properties and fast implementation," *International Journal of Machine Learning and Cybernetics*, vol. 7, pp. 857–875, 2016

List of Conference papers

1. N. Jain and S. Ghosh, "An unsupervised band selection method for hyperspectral images using mutual information based dependence index," in *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium, Kuala Lumpur, Malaysia*, pp. 783–786, 2022

2. N. Jain and S. Ghosh, "Machine Learning Method to Discover the Novel Association of Vaccine and Vitamin A Supplement on Covid 19 Infection: A Biclustering Approach," in *2021 IEEE 18th India Council International Conference (INDICON), Guwahati, India*, pp. 1–6, 2021

Pre-print

1. N. Jain, S. Ghosh, and C. A. Murthy, "RelDenClu: A Relative Density based Biclustering Method for identifying non-linear feature relations," *arXiv:Computer Vision and Pattern Recognition, Preprint, arXiv:1811.04661*, 2021

List of Presentations in National/International Conferences

1. N. Jain and S. Ghosh, "An unsupervised band selection method for hyperspectral images using mutual information based dependence index," in *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium, Kuala Lumpur, Malaysia*, pp. 783–786, 2022

2. N. Jain and S. Ghosh, "Machine Learning Method to Discover the Novel Association of Vaccine and Vitamin A Supplement on Covid 19 Infection: A Biclustering Approach," in *2021 IEEE 18th India Council International Conference (INDICON), Guwahati, India*, pp. 1–6, 2021

Statement of Originality

I Namita Jain, registered on 8^{th} June, 2018, do hereby declare that this thesis entitled "Development of Novel Biclustering Algorithms and a Non-Linear Feature Relationship Index for Pattern Recognition Tasks" contains literature survey and original research work done by the undersigned candidate as part of doctoral studies.

All information in this thesis have been obtained and presented in accordance with existing academic rules and ethical conduct. I declare that, as required by these rules and conduct, I have fully cited and referred all materials and results that are not original to this work.

I also declare that I have checked this thesis as per the "Policy on Anti Plagiarism, Jadavpur University, 2019", and the level of similarity as checked by iThenticate software is 5%.

Signature of Candidate: Namita Jain
Date : 04/06/2025

Susmita Ghosh 04.06.2025
Certified by Supervisor:
(Signature with date, seal)

Professor
Computer Sc. & Engg. Department
Jadavpur University
Kata-700032

CERTIFICATE FROM THE SUPERVISORS

This is to certify that the thesis entitled "Development of Novel Biclustering Algorithms and a Non-Linear Feature Relationship Index for Pattern Recognition Tasks" submitted by Namita Jain, who got her name registered on 8^{th} June, 2018 for the award of Ph.D. (Engineering) degree of Jadavpur University is absolutely based upon her own work under the supervision of Prof. Susmita Ghosh, Department of Computer Science & Engineering, Jadavpur University, Kolkata and that neither her thesis nor any part of the thesis has been submitted for any degree/diploma or any other academic award anywhere before.

Susmita Ghosh 04.06.2025

Professor
Computer Sc. & Engg. Department
Jadavpur University
Kolkata-700032

Dr. Susmita Ghosh
Professor
Department of Computer Science & Engineering
Jadavpur University
Kolkata - 700 032

Acknowledgments

This thesis marks the culmination of a long and transformative journey, and I am deeply grateful to all those who have supported me through it.

First, I am immensely grateful to my supervisor, Professor Susmita Ghosh, for her steady guidance, valuable insights, and generous support throughout my doctoral research. Her encouragement and critical feedback helped shape both this work and my academic thinking.

I wish to pay special tribute to the late Professor C.A. Murthy. His dedication to bringing out the best in every student was truly extraordinary. His ideas, discussions, and perspective helped shape the direction of the work presented in this thesis in significant ways. His intellectual rigor, generosity of spirit, and deep commitment to learning continue to inspire me. Though he is no longer with us, his influence is woven throughout these pages.

I am also very grateful to Professor Ashish Ghosh, who took the time to listen to my ideas, offer constructive feedback, and review my work in critical stages. His support in helping me refine my thoughts and in presenting my work with greater clarity and detail has been invaluable.

I would like to thank my family for being my foundation. My parents (Shri A.K. Jain and Smt. Asha Jain), my in-laws (Smt. Pushpa Agrawal and the late Shri N.D. Agrawal), and my siblings (Abhisek Jain and Nehul Jain) have been constant sources of encouragement, patience, and strength. My father-in-law's support and belief in me, especially during the early stages of this journey, meant a great deal. Though he is no longer with us, his quiet strength and values continue to inspire me.

The support and sacrifices of my husband, Raja Saket, have sustained me through the most challenging times. His steadfast presence, willingness to shoulder everyday responsibilities, and thoughtful, positive criticism and feedback helped me improve and stay focused. I will forever be grateful.

I would also like to thank my daughter, Siddhi Agrawal, who has grown from a child to a thoughtful and perceptive teenager over the course of this journey. Her presence has been a constant source of joy and in recent years she has quietly supported me in many ways, offering help, sharing her observations, and bringing fresh perspectives that I came to deeply value.

I thank my colleagues, Rahul Roy and Abhiroop Chatterjee, who generously shared their valuable suggestions and feedback throughout this work. Their insights

and support contributed significantly to the development and refinement of this thesis.

Lastly, I express my gratitude to my friends and all who contributed to this journey.

Namita Jain
04/06/2025

Abstract

In this thesis, our main objective is to develop methods to explore non-linear relationships between features. For this, we have developed a feature relationship index (named MIDI) that can be used to quantify non-linear feature relationships, including the non-monotonous ones. We have also designed three biclustering methods that find biclusters based on non-linear feature relationships. All these methods have been applied to real-world datasets, allowing us to learn the patterns in the data.

This thesis introduces a novel feature relationship index, MIDI (Mutual Information-based Dependence Index), designed to quantify both linear and non-linear relationships between features. MIDI estimates feature dependence using mutual information derived from histogram-based density estimates. It employs an innovative bin length selection method based on the maximal spacing between consecutive data points, ensuring consistent and reliable density estimation. The resulting density estimate is used to compute a normalized mutual information score. MIDI does not assume any specific functional form, making it capable of detecting a wide range of complex relationships. Theoretical properties of MIDI are also discussed, and power statistics demonstrate its effectiveness in identifying non-linear relationships. Additionally, MIDI has been used to develop a novel feature selection method that shows promising results compared to existing techniques across several datasets.

This thesis presents three feature-relationship-based biclustering methods. The first of these is a density-based biclustering method. Unlike many existing density-based biclustering algorithms, which find clusters using spatial proximity, the existence of common high-density regions is used for grouping features by this method. This method is called Connectedness-Based Subspace Clustering (CBSC). In contrast to conventional density-based biclustering algorithms, for CBSC the user does not need to provide the parameter values for density estimation. These values are calculated for each pair of features using the data distribution in the space corresponding to the particular pair of features. Thus, the proposed approach can adapt to different data distributions in two-dimensional spaces, allowing it to find feature relation-based biclusters with greater accuracy. This approach allows CBSC to find biclusters based on non-linear and even non-monotonous relationships between features.

Another method for finding biclusters based on non-linear relationships, named RelDenClu, has also been presented. RelDenClu uses the local variations in the marginal and joint densities of each pair of features to find the subset of observations

that forms the basis of the relation between them. Then it finds the set of features connected by a common set of observations, resulting in a bicluster. This approach allows RelDenClu to adapt to local variations in density, leading to improved performance on synthetic and real-world datasets. It can find biclusters based on feature relations, including non-linear and non-monotonous ones.

In this thesis, a novel parameter-free biclustering method named PF-RelDenBi has also been presented. Like RelDenClu, PF-RelDenBi also uses local variations in marginal and joint densities to find observations that form dense regions for pairs of features. But unlike CBSC and RelDenClu, it is a parameter-free method. The thresholds required to find the biclusters are calculated from the data on the fly. In addition, it uses the feature relationship index called MIDI to find sets of related features from the dense regions obtained using relative density. Thus, it uses MIDI to find biclusters based on feature relationships, including non-linear and non-monotonous relations.

The biclustering methods presented in this thesis have been used to study the relationship between lifestyle factors in different countries and the COVID-19 infection rates in these countries. They have also been used to find clusters in three UCI-ML datasets. The biclustering methods have also been used to enhance UCI-ML datasets in order to perform classification of these UCI-ML datasets with higher accuracy. This is done by generating new features using the biclustering methods.

Broadly, these methods offer a powerful approach for analyzing data from complex systems composed of multiple interacting processes and may serve as a foundation for subsequent causal analysis. As datasets from diverse sources continue to be integrated, further application areas for these techniques are likely to emerge.

Contents

List of Figures

List of Tables

List of Symbols

$\#_{i,j}$	Number of points in the cell in i^{th} row and j^{th} column of the histogram
$\Delta(\mathcal{A}^*, (x_1, \cdots, x_n))$	Maximum number of unique partitions induced by $\mathcal{A}^*$ on fixed sequence of points $x_1, x_2, \cdots, x_n$
$\Delta_n^\star(\mathcal{A}^*)$	Maximum number of unique partitions induced by $\mathcal{A}^*$ on any sequence of points $x_1, x_2, \cdots, x_n$
Π	Partition scheme given by sequence of partitioning rules
$\delta(B)$	Boundary of set B
δ_D	Relationship index or dependence index, D is used to indicate dependence
$\delta_x(f)$	Evaluation function defined on spaces of functions that returns the value of unction f at x, that is $\delta_x(f) = f(x)$
ϵ	Constant greater than 0
γ_j	Spacing between j^{th} and $(j+1)^{th}$ points with uniform distribution
λ	Lebesgue measure
λ_2	Maximal Information Compression Index
$\mathcal{V}(X, Y)$	Brownian Covariance between variable X and variable Y
$\mu()$	Probability measure
π^*	Partition of d-dimensional space
π_x^*	A partition of x axis created by MIDI using a given pair of features
π_y^*	A partition of y axis created by MIDI using a given pair of features

$\pi_n[x]$	Cell assigned to x by the partitioning rule π_n that partitions a set of n variables $\bar{X}_1, \bar{X}_2, \cdots, \bar{X}_n \in \mathbf{R}^d$ in cells
π_n	Partitioning rule that partitions a set of n variables $\bar{X}_1, \bar{X}_2, \cdots, \bar{X}_n \in \mathbf{R}^d$ in cells
$\rho(x, y)$	Correlation coefficient between variable x and variable y
$\rho_S(x, y)$	Spearman correlation between variable x and variable y
$\rho_\tau(x, y)$	Kendall's Rank correlation coefficient between variable x and variable y
σ_X	Standard deviation of variable X
${\mathcal{A}^*}_n$	A family of partitions of a sequence of n points
$\mathcal{A}^*$	A family (possibly infinite) of partitions of $\mathbf{R}^d$
$\mathring{A}_1, \mathring{A}_2, \cdots$	Cells in partition π^*
$\mathbf{A}$	A matrix
$\mathbf{A}_{*,j}$	Column j of matrix $\mathbf{A}$
$\mathbf{A}_{i,*}$	Row i of matrix $\mathbf{A}$
$\mathbf{A}^T$	Transpose of matrix $\mathbf{A}$
B	A set in $\mathbf{R}^d$
B'	Complement of set B
C	A constant
c	A constant relevant for the method
$Card(B)$	Cardinality of set B
Cl	The set of observations in a bicluster
Clf	The set of features in a bicluster
$\mathbf{D}$	Dataset with N observations and M features
d	Dimensionality of Real space $\mathbf{R}^d$
$diam()$	Diameter
$\mathbf{E}$	Expectation of random variable

$f_1, f_2, \cdots, f_m$	A set of features in list notation
F_1, F_2	Two distinct sets of features
$\mathcal{F}, \mathcal{G}$	Two distinct Reproducing Kernel Hilbert Spaces (RKHS)
f_X	Characteristic function for X
f_{XY}	Joint characteristic function for variables X and Y
g_x, g_y	Range along x and y axis
$H(X)$	Entropy of variable X
$\hat{H}_x$	Estimated entropy of variable X
$\mathring{\mathbf{H}}$	$\mathring{\mathbf{H}} = \mathbf{I}_{n\times n} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$ (the centering matrix)
$I(X, Y)$	Mutual information between the variables X and Y
$\hat{I}$	Estimated mutual information between the variables X and Y
$Id(x)$	Identity function maps the value x to itself, i.e. $Id(x) = x$
$\mathbf{I}_{n\times n}$	Identity matrix of size n
$k_{\mathcal{F}}(x, y)$	Kernel function representing dot product in RKHS $\mathcal{F}$
$\mathbf{K}_{\mathcal{F}}$	Gram matrix defined by kernel function $k_{\mathcal{F}}()$
l^*	Length of Minimal Spanning Tree
$l_{\mathcal{G}}(x, y)$	Kernel function representing dot product in RKHS $\mathcal{G}$
$\mathbf{L}_{\mathcal{G}}$	Gram matrix defined by kernel function $l_{\mathcal{G}}()$
L_{max}	The maximal spacing between points with continuous distribution
$min()$	Minimum
$max()$	Maximum
$M_k^{(n)}$	The k^{th} maximal spacing between points with continuous distribution
M_n	Maximal spacing for n points with uniform distribution
MST	Minimal Spanning Tree

$m(\mathcal{A}^*)^n$	Maximum number of cells in any partition of n points by the class of partitions $\mathcal{A}^*$
$N(x)$	Neighborhood of a point x
O_1, O_2	Two distinct sets of observations
O_C	Set of observations in a class
O_B	Set of observations in bicluster found by a method
$p(x)$	probability that the variable X assumes the value x
$\mathbf{R}^d$	d-dimensional space of real numbers
$R(\mathbf{A}_{*,u}, \mathbf{A}_{*,v})$	Relation between u^{th} and v^{th} columns of $\mathbf{A}$
$sgn(x)$	Function: $sgn(x) = 1$ if $x > 0$, $sgn(x) = 0$ if $x = 0$ and $sgn(x) = -1$ if $x < 0$
$S_i^{(n)}$	Spacing between points with continuous distribution
S_o	Set of observations in seed bicluster S
S_f	Set of 3 features in seed bicluster S
S	Seed bicluster defined by S_o and S_f
$\{U_k\}$	Points drawn iid from Uniform distribution
$\{U^*_{N,k}\}$	Ordered points drawn iid from Uniform distribution
$Var(X)$	Variance of variable X
V_n	Area of Parzen window for a spave with n points
$\bar{x}$	Mean of variable X
$\vec{X}_1, \vec{X}_2, \cdots \vec{X}_n$	An i.i.d. sequence of random variables each belonging to $\mathbf{R}^d$
X, Y	Random variables

Chapter 1

Introduction

1.1 Introduction

Pattern processing serves as the foundational core for the unique capabilities of the human brain, such as intelligence, linguistic ability, imagination, and invention [1]. Humans can perform pattern recognition and analysis even in non-ideal situations. However, it is difficult for humans to deal with a voluminous amount of data. On the other hand, computers are designed to handle a large amount of data. To ensure that machines can handle the uncertainties of the real world, we need algorithms to detect patterns present in the data, where the environment is not certain. Thus, the field of Pattern Recognition [2,3] deals with designing precise sets of instructions that computers can follow and display the ability to recognize patterns under uncertain conditions.

An important aspect of pattern recognition and knowledge discovery is understanding the relationship between different features in a dataset. In this thesis, we examine methods to discover non-linear relationships between different features. Here, we have explored two approaches to analyze feature relations. The first method is

to design a non-linear feature relationship index which is used to quantify the relationship present between features, and the second approach is feature relation-based biclustering. The latter allows us to discover feature relations existing in a part of a dataset, including non-linear and non-monotonous relationships.

1.2 What are feature relationship indices

In machine learning, we say that two variables are related if one variable can be used to predict the other. We say that the variable Y is related to X or depends on X if, for any given value of X, the value of Y is fixed. In such a case, we say that there is perfect dependence of Y on X. However, the dependence between variables is not always ideal. In many cases, having information about one variable may give partial information about the other. In many scenarios, identifying and quantifying such a feature relation is important.

1.2.1 Linear feature relationship indices

The most popular way to measure the relations between variables is the correlation coefficient, which quantifies the linear relations between variables. For two variables X and Y taking the values $x_1, x_2, \cdots, x_n$ and $y_1, y_2, \cdots, y_n$, respectively, the correlation coefficient $\rho(X, Y)$ is given by

$$\rho(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{1.1}$$

where $\bar{x}$ and $\bar{y}$ denote mean values of the variables X and Y, respectively. The mean can be calculated as $\bar{x} = \sum_{i=1}^{n} x_i$. We may notice that the correlation coefficient may have any value in the interval $[-1, 1]$. However, the strength of the relation is given

by the absolute value of the correlation coefficient. Thus, a value close to -1 also indicates that there is a nearly perfect linear relation between the variables.

Consider the standardized variables X' and Y' where $x'_i = (x_i - \bar{x})/\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}$ and $y'_i = (y_i - \bar{y})/\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}$, then the correlation coefficient is the dot product between X' and Y' :

$$\rho(X, Y) = \rho(X', Y') = X'.Y' \tag{1.2}$$

Since X' and Y' are unit vectors, this is also the cosine distance between them. Furthermore, if linear regression is performed between X' and Y' the regression coefficient is equal to $\rho(X', Y')$ and the sum of squared errors will be given by $(1 - \rho(X', Y')^2)$. Thus, it is easy to visualize how the correlation coefficient measures the linear relation. In fact, the correlation coefficient has been studied from several perspectives, making this simple measure very interesting [4].

1.2.2 Non-linear feature relationship indices

The correlation coefficient has been very useful in uncovering important relationships between different factors involved in many real-life situations. However, zero correlation does not imply independence of the variables [5]. In many situations, the relationship between variables may be non-linear. Non-linear relations may be monotonous or non-monotonous. In the case of monotonous relations, one variable may be approximated as an increasing or decreasing function of the other. An example would be $y = x^2$, where $x \in (0, 1)$. In other cases, the relation might be arbitrary. For example, $y = \sin(4\pi x)$, for $x \in (0, 1)$ is a non-monotonous relation. Ideally, a feature relationship index should perform equally well for linear, monotonous, or non-monotonous relations between the variables. Since the feature relationship index is

useful for comparing the strength of relationships between variables, its value should lie in a fixed range. The range $[0, 1]$ is a natural choice, with 0 showing independence and the value 1 depicting perfect feature relation. It is also preferable that the feature relationship index is invariant to one-to-one transforms of either or both variables. These requirements have been formalized by researchers who have listed properties that a feature relationship index δ_D should possess [6] [7]. Here we summarize Renyi's list [6] of properties that an ideal feature relationship index should fulfill.

- It should be defined for any pair of variables when neither is a constant and its value should lie in the interval $[0, 1]$.
- It should be symmetric i.e., $\delta_D(X, Y) = \delta_D(Y, X)$
- $\delta_D(X, Y) = 1$ if and only if there exists a Borel measurable [8] function f such that $Y = f(X)$ or there exists a Borel measurable function g such that $X = g(Y)$.
- If f and g are Borel measurable one-to-one onto functions then $\delta_D(f(X), g(Y)) = \delta_D(X, Y)$.
- $\delta_D(X, Y) = 0$ if and only if X and Y are independent.
- If X and Y have joint normal distribution, then $\delta_D(X, Y) = \rho(X, Y)$, where ρ is the correlation coefficient.

1.2.3 Feature relations and causality

Correlation implies association, but not causation. In contrast, causation implies association or dependence between variables [9]. Thus, statistical dependence is a necessary but not sufficient condition for causality. It may be noted that the relation

between the causally related variables may be non-linear [10]. Thus, the correlation between causally related variables may still be zero. If we have a method to find a non-linear relation between variables, it may help us discover causal relations by eliminating unrelated variables. The statistical dependence between two variables often occurs due to seen or unseen factors that impact both variables. Causal links between variables can be analyzed using control experiments or further data analysis [11].

1.2.4 Application of feature relationship indices to pattern recognition problems

Since feature relationship indices can be used to identify variables that carry similar information, they can be used to identify redundant features in a dataset. Thus, they can be used for unsupervised feature selection in datasets. It is well known that feature selection can lead to significantly better performance of pattern recognition and machine learning algorithms [12]. For some classification problems, feature selection is even more important than classifier choice [13].

As mentioned in the previous section, feature relationship indices can be used as a preliminary analysis before searching for causal links in the data. Thus, it is an important tool in data mining.

However, the causal relationship between real-life variables can be quite complex. Several factors may impact the outcomes represented by a single variable. In addition, the association between two variables may exist for a subset of observations rather than the entire dataset. One of the techniques that can be used to discover such an interaction between variables is biclustering. Biclustering will be discussed in the following sections of this chapter. Here we would like to mention that relationship

indices can be useful for finding biclusters based on feature relationships. Figure 1.1 depicts the role of feature relationship indices in machine learning systems.

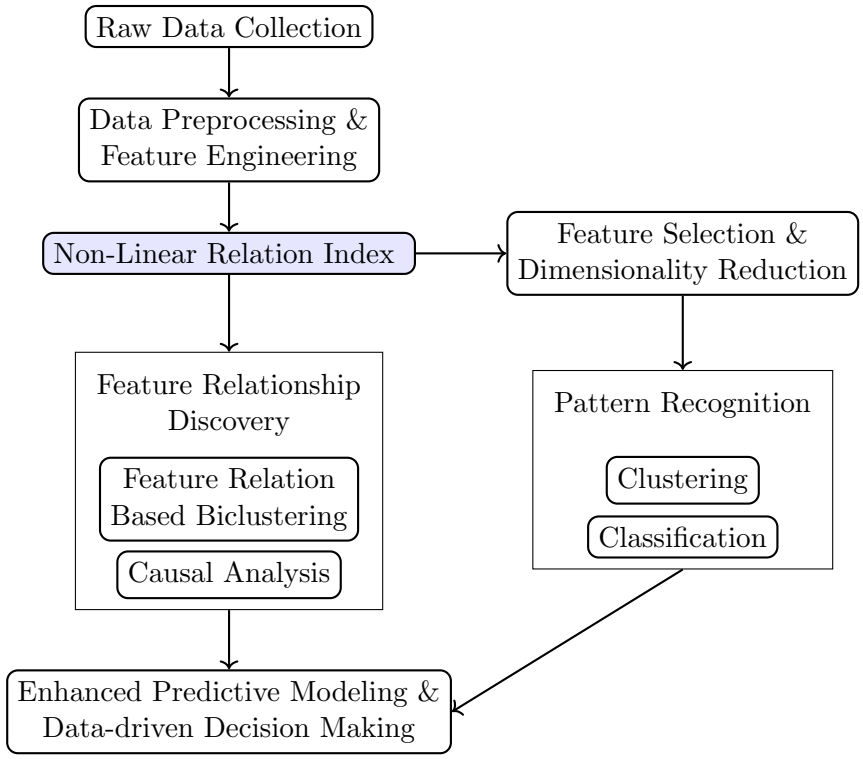


Figure 1.1: Role of Feature Relation Indices in Machine Learning Problems

1.3 Biclustering: An approach to find hidden feature relations in the data

In addition to a feature relationship index, we have explored the use of biclustering to study hidden feature relations. In this section, we discuss what is biclustering. We know that clustering algorithms partition the data matrix into disjoint sets of observations. On the other hand, there are methods to partition features based on the similarity between them. A bicluster is a pair of two sets: a set of observations and a set of features [14]. Biclustering methods may find one or many biclusters in the

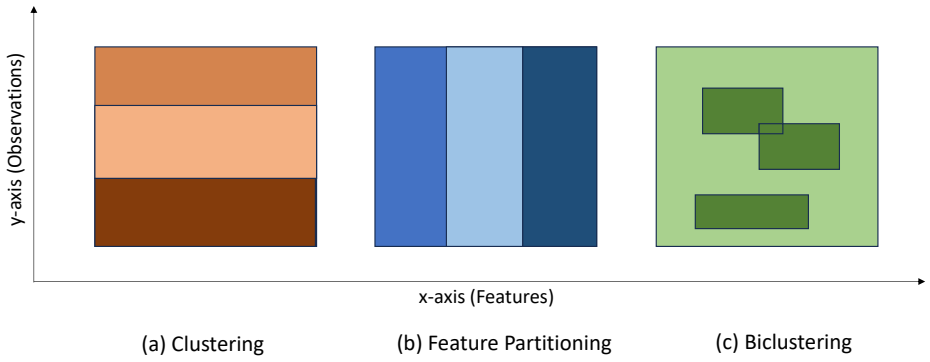


Figure 1.2: (a) shows clustering which partitions observations, (b) shows feature partitioning, (c) biclusters are seen as small rectangles within data matrix which depict subsets of observations and also subsets of features

data. Figure 1.2 represents the difference between clustering, feature partitioning, and biclusters diagrammatically. Each of the squares shown in Fig. 1.2 represents a data matrix with observations along the y-axis and features along the x-axis. As depicted in Figure 1.2 (a), traditional clusters are seen as horizontal rectangles. Figure 1.2 (b) depicts the feature partitions shown as vertical rectangles. Figure 1.2(c) shows the biclusters as arbitrary rectangles. Note that biclusters do not cover the entire data matrix. Also, a data matrix can contain partially overlapping biclusters i.e., sets of observations and features need not be disjoint.

1.3.1 Why we need biclustering

With increasing occurrence of high-dimensional data, biclustering has become popular [15]. Some scenarios where it is useful are briefly discussed here. Subspace clustering can be used when the intrinsic dimensionality of the data is lower than the apparent dimensionality. It can also be used when the set of relevant attributes for a given problem is a subset of all attributes in the dataset. Similarly, for some problems, only

a subset of observations may be relevant. Biclustering is useful for identifying such a set of observations. For knowledge discovery, biclustering may be used to find the relation between a few attributes which exist only for a subset of observations.

In the existing literature, several algorithms for subspace clustering and biclustering algorithms are available [15–18]. These vary according to their objective functions and methods to achieve the optimal solution or its approximation. For example, an algorithm may detect a bicluster if a set of points forms a high-density region in some hyperplane. Another algorithm may look for a high-density convex region in a subspace.

1.4 Goals of biclustering methods

Our main objective in this thesis is to examine the relationships between features. In the forthcoming chapters, we demonstrate that biclustering can be used as an approach to study the relations that exist for a subset of observations. However, biclustering algorithms are often designed with varying objectives. In this section, we discuss these objectives.

1.4.1 Spatial proximity

Many biclustering algorithms find sets of observations that lie close to each other in a subspace formed by selected features. In a traditional cluster, the observations within the same cluster are close to each other, as measured by a chosen distance, such as the Euclidean distance or the Manhattan distance. However, the distance between observations belonging to a bicluster is not calculated using the entire feature set. Instead, only features belonging to the bicluster are used. Thus, each bicluster has its unique definition of distance depending on the specific subspace. Using this distance,

the observations lying within a bicluster are close to each other. Just as in traditional clusters, the distance between the observations can be used in different ways to obtain the biclusters. For example, minimizing the distance between the observations and the center of the bicluster will result in convex biclusters. Thus, some methods search for biclusters in which observations form a dense convex set in the chosen subspace [19]. On the other hand, other methods do not consider convexity as a criterion and search for biclusters where observations form densely connected regions in the selected subspace [20]. We know that clustering methods such as K-means [21] and DBSCAN [22] have different objectives. Similarly, biclustering algorithms with the objectives of obtaining convex or connected biclusters have been designed and applied for data analysis by several researchers [16, 23].

1.4.2 Feature relation based biclustering

Achieving small distances between observations in the corresponding subspace is useful and is the main objective of many biclustering algorithms. However, other biclustering algorithms do not look for spatial proximity, but perform biclustering based on some other peculiarity in data. One of the criteria is the similarity between the features included in the biclusters.

Biclustering algorithms based on feature relationships examine the relationship between different columns or feature values for a selected set of observations. Thus, these algorithms calculate the distance between the features based on the selected sets of observations to find biclusters. While biclusters based on spatial proximity can be compared with traditional clustering methods, feature relation-based biclustering can be compared with feature partition algorithms. These biclustering algorithms may search for biclusters based on linear, monotonous, or non-monotonous

relations. Biclusters based on linear relations are often known as biclusters with coherent values. Biclusters based on non-linear but monotonous relations between the features are called biclusters with coherent evolution [18]. Finding biclusters with non-monotonous is more challenging, and fewer methods are available for this task [16].

1.5 Biclustering algorithms and their properties

The advantage of biclustering algorithms is that they allow us to explore the data in many more ways than the traditional clustering or feature partition algorithms. However, this flexibility comes at a computational cost and biclustering is known to be NP-hard [14]. Most of the algorithms use a heuristic approach to arrive at a solution. These algorithms can respond differently to data that is transformed in different ways. Here, we study some of these behaviors or properties of biclustering algorithms. This study is inspired by an article on the admissibility of clustering methods by [24] and discusses several properties of clustering methods, such as invariance to linear transforms, cluster proportionality, and point proportionality. However, many of these properties have not been discussed for biclustering methods. In the following paragraphs, we discuss these properties with regard to biclustering methods.

1.5.1 Consistency under data transforms

The biclustering methods may or may not be invariant to various data transforms. These include data scaling, translation by a fixed value, and linear transforms applied to data. Scaling and translation do not alter the shape of the biclusters, and it is easier to build algorithms invariant to these transforms. Biclustering algorithms can be designed to be invariant to dimensionality-preserving linear transforms. For

example, Pearson correlation-based biclustering methods will be invariant to such transforms [25].

1.5.2 Proportion admissibility

Fisher and Ness discuss the importance of proportionality, which reflects the geometric aspects of clustering [24]. Similar criteria are also useful for biclustering methods. We say that an algorithm has the property named proportion-admissibility if it gives consistent bicluster boundaries if some observations are duplicated. Since the repetition of observations does not change the shape of the biclusters, we can say that such algorithms give higher importance to the data topology.

1.5.2.1 Point-proportion

A procedure is said to be point-proportion admissible if, after duplication of one or more observations, any number of times the boundaries of the biclusters are not changed. As mentioned above, in the past this property has been discussed for clustering [24], but has not been discussed with reference to biclustering methods.

1.5.2.2 Cluster-proportion

A biclustering procedure is said to be cluster-proportion admissible if duplicating the instances lying in each cluster an arbitrary number of times does not change the results. In other words, when each observation within the same bicluster is duplicated the same number of times, biclusters having the same boundaries are generated. Similarly to clustering [24], we can evaluate if a biclustering method exhibits cluster proportionality. To the best of our knowledge, this property has not been discussed for biclustering methods earlier.

1.5.3 Bicluster robustness

As with other machine learning problems, it might be desirable that adding small noise to the data does not drastically alter the results. A procedure having this property is said to be robust. The robustness of biclustering methods has been discussed by several authors in the past [16, 18, 26, 27].

1.5.4 Changing indices

We want a biclustering algorithm to be invariant to changes in the permutation of observations and features. Since adding or removing features can unpredictably change distances between the points, biclustering algorithms need to examine a large number of subsets of features and observations to find an optimal solution. Most algorithms do not search for an optimal solution to avoid an exhaustive search [16]. A biclustering algorithm may produce a different output if the order of rows or columns changes. However, it is desirable that changing the permutation of rows or columns does not change the performance of the biclustering algorithm.

1.6 Scope of the thesis

This thesis is dedicated to the analysis of relations between continuous variables within a dataset. This is achieved through the introduction of a non-linear feature relationship index and the development of biclustering methods based on non-linear feature relations. Furthermore, this thesis undertakes an examination of the properties of biclustering methods.

As mentioned, one way we have explored feature relations is by designing a novel Mutual Information based Dependence Index (MIDI). MIDI can be used to quantify

statistical dependence between variables. MIDI uses normalized mutual information to evaluate feature relations. Thus, it is able to detect non-linear and non-monotonous relations in addition to linear relations between features. MIDI uses a data-dependent histogram scheme that leads to consistent density estimates. These estimates are used to calculate the mutual information between features. This scheme makes MIDI robust to the presence of noise. The properties of the method have been discussed and empirically demonstrated in [28]. MIDI has been used to design a novel feature selection method, which has been applied to several benchmark datasets [29]. MIDI and the novel feature selection method have been discussed in Chapter 3.

Another way in which the relationship between the variables is explored is biclustering. We have developed biclustering methods, namely CBSC [30], RelDenClu [31, 32] and PF-RelDenBi [33]. These are feature-relation-based biclustering methods and have been used to analyze several benchmark datasets. They have also been used to study lifestyle factors associated with the spread of the COVID-19 pandemic. These methods find biclusters based on non-monotonous relations.

CBSC has been presented in Chapter 4. It finds dense regions in two-dimensional spaces by analyzing the neighborhoods for observations and the amount of overlap between the neighborhoods. Later, it uses dense regions in two-dimensional spaces to obtain the biclusters using connectivity between the features.

RelDenClu also finds dense regions in a two-dimensional space, but instead of comparing the density of a small region to the average density of the entire two-dimensional space, it makes the comparison with the marginal densities. This allows it to find relationships between the features even when the density is more variable. Once dense regions are found, the biclusters are found using connectivity in a way similar to CBSC. This method is presented in Chapter 5.

Another method, PF-RelDenBi, also uses the comparison between the density of

smaller regions and the marginal density. However, it does not depend on user-defined parameters to obtain the biclusters, as it calculates all the thresholds required for the method from the data itself. In addition, it also uses the MIDI feature relationship index (discussed in Chapter 3) to combine dense regions obtained from the analysis of two-dimensional spaces. Thus, the features in a biclusters are connected to each other by relations as quantified by MIDI. PF-RelDenBi has been elaborated in Chapter 6. We have also conducted experiments with several synthetic datasets to study the properties of biclustering methods which are presented in this thesis.

As already discussed, finding biclusters based on non-monotonous relations is more challenging and has received a lesser number of solutions. Thus, we hope that biclustering methods to address this problem will be valuable in data mining applications.

1.7 Organization of the thesis

The rest of the thesis is organized as follows. Chapter 2 presents the literature review on some of the existing feature relationship indices and biclustering methods. Chapter 3 presents a novel mutual information-based feature relationship index (MIDI), which provides an efficient method to evaluate non-monotonous relationships between features. Chapter 4 presents a biclustering method (CBSC) that can be used to find biclusters based on feature relations using density in two-dimensional spaces. Chapter 5 presents biclustering method (RelDenClu) to find biclusters based on feature relations using relative density in two dimensional spaces. Chapter 6 presents a parameter-free biclustering method (PF-RelDenBi) that finds biclusters based on the feature relation index MIDI. Finally in Chapter 7, we summarize the work done in this thesis and discuss the scope for future work.

Chapter 2

Literature survey: Methods for understanding feature relations

2.1 Introduction

In this chapter, we discuss feature relationship indices that can be used to assess the relationship between variables. As discussed in the previous chapter, these include indices that can be used to quantify linear, monotonous, and non-monotonous relations. We will also review biclustering methods. Some of these biclustering methods can be used to find hidden relations present in the data.

2.2 Feature relation indices

In the previous chapter, we discussed the importance of feature relationship indices and looked at the definition of Pearson correlation [4, 34]. It is one of the first tools widely used to evaluate the relationship between variables. We also saw that Pearson correlation can only find linear relations between features. In this section, we discuss

a few other relationship indices. Many of them are non-linear feature relationship indices. Several methodologies have been used to find similarity in features. These include rank-based methods, kernel-based methods, information-theoretic methods, and methods designed for specific data types.

2.2.1 Indices to evaluate linear relationships between features

In the previous chapter, we saw that the correlation coefficient is the most popular method to evaluate the linear relations between variables. Another method used to assess linear relations between variables is the Maximal Information Compression Index (MICI) [35]. Given variables X and Y, MICI (λ_2) is defined as the smallest eigenvalue of the covariance matrix. It can be calculated as follows:

$$2\lambda_2(X,Y) = (var(X) + var(Y)) - \sqrt{(var(X) + var(Y))^2 - 4var(X)var(Y)(1 - \rho(X,Y)^2)}, \quad (2.1)$$

where $var(X)$ and $var(Y)$ denote the variance of X and Y, respectively, and the correlation between the two variables is denoted by $\rho(X,Y)$. MICI (λ_2) is invariant to translation, but not to scaling.

2.2.2 Rank based methods

Just as Pearson correlation coefficient is the most popular method used to evaluate linear relations, Spearman correlation [36, 37] is the most popular index for evaluating monotonic relations. It works by ranking the data for each variable and then calculating the Pearson correlation on those ranks. It will have values in the range $[-1, 1]$, and the strength of the relationship is indicated by the absolute value of the

Spearman correlation.

Let X and Y be variables with n observations each, and the i^{th} observations be denoted by x_i and y_i, respectively. Let u_i and v_i, respectively, denote the ranks of x_i in x and y_i in y. So, the i^{th} values of U and V are u_i and v_i, respectively. Then the Spearman correlation ρ_S is given by:

$$\rho_S(X,Y) = \rho(U,V), \tag{2.2}$$

where ρ denotes the Pearson correlation. Alternatively, it is also given by

$$\rho_S(X,Y) = \frac{\sum_{1\leq j<i\leq n}(u_i-u_j)(v_i-v_j)}{\sqrt{\sum_{1\leq j<i\leq n}(u_i-u_j)^2\sum_{1\leq j<i\leq n}(v_i-v_j)^2}}, \tag{2.3}$$

Like the Spearman correlation, Kendall's rank correlation coefficient [37,38] evaluates the strength of monotonic relations using the ranks of the observations. It is given by:

$$\rho_\tau(X,Y) = 2\frac{\sum_{1\leq j<i\leq n}\mathrm{sgn}(u_i-u_j)\mathrm{sgn}(v_i-v_j)}{n(n-1)}, \tag{2.4}$$

where sgn() is the signum function, i.e., sgn(x)=1 for $x>0$, sgn(x)=-1 for $x<0$ and sgn(x)=0 for $x=0$. Note that $\sum_{1\leq j<i\leq n}(\mathrm{sgn}(u_i-u_j))^2 = \sum_{1\leq j<i\leq n}(\mathrm{sgn}(v_i-v_j))^2 = n(n-1)/2$. Taking this into account, we find that Kendall's rank correlation (ρ_τ) is very similar to the Spearman correlation and can be obtained simply by replacing (u_i-u_j) and (v_i-v_j) by $\mathrm{sgn}(u_i-u_j)$ and $\mathrm{sgn}(v_i-v_j)$. It can also be seen as

$$\rho_\tau = 2\frac{(\text{(number of concordant pairs)} - \text{(number of discordant pairs)})}{n(n-1)}. \tag{2.5}$$

A pair of observations (x_i,y_i) and (x_j,y_j) is concordant if both $x_i > x_j$ and

$y_i > y_j$ or both $x_i < x_j$ and $y_i < y_j$. Otherwise, the pair of observations is discordant. This formulation makes Kendall's rank correlation more robust than the Spearman correlation. Both Spearman and Kendall's τ detect monotonic relations. They may have a low value if the relation between two variables is perfect but non-monotonic.

Another index named Xi correlation (XICOR) uses the rank of variables and detects non-monotonic relations [39]. For two variables X and Y, sort the values in increasing order $((X_{(1)}, Y_{(1)}), (X_{(2)}, Y_{(2)}), \cdots, (X_{(n)}, Y_{(n)}))$ so that $X_{(1)} \leq X_{(2)}, \cdots, X_{(n)}$. Let u_i be the rank of $Y_{(i)}$ then if there are no repeating values in X and Y the dependence index is given by:

$$\xi_n(X, Y) = 1 - \frac{3 \sum_{i=1}^{n} |u_{i+1} - u_i|}{n^2 - 1}. \tag{2.6}$$

If there is repetition of values in X or Y a modified formula has been provided to calculate $\xi_n(X, Y)$ [39]. It may be noted that the value of $\xi_n(X, Y)$ does not strictly lie in the range $[0, 1]$. The minimum value that $\xi_n(X, Y)$ can achieve is known to be $-1/2 + O(1/n)$ where O denotes the big O notation and n is the number of observations. The maximum value $\xi_n(X, Y)$ can achieve is $(n-2)/(n+1)$. Although for independent identically distributed (i.i.d.) points, the limiting value of the index lies in the range $[0, 1]$. XICOR is an asymmetric distance measure, that is, it is possible that $\xi_n(X, Y) \neq \xi_n(Y, X)$.

2.2.3 Kernel based methods

Researchers have proposed many kernel-based statistical independence measures in which data points are mapped from the input space to feature space, called the Reproducing Kernel Hilbert Space (RKHS) [40, 41]. Thus, kernels implicitly determine the mapping from the input space to the feature space and can be used to calculate

the inner product of data points in the feature space. The inner product can be used to calculate the cross-covariance in the feature space. Kernel-based dependence measures differ from each other in terms of the normalization they use and how they summarize the covariance operator spectrum [42].

This paragraph presents a brief overview of Reproducing Kernel Hilbert Spaces. Suppose $\mathcal{F}$ is a Hilbert space of functions $\mathcal{X} \to \mathbb{R}$ where $\mathcal{X}$ is a separable metric space. The evaluator operator at any point $x \in \mathcal{X}$ can be represented as $\delta_x(f) : \mathcal{F} \to \mathbb{R}$ and maps $f \in \mathcal{F}$ to $f(x)$, i.e. $\delta_x(f) = f(x)$. Then $\mathcal{F}$ is an RKHS if at each $x \in \mathcal{X}$, the operator δ_x is a bounded linear functional. There exists a mapping $\phi(x) \in \mathcal{F}$ such that the dot product $< \phi(x), \phi(x') >= k_{\mathcal{F}}(x, x')$ where $k_{\mathcal{F}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a unique positive definite kernel. Thus, the dot product in feature space can be calculated directly using the kernel function. This allows us to design feature relationship indices using kernel functions, and we do not need to find feature maps explicitly.

Constrained Covariance (COCO) [41] is one of the first feature relationship indices to use kernels. For two classes of functions denoted by $\mathcal{F} : \mathcal{X} \to \mathbb{R}$ and $\mathcal{G} : \mathcal{Y} \to \mathbb{R}$ and a given probability measure $P_{x,y}$, the authors define constrained covariance as:

$$COCO(P_{x,y}, \mathcal{F}, \mathcal{G}) = \sup_{f \in \mathcal{F}, g \in \mathcal{G}} cov(f(x), g(y)), \tag{2.7}$$

where *cov* denotes the covariance, i.e. $cov(x, y) = \mathrm{E}_{x,y}[x, y] - \mathrm{E}_x[x]\mathrm{E}_y[y]$. Given n independent observations $z = (x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$ the empirical value of COCO is given as:

$$COCO(z, \mathcal{F}, \mathcal{G}) = \sup_{f \in \mathcal{F}, g \in \mathcal{G}} [\frac{1}{n} \sum_{i}^{n} f(x_i)g(x_i) - \frac{1}{n^2} \sum_{i}^{n} f(x_i) \sum_{i}^{n} g(x_i)]. \tag{2.8}$$

Now suppose $\mathcal{F} : \mathcal{X} \to \mathbb{R}$ and $\mathcal{G} : \mathcal{Y} \to \mathbb{R}$ are two RKHS and the corresponding

kernel functions are given by $k_{\mathcal{F}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $l_{\mathcal{G}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Consider the gram matrices $\mathbf{K}_{\mathcal{F}}$ and $\mathbf{L}_{\mathcal{G}}$ in the spaces $\mathcal{F}$ and $\mathcal{G}$, respectively. Since the gram matrix corresponds to the inner product between the vectors in the RKHS, it can be calculated using the kernels $k_{\mathcal{F}}$ and $l_{\mathcal{G}}$. This is because the inner product can be expressed as $< \phi(x), \phi(x') >= k_{\mathcal{F}}(x, x')$ and $< \psi(y), \psi(y') >= l_{\mathcal{G}}(y, y')$ where ϕ and ψ are feature maps in RKHS $\mathcal{F}$ and $\mathcal{G}$, respectively. The centered version of the gram matrices can be represented as $\tilde{\mathbf{K}_{\mathcal{F}}} = \mathring{\mathbf{H}}\mathbf{K}_{\mathcal{F}}\mathring{\mathbf{H}}$ and $\tilde{\mathbf{L}_{\mathcal{G}}} = \mathring{\mathbf{H}}\mathbf{L}_{\mathcal{G}}\mathring{\mathbf{H}}$, where $\mathring{\mathbf{H}} = \mathbf{I}_{n\times n} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$. Note that $\mathbf{1}_n$ is $n \times 1$ vector of ones and $\mathbf{I}_{n\times n}$ is identity matrix of size $n \times n$. It has been shown that if F and G are unit balls in RKHS $\mathcal{F}$ and $\mathcal{G}$, respectively, then:

$$COCO(z, F, G) = \frac{1}{n}\sqrt{\left\|\tilde{\mathbf{K}_{\mathcal{F}}}\tilde{\mathbf{L}_{\mathcal{G}}}\right\|_{\lambda}} \tag{2.9}$$

where $\left\|\tilde{\mathbf{K}_{\mathcal{F}}}\tilde{\mathbf{L}_{\mathcal{G}}}\right\|_{\lambda}$ indicates the largest singular value of $\tilde{\mathbf{K}_{\mathcal{F}}}\tilde{\mathbf{L}_{\mathcal{G}}}$.

Instead of taking the largest singular value, the Hilbert-Schmidt norm can be used to evaluate independence. The resulting method is called the Hilbert-Schmidt Independence Criteria (HSIC) [43]. It can be estimated as:

$$HSIC((z, \mathcal{F}, \mathcal{G}) = (n-1)^{-2}\mathbf{trace}(\mathbf{K}_{\mathcal{F}}\mathring{\mathbf{H}}\mathbf{L}_{\mathcal{G}}\mathring{\mathbf{H}}) \tag{2.10}$$

HSIC can also be seen as the squared Maximal Mean Discrepancy (MMD) between the joint probability distribution $P(X, Y)$ of the two variables and the product of marginal distributions $P(X)P(Y)$ [44]. In this regard, it is analogous to Mutual Information, which measures the distance between the two distributions using KL-Divergence [45,46]. We will discuss mutual information-based methods in more detail in the following sections. For an RKHS $\mathcal{F}$ and two probability distributions P_X and

P_Y on variables x and y, MMD [47] between two distributions is given as:

$$MMD(\mathcal{F}, P_x, P_y) = \sup_{f \in \mathcal{F}}(E_x[f(x)] - E_y[f(y)]) \tag{2.11}$$

For two variables $X = (x_1, x_2, \cdots, x_n)$ and $Y = (y_1, y_2, \cdots, y_m)$ and an RKHS $\mathcal{F}$ with kernel $k_{\mathcal{F}}$, an empirical estimate of MMD is given as:

$$MMD(\mathcal{F}, X, Y) = \left[\frac{1}{n^2}\sum_{i,j=1}^{n} k_{\mathcal{F}}(x_i, x_j) + \frac{1}{m^2}\sum_{i,j=1}^{m} k_{\mathcal{F}}(y_i, y_j) - \frac{2}{mn}\sum_{i,j=1}^{n,m} k_{\mathcal{F}}(x_i, y_j)\right]^{1/2} \tag{2.12}$$

Poczos et al. proposed a Copula-based dependence index [48] using MMD. The method is based on the observation that the variables X and Y are independent of each other if and only if the Copula distribution is a uniform multivariate distribution. Thus, it uses empirical Copula transformation to obtain transformed data with a joint probability distribution that has uniform marginals. If the features are independent, the joint distribution should be uniform. Thus, the distance between the distribution obtained from the transformed data and the uniform distribution is then measured using an estimate of MMD. The Gaussian kernel [49] has been used for the methods discussed above, as it is a universal kernel, i.e. the associated RKHS is dense in the space of bounded continuous functions.

2.2.4 Energy distance based dependence measures

Another way of testing independence is by energy distance-based statistics [50]. One of the most popular energy-based statistics is Distance Covariance [51, 52]. Let $X \in$

$\mathbb{R}^u$ and $Y \in \mathbb{R}^v$ be two variables, then Distance covariance is given by:

$$\mathcal{V}^2(X,Y) = \|f_{XY}(t,s) - f_X(t)f_Y(s)\|_w^2 \tag{2.13}$$

$$= \int_{\mathbb{R}^{u+v}} |f_{XY}(t,s) - f_X(t)f_Y(s)|^2 w(t,s)dt\ ds \tag{2.14}$$

Here $w(t,s) = (c_u c_v |t|_u^{u+1} |s|_v^{v+1})^{-1}$ where $|.|_u$ is the Euclidean norm in $\mathbb{R}^u$ and $c_d = \frac{\pi^{(1+d)/2}}{\Gamma((1+d)/2)}$. The characteristic function is denoted by $f_X(t)$, that is, $f_X(t) = \mathbf{E}[e^{\mathbf{i}tX}]$ where $\mathbf{E}$ is the expectation and $\mathbf{i} = \sqrt{-1}$. Similarly $f_Y(s) = \mathbf{E}[e^{\mathbf{i}sY}]$ and the joint characteristic function $f_{XY}(t,s) = \mathbf{E}[e^{(\mathbf{i}tX + \mathbf{i}sY)}]$. It has been shown that for an observed sample, the Brownian covariance [51] can be calculated as follows. For an observed sample $(X,Y) = (x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)$, let $a_{kl} = |x_k - x_l|_u$; then the elements a_{kl} form a matrix representing the Euclidean distance between each pair of points in $\mathbf{R}^u$. The average of the k^{th} row is $\overline{a}_{k.} = \frac{1}{n}\sum_{l=1}^{n} a_{kl}$ and the average of the l^{th} column is $\overline{a}_{.l} = \frac{1}{n}\sum_{k=1}^{n} a_{kl}$ and the average of all elements of the matrix is $\overline{a}_{..} = \frac{1}{n^2}\sum_{k,l=1}^{n} a_{kl}$. Now, define $\breve{A}_{kl} = a_{kl} - \overline{a}_{k.} - \overline{a}_{.l} + \overline{a}_{..}$. Then $\breve{B}_{kl}$ can be calculated in a similar way to $\breve{A}_{kl}$. Now, Distance covariance $\mathcal{V}_n(X,Y)$ can be defined by:

$$\mathcal{V}_n(X,Y)^2 = \frac{1}{n^2}\sum_{k,l=1}^{n} \breve{A}_{kl}\breve{B}_{kl} \tag{2.15}$$

Using Distance covariance the value of Distance correlation (Dcor) can be calculated as:

$$Dcor(X,Y) = \frac{\mathcal{V}_n(X,Y)}{\sqrt{\mathcal{V}_n(X,X)\mathcal{V}_n(Y,Y)}} \tag{2.16}$$

Thus, the normalized distance correlation can be calculated by normalizing the distance covariance to the range $[0,1]$. For the case of linear feature relationships, the

value of Distance correlation is the same as the Pearson correlation. Thus, it can be seen as a generalization of Pearson correlation. However, Dcor is unable to identify highly non-monotonous relations between features [53].

2.2.5 Statistical methods

Heller Heller Gorfine (HHG) [54] distance is based on the simple observation that a continuous relationship between variables X and Y implies that the observations in the neighborhood of a point x_i on the x-axis should also lie in the neighborhood of y_i on the y-axis. Thus, for each observation (x_i, y_i) the remaining observations are placed in a 2×2 grid representing points 1) lying in the neighborhood of both x_i and y_i, 2) lying outside of both neighborhoods, 3) lying only in the neighborhood of x_i and 4) lying only in the neighborhood of y_i. HHG then calculates the Pearson Chi-square statistic [34, 55] for the grid mentioned above and averages it over all observations and neighborhoods of different radii. The different radii are based on the nearest neighbors. It may be noted that the test value obtained depends on the distributions of the variables x and y and is not restricted to the interval $[0, 1]$. Thus, to evaluate the strength of the relation, we perform the test mentioned above on random permutations of the same data. For random permutations, there is no relation between the variables, and the value is expected to be low. For the original permutation, if x and y are related, the value of the test statistic will be high. By finding how many times the value of the test statistic for x and y exceeds the value of the statistic for random permutations, we find the strength of the relation. We can obtain a normalized value in the range $[0, 1]$ by dividing this by the number of random permutations used for the testing.

2.2.6 Information theoretic methods

One way of approaching feature relations is from an information theory perspective. Entropy quantifies the uncertainty of a random variable. In other words, entropy is the amount of information required to convey the state of a random variable [45, 56]. The entropy of a random variable X taking values in the set $\mathcal{X}$, is given by $H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x)$, where $p(x)$ is the probability distribution. Similarly, we can quantify the uncertainty in the variable Y when X is given. It is called conditional entropy [56, 57] and can be calculated as follows:

$$H(Y|X) = -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)} \tag{2.17}$$

where Y takes the values in the set $\mathcal{Y}$. It can also be calculated as:

$$H(Y|X) = H(X,Y) - H(X) \tag{2.18}$$

where $H(X,Y)$ is the joint entropy of X and Y. The reduction in uncertainty about Y when X is known is given by :

$$I(X,Y) = H(Y) - H(Y|X) \tag{2.19}$$

This reduction in uncertainty can also be seen as the information that the variable X carries about Y. Note that this quantity is symmetric. It is called the mutual information between variables X and Y.

$$I(X,Y) = H(Y) - H(Y|X) = H(X) + H(Y) - H(X,Y) = H(X) - H(X|Y) \tag{2.20}$$

Mutual information can also be seen as the KL divergence between distributions

defined by joint probability $p(x, y)$ and the product of marginal probabilities.

$$I(X, Y) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \tag{2.21}$$

Of course, both definitions mentioned above are equivalent. This can be seen as follows.

$$\begin{aligned} I(X, Y) &= H(Y) - H(Y|X) && (2.22)\\ &= -\sum_{y \in \mathcal{Y}} p(y) \log p(y) + \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)} && (2.23)\\ &= -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log p(y) + \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)} && (2.24)\\ &= \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} && (2.25) \end{aligned}$$

Note that the mutual information of a random variable with itself is its entropy. Thus, it is also expected value of self-information [58]. As conditional entropy $H(Y|X)$ never exceeds the entropy $H(X)$, mutual information is always non-negative. The upper bound of mutual information $I(X, Y)$ is given by the entropies $H(X)$ and $H(Y)$. Since mutual information can be arbitrarily large, a direct comparison of mutual information cannot be used to compare the strength of the relationship between different pairs of variables. Thus, researchers have suggested different ways to normalize mutual information [59, 60]. Normalizing mutual information by maximum entropy results in a metric $I(X, Y)/\max(H(X), H(Y))$ that can be used to evaluate feature relations [61]. Another index is obtained by normalizing mutual information by average entropy, which is given as $2I(X, Y)/((H(X) + H(Y))$ [62].

It has been noted that even in case of perfect dependence these indices may not obtain a value close to zero if the relation is many-to-one. Thus, these indices

may miss out on non-monotonic relations between variables. We may also note that entropies and mutual information can be calculated directly for variables with a finite number of possible values but not for continuous variables. For such variables, we need to estimate densities using histograms or kernels. MIC is a method that uses a histogram to calculate the dependence between variables [63]. MIC does not use any of the normalizations mentioned above. Instead, it creates a data-dependent histogram scheme and uses the number bins along each dimension to normalize the mutual information. In short, it calculates the mutual information using different histogram schemes obtained with the constraint that the number of cells in the histogram is less than $n^{0.6}$ where n is the number of observations. Let the mutual information obtained using a partition G be called I_G. Let the number of cells along the X and Y axes for partition G be given by n_{Gx} and n_{Gy}. Then the dependence is calculated as $\max_G(I_G(X,Y)/\min(n_{Gx}, n_{Gy}))$. This formulation allows MIC to attain a value close to 1 for non-monotonic relations. In this regard, MIC detects a wider range of associations than indices such as HSIC and Dcor.

Some similarity indices have been designed for specific data types. For example, the Jaccard index [64,65] and the Dice coefficient [66,67] are used to find the similarity between two sets; cosine similarity measures the cosine of the angle between two vectors [68, 69].

2.2.7 Application of relationship indices for unsupervised feature selection

An important application of feature relationship indices is feature selection. The objective of feature selection methods is to identify and retain relevant features and to remove redundant ones. Feature selection methods can use a wrapper approach

in which the selection of features is based on the performance of a specific learning method [70]. On the other hand, in the filter approach feature selection is independent of the learning method [71]. Since the aim of these methods is to preserve the information content of the feature set, they often use feature relationship indices to identify redundant features [72, 73]. In the case of supervised feature selection methods, the relevance of the feature may be evaluated using the label information. Unsupervised feature selection methods do not require label information. They perform feature selection by reducing redundancy in the data. In the following paragraphs, we will briefly discuss some of the methods designed for unsupervised feature selection.

Ferreira and Figueiredo [74] proposed an unsupervised approach that combined feature discretization and selection. Continuous features are first discretized using the Linde–Buzo–Gray [75] vector quantization algorithm to reduce noise and represent the data in a compact way. Feature selection is then performed in the discretized space by identifying redundant features. A combination of hierarchical clustering and fuzzy computing has been used to identify relevant features in [76]. SVD-Entropy measures the entropy of data through singular values and has been used for feature selection [77, 78]. Feature selection has been performed by minimizing the reconstruction error for the projection matrix corresponding to the selected features [79].

One way to approach unsupervised feature selection is to use spectral clustering [80] to find a minimal set of features that spans the same subspace as the original feature set. These methods use the eigenvalues and eigenvectors of the normalized Laplacian matrix to analyze the underlying structure of the data. These methods use different regularization methods that result in different objective functions [81–84]. The USFS method performs spectral analysis of the l_1 norm graph for feature selection [85]. A recent method compares the principal axes of the subspaces spanned by the original features and the subspace spanned by selected features [86].

One of the approaches for unsupervised feature selection is to quantify the dependence between features and use it to select a set of features. Pearson correlation coefficient is one of the most popular similarity measures used for feature selection [87,88]. Indices for evaluating non-linear but monotonous feature relations include the Spearman correlation [36], Kendall rank correlation coefficient [38], and Brownian correlation [51]. These have been used for feature selection by several methods [89–91]. A popular method [92] uses the maximum variance between two features and the Spearman correlation to calculate the distance between features. A higher variance would mean that a feature is more versatile, while a smaller value of the Spearman correlation implies less redundancy. The distance matrix forms the adjacency matrix for feature selection through graph search.

FSFS performs feature selection using the feature relationship index (MICI) based on the eigenvalues of the covariance matrix for a pair of features [35]. They define a feature relationship index between two variables using the smallest eigenvalue for the covariance matrix of the two variables. Using the feature relationship index, the feature pairs with the highest similarities are identified. One feature of each pair of these is retained while the other is discarded.

In general, we find that many of the feature selection methods make use of feature relationship indices. Thus, novel and efficient feature relationship indices also help to improve feature selection methods.

2.2.8 Challenges in designing non-linear relationship indices

It has been seen that many indices work well for monotonous feature relations but not for non-monotonous ones. Although some indices such as MIC [53] are capable of detecting non-linear relations, they are computationally expensive. In addition,

the value of MIC drops quickly when the relation is noisy, which is often the case in real-world datasets. Another dependence measure that finds non-monotonous relationships is XICOR [39]. However, it is a non-symmetric dependence index. In addition, this index does not strictly lie in the range $[0, 1]$, as discussed in Section 2.2.2. Furthermore, for smooth relations such as $Y = X^2$, the sensitivity of this test decreases rapidly with the amount of noise in the data [39, 93]. In this thesis, we have presented a mutual information-based feature relationship index that can be computed quickly and performs well even in the presence of noise.

2.3 Biclustering

As we have seen in previous sections, feature relationship indices can be used to evaluate the relationship between two variables. However, it has been seen that the relationship between variables might not occur for the entire set of observations but rather for a subset of observations. The usual feature relationship indices may not be able to identify the data dependencies for a subset of data. However, such dependencies might occur for several pairs of variables. Thus, we might need to identify a set of observations for which a set of features is related, i.e., we need to cluster both rows and columns. The task of clustering both rows and columns simultaneously is called biclustering. However, the objective of biclustering is not limited to understanding feature relations [16]. Biclustering can be performed based on different criteria, e.g., minimizing the variance over all elements in a bicluster. Since biclustering is a computationally complex problem, several approaches are used to obtain biclusters according to the desired objectives [16, 27, 94–96]. Some of these objectives and approaches for biclustering are discussed in the following section.

2.3.1 Constant value biclusters

The simplest biclustering method finds biclusters of constant value by minimizing variance within a bicluster. An example of this includes the procedure called Direct Clustering [97]. Cho et al. [98] also designed a biclustering method with this objective. Since the idealized goal of these biclustering algorithms is to find biclusters with a constant value, the biclusters are convex in the subspace corresponding to the selected set of features.

2.3.2 Constant row or column biclusters

Some algorithms find biclusters based on constant rows or constant columns. BM-FM [99] finds biclusters with constant columns and uses them to forecast the trading points in the stock market. Constant row and column biclusters have been used for feature ranking and selection for classification [100]. The method BIC-K-NN [101] also uses biclustering to study patterns in the stock market.

2.3.3 Linear algebra based approaches for biclustering

Some biclustering methods use the linear algebra-based approach [15, 102]. Many of these use techniques like Singular value decomposition. Examples are MESBC [103], and Non-smooth non-negative matrix factorization [104]. The obtained biclusters can be described using linear constraints. These algorithms result in convex biclusters. They are not invariant to distance-preserving transforms.

Some authors like Luxburg [80] have modified the linear algebra-based biclustering algorithms so that non-convex biclusters can be found. This approach relies on finding linearly constrained biclusters in a high-dimensional space corresponding to a given input space. However, the location of points need not be calculated explicitly in a

high-dimensional space. MVMC [105] is a multiview clustering method in which each bicluster uses a distinct dissimilarity matrix.

2.3.4 Iterative approaches for biclustering

Many algorithms find biclusters iteratively using measures for linear relations such as the correlation coefficient, for example, BICLIC [106, 107], and ICS [108]. Procedures like coupled two-way clustering (CTWC) [107] depend on alternate grouping between observations and features. Here, the result of the previous step is used as the basis for the next phase of selection. This simple concept gives rise to an entire class of biclustering procedures, as the results will depend on the clustering procedure used, along with features and observations.

2.3.5 Statistical approaches for biclustering

Algorithms such as SAMBA [109] model the biclustering problem as a search for a heavy subgraph embedded in a bipartite graph, with two partitions representing observations and features. The Plaid model [110] analyses the input matrix as a superposition of layers of biclusters. Both the SAMBA and the Plaid models use statistical techniques to identify biclusters.

2.3.6 Use of evolutionary computing to find biclusters

Several methods make use of evolutionary computing for biclustering. BP-EBA [111] finds biclusters using evolutionary computing in two phases. Evolutionary computing-based biclustering methods have been applied to genome analysis [112, 113]. They have also been used to identify genes related to cancer [114]. Evolutionary computing techniques have also been used to solve biclustering problems such as multi-objective

optimization problems. Here, the length, breadth, and homogeneity of the biclusters are optimized. Mitra and Banka [115] proposed a popular algorithm of this category. Several algorithms like SEBI [116], BiHEA [117] and MOGAB [118] also use evolutionary computing to optimize criteria such as variance, mean squared residue, correlation, etc. TSTP [119] is a multi-objective evolutionary biclustering method that maintains three co-evolving populations: a bicluster population, a row population, and a column population. Row and column populations are used to guide the evolution and refinement of the bicluster population, improving both coherence and coverage across the data matrix. In general, biclustering can be formulated as an optimization problem by defining an objective function, and different metaheuristics like evolutionary computing, simulated annealing, etc. can be applied to solve the problem [120].

2.3.7 Deep learning based approaches for biclustering

Deep learning-based methods [121], aim to find a subspace in which observations lie close together. DPSC [122], DASC [123] and DSCN [124] learn latent space representation using deep learning.

2.3.8 Density based approaches for biclustering

Density-based biclustering algorithms, such as Subclu [20], P3C [125], FIRES [126], Proclus [127], and CLIQUE [128], find high-density regions in the corresponding subspaces. Just as DBSCAN [22] can find non-convex clusters, these algorithms can find non-convex biclusters. These methods may be capable of invariance under distance-preserving transforms, as high-density regions in the original space continue to be high-density regions after the application of order-preserving transform. These

methods start by finding dense regions in a one-dimensional space and incrementally grow biclusters using the Apriori approach. The Apriori algorithm discovers patterns by repeatedly expanding the sets of items that often occur together in the data [129]. These algorithms identify high-density regions by checking a neighborhood defined as an open disk of a fixed radius. The radius used to define the neighborhood is often a user-defined parameter. Another user-defined parameter often used by these algorithms is the minimum number of points that should be in the neighborhood of an observation. The choice of these parameters makes a huge difference in the performance of density-based methods [22]. Most of these methods determine the neighborhood using a radius that is independent of the particular set of dimensions chosen. However, the same radius might not be suitable for different dimensions. Note that determining these parameters for biclustering is inherently more complex compared to determining the parameters for clustering. For clustering procedures, there are known methods to estimate the parameters. For example, in DBSCAN the minimum number of points in a disk can be estimated from a plot of the K nearest neighborhood distances. However, similar techniques cannot be applied to biclustering methods, which build biclusters using density-based methods in a bottom-up fashion, as the set of dimensions for different clusters varies.

The method named Curler [130] finds non-linear clusters by combining a density-based approach with the principal component [131, 132] method. Curler finds high-density Gaussian regions using Expectation Maximization [133, 134] and combines them using the relationship between the directional information of the resultant biclusters. In general, the density-based methods mentioned above find biclusters based on the proximity of observations in a subspace.

2.3.9 Kernel based biclustering methods

Kernel based methods have also been applied to biclustering. The mean shift method [135] uses the kernel density estimation and iteratively locates the local maxima of the kernel mixture. Inspired by the energy distance discussed in Section 2.2.4 a biclustering technique has been proposed to cluster observations in separable Hilbert spaces [136].

2.3.10 Feature relationship based biclustering

Some algorithms find biclusters based on sets of observations where there is a direct relation between two features. In other words, for a selected set of observations, values for one feature can be calculated by applying a simple function to values of another feature. This is equivalent to searching for submatrices in the data where features are related on each other. One such example is the algorithm proposed by Cheng and Church [137]. The method named Robust Convex Biclustering [138] formulates biclustering as a convex optimization problem to group similar rows and columns simultaneously, resulting in robust and convex biclusters.

Many methods like FABIA [139] search for biclusters based on the multiplicative model, also known as the scaling pattern. Here, one row is a multiple of another. When we are looking for specific relations between rows or columns, the resulting algorithm will not be invariant to distance-preserving transforms. Few authors [140, 141] provide algorithms to search for both shifting and scaling.

Some algorithms are designed to find co-regulated clusters. ISA [142] starts with a subset of genes and iteratively identifies columns for which the values are consistently high or low, resulting in co-regulated clusters. The algorithm OPSM [143] finds biclusters by finding a permutation of columns such that the sequence of values in each

row increases. In other words, it finds order-preserving submatrices. QUBIC [144] finds biclusters in which the rows are correlated by first converting the input matrix to integers and then checking if the integer values are the same.

Relationship-based biclustering methods have been developed to find biclusters where features are related to each other in arbitrary ways. These include algorithms like UniBic based on the Longest Common Subsequence. UniBic [145] identifies the common subsequence of observations that increase or decrease monotonically between pairs of features. Sequences are considered only if they are long enough so that random occurrences of such a long sequence is unlikely. UniBic then uses these long sequences as seeds to add rows and columns to form large biclusters. Another method, ARBic [146], measures trend-preserving patterns between rows as paths in directed acyclic graphs. Then it combines them to find biclusters. Similarly, TransBic [147] also finds generalized trend-preserving biclusters using acyclic graphs. However, it uses a relaxed approach, where trend-preserving pattern is seen between groups of features. This makes TransBic robust in the presence of noise. These algorithms can find non-convex biclusters. In general, algorithms based on an arbitrary relation between rows or columns can result in non-convex biclusters.

2.3.11 Challenges in biclustering

Most of the existing biclustering algorithms use several parameters. The appropriate values of the parameters can change from one data set to another, making its application to diverse real-world data sets difficult [148]. Some biclustering methods have been developed to overcome this problem [103, 148, 149]. An approach used by researchers to develop parameter-free biclustering methods is to use hierarchical methods. This includes the work done to structure the data in coupled hierarchies

of observations and features [150]. A parameter-free recommender system [149] has also been built using hierarchical biclustering.

For spectral clustering, several researchers have analyzed the multiplicities of eigenvalues close to zero [80] to determine the number of spectral clusters. The method named MESBC finds mutually exclusive biclusters and only requires the number of clusters in the dataset as input [103]. ROCCO [148] is a robust biclustering method that uses a mutual K-Nearest Neighbor graph to perform biclustering, with minimal parameter tuning. Although K is user-defined, the method has been shown to perform well when the value of K lies in a particular range.

Further, most of the existing feature relation-based biclustering methods are designed to find biclusters using monotonous feature relations. As mentioned earlier, in this thesis, three density-based biclustering methods, named CBSC, RelDenClu, and PF-RelDenBi, are presented. These methods are capable of finding biclusters based on non-linear and non-monotonous relations. Of these, PF-RelDenBi is a parameter-free method. It leverages a feature relationship index named MIDI to perform non-linear feature relation-based biclustering. As mentioned earlier, MIDI which is presented in the following chapter, is a mutual information-based dependence index. It allows PF-RelDenBi to find non-linear feature relation-based biclusters.

Chapter 3

MIDI: A new estimate of mutual information-based dependence measure between two variables

3.1 Introduction

In the modeling of complex systems, it is frequently observed that non-linear dependencies manifest between variables. In certain instances, these relationships can be expressed through a functional representation. However, there are cases where the relationship cannot be adequately represented by functional forms. For example, in biological systems and weather forecasting, the behavior of variables may seem random, but repetition of a fixed random pattern may reveal a relationship between variables [151]. Even if the relationship can be represented using a simple functional form, it is impractical to test against each potential functional form individually, given that there are infinitely many functional forms.

3.1.1 Some methods commonly used to measure dependence between variables

As discussed in the previous chapter, there are several measures of dependence between two variables in the literature. The correlation coefficient (ρ) is the most popular dependence measure. The correlation coefficient possesses several advantageous properties; however, it is limited to capturing only linear relationships between variables. A value of zero for the coefficient does not necessarily indicate the independence of the variables. Therefore, we need an index that can evaluate non-linear relationships between two features, without relying on any assumed theoretical probability distribution.

In the literature on non-linear feature relationship indices, some authors focused on the properties that an index should possess. Studies by Urbach [152] and Dionisio et al. [153] support a strong relationship between entropy, dependence, and predictability. This relation has been studied by several authors [60, 63, 154–157]. The dependence measure called MIC [63] uses mutual information to evaluate feature relationships. It is capable of quantifying the dependence between variables that exhibit different types of relationships. Kvålseth [59] has briefly defined an information-theoretic measure of dependence between two variables in his paper. This measure has several desirable properties. However, he has not discussed how this measure can be estimated for a given set of points. A more comprehensive discussion of existing mutual information-based measures and their methods of estimation is presented in Section 3.1.2.

Other authors provided different dependence measures which were extensively discussed in Chapter 2. These include kernel-based methods, rank-based methods, and energy-distance-based methods to find dependence between features. Distance corre-

lation [51] is an energy-distance-based dependence measure that has several desirable properties but does not detect relationships such as sinusoidal waves, circles, etc. where there is a one-way relationship [158].

In this chapter, we present a method named MIDI (Mutual Information based Dependence Index) for calculating mutual information-based dependence between features. To calculate MIDI, we need to estimate mutual information and entropy. This estimation is done using density estimates obtained from the histogram. It is well-known that the consistency of histogram-based density estimation depends on the bin size. MIDI calculates the bin size using simple statistical properties of the data, which results in consistent density estimates. Consequently, this index can efficiently find the dependence between variables related to each other in different ways without overestimating the dependence.

The definitions of Mutual Information and Entropy were discussed in the section 2.2.6 of Chapter 2. As mentioned above, calculating mutual information for continuous features is not a straightforward task. In the following paragraphs, we discuss some of the existing methods for calculating mutual information-based dependence measures.

3.1.2 Existing methods for estimating mutual information and entropy based feature relationship Indices

Given the unknown nature of the underlying probability distributions, it is necessary to develop techniques to estimate the mutual information based feature relationship indices. This estimation can be achieved using either a non-parametric approach or a parametric method. Parametric methods need a specific form of stochastic processes [156], so they cannot be generalized to datasets with arbitrary distributions.

One of the existing non-parametric methods for estimating mutual information uses the lengths of the edges of the nearest-neighbor graph [159]. However, this method sometimes gives very small negative values, which is not desirable since the estimated value of the dependence measure should be non-negative.

One of the most popular methods for non-parametric density estimation is histogram-based density estimation. A method that uses histograms for density estimation in order to calculate the dependence is MIC [63]. In this approach, equiprobable partitions are created along one axis, with the opposing axis being segmented in such a manner that the value of the measure is maximized. The number of partitions is constrained by $n^{0.6}$, where n denotes the number of data points in the dataset. This procedure is repeated by interchanging the axes, after which the greater value of the two is selected. An intricate methodology is used to determine the length of the bin for the computation of mutual information. The following paragraph explains how the MIC value is calculated using different partitions.

Suppose that mutual information calculated using a particular partition G is denoted by I_G. Let the number of cells along the X and Y axes for the grid G be given by n_{Gx} and n_{Gy}. Then the dependence is calculated as:

$$MIC(X,Y) = \max_{G}\left(\frac{I_G(X,Y)}{\min\left(n_{Gx}, n_{Gy}\right)}\right) \tag{3.1}$$

Note that this process of finding the bin length is computationally expensive. As the partition is chosen with a preference for higher estimate values, the value achieved for independent variables is relatively high, and therefore it generally overestimates the dependency value [158].

In his paper, Kvålseth [59] described three measures to evaluate relationships

between features. They are given below.

$$KM1(X,Y) = \frac{I(X,Y)}{\min(H(X), H(Y))} \tag{3.2}$$

$$KM2(X,Y) = \frac{I(X,Y)}{\max(H(X), H(Y))} \tag{3.3}$$

$$KM3(X,Y) = \frac{2 * I(X,Y)}{(H(X) + H(Y))} \tag{3.4}$$

Among these, KM2 has been used for clustering [157,160]. KM3 has been widely used for feature selection [161]. Kvålseth [59] has mentioned that the value of all three measures is 1 in the case of a strict one-to-one association between the variables. However, KM1, achieves the value 1 even when the relationship between the features is many-to-one, making it suitable for finding non-monotonous relationship between features. It can be shown that KM1 attains a value of 1 if there is a strict one-way association, that is, either of the variables is a function of the other. This allows us to detect functional forms that have one-way dependence, for example $y = x^2$, $y = \sin(x)$. Though Kvålseth [59] describes KM1 briefly, no method has been proposed to estimate the defined measure.

In this chapter, we present a fast and efficient method to estimate KM1, resulting in an index MIDI, that can be used to assess non-linear relationships between variables quickly. MIDI is calculated using density estimation based on histograms. We have used a partition scheme that leads to better performance in the case of noisy data. The partitioning is done using the statistical properties of the data which can be easily calculated. No optimization is required over different ways of partitioning same data. This makes the presented method efficient in terms of execution time and memory. Consequently, it can be used for large volumes of data where the application of many existing methods becomes infeasible due to time and memory constraints.

3.2 MIDI: properties and implementation

3.2.1 Dependence measure estimated in MIDI: Properties and their implications

In the previous paragraphs, we have mentioned that the presented method, MIDI is an estimate of mutual information normalized by minimum entropy. In order to calculate MIDI, the values of $I(X,Y), H(X)$ and $H(Y)$ are estimated using a new density estimation method described in Section 3.2.2. The theoretical reasons supporting the use of the histogram scheme for the density estimation are discussed in Sections 3.2.3 and 3.2.4.

The resulting density estimates are then used to calculate an estimate of Mutual Information normalized by the minimum of the entropies of the two variables. The resulting relationship has been named Mutual Information Based Dependence Index (MIDI) and is given as:

$$MIDI = \frac{I(X,Y)}{min(H(X), H(Y))} \tag{3.5}$$

To understand the properties [162] of MIDI we recall some properties of mutual

information and entropy:

$$I(X,Y) \geq 0 \tag{3.6}$$

$$I(X,Y) = 0, \textit{ iff X, Y are statistically independent} \tag{3.7}$$

$$I(X,Y) = H(X), \textit{ iff X is a function of Y} \tag{3.8}$$

$$I(X,Y) = H(Y), \textit{ iff Y is a function of X} \tag{3.9}$$

$$I(X,Y) \leq min(H(X), H(Y)) \tag{3.10}$$

From equations 3.5 and 3.10, it can be seen that MIDI achieves a value of 1 only when X is a function of Y or Y is a function of X or both. Since $I(X,Y) = 0$ holds true if and only if the variables X and Y are statistically independent, MIDI obtains a value of 0 under conditions of statistical independence. The value of MIDI always lies between 0 and 1. Given that the density estimates discussed are demonstrated to be L_1-consistent in the subsequent sections, it can be inferred that the value of MIDI approaches 1 when there exists a perfect relationship (possibly one-way) between the features and 0 when the variables are entirely independent, as the sample size increases. Consequently, MIDI possesses the following attributes:

$$0 \leq MIDI \leq 1 \tag{3.11}$$

$$MIDI(X,Y) = 0 \quad \textit{, iff X, Y are statistically independent} \tag{3.12}$$

$$MIDI(X,Y) = 1 \quad \textit{, iff X is a function of Y or if Y is a function of X} \tag{3.13}$$

In this chapter, the entropy and mutual information of variables are estimated by dividing each axis into bins. The proportion of the number of points lying in the bin to the total number of points is used as the value of the probability mass function for

the points lying in the bin. Note that mutual information and entropy for discrete random variables are invariant to scaling. Also, as we shall see, scaling does not affect how the points are partitioned by MIDI. Thus, MIDI is invariant to scaling. The following section provides a detailed procedure to calculate MIDI.

3.2.2 Determining bin length and calculating dependence

If the histogram approach for density estimation is used, the length of the bins becomes a very important parameter that affects the value of the estimate [153]. Should the bin length ϵ be excessively large, the index becomes ineffective in capturing nonlinear relationships, as it will be unable to establish a unique mapping between a pair of axes. So, for a set of n points, $\lim_{n\to\infty} \epsilon \to 0$ is required. However, if the bin length is too small, the distribution will not seem to be continuous. As the number of data points increases, it is advisable to reduce the length of the bins accordingly. However, if the reduction in the length of the bin occurs too rapidly with increasing n, it is likely that numerous partitions remain unoccupied. Parzen [132], Lugosi and Nobel [163] have given criteria for the selection of the appropriate bin size. This chapter introduces a method for determining the suitable bin length, which is both computationally efficient and yields an estimated index value that approaches zero quickly when the variables are independent. Moreover, the resulting estimate demonstrates reduced susceptibility to noise.

The subsequent sections present a methodology for computing the dependence between two features. The entropy and mutual information of the associated variables are determined by dividing each axis into bins. The bin length on one axis is calculated using a function of the maximal spacing along that axis, where the maximal spacing for an axis is defined as the maximum distance between any two consecutive points.

On the other axis, the number of bins is proportional to the logarithm of the sample size. The ratio of the count of observations contained within a bin to the overall number of observations serves as the value of the probability mass function for the observations located within that bin. These estimates are then used to calculate the mutual information and the entropy along both dimensions, which are used to calculate the value of $MIDI$. The process is repeated by defining the bin length as a function of the maximal spacing on the second axis and as a function of the logarithm of the data sample size on the first axis. The greater of the two computed values is selected as the dependence index. The theoretical properties of the density estimation method employed are further elaborated in Section 3.2.4.

3.2.2.1 Method for determining the bin length

This section presents the methodology for calculating the length of the bin. Theoretical reasons behind this choice is discussed in Sections 3.2.3 and 3.2.4. In this chapter, we use the maximal spacing [164] to determine the length of the bin along one dimension. The maximal spacing is also called the connectivity distance. The following paragraphs define the maximal spacing and present its use for finding the bin length.

This paragraph states the conditions under which the presented scheme provides consistent density estimates. Let the given data be $\{(x_1, y_1), \cdots, (x_n, y_n)\} \subset \mathbf{R}^2$. We assume that the points are drawn i.i.d. from the set B following a continuous probability density function f, which is unknown. Let B be path connected, compact, $Closure(Interior(B)) = B$ and boundary of B, denoted by $\delta(B)$ is such that $\lambda(\delta B) = 0$, λ is the Lebesgue measure. Let the support of f be B, that is, $f(x) = 0\ \forall x \in B'$, where B' is the complement of B and $\int_v f(x)dx > 0\ \forall v$, v open and $v \cap B \neq \phi$.

The details of the method to find the bin length is as follows. The data is sorted

along the x-axis. Let the sorted values be $x_{(1)} \cdots x_{(n)}$. The maximum difference between consecutive ordered points be $L_{max} = \sup\{x_{(i+1)} - x_i : 1 \leq i \leq n-1\}$, where n is the number of points in the sample. L_{max} is the maximal spacing along the x axis. The length of the bin along the x-axis is calculated as $n^c L_{max}$, where, c is a constant such that $0 < c < 1$. For a given fixed sample size n, as the parameter c approaches the value 0, the bin lengths are reduced. Consequently, the number of bins increases, which improves the resolution of the data representation. This increased granularity results in a higher likelihood of obtaining unique mappings, thereby yielding a higher value as an estimate of the dependence measure. Along the y-axis, the number of divisions is given by $log_{10}(n)$. Subsequent to determining the bin length, the computation of the dependence measure can be performed as described in Section 3.2.2.2. The procedure should be repeated with the x and y axes interchanged and the higher of the two resulting estimates should be considered as the final value.

Figure 3.1 shows the above-mentioned scheme to find the length of the bin. The figure shows 500 observations having uniform distribution on the x-axis, and the values on the y-axis are given as $y = 4(x - 0.5) + \text{noise}$, where the noise has a Gaussian distribution with 0 mean and 0.01 standard deviation. The figure shows the projections of the observations on the x-axis and the y-axis as red and green dots, respectively. The maximal spacing on the x-axis is shown as the distance between two red lines, and on the y-axis as the distance between two green lines. On the right side of the image we see the two partitions used to calculate MIDI using maximal spacing on one axis and $log_{10}n$ partitions on the other. The maximum of the values calculated using both partitions is taken as the final value of MIDI which turns out to be 0.98.

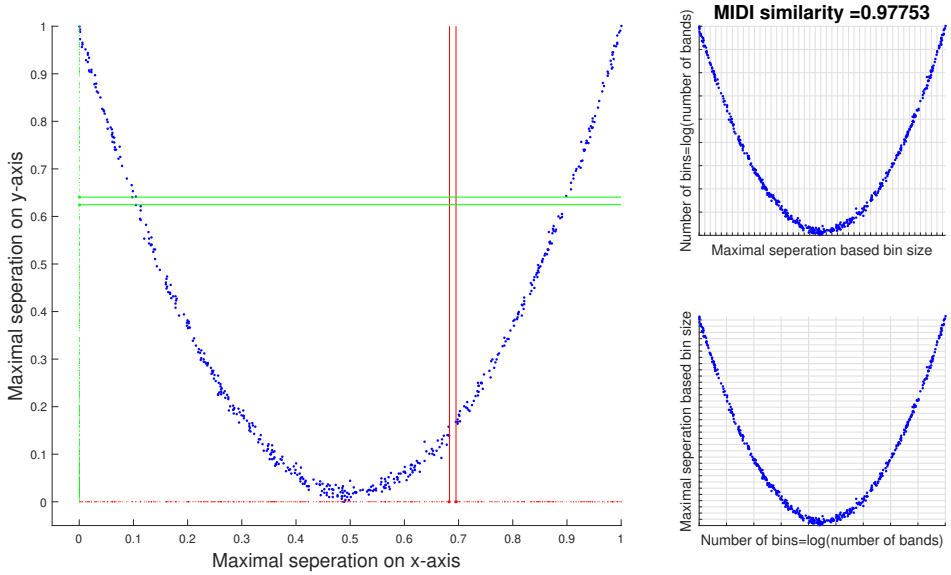


Figure 3.1: Histogram technique used by MIDI to obtain density estimates

3.2.2.2 Calculating mutual information based dependence measure: Calculating MIDI

After determining the bin lengths, we can calculate the value of MIDI as described below. The origin of the interval is taken as a starting point for the histogram calculation. The mutual information according to densities estimated using the above-mentioned scheme is given by

$$\hat{I} = \sum_{i \in \pi_x} \sum_{j \in \pi_y} \frac{\#_{i,j}}{n} \log \left[\frac{n \#_{i,j}}{\#_i \#_j} \right] \tag{3.14}$$

where π_x and π_y are partitions along x and y directions, respectively. The number of points in the cell given by the intersection of the interval i along the x-axis and j along the y axis is given by $\#_{i,j}$. The number of points in the interval i along the x-axis is given by $\#_i$. Similarly, the number of points in the interval j along the y-axis is given by $\#_j$. The entropy values along the x and y axes are calculated similarly

to $\hat{H}_x = \sum_{i \in \pi_x} \frac{\#_i}{n} \log(\frac{n}{\#_i})$ and $\hat{H}_y = \sum_{j \in \pi_y} \frac{\#_j}{n} \log(\frac{n}{\#_j})$. $MIDI$ is now calculated as $\hat{I}/min(\hat{H}_x, \hat{H}_y)$.

It should be noted that the proportion of maximal spacing between points on a given axis to the range of that axis remains unaffected by variations in scale; thus, the division of each axis exhibits invariance to scaling. Consequently, MIDI is characterized by its invariance under scaling transformations. Likewise, the structure of the bins and the estimate provided are also invariant under translational transformations. Under the aforementioned conditions, the estimate's value remains unchanged. However, the estimate is not invariant to any transform that is not order-preserving. In the following sections, we elucidate the rationale for selecting partitions in the manner previously delineated.

3.2.3 Theorems used to design the methodology for determining bin length

This section discusses the conditions required for consistent density estimates using a histogram scheme and the asymptotic behavior of maximal spacing. Existing theorems that clearly state the required conditions for consistency have been presented here. Theorems on the behavior of maximal spacing have also been included for completeness. In the next section, these will be used to prove that the bin lengths used by MIDI lead to consistent density estimates.

Theorem 3.1 presents the necessary conditions for consistency and was proposed by Lugosi and Nobel [163]. Theorems 3.2 by Darling [165], and Theorems 3.3, 3.4 by Deheuvels [166], describe the asymptotic behavior of maximal spacing. These theorems have been used to prove that the histogram scheme used for MIDI leads to consistent density estimates. This is given in the next section (Section 3.2.4).

3.2.3.1 $L-1$ consistency of density estimates

The first theorem provides the required conditions for $L-1$ consistent density estimates for any histogram scheme. The following notations have been used for the theorems presented in this chapter. Let $\mathbf{R}^d$ denote the d-dimensional Euclidean space. By a partition of $\mathbf{R}^d$ we mean a finite collection $\pi^* = \{\mathring{A}_1, \cdots \mathring{A}_q\}$ of Borel-measurable subsets of $\mathbf{R}^d$, called cells, with the property that (i) $\bigcup_{j=1}^{q} \mathring{A}_j = \mathbf{R}^d$ and (ii) $\mathring{A}_i \cap \mathring{A}_j = \emptyset$, if $i \neq j$. Let $|\pi^*|$ denote the number of cells in π^*.

For a set consisting of n i.i.d. random variables $\vec{X}_1, \cdots, \vec{X}_n$ taking values in a set $\mathcal{X} = \mathbf{R}^d$, a partitioning rule and the family of partitions associated with a partitioning rule can be defined as follows. An n-sample partitioning rule for $\mathbf{R}^d$ is a function π_n that associates every n-tuple $(x_1, \cdots x_n) \in \mathcal{X}^n$ with a measurable partition of $\mathbf{R}^d$. Applying rule π_n to $\vec{X}_1, \cdots, \vec{X}_n$ produces a random partition $\pi_n(\vec{X}_1, \cdots, \vec{X}_n)$. A partition scheme for $\mathbf{R}^d$ is a sequence of partitioning rules $\Pi = \pi_1, \pi_2, \cdots$. Associated with every rule π_n there is a fixed, non-random family of partitions ${\mathcal{A}^*}_n = \pi_n(\vec{X}_1, \cdots, \vec{X}_n) : \vec{X}_1, \cdots, \vec{X}_n \in \mathcal{X}$. Thus, every partition scheme Π is associated with a sequence ${\mathcal{A}^*}_1, {\mathcal{A}^*}_2, \cdots$ of partition families. In the following paragraphs, the random partitions $\pi_n(\vec{X}_1, \cdots, \vec{X}_n)$ are simply denoted by π_n. An ordered sequence $x_1, \cdots, x_n \in \mathbf{R}^d$ will be denoted by x_1^n.

Let $\mathcal{A}^*$ be a family (possibly infinite) of partitions of $\mathbf{R}^d$. The maximum cell count of $\mathcal{A}^*$ is given by $m(\mathcal{A}^*) = \sup_{\pi^* \in \mathcal{A}^*} |\pi^*|$. The complexity of $\mathcal{A}^*$ will be measured by a combinatorial quantity similar to the well-known growth function for classes of sets that was given by Vapnik and Chervonenkis [167]. Fix n points $x_1, \cdots, x_n \in \mathbf{R}^d$ and let $B = \{x_1, \cdots, x_n\}$. Let $\Delta(\mathcal{A}^*, (x_1, \cdots, x_n))$ be the number of distinct partitions of the set B. Note that $\{\mathring{A}_1 \cap B, \cdots, \mathring{A}_q \cap B\}$ is a distinct partition of the finite set B induced by a partition $\{\mathring{A}_1, \cdots, \mathring{A}_q\} \in \mathcal{A}$. It is easy to see that

$\Delta(\mathcal{A}^*, (x_1, \cdots, x_n)) \leq m(\mathcal{A}^*)^n$. The growth function of $\mathcal{A}^*$ is defined as $\Delta_n^{\bigstar}(\mathcal{A}^*) = \max_{(x_1,\cdots,x_n)\in\mathbf{R}^{n\times d}} \Delta(\mathcal{A}^*, (x_1, \cdots, x_n))$ is the largest number of distinct partitions of any n point subset of $\mathbf{R}^d$ that can be induced by the partitions in $\mathcal{A}^*$. In other words, it is the maximum number of ways in which any set of n fixed points can be partitioned.

With this convention in mind, for every $x \in \mathbf{R}^d$ let $\pi_n[x]$ be the unique cell of π_n that contains the point x. Let A be any subset of $\mathbf{R}^d$. The diameter of A is the maximum Euclidean distance between two points of A, $diam(A) = \sup_{x,y\in A} ||x - y||$. Let μ be a probability measure on $\mathbf{R}^d$ having density f, so that $\mu(A) = \int_A f(x)dx$ for every Borel subset A of $\mathbf{R}^d$. Let $\vec{X}_1, \vec{X}_2, \cdots$ be random vectors with i.i.d. in $\mathbf{R}^d$, each distributed according to μ, and let μ_n be the empirical distribution of $\vec{X}_1, \cdots, \vec{X}_n$. Fix a partition scheme $\Pi = \pi_1, \pi_2, \cdots$ for $\mathbf{R}^d$, where the rule π_n is applied to a n-tuple. Applying the n^{th} rule in Π to $\vec{X}_1, \cdots, \vec{X}_n$ produces a partition $\pi_n = \pi_n(\vec{X}_1, \vec{X}_2, \cdots, \vec{X}_n)$ of $\mathbf{R}^d$. The partition π_n, in turn, gives rise to a natural histogram estimate of f as follows. For each vector $x \in \mathbf{R}^d$ let

$$f_n(x) = \mu_n(\pi_n[x])/\lambda(\pi_n[x]) \text{ } \textit{if } \lambda(\pi_n[x]) < \infty \tag{3.15}$$

$$= 0 \text{ } \textit{otherwise} \tag{3.16}$$

Here λ denotes the Lebesgue measure on $\mathbf{R}^d$. Note that f_n is itself a function of the training set $\vec{X}_1, \cdots \vec{X}_n$ and that f_n is piecewise constant on the cells of π_n. The sequence of estimates f_n is said to be *strongly L_1-consistent* if $\int |f(x)-f_n(x)|dx \to 0$ with probability one as $n \to \infty$. Theorem [163] is stated below using the notation mentioned above.

Theorem 3.1. *Let $\vec{X}_1, \vec{X}_2, \cdots, \vec{X}_n$ be random vectors with i.i.d. in $\mathbf{R}^d$ whose common distribution μ has a density f. Let $\Pi = \{\pi_1, \pi_2, \cdots\}$ be a fixed partition scheme*

for $\mathbf{R}^d$*, and let* ${\mathcal{A}^*}_n$ *be the collection of partitions associated with the rule* π_n*. As n tends to infinity,*

$$n^{-1}m({\mathcal{A}^*}_n) \to 0 \tag{3.17}$$

$$n^{-1}\log \Delta_n^{\bigstar}({\mathcal{A}^*}_n) \to 0 \tag{3.18}$$

$$\mu\{x : diam(\pi_n[x]) > \epsilon\} \to 0 \tag{3.19}$$

with probability one for every $\epsilon > 0$*, then the density estimates* f_n *are strongly consistent in L1:*

$$\int |f(x) - f_n(x)|dx \to 0 \tag{3.20}$$

with probability one.

3.2.3.2 Asymptotic behavior of maximal spacing

The following theorems describe the asymptotic behavior of maximal spacing. First, the specific case for uniform distribution is stated. This is followed by a more generalized case of arbitrary distributions that meet the conditions mentioned in Section 3.2.2.1. The notation introduced here has also been used in Section 3.2.4 for the theorems proposed in this work.

Let $\{U_k\}$, $1 \leq k \leq N$ be uniform i.i.d., in [0, 1], ordered to give $\{U^*_{N,k}\}$ We write $U^*_{N,0} \equiv 0$, $U^*_{N,N+1} \equiv 1$. Then $\gamma_j = U^*_{N,j+1} - U^*_{N,j}$ for $0 < j <_N$ are called uniform spacings. Let $M_n = \max \gamma_j$ be the largest uniform spacing. Darling [165] proved following theorem:

Theorem 3.2. *Almost surely as* $n \to \infty$,

$$|M_n - \log(n)/n| = O(\log(\log(n))/n) \tag{3.21}$$

It has been specified that this implies for a constant C *as* $n \to \infty$,

$$|M_n - \log(n)/n| = C(\log(\log(n))/n) \tag{3.22}$$

The following two theorems given by Deheuvels [166] establish the upper limit for maximal spacing in the case of different distributions.

Theorem 3.3. *Let* $\vec{X}_1, \vec{X}_2, \cdots$ *be an i.i.d. sequence of random variables with a continuous distribution function* F. *Let* $\vec{X}_{1,n} < \vec{X}_{2,n} < \cdots < \vec{X}_{n,n}$ *denote the order statistics of* $\vec{X}_1, \vec{X}_2, \cdots, \vec{X}_n$, *and let* $S_i^{(n)} = \vec{X}_{i+1,n} - \vec{X}_{i,n}$, $i = 1, 2 \cdots, n-1$ *define the corresponding spacings. Denote the order statistics of* $S_1^{(n)}, \cdots, S_{n-1}^{(n)}$ *by* $M_{n-1}^{(n)} < \cdots < M_2^{(n)} < M_1^{(n)}$. *If the following conditions are satisfied*

1. $F(x) = P\{X \leq x\}$ *has a continuous first derivative* $f(x) > 0$ *on* (A_{lm}, B_{lm}) *where* $A_{lm} = inf\{x; F(x) > 0\} < B_{lm} = \{x; F(x) < 1\}$.

2. *The distribution* F *has a bounded support* $(-\infty < A_{lm} < B_{lm} < +\infty)$, *and there exists an* $x_0 \in (A_{lm}, B_{lm})$ *such that, for all* $x \in (A_{lm}, B_{lm})$, $x \neq x_0$, $f(x) > f(x_0) > 0$.

3. *There exists a* q, $0 < q < \infty$, *such that*

$$\liminf_{h \downarrow 0} \frac{f(x_0 + h) - f(x_0)}{|h|^q} = d_q,\ 0 < d_q < +\infty \tag{3.23}$$

Then for any $t \geq 5$, $k \geq 1$ *and* $\epsilon > 0$,

$$P(\limsup_{n\to\infty}\{nM_k^{(n)}f(x_0) > \log(n) - (1/q)\log_2(n) \tag{3.24}$$

$$+1/k(2\log_2(n) + \log_3(n) + \cdots + \log_{t-1}(n) \tag{3.25}$$

$$+(1+\epsilon)\log_t(n))\}) = 0 \tag{3.26}$$

Theorem 3.4. *In addition to conditions for Theorem 3.3 if the following condition is satisfied*

1. *There exists an* q, $0 < q < \infty$, *such that*

$$\limsup_{h\downarrow 0}\frac{f(x_0+h)-f(x_0)}{|h|^q} = D_q,\ 0 < D_q < +\infty \tag{3.27}$$

and $0 < d_q \leq D_q < +\infty$ *Then, we have, almost surely,*

$$\limsup_{n\to\infty}\frac{nM_k^{(n)}f(x_0)-\log(n)}{\log_2(n)} = \frac{2}{k} - \frac{1}{q} \tag{3.28}$$

3.2.4 Lemmas proposed to show the L_1 consistency of density estimates of the scheme used in MIDI

For consistency of density estimates using the histogram method, the length of bins is a very important parameter. General sufficient conditions for the strong $L_1 - consistency$ of histogram density estimates based on data-dependent partitions have been discussed by researchers [163]. The relevant theorem given by Lugosi and Nobel has been reproduced as Theorem 3.1 in Section 3.2.3. In each case, the desired consistency requires shrinking cells, subexponential growth of a combinatorial complexity measure, and sublinear growth of the number of cells. Under the assumptions

of the theorem, $L_1 - error$ of the normalized partitioning density estimates converges to zero with probability one.

It has been seen from the Lemmas 3.1 and 3.2 given below that the number of cells in sublinear and the combinatorial complexity is subexponential as $n \to \infty$ for the presented method. The diameter of each cell is also shown to shrink to 0 as $n \to \infty$, as shown by Lemmas 3.3 and 3.4. Let ${\mathcal{A}^*}_n$ denote the family of partitions created for n observations using the histogram scheme presented in Section 3.2.2. Note that the maximal spacing L_{max} defined in Section 3.2.2 for a set of n observations will be given as $M_1^{(n)}$ in Theorem 3.3.

Lemma 3.1. *For the histogram scheme used by MIDI, taking a fixed c such that $0 < c < 1$,*

$$n^{-1} m({\mathcal{A}^*}_n) \to 0, \text{ as } n \to \infty \tag{3.29}$$

Proof. In the presented scheme, the smallest possible value of L_{max} will be attained when all points are equidistant to each other along the x-axis. In this case, the value for L_{max} will be $g_x/(n-1)$, where g_x is the range along the x-axis. In this case, the number of divisions along the x axis will be $\frac{g_x}{n^c.L_{max}} = \frac{(n-1)}{n^c}$. The number of divisions along the y axis is given as $log_{10}(n)$. So, the maximum number of cells in a given scheme is $\frac{(n-1)log_{10}(n)}{n^c}$ where $0 < c < 1$.

$$m({\mathcal{A}^*}_n) \leq \frac{(n-1)log_{10}(n)}{n^c} \tag{3.30}$$

$$n^{-1} m({\mathcal{A}^*}_n) \leq \frac{(n-1)log_{10}(n)}{n^{(c+1)}} \tag{3.31}$$

$$< \frac{n.log_{10}(n)}{n^{(c+1)}} \tag{3.32}$$

$$= \frac{log_{10}(e).log(n)}{n^c} \tag{3.33}$$

Note that $\frac{log(n)}{n^c} \to 0$ for a fixed $c > 0$ for $n \to \infty$

$$\implies n^{-1} m(\mathcal{A}^*{}_n) = 0 \ , as\ n \to \infty \tag{3.34}$$

□

Lemma 3.2. *For the histogram scheme used by MIDI, taking a fixed c such that* $0 < c < 1$,

$$n^{-1} \log(\Delta_n^{\bigstar}(\mathcal{A}^*{}_n)) \to 0 \ as\ n \to \infty \tag{3.35}$$

Proof. The maximum number of divisions along the x-axis is given by $\frac{(n-1)}{n^c}$. Note that $\frac{(n-1)}{n^c} < n^{1-c}$. Let $\lceil x \rceil$ denote the ceiling of positive real number x. So, the maximum number of distinct partitions along the x axis is bounded by $\binom{\lceil n+n^{1-c} \rceil}{\lceil n^{1-c} \rceil}$. Similarly, with the maximum number of divisions along the y-axis being $log_{10}(n)$, the maximum number of distinct partitions along the y-axis is bounded by $\binom{\lceil n+log_{10}(n) \rceil}{\lceil log_{10}(n) \rceil}$. So, the maximum number of distinct partitions considering both dimensions, $\Delta_n^{\bigstar}(\mathcal{A}^*{}_n)$ is bounded by the product $\binom{\lceil n+log_{10}(n) \rceil}{\lceil log_{10}(n) \rceil}\binom{\lceil n+n^{1-c} \rceil}{\lceil n^{1-c} \rceil}$, where $0 < c < 1$.

$$\begin{aligned} n^{-1} \log(\Delta_n^{\bigstar}(\mathcal{A}^*{}_n)) &\leq n^{-1} \log \left[\binom{\lceil n + \log_{10}(n) \rceil}{\lceil \log_{10}(n) \rceil} \binom{\lceil n + n^{1-c} \rceil}{\lceil n^{1-c} \rceil} \right] \\ &= \frac{1}{n} \log \left[\binom{\lceil n + log_{10}(n) \rceil}{\lceil log_{10}(n) \rceil} \right] + \frac{1}{n} \log \left[\binom{\lceil n + n^{1-c} \rceil}{\lceil n^{1-c} \rceil} \right] \end{aligned} \tag{3.36}$$

For any given $\mathring{n}$ and $\mathring{k} < \mathring{n}$, the upper bound of $\log \left[\binom{\mathring{n}}{\mathring{k}} \right]$ is $\mathring{n} H(\mathring{k}/\mathring{n})$ [168] , where $H(\epsilon)$ represents the binary entropy of ϵ, defined by $H(\epsilon) = -(1-\epsilon)\log(1-\epsilon) - (\epsilon)\log(\epsilon)$.

It follows that

$$n^{-1}\log(\Delta_n^{\bigstar}({\mathcal{A}^*}_n)) \leq \frac{n+\log_{10}(n)}{n} H(\frac{\log_{10}(n)}{n+\log_{10}(n)}) + \frac{n+n^{1-c}}{n} H(\frac{n^{1-c}}{n+n^{1-c}}) \quad (3.37)$$

$$\text{However, as } n \to \infty,\ \frac{\log_{10}(n)}{n+\log_{10}(n)} \to 0 \quad (3.38)$$

$$\text{As } n \to \infty, \frac{n^{1-c}}{n+n^{1-c}} \to 0, \text{ where } 0 < c < 1 \quad (3.39)$$

$$\text{As } n \to \infty,\ \frac{n+\log_{10}(n)}{n} \to 1 \quad (3.40)$$

$$\text{As } n \to \infty, \frac{n+n^{1-c}}{n} \to 1, \text{ where } 0 < c < 1 \quad (3.41)$$

Note that H is increasing on $(0, 1/2]$, H is symmetric about $1/2$, and $H(\epsilon) \to 0$ as $\epsilon \to 0$. Therefore, the entropies of terms given in (3.38) and (3.39) vanish as $n \to \infty$. So both terms on the right-hand side of inequality (3.37) vanish as $n \to \infty$. We find $n^{-1}\log(\Delta_n^{\bigstar}({\mathcal{A}^*}_n)) = 0$ as $n \to \infty$.

□

The Lemmas 3.3 and 3.4 show that the diameter of each cell shrinks to 0 as $n \to \infty$ for cases of uniform and non-uniform distributions, respectively. For this purpose studies on the limits of maximal spacings have been used. The largest spacings for uniformly distributed i.i.d. sequences in [0, 1] have been studied by several authors [165, 169]. Theorem 3.2 [165] presented in Section 3.2.3 has been used to prove the Lemma 3.3.

Lemma 3.3. *For the histogram scheme used by MIDI, taking a fixed c such that $0 < c < 1$, in the case of uniform distribution, $\mu\{x : diam(\pi_n[x]) > \epsilon\} \to 0$, with probability one for every $\epsilon > 0$.*

Proof. By Theorem 3.2

$$|M_n - \log(n)/n| = C(\log(\log(n))/n) \tag{3.42}$$

Multiplying both sides by n^c, where c is a constant such that $0 < c < 1$, for $n \to \infty$

$$|n^c.M_n - \log(n)/n^{1-c}| = C(\log(\log(n))/n^{1-c}) \tag{3.43}$$

Note that, $\log n/n^{1-c} \to 0$, as n goes to infinity. Also, $\log(\log(n))/n^{1-c} \to 0$, as n goes to infinity. So $n^c M_n \to 0$ as $n \to \infty$. It may be noted here that L_{max} is the same as $g.M_n$ where g is the range. So $n^c L_{max} \to 0$ as $n \to \infty$, in case of uniform density. □

Similarly, limit results for ordered spacings (where a uniform distribution has not been assumed) can be found in an article by [166]. Theorems 3.3 and 3.4 proposed by [166] have been used to show that the diameter of each cell shrinks to 0 in the given scheme as $n \to \infty$ for non-uniform distributions in Lemma 3.4. These two theorems proposed by [166] have been stated in Section 3.2.3. Using these theorems, we state the following lemma.

Lemma 3.4. *For the histogram scheme used by MIDI, taking a fixed c such that* $0 < c < 1$*,*

$$\mu\{x : diam(\pi_n[x]) > \epsilon\} \to 0, \textit{ with probability one for every } \epsilon > 0 \tag{3.44}$$

Proof. We note that, if the conditions stated in theorems 3.3 and 3.4 given by De-

heuvels [166] and stated in Section 3.2.3 are fulfilled,

$$\limsup_{n\to\infty} \frac{nM_k^{(n)} f(x_0) - \log(n)}{\log_2(n)} = \frac{2}{k} - \frac{1}{r} \tag{3.45}$$

$$\implies \limsup_{n\to\infty} n^c M_k^{(n)} = \frac{(\frac{2}{k} - \frac{1}{r})\log_2(n) + \log(n)}{n^{1-c} f(x_0)} \tag{3.46}$$

Note that, for $0 < c < 1$ and a fixed k

$$\lim_{n\to\infty} log(n)/n^{1-c} = 0 \tag{3.47}$$

$$\text{Also, } \lim_{n\to\infty} (2/k - 1/r) log_2(n)/n^{1-c} = 0 \tag{3.48}$$

Maximal spacing between the points is given by $L_{max} = M_1^{(n)}$. It follows from (3.46), (3.47) and (3.48) that $n^c L_{max} \to 0$ as n goes to ∞, under the given conditions. Suppose that we have calculated the length of the bin using the maximal spacing along the x axis, then $n^c L_{max}$ is taken as the length of each cell along the x axis. Along the other axis, say the y axis, the length of each division is taken as $g_y/\log_{10}(n)$ where g_y is the range along the y axis. For a distribution with compact support, g_y is finite, so $g_y/\log_{10}(n) \to 0$ as $n \to \infty$. As the length along each direction converges to 0 as n goes to infinity, we can say that the diameter will shrink to 0 as $n \to \infty$.

□

As a consequence of Theorem 3.1 given in Section 3.2.3, and the Lemmas 3.1, 3.2, and 3.4 given above, the following theorem can be stated.

Theorem 3.5. *The histogram scheme used by MIDI leads to L_1 consistent density estimates.*

Although we cannot say that the presented method is the best way to obtain the bin length, it satisfies the requirements of L_1 *consistency* [163]. It is a simple and

efficient method compared to existing methods.

3.3 Algorithm

The algorithm 3.1 provides the steps to calculate the estimate of the dependence index. Figure 3.2 presents an overview of MIDI.

Algorithm 3.1: Algorithm for MIDI

`/* Dataset` D' `has dimension` $n \times m$ `*/`
`/* Normalize the data to closed interval */`

1 Sort the data along the x-axis. Find the maximal spacing between consecutive points along the x-axis. This maximal spacing can be denoted by the value L_{max}.
2 Define the bin width along the x-axis as $B_x = n^c.L_{max}$, with n representing the total number of data points and c is a constant such that $0 < c < 1$.
3 Divide range y axis in $log_{10}(n)$ bins of equal length.
4 For each region formed by the intersection of bin i along the x-axis and j along the y-axis, determine the number of points denoted as $\#_{i,j}$.
5 Calculate the number of points for each region along the x-axis as denote it as $\#_i$.
6 Calculate the number of points for each region along the y-axis as denote it as $\#_j$.
7 Calculate the value $\hat{I} = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \frac{\#_{i,j}}{n} \log(\frac{n\#_{i,j}}{\#_i \#_j})$, where n_x and n_y are the number of divisions along each direction.
8 Calculate the value of $\hat{H}_x = \sum_{i=1}^{n_x} \frac{\#_i}{n} \log(\frac{n}{\#_i})$.
9 Calculate the value of $\hat{H}_y = \sum_{j=1}^{n_y} \frac{\#_j}{n} \log(\frac{n}{\#_j})$.
10 Calculate $MIDI_x = \frac{MI}{min(\hat{H}_x, \hat{H}_y)}$
11 Repeat the procedure after interchanging x and y axes and take the maximum of the two values $MIDI = max(MIDI_x, MIDI_y)$.

3.4 Experimental setup for analysis with MIDI

In this section, the methodology for conducting the experiments has been discussed. This includes experiments on synthetic datasets and details of a feature selection

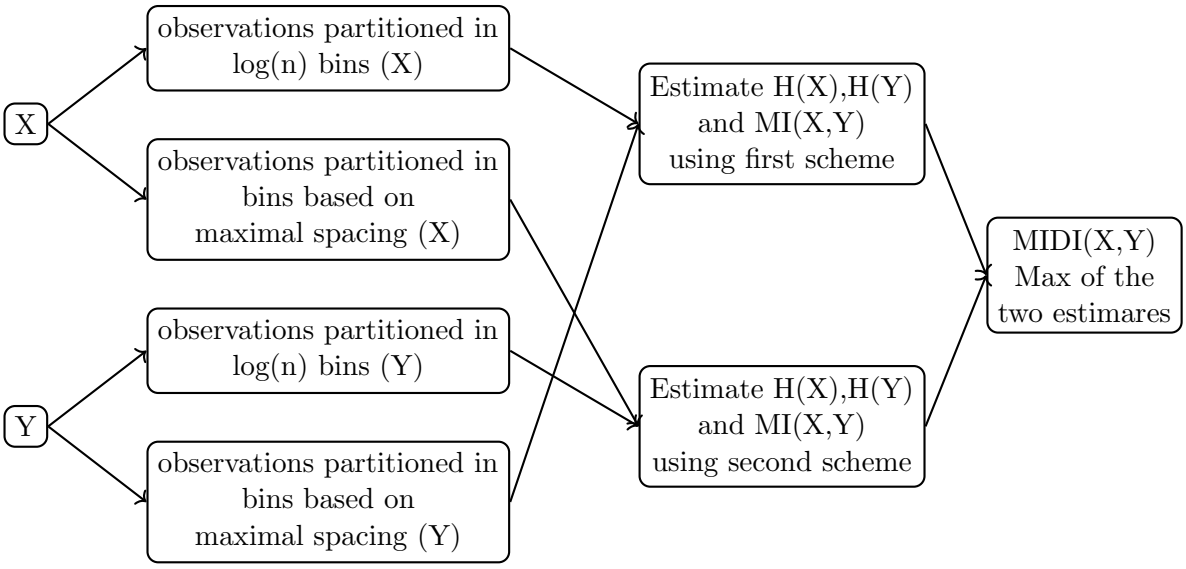


Figure 3.2: Broad outline showing how MIDI works

method designed using MIDI. The novel feature selection method has been applied to real-world datasets which are described in this section.

3.4.1 Setup for experiments on synthetic datasets with MIDI and other dependence indices

Several datasets have been used to present experimental results for MIDI. Recall that we calculated the bin length as $n^c L_{max}$, where L_{max} is the maximal spacing and c is a constant. During experiments with synthetic datasets, the value of the constant c is taken to be 0.1. Experiments with synthetic datasets have been performed with MIDI because the relationships between the features are known, facilitating a more straightforward evaluation of the results.

Three distinct categories of relationships can be identified between two features. These categories are:

1. Relationship between two features X and Y where X is a function of Y or vice

versa.

2. Relationship where one-to-one mapping does not exist but there is a functional form (e.g. Cartesian coordinates for circle) between them.

3. The relationship is not functional (e.g., two normal distributions with correlation coefficient ρ where $|\rho| < 1$).

We have conducted experiments involving the three aforementioned types of feature relations. For comparative analysis, we used both MIC [63] and distance correlation (Dcor) [51], as these methods are well-established within the literature and demonstrate superior performance across numerous data sets compared to traditional approaches such as the Spearman correlation coefficient and Pearson's correlation coefficient. Simon and Tibshirani also performed a comparative study of MIC and Dcor [158]. MIC, Dcor, and MIDI have been computed for the datasets described in Table 3.1, as well as for data sampled from a normal distribution exhibiting varying correlation coefficient values. The following paragraphs describe the method to generate datasets with and without noise. The methodology for evaluating the power (complement of type 2 error) of MIDI and other indices has also been presented.

3.4.1.1 Generation of noiseless data

In this section, we present the methodology used to generate the noiseless data. Here, three types of noiseless data are considered. Table 3.1 provides the functional forms corresponding to the first and second types of associations of the three different relationships stated above. Note that the same functional forms were used by Reshef et al. to illustrate MIC for noiseless data. The generation of data corresponding to the functional forms of Table 3.1 is carried out as follows. The points for the variable X are drawn randomly from the uniform distribution and the value of Y is calculated

Table 3.1: Different datasets used for experiments

Function	Description	Domain
line	$y = x$	$x \in [0, 1]$
half-parabola	$y = x^2$	$x \in [0, 1]$
parabola	$y = (x - 0.5)^2$	$x \in [0, 1]$
exponential	$y = 10^x$	$x \in [0, 1]$
sinusoidal	$y = \sin(10\pi x + x)$	$x \in [0, 1]$
sin (fourier frequency)	$y = \sin(16\pi x)$	$x \in [0, 1]$
sin (non fourier frequency)	$y = \sin(13\pi x)$	$x \in [0, 1]$
sin (varying frequency)	$y = \sin(7\pi x(1 + x))$	$x \in [0, 1]$
circle	$y = (2 * z - 1) * (\sqrt{1 - (2 * x - 1)^2})$, where z is randomly chosen from $\{0, 1\}$	$x \in [0, 1]$
normal uncorrelated	$\mu_1 = 0$, $\sigma_1 = 1$, $\mu_2 = 0$, $\sigma_2 = 1$, $\rho = 0$	$(x, y) \in (-\infty, \infty) \times (-\infty, \infty)$
uniform	random number generator	$(x, y) \in [0, 1] \times [0, 1]$

as a function of X. Different sample sizes are considered, and the values of the indices corresponding to these sizes are given in Table 3.3.

For comparison of performance in the case of a non-functional relationship, Table 3.4 provides the values of the three indices for different sample sizes in the case of a bivariate normal distribution with mean 0 and variance 1 for both X and Y, and different values of the correlation coefficient ($\rho = 1, 0.95, 0.9, 0.8, 0.7, 0.6, 0.5, 0.3, 0.01$). The results have been discussed in Section 3.5.1.1.

3.4.1.2 Generation of noisy data

A further aspect of the investigation is to examine the behavior of relationship indices when noise is added to the dataset. With the introduction of additional noise into a dataset, the relationship between the variables becomes attenuated, leading to a reduction in the value of any relationship index. Hence, datasets with different noise levels are considered for testing MIDI.

To calculate the values of the indices, 1000 points are drawn from a uniform

distribution. The value of Y is calculated using the function in Table 3.1, as in the case of noiseless data. We add Gaussian noise with mean $\mu = 0$ and variance σ^2. The different values of σ^2 used to generate noise are $10^{-6}, 10^{-4}, 10^{-2}, 10^{-1}, 1, 10^1, 10^2$. The noise is added to the variable Y, and the values of various indices are computed. These results are documented in Table 3.5.

To study the non-functional relationship, X and Y are sampled from a bivariate normal distribution as described in Section 3.4.1.1 for noiseless data. But here 1000 points are considered and different levels of noise are added to Y as described in the previous paragraph. The findings are documented in Table 3.6.

3.4.1.3 Methodology to evaluate the power of different methods

A critical aspect under examination is the power of feature relation indices, with the alternative hypothesis positing the absence of association. A high power indicates that the index in question is deemed adequate. Simon and Tibshirani [158] employed certain functions to evaluate power, as enumerated in Table 3.2. We have similarly used these functions and the methodology adopted by Simon and Tibshirani [158] to compare the power of MIC, Dcor, and MIDI. A concise description of the methodology is provided below.

For each function considered, the values of various indices are computed for X and Y, where X comprises 1000 data points sampled from a uniform distribution. The dataset for Y is produced by applying a specified function and adding noise drawn from a normal distribution with a mean of zero and different variance levels. Thirty different levels of standard deviation are considered for each function. The standard deviation for noise is given as σ = (noise scale $\times$ noise level), where the noise scale for each function is mentioned in Table 3.2 and the noise level takes values $1/10, 2/10 \cdots, 30/10$. For example, for the case of a cubic function, the noise scale is

Table 3.2: Data sets used to determine power of the indices at different noise levels. Values of standard deviation used for each function are given by the product of the corresponding noise scale with $1/10, 2/10, \cdots, 30/10$

Function	Description	Domain	Noise scale
line	$y = x$	[0,1]	1
quadratic	$y = 4 * (x - .5)^2$	[0,1]	1
cubic	$y = 128 * (x - 1/3)^3 - 48 * (x - 1/3)^2 - 12 * (x - 1/3)$	[0,1]	10
sin 1/8	$y = \sin(4 * pi * x)$	[0,1]	2
sin 1/2	$y = \sin(16 * pi * x)$	[0,1]	1
fourth root	$y = x^{1/4}$	[0,1]	1
circle	$y = (2 * z - 1) * (\sqrt{1 - (2 * x - 1)^2})$, where z is randomly chosen from $\{0, 1\}$	[0,1]	1/4
step	$y = 1$,if $(x > 0.5)$, $y = 0$, otherwise	[0,1]	5

10, so the values of σ are $10/10, 20/10, 30/10, \cdots, 300/10$. The standard deviations used to generate the noise are the same as those used by Simon and Tibshirani [158]. For every functional form and each specified value of the standard deviation, a total of 500 datasets are generated, each comprising 1000 data points. The values of various indices are computed for these datasets. In addition, index values are determined for 500 datasets in which the features are independent by sampling both X and Y from a uniform distribution. Thus, we obtain the values of the indices for the null scenario. The cutoff value of the index is calculated as 0.95 quantile of indices generated for the null scenario. Among the data sets where a particular functional relationship exists, the ratio of data sets for which the value of the index is above the cut-off for a given σ to a total number of data sets generated for the same σ is taken as the power at the particular noise level for the relationship under consideration. The results are presented in Figure 3.3.

3.4.1.4 Setup for experiments to determine execution time of MIDI on datasets of varying size

This section presents the methodology for evaluating the execution time required to calculate the relationship index between all pairs of features in datasets of different sizes. One of the advantages of using the maximal spacing is that the bin length for each feature is calculated independently. For calculating the bin sizes, the time complexity is $O(mn\log(n))$. The maximum number of bins is given by $n^{1-c}\log_{10}(n)$, as discussed in Section 3.2.4. The probabilities are estimated using these bins, so the overall time complexity is $mn\log(n) + m^2n^{1-c}\log(n)$. However, we have already shown in Section 3.2.4 that the number of bins grows at a sublinear rate as n increases. Thus, it has been seen that MIDI has a shorter execution time in comparison with Dcor and MIDI. The index MIC also uses a histogram to calculate mutual information but calculates the maximum of normalized mutual information over several possible partitions for each feature pair, requiring longer execution times. Dcor calculates the distance between observations for each pair of features. Table 3.7 reports the execution time required by MIDI, Dcor and MIC for datasets of different sizes. Experiments have been performed with data drawn from a uniform distribution to determine the execution time for MIDI, and other methods for datasets of varying size and the results are reported in Table 3.7.

3.4.2 Setup and methodology for a novel feature selection using MIDI

High-dimensional datasets often contain several redundant features. Thus, feature selection [170] is an important pre-processing step before applying any machine learning method. Feature extraction methods such as PCA and Kernel PCA [171], often result

in improved classification performance. However, the physical interpretation of the features obtained by these methods is difficult. The same applies to deep learning-based methods. In contrast, directly selecting a subset of features allows us to better understand the physical meaning of the data [172]. Additionally, obtaining the ground truth of labels for datasets is often a time-consuming and expensive process. Thus, unsupervised feature selection methods can be advantageous for the preliminary analysis of new datasets. This section presents a novel unsupervised feature selection method using MIDI.

3.4.2.1 Algorithm for MIDI-based feature selection

This method uses MIDI to find similarity between features, so that redundancy in the data can be reduced. Since MIDI is a dependence index that has value in the range $[0, 1]$, the distance between features can be calculated by subtracting the similarity from one. Then, a greedy search algorithm is used to reject one feature at each step and the other feature is kept for further consideration. At each step, the next distance (sorted in increasing order) is considered. Between the two features associated with this distance, if one is already selected (in an earlier iteration), then the other is rejected. If both are already rejected or selected, we move on to the next iteration. If both are considered for the first time, one is retained and the other is rejected. In the following paragraphs, we present the method in detail.

Due to the histogram scheme, MIDI works best when the density function has compact support (i.e., the function has non-zero values only on a closed interval). Thus, before applying feature selection, we normalized the data to $[0, 1] \times [0, 1]$. It should be mentioned that the data may come from a distribution with an unbounded base or a bounded base with an unknown range. Let the data be given as a matrix where the element in the i^{th} row and in the j^{th} column is denoted by $a_{i,j}$. In order to

normalize data originating from various distributions to the specified range $[0, 1]$, we have employed one of the two transformations detailed in Equations 3.49 and 3.50:

$$norm_1(a_{i,j}) = \frac{a_{i,j} - \min\limits_{1 \leq k \leq n}(a_{k,j})}{\max\limits_{1 \leq k \leq n}(a_{k,j}) - \min\limits_{1 \leq k \leq n}(a_{k,j})}; \tag{3.49}$$

$$norm_2(a_{i,j}) = norm_1(\tan^{-1}(a_{i,j})/\pi + 0.5). \tag{3.50}$$

The transformation $norm_1$ is a popular way to map data from a bounded base to the interval $[0, 1]$, while transformation $norm_2$ can be used to map data from the interval $(-\infty, \infty)$ to the interval $[0, 1]$. Thus, $norm_2$ maps points from an unbounded region to a bounded interval. Consequently, this transformation is suited for application in scenarios where data is drawn from a distribution with non-compact base, like the Gaussian distribution.

After the above-mentioned normalization, pairwise distances between features are calculated, and features are selected or rejected based on increasing order of the distances. The steps are described in detail below, and the algorithm for feature selection is written as Algorithm 3.2.

1. Normalize the data. Calculate the distance between each pair of variables X and Y as $Dist(X, Y) = 1 - MIDI(X, Y)$.

2. Sort the distance values (increasing). Initialize sets of selected and rejected features as null.

3. Iterate considering distances in increasing order in a while loop until the required number of features is selected.

4. The i^{th} distance from the sorted array is selected.

5. If both the features associated with the selected distance are already included or any of the features are already rejected, go to the next iteration.

6. If one feature is selected while the other is not yet tested, we reject the newly tested feature.

7. If both features are tested for the first time, we find the sum of distances between the feature under consideration and the features not yet selected. We select the feature for which the calculated sum is greater and reject the other .

8. Iterate in the while loop if the desired number of features has not yet been selected.

3.4.2.2 Datasets used for feature selection using MIDI

We have applied the MIDI-based feature selection method (Fs-MIDI) to two hyperspectral datasets named (1) "Indian Pines" having image size 145×145, 200 spectral bands and 16 classes [173] and (2)"Botswana" [174] having image size 1476×256, 145 spectral bands and 14 classes. The hyperspectral images are converted to 2 dimension matrices by treating each pixel as an observation and each spectral band is considered as a feature. The data for hyperspectral images is normalized using $norm_2$ (described in Equation 3.50), as it provides better results. The number of features to be selected varies from 2 to 32, with a step size of 2. The results are reported in Tables 3.8 and 3.9 and in Figure 3.4.

We have also applied the MIDI-based feature selection method (Fs-MIDI) to the UCI-ML datasets, and the results have been reported in Tables 3.10-3.14. These are: (1)"Default of credit card clients" dataset [175] of the banking sector that has 30000 observations with 23 features and two classes. (2) "Predict Student Dropout and

Algorithm 3.2: MIDI based Feature Selection

```
   /* Normalize the data D having dimensions N × M                    */
   /* Calculate the feature similarity using using MIDI               */
1  S = MIDI(D)
   /* Get distances by subtracting each similarity value from 1       */
2  Dist = {1}_{M×M} − S
3  Sort the distances to obtain Sorted distances and corresponding features
     defined by row_index, column_index
4  while number_Selected < to_Select do
5      if number_Rejected == (M − to_Select) then
6          break while loop
7      Consider next element of Sorted distances
8      if either row_index or column_index is rejected or both are selected then
9          continue while loop
10     if only one is selected then
11         reject other feature and continue while loop
12     if both are neither selected nor rejected then
13         RowDist = Sum(Dist(row_index, not yet selected features ))
           ColDist = Sum(Dist(column_index, not yet selected features)) if
           RowDist > ColDist then
14             Select row_index feature
15             Reject column_index feature
16         else
17             Select column_index feature
18             Reject row_index feature
```

Academic Success" dataset [176] from the education sector has 4424 observations with 36 features and three classes. (3) "Magic gamma telescope" dataset [177] from the application area named particle physics, has 19020 observations with 10 features and 2 classes.(4) "Breast cancer (original Wisconsin)" dataset [178] from the medical science sector has 683 observations, 9 features and 2 classes.(5) "Electrical Grid Stability Simulated Data" dataset [179] from the application area electrical engineering, has 10000 observations, 12 features and 2 classes. The number of selected features varies from 2 to $m-2$ where m is the number of features in the dataset, with step size 2.

The data is normalized using $norm_1$ (described in Equation 3.49). The results are reported in Tables 3.10-3.14 and Figure 3.5.

The results of Fs-MIDI (feature selection using MIDI) have been compared with the methods given by Dey et al. [161] and VCSDFS [86]. We have also performed the feature selection by replacing MIDI with Dcor in Algorithm 3.2. We have called this method Fs-Dcor and compared it with Fs-MIDI. However, feature selection using MIC has not been performed due to the high computation time required by MIDI as discussed in Sections 3.4.1.4 and 3.5.1.4. The parameters used by the first method are the same as those reported in the article [161]. For VCSDFS, we have used the implementation provided by the author with the default parameters. For Fs-MIDI, we have used the value of c as 0.01 to calculate MIDI, as it is seen to provide better results for feature selection experiments.

The classification of observations with selected features (as obtained by the MIDI based and three other methods) has been done using a Support Vector Machine (SVM). A train-test ratio was considered as 30:70. The average (over 10 simulations) of the overall accuracy (OA) and the Kappa coefficient (K) have been used to evaluate the results. The standard deviation for Overall Accuracy (over 10 simulations) is also reported. These results have been reported in Tables 3.8-3.14 and in Figures 3.4 and 3.5.

3.5 Results

This section presents the results of the experiments described in the previous section.

3.5.1 Results on Synthetic datasets

This section presents results for data where the variables are in perfect functional relation as well as results for synthetic noisy data. This includes results that report the power of MIDI for different functions.

3.5.1.1 Results on noiseless data

For the data generated using the methodology given in Section 3.4.1.1 the results are reported in Table 3.3. The table reports the value of the indices for a different number of points and the functions named in Table 3.1. It is seen that, as the sample size increases, the MIDI value approaches 1 for variables that are related to each other and goes to 0 when the variables are independent. It may be noted that the value of MIC is higher than the value obtained by MIDI for perfect relations. However, even in the case of uniform and Gaussian distributions, where the variables are generated independently and are expected to be completely uncorrelated, the MIC value is relatively high. This is undesirable because it increases the possibility of a type 2 error, as we will see from the results presented in Section 3.5.1.3. This happens because MIC chooses the grid size that maximizes the value of the measure. Further Dcor is not able to detect non-monotonous relations between variables effectively. On the other hand, MIDI can detect non-monotonous relationships.

The results for data with Gaussian distribution are presented in Table 3.4 and indicate that the MIDI value decreases as the correlation coefficient decreases. An increase in the number of data points results in the MIDI value remaining consistent when the correlation coefficient is high, whereas it declines substantially when the coefficient is low.

Table 3.3: Values of dependence measure estimates on synthetic noiseless data sets with varying sizes

Number of points	1000			2000			5000			10000		
Function	MIC	Dcor	MIDI	MIC	Dcor	MIDI	MIC	Dcor	MIDI	MIC	Dcor	MIDI
line	1.0000	1.0000	0.9969	1.0000	1.0000	0.9971	1.0000	1.0000	0.9978	1.0000	1.0000	0.9992
half-parabola	1.0000	0.9818	0.9969	1.0000	0.9823	0.9980	1.0000	0.9649	0.9980	1.0000	0.9666	0.9983
parabola	1.0000	0.5070	0.9792	1.0000	0.493	0.9856	1.0000	0.2453	0.9916	1.0000	0.2409	0.9941
exponential	1.0000	0.9768	0.9770	1.0000	0.9770	0.994	1.0000	0.9570	0.9940	1.0000	0.9558	0.9941
sinusoidal	1.0000	0.1289	0.8705	1.0000	0.1359	0.9235	1.0000	0.0207	0.9625	1.0000	0.0242	0.9783
sin (fourier frequency)	1.0000	0.1799	0.8299	1.0000	0.1277	0.8544	1.0000	0.0108	0.9456	1.0000	0.0126	0.9658
sin (non fourier frequency)	1.0000	0.1259	0.8700	1.0000	0.1056	0.9189	1.0000	0.0102	0.9588	1.0000	0.0113	0.9655
sin (varying frequency)	1.0000	0.1734	0.847	1.0000	0.1752	0.8966	1.0000	0.0105	0.9471	1.0000	0.0262	0.9625
circle	0.6770	0.1627	0.4732	0.7098	0.159	0.4903	0.7098	0.0241	0.5000	0.7377	0.024	0.5000
normal uncorrelated	0.1320	0.0859	0.007	0.1012	0.0361	0.0032	0.0810	0.0005	0.0030	0.0619	0.0002	0.0020
uniform	0.1134	0.0567	0.0538	0.1012	0.0481	0.0483	0.0862	0.0003	0.0407	0.0612	0.0004	0.0340

Table 3.4: Values of dependence measure for bivariate normal distribution with a given correlation coefficient for data sets of varying sizes

Number of points	1000			2000			5000			10000		
cor (ρ)	MIC	Dcor	MIDI	MIC	Dcor	MIDI	MIC	Dcor	MIDI	MIC	Dcor	MIDI
1	1.0000	1.0000	0.8007	1.0000	1.0000	0.8181	1.0000	1.0000	0.8176	1.0000	1.0000	0.8211
0.95	0.8012	0.8742	0.6701	0.7705	0.8761	0.6727	0.7528	0.8651	0.6775	0.7348	0.8678	0.6624
0.9	0.6249	0.7281	0.5400	0.6237	0.7678	0.5517	0.6338	0.7613	0.5593	0.6093	0.7587	0.5431
0.8	0.5280	0.6105	0.3721	0.4892	0.5813	0.3708	0.4756	0.5602	0.3778	0.4500	0.5625	0.3597
0.7	0.4210	0.4287	0.2545	0.3774	0.4171	0.2801	0.3529	0.4288	0.3008	0.3358	0.4252	0.2683
0.6	0.3218	0.3179	0.2165	0.2671	0.3218	0.2014	0.2081	0.3065	0.2005	0.2463	0.3062	0.1774
0.5	0.2707	0.2139	0.1630	0.2245	0.2009	0.1568	0.2162	0.2087	0.1192	0.1919	0.2057	0.1266
0.3	0.1744	0.0776	0.1377	0.1502	0.0556	0.0446	0.1181	0.0774	0.0475	0.1075	0.0745	0.0406
0.01	0.1400	0.0030	0.0084	0.1000	0.0039	0.0047	0.0700	0.0011	0.0017	0.0646	0.0004	0.0008

3.5.1.2 Noisy data

In this section, we report the results for the datasets generated using the methodology described in Section 3.4.1.2. Table 3.5 reports the values of the indices for datasets having 1000 observations with varying amount of Gaussian noise. The standard deviation of the noise is mentioned in the table. It can be seen that the MIDI values for the low-noise data become closer to 1 if a relationship exists and move to 0 when there is no relationship between the variables. The value of MIDI decreases smoothly with increasing noise levels.

Similarly for the results for the data with normal distribution reported in Table 3.6, it is seen that the value of MIDI also decreases with increasing noise level.

3.5.1.3 Results for experiments to determine the power of MIDI

From the experiments described in Section 3.4.1.3 we report the performance of MIDI in terms of power in Figure 3.3. It is seen that MIDI has a better capability to find feature relations in the presence of noise. For linear relationships, distance correlation works better than MIDI and MIC. Across the seven non-linear scenarios examined, MIDI is observed to possess greater statistical power than Dcor in the majority of instances, which aligns with anticipations. This is attributable to the tendency of Dcor to demonstrate a predisposition towards linear relationships. The distance correlation reduces to the correlation coefficient between two variables when the distributions are normal. Consequently, Dcor is expected to more effectively capture linear relationships compared to MIDI. In the case of non-linear relations, MIDI performs better than both Dcor and MIC.

Table 3.5: MIDI, MIC and Dcor results for different noise levels using 1000 data points

Function/Noise level (σ^2)	Method	10^{-6}	10^{-4}	10^{-2}	10^{-1}	1	10	10^2
line	MIDI	0.9983	0.9701	0.7825	0.3356	0.1496	0.1293	0.0901
	Dcor	1.0000	1.0000	0.9383	0.6592	0.2339	0.0654	0.0415
	MIC	1.0000	1.0000	0.8752	0.4173	0.1831	0.1382	0.1326
half-parabola	MIDI	0.9854	0.9588	0.7106	0.2912	0.1309	0.1154	0.0805
	Dcor	0.9821	0.9804	0.9236	0.6548	0.2292	0.1457	0.0502
	MIC	1.0000	1.0000	0.8001	0.4103	0.1624	0.1159	0.1413
parabola	MIDI	0.9737	0.89364	0.2345	0.1589	0.1212	0.1254	0.0789
	Dcor	0.4978	0.4899	0.2879	0.1301	0.0562	0.0432	0.0512
	MIC	1.0000	0.9763	0.3301	0.1836	0.1412	0.1283	0.1367
exponential	MIDI	0.9944	0.9856	0.9582	0.8927	0.6339	0.2691	0.1134
	Dcor	0.9773	0.9773	0.9771	0.9678	0.8832	0.5626	0.2309
	MIC	1.0000	1.0000	1.0000	0.9632	0.7139	0.3576	0.1548
sinusoidal	MIDI	0.8751	0.8745	0.8367	0.5756	0.2511	0.1165	0.0869
	Dcor	0.2595	0.2593	0.2556	0.2143	0.1584	0.0987	0.0412
	MIC	1.0000	1.0000	0.9756	0.7839	0.348	0.1564	0.1385
sin (fourier frequency)	MIDI	0.8267	0.8047	0.7893	0.5745	0.2678	0.1596	0.0783
	Dcor	0.1596	0.1538	0.1512	0.1287	0.0774	0.0732	0.0451
	MIC	1.0000	1.0000	0.9945	0.7688	0.2889	0.1648	0.1374
sin (non fourier frequency)	MIDI	0.8545	0.8543	0.8275	0.5478	0.2463	0.1574	0.0914
	Dcor	0.1282	0.1285	0.1233	0.12	0.0775	0.0657	0.0548
	MIC	1.0000	1.0000	0.9732	0.8782	0.2919	0.1587	0.1416
sin (varying frequency)	MIDI	0.8314	0.8198	0.7856	0.5571	0.2352	0.1267	0.0805
	Dcor	0.1367	0.1367	0.1367	0.1354	0.1051	0.0545	0.0512
	MIC	1.0000	1.0000	0.9784	0.7646	0.3023	0.1567	0.1389
circle	MIDI	0.4858	0.4817	0.4784	0.2778	0.1231	0.0734	0.0654
	Dcor	0.1678	0.1658	0.1502	0.1501	0.0560	0.0312	0.0562
	MIC	0.6567	0.6489	0.6445	0.4578	0.1293	0.1423	0.1326
normal uncorrelated	MIDI	0.0192	0.0121	0.0127	0.0249	0.0127	0.0127	0.0072
	Dcor	0.0515	0.0670	0.0571	0.0419	0.0573	0.0459	0.0592
	MIC	0.1353	0.1245	0.1332	0.1337	0.1355	0.1257	0.1389
uniform	MIDI	0.0551	0.0624	0.0556	0.0537	0.0599	0.0637	0.0538
	Dcor	0.0593	0.1026	0.0593	0.0524	0.0560	0.0468	0.0562
	MIC	0.1314	0.1407	0.1242	0.1319	0.1293	0.1309	0.1442

Table 3.6: MIDI, MIC and Dcor results for different noise levels using 1000 data points for bivariate normal distribution

Correlation coeff/Noise level (σ^2)	Method	10^{-6}	10^{-4}	10^{-2}	10^{-1}	1	10	10^2
1	MIDI	0.8121	0.8101	0.7989	0.6445	0.2824	0.0464	0.0157
	Dcor	1.0000	0.9963	0.9941	0.9317	0.6678	0.2737	0.0412
	MIC	1.0000	1.0000	0.9765	0.8145	0.4132	0.1855	0.1178
0.95	MIDI	0.6865	0.6558	0.6501	0.5282	0.2714	0.0155	0.0123
	Dcor	0.9114	0.9215	0.9109	0.86576	0.6312	0.2201	0.0794
	MIC	0.7617	0.7608	0.7212	0.6505	0.3816	0.1516	0.1367
0.9	MIDI	0.5212	0.5354	0.5263	0.4674	0.2412	0.0237	0.0071
	Dcor	0.8664	0.8657	0.8419	0.8013	0.5943	0.2516	0.0659
	MIC	0.6814	0.6505	0.6126	0.5637	0.3509	0.1796	0.1449
0.8	MIDI	0.3764	0.3679	0.3243	0.3278	0.2091	0.0236	0.0102
	Dcor	0.7578	0.7417	0.7532	0.7267	0.5323	0.1607	0.0516
	MIC	0.5001	0.5101	0.5297	0.4712	0.3219	0.1311	0.1453
0.7	MIDI	0.2601	0.2600	0.2597	0.2254	0.1096	0.0305	0.0077
	Dcor	0.6225	0.6431	0.6439	0.6295	0.4842	0.2010	0.0496
	MIC	0.3627	0.4679	0.4234	0.3684	0.2882	0.1498	0.1205
0.6	MIDI	0.1912	0.1911	0.1853	0.1149	0.0815	0.0256	0.0050
	Dcor	0.5762	0.5384	0.5559	0.5214	0.4184	0.1677	0.0742
	MIC	0.3549	0.3192	0.3189	0.2855	0.2365	0.1401	0.1384
0.5	MIDI	0.1198	0.1203	0.1219	0.1163	0.0738	0.0237	0.0094
	Dcor	0.4677	0.4674	0.4701	0.4183	0.2915	0.1584	0.0635
	MIC	0.2643	0.2648	0.2705	0.2375	0.1839	0.1596	0.1263
0.3	MIDI	0.0578	0.0594	0.0527	0.0472	0.0268	0.0100	0.0078
	Dcor	0.2521	0.2520	0.2519	0.2147	0.1987	0.0925	0.0456
	MIC	0.1745	0.1593	0.1724	0.1674	0.1465	0.1315	0.1300
0.01	MIDI	0.0035	0.0031	0.0029	0.0032	0.0030	0.0027	0.0031
	Dcor	0.0503	0.0514	0.0495	0.0496	0.0435	0.0503	0.0459
	MIC	0.1312	0.1285	0.1314	0.1358	0.1349	0.1237	0.1372

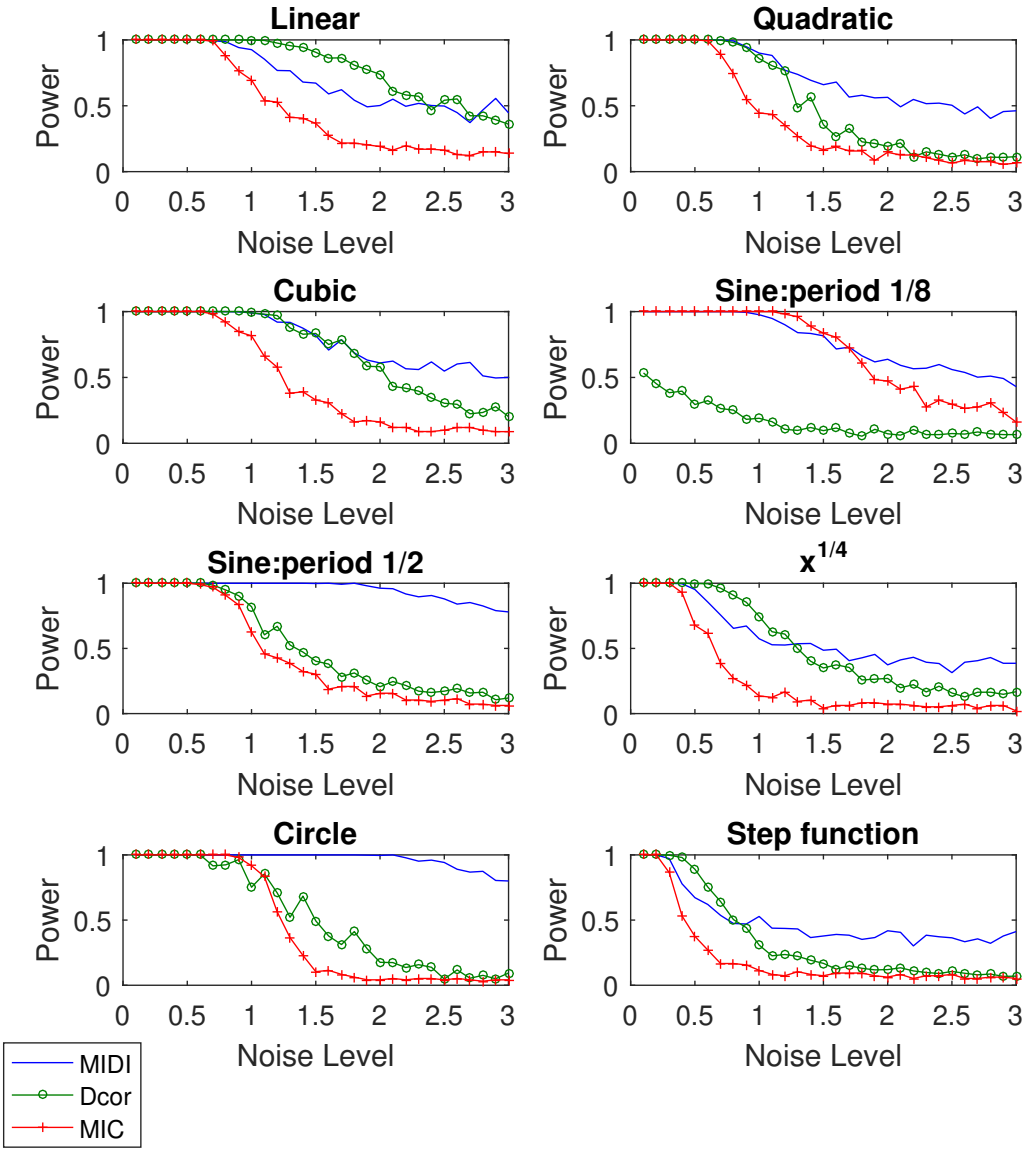


Figure 3.3: Power of different indices of dependence

3.5.1.4 Execution time of MIDI on datasets of varying size

The time complexity of MIDI is discussed in Section 3.4.1.4, along with the methodology to perform experiments to determine the execution time of MIDI for datasets of different sizes. It is seen that the execution time for MIDI is much shorter than the time required by Dcor and MIC. MIC has the highest execution time. The results are reported in Table 3.7. The table reports the number of features and the number of observations for different datasets and the execution time in seconds.

Table 3.7: Execution time for calculating dependence

Method	Number of Features	Number of Observations	Execution time
MIDI	10	1000	0.030
		5000	0.069
		10000	0.088
	20	1000	0.049
		5000	0.171
		10000	0.267
Dcor	10	1000	0.363
		5000	1.591
		10000	2.976
	20	1000	1.283
		5000	6.812
		10000	13.273
MIC	10	1000	13.156
		5000	173.513
		10000	546.012
	20	1000	57.335
		5000	756.654
		10000	2364.013

3.5.2 Results for unsupervised Feature Selection method using MIDI

The performance of the MIDI-based feature selection method (Fs-MIDI) and other feature selection methods is presented in Tables 3.8-3.14. The accuracy is also shown graphically in Figures 3.4 and 3.5. In the graphs, the best accuracy for each dataset

Table 3.8: Overall accuracy and Kappa Coefficient obtained using the features selected by different algorithms for the Indian Pines dataset

Number of features	Dey et. al. [161]			VCDFS [86]			Fs-Dcor			Fs-MIDI		
	OA	K	SD	OA	K	SD	OA	K	SD	OA	K	SD
2	0.483	0.41	0.008	0.408	0.31	0.004	0.400	0.31	0.006	0.535	0.46	0.004
4	0.591	0.53	0.002	0.638	0.58	0.006	0.524	0.45	0.003	0.690	0.64	0.008
6	0.679	0.63	0.006	0.703	0.66	0.004	0.551	0.48	0.006	0.725	0.68	0.006
8	0.787	0.76	0.006	0.750	0.71	0.005	0.580	0.52	0.007	0.742	0.70	0.006
10	0.816	0.79	0.006	0.756	0.72	0.004	0.610	0.55	0.004	0.749	0.71	0.006
12	0.816	0.79	0.005	0.782	0.75	0.004	0.669	0.62	0.005	0.760	0.72	0.006
14	0.829	0.80	0.003	0.778	0.75	0.003	0.714	0.67	0.005	0.769	0.73	0.005
16	0.829	0.80	0.005	0.780	0.75	0.004	0.773	0.74	0.004	0.774	0.74	0.007
18	0.831	0.81	0.004	0.782	0.75	0.003	0.821	0.80	0.005	0.785	0.75	0.003
20	0.832	0.81	0.006	0.769	0.74	0.003	0.826	0.80	0.003	0.794	0.76	0.005
22	0.835	0.81	0.004	0.804	0.78	0.005	0.833	0.81	0.004	0.838	0.81	0.004
24	0.832	0.81	0.006	0.811	0.78	0.004	0.852	0.83	0.003	0.852	0.83	0.007
26	0.839	0.82	0.004	0.791	0.76	0.004	0.856	0.84	0.003	0.858	0.84	0.006
28	0.837	0.81	0.006	0.792	0.76	0.005	0.861	0.84	0.005	0.857	0.84	0.003
30	0.806	0.78	0.005	0.802	0.77	0.004	0.864	0.84	0.005	0.860	0.84	0.003
32	0.803	0.77	0.006	0.819	0.79	0.003	0.865	0.85	0.006	0.866	0.85	0.003

is marked with a black circle.

The results show that the MIDI based method (Fs-MIDI) performs better than the compared methods for both hyperspectral image datasets. For the Indian Pines dataset [173], the classification accuracy increased with an increase in the number of features for most methods (all except [161]). For this dataset, the MIDI-based method provides good accuracy when the number of features selected is greater than 20. For the Botswana dataset [174], the MIDI-based method achieves the highest accuracy compared to other methods.

For the credit card dataset [175], Fs-MIDI achieves the highest accuracy. For the Student Dropout dataset [176], Fs-MIDI and Fs-Dcor achieve the highest accuracy. But when the number of features selected is small, other methods achieve better performance for this dataset. For the Magic Gamma dataset [177], the method by Dey et al. achieves the best accuracy. MIDI based feature selection (Fs-MIDI) has the second highest accuracy. For the Breast Cancer dataset [178], the MIDI based method (Fs-

Table 3.9: Overall accuracy and Kappa Coefficient obtained using the features selected by different algorithms for Botswana dataset

Number of features	Dey et. al. [161]			VCDFS [86]			Fs-Dcor			Fs-MIDI		
	OA	K	SD	OA	K	SD	OA	K	SD	OA	K	SD
2	0.451	0.41	0.009	0.530	0.49	0.008	0.671	0.64	0.007	0.697	0.67	0.009
4	0.763	0.74	0.008	0.693	0.67	0.009	0.796	0.78	0.005	0.869	0.86	0.007
6	0.797	0.78	0.008	0.864	0.85	0.006	0.828	0.81	0.007	0.893	0.88	0.005
8	0.807	0.79	0.004	0.884	0.87	0.005	0.834	0.82	0.007	0.900	0.89	0.005
10	0.825	0.81	0.007	0.891	0.88	0.004	0.841	0.83	0.006	0.899	0.89	0.007
12	0.829	0.81	0.009	0.899	0.89	0.008	0.849	0.84	0.006	0.895	0.89	0.003
14	0.841	0.83	0.008	0.902	0.89	0.008	0.855	0.84	0.004	0.906	0.90	0.007
16	0.853	0.84	0.004	0.905	0.90	0.006	0.860	0.85	0.007	0.912	0.90	0.009
18	0.853	0.84	0.008	0.909	0.90	0.004	0.858	0.85	0.009	0.914	0.91	0.006
20	0.855	0.84	0.005	0.905	0.90	0.008	0.877	0.87	0.005	0.915	0.91	0.005
22	0.856	0.84	0.006	0.909	0.90	0.004	0.881	0.87	0.006	0.917	0.91	0.005
24	0.884	0.87	0.004	0.911	0.90	0.004	0.898	0.89	0.004	0.921	0.91	0.005
26	0.881	0.87	0.007	0.920	0.91	0.005	0.903	0.89	0.005	0.919	0.91	0.006
28	0.890	0.88	0.008	0.918	0.91	0.003	0.901	0.89	0.006	0.919	0.91	0.003
30	0.892	0.88	0.006	0.917	0.91	0.005	0.904	0.90	0.004	0.923	0.92	0.007
32	0.909	0.90	0.008	0.919	0.91	0.006	0.906	0.90	0.006	0.917	0.91	0.006

MIDI) has the best accuracy. The Dcor-based method (Fs-Dcor) also achieves similar accuracy. Fs-MIDI provides the best performance for the Electric Grid dataset. In summary, the MIDI-based method (Fs-MIDI) has second-best performance for one of the datasets. It achieves best accuracy for four of five UCI-ML datasets and for the two hyperspectral image datasets named Indian Pines and Botswana. Also, from Table 3.15 we find that VCSDFS requires the least execution time among all methods. MIDI based feature selection requires more time than VCSDFS but its execution time is much shorter than FS-Dcor, and the method by Dey et al.

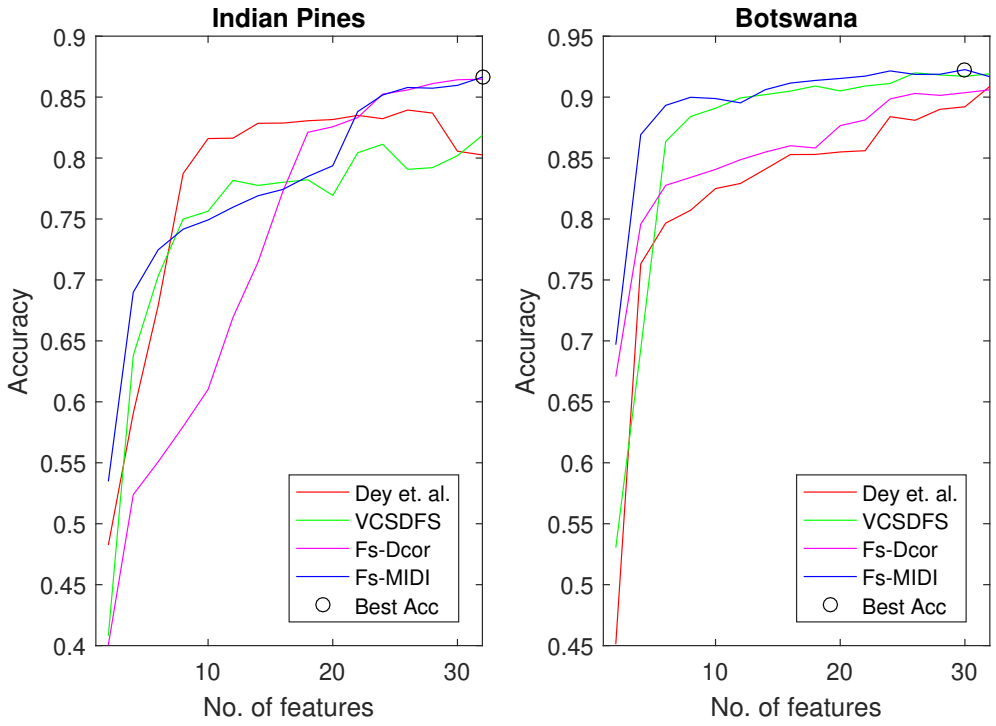


Figure 3.4: Overall accuracy for different algorithms for Indian Pines and Botswana datasets

Table 3.10: Overall accuracy and Kappa Coefficient obtained using the features selected by different algorithms for Credit Card dataset

Number of features	Dey et. al. [161]			VCDFS [86]			Fs-Dcor			Fs-MIDI		
	OA	K	SD	OA	K	SD	OA	K	SD	OA	K	SD
2	0.753	0.00	0.003	0.740	0.02	0.004	0.799	0.26	0.001	0.736	0.04	0.002
4	0.753	0.02	0.003	0.744	0.03	0.002	0.798	0.27	0.001	0.782	0.21	0.008
6	0.753	0.03	0.002	0.768	0.14	0.004	0.786	0.23	0.005	0.801	0.29	0.004
8	0.767	0.12	0.003	0.767	0.14	0.004	0.786	0.24	0.003	0.798	0.29	0.003
10	0.767	0.12	0.002	0.765	0.17	0.002	0.784	0.23	0.002	0.795	0.27	0.002
12	0.768	0.12	0.002	0.745	0.06	0.004	0.781	0.23	0.002	0.789	0.27	0.004
14	0.798	0.27	0.004	0.788	0.26	0.004	0.779	0.23	0.003	0.788	0.29	0.003
16	0.789	0.29	0.003	0.788	0.27	0.003	0.779	0.23	0.003	0.790	0.29	0.004
18	0.797	0.30	0.002	0.791	0.29	0.004	0.777	0.22	0.003	0.787	0.28	0.004
20	0.797	0.29	0.005	0.791	0.29	0.003	0.791	0.28	0.004	0.790	0.29	0.003
22	0.798	0.27	0.004	0.790	0.29	0.003	0.789	0.28	0.004	0.790	0.29	0.004

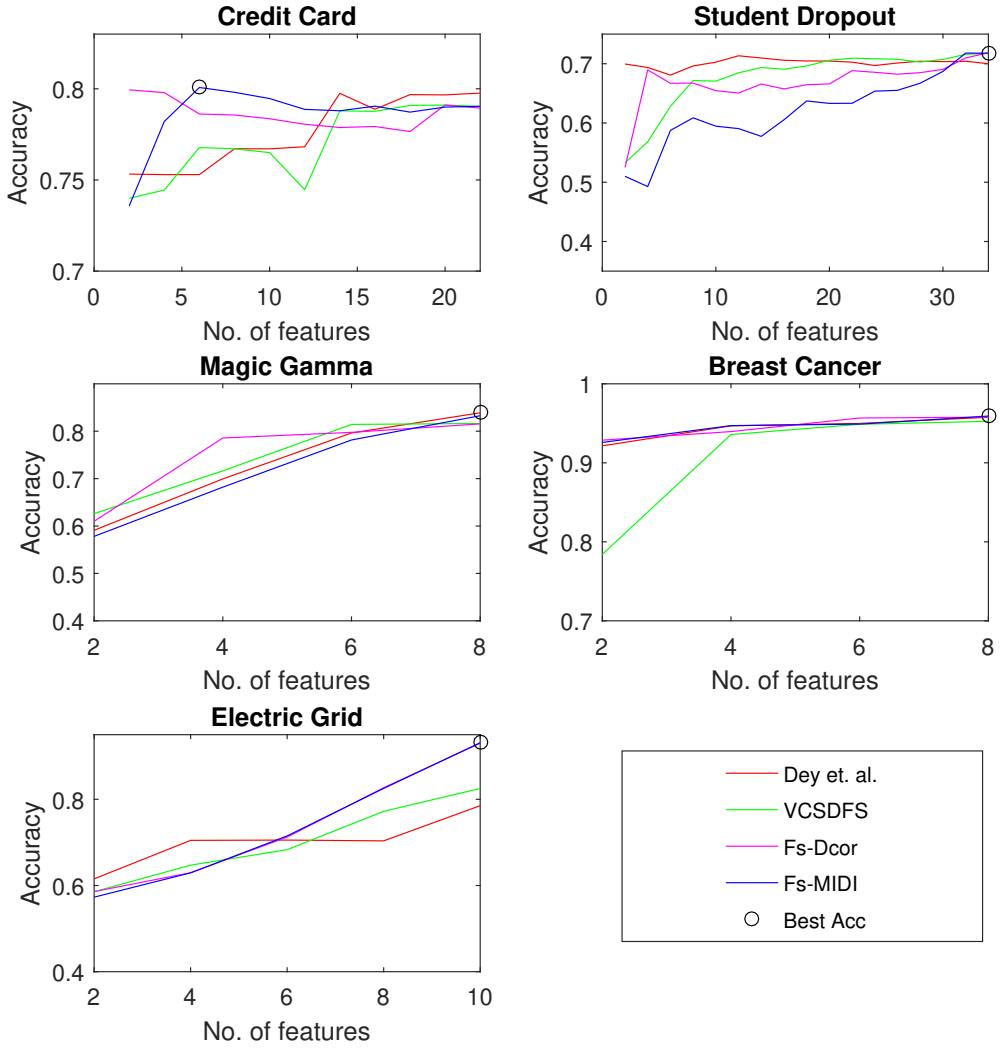


Figure 3.5: Overall accuracy for different algorithms for UCI-ML datasets

Table 3.11: Overall accuracy and Kappa Coefficient obtained using the features selected by different algorithms for Student Dropout dataset

Number	Dey et. al. [161]			VCDFS [86]			Fs-Dcor			Fs-MIDI		
of features	OA	K	SD	OA	K	SD	OA	K	SD	OA	K	SD
2	0.700	0.49	0.005	0.533	0.18	0.009	0.525	0.14	0.006	0.510	0.06	0.004
4	0.694	0.48	0.005	0.568	0.27	0.011	0.690	0.46	0.009	0.493	0.12	0.009
6	0.681	0.47	0.004	0.628	0.37	0.006	0.667	0.43	0.015	0.587	0.30	0.012
8	0.696	0.49	0.007	0.672	0.45	0.007	0.667	0.44	0.005	0.609	0.33	0.008
10	0.703	0.50	0.006	0.671	0.45	0.007	0.655	0.42	0.008	0.595	0.31	0.010
12	0.713	0.52	0.006	0.685	0.47	0.007	0.651	0.42	0.009	0.591	0.30	0.005
14	0.710	0.51	0.005	0.694	0.49	0.005	0.666	0.44	0.006	0.577	0.28	0.006
16	0.706	0.51	0.005	0.691	0.49	0.007	0.658	0.42	0.009	0.605	0.33	0.009
18	0.705	0.50	0.006	0.696	0.49	0.006	0.665	0.44	0.008	0.637	0.38	0.006
20	0.705	0.51	0.006	0.706	0.51	0.006	0.666	0.44	0.009	0.633	0.38	0.009
22	0.703	0.50	0.007	0.709	0.52	0.007	0.688	0.48	0.006	0.633	0.38	0.005
24	0.697	0.49	0.009	0.708	0.51	0.007	0.686	0.47	0.006	0.654	0.42	0.005
26	0.701	0.50	0.005	0.707	0.51	0.005	0.682	0.47	0.008	0.655	0.42	0.008
28	0.704	0.50	0.007	0.703	0.51	0.004	0.685	0.47	0.008	0.667	0.44	0.008
30	0.704	0.50	0.004	0.707	0.51	0.008	0.691	0.48	0.003	0.687	0.48	0.006
32	0.704	0.50	0.010	0.716	0.52	0.007	0.710	0.52	0.006	0.718	0.53	0.005
34	0.700	0.50	0.007	0.717	0.53	0.009	0.718	0.53	0.007	0.718	0.53	0.006

Table 3.12: Overall accuracy and Kappa Coefficient obtained using the features selected by different algorithms for Magic Gamma dataset

Number	Dey et. al. [161]			VCDFS [86]			Fs-Dcor			Fs-MIDI		
of features	OA	K	SD	OA	K	SD	OA	K	SD	OA	K	SD
2	0.591	0.01	0.007	0.626	0.10	0.038	0.610	0.08	0.008	0.578	0.02	0.008
4	0.699	0.33	0.007	0.716	0.37	0.008	0.786	0.52	0.004	0.68	0.30	0.006
6	0.796	0.54	0.011	0.814	0.58	0.007	0.798	0.55	0.005	0.782	0.51	0.004
8	0.839	0.64	0.006	0.816	0.59	0.003	0.815	0.59	0.004	0.833	0.63	0.005

Table 3.13: Overall accuracy and Kappa Coefficient obtained using the features selected by different algorithms for Breast Cancer dataset

Number	Dey et. al. [161]			VCDFS [86]			Fs-Dcor			Fs-MIDI		
of features	OA	K	SD	OA	K	SD	OA	K	SD	OA	K	SD
2	0.922	0.83	0.012	0.784	0.46	0.009	0.929	0.85	0.007	0.926	0.84	0.010
4	0.947	0.88	0.013	0.936	0.86	0.012	0.940	0.87	0.006	0.947	0.88	0.009
6	0.950	0.89	0.010	0.949	0.89	0.006	0.957	0.90	0.005	0.949	0.89	0.008
8	0.958	0.91	0.004	0.953	0.90	0.010	0.958	0.91	0.009	0.959	0.91	0.004

Table 3.14: Overall accuracy and Kappa Coefficient obtained using the features selected by different algorithms for Electric Grid dataset

Number of features	Dey et. al. [161]			VCDFS [86]			Fs-Dcor			Fs-MIDI		
	OA	K	SD	OA	K	SD	OA	K	SD	OA	K	SD
2	0.615	0.15	0.004	0.586	0.07	0.009	0.586	0.07	0.007	0.573	0.05	0.005
4	0.705	0.36	0.005	0.647	0.22	0.004	0.630	0.19	0.005	0.629	0.19	0.005
6	0.705	0.36	0.006	0.683	0.30	0.005	0.712	0.37	0.003	0.715	0.38	0.006
8	0.704	0.35	0.005	0.772	0.50	0.004	0.827	0.62	0.004	0.825	0.62	0.005
10	0.785	0.53	0.003	0.825	0.62	0.004	0.930	0.85	0.004	0.931	0.85	0.003

Table 3.15: Execution time for feature selection required by different methods

Methods / Datasets	Dey et. al. [161]	VCDFS [86]	Fs-Dcor	Fs-MIDI
Credit Card	876.230	0.094	55.300	0.845
Student Dropout	34.430	0.073	29.650	0.349
Magic Gamma	345.730	0.059	11.000	0.149
Breast Cancer	0.114	0.040	0.341	0.032
Electric Grid	156.730	0.090	7.507	0.486
Indian Pines	40275.276	1.179	2762.413	20.267
Botswana	1794.457	0.090	785.340	3.647

3.6 Limitations

It is seen that MIDI can be used to detect non-linear relations for continuous variables. It does this by estimating mutual information using a maximal spacing based histogram scheme. However, as seen in Section 3.2.4, the values estimated using the histogram scheme will converge only if the base for the distribution is path-connected. Thus, the method works well only for continuous distributions. Alternative histogram schemes may be developed for datasets with variables that do not fulfill this requirement. Additionally, we do not claim that the presented histogram scheme is the best one even for continuous distributions. Thus, there might be different ways to estimate the values. However, we find that the proposed scheme leads to shorter execution time and provide good performance in terms of the power of the index.

3.7 Summary

This chapter introduces a computational method to estimate non-linear dependence between two variables. A function of maximal spacing has been used to determine the length of the bin to calculate the density estimates. The density estimates obtained by this method are shown to be strongly consistent. These density estimates are used to calculate a novel dependence index named the Mutual Information-Based Dependence Index (MIDI). Through several experiments, we find that the value of MIDI goes to the true value of dependence as the number of points increases. We also find that the value of MIDI goes down smoothly if noise is added gradually to variables having perfect dependence.

The performance of MIDI has been compared to generally accepted measures like distance correlation (Dcor) and maximal Information Coefficient (MIC) in terms

of power and computational complexity with the help of several artificial data sets with different amounts of noise. MIDI overcomes the problem of overestimation that occurs with MIC. MIDI is also capable of detecting non-linear relationships where either of the variables is a function of the other, unlike Dcor, which requires the relation to be one-to-one and onto. Thus, for non-linear datasets, it achieves better performance than MIC and Dcor in terms of power. The execution time for MIDI is less in comparison with both MIC and Dcor.

MIDI has also been used to design a novel feature selection algorithm named Fs-MIDI. The feature selection method has been applied to UCI-ML datasets. It has also been applied to hyperspectral images, as they contain features representing narrow contiguous bands, and the similarity between adjacent bands is high. Experiments show that the MIDI based feature selection method named Fs-MIDI resulted in better classification performance in terms of overall accuracy for most of the datasets.

In general, we may say that MIDI can detect many types of relationships between variables without making any assumption about the functional form of the relationship. The power statistics of MIDI illustrate their effectiveness in detecting non-linear relationships.

Chapter 4

CBSC: A Novel Method for Connectedness-Based Subspace Clustering

4.1 Introduction

In this chapter, an algorithm for density-based biclustering is introduced. Biclustering is conceptualized as the identification of subsets of features and observations wherein these observations exhibit similar behaviors concerning the selected features. The interpretation of similarity in the behavior of the observations is dependent on the specific problem context. The objective in this chapter is to identify subspace clusters in which the feature vectors are interrelated with reference to the observations constituting the subspace cluster. This variety of subspace clustering is alternatively termed pattern-based subspace clustering or biclustering [95]. The terms biclustering and subspace clustering are used interchangeably in the context of this problem [27, 94]. In this chapter, we have used both terms interchangeably.

In contrast to traditional density-based subspace clustering algorithms that identify clusters through spatial proximity, the approach we have adopted relies on common high-density regions to group features on the basis of feature relations. The method presented here, CBSC, is capable of identifying subspace clusters based on both linear and non-linear interrelationships among features. A notable advantage of CBSC compared to previous density-based subspace clustering methodologies is that it eliminates the need for users to specify parameter values for density estimation. Instead, these values are deduced for each pair of features on the basis of the data distribution within the space pertinent to the specific pair of features. Consequently, CBSC is adept at identifying subspace clusters even when the relationships between different features exist on different scales. The following section describes the motivation behind CBSC and its objective.

4.2 Motivation and problem definition

4.2.1 Motivation

In recent years, techniques for the identification of non-linear relations between features or observations have been developed to a greater extent. Some examples are distance correlation [51], MIC [63], and MIDI [28]. The previous chapter discusses MIDI in detail. It seems natural and useful to extend methods of non-linear exploration to subspace clustering problems. In this chapter, we present a biclustering technique that can be used to explore non-linear relations in data. Many of the previous biclustering methods use spatial proximity to find biclusters [27,95,96,102,180]. On the other hand, most of the previously existing feature relation based methods are restricted to biclusters based on linear or monotonic relations [145]. In this chapter,

we present a feature relation based biclustering method that can find biclusters based on non-linear and even non-monotonous relations.

This chapter addresses subspace clustering by identifying non-linear relationships between features by finding observations forming common high-density regions between these features. Thus, finding non-linear correlations pertaining to a subset of observations (rather than the set of all observations) results in the formation of subspace clusters. In contrast to other density-based algorithms, CBSC does not require the specification of a radius or number of points to define neighborhoods and detect high-density regions. These parameters dynamically adjust with each pair of dimensions, thereby enhancing the adaptability of the procedure. These values are derived from the data using a function of the minimal spanning tree.

4.2.2 Problem definition

The objective of this study is to identify biclusters within a dataset, in which the concept of a bicluster is defined as follows. For a given data matrix $\mathbf{X} = [x_{ij}]$ with N observations indexed by $i = 1, \cdots, N$ and M features indexed by $j = 1, \cdots, M$, one needs to select a subset of n observations represented by the set of indices $I_{in} = \{i_1, i_2, \cdots, i_n\}$ and a subset of m features represented by the set of indices $J_{in} = \{j_1, j_2 \cdots, j_m\}$ such that the column vectors in the matrix $\mathbf{A}$ are related to each other directly or indirectly, where $\mathbf{A} = \mathbf{X}_{I_{in}, J_{in}}$. This notation indicates that the element in the p^{th} row and the q^{th} column of the matrix $\mathbf{A}$ is given as $a_{p,q} = x_{i_p, j_q}$.

$$\mathbf{A} = \begin{pmatrix} x_{i_1,j_1} & x_{i_1,j_2} & \cdots & x_{i_1,j_m} \\ x_{i_2,j_1} & x_{i_2,j_2} & \cdots & x_{i_2,j_m} \\ \vdots & \vdots & & \vdots \\ x_{i_n,j_1} & x_{i_n,j_2} & \cdots & x_{i_n,j_m} \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix}.$$

Numerous scholarly articles exist within the literature that explore methods for determining spatial proximity between two data points. The objective of CBSC is not to assess the proximity between data points, although such measurements were used in intermediate steps for finding biclusters. Instead, we find biclusters based on feature relations. We now define the relationship between the features. Let the k^{th} column vector of the matrix $\mathbf{A}$ be represented by

$$\mathbf{A}_{*,k} = \begin{pmatrix} x_{i_1,j_k} \\ x_{i_2,j_k} \\ \vdots \\ x_{i_n,j_k} \end{pmatrix} = \begin{pmatrix} a_{1,k} \\ a_{2,k} \\ \vdots \\ a_{n,k} \end{pmatrix}.$$

Vectors $\mathbf{A}_{*,u}$ and $\mathbf{A}_{*,v}$ are said to be directly related if $p(a_{t,u}, a_{t,v})$ is high for all $t = 1, 2, \cdots, n$, where $p(a_{t,u}, a_{t,v})$ is the joint density function for the u^{th} and v^{th} features of $\mathbf{A}$. The direct relation between vector A_u and A_v is denoted by $R(\mathbf{A}_{*,u}, \mathbf{A}_{*,v}) = True$.

Vectors $\mathbf{A}_{*,u}$ and $\mathbf{A}_{*,v}$ are said to be related indirectly if there exists a sequence $w_1, w_2, \cdots, w_z$ such that condition $R(\mathbf{A}_{*,u}, \mathbf{A}_{*,w_1}) = True \wedge R(\mathbf{A}_{*,w_1}, \mathbf{A}_{*,w_2}) = True \wedge \cdots, \wedge R(\mathbf{A}_{*,w_z}, \mathbf{A}_{*,v}) = True$ is satisfied.

In terms of graphs, two features j_u, j_v are said to be connected by an edge with reference to observations $i_1, i_2, \cdots, i_n$ if Cartesian points $(x_{i_t,j_u}, x_{i_t,j_v})$ for $1 \leq t \leq n$ form a high-density connected region in the two-dimensional space populated by

(x_{s,j_u}, x_{s,j_v}) for $1 \leq s \leq N$. It may be noted that the main objective of this method is not to find sets of observations which lie close together in a smaller subspace but to find features which are related to each other for a subset of observations. In some way, we have analyzed whether the value of one feature gives us information about the value of other features for a given set of observations.

Note that we have departed from the more common approach for biclustering, where one searches for heavy subgraphs. In this work, we are searching for non-linear relations between features for a subset of observations. Non-linear relations may often be unidirectional. Methods that search for heavy subgraphs may miss out on many features that are related to other features in the bicluster. Therefore, we search for longest paths in the graphs instead of heavy subgraphs. However, this bicluster is easily impacted by noise. To overcome the effect of noise, we use the following technique after identifying connected dense regions in two-dimensional space.

An edge is said to connect two features j_u and j_v if there exists a feature $j_s \in \{j_1, j_2, \cdots, j_n\} - \{j_u, j_v\}$ such that significantly large overlapping dense regions are found for feature pairs $< j_u, j_v >$ and $< j_u, j_s >$ and $< j_s, j_v >$. This gives us a robust solution that is close to our objective. The following section discusses the ideas behind the procedure in greater detail.

4.3 Intuition and methodology for finding densely connected regions

As discussed in Section 4.2.2, we need to find dense regions in a 2-dimensional Euclidean space corresponding to each pair of features. If one or more dense regions exist in the 2-dimensional space corresponding to two features, they are said to be

related with reference to observations forming the dense regions; otherwise, they are unrelated. Later, the pairs of features will be combined as described in Section 4.4. The approach for identifying dense regions has been described in this section. Connectivity-based clustering has been performed using methods such as the (K, r)-cluster approach [181] and DBSCAN [22], where connectivity is derived from overlap between open spheres of specified radius with some additional constraints applied. To establish connectivity in the two-dimensional space for each pair of dimensions, CBSC also uses overlap between open spheres but applies a more stringent set of conditions. The following paragraphs describe the definition of connectivity used by CBSC in detail and also distinguish it from the connectivity used by methods such as DBSCAN and (K, r) clustering.

For clustering, Definitions 4.1, 4.2, and 4.3 provided below are used by DBSCAN [22] and (K, r)-clustering [181]. These definitions are used similarly by Kailing et al. [20] to identify dense units for the detection of biclusters. These conditions stipulate that for any pair of observations x and y within a cluster, there exists a path from x to y such that each edge of the path is shorter than r, and each vertex on the path encompasses at least K observations within its neighborhood of radius r.

CBSC uses additional constraints given in Definitions 4.4 and 4.5 to identify dense regions in 2-dimensional space. This ensures that for each pair of consecutive observations on a path, their neighborhoods have at least δ points in common. Thus, CBSC uses a stronger condition to cluster observations in two-dimensional space but a milder condition for combining features in a bicluster. We formally present the specific conditions implemented by CBSC. Let X represent all points in a data matrix.

Definition 4.1. *Given $(X, dist)$ and $r > 0$, a non-empty subset Cl of X is called $r-$connected, if for each pair of points (x, y) in Cl there is a $r-$chain of points*

connecting x and y, that is, there exists a sequence of points $x \equiv x_1, x_2, x_3, \cdots, x_s \equiv y$ such that $dist(x_i, x_{i+1}) < r$ for $i = 1, 2, \cdots, (s-1)$, where dist denotes distance.

Definition 4.2. *Given $(X, dist)$, $r > 0$, and a positive integer K, a non-empty subset Cl of X is called $(K, r) - bonded$, if for each element $x \in Cl$ there is a subset $T \subset X - \{x\}$ having K elements such that $t \in T \implies dist(x, t) < r$. In other words, x has a neighborhood $N(x)$ which is an open disk of radius r and contains at least K points except x.*

Definition 4.3. *Given $(X, dist)$, $r > 0$, and a positive integer K, a non-empty subset Cl of X is called $(K, r) - connected$, if Cl is $r - connected$ and $(K, r) - bonded$.*

Ling has used $(K, r) - bonded$ sets to identify clusters in the data matrix, while DBSCAN uses them to identify the core points and includes the points if they lie within an open sphere of radius r at any core point. In this work, we impose an additional condition that is used to decide whether points lie in the same cluster. We choose to call this property $(r, \delta) - coshared$.

Definition 4.4. *Given $(X, dist)$, $r > 0$, a positive integer δ, a non-empty subset Cl of X is called $(r, \delta) - coshared$ if for each pair of points (x, y) in Cl there exists a sequence of points $x \equiv x_1, x_2, x_3, \cdots, x_s \equiv y$ such that $dist(x_i, x_{i+1}) < r, \forall i$ and $card(N(x_i) \cap N(x_{i+1})) > \delta$ for $i = 1, 2, \cdots, (s-1)$, where $N(x)$ denotes an open disk of radius r around x.*

Definition 4.5. *Given $(X, dist)$, $r > 0$ and positive integers K and δ, a non-empty subset Cl of X is called $(K, r, \delta) - connected$, if Cl is $(K, r) - bonded$ and $(r, \delta) - coshared$.*

In this work, we identify the edges that would potentially lie in the subspace cluster by finding $(K, r, \delta) - connected$ subsets for each pair of features. Note that

the underlying property of connectedness between each pair of elements contained in the cluster is reflexive, symmetric, and transitive. Therefore, the clusters form an equivalence class. An example in which additional constraints help to find true clusters can be seen in Figure 4.1.

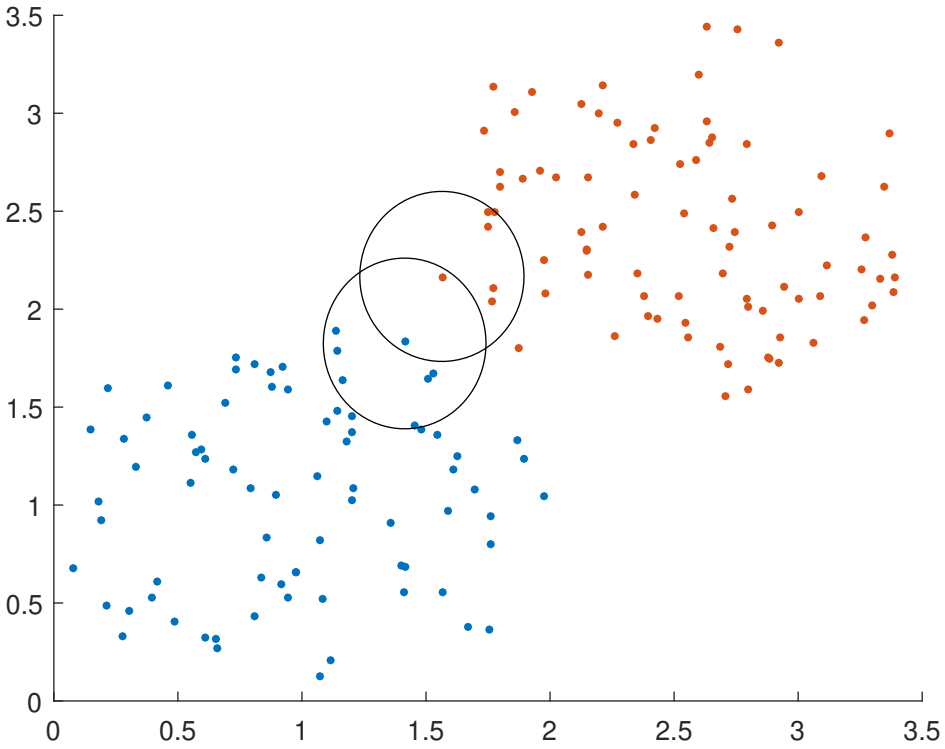


Figure 4.1: Given set of points is (K,r) connected for K=5. Adding a co-sharing constraint with $\delta = 3$ separates the set of points into 2 clusters

4.3.1 Automatic selection of parameters for density-based edge identification

The effectiveness of density-based clustering algorithms, in which density is estimated using a neighborhood defined by a radius r, depends on the determination of an

optimal radius. It is necessary to select a radius that is large enough to represent continuity in the underlying distribution. However, it is also essential to avoid an excessively large radius, which may lead to loss of details.

In the past, the square root of the average length of the edge of the minimal spanning tree has been used as the radius to estimate the alpha cover [182]. It has also been used to estimate normalized mutual information [183]. In this work, we use a function of the average edge length of the minimal spanning tree to find an appropriate value of r.

We first normalize the data to the interval $[0, 1]$, using one of the two transformations described in Section 4.5.1.1. For a pair of features, let l^* be the length of the minimal spanning tree obtained from the points lying in the two-dimensional space after normalization. We use the function $(l^*/n)^c$ as the radius of the neighborhood for clustering, where $0 < c < 1$. The density estimates using this neighborhood will be consistent with Parzen's criteria [132]. This has been shown in Lemma 4.1. It can be seen that the value of c can be varied in the interval $(0, 1)$ without losing the consistency of density estimates. Therefore, c can be used to control the resolution at which the clustering is done. The value $l^*/n < 1$ is shown in Lemma 4.2. Therefore, a smaller value of c will lead to a larger neighborhood and vice versa.

Lemma 4.1. *For a given set of points $z_1, z_2, \cdots, z_n \in \mathbf{R^2}$ if the length of the minimal spanning tree is denoted by l^*, the density estimation can be performed using open spheres of radius $(l^*/n)^c$, where $0 < c < 1$.*

Proof. According to Parzen, for a consistent estimate of volume V_n of the window

used for density estimation should fulfill the following criteria

$$\lim_{n\to\infty} V_n \to 0,$$

$$\lim_{n\to\infty} nV_n \to \infty.$$

According to Steele and Snyder [184], we know that for any data lying in a unit square $[0,1]\times[0,1]$, the length of the Minimal spanning tree l^* converges to $O(\sqrt(n))$. For open spheres in $\mathbf{R^2}$, the volume is given by $V = \pi r^2$. So, the volume of the window would converge to πn^{-c}. This satisfies the conditions given by Parzen for $0 < c < 1$. □

Remarks: Successful identification of dense regions depends on consistent density estimates. This theorem shows that the parameters chosen for the Parzen windows lead to consistent density estimates.

Lemma 4.2. *For any n number of points in a unit square, the length of a minimal spanning tree is less than n.*

Proof. Let $z_1, z_2, \cdots, z_n$ be a set of n points lying in a unit square. Let the unit square be divided into 4 quadrants. We can always construct a spanning tree by connecting the minimal spanning trees for points lying in each quadrant. Let the length of the minimal spanning tree be denoted by l^*. Let the length of the minimal spanning tree lying in each quadrant be denoted by l_i^*. Let the number of points lying in each quadrant be given by n_i for $1 \leq i \leq 4$. Note that $n = \Sigma n_i$. Since any spanning tree must be longer than a minimal spanning tree, we can say that $l^* \leq \Sigma_{i=1}^4 l_i^* + l_c^*$ where l_c^* denotes the total length of edges connecting spanning trees in different quadrants.

If all points are in a single quadrant, each edge is less than $1/\sqrt{(2)}$. Therefore, the overall length is less than $(n-1)/\sqrt{(2)} < n$.

If all points are in two quadrants, then the edge connecting minimal spanning trees lying in different quadrants cannot be longer than $\sqrt{(2)}$. The length of the edges lying in each quadrant cannot exceed $1/\sqrt{(2)}$. So, $l^* \leq \sqrt{2} + \Sigma_{i=1}^{4} \frac{\max(n_i - 1, 0)}{\sqrt{2}} \leq \sqrt{2} + \frac{n-2}{\sqrt{2}} = \frac{n}{\sqrt{2}} < n$.

If all points lie in three quadrants, then the edge connecting the minimal spanning trees lying in adjacent quadrants cannot be longer than $\sqrt{5}/2$. The length of the edges lying in each quadrant cannot exceed $1/\sqrt{(2)}$. So, $l^* \leq 2\frac{\sqrt{5}}{2} + \Sigma_{i=1}^{4} \frac{\max(n_i - 1, 0)}{\sqrt{2}} \leq \sqrt{5} + \frac{n-3}{\sqrt{2}} \leq n$ for $n \geq \frac{\sqrt{10}-3}{\sqrt{2}-1} \approx 0.3918$. Since points are scattered in 3 quadrants, this condition is true.

If points are in the four quadrants, then the edge connecting minimal spanning trees lying in adjacent quadrants cannot be longer than $\sqrt{5}/2$. The length of the edges lying in each quadrant cannot exceed $1/\sqrt{(2)}$. So, $l^* \leq 3\frac{\sqrt{5}}{2} + \Sigma_{i=1}^{4} \frac{\max(n_i - 1, 0)}{\sqrt{2}} \leq 3\frac{\sqrt{5}}{2} + \frac{n-4}{\sqrt{2}} \leq n$, if $n \geq \frac{3\sqrt{5}-4\sqrt{2}}{2-\sqrt{2}} \approx 1.7948$. Since points are scattered in 4 quadrants, this condition is true.

□

Remarks: The previous theorem showed that density estimates will be consistent for any value of c in the interval $(0, 1)$. This theorem shows that $l^*/n < 1$. Thus, the radius of the window used for density estimates will increase as the value of c decreases and vice versa. Thus, for coarser estimates, we need to use a smaller value of c.

4.4 The idea of combining high-density regions to form biclusters

Existing density-based subspace clustering algorithms like SUBCLU [20] use a technique similar to DBSCAN from the bottom up. However, in this work, our goal is to identify important relations within subspaces. To determine the set of interrelated points, we identify $(K, r, \delta) - connected$ subsets within the two-dimensional Euclidean space defined by each pair of features, using Definitions 4.1-4.5. For any triplet of features f_1, f_2, f_3, should the intersection between the three identified subsets exhibit a significant size, it is possible to determine notable edges. These edges serve as connections between the points within the intersection and the features f_1, f_2, f_3. It is evident that the addition of features to this subspace cluster does not increase the number of observations possible within the respective subspace cluster. Consequently, it can be stated that a maximal set of observations is achieved, which can be included within a subspace cluster characterized by these three features. Upon identification of this set, additional feature-triplets that display substantial overlap with the corresponding $(K, r, \delta) - connected$ subsets and identified points are incorporated.

Identifying a pair of features f_1, f_2 that give rise to a $(K, r, \delta) - connected$ set S', we get an idea of the connectivity between features f_1 and f_2. In many application areas, this connectivity would point to potential causal relationships between variables under a given set of conditions. To overcome the effect of noise, we consider only those points that result from the intersection of $(K, r, \delta) - connected$ subsets, obtained by three pairs of features, say, (f_1, f_2), (f_2, f_3), and (f_1, f_3). In terms of graphs, the edge between features f_1, f_2 is admitted only if there exists a feature $f \notin \{f_1, f_2\}$ such that edges (f_1, f), (f, f_2) also occur for the same set of points. In this case, the features

f_1 and f_2 are directly connected with reference to a set of points S. The features f_1 and f_2 are said to be connected with reference to the point set S if there is a path between them. Thus, the idea of connectivity between features is used to find a set of features corresponding to a subspace cluster using an approach similar to that of connected components.

4.5 Details of the procedure CBSC and algorithm

This section describes the procedure for CBSC in detail. The pseudocode for CBSC has been provided as Algorithms 4.1 and 4.2. Figure 4.3 briefly represents the functioning of CBSC.

4.5.1 Biclustering based on dense areas

4.5.1.1 Preprocessing data

As discussed in the previous section, consistent density estimates can be achieved using a Parzen window with a radius given by the average length of edges of the minimal spanning tree for a compact set. However, in many cases, the range of data is not known, and hence we cannot treat the data as compact. To overcome this problem, we need to normalize the data so that the points in each dimension lie in a closed interval $[0, 1]$. For this we have applied transforms $norm_1$ or $norm_2$ described as equations 3.49 and 3.50 in Chapter 3. The transformation $norm2()$ maps the data from the interval $(-\infty, \infty)$ to the interval $(0, 1)$, that is, a subset of $[0, 1]$. Thus, we can say that $norm2()$ maps data from an unbounded interval to a bounded interval. Transformation $norm1()$ is commonly applied to map data to the interval $[0, 1]$.

4.5.1.2 Finding dense areas for pair of features

The next step is to find dense regions for each pair of dimensions. This is carried out using $(K, r, \delta) - \textit{connectivity}$ as defined in Section 4.3. Here, r is calculated as a function of the length of the minimal spanning tree in a 2-dimensional space l^*. The radius for density-based connectivity is taken as $r = (l^*/n)^c$. The value of the parameter K is taken as $\pi r^2 n$. This is the threshold used to find dense regions. Since we have scaled the data to $[0, 1] \times [0, 1]$, this is equivalent to the average number of points that can be found in a disc of radius r. The parameter δ reflects the minimum number of points that lie in the overlapping region between two disks, and the value used for δ is 2.

Suppose that the number of points that form a dense region for the dimension pair $< i, j >$ is n_{dense_ij} then the value $log(n_{dense\_ij})/n_{dense\_ij}$ is used as the threshold to find connected components within the selected dense points, i.e., two points are in different sets if the MST of dense points in the 2-dimensional space contains no path between these points such that each edge that forms the path is smaller than $log(n_{dense\_ij})/n_{dense\_ij}$.

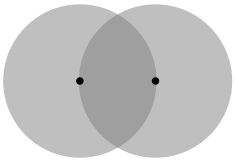


Figure 4.2: The overlapping area between open disks of radius r, is used to find the number of points which should be co-shared

4.5.1.3 Identifying seed biclusters as the overlap of dense regions for three pairs

Each pair of dimensions may have zero, one, or many connected regions. A list of these dense regions with the corresponding dimensions is maintained. For each triplet of dimensions $< i, j, k >$, we already have a list of dense regions for $< i, j >$, $< j, k >$, and $< i, k >$, and we find intersections between these dense regions. If we find a significant overlap, this dense region is stored with the corresponding dimensions as $< i, j, k >$. Significant overlap is said to occur if the number of points found in the intersection of three sets exceeds a user-defined threshold named `MinCompts`. These dense regions are called seed biclusters. Each seed bicluster has a corresponding set of observations and feature triplets.

4.5.1.4 Forming biclusters by merging bases

The list of seed biclusters is further processed as follows. For each seed bicluster S^* with the corresponding set of observations S_o^* and the feature triplet $S_f^* = (f_1, f_2, f_3)$ we initialize the feature list for the cluster based on S^* as $Clf = (f_1, f_2, f_3)$. Recall that several seed biclusters were identified in the previous step. We have initialized Clf using S_f^*. Now we check whether other seed biclusters can be merged with Clf. To do this, we check if a seed bicluster has a feature present in Clf. If such a feature is present, we find the number of observations shared by this seed bicluster and S^*. A user-defined parameter `RatioMerge` is used to decide whether the seed biclusters have a similar set of observations. Specifically, a seed bicluster S (having an observation set S_o and a feature set S_f) is included in the cluster based on S^* if $Clf \bigcap S_f \neq \emptyset \wedge length(S_o^* \bigcap S_o) >$ `RatioMerge` $\times\, length(S_o^*)$. If this condition is satisfied, Clf is updated to $Clf \cup S_f$, and the list of observations S_o is saved so that

it can be used to calculate the list of observations Cl after all seed biclusters similar to S^* have been found. We call S^* the base of the thus-generated cluster.

Another parameter `Reuse_All_Bases` is used to decide whether a dense region having a high similarity to S^* will be used as the base to find new biclusters. If this parameter is set to True, all seed biclusters are used as the base for finding biclusters. If the parameter `Reuse_All_Bases` is set to False, the parameter `Reuse_Base_Ratio` is used to identify dense regions, which will act as the basis for the discovery of new biclusters. The seed biclusters having an overlap of more than $\texttt{Reuse\_Base\_Ratio} \times \texttt{RatioMerge} \times length(S^*_o)$ with the base S^*, are marked with the label *ignore*. These seed biclusters will not be used as bases for the search for new biclusters. The parameter `Reuse_Base_Ratio` is not required if `Reuse_All_Bases` is set to True.

The merging process is repeated for the base S^* until no more seed biclusters can be added. After completion of the process of merging all seed biclusters with S^*, the observations shared by the base and at least two other similar seed biclusters (identified to be similar to S^* in the previous paragraph) are used as a set of observations Cl contained in the bicluster based on S^*. Thus, Cl need not contain all the points in S^*_o. In this way, we get the cluster $< Cl, Clf >$ based on S^*. Similarly, we find biclusters based on other unmarked seed biclusters.

Once we have found biclusters using the above-mentioned procedure, we will remove biclusters that are very similar to each other. The similarity between two biclusters is defined as the product of cosine similarity between the set of points and cosine similarity between the set of features. Let the observations corresponding to the first bicluster be O_1 and the second bicluster be O_2. Let F_1 and F_2 denote the feature sets for the first and second biclusters. The cardinality of the set O_1 is denoted by $Card(O_1)$. Then the similarity between two biclusters is defined as

$\frac{Card(O_1 \bigcap O_2)}{\sqrt{(Card(O_1) \times Card(O_2))}} \times \frac{Card(F_1 \bigcap F_2)}{\sqrt{(Card(F_1) \times Card(F_2))}}$. If two clusters have a similarity greater than a user-defined threshold denoted by `Clus_Overlap`, the smaller of the two clusters is discarded. While `Reuse_Base_Ratio` and `Reuse_Base_Ratio` are used to specify the maximum overlap between cluster bases in terms of observations, `Clus_Overlap` is used to specify the overlap between clusters in terms of both observations and features.

We now have the list of biclusters contained in the data matrix. In the following section, we present an approximation of this method which has a lower time complexity.

4.5.2 Approximation using grid

The algorithm consists of 4 steps described in 4.5.1. Of these, the second step of finding densely connected areas for all pairs of dimensions is the most time-consuming. An approximation of this step can be implemented by mapping each two-dimensional space $[0, 1] \times [0, 1]$ to a cell of size $n_g \times n_g$ superimposed on $[0, 1] \times [0, 1]$ space. After this step, all feature values are assigned to the maximum $n_g \times n_g$ points in the grid space. It may be noted that some of the cells may remain empty. These cells are ignored. Let the number of cells populated be n_p. Now, we find the MST between these n_p points. Let the length of MST in 2-dimensional space be l_g^*, then the radius for density-based connectivity is taken as $r = (l_g^*/n_p)^c + 1/n_p$. As before, disks are dense if they have more points than $\pi r^2 n$. The disk is now not perfectly smooth; all the points that populate cells within the radius r are considered. A cell representing the area $(x_1, x_2) \times (y_1, y_2)$ is said to be within radius r if the corner (x_1, y_1) lies within radius r. If this number is greater than $K = \pi r^2 n$, the disk is said to be dense.

The parameter δ reflects the minimum number of points in the overlapping region

between two disks and is calculated as $\delta = 0.6142r^2n$. This threshold is taken as half the overlap area between two disks of radius r such that the center of one lies on the perimeter of the other. It is given by $r^2(\pi/3 - \frac{\sin(\pi/3)}{2}) \approx 0.6142r^2$.

Suppose that the number of cells that form the dense region of the dimension pair $< i, j >$ is given by n_{dense_ij}, then the value $log(n_{dense\_ij})/n_{dense\_ij}$ is used to find connected components within the selected dense cells, that is, two points are in different sets if the MST of dense points in a 2-dimensional space contains no path between these points such that each edge forming the path is smaller than $log(n_{dense\_ij})/n_{dense\_ij}$.

Thus, we identify cells that form clusters in a two-dimensional space. All the points contained in these cells are taken to be points forming dense regions in the following steps as before.

It may be noted that the approximation method has been used on all datasets in which the number of observations is at least 1000.

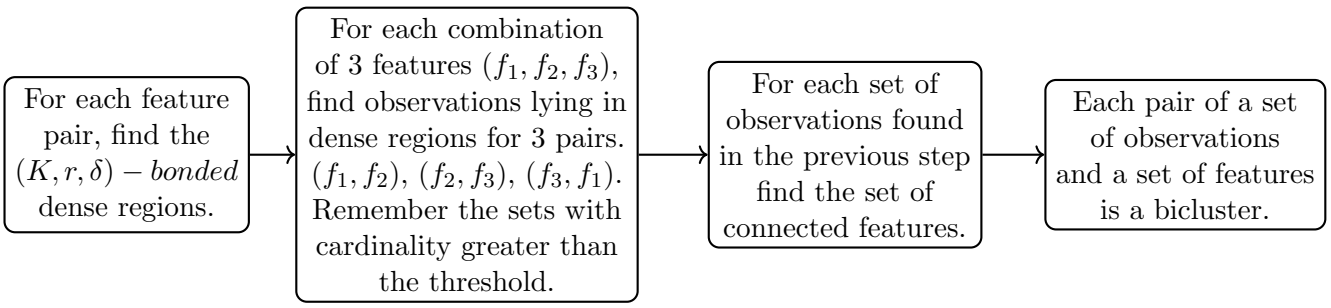


Figure 4.3: Broad outline showing how CBSC works

4.5.3 Parameters required for CBSC

CBSC uses several user-defined parameters that have previously been mentioned in Section 4.5.1. A list of these parameters is given below for convenience. We also presented a discussion of factors that affect the choice of these parameters.

1. c: This parameter is used to find the radius of an open disk in the formula $r = (l^*/N)^c$, where l is the length of the minimal spanning tree and N is the number of points in the data. Recall that r is the radius of neighborhood used to find dense regions in two-dimensional space.

2. `RatioMerge`: CBSC compares the base seed biclusters with other seed biclusters and if the number of common observations between the two is at least `RatioMerge` $\times$ *'number of observations in base cluster'*, then both of these seed biclusters are said to have a similar set of observations. We suggest using a value in the range $[0.75, 0.99]$ for `RatioMerge`.

3. n_g: Number divisions made along each dimension to form the grid. The value of this parameter is taken as 10. Increasing the value will increase the execution time of the method.

4. `Reuse_All_Bases`: If this parameter is set to True, every seed bicluster found using procedure descibed in Section 4.5.1.3, is used as a base to be compared with other seed biclusters, otherwise, seed biclusters with high overlap with other seed biclusters are not used as the base to form new biclusters.

5. `Reuse_Base_Ratio`: If the parameter `Reuse_All_Bases` is set to False, seed biclusters having an overlap of more than `Reuse_Base_Ratio` $\times$ `RatioMerge` $\times$ *'number of observations in base cluster'* with the current base seed bicluster are not used as base for forming new clusters. Setting `Reuse_All_Bases` to True allows the method to find all the overlapping clusters. In case of overlapping biclusters, if `Reuse_All_Bases` is set to False, the number of points discovered might decrease when `Reuse_Base_Ratio` $\leq \frac{\text{number of overlapping observations}}{\text{number of observations in smaller bicluster}}$. This happens because dense regions are not treated as the base if they have

a greater overlap than `Reuse_Base_Ratio` with a larger base.

We suggest using `Reuse_All_Bases` = $True$ if we want to find all the overlapping biclusters. In this case, the parameter `Reuse_Base_Ratio` is not used. If `Reuse_All_Bases` = $False$, the value of `Reuse_Base_Ratio` is used. Let the maximum overlap between bicluster observations be defined as the ratio of the number of common observations in both biclusters divided by the number of observations in the smaller cluster. The value of `Reuse_Base_Ratio` should be set to the desired maximum overlap.

6. `MinCompts`: A dense region is said to be significant if it contains at least `MinCompts` numbers of points. `MinCompts` is a parameter that controls the size of the biclusters. Choosing a value $n_{\texttt{MinCompts}}$ will give all biclusters for which the size of the base seed bicluster used to obtain the bicluster is at least $n_{\texttt{MinCompts}}$. Here, the base seed bicluster is the one that is compared to all other seed biclusters in step 4.5.1.4 of the algorithm. Consequently, it will also give all biclusters with the number of points greater than $n_{\texttt{MinCompts}}$.

7. `Clus_Overlap`: This parameter is used to remove the bicluster that has a cosine similarity greater than `Clus_Overlap`. The cosine similarity used here is defined in Section 4.5.1.4. `Clus_Overlap` is used to drop off biclusters that have a high similarity. This is done in the last stage after the search for biclusters is complete.

In this chapter, several experiments have been performed with fixed parameter values that are mentioned in Section 4.7. Later, in Chapter 6, the results for CBSC have been aggregated over several possible parameter values, providing insight into the behavior of CBSC when different parameter values are selected.

Algorithm 4.1: main()

Normalize data $\mathbf{D}$
for *each pair of features* $< i, j >$ **do**
    $Pts(i, j, 1), Pts(i, j, 2), \cdots$=DenseRegions($\mathbf{D}_{*,fi}$, $\mathbf{D}_{*,fj}$)
$List_S = \emptyset$
`/* Suppose` q `dense regions are found for features` i `and` j, `*/`
`/* then` $1 \leq t \leq q$, `Similarly consider each dense regions */`
`/*` u `for` $< j, k >$ `and` v `for` $< i, k >$ `*/`
for *each triplet of 3 features* i, j, k *and each dense region* t, u, v **do**
    $S_o = Pts(i, j, t) \cap Pts(j, k, u) \cap Pts(i, k, v)$
    if *length(*$S_o <$ `MinCompts`*)* **then**
        delete (S_o)
    else
        $S_f =< i, j, k >$
        $S =< S_o, S_f >$
        Add S to $List_S$
Sort $List_S$ in descending order of length of S_o.
for *each* $S \in List_S$ *not marked as ignore* **do**
    $Clf = S_f$
    do
        for *each* $W \in List_S$ **do**
            if $(Clf \bigcap W_f) \neq \emptyset \wedge length(S_o \cap W_o) >$ `RatioMerge` $\times\, length(S_o)$ **then**
                $Clf = Clf \cup W_f$
            if `Reuse_All_Bases` $== False \wedge$ $length(S_o \cap W_o) >$ `Reuse_Base_Ratio` $\times$ `RatioMerge` $\times\, length(S_o)$ **then**
                Mark W as ignore
    while *dimensions are added to* Clf *in last iteration*;
    $Cl =$ Observations present in S_o and atleast two other seed biclusters present in $List_s$
    if $Cl \neq \emptyset$ **then**
        $Clusterlist = Clusterlist \cup < Cl, Clf >$

Algorithm 4.2: DenseRegions

Tr = Minimal spanning tree $(\mathbf{D}_{*,i}, \mathbf{D}_{*,j})$ /* n = number of observations
$radius$=$(length(Tr)/n)^c$
$K = (radius^2)n\pi$ /* average number of points in radius
for *each pair of points* z_i, z_j **do**
    if *distance* $(z_i, z_j) < radius$ **then**
        connection$(z_i, z_j) = 1$
    else
        connection$(z_i, z_j) = 0$
        Add z_j to $neigh(z_i)$
        Add z_i to $neigh(z_j)$
$finset = \emptyset$
if $Card(neigh(z_i)) > K$ **then**
    $finset = finset \cup z_i$
do
    for *each pair of points in finset* $< z_i, z_j >$ **do**
        if $Card(neigh(z_i) \cap neigh(z_j)) > \delta$ **then**
            merge connection$(z_i, :)$ and connection$(z_j, :)$
while *no more connections are merged*;
Remove points that have not merged with any points from $finset$
for *mergeSet=each set of points obtained after merging* **do**
    $npc = Card(mergeSet)$
    Find connected components in *mergeSet* using radius $rad = \ln(npc)/npc$
    $DenseRegions = \emptyset$
    for *connected components numbered* $i_{cc} = 1, 2 \cdots, m_{con\_comp}$ **do**
        $DenseRegions = DenseRegions \cup$ *connected component* i_{cc}
Return *DenseRegions*

4.6 Time complexity of CBSC

The time complexity of CBSC without approximation is $M^2N^3 + M^3N$, where N is the number of points, M is the number of dimensions in the data from which the biclusters will be searched. However, the time complexity after implementing the approximate solution of CBSC is $CM^2 + M^3N + m^2N$, where C is a constant. Thus, this approximate solution is linear in the number of points. It is seen that the second term is cubic in a number of dimensions. This is because the dense area found for a pair $< i, j >$ of dimension is compared to the dense area found for all pairs that have $< i, p >$ and $< j, p >$ with $p = 1, 2, \cdots, m$. In practice, this value is likely to be much smaller than M^3, as we only need to consider dense areas that have the minimum number of points defined by the parameter `MinCompts`.

The method Proclus [127] works in a top-down manner and searches for k clusters, it has a time complexity $O(Nm + kNM)$. This method is biased towards biclusters that are hyper-spherical in shape. UniBic [145] has a time complexity $O(N^2M^2)$. The algorithm CLIQUE [128] is linear in N but exponential in $m' = max(m_1, m_2, \cdots, m_k)$, i.e. the maximum dimensionality of a bicluster. The complexity of CLIQUE is given by $O(C^{m'} + Nm')$. SUBCLU also has a runtime proportional to $C^{m'}$.

4.7 Experimental setup and methodology for analysis with CBSC

In this section, we present the methodology used to perform experiments with synthetic and real-world datasets. Real-world datasets include five UCI-ML [185] datasets and two datasets designed to understand the spread of COVID-19 disease.

4.7.1 Experimental setup for synthetic datasets

To test the efficacy of CBSC, we conducted experiments on synthetic and real-world datasets. Experiments with synthetic datasets allow us to study the behavior of CBSC on datasets with known characteristics. To check whether CBSC can detect biclusters based on linear and non-linear relationships, we have experimented with several synthetic datasets. We also wanted to see the effect of applying different data transforms on the performance of the algorithm.

Each column in Tables 4.2-4.5, corresponds to datasets with particular characteristics. For each column, ten instances of synthetic datasets are generated using the method given in Section 4.7.1.2, and the average accuracy and standard deviation obtained by different algorithms are reported. The parameters used for CBSC are `MinCompts` $= n/8$, `RatioMerge` $= 0.85$, `Reuse_All_Bases` $= False$, `Reuse_Base_Ratio` $= 0.75$, `Clus_Overlap` $= 1$, $n_g = 10$ and $c = 0.85$, where n is the number of observations in the dataset. For datasets "Normal" and "Noisy Normal" data is normalized using $norm_2$. For all other datasets $norm_1$ is used.

4.7.1.1 Methods used for comparison with CBSC

The performance of CBSC is compared with six other algorithms namely CLIQUE [128], Proclus [127], ITL [186], Subclu [20], P3C [125], and UniBic [145]. The methods CLIQUE, P3C and Subclu are density-based. ITL identifies biclusters by maximizing mutual information between subsets of rows and columns, aiming to uncover locally informative patterns in the data. Proclus is a k-medoid based method for subspace clustering. Comparison with UniBic is important as it specifically aims at finding biclusters based on non-linear dependence. For UniBic, P3C, Subclu, Proclus, and CLIQUE, the R packages 'runibic' and 'subspace' have been used for the experiments

with default parameters [187, 188]. For ITL, the Matlab Toolbox for Biclustering Analysis has been used with default parameters [189].

4.7.1.2 Methodology used to generate synthetic datasets

This section presents the methodology used to generate synthetic datasets that are used to analyze the properties of CBSC. The datasets for Table 4.2 require several functions which are reported in Table 4.1. The rest of the data is generated using linear functions.

Table 4.1: Functions used to generate Non-Monotonous, Non-Linear 1 and Non-Linear 2 datasets. Non-Linear 2 uses similar functions as Non-Linear 1, but the range of the features is scaled and translated to interval [0,1].

h_i	Non-Monotonous	Non-Linear 1	Non-Linear 2
$h_1(z)$	$Id(z) = z$	$Id(z) = z$	$Id(z) = z$
$h_2(z)$	$0.5\sin(2\pi z) + 0.5$	$\sin(z)$	$\sin(z)$
$h_3(z)$	$0.5\sin(3\pi z) + 0.5$	z^2	z^2
$h_4(z)$	$0.5\sin(15\pi z) + 0.5$	z^{10}	z^{10}
$h_5(z)$	$0.5\sin(25\pi z) + 0.5$	$\sin(\pi z)$	$0.5\sin(\pi z)$
$h_6(z)$	$2z$	$\sin(2\pi z)$	$0.5\sin(2\pi z) + 0.5$
$h_7(z)$	$\sin(2\pi z) + 1$	z^3	z^3
$h_8(z)$	$\sin(3\pi z) + 1$	$4z^2$	z^2
$h_9(z)$	$\sin(15\pi z) + 1$	$\sin(4\pi z)$	$0.5\sin(4\pi z) + 0.5$
$h_{10}(z)$	$\sin(25\pi z) + 1$	$4z^3$	z^3

In Table 4.2, the first column shows the results for the datasets generated using the process given below. Initially, a random matrix of size 1000×20 is generated. Subsequently, the submatrix 500×10 of the data is substituted so that the column i^{th} of this submatrix is defined by $h_i(z)$, where z denotes the value of the first column of the submatrix, and the definitions of h_i for $i = 1, 2, \cdots, 10$ are detailed in the second column of Table 4.1. Consequently, a bicluster of size 500×10 is embedded within the data. These functions result in a bicluster based on non-monotonous feature relationships.

Similarly, the second column of Table 4.2 shows the results of the data generated using the functions in the third column of Table 4.1. Here we may note that some of the values generated using these functions may lie outside the interval $[0, 1]$, making the marginal densities in two-dimensional plots highly variable.

The data sets used for the results shown in the third column of Table 4.2 are generated using functions reported in the fourth column of Table 4.1. These functions are modifications of the functions given in the third column of Table 4.1, such that the data in the bicluster lie in the same range as the rest of the data. It may be noted that the function z^2 occurs both as h_3 and h_8 for the dataset, and z^3 occurs both as h_7 and h_{10}. Thus, some of the columns in the bicluster are identical to each other. However, this has been done, so that the functions used to generate the data are similar to those used for the second column and differ only in the range of the data.

In the first column of Table 4.3, the results derived for datasets of size 1000×20 drawn from a uniform distribution are presented. The dataset incorporates a bicluster of size 500×10, generated using functions $h_1(z) = Id(z) = z$, where I denotes the identity function, and $h_i(z) = a_i * z$ with $i = 2, 3 \cdots, 10$. The variables a_i with $i = 2, 3, \cdots, 10$ represent random values within the interval $(0, 1)$. This constitutes our base data, generated utilizing uniform distribution datasets. Different transformations have been applied to the base data for the purpose of analyzing the properties of biclustering algorithms, and the respective results are presented in the following columns.

The second column of Table 4.3 presents the dataset results derived by applying a scaling factor, selected randomly from the interval $(0, 1)$, to each feature in the datasets initially generated for the first column of Table 4.3.

The third column of Table 4.3 presents the results obtained for the datasets by

adding a random number within the interval $(0, 1)$ to each feature of the dataset generated for the first column of Table 4.3. These results facilitate the analysis of the algorithms' scaling and translation properties.

The results presented in the fourth column of Table 4.3 pertain to datasets derived through the application of a linear transformation to each feature of the datasets originally produced for the first column of Table 4.3. This transformation encompasses aspects of both scaling and translation. For each column, two random numbers a_1 and a_2 are generated, upon which the transformation $a_1 z + a_2$ is executed. Notably, scaling, translation, and linear transformations are specialized instances of distance-preserving transformations.

The fifth column of Table 4.3 presents results derived from datasets in which each observation of the dataset used for the first column of Table 4.3 has been duplicated. Consequently, these datasets comprise 2000 observations.

In the sixth column of Table 4.3, the results for the datasets, which have been produced by duplicating each observation within the bicluster of the datasets generated for the first column in Table 4.3, are presented. Consequently, these datasets have 1500 observations.

The seventh column of Table 4.3 presents the results for the datasets generated by adding a random value within the range [0,0.1] to each element of the data matrices. This is equivalent to adding random noise sampled from a uniform distribution within the specified range. This methodology is utilized to evaluate the algorithms' robustness against noise.

The eighth column of Table 4.3, presents the results for the datasets derived by random permutation of rows and columns from the data generated for the first column of Table 4.3.

The first column of Table 4.4 presents the results for datasets of dimension $1000 \times$

20 derived from a Gaussian distribution, wherein the bicluster submatrix of size 500×10 is constructed in a manner analogous to the first column of Table 4.3.

The second column of Table 4.4 presents the results for datasets subjected to Gaussian noise, characterized by a standard deviation of 0.1, added to the data matrix initially used to obtain the results reported in the first column of Table 4.4. This analysis serves to evaluate the robustness of the algorithm against Gaussian-distributed data.

The third column of Table 4.4 presents results from datasets that have Gaussian noise, added to each element of the first column data matrix, using a standard deviation 0.2 to assess the robustness of the algorithm.

The first and second columns of Table 4.5 contain results for datasets containing two overlapping biclusters. The data is generated using uniform distribution. Each row of biclusters has a constant value chosen randomly from the interval [0,1]. The first column shows the accuracy for a bicluster with 500 rows and 10 columns. The second column shows the accuracy for another bicluster with 400 rows and 10 columns. There is an overlap of 200 rows and 3 features in both biclusters (12% of the larger bicluster).

Similarly, the third and fourth columns of Table 4.5 contain results for datasets that contain two overlapping biclusters. Data is generated using a uniform distribution, and each row of the biclusters has a constant value in the range [0,1]. The third column shows the accuracy for a bicluster with 500 rows and 10 columns. The fourth column shows the accuracy for another bicluster with 400 rows and 10 columns. There is an overlap of 200 rows and 5 features in both biclusters (20% of the larger bicluster).

4.7.1.3 Evaluating the performance on synthetic datasets

For the synthetic datasets, the performance evaluation is performed as follows. For data in an $N \times M$ matrix, accuracy is calculated in following way: For each element in the dataset, let $e_i^j = 1$ if the element in row i and column j is selected; otherwise $e_i^j = 0$. Similarly, for each element in the dataset, let $a_i^j = 1$ if the element belongs to the actual bicluster, and $a_i^j = 0$ otherwise. The accuracy is calculated as $sum(XNOR(e_i^j, a_i^j))/(NM)$. In other words, an element is said to be correctly identified if it is either present in the actual bicluster and also in the bicluster found by the algorithm, or if the element is absent in both of them. The accuracy for identifying a bicluster is calculated as the expectation of correctness over all the elements of the data matrix. We have reported the best match for each bicluster present in the data. The accuracy for synthetic datasets is reported in Tables 4.2-4.5, and Figures 4.4-4.7. The execution times for the synthetic datasets are reported in Tables 4.6-4.9.

4.7.2 Experimental setup for UCI-ML datasets

Using UCI-ML datasets, we apply CBSC in two ways. First, biclustering algorithms are used to enhance supervised learning by generating new features that improve classification accuracy for two datasets, as discussed in Section 4.8.2.1.

Second, in Section 4.8.2.2, we use CBSC for clustering three datasets and then compare the obtained clusters with known classes in the data. The datasets used for these experiments are taken from the UCI ML repository [185] and are suitable for the classification task. These datasets are from various application areas, namely banking, education, medical science, particle physics, and electrical engineering. The description of the datasets is given in Sections 4.8.2.1 and 4.8.2.2. For the chosen datasets, each feature is presented in numeric form. Each of these datasets has at

least 600 observations and a minimum of 9 features. For each application area, the dataset with the highest number of views is selected. These datasets were also used for feature selection in Chapter 3.

It may be noted that non-linear feature relation based biclustering might not be useful for all datasets. For example, consider the hyperspectral datasets on which feature selection was applied in Chapter 3. In the case of hyperspectral datasets, consecutive bands are related to each other for almost all the observations [190]. Thus, the features in the dataset are directly or indirectly connected for all observations. Thus, analyzing these datasets with feature relation based biclustering is not useful. Thus, hyperspectral datasets have not been analyzed using biclustering methods in this thesis.

For real-world datasets, we initially apply normalization $norm_1$. If no biclusters are produced using $norm_1$, then the biclusters are found after applying $norm_2$ to the dataset.

4.7.2.1 Setup for experiments with classification of UCI-ML datasets

CBSC and various other biclustering techniques have been employed to create new binary features, which help improve the classification accuracy on two UCI ML datasets [185]. These datasets are: (1)"Default of credit card clients" dataset [175] of the banking sector that has 30000 observations with 23 features and two classes. (2) "Predict Student Dropout and Academic Success" dataset [176] from the education sector has 4424 observations with 36 features and three classes.

Biclusters are identified through various methods, including CBSC and six additional techniques. For a specific algorithm, each bicluster is used to create a new feature. If an observation is part of this bicluster, the new feature is assigned a value of 1; conversely, if it is not, the value is 0. This implies that the corresponding

membership value is introduced as a new feature. Subsequently, the Naive Bayes classifier, recognized for its simplicity, is applied following augmentation of the database with these new features. The parameter values for CBSC are chosen using a 10-fold cross-validation. For this, parameters have been tested from the following range. For `MinCompts`, we consider the values $\{N/8, N/16\}$ where N is the number of observations in the dataset. For `RatioMerge`, we have taken the values $\{0.75, 0.85, 0.95\}$. The parameter `Reuse_All_Bases` can be set to $True$ or $False$. We have experimented with both values. For the parameter `Reuse_Base_Ratio`, we have experimented with the values $\{0.25, 0.5, 0.75\}$. For the parameter `Clus_Overlap` we have experimented with the values $\{0.5, 1\}$. We have taken $n_g = 10$ and for c, we have considered the values $\{0.75, 0.85, 0.95\}$. As in the case of synthetic datasets, the default parameter values have been used for other methods.

Performance is evaluated in terms of accuracy, normalized mutual information (NMI) [61], adjusted rand index (ARI), precision, recall, and G-Score [191]. Note that ARI can sometimes attain negative values, and the lower bounds have been well studied [192]. Also note that since the average precision, recall, and G-Score are reported, the reported G-Score values may be less than the geometric mean of the reported precision and recall. The results are reported in Table 4.10 and Figures 4.8a-4.8f.

4.7.2.2 Setup for experiments with clustering of UCI-ML datasets

We apply CBSC for unsupervised learning to see whether the biclusters detected by it correspond to the meaningful structures in the data. We examine whether any of the identified biclusters align with the known classes within the dataset. This analysis was carried out for three UCI ML [185] datasets: (1) "Magic gamma telescope" dataset [177] from the application area named particle physics, which has 19020 observations

with 10 features (2) "Breast cancer (Wisconsin Original)" dataset [178] of the sector of medical science has 683 observations and 9 features. (3) "Electrical Grid Stability Simulated Data" dataset [179] from the application area of electrical engineering, has 10000 observations and 12 features. For these datasets, the parameters used for CBSC are `MinCompts` $= N/8$, where N is the number of observations in the dataset, `RatioMerge` $= 0.85$, `Reuse_All_Bases` $= False$, `Reuse_Base_Ratio` $= 0.75$, `Clus_Overlap` $= 1$, $n_g = 10$, and $c = 0.85$. Since it is not possible to determine the values of the ideal parameters in the case of clustering, we have used fixed values here. Later, in Chapter 6, the performance of CBSC on these datasets has been reported by computing the average over several parameter values.

The datasets used in this analysis comprise two classes, designated as 0 and 1. Define $O_C(i)$ as the indicator of whether observation i belongs to class 1 or 0. Let $O_B(i) = 1$ express the inclusion of observation i in the determined bicluster, and 0 otherwise. For each bicluster, the computation of the number of match between the membership of the bicluster and the class label is calculated as follows:

$$Acc = \frac{max(sum(XNOR(O_B(i),O_C(i))),sum(XOR(O_B(i),O_C(i))))}{N}; \tag{4.1}$$

where N is the number of observations in the dataset and $XNOR$ and XOR are the logical operators. Using this formula, we check whether a bicluster is similar to any one of the two classes present in the dataset. For the bicluster that obtains the best accuracy according to equation 4.1, six performance indices are reported. These indices are accuracy, NMI, ARI, precision, recall, and G-score. The results are reported in Table 4.11 and Figures 4.9a-4.9f.

4.7.3 Setup and methodology for application of CBSC to COVID-19 datset for finding relevant lifestyle factors

We have also used CBSC to find lifestyle factors that impact COVID-19 infection rates. Toward the end of 2019, a highly contagious disease named COVID-19 emerged that caused a pandemic of unprecedented scale. However, it is seen that some countries were more affected by this disease than others. During the early phase of the spread of COVID-19, it was important for policymakers and medical professionals to understand the behavior of the disease. For this, there were attempts to find the relationship between existing lifestyle factors and COVID-19 rates.

One such dataset was created by selecting features from World Development Indicators taken from the World Bank website [193] and the script to generate the dataset was made available at Kaggle [194]. In summary, the creator of the dataset considered 74 features of the World Bank indicators dataset that intuitively seemed related to COVID-19 infection rates and eliminated the features where missing values were greater than 25%. This resulted in a WDI (World Bank Indicators) dataset with 27 columns. The missing values were treated using KNN imputation [195]. The creator then joined the WDI dataset with the COVID-19 rate dataset for correlation analysis. However, the correlation coefficient values at this time were small as the disease had spread to a few countries and the relationship between COVID-19 rates and other features was obscured.

For our analysis using CBSC, we have used the extracted WDI features in the same way as described above and joined this dataset with the table of the cumulative number of confirmed cases of COVID-19 on 31 January 2020 [196, 197].

CBSC did not find a correlation between the features for the entire set of observations. Instead, it finds subsets of observations where features are related to

each other by linear or non-linear relations. This allows CBSC to find relations limited to a smaller subset of data. We may also say that the relation between the selected features for the selected observations is different from the relation between the same set of features for the observations outside the bicluster. Thus, a bicluster that separates countries with a higher occurrence of COVID-19 from other countries would contain features that are more likely to affect the COVID-19 infection rates. Note that traditional clustering methods cannot be used for this analysis as they do not select features. Similarly, feature relation indices like the Spearman correlation, MIDI, cannot be used to find relevant features; as in the earlier stages of the spread of COVID-19, the relation between features and infection rates may exist only for a small number of countries, and thus for a subset of observations. Biclustering is useful in scenarios where subsets of observations as well as features are required together. Thus, we applied CBSC for the analysis of lifestyle factors that impact COVID-19 rates. The details of the methodology and the results are provided in the following sections.

4.7.3.1 Methodology for finding relevant features in COVID-19 dataset

To identify the features that affect the infection rate, we utilize the cumulative number of confirmed COVID-19 cases documented on 31 January 2020. This dataset, in conjunction with the World Development Indicators (WDI) features, serves as input for the CBSC methodology. The infection rate is defined as the number of confirmed cases of COVID-19 per million inhabitants in a specific region. Countries that surpass the 90th percentile in terms of infection rate are identified as countries with high COVID-19 infection rates. The bicluster that has the highest match with the set of countries that exhibit elevated COVID-19 infection rates is selected from the suite of biclusters generated through CBSC. This optimal match is determined by the

accuracy metric obtained according to equation 4.1. It is important to note that this accuracy metric does not reflect the resemblance between the bicluster and the group of countries with heightened COVID-19 infection rates, but rather the bicluster's ability to distinguish between regions characterized by high and low infection rates. Given that biclusters are derived based on the similarity of features among particular observations, one may deduce that the interaction among these features differentiates the specified set of observations from others. Consequently, these features are likely to influence the transmission of the disease.

For this analysis, we used the data available at an early stage on 31 January 2020. We have taken a 90 percentile to cover the major portion of the dataset as the size of the dataset is very small. The main aim was to obtain information from a large part of the available data.

Using data for January 2020, we found that CBSC identified some factors related to the disease occurrence rate. At that time, the Spearman correlations of selected features with disease occurrence were low. However, Spearman correlations of the features selected by CBSC with disease occurrence on 31 December 2020 (after 1 year of disease spread, before the COVID-19 vaccination programs began in many countries) are high (Table 4.12), indicating that the selected features are indeed important. During the early phases of the disease, these relations were obscured by data from countries where the disease was still in its nascent stages. CBSC is able to mine this obscured information as it searches for relations between subsets of data. Early detection of factors that affect the spread of diseases makes CBSC a useful tool for understanding new diseases and epidemics. Here we might note that we have used Spearman correlation to evaluate the association between lifestyle factors and infection rates for December 2020. This is because MIDI works well with datasets that have a distribution that has a compact, path-connected base. Thus, it is not suitable

for analysis with datasets having a discontinuous distribution. As the dataset in this section has several discontinuities, MIDI has not been applied to the analysis of the dataset directly. However, MIDI has been used to design the biclustering method PF-RelDenBi which is applied for analysis of this dataset and will be discussed in Chapter 6. The parameters used for CBSC are `MinCompts` = 10, `RatioMerge` = 0.95, `Reuse_All_Bases` = $False$, `Reuse_Base_Ratio` = 0.5, `Clus_Overlap` = 1, $n_g = 10$ and $c = 0.85$. We have chosen a smaller value of `MinSeedSize` or `MinCompts` to explore the relatively small dataset. A higher value is chosen for `Sim2Seed` or `RatioMerge` so that we can choose features having higher similarity. Similar experiments have also been conducted using UniBic, and a comparison has been presented. UniBic has been used for comparison as it is a method designed to find non-linear feature relations. Results of this analysis are reported in Table 4.12.

4.7.4 Setup and methodology for application of CBSC to identify vaccines existing before COVID-19 that might help in reducing COVID-19 infection rates

In this section, we present the methodology for applying CBSC to discover the new association of vaccines (dataset by WHO [198]) for different countries and the spread of COVID-19. Among various factors that affect the infection rate of COVID-19 in countries, vaccine coverage for existing diseases could contain important information both for policymakers and researchers trying to design drugs/vaccines for the treatment of COVID-19. In the following paragraphs, we present the details of the datasets and methodology used for this analysis to find the associated features of the COVID-19 infection rate.

4.7.4.1 Datasets used for the study

The dataset we used for the experiments includes coverage rates for 50 vaccines in various countries [196, 198]. Such coverage rates are available for the years 1966 to 2010. During the initial spread of COVID-19, it was found that people in higher age groups were more severely impacted by the disease [199]. Thus, biclustering is performed for the vaccine coverage for the years 1966 to 1985. So, we have 20 two-dimensional data matrices, one for each year. Each row in the matrix represents a country, and each column represents data for a particular vaccine. In each of these matrices, we add a feature that represents the COVID-19 infection rates on 31 December 2020 in these counties.

4.7.4.2 Methodology for analysis of COVID-19 and vaccine dataset

We use CBSC to find biclusters in each of the matrices (each matrix includes vaccine coverage for each year and COVID-19 infection rates on 31 December 2020). We proceed by selecting the bicluster for which the set of observations has a maximum match with the set of countries that lie above the 90th percentile in terms of infection rate. To be precise, we have calculated the accuracy using formula 4.1. Since the given set of countries forms the basis for the relation between features, it follows that the features included in the bicluster contain enough information to distinguish countries with low rates of COVID-19 infection from other countries. After the analysis, the features that have been selected several times are likely to be the features that have a relationship with the COVID-19 infection rates.

However, the analysis with CBSC does not indicate whether the increase in vaccine coverage is associated with a decrease or an increase in the COVID-19 infection rate. Thus, for deciding the direction of the relation between infection rate and other

vaccine coverage rates, we use the Spearman correlation sign between the average of the vaccine coverage rate over the years 1966-2010 and COVID-19 infection rates. We select vaccines for which the Spearman correlation is negative, that is, an increase in vaccine coverage is associated with a decrease in the COVID-19 rate. In this way, we could obtain a set of features associated with lower COVID-19 infection rates. Since Spearman correlation evaluates the relation for all observations and not for subsets of observations, its absolute value can be very low and cannot be used to judge the significance of the features. Only the sign of Spearman correlation is used. Note that a dependence measure like MIDI can not be used in place of Spearman correlation because its value lies in $[0, 1]$ and so it does not tell whether the association is positive or negative. In summary, we only consider the features with negative Spearman correlations, as we are interested in identifying vaccines for which increased coverage could help reduce COVID-19 infection rates. Finally, among the features having negative Spearman correlation, we find the feature that has been selected by CBSC the maximum number of times. The parameters used for CBSC are `MinCompts` $= 20$, `RatioMerge` $= 0.75$, `Reuse_All_Bases` $= False$, `Reuse_Base_Ratio` $= 0.75$, `Clus_Overlap` $= 1$, $n_g = 10$ and $c = 0.85$.

It should be mentioned that several studies have been conducted to understand the cross-reactive immunity provided by a specific vaccine against COVID-19 infection. However, we did not find any studies exploring a large number of existing vaccines for this purpose. Therefore, we studied this association using the biclustering approach. This study allows us to explore the potential of using pre-existing vaccines as an additional tool against the spread of COVID-19. This would be especially useful for groups of people for whom the administration of the COVID vaccine has not yet been proven to be safe. The results are discussed in Section 4.8.4.

4.8 Results

This section presents results for experiments described in the previous section.

4.8.1 Results on synthetic datasets

In this section the results for experiments described in Section 4.7.1 are presented.

4.8.1.1 Accuracy for synthetic datasets

Table 4.2: Accuracy for Non-Linear synthetic datasets (Mean and Standard Deviation for 10 datasets of each type)

Method	Accuracy	Datasets		
		Non-Monotonous	Non-Linear 1	Non-Linear 2
CBSC	Mean	0.788	0.874	0.907
	Std. Dev.	0.030	0.014	0.017
UniBic	Mean	0.764	0.804	0.839
	Std. Dev.	0.004	0.142	0.107
P3C	Mean	0.75	0.765	0.764
	Std. Dev.	0.001	0.004	0.003
Subclu	Mean	0.746	0.785	0.793
	Std. Dev.	0.005	0.017	0.016
ITL	Mean	0.75	0.75	0.75
	Std. Dev.	0.009	0.004	0.002
Proclus	Mean	0.752	0.781	0.792
	Std. Dev.	0.009	0.014	0.007
CLIQUE	Mean	0.75	0.784	0.782
	Std. Dev.	0.000	0.002	0.002

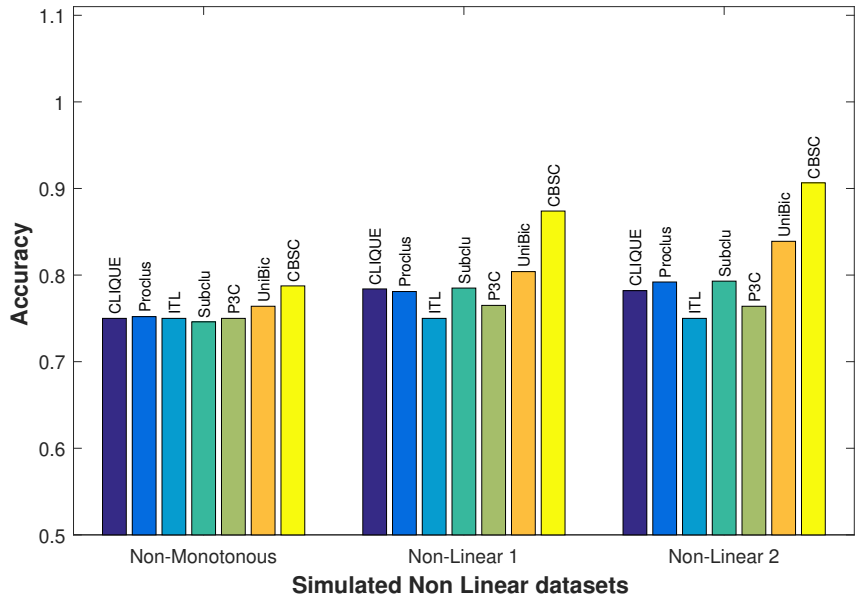


Figure 4.4: Accuracy obtained using various algorithms for Non-Linear datasets

Table 4.3: Accuracy for Base and transformed synthetic datasets (Mean and Standard Deviation for 10 datasets of each type)

Method	Accuracy	Datasets							
		Base	Scaled	Translated	Linear Trans-form	Point Propor-tion	Cluster Propor-tion	Noisy Uniform	Permut-ations
CBSC	Mean	0.935	0.935	0.935	0.936	0.937	0.822	0.936	0.935
	Std. Dev.	0.057	0.057	0.057	0.057	0.060	0.111	0.043	0.057
UniBic	Mean	0.900	0.848	0.802	0.793	0.603	0.641	0.832	0.900
	Std. Dev.	0.034	0.211	0.159	0.034	0.046	0.078	0.124	0.034
P3C	Mean	0.781	0.782	0.779	0.779	0.787	0.802	0.772	0.783
	Std. Dev.	0.040	0.040	0.041	0.041	0.022	0.092	0.032	0.015
Subclu	Mean	0.794	0.759	0.788	0.765	0.802	0.724	0.795	0.798
	Std. Dev.	0.023	0.016	0.019	0.025	0.030	0.020	0.017	0.022
ITL	Mean	0.751	0.752	0.750	0.751	0.750	0.750	0.750	0.750
	Std. Dev.	0.005	0.004	0.004	0.001	0.001	0.002	0.002	0.001
Proclus	Mean	0.785	0.768	0.803	0.770	0.800	0.711	0.792	0.785
	Std. Dev.	0.013	0.015	0.023	0.028	0.024	0.024	0.010	0.015
CLIQUE	Mean	0.776	0.776	0.776	0.776	0.776	0.683	0.720	0.773
	Std. Dev.	0.016	0.016	0.016	0.016	0.016	0.012	0.011	0.015

Table 4.4: Accuracy for synthetic datasets with Normal Distribution (Mean and Standard Deviation for 10 datasets of each type)

Method	Accuracy	Datasets		
		Normal	Noisy Normal 1	Noisy Normal 2
CBSC	Mean	0.895	0.901	0.897
	Std. Dev.	0.062	0.059	0.053
UniBic	Mean	0.869	0.794	0.776
	Std. Dev.	0.025	0.098	0.009
P3C	Mean	0.767	0.758	0.765
	Std. Dev.	0.017	0.047	0.009
Subclu	Mean	0.783	0.78	0.769
	Std. Dev.	0.010	0.015	0.006
ITL	Mean	0.75	0.749	0.750
	Std. Dev.	0.003	0.001	0.001
Proclus	Mean	0.783	0.772	0.769
	Std. Dev.	0.017	0.011	0.006
CLIQUE	Mean	0.799	0.787	0.776
	Std. Dev.	0.011	0.015	0.008

Table 4.5: Accuracy for Overlapping synthetic datasets (Mean and Standard Deviation for 10 datasets of each type)

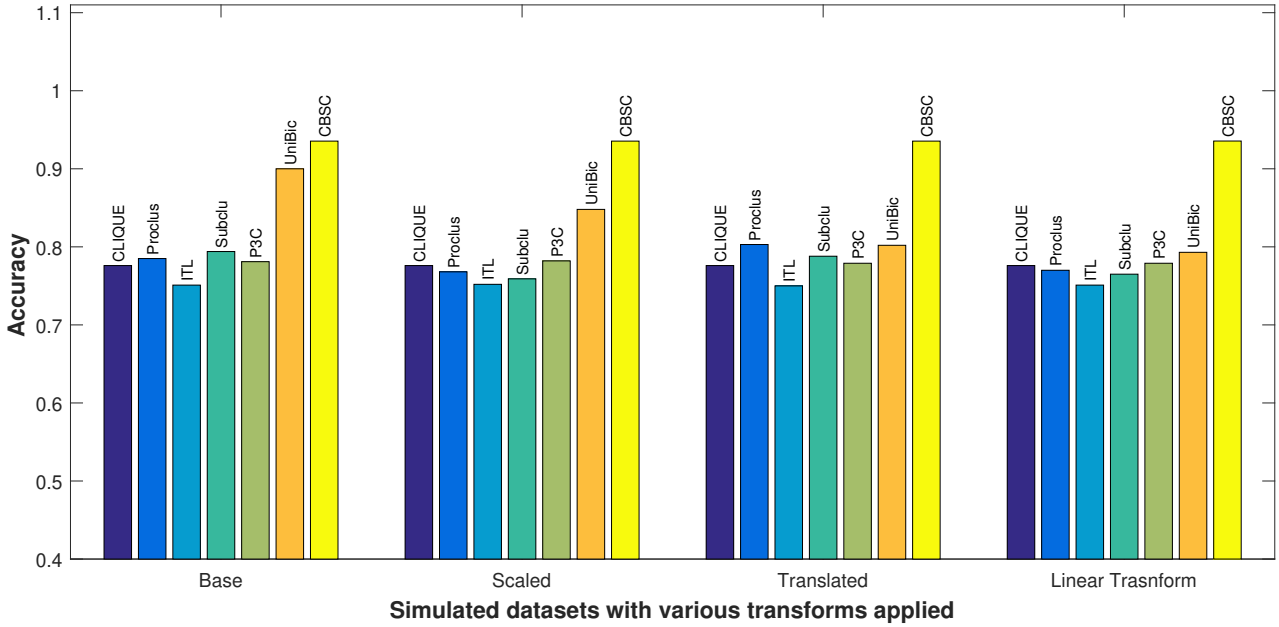
Method	Accuracy	Datasets			
		Overlap 1		Overlap 2	
		First	Second	First	Second
CBSC	Mean	0.953	0.945	0.952	0.949
	Std. Dev.	0.010	0.003	0.011	0.009
UniBic	Mean	0.940	0.626	0.833	0.775
	Std. Dev.	0.037	0.032	0.130	0.143
P3C	Mean	0.751	0.801	0.750	0.800
	Std. Dev.	0.002	0.004	0.001	0.001
Subclu	Mean	0.761	0.801	0.763	0.801
	Std. Dev.	0.005	0.002	0.004	0.002
ITL	Mean	0.750	0.800	0.750	0.800
	Std. Dev.	0.001	0.001	0.001	0.001
Proclus	Mean	0.760	0.804	0.761	0.802
	Std. Dev.	0.009	0.004	0.008	0.003
CLIQUE	Mean	0.750	0.800	0.750	0.800
	Std. Dev.	0.001	0.001	0.001	0.001

For synthetic data, experiments are carried out with 10 different simulations and the average accuracy (over 10 simulations) and corresponding standard deviation (denoted as "deviation") values are reported in Tables 4.2-4.5. Comparison is made with six other methods. We see that CBSC provides better accuracy for the sixteen synthetic datasets.

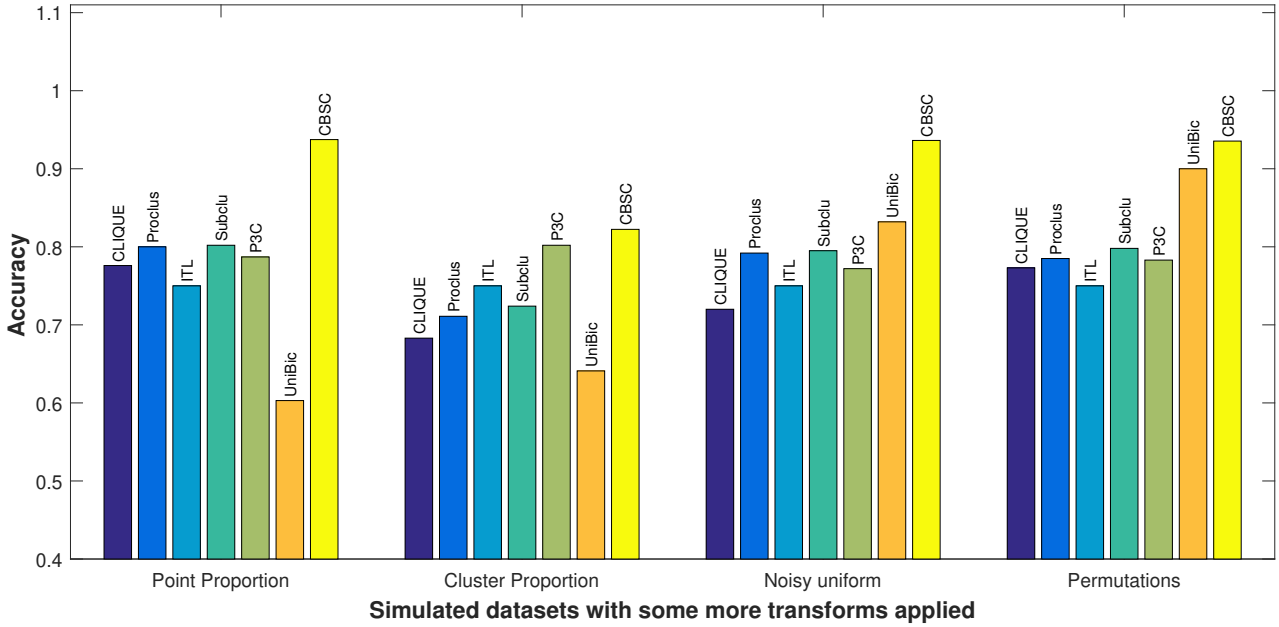
The columns ("Non-Monotonous", "Non-Linear 1" and "Non-Linear 2") of Table 4.2 represent datasets having biclusters based on non-linear relations. We see that CBSC provides higher accuracy for these three datasets. Unlike other algorithms, CBSC finds the biclusters accurately for datasets containing highly non-monotonous relations as seen from the first column ("Non-Monotonous"). The accuracy of CBSC for dataset containing bicluster based on non-linear relations is higher in comparison with other algorithms as seen from columns "Non-Linear 1" and "Non-Linear 2". The values of accuracy obtained using different algorithms on these datasets are also presented in Figure 4.4 as bar graphs, for better illustration. The experiments on these three datasets illustrate that CBSC can find biclusters where features are related to each other with non-linear or non-monotonous relations as mentioned in Section 6.2.

The first column of Table 4.3 ("Base") represents a dataset containing biclusters based on a linear relation. The other seven columns of the table ("Scaled", "Translated", ..., "Permutations") give results for datasets obtained by applying different transforms to the "Base" dataset. These datasets allow us to analyze the behavior of CBSC in a way similar to the analysis done by Ness [24] for clustering algorithms. From Figure 4.5 (shown in parts 4.5a and 4.5b) and the table mentioned above, we find that CBSC has produced better results compared to other methods for the "Base" dataset and its transforms, i.e., "Scaled",..., "Permutations".

The three columns of Table 4.4 ("Normal", "Noisy Normal 1", and "Noisy Normal 2") contain results for datasets generated using the normal distribution. "Noisy



(a) Accuracy obtained for Base dataset and its transforms



(b) Accuracy for 4 more transforms applied to Base dataset

Figure 4.5: Accuracy obtained for synthetic datasets with various transforms applied

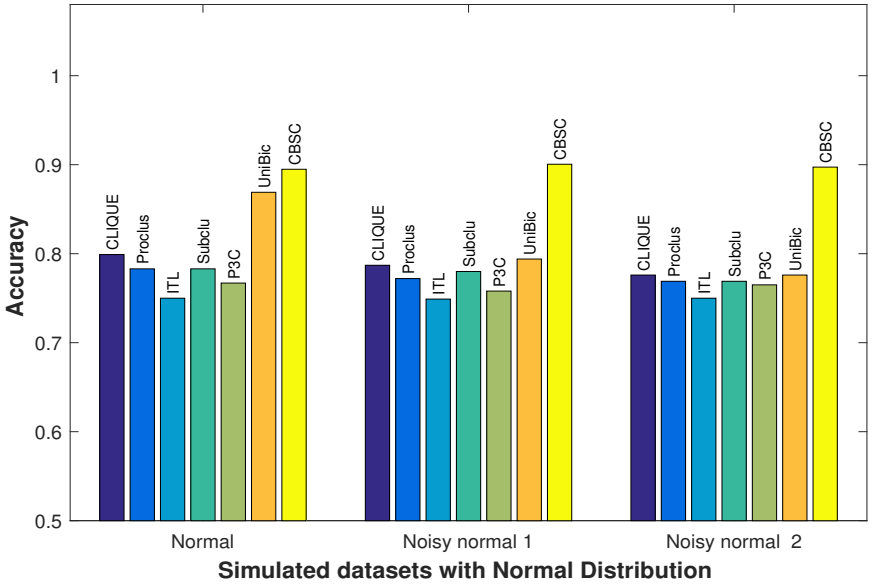


Figure 4.6: Accuracy obtained using various algorithms for Normal and Noisy Normal datasets

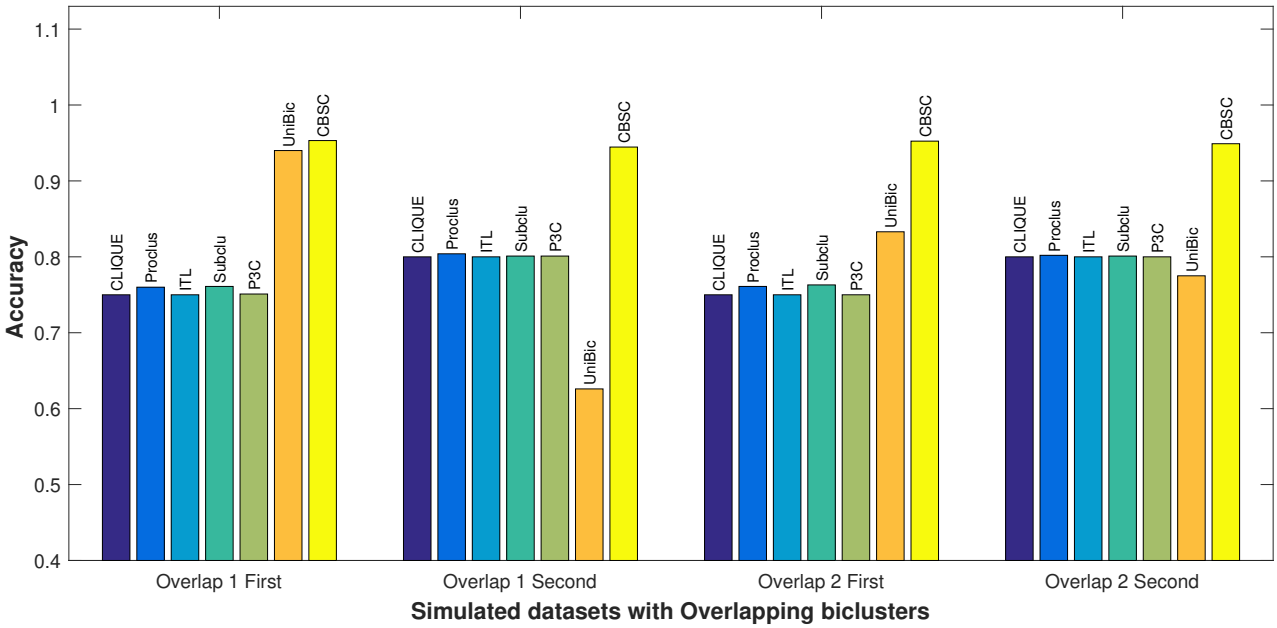


Figure 4.7: Accuracy obtained using various algorithms for Overlapping datasets

Normal 1" and "Noisy Normal 2" datasets have noise with standard deviation 0.1 and 0.2, respectively. CBSC is seen to provide better accuracy than other methods on these datasets. The results for these datasets have also been presented graphically in Figure 4.6.

The first two columns of Table 4.5 ("Overlap 1"), present results for a dataset containing two biclusters with an overlap of 12%. The third and fourth columns present results for a similar dataset with two biclusters but with an overlap of 20%. We find that CBSC has the best performance among the methods used for comparison. These results are presented graphically in Figure 4.7. These results for synthetic datasets suggest some properties of CBSC that will be discussed in Section 4.8.1.3.

4.8.1.2 Execution time for synthetic datasets and time complexity

Table 4.6: Execution time in seconds for Non-Linear synthetic datasets

Method	Datasets		
	Non-Monotonous	Non-Linear 1	Non-Linear2
CBSC	61.674	93.630	69.359
UniBIc	0.012	0.583	0.612
P3C	1.440	0.359	0.650
Subclu	0.037	0.064	0.052
ITL	15.000	9.096	9.248
Proclus	0.026	0.060	0.055
CLIQUE	0.001	0.011	0.007

Table 4.7: Execution time in seconds for Base and transformed synthetic datasets

Method	Datasets							
	Base	Scale	Translated	Linear	Point Proportion	Cluster Proportion	Noisy	Permutations
CBSC	81.585	82.717	81.450	81.840	97.645	94.466	96.439	81.654
UniBIc	0.583	0.612	0.649	0.673	2.596	1.436	0.621	0.583
P3C	0.513	0.955	0.952	0.433	1.083	0.848	0.458	0.588
Subclu	0.056	0.043	0.050	0.055	0.120	0.082	0.049	0.070
ITL	3.171	3.302	3.498	3.276	9.994	6.519	2.94	3.272
Proclus	0.052	0.055	0.049	0.055	0.120	0.088	0.056	0.063
CLIQUE	0.105	0.015	0.015	0.018	0.033	0.021	0.014	0.013

The execution time for CBSC and other methods is reported in Tables 4.6-4.9. The datasets used here are the same as those used in Tables 4.2-4.5. The algorithm

Table 4.8: Execution time in seconds for synthetic datasets with Normal Distribution

Method	Datasets		
	Normal	Noisy Normal 1	Noisy Normal 2
CBSC	67.290	88.490	73.850
UniBic	0.531	0.577	0.001
P3C	18.840	9.751	6.322
Subclu	0.071	0.061	0.050
ITL	1.187	1.259	20.639
Proclus	0.054	0.052	0.084
CLIQUE	0.071	0.044	0.715

Table 4.9: Execution time in seconds for Overlapping synthetic datasets

Method	Datasets	
	Overlap 1	Overlap 2
CBSC	72.647	68.848
UniBic	2.092	2.775
P3C	0.049	0.016
Subclu	0.074	0.158
ITL	15.102	15.091
Proclus	0.074	0.087
CLIQUE	0.014	0.032

is executed with an Intel Core i7 CPU and 8 GB memory. The execution time of CBSC is seen to be longer than that of other methods. However, it is already seen in Tables 4.2-4.5, that the accuracy of CBSC is better than these methods for most of the synthetic datasets.

4.8.1.3 A discussion on properties of CBSC as seen from experiments on synthetic datasets

CBSC has been shown to identify relation based biclusters with better accuracy relative to other algorithms when applied to synthetic datasets, as indicated in Tables 4.2-4.5 and Figures 4.4-4.7. This section discusses some of the observed characteristics of CBSC.

CBSC is invariant to scaling, translation, and linear transforms, as these transformations do not alter the bin length or the corresponding estimated densities. However, theoretically, the procedure lacks invariance to arbitrary order-preserving transfor-

mations, as these may alter the observation densities. An analysis of the accuracy achieved on the "Base" dataset and its various transformations reveals that there are no significant differences in CBSC performance across the datasets "Scaled", "Translated", "Linear Transform", "Point Proportion" and "Permutations". For "Cluster Proportion" dataset, the accuracy of CBSC has decreased in comparison with the "Base" dataset. This happens because repetition of observations within a cluster causes a change in the density. However, the accuracy of CBSC is still more than that of the other method used for comparison. The accuracy achieved by CBSC on the "Noisy Uniform", "Noisy Normal 1", and "Noisy Normal 2" datasets is slightly reduced compared to the "Base" and "Normal" datasets. However, CBSC exhibits superior accuracy in the presence of noise when compared to other algorithms.

We may note that the biclusters in the "Base" dataset can also be seen as scaling biclusters, since each column in the bicluster can be obtained by multiplying another column by a constant value. In addition, the biclusters in the "Translated" datasets are obtained by adding a random selected constant value to each column of the "Base" dataset. Hence, the resulting biclusters can be seen as a combination of scaling and shift biclusters. This happens because scaling and shifting are examples of linear and affine transformations, respectively. Since CBSC finds biclusters based on feature relations, it will be able to find biclusters based on scaling and shifting of columns. However, CBSC will not be able to find biclusters based on the shifting and scaling of rows, as it specifically aims to find biclusters based on feature relations.

Overall, we find that CBSC performs well on the data after applying different transforms or adding noise. Its performance on datasets containing non-linear datasets is also good.

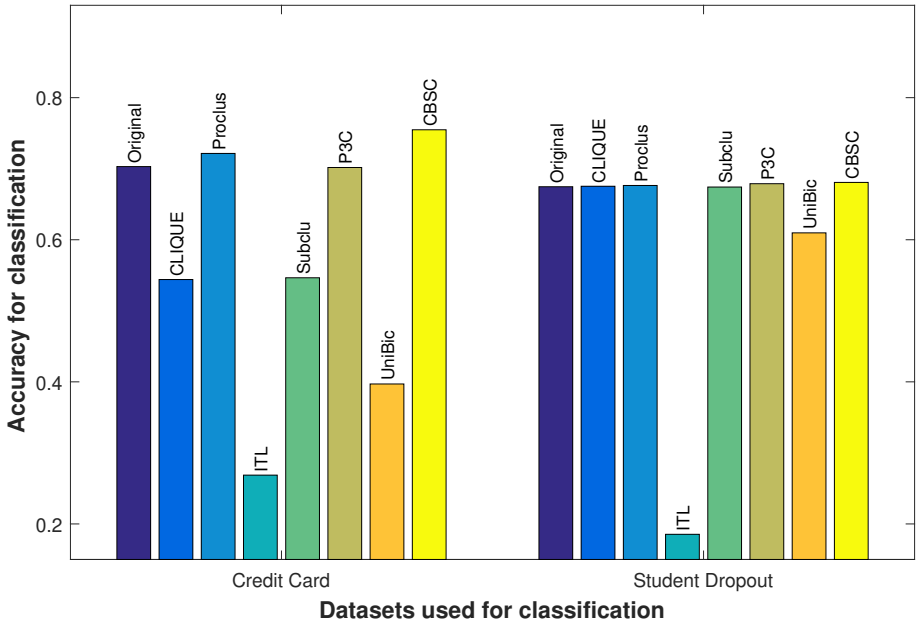
4.8.2 Results for experiments with UCI-ML datasets

4.8.2.1 Experiments with classification of UCI-ML datasets

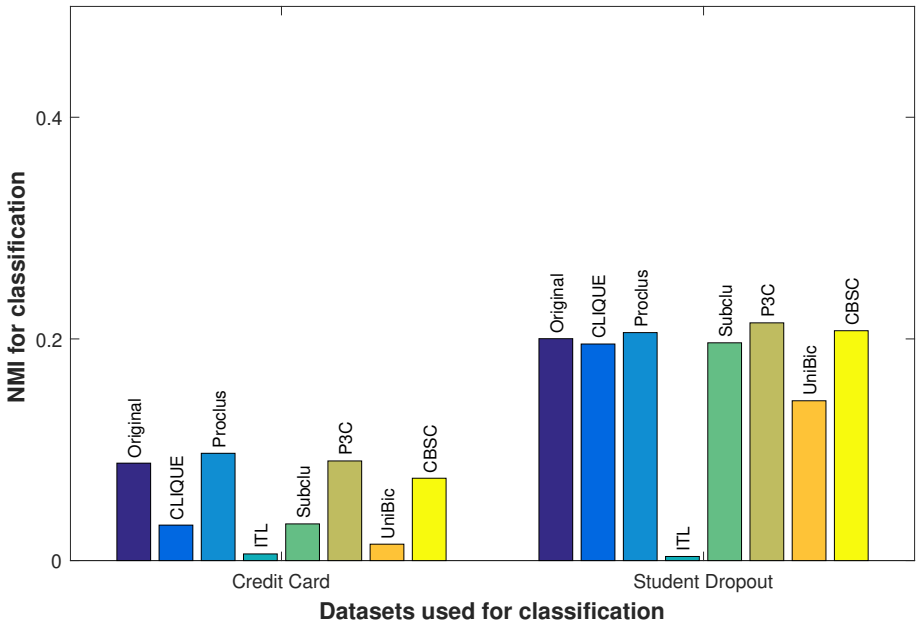
Table 4.10: Classification performance for Credit card and Student Dropout datasets (after adding features generated using different biclustering methods)

Dataset	**Credit Card [175]**						
Method	Accuracy		NMI	ARI	Average		
	Mean	S. Dev.			Prec.	Recall	G-Score
CBSC	0.755	0.006	0.074	0.185	0.647	0.651	0.649
UniBic	0.397	0.006	0.015	-0.047	0.559	0.562	0.472
P3C	0.702	0.012	0.090	0.149	0.642	0.690	0.654
Subclu	0.547	0.098	0.033	0.004	0.581	0.616	0.560
ITL	0.269	0.010	0.006	-0.035	0.562	0.520	0.359
Prolus	0.722	0.015	0.097	0.172	0.650	0.696	0.665
CLIQUE	0.544	0.006	0.032	0.003	0.580	0.614	0.558
Original	0.703	0.028	0.088	0.147	0.640	0.689	0.654
Dataset	**Student Dropout [176]**						
Method	Accuracy		NMI	ARI	Average		
	Mean	S. Dev.			Prec.	Recall	G-Score
CBSC	0.681	0.016	0.207	0.302	0.604	0.583	0.589
UniBic	0.610	0.009	0.144	0.125	0.550	0.450	0.455
P3C	0.679	0.020	0.215	0.312	0.595	0.581	0.585
Subclu	0.674	0.018	0.197	0.291	0.602	0.586	0.591
ITL	0.185	0.003	0.004	-0.002	0.417	0.336	0.184
Proclus	0.676	0.018	0.206	0.305	0.601	0.589	0.593
CLIQUE	0.675	0.022	0.195	0.295	0.597	0.576	0.582
Original	0.675	0.015	0.200	0.295	0.593	0.576	0.580

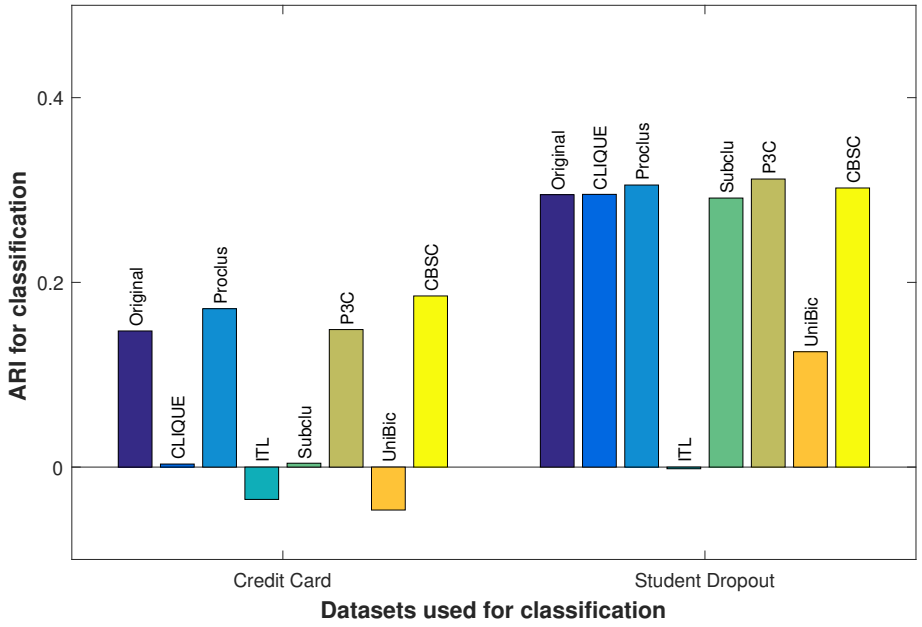
The classification results are reported in Table 4.10 and the performance is also presented as bar graphs in Figures 4.8a-4.8f. We find that for the Student Dropout dataset, CBSC and P3C provide an improvement over the original accuracy. For the Credit Card dataset, the features generated by CBSC provide a greater improvement in accuracy compared to six other algorithms, thereby leading to better identification of credit card defaulters.



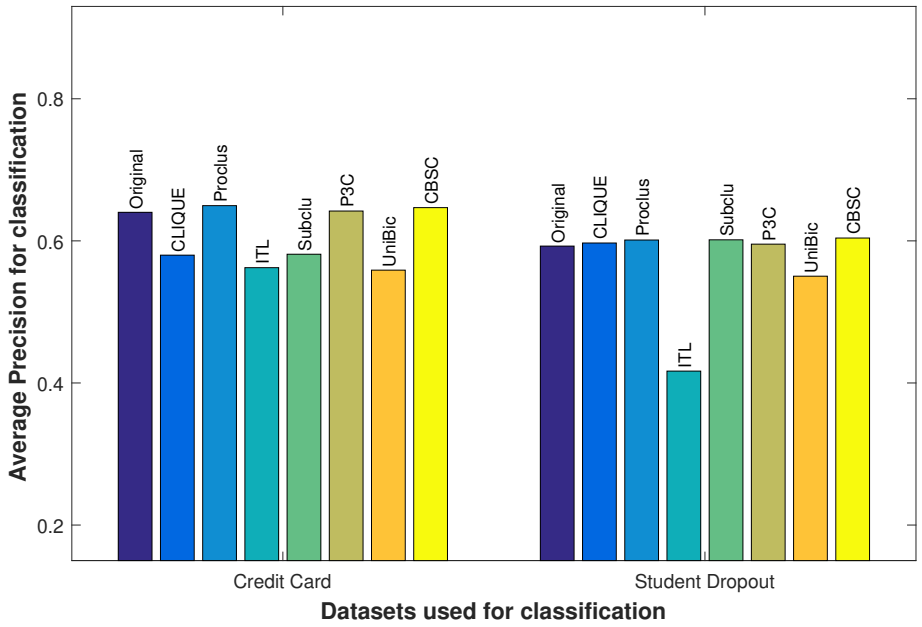
(a) Accuracy of various algorithms for classification



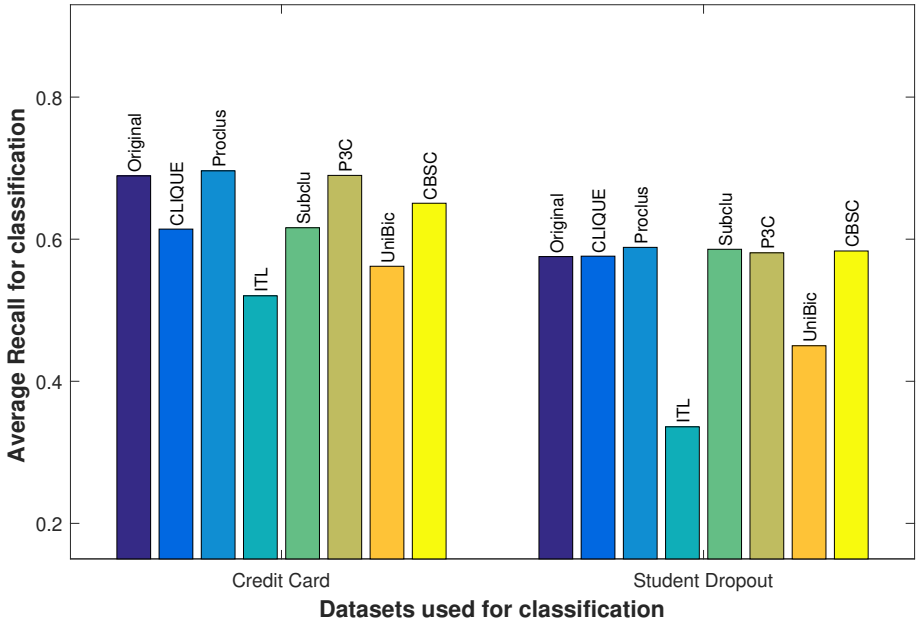
(b) NMI obtained by various algorithms for classification



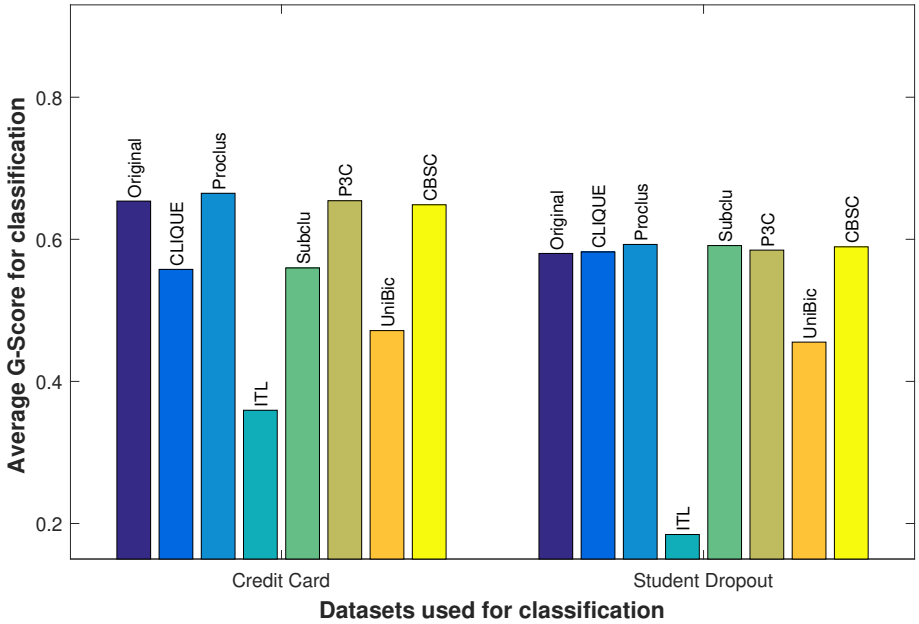
(c) ARI obtained by various methods for classification



(d) Average precision obtained by various algorithms for classification



(e) Average recall obtained by various algorithms for classification

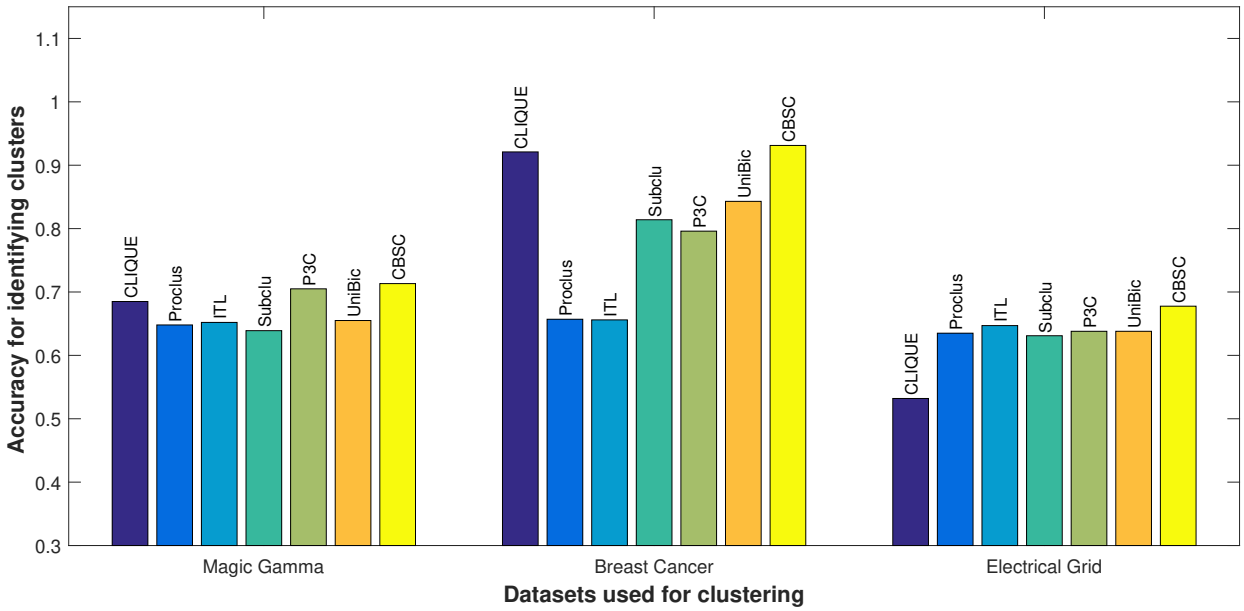


(f) Average G-Score obtained by various algorithms for classification

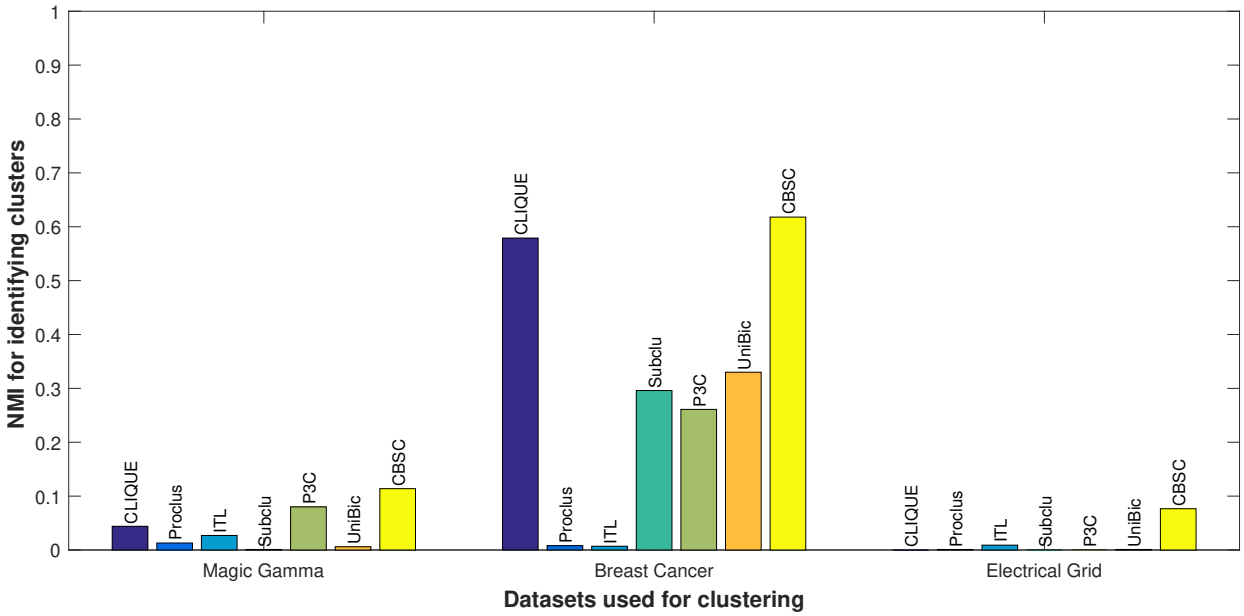
Figure 4.8: Performance of various algorithms for classification

Table 4.11: Results for clustering

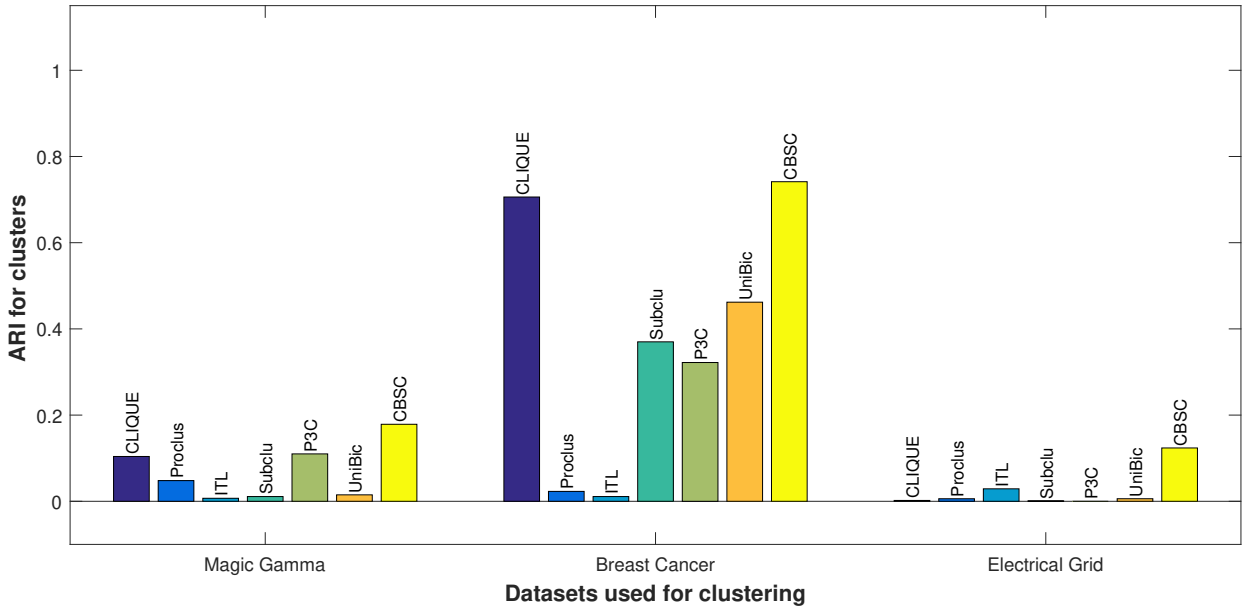
Dataset	**Magic Gamma [177]**								
Method	Accuracy	NMI	ARI	Class 1			Class 2		
				Prec.	Recall	G-Score	Prec.	Recall	G-Score
CBSC	0.713	0.114	0.179	0.582	0.655	0.617	0.799	0.745	0.772
UniBic	0.655	0.006	0.015	0.654	0.991	0.805	0.677	0.035	0.153
P3C	0.705	0.080	0.110	0.690	0.992	0.827	0.919	0.177	0.404
Subclu	0.639	0.001	0.011	0.653	0.943	0.785	0.425	0.077	0.181
ITL	0.652	0.027	0.007	0.953	0.012	0.107	0.651	1.000	0.807
Proclus	0.648	0.013	0.048	0.676	0.878	0.770	0.498	0.223	0.333
CLIQUE	0.685	0.044	0.104	0.703	0.889	0.791	0.600	0.308	0.430
Dataset	**Breast Cancer (Wisconsin Original)** [178]								
Method	Accuracy	NMI	ARI	Class 1			Class 2		
				Prec.	Recall	G-Score	Prec.	Recall	G-Score
CBSC	0.931	0.618	0.742	0.893	0.912	0.903	0.952	0.941	0.947
UniBic	0.843	0.330	0.462	0.833	0.690	0.758	0.847	0.926	0.886
P3C	0.796	0.261	0.322	0.981	0.427	0.647	0.763	0.995	0.872
Subclu	0.814	0.296	0.370	0.975	0.481	0.685	0.781	0.993	0.880
ITL	0.656	0.007	0.011	0.833	0.021	0.132	0.654	0.998	0.808
Proclus	0.657	0.008	0.023	0.600	0.063	0.194	0.660	0.977	0.803
CLIQUE	0.921	0.579	0.706	0.881	0.895	0.888	0.943	0.935	0.939
Dataset	**Electrical Grid**								
Method	Accuracy	NMI	ARI	Class 1			Class 2		
				Prec.	Recall	G-Score	Prec.	Recall	G-Score
CBSC	0.678	0.077	0.124	0.549	0.616	0.582	0.766	0.712	0.739
UniBic	0.638	0.001	0.006	0.497	0.026	0.114	0.641	0.985	0.794
P3C	0.638	0.000	0.000	0.638	1.000	0.799	0.000	0.000	0.000
Subclu	0.631	0.000	0.001	0.383	0.030	0.107	0.639	0.973	0.788
ITL	0.647	0.009	0.029	0.572	0.096	0.235	0.652	0.959	0.791
Proclus	0.635	0.001	0.006	0.444	0.038	0.129	0.641	0.973	0.790
CLIQUE	0.532	0.000	0.002	0.372	0.425	0.398	0.645	0.593	0.619



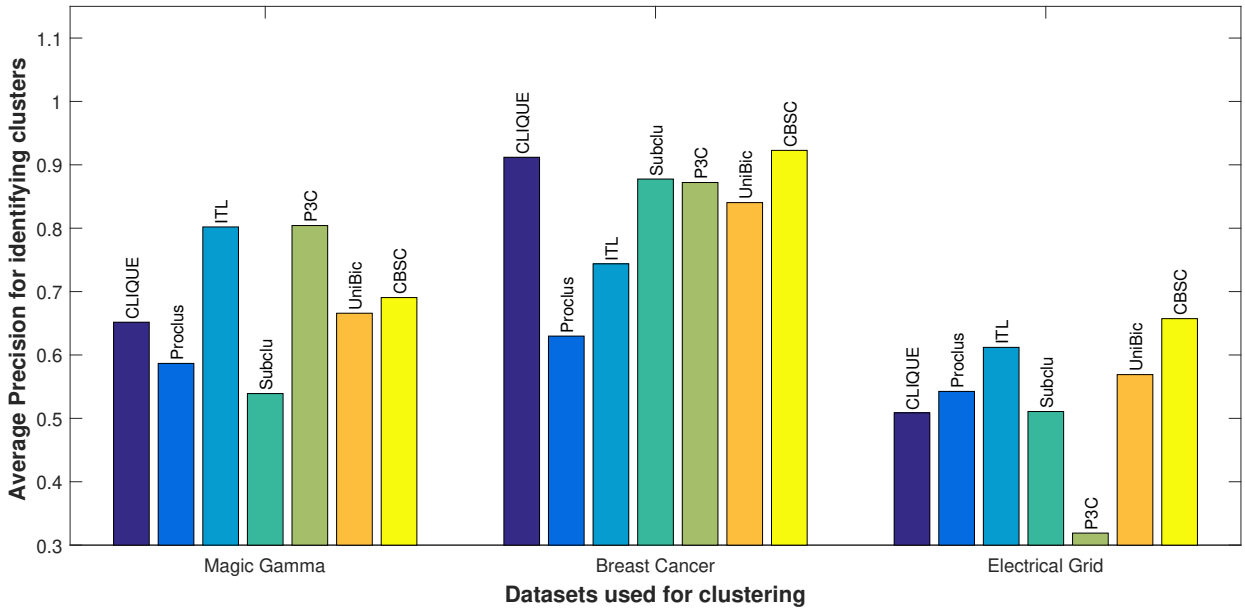
(a) Accuracy of various algorithms for clustering of UCI-ML datasets (calculated using equation 4.1)



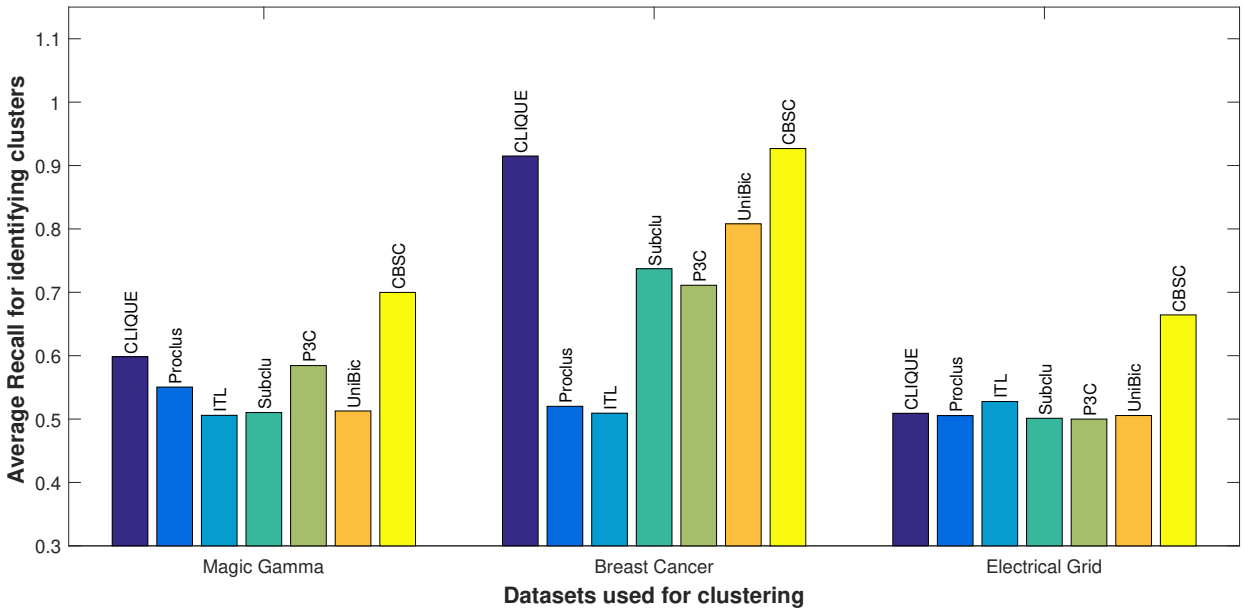
(b) NMI obtained by various algorithms for clustering of UCI-ML datasets



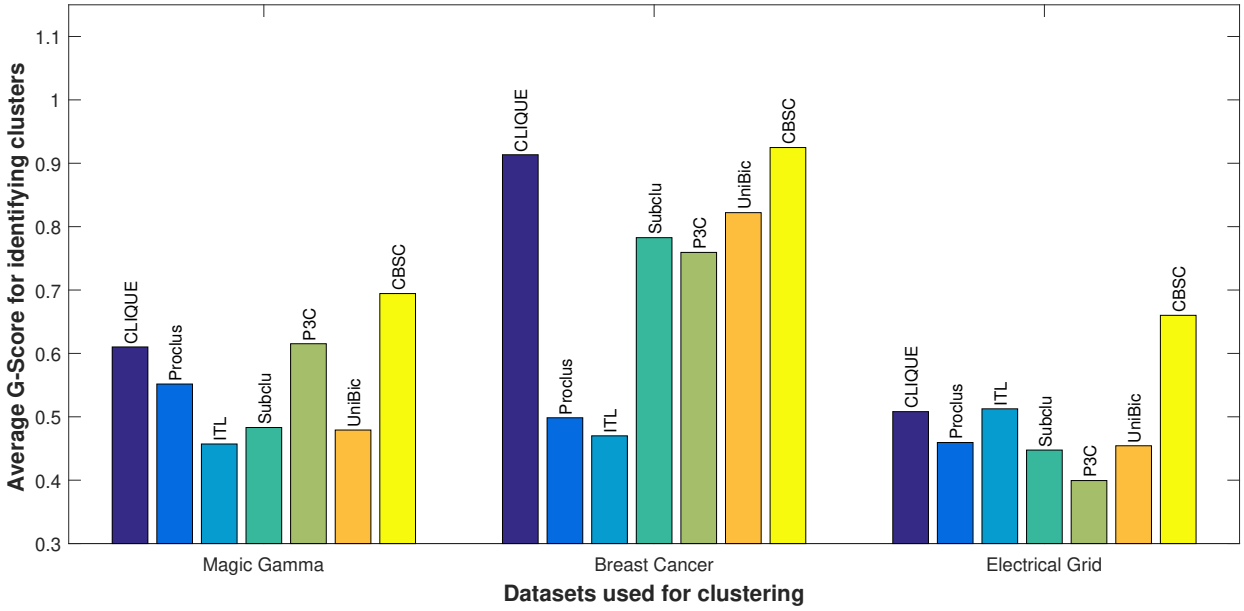
(c) ARI obtained by various algorithms for clustering of UCI-ML datasets



(d) Average precision obtained by various algorithms for clustering of UCI-ML datasets



(e) Average recall obtained by various algorithms for clustering of UCI-ML datasets



(f) Average G-Score obtained by various algorithms for clustering of UCI-ML datasets

Figure 4.9: Performance of various algorithms for clustering of UCI-ML datasets

4.8.2.2 Experimental results for clustering of UCI-ML datasets

The results for clustering UCI-ML datasets are reported in Table 4.11. CBSC is seen to have the best accuracy for the datasets used for the investigation. For better visualization, the accuracies obtained by different methods are presented in Figure 4.9a as bar graphs. NMI and ARI are shown in Figures 4.9b and 4.9c, respectively. The average Precision, Recall and G-score of both classes have been shown in Figures 4.9d-4.9f, respectively. We find that CBSC obtains better accuracy, NMI, ARI, Precision, Recall, and G-Score for three datasets, namely Magic Gamma, Breast cancer, and Electric Grid.

4.8.3 Relevant features in COVID-19 dataset obtained using CBSC during early stages of the pandemic

Table 4.12 enumerates the 27 features used for analysis [193]. The features identified by UniBic and CBSC based on data from January 2020 are denoted by a tick mark in the corresponding columns. In addition, the table presents the Spearman correlation values (ρ_s) between the selected features and the COVID-19 infection rate in various countries on 31 January 2020 (denoted as, ρ_s(Jan)) and 31 December 2020 (denoted as, ρ_s(Dec)). The features are organized in descending order according to the absolute value of ρ_s(Dec).

CBSC identified 11 out of 27 WDI features effective in distinguishing regions with elevated infection rates. It is observed that 9 of the 11 features selected by CBSC exhibit a Spearman correlation (absolute value) exceeding 0.5 with the disease occurrence rate as of 31 December 2020. This finding suggests that CBSC could ascertain the significance of these features considerably earlier, specifically from the January data, despite the Spearman correlation not yet demonstrating this relationship. Con-

Table 4.12: Features (WDI dataset) selected by biclusters based on countries having high COVID-19 occurrence (90 percentile) in January 2020

S. No.	Feature description	ρ_s(Jan)	ρ_s(Dec)	UniBic	CBSC
1	Current health expenditure per capita, PPP (current international $)	0.22	0.69	✓	-
2	Cause of death, by communicable diseases and maternal, prenatal and nutrition conditions (% of total)	-0.09	-0.68	✓	✓
3	Cause of death, by non-communicable diseases (% of total)	0.12	0.67	✓	✓
4	Mortality rate, adult, female (per 1,000 female adults)	-0.27	-0.67	-	✓
5	Survival to age 65, female (% of cohort)	0.27	0.67	✓	✓
6	People using at least basic sanitation services (% of population)	0.23	0.64	✓	✓
7	Life expectancy at birth, total (years)	0.26	0.64	✓	-
8	GDP per capita, PPP (current international $)	0.22	0.62	✓	✓
9	Tuberculosis case detection rate (%, all forms)	0.13	0.61	-	-
10	Population ages 65 and above (% of total)	0.25	0.59	✓	-
11	Survival to age 65, male (% of cohort)	0.24	0.58	✓	✓
12	Urban population (% of total)	0.12	0.58	-	-
13	Mortality rate, adult, male (per 1,000 male adults)	-0.23	-0.57	-	✓
14	Incidence of tuberculosis (per 100,000 people)	-0.05	-0.51	✓	✓
15	Population ages 15-64 (% of total)	0.07	0.46	✓	-
16	International tourism, number of arrivals	0.4	0.44	✓	-
17	Mortality from CVD, cancer, diabetes or CRD between exact ages 30 and 70 (%)	-0.23	-0.4	-	✓
18	PM2.5 air pollution, population exposed to levels exceeding WHO guideline value (% of total)	-0.23	-0.38	-	-
19	Air transport, passengers carried	0.44	0.37	-	-
20	International migrant stock, total	0.35	0.27	-	-
21	Trade (% of GDP)	0.03	0.26	-	-
22	Out-of-pocket expenditure (% of current health expenditure)	-0.09	-0.25	-	✓
23	Labor force participation rate, total (% of total population ages 15+) (modeled ILO estimate)	0.04	-0.24	-	-
24	Population density (people per sq. km of land area)	0.23	0.16	✓	-
25	Population, total	0.34	-0.16	✓	-
26	Tuberculosis treatment success rate (% of new cases)	-0.08	-0.15	✓	-
27	Death rate, crude (per 1,000 people)	0.02	0.07	✓	-

versely, among the features omitted from the bicluster, only 5 out of 16 exhibit a Spearman correlation (absolute value) greater than 0.5 with disease occurrence rates on 31 December 2020.

The aforementioned findings indicate that CBSC holds potential for identifying critical factors influencing emerging diseases. This approach could consequently facilitate the development of more scientifically grounded strategies for managing pandemic scenarios in various countries. Notably, certain features selected for COVID-19 have been analyzed previously and deemed significant, as elaborated in the subsequent paragraphs, where references to features are made using their serial numbers (S. No.) as listed in Table 4.12.

Among the features selected by CBSC, the percentage of deaths in the region due to communicable and non-communicable diseases (S. Nos. 2 and 3 of Table 4.12) is in line with the report given by Chakrabarti et al. [200]. The features showing the life expectancy and mortality rate for the overall population and for each gender affect the age distribution of people in a region (S. Nos. 4,5,11,13,17 of Table 4.12) are selected by CBSC. The feature "People using at least basic sanitation services" (S. Nos. 6 of Table 4.12) is related to the spread of communicable diseases and the corresponding immunity. The feature GDP per capita (S. No. 8) is also known to be associated with basic sanitation services and life expectancy, thus having an association with COVID-19 infection rates. The feature "Incidence of tuberculosis" (S. No. 14 of Table 4.12), is interesting. Countries with higher tuberculosis incidences are also seen to have relatively lower rates of COVID-19 infection. A study [201] has already noted that the administration of the BCG vaccine is likely to reduce the severity of the symptoms of COVID-19. Another factor that may contribute to this association is that tuberculosis is more rampant in warm and humid climates, while COVID-19 is expected to spread faster in cold and dry climates [202].

As mentioned earlier, all features selected by CBSC, except two (S.Nos. 17 and 22 of Table 4.12) have a high Spearman correlation with COVID infection rates on 31 December 2020, although the biclusters are obtained using the least amount of data from 31 January 2020.

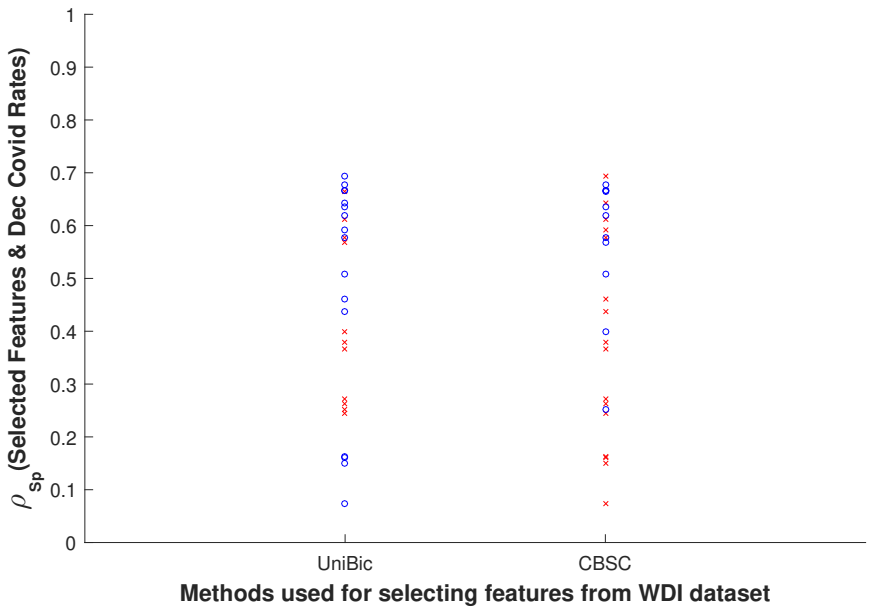


Figure 4.10: Spearman correlation (absolute value) between Dec 31 2020 COVID infection rates and features selected from WDI database by biclustering methods. Selected features are marked in blue and rejected in red

The performance comparison between CBSC and UniBic is shown in Figure 4.10, which shows a scatter plot for the Spearman correlation (absolute values). For each method, selected features are marked in blue and rejected features in red. For CBSC, the selected features are mainly those that have a higher correlation, while UniBic also selects features that have a low Spearman correlation.

In summary, most of the features selected by CBSC are relevant, and the presented methodology allows us to explore some previously unknown associations. Once features are identified from larger datasets, a more detailed analysis could be performed for individual features.

4.8.4 Results of analysis on COVID-19 and Vaccine dataset

We have applied CBSC to find the vaccine that could provide protection against COVID-19 infection using the methodology given in Section 4.7.4. The feature selected by CBSC is the percentage of newborns who have received the Tetanus-Toxoid vaccine. A study [203] has already noted that a dose of the Tetanus-Toxoid vaccine may result in a reduction in the severity of COVID-19 symptoms. Their study is based on similarities between the COVID-19 spike protein and Tetanus Toxin. Our experiments reinforce this hypothesis. We have also repeated the above-mentioned experiments with UniBic. The analysis using UniBic also indicates that the coverage of the Tetanus Toxoid vaccine may be related to the infection rate of COVID-19.

4.9 Limitations of CBSC

In our experiments, CBSC is seen to achieve better performance on the synthetic and real-world datasets used in this experiment. However, it also has a longer execution time. In the future, parallel computing techniques can be used to reduce the runtime of this method. As discussed earlier, the detection of high-density regions for each pair of features is independent. Parallel processing is thus expected to significantly reduce the runtime of the method.

4.10 Summary

This chapter presents CBSC, a novel algorithm for subspace clustering that leverages connectivity between observations in the two-dimensional space to identify dense regions. These regions are used for the discovery of subspace clusters in higher-dimensional spaces. By capturing both direct and indirect relationships among fea-

tures, CBSC is capable of detecting non-linear relationship-based biclusters. CBSC has been applied to several synthetic datasets with varying characteristics to identify embedded biclusters. CBSC has also been applied for the identification of clusters in three UCI-ML datasets. Experimental results on both synthetic and real-world datasets demonstrate that CBSC identifies meaningful biclusters and outperforms existing biclustering techniques in terms of accuracy. Furthermore, CBSC has been utilized to enhance two UCI-ML datasets, leading to improved classification performance. Its successful application in uncovering lifestyle factors associated with COVID-19 infection rates further highlights its practical utility. CBSC was also applied to a vaccine coverage dataset along with COVID-19 infection rates revealing a potential protective association between the Tetanus Toxoid vaccine and COVID-19. Overall, CBSC is an effective tool for uncovering hidden patterns in data.

Chapter 5

RelDenClu: Relative Density based Biclustering

5.1 Introduction

In this chapter, we present a non-linear relation based biclustering method named RelDenClu. It uses joint and marginal densities for pairs of variables to discover non-linear relation based biclusters. The main objective of RelDenClu is to find biclusters formed by features related to each other with a subset of observations as a base. In the previous chapter, we presented a method named CBSC [30] with a similar objective. However, CBSC tends to fragment biclusters when the marginal densities are highly variable.

RelDenClu analyzes two-dimensional spaces to identify regions where the joint density exceeds the marginal densities. Subsequently, it identifies pairs of features for which the same sets of observations constitute dense regions, and determines the sets of interconnected features within these regions that form biclusters. Consequently, RelDenClu distinguishes itself from CBSC by the approach it uses to find the dense

regions. By evaluating both joint and marginal distributions, RelDenClu effectively mitigates the issue of fragmented biclusters. After identifying the dense regions, its method of finding features connected through identical sets of observations closely resembles that of CBSC.

RelDenClu does not require additional parameters to identify dense regions. However, it requires input parameters specifying the minimum size of dense regions and the amount of overlap between the dense regions. RelDenClu can find relations based on both monotonous and non-monotonous relations. The following section presents the RelDenClu method in detail.

5.2 Non-linear relation based Biclustering Method: RelDenClu

The problem addressed by RelDenClu is the same as that addressed by CBSC. Thus the problem statement is same as that stated in Section 4.2.2 in Chapter 4. RelDenClu differs from CBSC in its method of finding the dense regions for each pair of dimensions. Unlike CBSC, RelDenClu uses both the joint and marginal densities of the variables to identify the dense regions. In this section, we present a procedure for finding dense sets for each pair of features, which is the main contribution of the present chapter.

5.2.1 Technique used to find the set of observations showing dependence for a feature pair

This section describes a novel method for identifying subsets of observations which show dependence between two features, by comparing joint distribution and marginal

distributions using the histogram technique. The data in the Euclidean space are normalized to space $[0,1] \times [0,1]$. A grid or rolling window is used to calculate the density. Data is analyzed by considering small rectangular regions called cells. The terminology used in this section is as follows: The region given by $(v_x - xlen/2, v_x + xlen/2] \times (v_y - ylen/2, v_y + ylen/2]$ is said to be a cell of size $(xlen, ylen)$ centered at the point(v_x, v_y). The phrase "corresponding marginal cells" refers to rectangular regions: $(v_x - xlen/2, v_x + xlen/2] \times [0,1]$ and $[0,1] \times (v_y - ylen/2, v_y + ylen/2]$. Here we use the term 'density' for 'histogram density estimate' i.e., density is given by the number of observations in a cell divided by product of cell area and total number of observations. A cell is said to be dense if its density is higher than the average density of the corresponding marginal cells.

As examining cells centered at each observation may not be computationally feasible for large datasets, we use slightly different schemes for small and large datasets. This allows us to achieve a balance between accuracy and ease of computation. Whether a dataset is considered large or small depends on the availability of computational power. Both the methods for small and large datasets are given below.

5.2.1.1 Finding related dense regions for small datasets

For small datasets, the cell centered around each observation is examined. Note that we will have many overlapping cells. For each observation with coordinates (v_x, v_y), we find observations lying in the cell centered on it. More precisely, we check the cell given by $(v_x - xlen/2, v_x + xlen/2] \times (v_y - ylen/2, v_y + ylen/2]$ to see if its density is higher than the average density of both marginal cells, which are the cells given by $(v_x - xlen/2, v_x + xlen/2] \times (0,1]$ and $(0,1] \times (v_y - ylen/2, v_y + ylen/2]$. If this condition is found to be true and the density of the cell is also higher than the average density of the entire region, we mark the cell centered at (v_x, v_y) as 'dense'. Two dense cells

are merged only if the center of each cell lies within the other cell and the average density of the overlapping region between two cells is high. The density is considered high if it is not less than the density of each of the marginal cells corresponding to the two overlapping cells and is also not less than the average density of the entire space. Repeating these steps iteratively allows us to find larger connected dense regions. The algorithm 5.1 provides pseudo-code for finding dense regions as described here.

The size of a cell along a dimension X is calculated using maximal spacing along X. The definition of maximal spacing was given in Chapter 3 in Section 3.2.2.1. For each pair of dimensions X and Y let the maximal spacings be s_x and s_y, respectively. The intervals used to find dense regions along X and Y are given by $xlen = {s_x}^c$ and $ylen = {s_y}^c$, where $0 < c < 0.5$ and is close to 0.5. We have taken $c = 0.4999$. The reason for choosing this value has been discussed in Section 5.2.1.3.

As the maximal spacing does not go to zero for datasets having non-continuous distribution, this method cannot be used for such datasets. Therefore, for such datasets and for large datasets, we present a simpler method which does not use maximal spacing. This is discussed in the following section.

5.2.1.2 Finding related dense regions for large datasets

For larger datasets the method previously described becomes infeasible due to the computational burden of determining the number of observations in the neighborhood of each individual observation. To mitigate this limitation, we introduce a simplified methodology for identifying dense regions within extensive datasets. Specifically, for sizable datasets, the space is segmented into $3\log(N)$ equal intervals along each axis, where N represents the number of observations in the dataset. The examination is confined to disjoint cells, thereby optimizing the execution time. A cell is classified as dense if its density surpasses both the densities of marginal cells and the average

density across the entire space. Dense regions are merged if they are contiguous either horizontally, vertically, or diagonally. A singular dense region can be connected to a maximum of eight adjacent dense regions. Subsequent to the identification of dense regions in two-dimensional space, noise elimination is achieved by evaluating the overlap among three two-dimensional spaces for every trio of features. Consequently, redundancy introduced by using three features facilitates the elimination of noise. A detailed description of this step can be found in Section 5.2.2.

Figure 5.1 shows how the dense cell (marked in red) has a higher density than the two marginal cells (marked in green). The adjacent dense cells (marked pink) will be combined together to form dense regions in 2-dimensional space.

5.2.1.3 Choosing parameters for density estimates and convergence

The performance of histogram-based density methods depends on the appropriate bin length (size of partitions used for density estimation). If the bin length reduces too quickly, the estimated density will be incorrect as some of the bins (partitions) may remain empty. However, if the bin length decreases too slowly, it will not converge to the true density as it will be averaging over a large area. [132] has given the criterion for convergence of density estimates as follows: The area of each bin A_N should converge to zero as the number of observations N goes to infinity. In addition, the product of the number of observations N and the area of each cell should tend to infinity as N approaches infinity. We can write the condition as $\lim_{N\to\infty} A_N \to 0$ and the product $\lim_{N\to\infty}(N \times A_N) \to \infty$.

In this chapter, we use a bin length based on maximal spacing. As discussed in Section 5.2.1.1, for small datasets, we use bin lengths $x_{len} = s_x^c$ and $y_{len} = s_y^c$ where s_x and s_y are maximal spacings along the X and Y axes, respectively, and c is the constant which should have a value close to 0.5, but also less than 0.5. We will now

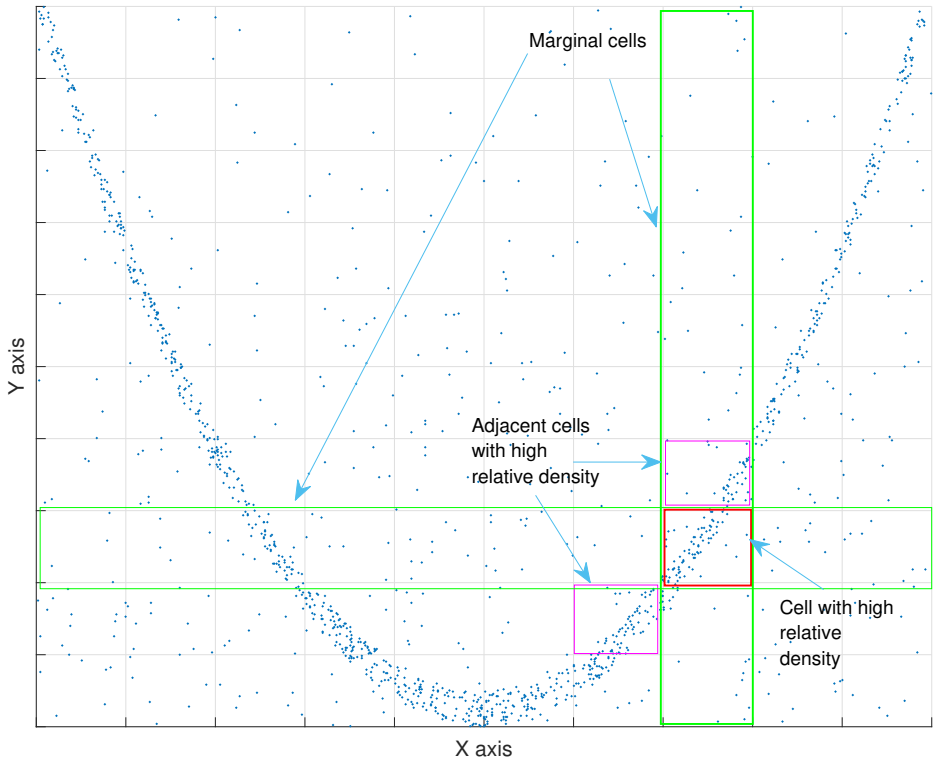


Figure 5.1: Choosing dense cells with density higher than marginal cells and merging them with adjacent dense cells

discuss how this choice leads to consistent estimates.

Suppose N observations are distributed in the interval $[0, 1]$ along a particular axis and the maximal spacing along this axis is given by s. The work done by Deheuvels [166], discussed in detail in Section 3.2.3, shows that if certain regularity conditions are fulfilled and probability density for each feature fulfills the following conditions: 1) probability density function is defined on a bounded interval and 2) probability density has non-zero minima on this interval, then the limits for maximal spacing are given by $\lim_{N\to\infty} s_x = C_1 \log(N)/N$ and $\lim_{N\to\infty} s_y = C_2 \log(N)/N$, where C_1 and C_2 are different constants. For small datasets, the area of each cell is given by

$A_N = s_x^c \times s_y^c$. Since $c < 0.5$, we see that upper limit is given by:

$$\lim_{N\to\infty} A_N = \lim_{N\to\infty} C_1 C_2 (\log(N)/N)^{2c} = 0. \tag{5.1}$$

Parzen's second condition is also fulfilled as a product of area, and the number of observations goes to infinity. We find that lower limit is given by:

$$\begin{aligned} \lim_{N\to\infty} N \times A_N &= \lim_{N\to\infty} N \times C_1 C_2 (\log(N)/N)^{2c} \\ &= \lim_{N\to\infty} N^{(1-2c)} C_1 C_2 (\log(N))^{2c} = \infty, \\ &\text{as } 1 - 2c > 0. \end{aligned} \tag{5.2}$$

Thus, estimated density for small datasets is consistent, according to Parzen density estimates. For larger datasets, we simply divide the $[0, 1]$ interval along each axis into $3\log(N)$ partitions. Thus, for a two-dimensional space normalized to $[0, 1] \times [0, 1]$, the area of each cell is $\lim_{N\to\infty} \frac{1}{9\log^2(N)}$. It should be noted that $\lim_{N\to\infty} \frac{1}{9\log^2(N)} = 0$ and $\lim_{N\to\infty} \frac{N}{9\log^2(N)} = \infty$.Thus, we obtain consistent density estimates according to Parzen's rules.

5.2.2 Obtaining biclusters from dense regions found with different dimension pairs

Once we have obtained dense regions in two-dimensional spaces, this information is used to obtain biclusters in higher dimensions. The pseudocode for the procedure is given as Algorithm 5.2 and the procedure is described in detail below.

5.2.2.1 Finding seed biclusters

To determine the set of observations related to each other, we identify subsets based on relative density within the two-dimensional Euclidean space defined by each pair of features using the procedure outlined in Section 5.2.1. However, because background noise can exhibit arbitrary distributions, this process of uncovering associations may yield considerable noise. Consequently, to mitigate noise interference, we seek a set of three features (feature triplet) for which a specified set of observations forms dense regions across each pair of features within the feature triplet. Accordingly, for a specific set of features $\{f_1, f_2, f_3\}$, dense regions are identified for the pairs $< f_1, f_2 >$, $< f_2, f_3 >$, and $< f_3, f_1 >$. The common observations forming dense regions are then determined by computing the intersection of these three pairs. If the resulting set of observations is significant, that is, the number of observations is larger than a user-defined parameter named `ObsInMinBase`, we consider it to be a seed bicluster. A discussion of the values of `ObsInMinBase` and other user-defined parameters is given in Section 5.3.1. Once the set of observations has been identified, we treat it as a seed bicluster and add other feature triplets which have significant overlap in their corresponding observation subsets. Note that this method of finding biclusters from seed biclusters is similar to that used by CBSC, but not exactly the same. Thus, the following subsection describes the procedure for obtaining biclusters in detail.

5.2.2.2 Using seed biclusters to get larger biclusters

We first arrange seed biclusters in descending order by the number of observations present in each seed bicluster. We iteratively used these seed biclusters as a base to find biclusters. When we use a particular seed bicluster for building a larger bicluster, we call it a base seed bicluster. We initialize a bicluster with a set of observations

Cl and a set of characteristics Clf as $< Cl, Clf >$ using a base seed bicluster S^*. Let S^* have a set of observations S_o^* and a set of characteristics S_f^*, then we initialize $Cl = S_o^*$ and $Clf = S_f^*$. We consider other seed biclusters S (except S^*) given by the observation set S_o and the feature set S_f. If the length of S_o^* is denoted by $length(S_o^*)$, we find seed biclusters that have at least $length(S_o^*) \times$ `Sim2Seed` observations in common with Cl, where `Sim2Seed` is a user-defined parameter and $S_f \bigcap Clf \neq \emptyset$. As we continue to find seed biclusters that match Cl, we update Clf by adding all the features present in the matching seed bicluster. This is repeated until no more additions can be made to the bicluster. Then we find observations that occur in at least `ObsInMinBase` seed biclusters, where `ObsInMinBase` is a user-defined parameter. These observations and features corresponding to the matching seed biclusters form an output bicluster.

The user also has the option of ignoring seed biclusters that have a very high overlap with some other seed bicluster already used as the base. To use this option, one must set the user-defined parameter `ReuseAllSeeds` to false. In this case, the user-defined parameter named `ReuseSeedSim` gives the threshold for the amount of overlap between seed biclusters. If the overlap between a base seed bicluster S^* and another seed bicluster is greater than `ReuseSeedSim` $\times$ `Sim2Seed` $\times$ $length(S^*)$, we do not use the matching seed bicluster as a base seed bicluster in future iterations. On the other hand, if the parameter `ReuseAllSeeds` is set to true, we will not ignore seed biclusters, and they will be used as a base in subsequent iterations. These two parameters control the amount of overlap between the biclusters in terms of observations.

5.2.2.3 Removing out similar biclusters

Biclusters that are similar to each other are removed using cosine similarity in the same way as was done in Section 4.5.1.4 for CBSC. The parameter `ClusSim` is used to remove overlapping biclusters, as it controls the maximum amount of similarity allowed between biclusters. For each pair of biclusters, we calculate the similarity between them using the formula: $\frac{Card(O_1 \bigcap O_2)}{\sqrt{Card(O_1)} \times \sqrt{Card(O_2)}} \times \frac{Card(F_1 \bigcap F_2)}{\sqrt{Card(F_1)} \times \sqrt{Card(F_2)}}$. Here, O_1 and O_2 denote the sets of observations in the first and second biclusters, respectively. Similarly, F_1 and F_2, respectively, denote the sets of features in the first and second biclusters. $Card()$ is used to denote the cardinality of a set. If the similarity between the two biclusters is found to be greater than `ClusSim`, we discard the smaller ones.

While CBSC [30] compares the density of a cell to the average density, RelDenClu compares the cell density to marginal densities, as discussed in Section 5.2. Both algorithms use similar pre-processing and a similar procedure to obtain biclusters from dense related regions. Many algorithms like those proposed by [20, 126, 139] use the apriori method to obtain biclusters from information obtained in low-dimensional spaces. The procedure used by RelDenClu and CBSC is different from the apriori method.

5.3 Algorithm for RelDenClu

In this section, we present the algorithm for RelDenClu. Before we find dense regions, we need to normalize the data to $[0, 1] \times [0, 1]$. This is done in the same way as CBSC using the functions $norm1$ or $norm2$ given as equations 3.49 and 3.50 and discussed in Sections 3.4.2 and 4.5.1.1.

Once the data is normalized, we find the dense regions using the procedure de-

scribed in Section 5.2.1. This step has been discussed extensively in Section 5.2.1 and the pseudocode is given as Algorithm 5.1. Then we find the biclusters using the process described in Section 5.2.2. The pseudocode for these steps has been presented as Algorithm 5.2. The steps used by RelDenClu have been summarized in Figure 5.2. Here we may note that the main difference between CBSC (summarized in Figure 4.3) and RelDenClu lies in the process of finding dense regions. This difference can be seen from the first block in Figure 4.3 and the first block in Figure 5.2. While, CBSC uses $(K, r, \delta) - \textit{bonded}$ sets to find dense regions, RelDenClu uses joint and marginal densities to find dense regions in a two-dimensional space.

In the following paragraphs, we discuss different parameters required by RelDenClu and also present some guidelines on the choice of the parameters.

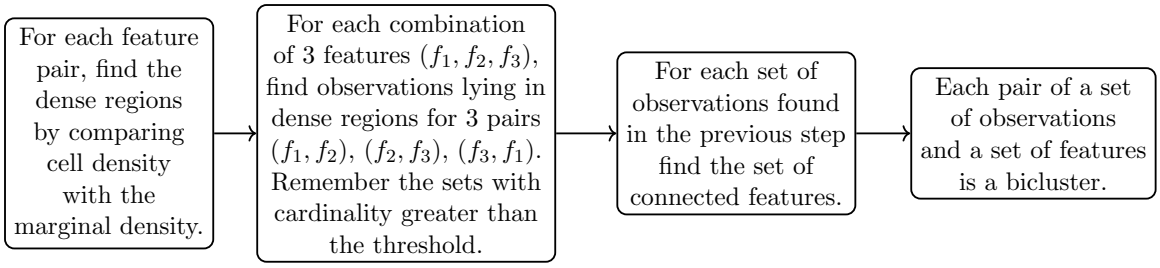


Figure 5.2: Broad outline showing how RelDenClu works

5.3.1 Choice of parameters for RelDenClu

The parameters used by RelDenClu have been described while discussing the details of the method. However, in this section, a list of parameters required by RelDenClu and the recommended range for these parameters is presented for better readability.

`MinSeedSize`: When choosing seed biclusters in Step 6 in Algorithm 5.2, this parameter is used. We make a list of all seed biclusters longer than `MinSeedSize` and use only these biclusters for further processing. Other seed biclusters are deleted.

Algorithm 5.1: RelDenseRegions()

```
   /* ---------------------------------------------------                    */
   /* This is an algorithm for finding dense regions in                      */
   /* two-dimensional space defined by dimensions i and j,                   */
   /* for small dataset method described in Section 5.2.1.1                  */
   /* Input is list of all observations in two-dimensional space defined by i and j */
   /* ---------------------------------------------------                    */
   /* sep_i is the maximum separation in dimension i                         */
 1 for each point p_a do
       /* neigh(p_a) is the set of points representing neighborhood of p_a   */
       /* Initialize neighborhood of each point                              */
 2     neigh(p_a) = ∅
 3 for each pair of points p_a, p_b do
       /* d_i denotes distance in dimension i and d_j denotes distance in dimension j */
 4     if ( d_i(p_a, p_b) < sep_i and d_j(p_a, p_b) < sep_j ) then
 5         connection(p_a, p_b) = 1
 6         Add p_b to neigh(p_a)
 7         Add p_a to neigh(p_b)
 8     else
 9         connection(p_a, p_b) = 0
10 finset = ∅
11 for each point p_a do
       /* MarginalNeigh_i(p_a) is set of points lying in the region given by
          (p_a − sep_i/2, p_a + sep_i/2) in dimension i and (0, 1) in dimension j */
       /* Card denotes cardinality of a set                                  */
       /* Here, n is the 'number of observations in dataset'                 */
12     if Card(neigh(p_a))/(sep_i * sep_j) > n and
         Card(neigh(p_a))/(sep_i * sep_j) > Card(MarginalNeigh_i(p_a))/sep_i and
         Card(neigh(p_a))/(sep_i * sep_j) > Card(MarginalNeigh_j(p_a))/sep_j then
13         finset = finset ∪ p_a
14 for each pair of points p_a, p_b in finset do
15     if Card(neigh(p_a) ∩ neigh(p_b)) * 4/(sep_i * sep_j) >
         max(MarginalNeigh_i(p_a)/sep_i, MarginalNeigh_j(p_a)/sep_j, MarginalNeigh_i(p_b)/sep_i, MarginalNeigh_j(p_b)/sep_j)
         then
           /* Merge the two cells i.e., cell connected to one is automatically connected to
              other                                                          */
           /* connection(p_a, :) denotes all connections of p_a             */
16         connection(p_a, :) = connection(p_a, :) ∪ connection(p_b, :)
17         connection(p_b, :) = connection(p_a, :)
   /* Perform connected components according to new connections              */
18 while new connections are formed do
19     for each pair of points p_a, p_b connected to each other do
20         connection(p_a, :) = connection(p_a, :) ∪ connection(p_b, :)
21         connection(p_b, :) = connection(p_a, :)
   /* Retain only those points from finset which have merged with other points */
22 finset=finset[sum(connection(p_a, :)) > 0]
23 npc = Card(finset)
24 Find connected components in finset using radius rad = log(npc)/npc
   /* k^th connected component is denoted by Pts(i, j, k)                    */
   /* Let total number of connected components for given pair of dimensions be m_cc */
25 Return points < Pts(f_i, f_j, 1), Pts(f_i, f_j, 2) ··· , Pts(f_i, f_j, m_cc) >
```

Algorithm 5.2: GetBiClusters

```
   /* This algorithm gives the process of obtaining biclusters from dense regions obtained by
      Algorithm 5.1                                                                          */
   /* Let the data be D                                                                      */
   /* Normalize the data using norm_1 if data seems to be                                    */
   /* coming from a bounded distribution otherwise use norm_2                                */
 1 Normalize data using the procedure given in Section 5.3
   /* Each pair of dimensions may contain several sets representing dense regions            */
 2 for each pair of features < f_i, f_j > do
       /* This is described in Section 5.2.1.1                                              */
 3     Pts(f_i, f_j, 1), Pts(f_i, f_j, 2), ···=RelDenseRegions(D_{*,fi}, D_{*,fj})
   /* Initialize list of seed biclusters as given in the first paragraph of Section 5.2.2   */
 4 Tlist = ∅ Flist = ∅ for each set of 3 features f_i, f_j, f_k and each combination of u, v, w do
       /* Check if a seed bicluster is formed by each dense region for a set of three features
          */
       /* dense regions numbered u, v and w in dimension pairs < f_i, f_j >, < f_j, f_k >   */
       /* and < f_k, f_i > are checked for each combination between dense regions           */
 5     T_x = Pts(f_i, f_j, u) ∩ Pts(f_j, f_k, v) ∩ Pts(f_k, f_i, w)
 6     if length(T_x < MinSeedSize) then
 7         Delete (T_x)
 8     else
 9         F_x =< u, v, w >
10         Add T_x to Tlist
11         Add F_x to Flist
12 Sort Tlist in descending order of length
   /* Remaining part of algorithm corresponds to Section 5.2.2                               */
13 Clusterlist = ∅
14 for each T_x in Tlist not marked as ignore do
15     Clf = F_x
16     SelectedTlist = T_x
17     for each T_y in Tlist do
18         if (Clf ⋂ F_y) ≠ ∅ ∧ length(T_x ⋂ T_y) > Sim2Seed × length(T_x) then
19             Clf = Clf ∪ F_y
20         if ReuseAllSeeds == FALSE ⋀ length(T_x ⋂ T_y) > ReuseSeedSim × Sim2Seed × length(T_x) then
               /* This seed bicluster will not be used as base                           */
21             Mark T_y as ignore
22     Goto 17 if seed biclusters have been added
23     Cl is list of observations which occur in at least ObsInMinBase bases present in SelectedTlist
24     if Cl ≠ ∅ then
25         Clusterlist = Clusterlist ∪ < Cl, Clf >
26 for each pair of biclusters in Clusterlist < Cl_i, Clf_i >, < Cl_j, Clf_j > do
       /* Formula for cosine similarity is given in Section 5.2.2                        */
27     if Cosine Similarity between < Cl_i, Clf_i >, < Cl_j, Clf_j >> ClusSim then
28         Delete the smaller bicluster
```

Setting lower values will allow the algorithm to detect biclusters with fewer observations. Setting a larger value will leave out smaller biclusters and report only large ones. If we want all small and large biclusters, it can be set to three, as we have defined a common dense region based on the connectedness for three pairs in Section 5.2.2.

`Sim2Seed`: In Section 5.2.2, a base seed bicluster is compared to other seed biclusters to see if the latter can be included in the bicluster, based on the similarity between the two. This similarity between the base seed bicluster S and other seed biclusters W is judged using the number of observations present in both S and W. If this number is greater than `Sim2Seed` $\times\ length(S)$, they are said to be similar. The range $(0.6, 0.98)$ is seen to give stable results.

`ReuseAllSeeds`: Each seed bicluster can be used as a base for finding a bicluster. For using each seed bicluster as a base, this parameter should be set to `TRUE`. Otherwise, seeds previously found to have a high overlap with base seed bicluster, are not used as bases in further calculations.

`ReuseSeedSim`: If we ignore some seed biclusters while choosing the bases, we set `ReuseAllSeeds` to `FALSE`. In this case, we need to set the parameter `ReuseSeedSim`. If the number of common observations between the base seed bicluster S and other seed bicluster W is greater than `ReuseSeedSim` $\times$ `Sim2Seed` $\times\ length(S)$, then W is not used as a base. The value of this parameter lies in $(0, 1)$. Setting a higher value of this parameter allows the algorithm to detect biclusters with a greater amount of overlap. If the parameter is set to a lower value, less overlap occurs and computation time is also less.

`ObsInMinBase`: If an observation is present in at least `ObsInMinBase` of dense regions forming a bicluster, it is included in the set of observations corresponding to the bicluster.

`ClusSim`: If the cosine similarity defined in Section 5.2.2, between biclusters is found to be greater than `ClusSim`, the smaller bicluster is ignored. Unlike `ReuseSeedSim`, here we compare the biclusters in terms of both the number of observations and the number of features. Also, it weeds out biclusters once they are found, and thus provides finer control.

In this chapter, several experiments have been performed with fixed parameter values that are mentioned in Section 5.4. Later, in Chapter 6, the results for RelDenClu have been aggregated over several possible parameter values, providing insight into the behavior of RelDenClu when different parameter values are selected.

5.3.2 Time complexity of RelDenClu

The time complexity of ROCCO has been reported by He and Moreira-Matias [148]. The time complexity of other methods used for comparison has been discussed in the previous chapter. We are reporting the time complexity of RelDenClu, where N and M, respectively, are the number of observations and the number of features in a given dataset. The time complexity of RelDenClu is $(log(N)M)^2 + NM^3$, where N and M represent the number of observations and features in the dataset, respectively.

5.4 Experimental setup and methodology for analysis with RelDenClu

In this section, we present the experimental setup and methodology used to perform investigation with synthetic and real-world datasets. Real-world datasets include five UCI-ML datasets [185] and two datasets designed to understand the spread of the COVID-19 disease [193, 197, 198].

5.4.1 Experimental setup for synthetic datasets

To test the efficacy of RelDenClu, we performed experiments on simulated and real-world datasets. Experiments with simulated datasets allow us to study the behavior of RelDenClu on datasets with known characteristics. The synthetic datasets used for the experiments and the methodology to evaluate them is the same as that described in the Chapter 4 in Section 4.7.1.

The parameters used for RelDenClu for this analysis are `MinSeedSize` $= N/8$ where N is the number of observations in the dataset, `ObsInMinBase` $= 5$, `Sim2Seed` $= 0.85$, `ReuseAllSeeds` $= False$, `ReuseSeedSim` $= 0.75$, `ClusSim` $= 1$.

5.4.2 Methods used for comparison with RelDenClu

The performance of RelDenClu is compared with eight other algorithms namely CLIQUE [128], Proclus [127], ITL [186], Subclu [20], P3C [125], UniBic [145], CBSC [30] and ROCCO [148]. CLIQUE, P3C, and Subclu are density-based methods. ITL is an information-theoretic method, while Proclus is known to provide stable results. ROCCO is a parameter-free coclustering method. Comparison with UniBic and CBSC is important as they specifically aim at finding biclusters based on non-linear dependence. The parameters used by all the methods used for the comparison except ROCCO are the same as those used in the previous chapter. For ROCCO the parameter K is required for finding the K nearest neighbors adjacency matrix for spectral analysis. The authors have shown that the performance of ROCCO is consistent for any of the values of K of 8, 9 or 10, making it a hyperparameter-free algorithm [148]. Thus, we have chosen $K = 9$ for our experiments.

The performance of RelDenClu is evaluated in terms of accuracy and execution time in the same way as done for CBSC in Chapter 4. The accuracy for synthetic

datasets has been reported in Tables 5.1-5.4 and Figures 5.3-5.6. The execution time for these datasets has been reported in Tables 5.5-5.8.

5.4.3 Experimental setup for UCI-ML datasets

Like CBSC, we demonstrate the use of RelDenClu on UCI-ML datasets in two different ways. First, as an application to supervised learning, biclustering algorithms have been used to generate new features for two different datasets that lead to improved classification accuracy. This is discussed in Section 5.4.3.1

Second, in Section 5.4.3.2, we describe the methodology for using RelDenClu for the clustering of three datasets.

As in the case of CBSC, for real-world datasets, we initially apply normalization $norm_1$. If no biclusters are produced using $norm_1$, then the biclusters are found after applying $norm_2$ to the dataset.

5.4.3.1 Setup for for experiments with classification of UCI-ML datasets

RelDenClu has been used to generate new binary features that are used to improve the classification performance on two UCI ML datasets [185]. Such experiments were also performed with CBSC and the datasets and methodology are described in detail in Section 4.7.2.1 of Chapter 4. The datasets used for the classification experiments are described in Chapter 4 in Section 4.7.2.1.

For classification, the parameter values for RelDenClu are selected using a 10-fold cross-validation. The different parameter values used for the 10-fold cross-validation are as follows. For the parameter `MinSeedSize`, we have taken the values $\{N/8, N/16\}$, where N is the number of observations in the dataset. We have experimented with the values $\{3, 5\}$ for `ObsInMinBase`. For the parameter `Sim2Seed`

and we have taken the values $\{0.65, 0.8, 0.95\}$. The parameter `ReuseAllSeeds` can be set to $True$ or $False$. We have experimented with both values. For parameter `ReuseSeedSim`, we have experimented with the values $\{0.25, 0.5, 0.75\}$. For the parameter `ClusSim`, we have experimented with the values $\{0.5, 1\}$. For ROCCO we use the parameter value $K = 9$. The parameter values for all other methods are the same as mentioned in Section 4.7.2.1. The evaluation criterion is also the same, that is accuracy, NMI, ARI, precision, recall, and G-Score. The results of these experiments are reported in Table 5.9 and Figures 5.8a-5.8f.

5.4.3.2 Setup for for experiments with clustering of UCI-ML datasets

We use RelDenClu for unsupervised learning to identify if its detected biclusters match known data classes, as we did for CBSC. This was done for three UCI ML datasets mentioned in Section 4.7.2.2 of Chapter 4. The methodology for evaluating performance has also been discussed there.

For these datasets, the parameters used for RelDenClu are `MinSeedSize` $= N/8$ where N is the number of observations in the dataset, `ObsInMinBase` $= 3$, `Sim2Seed` $= 0.6$, `ReuseAllSeeds` $= False$, `ReuseSeedSim` $= 0.75$, `ClusSim` $= 1$. Six performance indices are reported. These indices are accuracy, NMI, ARI, precision, recall and G-score. The results for these experiments are reported in Table 5.10 and Figures 5.8a-5.8f.

5.4.4 Setup and methodology for application of RelDenClu to COVID-19 dataset for finding relevant lifestyle factors

Like CBSC, we have also used RelDenClu to find lifestyle factors that impact COVID-19 infection rates. As mentioned in Section 4.7.3, the dataset is based on World Development Indicators and COVID-19 infection rates. As mentioned earlier, biclustering is useful in scenarios where subsets of observations and features are required together. Thus, we applied RelDenClu for the analysis of lifestyle factors that impact COVID-19 rates. The parameters used for RelDenClu for this analysis are `MinSeedSize` = 15, `ObsInMinBase` = 10, `Sim2Seed` = 0.95, `ReuseAllSeeds` = $False$, `ReuseSeedSim` = 0.75, `ClusSim` = 0.75. The remainder of the methodology is the same as that used in Section 4.7.3. The results are provided in Table 5.11 and Figure 5.9.

5.4.5 Setup and methodology for application of RelDenClu to identify vaccines existing before COVID-19 that might help in reducing COVID-19 infection rates

We have used RelDenClu to find out whether any of the existing vaccines before 2020 is likely to provide protection against COVID-19 using the same methodology as CBSC in Section 4.7.4. The parameters used for RelDenClu for this analysis are `MinSeedSize` = 20, `ObsInMinBase` = 5, `Sim2Seed` = 0.75, `ReuseAllSeeds` = $False$, `ReuseSeedSim` = 0.75, `ClusSim` = 1. The details of methodology and datasets can be found in Section 4.7.4. The results are reported in Section 5.5.4.

5.5 Results

This section presents the results obtained for synthetic and real-world datasets using the methodology discussed in Section 5.4.

5.5.1 Results on simulated datasets

In the section we report the accuracy and execution time of different biclustering methods for simulated datasets. We will also discuss the properties of RelDenClu with reference to the results on simulated datasets.

5.5.1.1 Accuracy for simulated datasets

Table 5.1: Accuracy for Non-Linear simulated datasets (Mean and Standard Deviation for 10 datasets of each type)

Method	Accuracy	Datasets		
		Non-Monotonous	Non-Linear 1	Non-Linear 2
RelDenClu	Mean	0.818	0.933	0.922
	Std. Dev.	0.019	0.032	0.009
ROCCO	Mean	0.751	0.82	0.836
	Std. Dev.	0.020	0.015	0.020
CBSC	Mean	0.788	0.874	0.907
	Std. Dev.	0.030	0.014	0.017
UniBic	Mean	0.764	0.804	0.839
	Std. Dev.	0.004	0.142	0.107
P3C	Mean	0.75	0.765	0.764
	Std. Dev.	0.001	0.004	0.003
Subclu	Mean	0.746	0.785	0.793
	Std. Dev.	0.005	0.017	0.016
ITL	Mean	0.75	0.75	0.75
	Std. Dev.	0.009	0.004	0.002
Proclus	Mean	0.752	0.781	0.792
	Std. Dev.	0.009	0.014	0.007
CLIQUE	Mean	0.75	0.784	0.782
	Std. Dev.	0.000	0.002	0.002

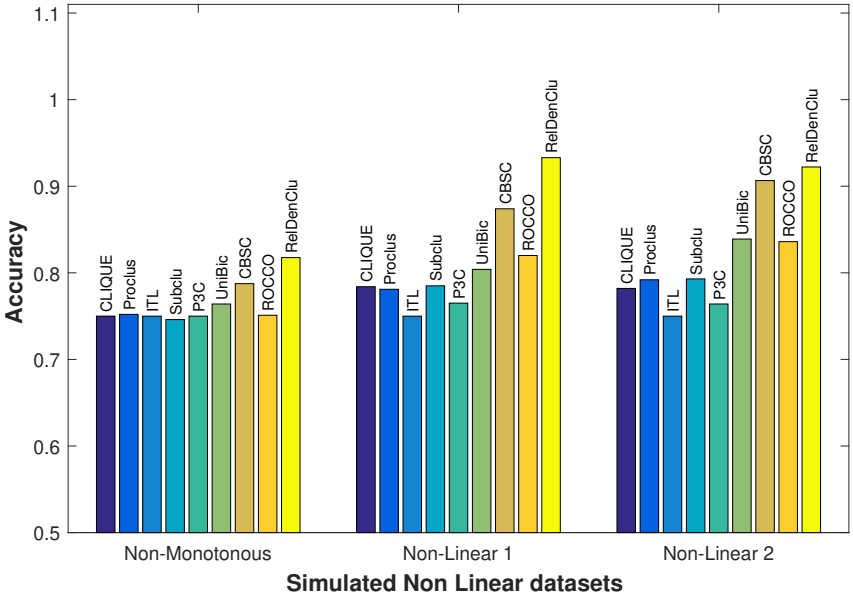


Figure 5.3: Accuracy obtained using various algorithms for Non-Linear datasets

Table 5.2: Accuracy for Base and transformed simulated datasets (Mean and Standard Deviation for 10 datasets of each type)

Method	Accuracy	Datasets							
		Base	Scaled	Translated	Linear Transform	Point Proportion	Cluster Proportion	Noisy Uniform	Permutations
RelDenClu	Mean	0.996	0.996	0.997	0.997	0.997	0.999	0.981	0.996
	Std. Dev.	0.002	0.002	0.002	0.002	0.003	0.002	0.008	0.002
ROCCO	Mean	0.813	0.881	0.883	0.851	0.881	0.829	0.839	0.819
	Std. Dev.	0.019	0.015	0.014	0.020	0.034	0.036	0.012	0.019
CBSC	Mean	0.935	0.935	0.935	0.936	0.937	0.822	0.936	0.935
	Std. Dev.	0.057	0.057	0.057	0.057	0.060	0.111	0.043	0.057
UniBic	Mean	0.900	0.848	0.802	0.793	0.603	0.641	0.832	0.900
	Std. Dev.	0.034	0.211	0.159	0.034	0.046	0.078	0.124	0.034
P3C	Mean	0.781	0.782	0.779	0.779	0.787	0.802	0.772	0.783
	Std. Dev.	0.040	0.040	0.041	0.041	0.022	0.092	0.032	0.015
Subclu	Mean	0.794	0.759	0.788	0.765	0.802	0.724	0.795	0.798
	Std. Dev.	0.023	0.016	0.019	0.025	0.030	0.020	0.017	0.022
ITL	Mean	0.751	0.752	0.750	0.751	0.750	0.750	0.750	0.750
	Std. Dev.	0.005	0.004	0.004	0.001	0.001	0.002	0.002	0.001
Proclus	Mean	0.785	0.768	0.803	0.770	0.800	0.711	0.792	0.785
	Std. Dev.	0.013	0.015	0.023	0.028	0.024	0.024	0.010	0.015
CLIQUE	Mean	0.776	0.776	0.776	0.776	0.776	0.683	0.720	0.773
	Std. Dev.	0.016	0.016	0.016	0.016	0.016	0.012	0.011	0.015

Table 5.3: Accuracy for simulated datasets with Normal Distribution (Mean and Standard Deviation for 10 datasets of each type)

Method	Accuracy	Datasets		
		Normal	Noisy Normal 1	Noisy Normal 2
RelDenClu	Mean	0.991	0.948	0.881
	Std. Dev.	0.004	0.025	0.020
ROCCO	Mean	0.753	0.755	0.752
	Std. Dev.	0.011	0.009	0.003
CBSC	Mean	0.895	0.901	0.897
	Std. Dev.	0.062	0.059	0.053
UniBic	Mean	0.869	0.794	0.776
	Std. Dev.	0.025	0.098	0.009
P3C	Mean	0.767	0.758	0.765
	Std. Dev.	0.017	0.047	0.009
Subclu	Mean	0.783	0.78	0.769
	Std. Dev.	0.010	0.015	0.006
ITL	Mean	0.75	0.749	0.750
	Std. Dev.	0.003	0.001	0.001
Proclus	Mean	0.783	0.772	0.769
	Std. Dev.	0.017	0.011	0.006
CLIQUE	Mean	0.799	0.787	0.776
	Std. Dev.	0.011	0.015	0.008

Table 5.4: Accuracy for Overlapping simulated datasets (Mean and Standard Deviation for 10 datasets of each type)

Method	Accuracy	Datasets			
		Overlap 1		Overlap 2	
		First	Second	First	Second
RelDenClu	Mean	0.998	0.993	0.997	0.992
	Std. Dev.	0.001	0.003	0.002	0.005
ROCCO	Mean	0.773	0.817	0.751	0.801
	Std. Dev.	0.008	0.007	0.002	0.002
CBSC	Mean	0.953	0.945	0.952	0.949
	Std. Dev.	0.010	0.003	0.011	0.009
UniBic	Mean	0.940	0.626	0.833	0.775
	Std. Dev.	0.037	0.032	0.130	0.143
P3C	Mean	0.751	0.801	0.750	0.800
	Std. Dev.	0.002	0.004	0.001	0.001
Subclu	Mean	0.761	0.801	0.763	0.801
	Std. Dev.	0.005	0.002	0.004	0.002
ITL	Mean	0.750	0.800	0.750	0.800
	Std. Dev.	0.001	0.001	0.001	0.001
Proclus	Mean	0.760	0.804	0.761	0.802
	Std. Dev.	0.009	0.004	0.008	0.003
CLIQUE	Mean	0.750	0.800	0.750	0.800
	Std. Dev.	0.001	0.001	0.001	0.001

For simulated data, experiments are carried out with 10 different simulations and the average accuracy (over 10 simulations) and corresponding standard deviation (denoted as "deviation") values are reported in Tables 5.1-5.4. Comparison is made with eight other methods. We see that RelDenClu provides better accuracy for fifteen of the sixteen simulated datasets. Data is normalized using $norm_2$ for "Normal" and "Noisy Normal" datasets. For all other datasets, $norm_1$ is used.

The columns ("Non-Monotonous", "Non-linear 1" and "Non-linear 2") of Table 5.1 represent datasets having biclusters based on non-linear relations. We see that RelDenClu provides higher accuracy for these three datasets. RelDenClu finds the biclusters with higher accuracy for "Non-Monotonous" dataset containing highly non-monotonous relations as seen from the first column. In spite of the significantly varied distribution of background observations, RelDenClu finds biclusters with greater accuracy, as seen from results presented in the "Non-Linear 1" column. Its accuracy for another dataset containing bicluster based on non-linear relations is also higher in comparison with other algorithms as seen from column "Non-Linear 2". The values of accuracy obtained using different algorithms on these datasets are presented in Figure 5.3 as bar graphs, for better illustration. The analysis conducted on these three datasets demonstrate that RelDenClu is capable of identifying biclusters where features exhibit non-linear or non-monotonous relationships, as discussed in Section 6.2. This capability stems from RelDenClu's utilization of relative density for the identification of the initial observation set and the implementation of the relation index MIDI to determine the final biclusters.

The first column of Table 5.2 ("Base") represents a dataset containing biclusters based on a linear relation. The other seven columns of the table ("Scaled", "Translated", ..., "Permutations") give results for datasets obtained by applying different transforms to the "Base" dataset. These datasets allow us to analyze the behavior

of RelDenClu in a way similar to the analysis performed by Ness [24] for clustering algorithms. From Figure 5.4 (shown in parts 5.4a and 5.4b) and the table mentioned above, we find that RelDenClu has produced better results compared to other methods for the "Base" dataset and its transforms, that is, "Scaled",..., "Permutations".

The three columns of Table 5.3 ("Normal", "Noisy Normal 1" and "Noisy Normal 2"), contain results for datasets generated using the normal distribution. For the "Normal" and "Noisy Normal 1" (noise with standard deviation 0.1) datasets RelDenClu performs better than the other methods. In the case of "Noisy Normal 2" (noise with standard deviation 0.2), we find that CBSC performs slightly better than RelDenClu. The performance for these datasets is reported in Table 5.3 and Figure 5.5.

The first two columns of Table 5.4 ("Overlap 1"), present results for a dataset containing two biclusters with an overlap of 12%. We find that RelDenClu has the best performance among the methods used for comparison. These results are presented graphically in Figure 5.6. Similarly, the third and fourth columns present results for a dataset containing two biclusters with an overlap of 20% ("Overlap 2"). For these two biclusters, RelDenClu also has the highest accuracy.

These results for simulated datasets suggest some properties of RelDenClu which will be discussed in Section 5.5.1.3.

5.5.1.2 Execution time for simulated datasets and time complexity

The execution times of the methods are reported in Tables 5.5-5.8. The datasets used here are the same as those used in Tables 5.1-5.4. The algorithm is executed with an Intel Core i7 CPU and 8 GB memory. As seen from the tables mentioned above, the execution time for RelDenClu is less than CBSC. In many cases, it is also less than ITL and ROCCO. It is high compared to density-based methods such as

Table 5.5: Execution time in seconds for Non-Linear simulated datasets

Method	Datasets		
	Non-Monotonous	Non-Linear 1	Non-Linear2
RelDenClu	0.298	2.913	0.737
ROCCO	31.402	4.087	3.164
CBSC	61.674	93.630	69.359
UniBIc	0.012	0.583	0.612
P3C	1.440	0.359	0.650
Subclu	0.037	0.064	0.052
ITL	15.000	9.096	9.248
Proclus	0.026	0.060	0.055
CLIQUE	0.001	0.011	0.007

Table 5.6: Execution time in seconds for Base and transformed simulated datasets

Method	Datasets							
	Base	Scale	Translated	Linear	Point Proportion	Cluster Proportion	Noisy	Permutations
RelDenClu	6.970	6.884	7.273	6.879	80.673	55.676	30.128	6.293
ROCCO	2.939	3.021	2.681	2.794	29.275	7.509	2.874	2.958
CBSC	81.585	82.717	81.450	81.840	97.645	94.466	96.439	81.654
UniBIc	0.583	0.612	0.649	0.673	2.596	1.436	0.621	0.583
P3C	0.513	0.955	0.952	0.433	1.083	0.848	0.458	0.588
Subclu	0.056	0.043	0.050	0.055	0.120	0.082	0.049	0.070
ITL	3.171	3.302	3.498	3.276	9.994	6.519	2.94	3.272
Proclus	0.052	0.055	0.049	0.055	0.120	0.088	0.056	0.063
CLIQUE	0.105	0.015	0.015	0.018	0.033	0.021	0.014	0.013

Table 5.7: Execution time in seconds for simulated datasets with Normal Distribution

Method	Datasets		
	Normal	Noisy Normal 1	Noisy Normal 2
RelDenClu	53.006	53.733	52.754
ROCCO	6.258	5.128	2.938
CBSC	67.290	88.490	73.850
UniBic	0.531	0.577	0.001
P3C	18.840	9.751	6.322
Subclu	0.071	0.061	0.050
ITL	1.187	1.259	20.639
Proclus	0.054	0.052	0.084
CLIQUE	0.071	0.044	0.715

Table 5.8: Execution time in seconds for Overlapping simulated datasets

Method	Datasets	
	Overlap 1	Overlap 2
RelDenClu	5.639	5.671
ROCCO	2.571	25.663
CBSC	72.647	68.848
UniBic	2.092	2.775
P3C	0.049	0.016
Subclu	0.074	0.158
ITL	15.102	15.091
Proclus	0.074	0.087
CLIQUE	0.014	0.032

Subclu, Proclus, and CLIQUE. However, it is already seen in Tables 5.1-5.4, that the accuracy of RelDenClu is better than these methods for most of the simulated datasets.

5.5.1.3 A discussion on properties of RelDenClu as seen from experiments on Simulated datasets

It is already seen that RelDenClu can find relation based biclusters with better accuracy compared to other algorithms for most of the simulated datasets, as seen from Tables 5.1-5.4 and Figures 5.3-5.6. In this section, we discuss some of the observed properties of RelDenClu.

RelDenClu exhibits invariance to scaling, translation, and linear transformations, as these transformations do not alter the bin length or estimated densities. However, theoretically, the methodology is not invariant to arbitrary order-preserving transformations, which have the potential to modify the density of the observations. An analysis was performed to compare the accuracy obtained for the "Base" dataset with its various transformations. It is observed that there is no significant differences in the accuracy of RelDenClu for data sets "Scaled", "Translated", "Linear Transform", "Point proportion", "Cluster Proportion" and "Permutations". Accuracy for the "Noisy Uniform", "Noisy Normal 1" and "Noisy Normal 2" datasets are only slightly lower than the "Base" and "Normal" datasets. Compared to other algorithms, RelDenClu still has a higher accuracy for noisy datasets.

We may note that the biclusters in the "Base" dataset can also be seen as scaling biclusters, since each column in the bicluster can be obtained by multiplying another column by a constant value. Also, the biclusters in the "Translated" datasets can be seen as a combination of scaling and shift biclusters. Since RelDenClu finds biclusters based on feature relations, it will be able to find biclusters based on scaling and

shifting of columns. However, RelDenClu will not be able to find biclusters based on the shifting and scaling of rows, as it specifically aims to find biclusters based on feature relations.

We find that RelDenClu has similar performance for the "Base" and various transformed datasets. However, RelDenClu has a higher accuracy for these datasets compared to other methods.

In conclusion, the RelDenClu algorithm demonstrates robust performance when applied to datasets subjected to various transformations or the introduction of noise. Its efficacy is also notable on datasets characterized by non-linear feature relationships.

5.5.2 Results for experiments with UCI-ML datasets

5.5.2.1 Experimental results for classification of UCI-ML datasets

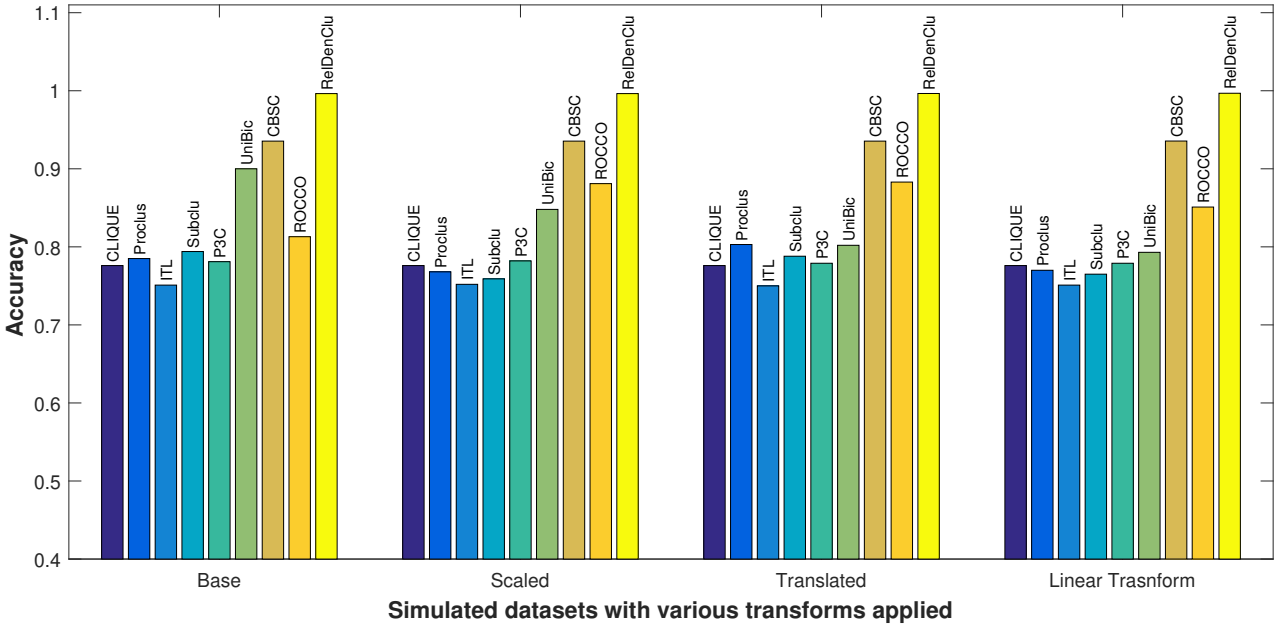
The results for the classification of the UCI-ML datasets are reported in Table 5.9 and the performance is also presented as bar graphs in Figures 5.7a-5.7f. The analysis of the student dropout dataset indicates that the application of RelDenClu, CBSC, and P3C results in an improvement over the original accuracy. For the Credit Card dataset, the features derived from RelDenClu lead to a substantial improvement in accuracy. The accuracy gain is greater compared to eight other methods, thereby facilitating a more effective identification of credit card defaulters.

Table 5.9: Classification performance for Credit card and Student Dropout datasets (after adding features generated using different biclustering methods)

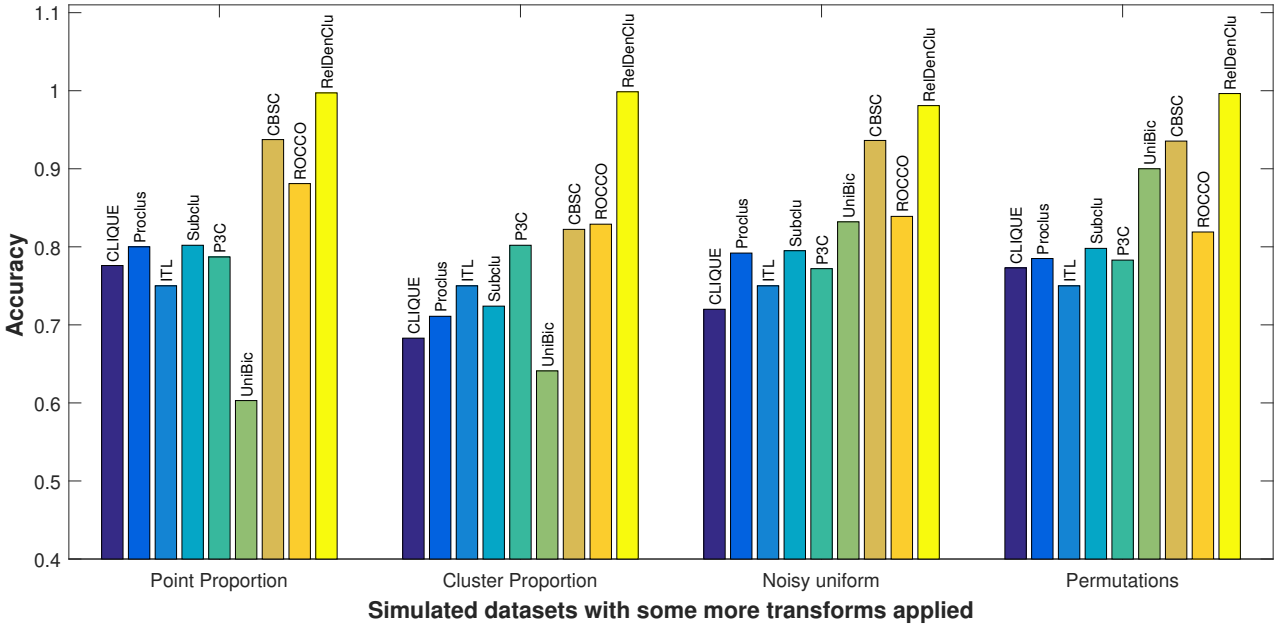
Dataset	**Credit Card [175]**						
Method	Accuracy		NMI	ARI	Average		
	Mean	S. Dev.			Prec.	Recall	G-Score
RelDenClu	0.761	0.006	0.108	0.227	0.672	0.702	0.685
ROCCO	0.269	0.005	0.000	-0.007	0.510	0.504	0.355
CBSC	0.755	0.006	0.074	0.185	0.647	0.651	0.649
UniBic	0.397	0.006	0.015	-0.047	0.559	0.562	0.472
P3C	0.702	0.012	0.090	0.149	0.642	0.690	0.654
Subclu	0.547	0.098	0.033	0.004	0.581	0.616	0.560
ITL	0.269	0.010	0.006	-0.035	0.562	0.520	0.359
Prolus	0.722	0.015	0.097	0.172	0.650	0.696	0.665
CLIQUE	0.544	0.006	0.032	0.003	0.580	0.614	0.558
Original	0.703	0.028	0.088	0.147	0.640	0.689	0.654
Dataset	**Student Dropout [176]**						
Method	Accuracy		NMI	ARI	Average		
	Mean	S. Dev.			Prec.	Recall	G-Score
RelDenClu	0.685	0.018	0.219	0.317	0.616	0.602	0.608
ROCCO	0.221	0.011	0.010	-0.003	0.434	0.360	0.264
CBSC	0.681	0.016	0.207	0.302	0.604	0.583	0.589
UniBic	0.610	0.009	0.144	0.125	0.550	0.450	0.455
P3C	0.679	0.020	0.215	0.312	0.595	0.581	0.585
Subclu	0.674	0.018	0.197	0.291	0.602	0.586	0.591
ITL	0.185	0.003	0.004	-0.002	0.417	0.336	0.184
Proclus	0.676	0.018	0.206	0.305	0.601	0.589	0.593
CLIQUE	0.675	0.022	0.195	0.295	0.597	0.576	0.582
Original	0.675	0.015	0.200	0.295	0.593	0.576	0.580

Table 5.10: Results for clustering of UCI-ML datasets

Dataset	**Magic Gamma [177]**								
Method	Accuracy	NMI	ARI	Class 1			Class 2		
				Prec.	Recall	G-Score	Prec.	Recall	G-Score
RelDenClu	0.737	0.179	0.225	0.593	0.808	0.692	0.870	0.699	0.780
ROCCO	0.653	0.006	0.009	0.845	0.016	0.117	0.652	0.998	0.807
CBSC	0.713	0.114	0.179	0.582	0.655	0.617	0.799	0.745	0.772
UniBic	0.655	0.006	0.015	0.654	0.991	0.805	0.677	0.035	0.153
P3C	0.705	0.080	0.110	0.690	0.992	0.827	0.919	0.177	0.404
Subclu	0.639	0.001	0.011	0.653	0.943	0.785	0.425	0.077	0.181
ITL	0.652	0.027	0.007	0.953	0.012	0.107	0.651	1.000	0.807
Proclus	0.648	0.013	0.048	0.676	0.878	0.770	0.498	0.223	0.333
CLIQUE	0.685	0.044	0.104	0.703	0.889	0.791	0.600	0.308	0.430
Dataset	**Breast Cancer (Wisconsin Original)** [178]								
Method	Accuracy	NMI	ARI	Class 1			Class 2		
				Prec.	Recall	G-Score	Prec.	Recall	G-Score
RelDenClu	0.956	0.724	0.831	0.927	0.950	0.938	0.973	0.959	0.966
ROCCO	0.660	0.011	0.022	0.733	0.046	0.184	0.659	0.991	0.808
CBSC	0.931	0.618	0.742	0.893	0.912	0.903	0.952	0.941	0.947
UniBic	0.843	0.330	0.462	0.833	0.690	0.758	0.847	0.926	0.886
P3C	0.796	0.261	0.322	0.981	0.427	0.647	0.763	0.995	0.872
Subclu	0.814	0.296	0.370	0.975	0.481	0.685	0.781	0.993	0.880
ITL	0.656	0.007	0.011	0.833	0.021	0.132	0.654	0.998	0.808
Proclus	0.657	0.008	0.023	0.600	0.063	0.194	0.660	0.977	0.803
CLIQUE	0.921	0.579	0.706	0.881	0.895	0.888	0.943	0.935	0.939
Dataset	**Electrical Grid**								
Method	Accuracy	NMI	ARI	Class 1			Class 2		
				Prec.	Recall	G-Score	Prec.	Recall	G-Score
RelDenClu	0.665	0.121	0.108	0.858	0.570	0.699	0.524	0.833	0.661
ROCCO	0.650	0.011	0.030	0.651	0.973	0.796	0.631	0.082	0.228
CBSC	0.678	0.077	0.124	0.549	0.616	0.582	0.766	0.712	0.739
UniBic	0.638	0.001	0.006	0.497	0.026	0.114	0.641	0.985	0.794
P3C	0.638	0.000	0.000	0.638	1.000	0.799	0.000	0.000	0.000
Subclu	0.631	0.000	0.001	0.383	0.030	0.107	0.639	0.973	0.788
ITL	0.647	0.009	0.029	0.572	0.096	0.235	0.652	0.959	0.791
Proclus	0.635	0.001	0.006	0.444	0.038	0.129	0.641	0.973	0.790
CLIQUE	0.532	0.000	0.002	0.372	0.425	0.398	0.645	0.593	0.619



(a) Accuracy obtained for Base dataset and its transforms



(b) Accuracy for 4 more transforms applied to Base dataset

Figure 5.4: Accuracy obtained for simulated datasets with various transforms applied

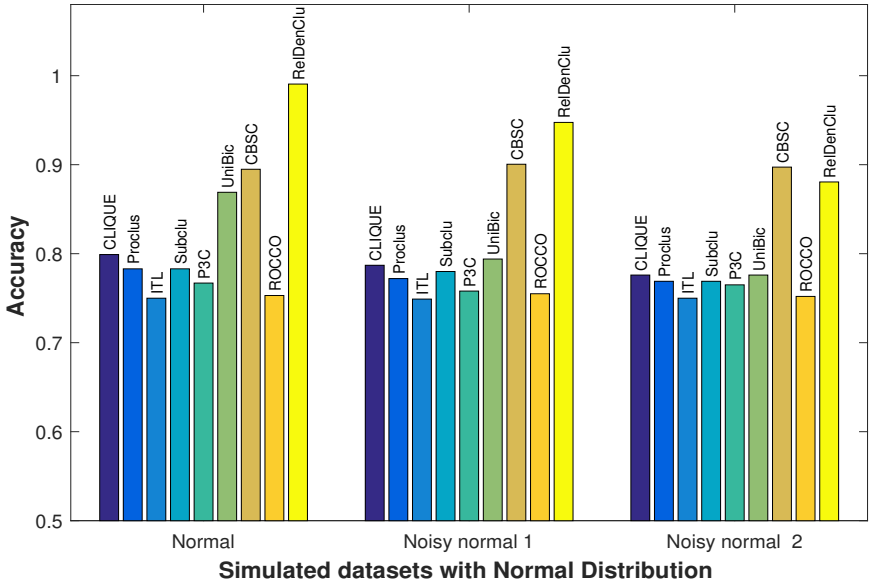


Figure 5.5: Accuracy obtained using various algorithms for Normal and Noisy Normal datasets

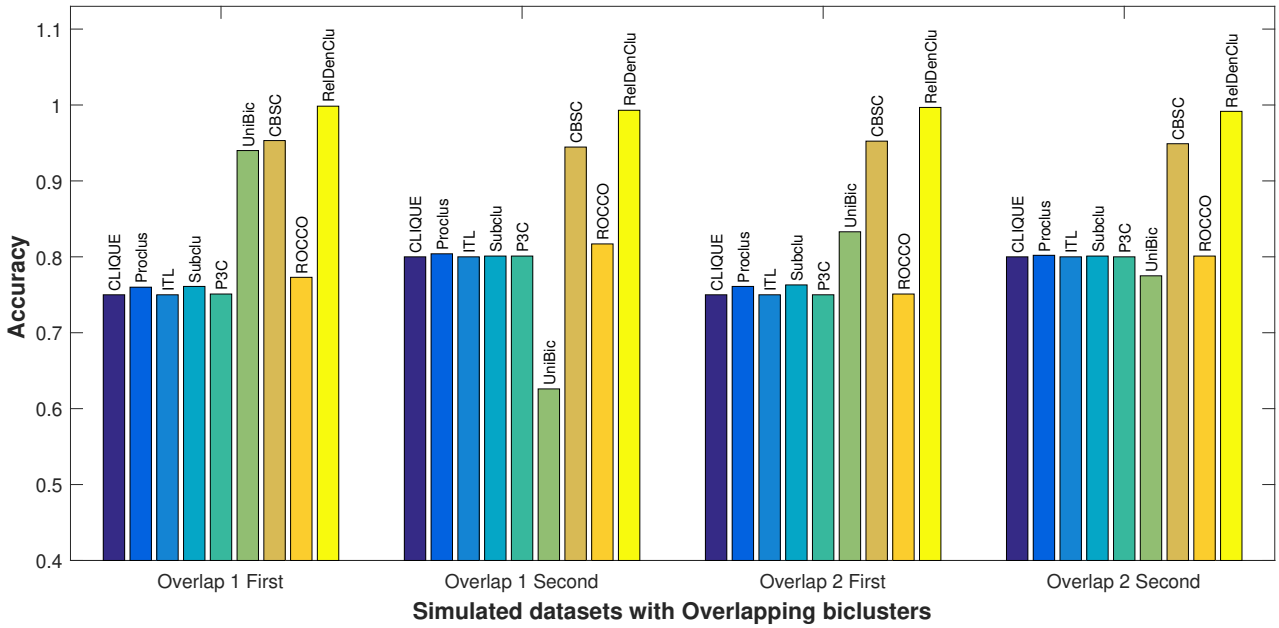
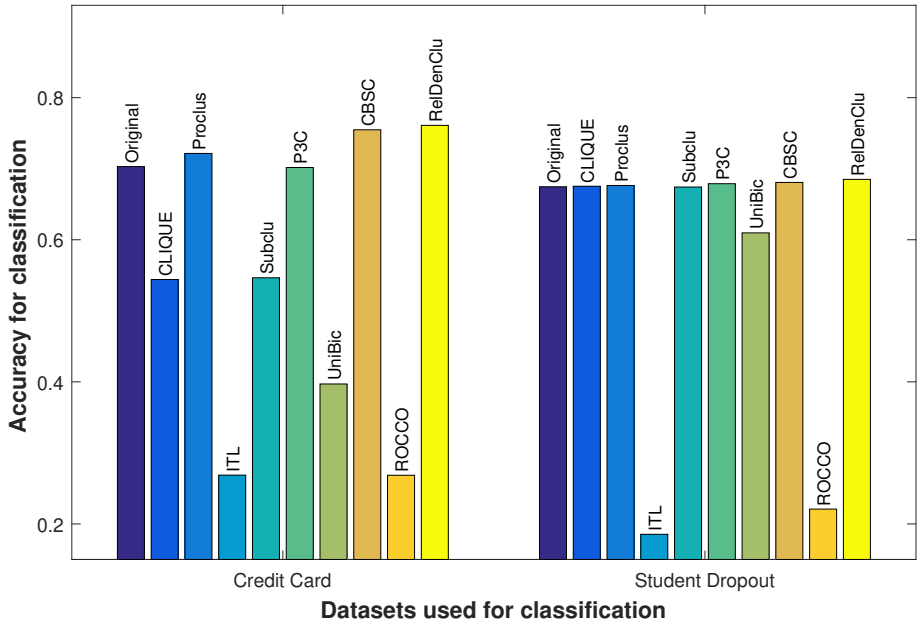
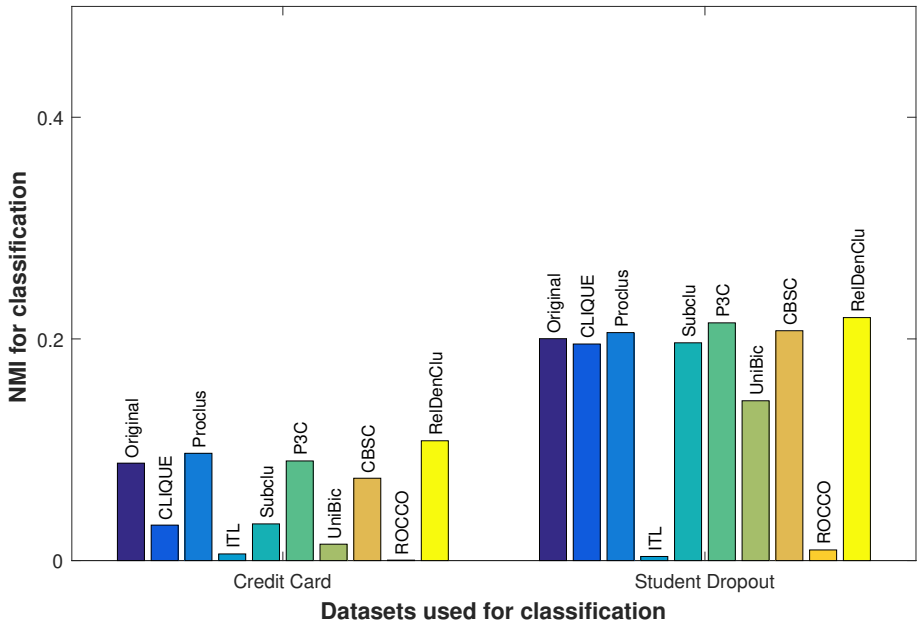


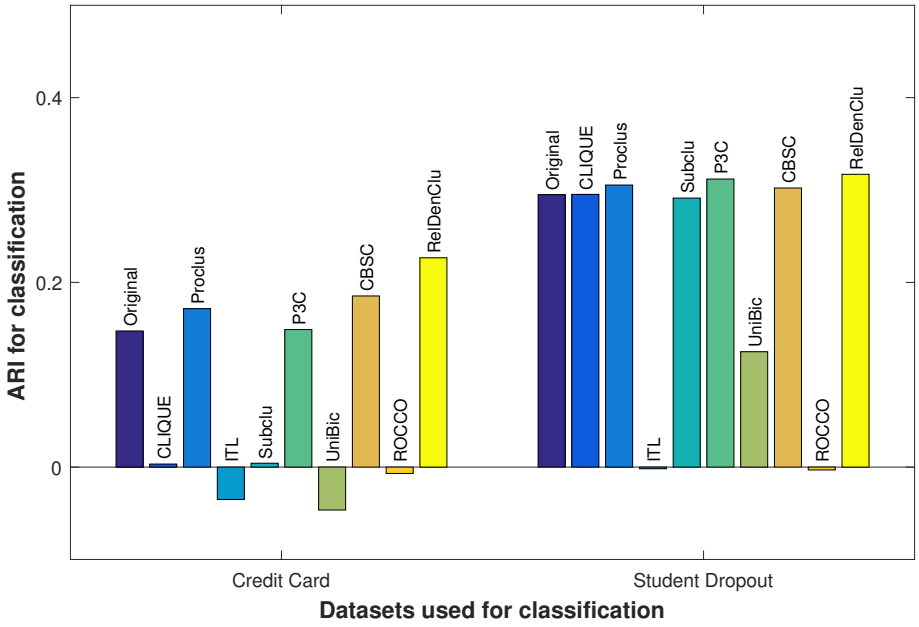
Figure 5.6: Accuracy obtained using various algorithms for Overlapping datasets



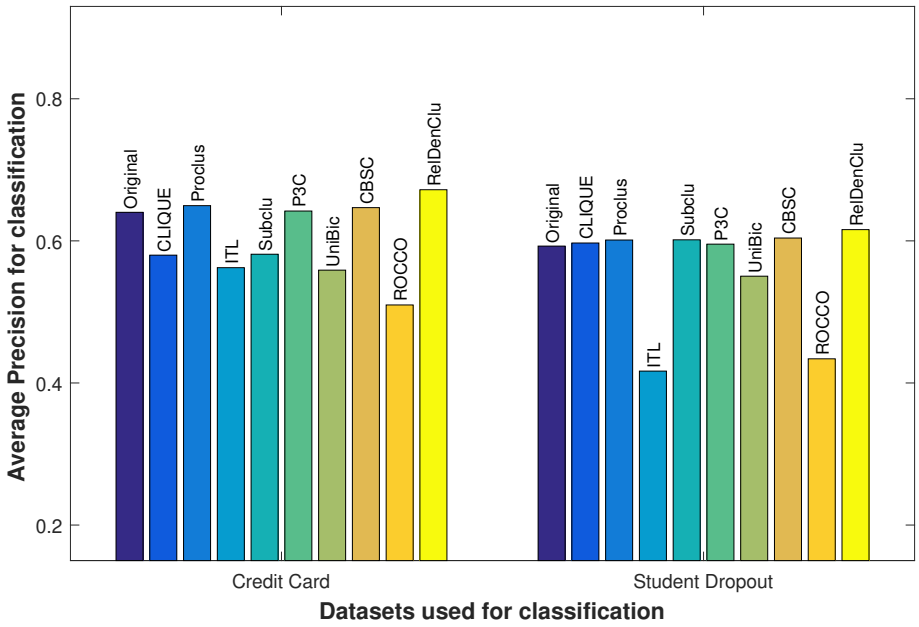
(a) Accuracy of various biclustering algorithms for classification



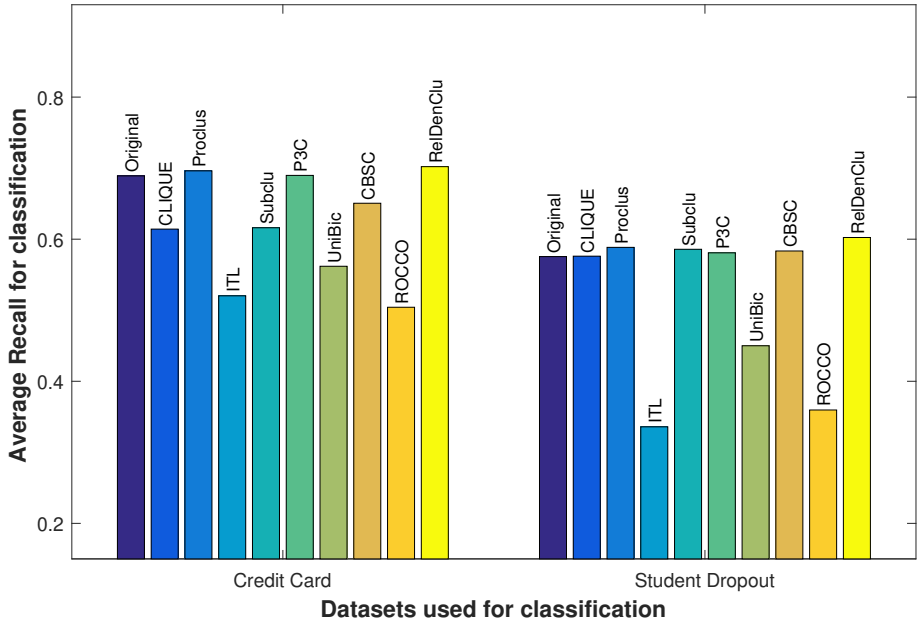
(b) NMI obtained by various algorithms for classification



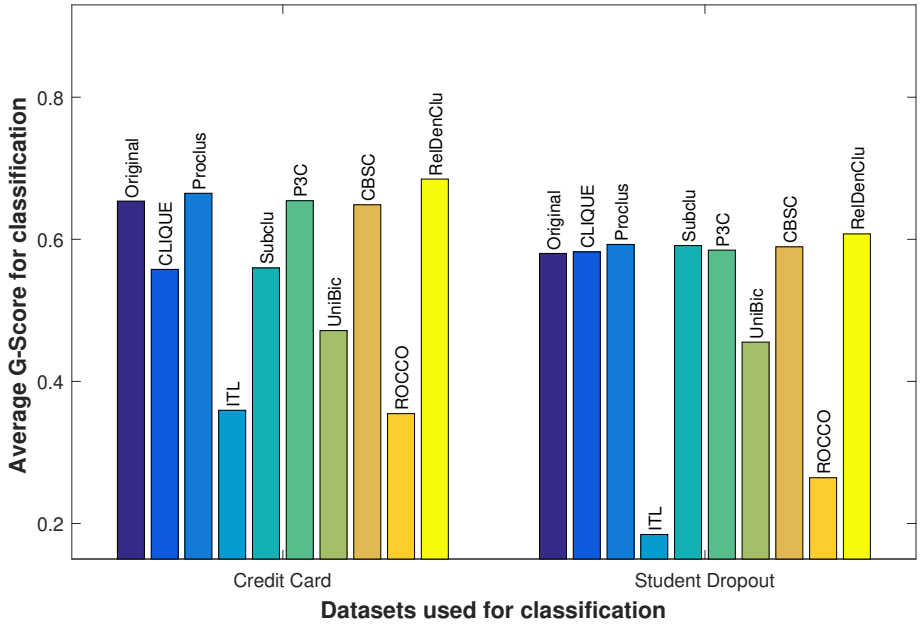
(c) ARI obtained by various biclustering methods for classification



(d) Average precision obtained by various biclustering algorithms for classification

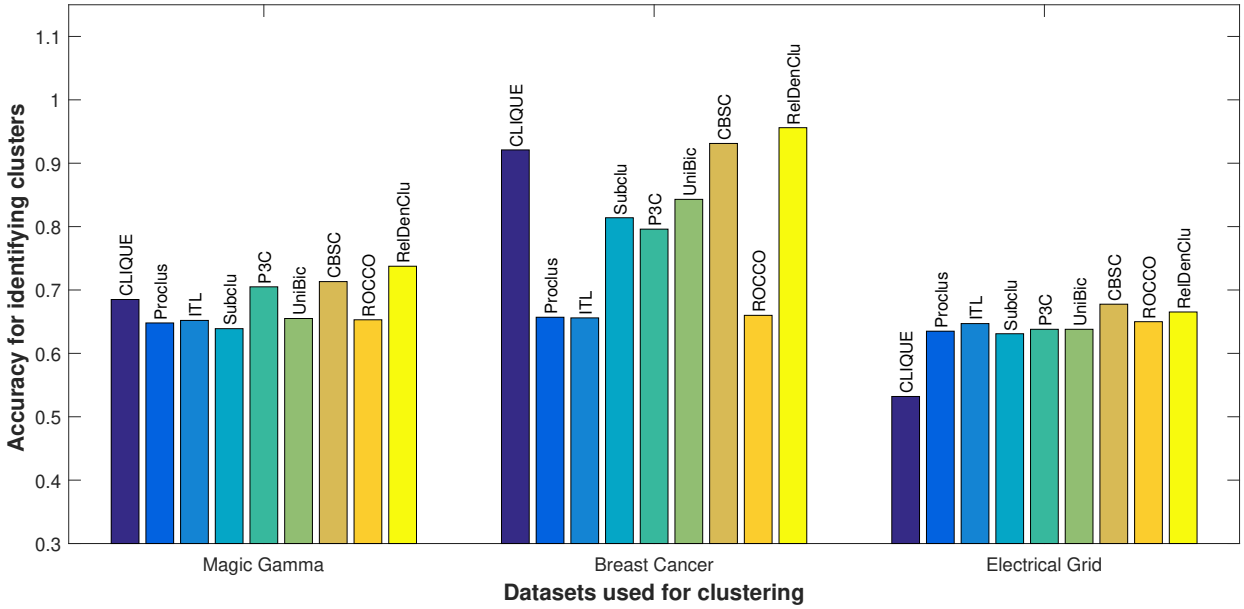


(e) Average recall obtained by various biclustering algorithms for classification

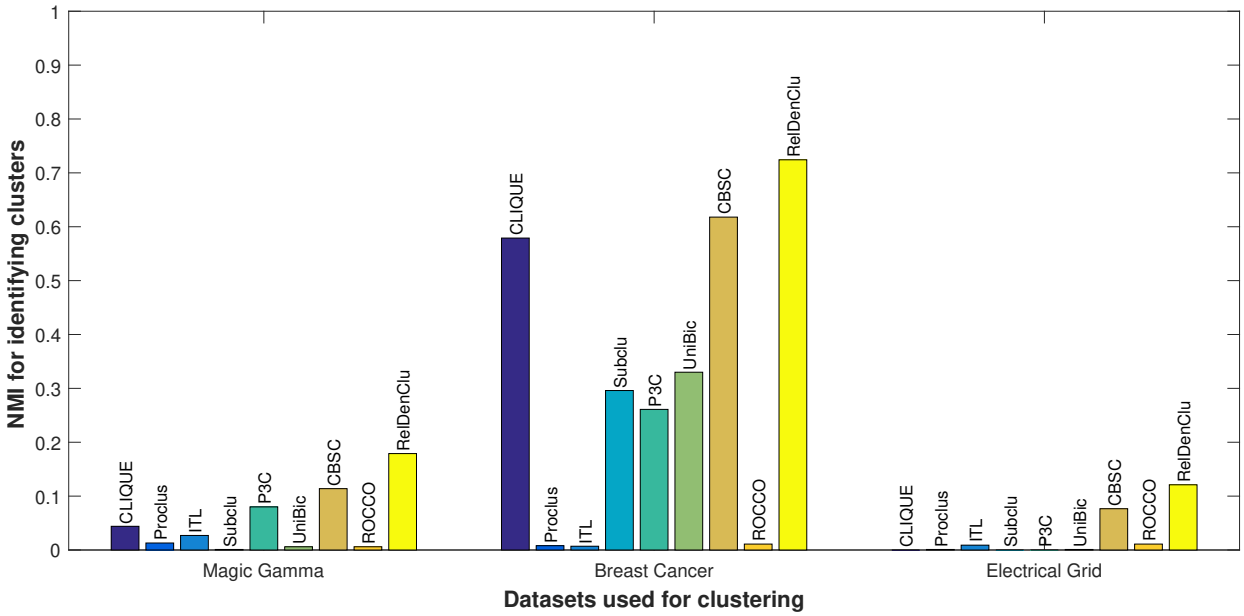


(f) Average G-Score obtained by various biclustering algorithms for classification

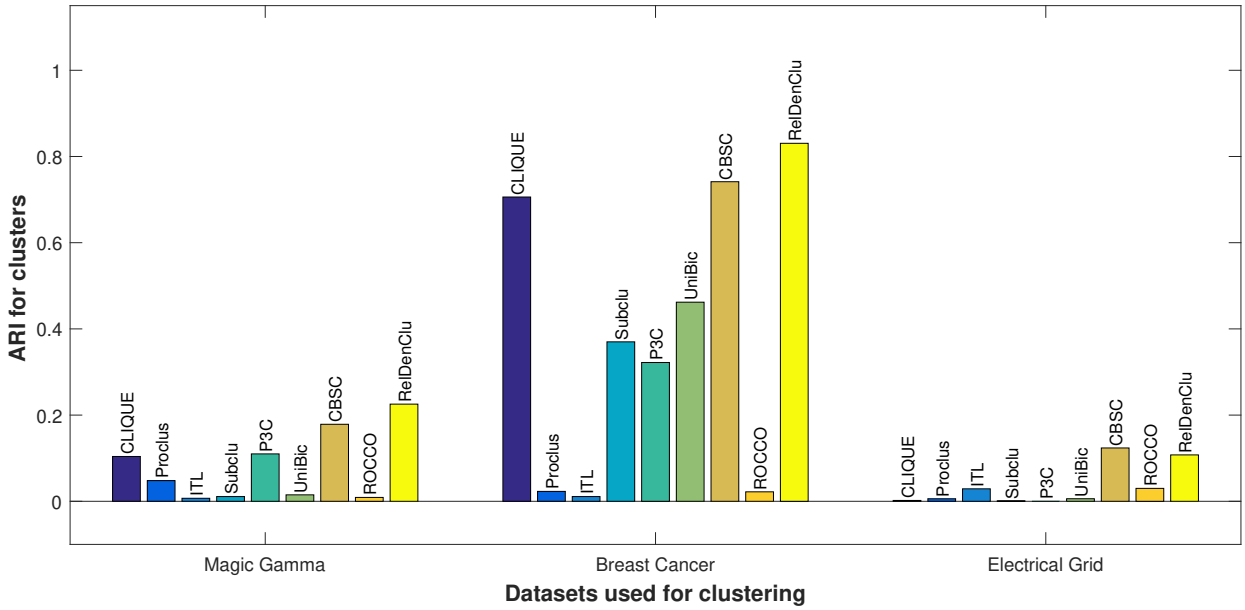
Figure 5.7: Performance of various biclustering algorithms for classification



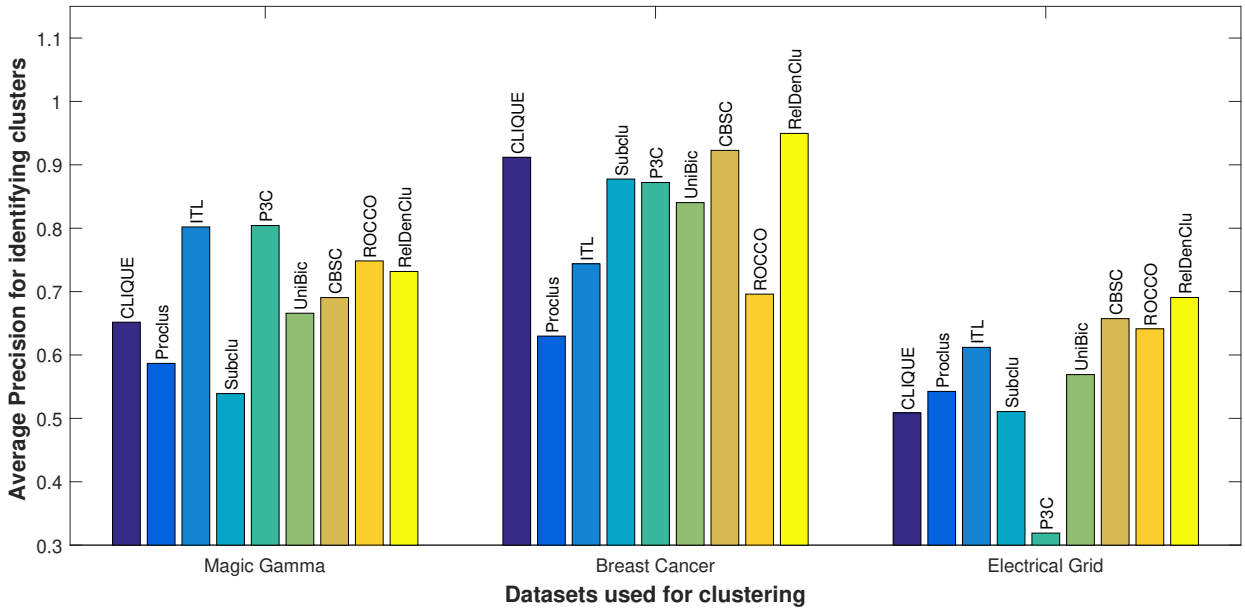
(a) Accuracy of various algorithms for clustering of UCI-ML datasets (calculated using Equation 4.1)



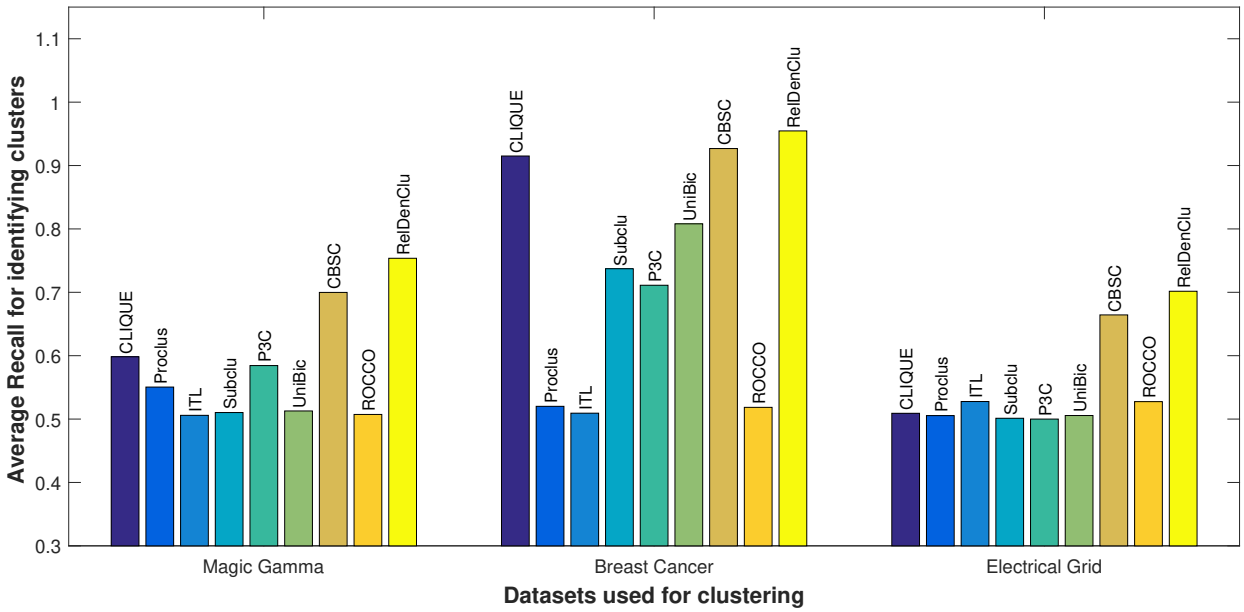
(b) NMI obtained by various algorithms for clustering of UCI-ML datasets



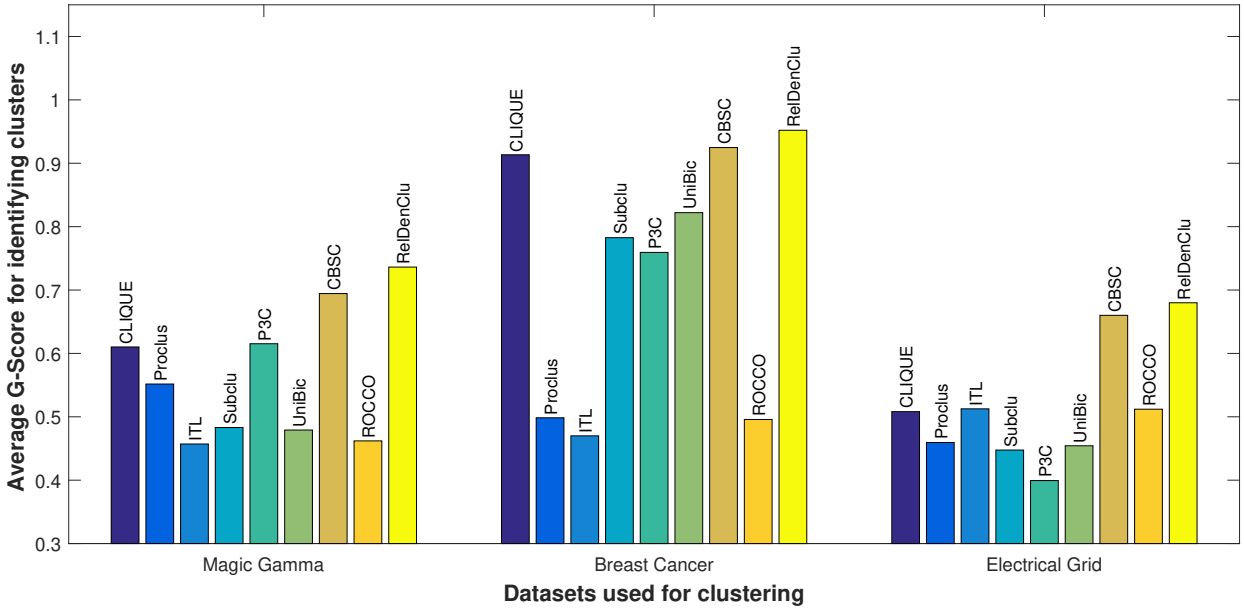
(c) ARI obtained by various algorithms for clustering of UCI-ML datasets



(d) Average precision obtained by various algorithms for clustering of UCI-ML datasets



(e) Average recall obtained by various algorithms for clustering of UCI-ML datasets



(f) Average G-Score obtained by various algorithms for clustering of UCI-ML datasets

Figure 5.8: Performance of various algorithms for clustering of UCI-ML datasets

5.5.2.2 Experimental Results with clustering of UCI-ML datasets

The results for clustering are reported in Table 5.10. RelDenClu is seen to have the best accuracy for the datasets used for the investigation. For better visualization, the accuracies obtained by different methods are presented in Figure 5.8a as bar graphs. NMI and ARI are shown in Figures 5.8b and 5.8c, respectively. The average Precision, Recall and G-score of both classes have been shown in Figures 5.8d-5.8f, respectively. We find that RelDenClu obtains better accuracy, NMI and ARI for the three datasets used for experiments, i.e., Magic Gamma, Breast cancer, and Electric Grid. It also obtains a better average G-Score compared to other methods.

5.5.3 Relevant features in COVID-19 dataset obtained using RelDenClu during early stages of the pandemic

Table 5.11 presents the 27 features used in the analysis [193]. A checkmark in the corresponding columns denotes the features identified by UniBic, CBSC, ROCCO, and RelDenClu, based on data from January 2020. The table also reports the values of Spearman correlation (ρ_s) between the chosen feature and the infection rate of COVID-19 for various countries for 31 January 2020 (denoted as ρ_s(Jan)) and 31 December 2020 (denoted as ρ_s(Dec)). The features are listed in decreasing order of absolute value of ρ_s(Dec).

RelDenClu selected 16 of 27 WDI features. We find that 12 out of 16 features selected by RelDenClu have a Spearman correlation (absolute value) greater than 0.5 with the disease occurrence rate on 31 December 2020. This suggests that it may have recognized the significance of these features earlier, for example, by using January's data, a period when the Spearman correlation failed to indicate this relationship. Within the features excluded from the bicluster, only 2 of the 11 show a Spearman

correlation (in terms of absolute value) exceeding 0.5 with disease occurrence rates on 31 December 2020. The reason for using Spearman's correlation to evaluate the features has been discussed in Section 4.8.4 in Chapter 4.

The above-mentioned results suggest that RelDenClu could be explored to identify important factors that affect a new disease. This would help combat the pandemic situation in countries with a more scientific approach. Some of these identified features for COVID-19 have been previously examined and deemed significant. These are detailed in the following paragraphs, where we cite the features using their serial numbers (S. No.) from Table 5.11.

Among the features selected by RelDenClu, the percentage of deaths in the region due to communicable and non-communicable diseases (S. Nos. 2 and 3 of Table 5.11) concurs with the findings of Chakrabarti et al. [200]. The features showing the life expectancy and mortality rate for the overall population and for each gender affect the age distribution of people in a region (S. Nos. 4,5,7,11,13,17 of Table 5.11) are selected by RelDenClu. The feature "People using at least basic sanitation services" (S. No. 6 of Table 5.11) is linked to the transmission of communicable diseases and related immunity. The feature GDP per capita (S. No. 8) is also known to be associated with basic sanitation services and life expectancy, thus having association with COVID-19 infection rates. The features "Tuberculosis detection rate" and "Incidence of tuberculosis" (S. Nos. 9,14 of Table 5.11), are also selected by RelDenClu. The feature "Urban Population" (S. No. 12 of Table 5.11) is seen to be important and could be related to the difference in facilities such as sanitation in cities.

As mentioned earlier, all features selected by RelDenClu, except four (S.Nos. 17, 22, 24 and 25 of Table 5.11) have a high Spearman correlation with COVID infection rates on 31 December 2020, although the biclusters are obtained using the least amount of data from 31 January 2020.

Table 5.11: Features (WDI dataset) selected by biclusters based on countries having high COVID-19 occurrence (90 percentile) in January 2020

S. No.	Feature description	ρ_s(Jan)	ρ_s(Dec)	UniBic	CBSC	ROCCO	RelDenClu
1	Current health expenditure per capita, PPP (current international $)	0.22	0.69	✓	-	-	-
2	Cause of death, by communicable diseases and maternal, prenatal and nutrition conditions (% of total)	-0.09	-0.68	✓	✓	-	✓
3	Cause of death, by non-communicable diseases (% of total)	0.12	0.67	✓	✓	-	✓
4	Mortality rate, adult, female (per 1,000 female adults)	-0.27	-0.67	-	✓	-	✓
5	Survival to age 65, female (% of cohort)	0.27	0.67	✓	✓	-	✓
6	People using at least basic sanitation services (% of population)	0.23	0.64	✓	✓	-	✓
7	Life expectancy at birth, total (years)	0.26	0.64	✓	-	-	✓
8	GDP per capita, PPP (current international $)	0.22	0.62	✓	✓	-	✓
9	Tuberculosis case detection rate (%, all forms)	0.13	0.61	-	-	-	✓
10	Population ages 65 and above (% of total)	0.25	0.59	✓	-	-	-
11	Survival to age 65, male (% of cohort)	0.24	0.58	✓	✓	-	✓
12	Urban population (% of total)	0.12	0.58	-	-	-	✓
13	Mortality rate, adult, male (per 1,000 male adults)	-0.23	-0.57	-	✓	-	✓
14	Incidence of tuberculosis (per 100,000 people)	-0.05	-0.51	✓	✓	-	✓
15	Population ages 15-64 (% of total)	0.07	0.46	✓	-	-	-
16	International tourism, number of arrivals	0.4	0.44	✓	-	-	-
17	Mortality from CVD, cancer, diabetes or CRD between exact ages 30 and 70 (%)	-0.23	-0.4	-	✓	-	✓
18	PM2.5 air pollution, population exposed to levels exceeding WHO guideline value (% of total)	-0.23	-0.38	-	-	✓	-
19	Air transport, passengers carried	0.44	0.37	-	-	-	-
20	International migrant stock, total	0.35	0.27	-	-	-	-
21	Trade (% of GDP)	0.03	0.26	-	-	-	-
22	Out-of-pocket expenditure (% of current health expenditure)	-0.09	-0.25	-	✓	-	✓
23	Labor force participation rate, total (% of total population ages 15+) (modeled ILO estimate)	0.04	-0.24	-	-	-	-
24	Population density (people per sq. km of land area)	0.23	0.16	✓	-	-	✓
25	Population, total	0.34	-0.16	✓	-	-	✓
26	Tuberculosis treatment success rate (% of new cases)	-0.08	-0.15	✓	-	-	-
27	Death rate, crude (per 1,000 people)	0.02	0.07	✓	-	-	-

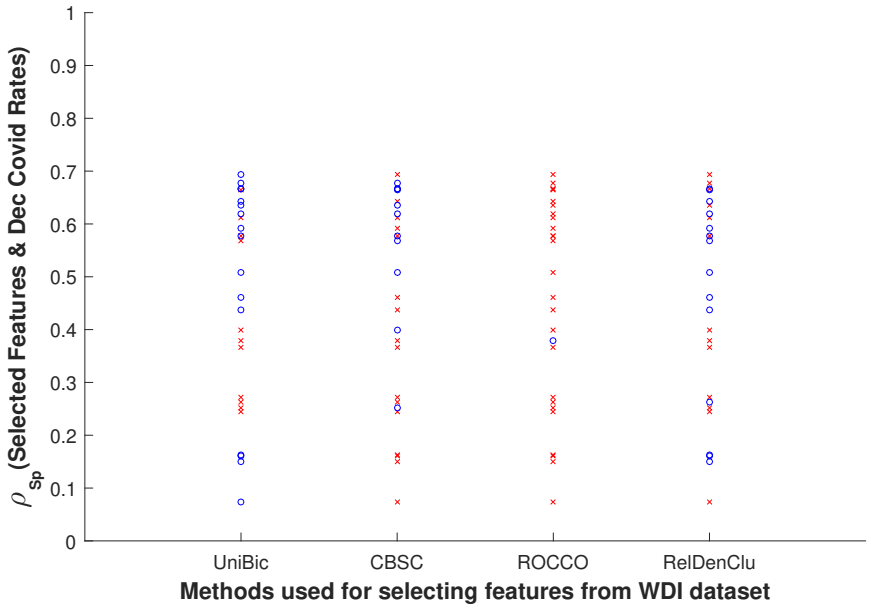


Figure 5.9: Spearman correlation (absolute value) between Dec 31 2020 COVID infection rates and features selected from WDI database by biclustering methods. Selected features are marked in blue and rejected in red

Performance comparison between RelDenClu and other methods is shown in Figure 5.9, which shows a scatter plot for the Spearman correlation (absolute values). For each method, selected features are marked in blue and rejected features in red. For RelDenClu, most of the selected features are those that have a higher correlation.

In conclusion, most of the features selected by RelDenClu demonstrate relevance and the methodology presented enables the investigation of previously unidentified associations. Following the identification of features from larger datasets, a more in-depth analysis could be conducted on individual factors.

5.5.4 Results of analysis on COVID-19 and vaccine dataset

When applied to the vaccine datasets described in Section 5.4.5, RelDenClu selects the percentage of newborns vaccinated with Tetanus-Toxoid. A study has already noted that a dose of the Tetanus-Toxoid vaccine can result in a reduction in the

severity of COVID-19 symptoms, based on similarities between the COVID-19 spike protein and Tetanus Toxin [203]. Our experiments support this hypothesis. Analysis using UniBic, CBSC and ROCCO methods also indicates that the coverage of the Tetanus Toxoid vaccine may be related to the infection rate of COVID-19.

5.6 Limitations of RelDenclu

In this chapter, we saw that RelDenClu achieves encouraging results when applied to various datasets. We also saw that its execution time is less than CBSC. However, RelDenClu still has a longer execution time. In the future, we would like to design algorithms that allow the use of parallel computing for this method. Furthermore, we note that RelDenClu calculates the density estimation parameters from the data but requires several other parameters to identify the biclusters. This makes the use of RelDenClu difficult for the end user.

5.7 Summary

In this chapter, we presented an algorithm named RelDenClu that finds biclusters based on non-linear relations between features. By analyzing local variations in density, the algorithm is seen to perform well on non-linear datasets. RelDenClu has been used to improve the classification performance of the credit card dataset. RelDenClu is seen to be effective in discovering existing classes for three real-world datasets of the UCI-ML repository, namely Magic Gamma, Breast Cancer, and Electric Grid. Furthermore, RelDenClu has been applied to a dataset that incorporates infection rates for COVID-19 in different regions and lifestyle indicators in these regions. From this analysis, it was able to discover lifestyle factors that are likely to affect COVID-19

infection rates. RelDenClu was also used to identify a negative association between Tetanus-Toxoid coverage and COVID-19 infection rates, which was supported by the existing literature. Finally, RelDenClu facilitates us to understand the relationship between subsets of a dataset which is otherwise obscured by unrelated subsets of observations or overlooked due to non-linearity.

Chapter 6

PF-RelDenBi: A Parameter free Relative Density based Biclustering Method for identifying non-linear feature relations

6.1 Introduction

Biclustering finds hidden patterns in the data by searching for paired subsets of features and observations. Each of these subsets is based on spatial proximity in space defined by selected features or similarity between features for selected observations. Most existing algorithms generally focus on the former, that is, finding biclusters based on the closeness in the selected subspace [15]. On the other hand, relation based biclustering methods search mainly for biclusters based on linear relations between features [141]. Some algorithms [143, 145] exist for finding non-linear relation based biclusters, where they search for biclusters with some constraints. For example,

UniBic finds biclusters based only on monotonous relations, thus UniBic will not be able to find a relation based on functions like the sine wave. As discussed in the previous chapter, CBSC does not adjust for variations in marginal distributions, limiting their ability to find various types of relation based biclusters. CBSC finds more general relations, but if marginal densities have sparse and dense regions, the actual biclusters in the data may not be obtained due to fragmentation. We had also seen in the previous chapter that RelDenClu overcomes the problem of fragmentation. However, it requires users to provide several parameter values, making it difficult to use RelDenClu for datasets from various domains. In this chapter, we try to overcome these problems.

The algorithm presented in this chapter finds biclusters based on non-linear continuous feature relations. By scrutinizing joint and marginal densities for each feature pair, we discern observations forming relationships, subsequently expanding these associations to identify dense regions of related features. Leveraging distances between features based on selected observations, the PF-RelDenBi method efficiently clusters features, resulting in the identification of biclusters possessing desirable properties such as invariance to permutations, scaling, and translation. In particular, this invariance ensures the consistency of PF-RelDenBi, regardless of changes in data representation. Additionally, it has been crafted to be entirely parameter-free, alleviating the user's burden by requiring the user to provide only the dataset for analysis. Since the thresholds used in the algorithm are calculated automatically based on the dataset itself, PF-RelDenBi is not dataset dependent. This makes PF-RelDenBi versatile enough to be applied to datasets in different domains.

Beyond its parameter-free nature, PF-RelDenBi distinguishes itself by incorporating a non-linear feature dependence index named MIDI for clustering features that share common dense regions. Recall that MIDI is the mutual information-based

dependence index presented in Chapter 3. The combination of parameter-free operation and the use of MIDI not only enhances the efficiency of the method, but also underlines its adaptability and robust performance in diverse datasets. PF-RelDenBi, therefore, stands as a pioneering advancement in biclustering methodologies, providing a powerful, accessible, and efficient tool for uncovering meaningful patterns in complex datasets.

6.2 Contributions

PF-RelDenBi finds biclusters based on continuous non-linear relationships using marginal and joint densities and information derived from two-dimensional spaces corresponding to each pair of features. The advantages of this approach are listed below.

1. The users do not need to provide any parameter values.

2. The algorithm does not assume that the relationship between features has a particular form such as linear or multiplicative. Thus, PF-RelDenBi finds biclusters with both linear and non-linear relations among features.

3. The algorithm does not assume that the relationship between features is monotonous. In other words, biclusters based on both monotonous and non-monotonous relations in the data can be found. For this, it uses a mutual information-based feature relation index named MIDI.

4. It avoids the fragmentation of biclusters by adapting to variations in marginal densities.

Briefly speaking, the main aim of PF-RelDenBi is to find relation based biclusters. This includes relations which are linear, non-linear, and non-monotonous. Finding the latter two is more challenging. PF-RelDenBi can find biclusters having feature relations even if they are non-linear or non-monotonous. In addition, the feature relations in the biclusters obtained by PF-RelDenBi are quantified by a feature relation index named MIDI.

6.3 Objective of PF-RelDenBi

In this section, we present the objective of PF-RelDenBi and provide relevant definitions. The biclusters obtained by PF-RelDenBi are submatrices in which features are directly or indirectly related. Let the dataset be denoted by a matrix D containing N rows each corresponding to observation and M columns each corresponding to a feature. In this chapter, a direct relationship between two columns of a matrix means the following: there is dependence between the features corresponding to these two columns when the subset of observations contained in the bicluster is used as the base. Here, dependence is estimated using the MIDI that was discussed in Chapter 3. Two columns are connected indirectly if there exists a chain of columns connecting them such that each consecutive pair of columns in this chain is directly connected. This has been stated in the following definitions.

Definition 6.1. *For a matrix A let the i^{th} column be denoted by $A_{*,i}$. In this chapter, we call the i^{th} and j^{th} columns of a matrix directly connected if there is a dependence between these two columns. Perfect dependence occurs when $A_{*,i}$ can be used to determine $A_{*,j}$, or vice-versa.*

Let us assume each direct relationship between features to be an edge connecting

two vertices, each representing a column. Two columns are said to be indirectly connected if they are connected by a path. Note that a direct or indirect connection between the columns exists only on the base of rows contained in the submatrix.

Definition 6.2. *We call the i^{th} and the j^{th} columns of a matrix A to be connected if these two columns are directly connected or there exist columns $A_{*,r_1}, \cdots, A_{*,r_k}$, such that (1) each column in this list is directly connected to the consecutive column in the list (e.g. A_{*,r_2} is directly connected to A_{*,r_3}), (2) $A_{*,i}$ is directly connected to A_{*,r_1}, and (3) A_{*,r_k} is directly connected to $A_{*,j}$.*

Using the definition of connected features, we now define a bicluster as follows:

Definition 6.3. *A bicluster in the data matrix D is a pair of two sets given by $O = \{oi_1, oi_2, \cdots, oi_n\}$ and $F = \{fi_1, fi_2, \cdots, fi_m\}$. The matrix D, restricted to observations in O and features in F gives us a submatrix A corresponding to the bicluster $< O, F >$. More specifically, the submatrix corresponding to $< O, F >$ is given by A, where the q^{th} element of the p^{th} row is given by $A[p, q] = D[oi_p, fi_q]$. A bicluster is said to be relation based if all the columns of the corresponding submatrix are connected, either directly or indirectly.*

It may be noted that PF-RelDenBi finds biclusters based on the relation between feature pairs and so it can result in biclusters with disconnected regions. Thus, PF-RelDenBi can be seen as grouping related features based on observations.

6.4 Biclustering Method: PF-RelDenBi

We now present the procedure for finding dense sets for each pair of features and then combine them to obtain biclusters.

6.4.1 Finding the set of observations having dependence for a given feature pair

This section describes a method for identifying subsets of observations which show dependence between two features, by comparing joint distribution and marginal distributions using histograms. After normalising the data to the range $[0, 1]$, we take $3\log(N)$ equal intervals along each axis, where N is the number of observations in the dataset.

Since the data are in the range $[0, 1]$, taking $3\log(N)$ cells along each axis in a two-dimensional space leads to the cell area $A = 1/(3\log(N))^2$. This leads to consistent density estimates [132] as we know that $\lim_{N\to\infty} A = \lim_{N\to\infty} 1/(3\log(N))^2 = 0$ and the product of the cell area and number of observations given by $\lim_{N\to\infty} NA = \lim_{N\to\infty} N/(3\log(N))^2 = \infty$. We have taken $3\log(N)$ instead of $\log(N)$ so that there is a sufficient number of cells even when N is small. Using this scheme, the joint density and marginal density of each cell are estimated. Of course, any histogram scheme that leads to consistent density estimates can be used here. However, we chose to use the simple scheme with $3\log(N)$ cells, since it leads to shorter execution times.

Thus, each cell obtained using the above-mentioned scheme can be represented as a region $I_x \times I_y$, where I_x and I_y are intervals of length $1/(3\log(N))$, along X and Y axes, respectively. The regions $I_x \times [0, 1]$ and $[0, 1] \times I_y$ can be used to calculate the marginal densities of the cell $I_x \times I_y$. We call these regions the marginal cells for convenience. We then decide whether a cell is dense if its density is greater than the densities of marginal cells as well as the average density of the entire space. The thought behind this step is that a higher cell density compared to marginal density implies a higher conditional probability of a point lying in a particular cell within a

given marginal cell.

Two dense regions are merged if they are adjacent to each other horizontally, vertically, or diagonally. A dense region can have at most eight dense regions connected to it. After finding dense regions in two-dimensional space, noise is removed by using the overlap between three two-dimensional spaces for every set of three features. Thus, redundancy introduced by using three features is used to weed out the noise. Up to this point, the procedure for finding the dense cells used by RelDenClu and PF-RelDenBi is the same. But in the next step, RelDenClu needs user-defined parameters to prune the noise, while PF-RelDenBi calculates the thresholds required for pruning automatically. A detailed description of this step is given below.

6.4.2 Finding the pruned set of observations for biclusters

To identify the set of observations related to each other, we find relative density-based subsets in the two-dimensional Euclidean space given by each pair of features. However, because the background noise can be distributed in arbitrary ways, this procedure of finding relations can result in a lot of noise. Therefore, to weed out noise, we search for a set of three features for which a given set of observations forms dense regions for each pair of features belonging to the set. Thus, for a set of three features $\{f_1, f_2, f_3\}$, we find dense regions for pairs $< f_1, f_2 >$, $< f_2, f_3 >$, and $< f_3, f_1 >$. The intersection of observations forming dense regions for these three pairs is found. This observation set is retained only if it contains a significant number of observations. Let us call this significant number $ObsMin$. The value of $ObsMin$ is calculated using the dense regions found earlier in Section 6.4.1. The method of calculating $ObsMin$ is discussed in the following paragraphs.

In Section 6.4.1, we calculated dense regions for each pair of features. A dataset

having M features will have $M(M-1)/2$ feature pairs. The number of points lying in dense regions, for each feature pair, may vary between 0 and N (number of observations in the dataset). Thus, $M(M-1)/2$ values (number of observations lying in dense regions for each feature pair) are distributed over the range $[0, N]$. This distribution can give us an idea about the size of biclusters in the dataset. This is because the distribution will tend to peak around the actual size of the biclusters since there will be many dense regions of this size. This distribution can be examined using histograms. As discussed above, the number or size of histogram bins plays an important role. Since $M(M-1)/2$ values are distributed over $[0, N]$, according to the discussion in Section 6.4.1, we can divide this range to $3log_2(M(M-1)/2)$ bins to obtain consistent density estimates.

Since many of the feature pairs will have no dense regions or might have a very small number of points in dense regions caused by noise, we will ignore the first bin of the histogram. We then find the first bin b_{lim} such that the next two consecutive bins are populated (i.e. one or more feature pairs have dense regions such that 'the number of observations in the regions' lies in the bin range). The lower edge of b_{lim} denoted by $lEdgeB$, can be taken as the lower limit of the minimum number of observations in any bicluster. The reason for considering two consecutive bins is that we want to identify the place where the distribution starts to increase.

For each set of three features, we can now retain the observation sets containing more than $ObsMin$ observations and discard the rest. We refer to the retained observation sets as the "Triplet observation sets" for convenience. Each of these sets is now represented as a binary array. The presence of observation is indicated by 1 and the absence by 0. However, many of the sets may be very similar to each other. To combine these similar sets, we perform single-linkage clustering on the binary arrays that represent the observation sets. Cosine similarity is used for this clustering. Thus,

we obtain several clusters of sets of observations.

The hierarchical clustering mentioned above requires a parameter that indicates the maximum number of clusters to be found by the algorithm. Let us say that PF-RelDenBi sets this value automatically at $NclusObsMax$. We will now discuss how the value of $NclusObsMax$ is calculated.

The maximum number of clusters that any clustering algorithm can find should be greater than 2. Thus, $NclusObsMax$ should be at least 3. Furthermore, we note that if $ObsMin$ (minimum number of observations in "triplet observation sets") were larger, we would have a greater variety of observation sets. This is because a larger value of $ObsMin$ leads to a greater number of possible combinations.

On the other hand, if the sum of the number of rows in different biclusters in a dataset is fixed, we tend to have a greater number of "Triplet observation sets" if the number of biclusters is less and vice versa. This can be understood as follows. Suppose a dataset has i biclusters with $m_1, m_2, \cdots, m_i$ rows each while another dataset has a single bicluster having m rows where $m = \sum_{k=1}^{i} m_k$. Let us assume that all rows are connected. We know $m^3 \geq \sum_{k=1}^{i} m_k^3$. So, the dataset with a smaller number of biclusters will have a larger number of "Triplet observation sets".

Thus, we want $NclusObsMax$ to increase with $ObsMin$ but decrease with the number of "Triplet observation sets" denoted by $NtripleSets$. So we set $NclusObsMax = ceil(3 + ObsMin/NtripleSets)$, where the function $ceil(x)$ gives the smallest integer greater than or equal to x.

Thus, using hierarchical clustering, we may obtain many clusters (less than or equal to $NclusObsMax$) of "Triplet observation sets". From each of these clusters, we obtain a single set of observations by finding the observations that appear in many "Triplet observation sets" lying in the cluster. We say that observations are located in many "Triplet observation sets" if they are located in more than $1/NclusObsMax$

of all the "Triplet observation sets" that form the cluster. This is because, a smaller number of biclusters means a greater number of "Triplet observation sets" in each, thus even if a smaller proportion of the "Triplet observation sets" contain an observation, we still have sufficient redundancy. At this stage, only the set of observations having more than $ObsMin$ elements should be retained, and the rest should be ignored. Note that the same observation may be present in several clusters.

6.4.3 Finding feature sets for each bicluster

In the previous step, we found several sets of observations. We now need to find the corresponding feature sets. For each set of observations, we perform single-linkage clustering and choose the largest feature cluster. The maximum number of clusters is taken to be $M/2$, where M is the number of features in the dataset. Note that this is an upper limit and a smaller number of feature clusters can be obtained, thus a high value for the upper limit is chosen.

The distance between features f_1 and f_2 is taken as $1 - MIDI(f_1, f_2)$, where MIDI is a feature dependence index [28]. Recall that MIDI is the Mutual Information based Dependence Index presented in Chapter 3 that uses normalized mutual information as a feature dependence. In Chapter 3, we saw that MIDI is capable of evaluating non-linear and even non-monotonous relations between features. It was also found to have a small execution time in comparison to other non-linear relation indices like MIDI. Therefore, we used MIDI to find the distance between the features for the selected set of observations. Figure 6.1 shows the general outline of how PF-RelDenBi works.

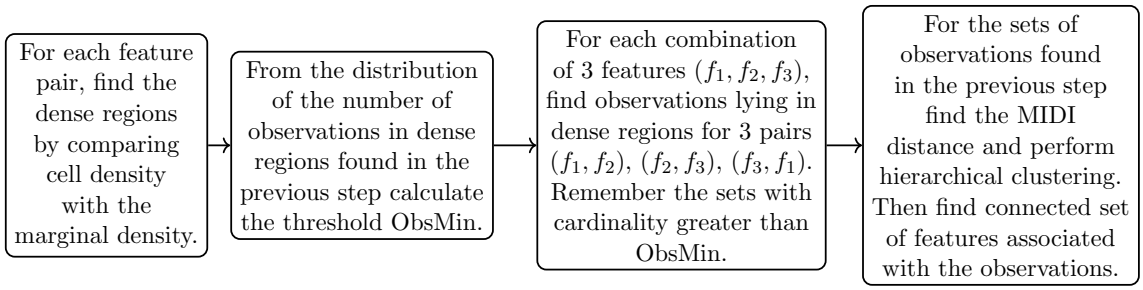


Figure 6.1: Broad outline showing how PF-RelDenBi works

Algorithm 6.1: PF_RelDenseRegions()

```
/* Finding dense regions in 2-dim space                    */
/* N is the total number of observations                   */
```

Find histogram cells, where the number of cells along each axis is given by $3\log(N)$
for *each cell in the grid* **do**
  if *density(cell) is greater than marginal densities* **then**
    Mark as dense
for *each region marked as dense* **do**
  if *At least one adjoining cell is marked dense* **then**
    Mark the observations in the cell as True
Return state of all observations

6.4.4 Algorithm

This section presents the algorithm and some details required for the implementation of PF-RelDenBi. For finding dense regions, we need to normalize the data to $[0, 1] \times [0, 1]$. To normalize data from different distributions to the range $[0, 1]$ we have applied one of the two transformations, presented as equations 3.49 and 3.50 in Chapter 3 and also discussed in Section 4.5.1.1 of Chapter 4.

The next step is to find dense regions using the procedure described in Section 6.4.1. The pseudocode is given as Algorithm 6.1. Once we have obtained dense regions in two-dimensional spaces, this information is used to obtain biclusters in higher dimensions. The details of the procedure are given in Sections 6.4.2 and 6.4.3,

Algorithm 6.2: GetBiClusters()

```
/* This algorithm discusses the process of obtaining biclusters
   from dense regions obtained by Algorithm 6.1                  */
/* M is the total number of features                             */
```
Normalise data using the procedure given in Section 6.4.4
```
/* Each pair of dimensions may contain several sets representing
   dense regions                                                 */
```
for *each pair of features* **do**
    Find observations forming dense regions using Algorithm 6.1

Find the number of observations in dense regions for each feature pair
```
/* There are M(M − 1)/2 values                                   */
```
Plot a histogram for these values with the number of bins given by $3\log_2(M(M-1)/2)$. Ignore first bin

Find the first bin b_{lim} such that two consecutive bins after b_{lim} are populated. $ObsMin$=lower edge of the bin b_{lim}

for *each set of 3 features* $< f_1, f_2, f_3 >$ **do**
    Find the intersection of observations forming dense regions for all 3 pairs $< f_1, f_2 >< f_2, f_3 >< f_3, f_1 >$
    if *resulting set is larger than* $ObsMin$ **then**
        Store observation set as binary array

Let the number of observation sets be $NtripleSets$
```
/* The function ceil(x) returns the smallest integer greater than
   or equal to x                                                 */
```
Perform single-linkage clustering of observation sets with the maximum number of clusters given by $NclusObsMax = ceil(3 + ObsMin/NtripleSets)$

for *each cluster obtained* **do**
    Calculate the bicluster observation set given by observations occurring in more than $1/NclusObsMax$ of the 'Triplet observation sets' in the cluster.

for *each bicluster observation set found in the previous step* **do**
    Calculate feature dependence (MIDI) using all features in the dataset and selected observations Perform single-linkage clustering using distance $1 - MIDI$ between features, and select the largest feature cluster as the feature set for the bicluster

Return biclusters obtained

and the pseudo-code is given as Algorithm 6.2.

6.4.4.1 Time complexity of PF-RelDenBi

The time complexity of ITL has been reported by Dhillon et al. [186], and Liu et al. have reported the time complexity of ARBic [146]. The time complexity of ROCCO has been reported by He and Moreira-Matias [148]. For MESBC, the most computationally expensive operation is to calculate the Singular Value Decomposition which is known to have a complexity of MN^2, where N and M, respectively, are the number of observations and the number of features. However, faster approximations have recently been designed, which may lead to faster implementations [204]. The time complexity of other methods used for comparison has been reported in Section 4.6 in Chapter 4. Here, we report the time complexity of PF-RelDenBi, where N and M, respectively, are the number of observations and the number of features in a given dataset.

The time complexity of PF-RelDenBi is $(log(N)M)^2 + NM^3$ where N and M represent the number of observations and features in the dataset, respectively. For finding dense regions, the complexity is given by $(log(N)M)^2$. The complexity is NM^3 for the pruning step because the intersection of observations is calculated for a set of 3 dimensions. However, this is the worst-case scenario, and in reality, we do not check for all feature triplets, but only find the intersections for dense regions calculated previously.

Theoretically, the time complexity of CBSC [30] and PF-RelDenBi is similar. However, the execution time for PF-RelDenBi is found to be much shorter than CBSC for the high-dimensional dataset. This happens because PF-RelDenBi does not calculate the Minimal Spanning Tree for each pair of dimensions, which is done in CBSC. The time complexity of PF-RelDenBi is similar to that of RelDenClu. However, in practical applications, PF-RelDenBi may save time, as the user does not

need to experiment with different parameter settings.

6.5 Experimental setup and methodology for analysis with PF-RelDenBi

In this section, we discuss the experimental setup and methodology employed to conduct experiments using both synthetic and real-world datasets. Real-world datasets include five UCI-ML datasets [185] and two datasets designed to understand the spread of COVID-19 disease [193, 197, 198].

6.5.1 Experimental setup for synthetic datasets

To test the efficacy of PF-RelDenBi, we performed experiments on synthetic and real-world datasets. Experiments with synthetic datasets allow us to study the behavior of PF-RelDenBi on datasets with known characteristics. The synthetic datasets used for the experiments and the methodology to evaluate them is the same as that used in the chapter 4 in Section 4.7.1.

6.5.2 Methods used for comparison with PF-RelDenBi

The performance of PF-RelDenBi is compared with eleven other algorithms namely CLIQUE [128], Proclus [127], ITL [186], Subclu [20], P3C [125], UniBic [145], CBSC [30], ROCCO [148], RelDenClu [32], MESBC [103], and ARBic [146]. CLIQUE, P3C, and Subclu are density-based methods. ITL is an information-theoretic method, while Proclus is known to provide stable results. ROCCO is a parameter-free coclustering method [148]. ARBic is a recent biclustering method for finding feature relation biclusters which is shown to have better results in comparison with several established

methods like FABIA, OPSM and UniBic. It also shows better results compared to other recent methods such as Qubic2 [26]. Comparison with UniBic, CBSC, RelDenClu and ARBic is important as they specifically aim at finding biclusters based on non-linear dependence. MESBC is a recent biclustering method that uses spectral analysis to find biclusters. The single parameter required by the user is the number of clusters in the dataset. This parameter is set to the minimum value of 2 for MSBEC as we know that each of the synthetic datasets contains a single embedded bicluster. For ARBic, the implementation provided by the author [146] is used and the default parameters are used. As in the previous chapter, for ROCCO we use the parameter value $K = 9$. For UniBic, P3C, SubClu, ProClus and CLIQUE the R packages 'runibic' and 'subspace' have been used for the experiments with default parameters [187, 188]. For ITL, the Matlab Toolbox for Biclustering Analysis has been used with default parameters [189].

In this chapter, we experiment with different parameter values for CBSC and RelDenClu. For the synthetic datasets, we report the performance of CBSC by averaging the performance obtained using different parameter values. The parameter values considered are the same as those described in Section 4.7.2.1 in Chapter 4. Similarly, for RelDenClu we have also reported the average of results obtained with different parameter values. The parameter values considered for RelDenClu are reported in Section 5.4.3.1 in Chapter 5. Data is normalized using $norm_2$ for "normal" and "Noisy Normal" datasets. For all other synthetic datasets, $norm_1$ is used. The results for synthetic datasets are given in Tables 6.1-6.4 and Figures 6.2-6.5. The execution time on synthetic datasets is reported in Tables 6.5-6.8.

6.5.3 Experimental setup for UCI-ML datasets

Like CBSC and RelDenClu, we demonstrate the use of PF-RelDenBi on UCI-ML datasets in two different ways. First, as an application to supervised learning, biclustering algorithms have been used to generate new features for two different datasets that lead to improved classification accuracy.

Second, we used PF-RelDenBi to cluster three datasets and then compared the clusters obtained with known classes in the data. The datasets used for these experiments are taken from the UCI ML repository [185] and are described in Chapter 4 in Sections 4.7.2.1 and 4.7.2.2.

As in the case of CBSC and RelDenClu, for real-world datasets, we initially apply normalization $norm_1$. If no biclusters are produced using $norm_1$, then the biclusters are found after applying $norm_2$ to the dataset.

6.5.3.1 Setup for for experiments with classification of UCI-ML datasets

PF-RelDenBi has been used to generate new binary features that are used to improve the classification performance on two UCI ML datasets [185]. Its performance is compared with eleven other methods mentioned in Section 6.5.2. Such experiments were also performed with CBSC and RelDenClu. The datasets and methodology are described in detail in Section 4.7.2.1 of Chapter 4. For the methods RelDenClu and CBSC, we needed to choose one set of parameter values for which the biclusters could be used to generate new features. The choice is made by selecting the values of the parameters using a 10-fold cross-validation from the combinations of parameters mentioned in Section 4.7.2.1 for CBSC and 5.4.3.1 for RelDenClu. All other parameters for various algorithms are the same as those discussed in Section 6.5.2. The evaluation criterion is also the same, i.e. accuracy, NMI, ARI, precision, recall, and

G-Score. The classification results are reported in Table 6.9 and Figures 6.6a-6.6f.

6.5.3.2 Setup for for experiments with clustering of UCI-ML datasets

We use PF-RelDenBi for unsupervised learning to identify whether its detected biclusters match known data classes as we did for CBSC. This was done for three UCI ML datasets mentioned in Section 4.7.2.2 of Chapter 4. The methodology for evaluating performance has also been discussed there. The methods used for comparison and the parameter values are the same as those used in Section 6.5.2. Six performance indices are reported. These indices are accuracy, NMI, ARI, precision, recall, and G-score. The results for clustering are reported in Table 6.10 and Figures 6.7a-6.7f.

6.5.4 Setup and methodology for application of PF-RelDenBi to COVID-19 dataset for finding relevant lifestyle factors

Like CBSC and RelDenClu, we have also used PF-RelDenBi to find lifestyle factors that impact COVID-19 infection rates. As mentioned in Section 4.7.3, the dataset is based on World Development Indicators and COVID-19 infection rates. As mentioned earlier, biclustering is useful in scenarios where subsets of observations and features are required together. Thus, we applied PF-RelDenBi for the analysis of lifestyle factors that impact COVID-19 rates. The parameter values used by CBSC and RelDenClu are the same as those used in Sections 4.7.3 and 5.4.4, respectively. The parameter values used by UniBic, Rocco and MESBC are the same as those used in Section 6.5.2. The rest of the methodology is the same as that used in Section 4.7.3. The results are provided in Table 6.11 and Figure 6.8.

6.5.5 Setup and methodology for application of PF-RelDenBi to identify vaccines existing before COVID-19 that might help in reducing COVID-19 infection rates

We have used PF-RelDenBi to find out if any of the existing vaccines before 2020 is likely to provide protection against COVID-19 using the same methodology as CBSC in Section 4.7.4. The parameter values used by CBSC and RelDenClu are the same as those used in Sections 4.7.4 and 5.4.5, respectively. The parameter values used by UniBic, Rocco and MESBC are the same as those used in Section 6.5.2. The results are reported in Section 6.6.4.

6.6 Results

In this section, we present the results obtained using the methodology presented in the previous section.

6.6.1 Results on synthetic datasets

Results for the experiments with synthetic datasets have been presented in this section.

6.6.1.1 Accuracy for synthetic datasets

Table 6.1: Accuracy for Non-Linear synthetic datasets (Mean and Standard Deviation for 10 datasets of each type)

Method	Accuracy	Datasets		
		Non-Monotonous	Non-Linear 1	Non-Linear 2
PF-RelDenBi	Mean	0.973	0.939	0.927
	Std. Dev.	0.025	0.034	0.005
MSBEC	Mean	0.671	0.767	0.704
	Std. Dev.	0.023	0.001	0.043
ARBic	Mean	0.76	0.848	0.867
	Std. Dev.	0.022	0.029	0.040
RelDenClu	Mean	0.786	0.912	0.913
	Std. Dev.	0.069	0.077	0.031
ROCCO	Mean	0.751	0.82	0.836
	Std. Dev.	0.020	0.015	0.020
CBSC	Mean	0.79	0.827	0.892
	Std. Dev.	0.086	0.049	0.022
UniBic	Mean	0.764	0.804	0.839
	Std. Dev.	0.004	0.142	0.107
P3C	Mean	0.75	0.765	0.764
	Std. Dev.	0.001	0.004	0.003
Subclu	Mean	0.746	0.785	0.793
	Std. Dev.	0.005	0.017	0.016
ITL	Mean	0.75	0.75	0.75
	Std. Dev.	0.009	0.004	0.002
Proclus	Mean	0.752	0.781	0.792
	Std. Dev.	0.009	0.014	0.007
CLIQUE	Mean	0.75	0.784	0.782
	Std. Dev.	0.000	0.002	0.002

Table 6.2: Accuracy for Base and transformed synthetic datasets (Mean and Standard Deviation for 10 datasets of each type)

Method	Accuracy	Datasets							
		Base	Scaled	Translated	Linear Transform	Point Proportion	Cluster Proportion	Noisy Uniform	Permutations
PF-RelDenBi	Mean	0.986	0.983	0.985	0.986	0.984	0.981	0.984	0.986
	Std. Dev.	0.036	0.036	0.038	0.037	0.026	0.026	0.032	0.036
MSBEC	Mean	0.600	0.601	0.604	0.654	0.603	0.641	0.595	0.600
	Std. Dev.	0.041	0.041	0.059	0.033	0.037	0.042	0.054	0.034
ARBic	Mean	0.968	0.973	0.857	0.633	0.968	0.964	0.874	0.867
	Std. Dev.	0.032	0.060	0.100	0.103	0.032	0.036	0.051	0.033
RelDenClu	Mean	0.980	0.980	0.981	0.981	0.980	0.980	0.974	0.980
	Std. Dev.	0.057	0.057	0.073	0.057	0.042	0.051	0.058	0.057
ROCCO	Mean	0.813	0.881	0.883	0.851	0.881	0.829	0.839	0.819
	Std. Dev.	0.019	0.015	0.014	0.020	0.034	0.036	0.012	0.019
CBSC	Mean	0.914	0.914	0.913	0.914	0.915	0.924	0.906	0.913
	Std. Dev.	0.063	0.063	0.063	0.062	0.068	0.070	0.049	0.063
UniBic	Mean	0.900	0.848	0.802	0.793	0.603	0.641	0.832	0.900
	Std. Dev.	0.034	0.211	0.159	0.034	0.046	0.078	0.124	0.034
P3C	Mean	0.781	0.782	0.779	0.779	0.787	0.802	0.772	0.783
	Std. Dev.	0.040	0.040	0.041	0.041	0.022	0.092	0.032	0.015
Subclu	Mean	0.794	0.759	0.788	0.765	0.802	0.724	0.795	0.798
	Std. Dev.	0.023	0.016	0.019	0.025	0.030	0.020	0.017	0.022
ITL	Mean	0.751	0.752	0.750	0.751	0.750	0.750	0.750	0.750
	Std. Dev.	0.005	0.004	0.004	0.001	0.001	0.002	0.002	0.001
Proclus	Mean	0.785	0.768	0.803	0.770	0.800	0.711	0.792	0.785
	Std. Dev.	0.013	0.015	0.023	0.028	0.024	0.024	0.010	0.015
CLIQUE	Mean	0.776	0.776	0.776	0.776	0.776	0.683	0.720	0.773
	Std. Dev.	0.016	0.016	0.016	0.016	0.016	0.012	0.011	0.015

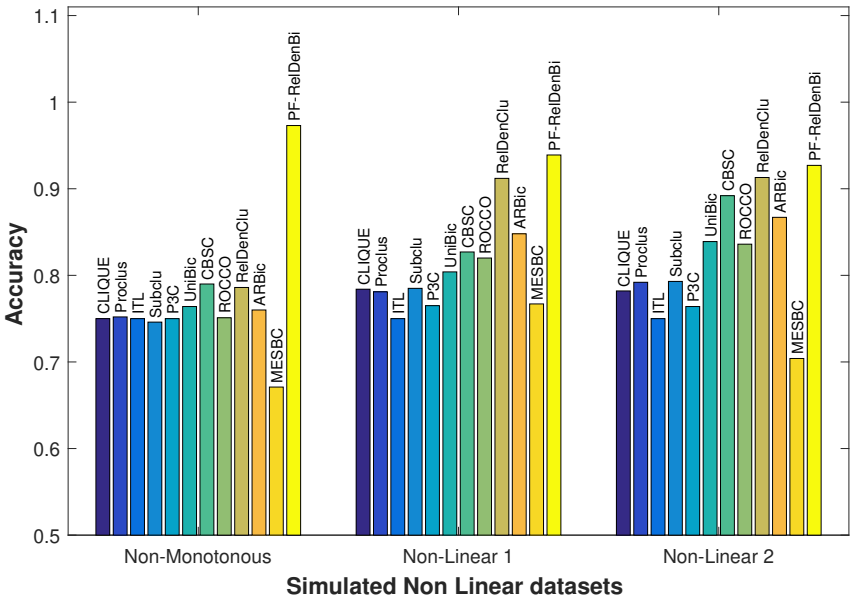


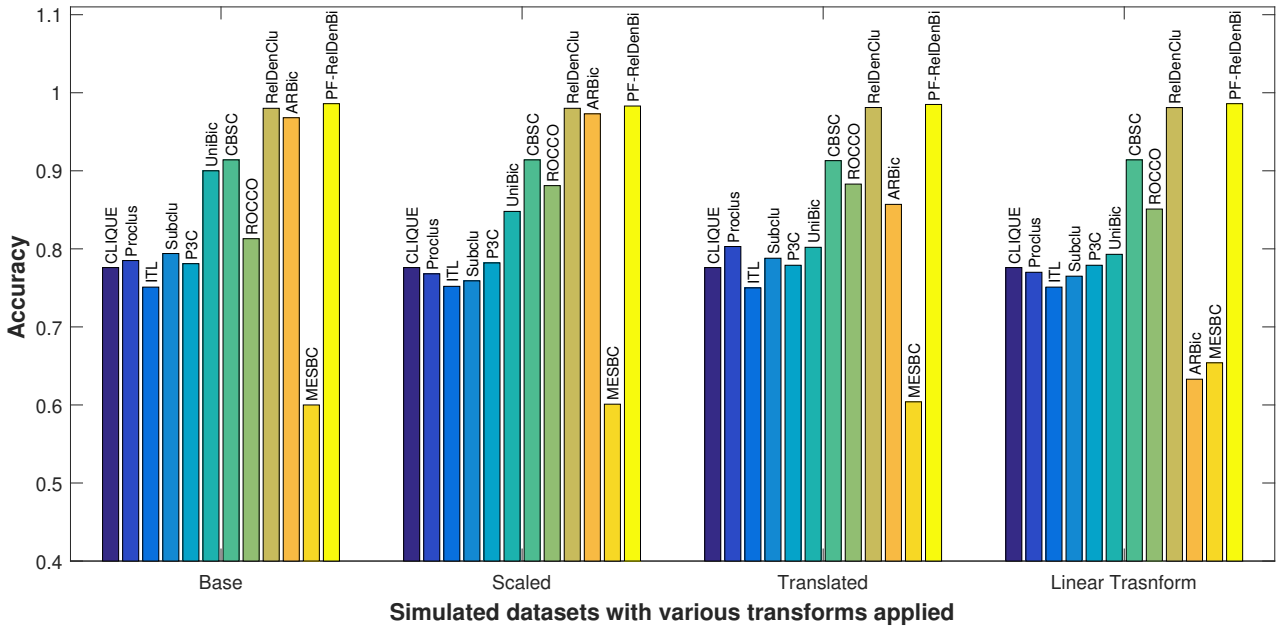
Figure 6.2: Accuracy obtained using various algorithms for Non-Linear datasets

Table 6.3: Accuracy for synthetic datasets with Normal Distribution (Mean and Standard Deviation for 10 datasets of each type)

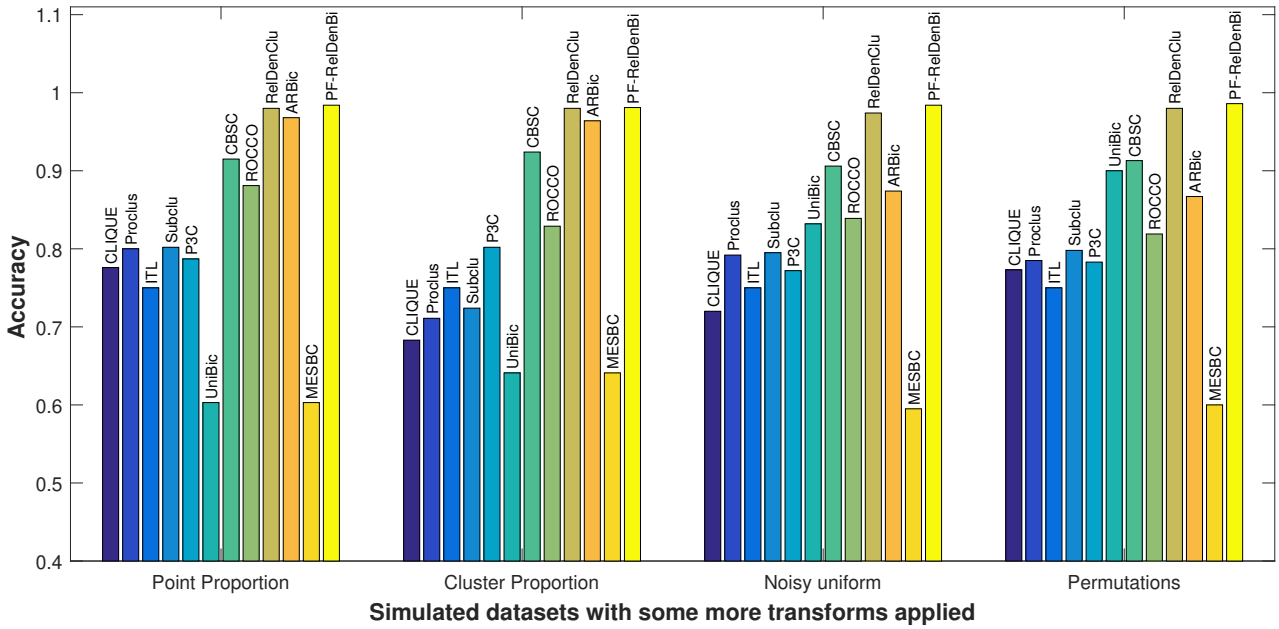
Method	Accuracy	Datasets		
		Normal	Noisy Normal 1	Noisy Normal 2
PF-RelDenBi	Mean	0.971	0.964	0.934
	Std. Dev.	0.022	0.015	0.020
MSBEC	Mean	0.746	0.747	0.746
	Std. Dev.	0.009	0.009	0.010
ARBic	Mean	0.975	0.812	0.784
	Std. Dev.	0.010	0.009	0.002
RelDenClu	Mean	0.958	0.941	0.810
	Std. Dev.	0.043	0.074	0.010
ROCCO	Mean	0.753	0.755	0.752
	Std. Dev.	0.011	0.009	0.003
CBSC	Mean	0.951	0.950	0.886
	Std. Dev.	0.075	0.074	0.074
UniBic	Mean	0.869	0.794	0.776
	Std. Dev.	0.025	0.098	0.009
P3C	Mean	0.767	0.758	0.765
	Std. Dev.	0.017	0.047	0.009
Subclu	Mean	0.783	0.78	0.769
	Std. Dev.	0.010	0.015	0.006
ITL	Mean	0.75	0.749	0.750
	Std. Dev.	0.003	0.001	0.001
Proclus	Mean	0.783	0.772	0.769
	Std. Dev.	0.017	0.011	0.006
CLIQUE	Mean	0.799	0.787	0.776
	Std. Dev.	0.011	0.015	0.008

Table 6.4: Accuracy for Overlapping synthetic datasets (Mean and Standard Deviation for 10 datasets of each type)

Method	Accuracy	Datasets			
		Overlap 1		Overlap 2	
		First	Second	First	Second
PF-RelDenBi	Mean	0.994	0.999	0.973	0.955
	Std. Dev.	0.010	0.001	0.028	0.108
MESBC	Mean	0.727	0.779	0.701	0.751
	Std. Dev.	0.015	0.009	0.005	0.005
ARBic	Mean	0.927	0.584	0.905	0.603
	Std. Dev.	0.029	0.020	0.031	0.016
RelDenClu	Mean	0.980	0.996	0.906	0.965
	Std. Dev.	0.004	0.002	0.002	0.004
ROCCO	Mean	0.773	0.817	0.751	0.801
	Std. Dev.	0.008	0.007	0.002	0.002
CBSC	Mean	0.924	0.935	0.872	0.883
	Std. Dev.	0.036	0.006	0.012	0.012
UniBic	Mean	0.940	0.626	0.833	0.775
	Std. Dev.	0.037	0.032	0.130	0.143
P3C	Mean	0.751	0.801	0.750	0.800
	Std. Dev.	0.002	0.004	0.001	0.001
Subclu	Mean	0.761	0.801	0.763	0.801
	Std. Dev.	0.005	0.002	0.004	0.002
ITL	Mean	0.750	0.800	0.750	0.800
	Std. Dev.	0.001	0.001	0.001	0.001
Proclus	Mean	0.760	0.804	0.761	0.802
	Std. Dev.	0.009	0.004	0.008	0.003
CLIQUE	Mean	0.750	0.800	0.750	0.800
	Std. Dev.	0.001	0.001	0.001	0.001



(a) Accuracy obtained for Base dataset and its transforms



(b) Accuracy for 4 more transforms applied to Base dataset

Figure 6.3: Accuracy obtained for synthetic datasets with various transforms applied

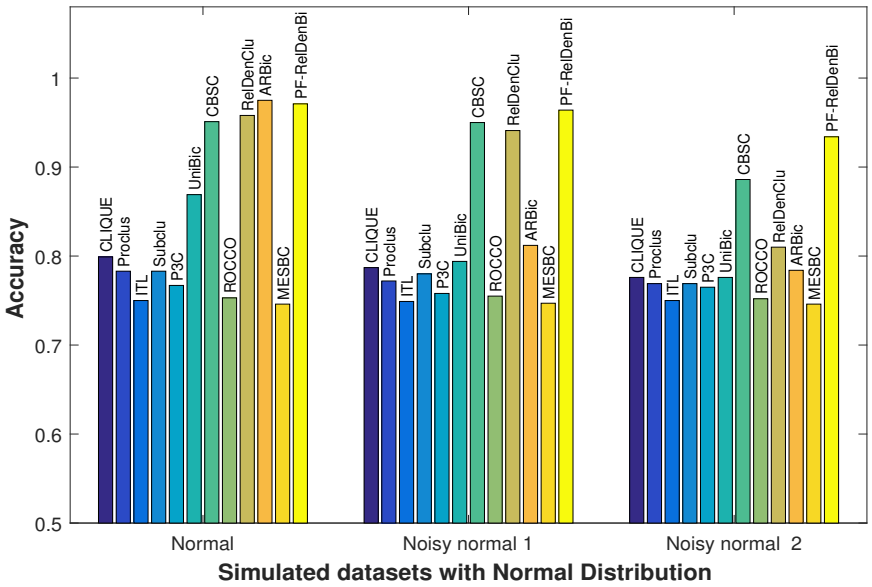


Figure 6.4: Accuracy obtained using various algorithms for Normal and Noisy Normal datasets

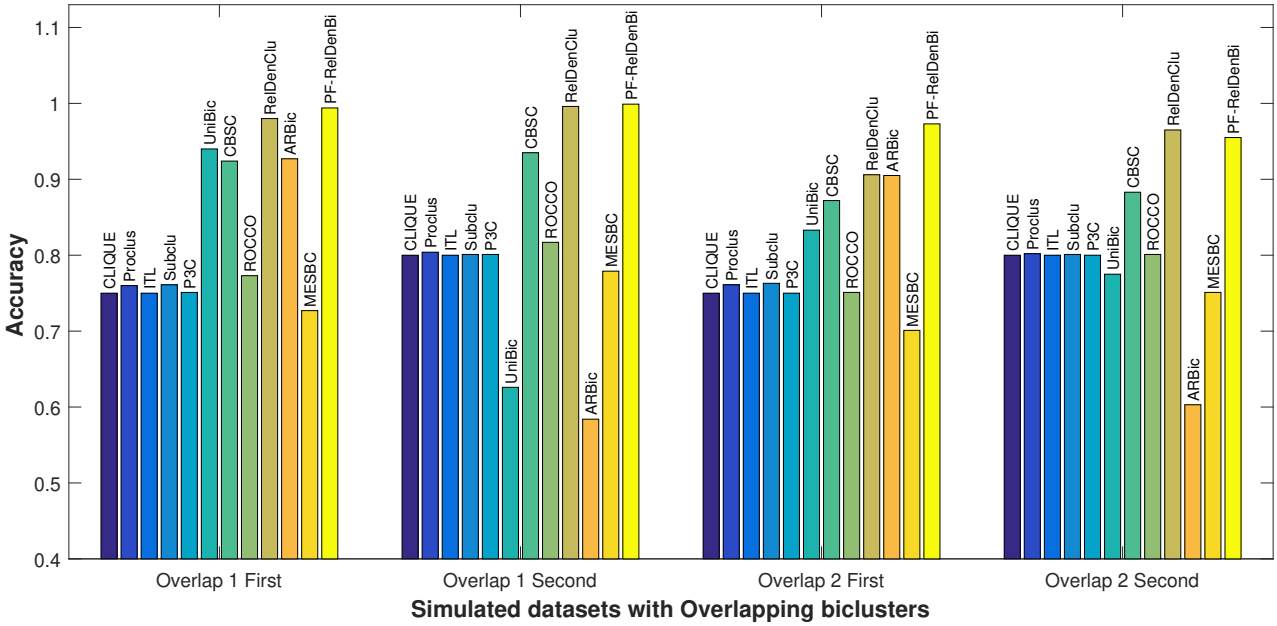


Figure 6.5: Accuracy obtained using various algorithms for Overlapping datasets

For synthetic data, experiments are conducted with 10 different simulations and the average (over 10 simulations) accuracy and corresponding standard deviation (denoted as, "deviation") values are reported in Tables 6.1-6.4. Comparison is done

with eleven other methods. We see that PF-RelDenBi provides better accuracy for fifteen out of the sixteen synthetic datasets.

The columns ("Non-Monotonous", "Non-linear 1" and "Non-linear 2") of Table 6.1 represent datasets having biclusters based on non-linear relations. We see that PF-RelDenBi provides higher accuracy for these three datasets. Unlike other algorithms, PF-RelDenBi finds the biclusters accurately for datasets containing highly non-monotonous relations as seen from the first column. Despite the highly variable distribution of background observations, PF-RelDenBi finds the biclusters with higher accuracy as seen from the column "Non-Linear 1". Its accuracy for another dataset containing bicluster based on non-linear relations is also higher in comparison with other algorithms as seen from column "Non-Linear 2". The values of accuracy obtained using different algorithms on these datasets are presented in Figure 6.2 as bar graphs, for better illustration. The experiments on these three datasets illustrate that PF-RelDenBi can find biclusters where features are related to each other with non-linear or non-monotonous relations as mentioned in Section 6.2. PF-RelDenBi can identify these relations due to its reliance on relative density to find the initial set of observations and the use of the relation index MIDI to find the final biclusters.

The first column of Table 6.2 ("Base") represents a dataset containing biclusters based on linear relation. The other seven columns of the table ("Scaled", "Translated", ..., "Permutations") give results for datasets obtained by applying different transforms to the "Base" dataset. These datasets allow us to analyze the behavior of PF-RelDenBi in a way similar to the analysis done by Ness [24] for clustering algorithms. From Figure 6.3 (shown in parts 6.3a and 6.3b) and the above-mentioned table, we find that PF-RelDenBi has produced better results as compared to other methods for the "Base" dataset and its transforms, i.e. "Scaled",..., "Permutations".

The three columns of Table 6.3 ("Normal", "Noisy Normal 1" and "Noisy Normal 2"), contain results for datasets generated using the normal distribution. In the case of the "Normal" dataset, we find that ARBic has slightly better performance than PF-RelDenBi. But from the results on the "Noisy Normal 1" (noise with standard deviation 0.1) and "Noisy Normal 2" (noise with standard deviation 0.2) datasets, we find that the performance of ARBic decreases more sharply when noise is added. On the other hand, PF-RelDenBi shows a more robust performance and achieves higher accuracy than other methods on "Noisy Normal 1" and "Noisy Normal 2" datasets. The performance for these datasets is reported in Table 6.3 and Figure 6.4.

The first two columns of Table 6.4 ("Overlap 1"), present results for a dataset containing two biclusters with an overlap of 12%. We find that PF-RelDenBi has the best performance among the methods used for comparison. The third and fourth columns present results for a similar dataset with two biclusters but with an overlap of 20%. In this case, we find that RelDenClu finds the second bicluster with the highest accuracy. On the other hand, PF-RelDenBi finds the first bicluster with the highest accuracy. If we take the average over both biclusters in the dataset "Overlap 2", PF-RelDenBi achieves the highest accuracy. These results are presented graphically in Figure 6.5.

These results for synthetic datasets suggest some properties of PF-RelDenBi which will be discussed in Section 6.6.1.3.

6.6.1.2 Execution time for synthetic datasets and time complexity

Execution time is considered as another evaluation criterion, and the results are reported in Tables 6.5-6.8. The datasets used here are the same as those used in Tables 6.1-6.4. The algorithm is executed with an Intel Core i7 CPU and 8 GB memory. As seen from the tables mentioned above, the execution time for PF-RelDenBi is less than

Table 6.5: Execution time in seconds for Non-Linear synthetic datasets

Method	Datasets		
	Non-Monotonous	Non-Linear 1	Non-Linear2
PF-RelDenBi	0.287	0.221	0.232
MSBEC	0.021	0.069	0.075
ARBic	0.258	0.232	0.254
RelDenClu	0.195	0.295	0.275
ROCCO	31.402	4.087	3.164
CBSC	19.294	37.464	31.82
UniBIc	0.012	0.583	0.612
P3C	1.440	0.359	0.650
SubClu	0.037	0.064	0.052
ITL	15.000	9.096	9.248
Proclus	0.026	0.060	0.055
CLIQUE	0.001	0.011	0.007

Table 6.6: Execution time in seconds for Base and transformed synthetic datasets

Method	Datasets							
	Base	Scale	Translated	Linear	Point Proportion	Cluster Proportion	Noisy	Permutations
PF-RelDenBi	0.250	0.238	0.243	0.236	0.488	0.549	0.228	0.243
MSBEC	0.104	0.089	0.063	0.071	0.209	0.153	0.073	0.136
ARBic	0.254	0.229	0.248	0.282	2.751	2.105	0.287	0.216
RelDenClu	0.293	0.291	0.273	0.302	11.103	7.768	0.288	0.293
ROCCO	2.939	3.021	2.681	2.794	29.275	7.509	2.874	2.958
CBSC	42.416	51.400	43.964	41.313	91.973	83.796	42.304	41.654
UniBIc	0.583	0.612	0.649	0.673	2.596	1.436	0.621	0.583
P3C	0.513	0.955	0.952	0.433	1.083	0.848	0.458	0.588
SubClu	0.056	0.043	0.050	0.055	0.120	0.082	0.049	0.070
ITL	3.171	3.302	3.498	3.276	9.994	6.519	2.94	3.272
Proclus	0.052	0.055	0.049	0.055	0.120	0.088	0.056	0.063
CLIQUE	0.105	0.015	0.015	0.018	0.033	0.021	0.014	0.013

Table 6.7: Execution time in seconds for synthetic datasets with Normal Distribution

Method	Datasets		
	Normal	Noisy Normal 1	Noisy Normal 2
PF-RelDenBi	0.234	0.225	0.296
MSBEC	0.069	0.076	0.859
ARBic	0.288	0.264	0.918
RelDenClu	1.398	1.395	71.117
ROCCO	6.258	5.128	2.938
CBSC	66.659	63.821	267.958
UniBic	0.531	0.577	0.001
P3C	18.840	9.751	6.322
SubClu	0.071	0.061	0.050
ITL	1.187	1.259	20.639
Proclus	0.054	0.052	0.084
CLIQUE	0.071	0.044	0.715

Table 6.8: Execution time in seconds for Overlapping synthetic datasets

Method	Datasets	
	Overlap 1	Overlap 2
PF-RelDenBi	0.379	0.430
MESBC	0.240	0.142
ARBic	0.590	0.811
RelDenClu	27.090	27.103
ROCCO	2.571	25.663
CBSC	110.521	115.285
UniBic	2.092	2.775
P3C	0.049	0.016
Subclu	0.074	0.158
ITL	15.102	15.091
Proclus	0.074	0.087
CLIQUE	0.014	0.032

ITL, P3C, UniBic, and CBSC. The execution times of ARBic and PF-RelDenBi are similar. It is high compared to MESBC and density-based methods such as Subclu, Proclus, and CLIQUE. However, it is already seen in Tables 6.1-6.4, that the accuracy of PF-RelDenBi is better than these methods for most of the synthetic datasets.

6.6.1.3 A discussion on properties of PF-RelDenBi as seen from experiments on Synthetic datasets

It is already seen that PF-RelDenBi can find relation based biclusters with better accuracy compared to other algorithms for most of the synthetic datasets, as seen from Tables 6.1-6.4 and Figures 6.2-6.5. In this section, we discuss some of the observed properties of PF-RelDenBi.

PF-RelDenBi is invariant to scaling, translation, and linear transforms because these transforms affect neither the bin length nor estimated densities. Theoretically, the procedure is not invariant to arbitrary order-preserving transforms, which may change the density of the observations. We compare the accuracy obtained for the "Base" dataset and its various transforms. We find that there is no significant difference in the performance of PF-RelDenBi for the data sets "Scaled", "Translated", "Linear Transform", "Point Proportion", "Cluster Proportion" and "Permutations".

Accuracy for the "Noisy Uniform", "Noisy Normal 1" and "Noisy Normal 2" datasets are only slightly lower than the "Base" and "Normal" datasets. Compared to other algorithms, PF-RelDenBi still has higher accuracy for noisy datasets.

We may note that the biclusters in the "Base" dataset can also be seen as scaling biclusters, since each column in the bicluster can be obtained by multiplying another column by a constant value. In addition, the biclusters in the "Translated" datasets are obtained by adding a random selected constant value to each column of the "Base" dataset. Hence, the resulting biclusters can be seen as a combination of scaling and shift biclusters. This happens because scaling and shifting are examples of linear and affine transformations, respectively. Since PF-RelDenBi finds biclusters based on feature relations, it will be able to find biclusters based on scaling and shifting of columns. However, PF-RelDenBi will not be able to find biclusters based on the shifting and scaling of rows, as it specifically aims to find biclusters based on feature relations.

We find that RelDenClu and CBSC also have similar performance for the "Base" and various transformed datasets. However, PF-RelDenBi obtains higher accuracy for these datasets.

Overall, we find that PF-RelDenBi performs well on data after applying different transforms or adding noise. Its performance on datasets containing non-linear datasets is also good.

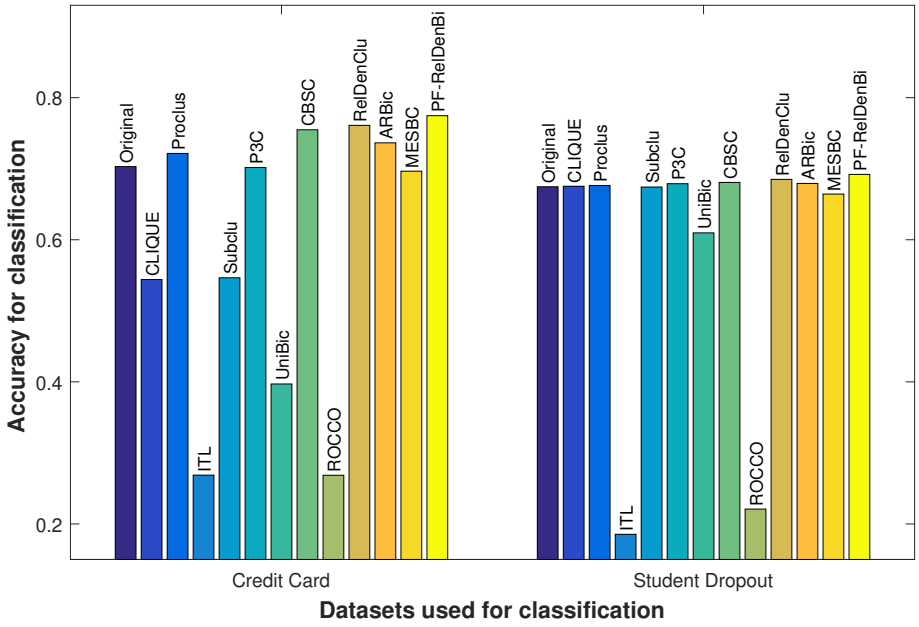
6.6.2 Results for experiments with UCI-ML datasets

6.6.2.1 Experimental results for classification of UCI-ML datasets

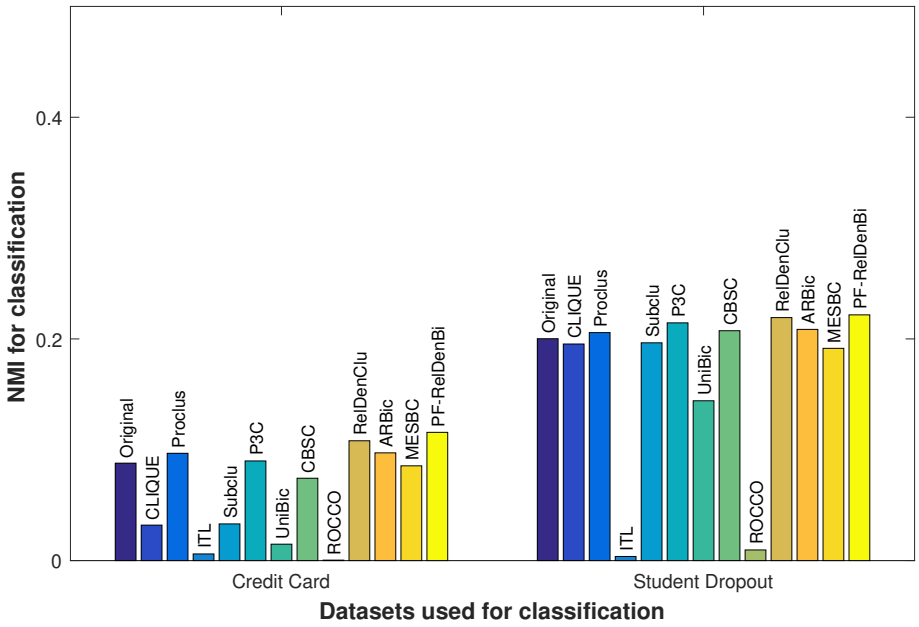
The results of the application of PF-RelDenBi to classification are reported in Table 6.9 and the performance in terms of various indices is presented as bar graphs in Fig-

Table 6.9: Classification performance for Credit card and Student Dropout datasets (after adding features generated using different biclustering methods)

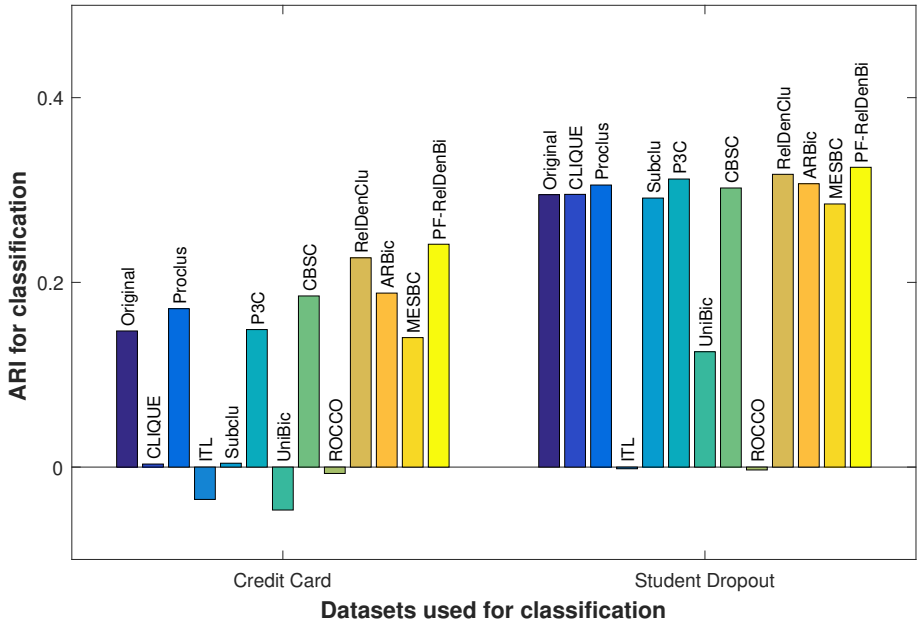
Dataset	**Credit Card [175]**						
Method	Accuracy		NMI	ARI	Average		
	Mean	S. Dev.			Prec.	Recall	G-Score
PF-RelDenBi	0.775	0.007	0.116	0.241	0.680	0.694	0.686
MESBC	0.697	0.015	0.086	0.140	0.638	0.687	0.650
ARBic	0.736	0.012	0.097	0.188	0.654	0.692	0.668
RelDenClu	0.761	0.006	0.108	0.227	0.672	0.702	0.685
ROCCO	0.269	0.005	0.000	-0.007	0.510	0.504	0.355
CBSC	0.755	0.006	0.074	0.185	0.647	0.651	0.649
UniBic	0.397	0.006	0.015	-0.047	0.559	0.562	0.472
P3C	0.702	0.012	0.090	0.149	0.642	0.690	0.654
Subclu	0.547	0.098	0.033	0.004	0.581	0.616	0.560
ITL	0.269	0.010	0.006	-0.035	0.562	0.520	0.359
Prolus	0.722	0.015	0.097	0.172	0.650	0.696	0.665
CLIQUE	0.544	0.006	0.032	0.003	0.580	0.614	0.558
Original	0.703	0.028	0.088	0.147	0.640	0.689	0.654
Dataset	**Student Dropout [176]**						
Method	Accuracy		NMI	ARI	Average		
	Mean	S. Dev.			Prec.	Recall	G-Score
PF-RelDenBi	0.692	0.015	0.222	0.325	0.624	0.600	0.609
MESBC	0.664	0.021	0.192	0.285	0.566	0.554	0.555
ARBic	0.679	0.020	0.209	0.307	0.602	0.583	0.589
RelDenClu	0.685	0.018	0.219	0.317	0.616	0.602	0.608
ROCCO	0.221	0.011	0.010	-0.003	0.434	0.360	0.264
CBSC	0.681	0.016	0.207	0.302	0.604	0.583	0.589
UniBic	0.610	0.009	0.144	0.125	0.550	0.450	0.455
P3C	0.679	0.020	0.215	0.312	0.595	0.581	0.585
Subclu	0.674	0.018	0.197	0.291	0.602	0.586	0.591
ITL	0.185	0.003	0.004	-0.002	0.417	0.336	0.184
Proclus	0.676	0.018	0.206	0.305	0.601	0.589	0.593
CLIQUE	0.675	0.022	0.195	0.295	0.597	0.576	0.582
Original	0.675	0.015	0.200	0.295	0.593	0.576	0.580



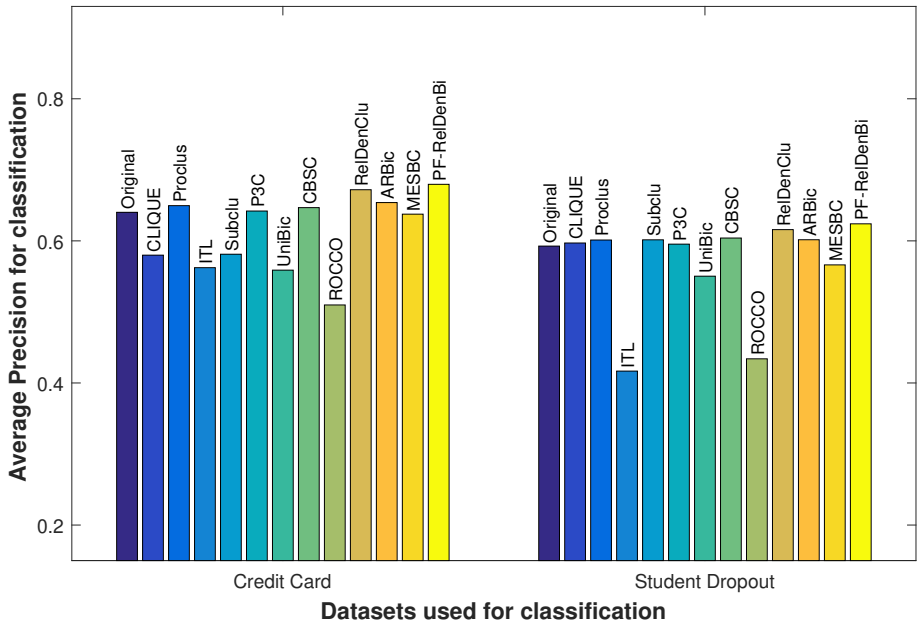
(a) Accuracy of various biclustering algorithms for classification



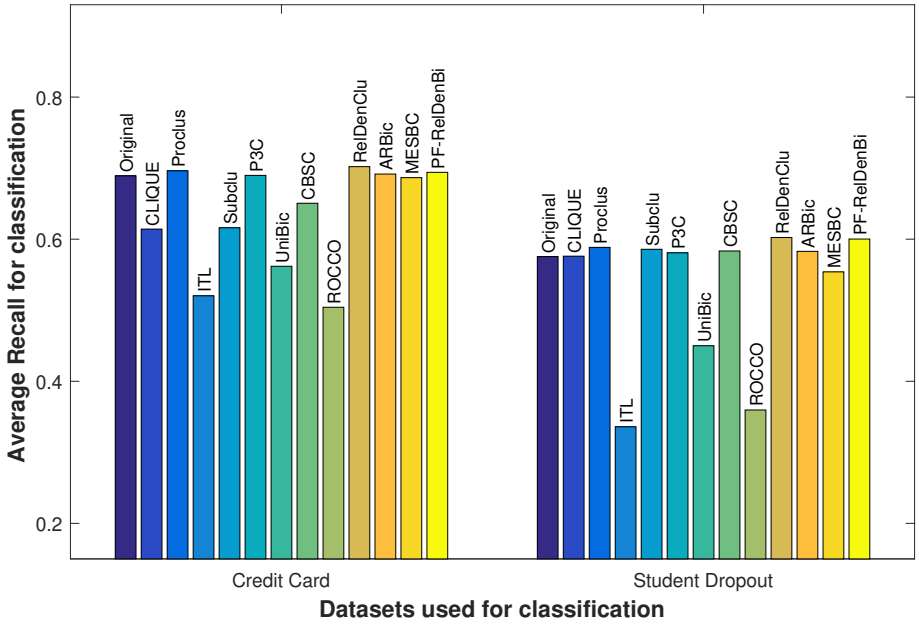
(b) NMI obtained by various algorithms for classification



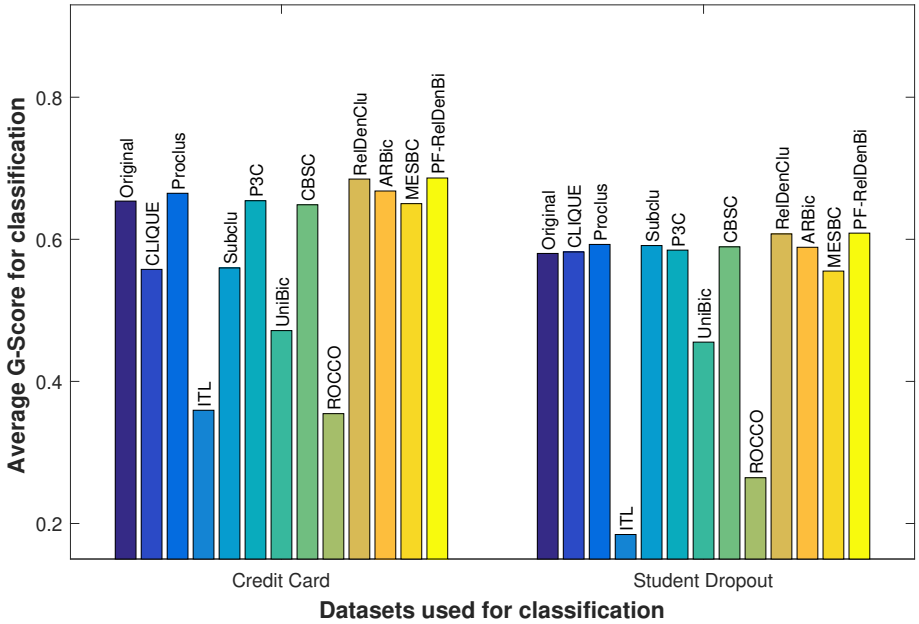
(c) ARI obtained by various biclustering methods for classification



(d) Average precision obtained by various biclustering algorithms for classification



(e) Average recall obtained by various biclustering algorithms for classification



(f) Average G-Score obtained by various biclustering algorithms for classification

Figure 6.6: Performance of various biclustering algorithms for classification

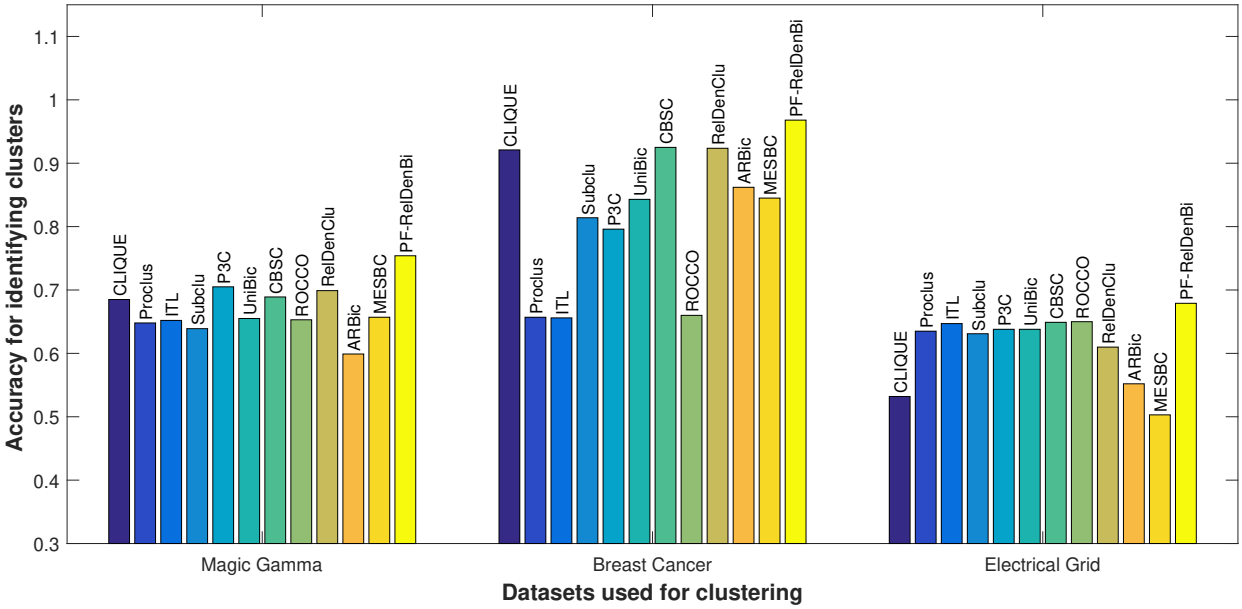
ures 6.6a-6.6f. We find that for the Credit Card dataset, the features generated by PF-RelDenBi provide a greater improvement in accuracy compared to eleven other algorithms, thereby leading to better identification of credit card defaulters. RelDenClu and CBSC also resulted in similar improvements. However, this was achieved only after the parameters were determined using cross-validation, making the process more computationally costly and time consuming. For the Student Dropout dataset, PF-RelDenBi provides an improvement over the original accuracy. It also results in better NMI and ARI. Here, too, RelDenClu provides a similar improvement only after determining the parameters with cross-validation. Overall, we find that PF-RelDenBi provides better accuracy for these datasets without the hassle of finding suitable parameters.

6.6.2.2 Experimental results with clustering of UCI-ML datasets

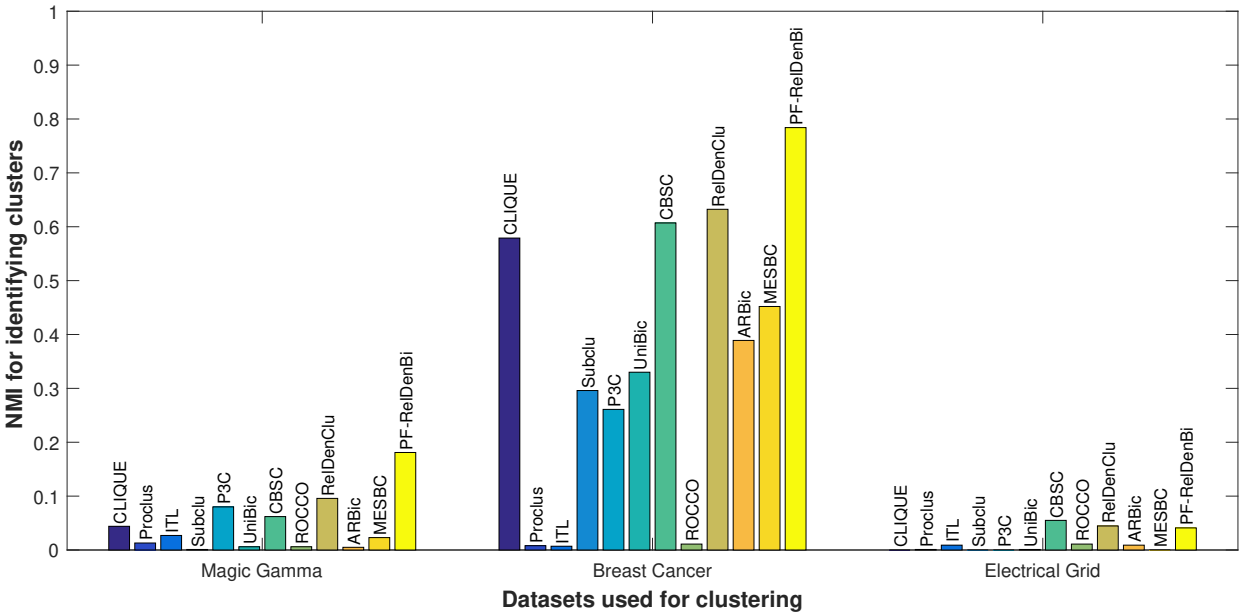
The results for clustering of UCI-ML datasets are reported in Table 6.10. PF-RelDenBi is seen to have the best accuracy for the datasets used for the investigation. For better visualization, the accuracies obtained by different methods are presented in Figure 6.7a as bar graphs. NMI and ARI are shown in Figures 6.7b and 6.7c, respectively. The average Precision, Recall and G-score of both classes have been shown in Figures 6.7d-6.7f, respectively. We find that PF-RelDenBi obtains better accuracy for three datasets namely Magic Gamma, Breast cancer, and Electric Grid. PF-RelDenBi obtains the best NMI and ARI for Magic and Breast Cancer datasets. It does not achieve a better G-Score for each class individually but achieves a higher G-Score averaged over the two classes for the Magic and Breast Cancer datasets.

Table 6.10: Results for clustering of UCI-ML datasets

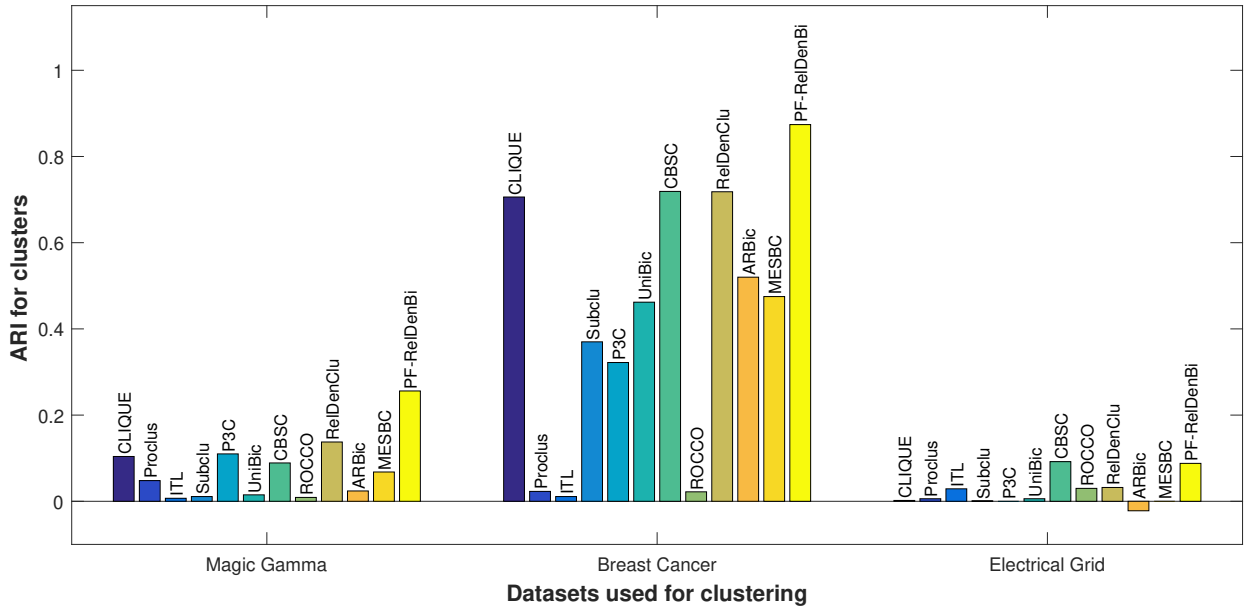
Dataset	**Magic Gamma [177]**								
Method	Accuracy	NMI	ARI	Class 1			Class 2		
				Prec.	Recall	G-Score	Prec.	Recall	G-Score
PF-RelDenBi	0.754	0.181	0.256	0.848	0.755	0.800	0.625	0.751	0.685
MESBC	0.657	0.023	0.068	0.690	0.857	0.769	0.522	0.289	0.388
ARBic	0.599	0.005	0.024	0.674	0.739	0.706	0.414	0.341	0.376
RelDenClu	0.699	0.096	0.138	0.629	0.500	0.536	0.770	0.807	0.779
ROCCO	0.653	0.006	0.009	0.845	0.016	0.117	0.652	0.998	0.807
CBSC	0.689	0.062	0.089	0.839	0.225	0.376	0.700	0.941	0.808
UniBic	0.655	0.006	0.015	0.654	0.991	0.805	0.677	0.035	0.153
P3C	0.705	0.080	0.110	0.690	0.992	0.827	0.919	0.177	0.404
Subclu	0.639	0.001	0.011	0.653	0.943	0.785	0.425	0.077	0.181
ITL	0.652	0.027	0.007	0.953	0.012	0.107	0.651	1.000	0.807
Proclus	0.648	0.013	0.048	0.676	0.878	0.770	0.498	0.223	0.333
CLIQUE	0.685	0.044	0.104	0.703	0.889	0.791	0.600	0.308	0.430
Dataset	**Breast Cancer (Wisconsin Original)** [178]								
Method	Accuracy	NMI	ARI	Class 1			Class 2		
				Prec.	Recall	G-Score	Prec.	Recall	G-Score
PF-RelDenBi	0.968	0.784	0.874	0.986	0.964	0.975	0.936	0.975	0.955
MESBC	0.845	0.452	0.475	0.697	0.983	0.828	0.988	0.770	0.873
ARBic	0.862	0.389	0.520	0.887	0.903	0.895	0.814	0.787	0.800
RelDenClu	0.924	0.632	0.718	0.834	0.981	0.904	0.989	0.893	0.939
ROCCO	0.660	0.011	0.022	0.733	0.046	0.184	0.659	0.991	0.808
CBSC	0.925	0.607	0.719	0.866	0.932	0.898	0.963	0.920	0.941
UniBic	0.843	0.330	0.462	0.833	0.690	0.758	0.847	0.926	0.886
P3C	0.796	0.261	0.322	0.981	0.427	0.647	0.763	0.995	0.872
Subclu	0.814	0.296	0.370	0.975	0.481	0.685	0.781	0.993	0.880
ITL	0.656	0.007	0.011	0.833	0.021	0.132	0.654	0.998	0.808
Proclus	0.657	0.008	0.023	0.600	0.063	0.194	0.660	0.977	0.803
CLIQUE	0.921	0.579	0.706	0.881	0.895	0.888	0.943	0.935	0.939
Dataset	**Electrical Grid**								
Method	Accuracy	NMI	ARI	Class 1			Class 2		
				Prec.	Recall	G-Score	Prec.	Recall	G-Score
PF-RelDenBi	0.679	0.041	0.088	0.680	0.939	0.799	0.673	0.220	0.385
MESBC	0.503	0.000	0.000	0.363	0.497	0.425	0.639	0.506	0.569
ARBic	0.552	0.009	-0.022	0.255	0.124	0.177	0.615	0.795	0.699
RelDenClu	0.610	0.045	0.032	0.415	0.497	0.437	0.737	0.674	0.693
ROCCO	0.650	0.011	0.030	0.651	0.973	0.796	0.631	0.082	0.228
CBSC	0.649	0.055	0.092	0.503	0.487	0.491	0.724	0.741	0.731
UniBic	0.638	0.001	0.006	0.497	0.026	0.114	0.641	0.985	0.794
P3C	0.638	0.000	0.000	0.638	1.000	0.799	0.000	0.000	0.000
Subclu	0.631	0.000	0.001	0.383	0.030	0.107	0.639	0.973	0.788
ITL	0.647	0.009	0.029	0.572	0.096	0.235	0.652	0.959	0.791
Proclus	0.635	0.001	0.006	0.444	0.038	0.129	0.641	0.973	0.790
CLIQUE	0.532	0.000	0.002	0.372	0.425	0.398	0.645	0.593	0.619



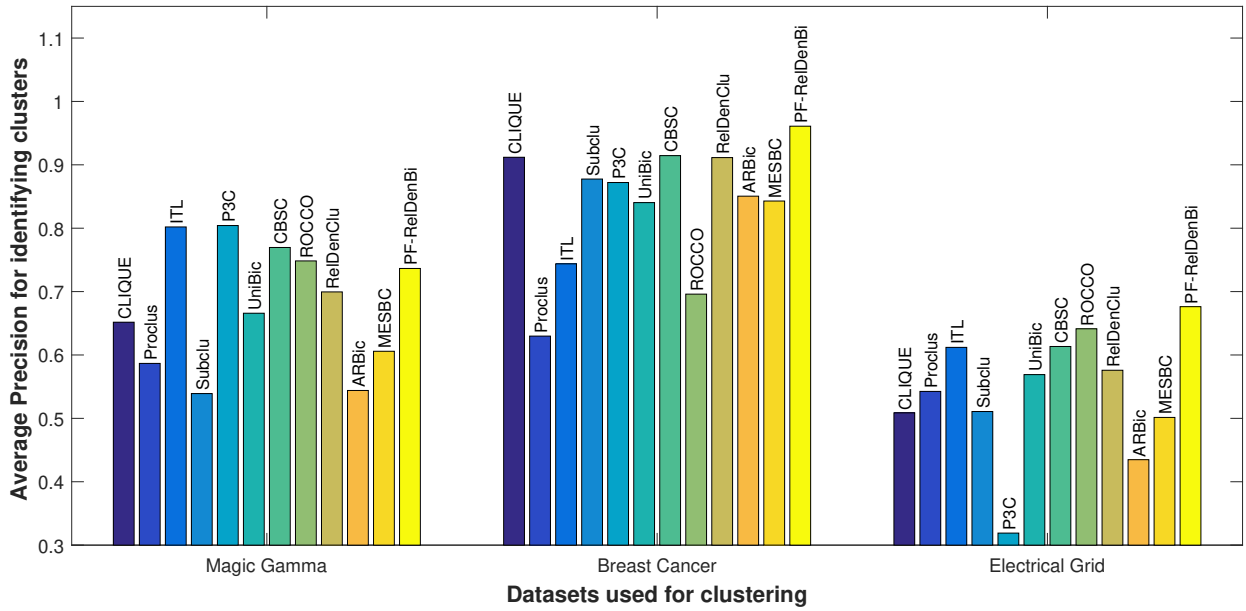
(a) Accuracy of various algorithms for clustering of UCI-ML datasets (calculated using Equation 4.1)



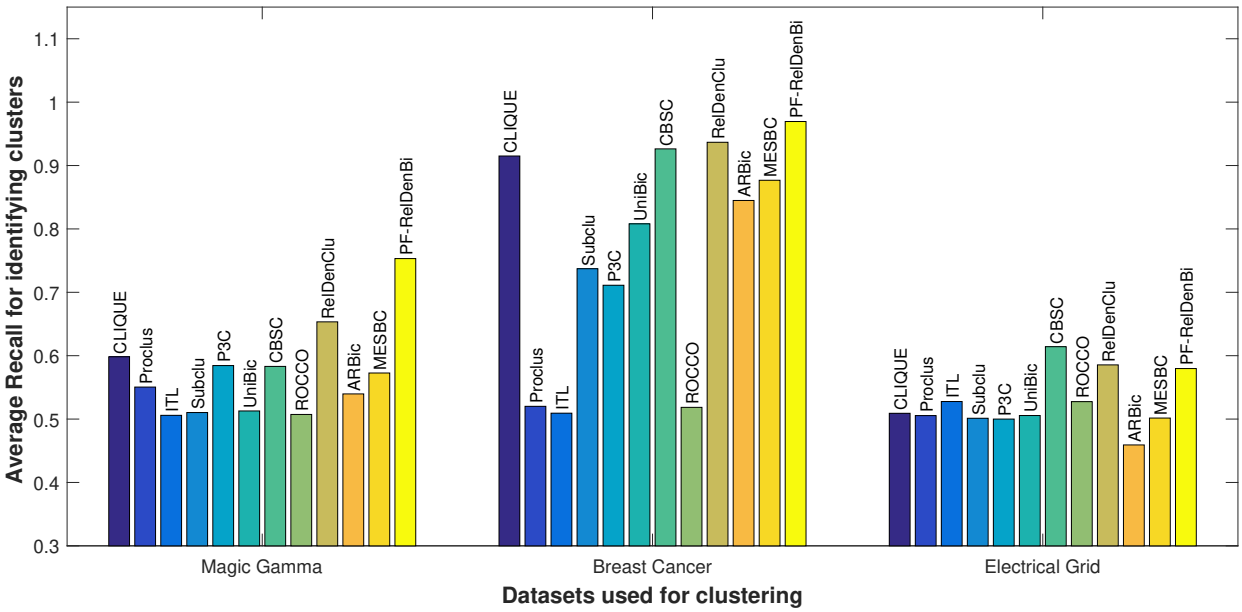
(b) NMI obtained by various algorithms for clustering of UCI-ML datasets



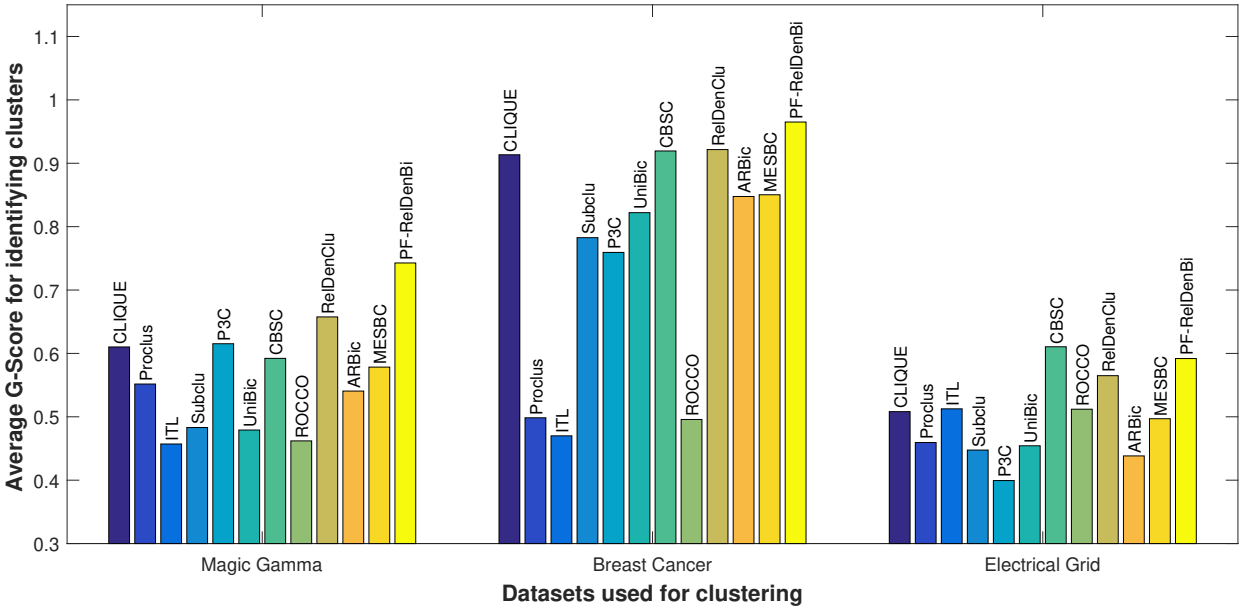
(c) ARI obtained by various algorithms for clustering of UCI-ML datasets



(d) Average precision obtained by various algorithms for clustering of UCI-ML datasets



(e) Average recall obtained by various algorithms for clustering of UCI-ML datasets



(f) Average G-Score obtained by various algorithms for clustering of UCI-ML datasets

Figure 6.7: Performance of various algorithms for clustering of UCI-ML datasets

Table 6.11: Features (WDI dataset) selected by biclusters based on countries having high COVID-19 occurrence (90 percentile) in January 2020

S. No.	Feature description	ρ_s(Jan)	ρ_s(Dec)	UniBic	CBSC	ROCCO	RelDenClu	ARBic	MESBC	PF-RelDenBi
1	Current health expenditure per capita, PPP (current international $)	0.22	0.69	✓	-	-	-	-	-	✓
2	Cause of death, by communicable diseases and maternal, prenatal and nutrition conditions (% of total)	-0.09	-0.68	✓	✓	-	✓	-	✓	✓
3	Cause of death, by non-communicable diseases (% of total)	0.12	0.67	✓	✓	-	✓	-	-	✓
4	Mortality rate, adult, female (per 1,000 female adults)	-0.27	-0.67	-	✓	-	✓	-	✓	✓
5	Survival to age 65, female (% of cohort)	0.27	0.67	✓	✓	-	✓	✓	-	✓
6	People using at least basic sanitation services (% of population)	0.23	0.64	✓	✓	-	✓	-	-	✓
7	Life expectancy at birth, total (years)	0.26	0.64	✓	-	-	✓	✓	-	✓
8	GDP per capita, PPP (current international $)	0.22	0.62	✓	✓	-	✓	✓	-	✓
9	Tuberculosis case detection rate (%, all forms)	0.13	0.61	-	-	-	✓	-	-	-
10	Population ages 65 and above (% of total)	0.25	0.59	✓	-	-	-	✓	-	-
11	Survival to age 65, male (% of cohort)	0.24	0.58	✓	✓	-	✓	-	-	✓
12	Urban population (% of total)	0.12	0.58	-	-	-	✓	-	-	-
13	Mortality rate, adult, male (per 1,000 male adults)	-0.23	-0.57	-	✓	-	✓	-	✓	✓
14	Incidence of tuberculosis (per 100,000 people)	-0.05	-0.51	✓	✓	-	✓	-	✓	✓
15	Population ages 15-64 (% of total)	0.07	0.46	✓	-	-	-	-	-	✓
16	International tourism, number of arrivals	0.4	0.44	✓	-	-	-	-	-	-
17	Mortality from CVD, cancer, diabetes or CRD between exact ages 30 and 70 (%)	-0.23	-0.4	-	✓	-	✓	-	✓	✓
18	PM2.5 air pollution, population exposed to levels exceeding WHO guideline value (% of total)	-0.23	-0.38	-	-	✓	-	✓	✓	-
19	Air transport, passengers carried	0.44	0.37	-	-	-	-	-	-	-
20	International migrant stock, total	0.35	0.27	-	-	-	-	-	-	-
21	Trade (% of GDP)	0.03	0.26	-	-	-	-	-	-	-
22	Out-of-pocket expenditure (% of current health expenditure)	-0.09	-0.25	-	✓	-	✓	-	✓	-
23	Labor force participation rate, total (% of total population ages 15+) (modeled ILO estimate)	0.04	-0.24	-	-	-	-	-	✓	-
24	Population density (people per sq. km of land area)	0.23	0.16	✓	-	-	✓	-	-	-
25	Population, total	0.34	-0.16	✓	-	-	✓	✓	✓	-
26	Tuberculosis treatment success rate (% of new cases)	-0.08	-0.15	✓	-	-	-	-	-	-
27	Death rate, crude (per 1,000 people)	0.02	0.07	✓	-	-	-	-	-	-

6.6.3 Relevant features in COVID-19 dataset obtained using PF-RelDenBi during early stages of the pandemic

Table 6.11 lists the 27 features used for analysis [193]. The features obtained by all the methods, using the data from January 2020 have been indicated by a tick in respective columns. The table also reports the values of Spearman correlation (ρ_s) between the chosen feature and the infection rate of COVID-19 for various countries for 31 January 2020 (denoted as, ρ_s(Jan)) and 31 December 2020 (denoted as, ρ_s(Dec)). The features are listed in decreasing order of absolute value of ρ_s(Dec).

PF-RelDenBi selected 13 of 27 WDI features capable of distinguishing regions with high infection rates. We find that 11 out of 13 features selected by PF-RelDenBi have a Spearman correlation (absolute value) greater than 0.5 with the disease occurrence rate on 31 December 2020. This indicates that PF-RelDenBi could identify the importance of these features much earlier, i.e., from the January data, when the Spearman correlation could not reflect this relation. Amongst the features not included in the bicluster, only 3 out of the 14 features have a Spearman correlation (absolute value) greater than 0.5 with disease occurrence rates on 31 December 2020.

The above-mentioned results suggest that PF-RelDenBi could be explored to identify important factors that impact a new disease, and this, in turn, will aid in preparing a strategy, more scientifically, to combat the pandemic situation across countries. Indeed some of these selected features for COVID-19 have already been studied and are found to be important, as discussed in the following paragraphs, where we refer to features using their serial number (S. No.) given in Table 6.11.

Among the features selected by PF-RelDenBi, the features "Current health expenditure per capita" and "GDP per capita" (S No. 1 and 8 in Table 6.11) may be related to the higher COVID-19 rate due to the higher number of tests. The per-

centage of deaths in the region due to communicable and non-communicable diseases (S. Nos. 2 and 3 of Table 6.11) is consistent with the report given by Chakrabati et al. [200]. The features showing the life expectancy and mortality rate for the overall population and for each gender affect the age distribution of people in a region (S. Nos. 4,5,7,11,13,15,17 of Table 6.11) are selected by PF-RelDenBi. The feature "People using at least basic sanitation services" (S. No. 6 of Table 6.11) is related to the spread of communicable diseases and the corresponding immunity. The feature "Incidence of tuberculosis" (S. No. 14 of Table 6.11), is interesting. Countries with higher tuberculosis incidences are also seen to have relatively lower rates of COVID-19 infection. A study [201] has already noted that the administration of the BCG vaccine is likely to reduce the severity of the symptoms of COVID-19. Another factor that may contribute to this association is that tuberculosis is more rampant in warm humid climates, whereas COVID-19 is expected to spread faster in cold dry climates [202].

As mentioned earlier, all features selected by PF-RelDenBi, except two (S.Nos. 15 and 17 of Table 6.11) have a high correlation with COVID infection rates on 31 December 2020, although the biclusters are obtained using the least amount of data from 31 January 2020.

Performance comparison between PF-RelDenBi and other methods is shown in Figure 6.8, which shows a scatter plot for Spearman correlation (absolute values). For each method, selected features are marked in blue and rejected features in red. For PF-RelDenBi, the selected features are mainly the ones that have a higher correlation. In fact, the minimum absolute value attained by any feature selected by PF-RelDenBi is 0.4, another selected feature attains a value of 0.46, while all other selected features lie above 0.5.

In summary, most of the features selected by PF-RelDenBi are relevant, and the

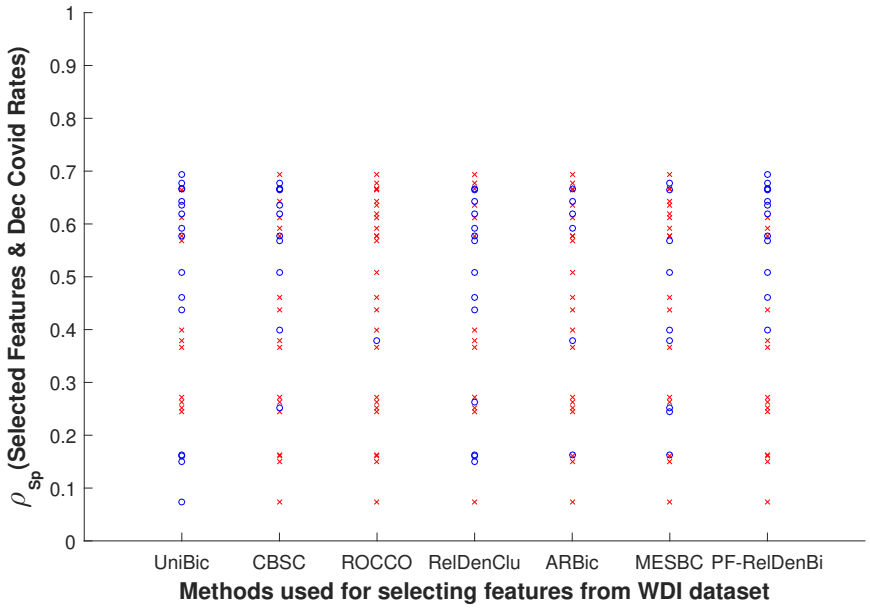


Figure 6.8: Spearman correlation (absolute value) between Dec 31 2020 COVID infection rates and features selected from WDI database by biclustering methods. Selected features are marked in blue and rejected in red

presented methodology allows us to explore some previously unknown associations. Once features are identified from larger datasets, a more detailed analysis could be performed for individual features.

6.6.4 Results of analysis on COVID-19 and vaccine dataset

We applied PF-RelDenBi to the vaccine dataset described in Section 6.5.5. PF-RelDenBi selects the percentage of newborns vaccinated with Tetanus-Toxoid. Previous research has identified that the administration of the Tetanus-Toxoid vaccine may contribute to a lower severity of symptoms associated with COVID-19. This is based on the observed similarities between the COVID-19 spike protein and the tetanus toxin [203]. Our experiments support this hypothesis. Analysis using UniBic, CBSC, ROCCO and RelDenClu methods also indicates that the coverage of the Tetanus Toxoid vaccine may be related to the infection rate of COVID-19.

6.7 Limitations and Future Work

With the encouraging results obtained on several datasets, we would like to further develop the proposed method. Some possibilities for future work are discussed below. It may be noted that the main objective of the proposed method is to find relations between features for a subset of observations. When used for clustering, it would work best for datasets where different clusters are defined by different relations between the features. However, in datasets where clusters are well separated only by spatial distances, the proposed method may not perform well. Therefore, in the future, we would like to experiment with ensemble methods utilizing the proposed method. We would also like to enhance the proposed algorithm so that it can utilize parallel computing allowing its application to larger datasets. Further, the proposed method either includes or excludes an observation or a feature in a bicluster; but does not provide a membership value to it. In the future, we would like to work on assigning a membership value to observations and features using fuzzy sets.

6.8 Summary

In this chapter, we present a parameter-free algorithm that finds biclusters based on non-linear relations between features. By analyzing local variations in marginal and joint densities, the algorithm is seen to perform well on non-linear datasets. PF-RelDenBi does not require any user-defined parameter, making it a versatile method that can be used to analyze datasets in different domains. Experiments on synthetic datasets have shown that PF-RelDenBi is consistent under linear transforms. It is also seen to provide better performance on datasets obtained by adding noise. The performance of PF-RelDenBi is compared to eleven state-of-the-art methods for these

synthetic datasets. PF-RelDenBi is seen to provide better accuracy in most cases.

PF-RelDenBi is seen to improve the classification accuracy when used as a precursor for supervised learning for the Credit Card dataset and Student Dropout datasets. As an unsupervised learning method, it can find classes hidden in the Magic Gamma, Breast Cancer, and Electrical grid datasets with greater accuracy.

PF-RelDenBi has also been applied to a dataset containing information on COVID-19 infection rates in different countries, along with the corresponding development indicators, to identify demographic factors that influence the number of confirmed infections in a region. Furthermore, PF-RelDenBi was applied to a dataset on vaccine coverage along with COVID-19 infection rates in different countries, where it identified an association between infection rates and the Tetanus-Toxoid vaccine. This finding is consistent with independent research on the COVID-19 spike protein [203]. Overall, PF-RelDenBi enables the discovery of relationships among features within specific subsets of the data, revealing feature relationships that may be obscured by unrelated observations or missed due to non-linear dependencies.

Chapter 7

Conclusion and Scope for Further Research

7.1 Conclusion

This thesis explores two complementary approaches to understand non-linear relationships between features.

The first approach involves the development of a novel non-linear feature index, MIDI (Mutual Information based Dependence Index) [28], which utilizes normalized mutual information for identifying feature dependencies. MIDI uses histogram for probability density estimation to calculate mutual information and entropy. MIDI determines the bin length for the histogram using the maximal spacing for each feature, allowing for efficient and consistent density estimation. This makes MIDI a practical and effective tool for capturing non-linear relationships among features. In addition, MIDI has also been used to design an unsupervised feature selection method [29]. The details of MIDI and the associated feature selection method are discussed in Chapter 3.

The second approach to understand non-linear relationships is feature relation-based biclustering. This approach aims to identify subsets of observations that exhibit linear or non-linear feature relationships. In this context, three novel biclustering algorithms have been developed.

The first biclustering method, CBSC [30], presented in Chapter 4, identifies dense regions by examining the connectivity among observations in two-dimensional space. These dense regions form the basis for discovering subspace clusters in higher dimensions. Regions are grouped into the same bicluster if they share similar observations and their corresponding feature sets are connected via a path, enabling the identification of biclusters governed by non-linear feature relationships.

The second biclustering method, RelDenClu [32], introduced in Chapter 5, identifies dense regions using joint and marginal densities. This approach allows the method to adapt to variations in marginal distributions. Like CBSC, RelDenClu extracts biclusters from connected regions in two-dimensional spaces, by grouping dense regions that share similar sets of observations and are also connected by common features.

The third method, PF-RelDenBi [33], described in Chapter 6, is built on RelDenClu by using MIDI to group similar dense regions, thus reducing computational time. Additionally, PF-RelDenBi calculates all the thresholds required to find the biclusters directly from the dataset, making it a parameter-free method. This enhances the practical utility of the method, particularly for unsupervised learning tasks, where hyperparameter tuning can be especially challenging.

Performances of these biclustering methods have been systematically evaluated on sixteen synthetic datasets to demonstrate their distinctive properties. Their practical utility has been further validated on several real-world datasets. Specifically, the methods have been applied to enhance two UCI-ML datasets in order to obtain higher classification accuracy. They have been applied to three UCI-ML datasets to detect

clusters in the data. They have also been used to uncover emerging patterns in COVID-19 datasets, to find relationships between lifestyle factors and infection rates across countries.

7.2 Scope for future Work

As discussed in the previous section, this thesis introduces a novel feature relationship index, MIDI, and three biclustering methods: CBSC, RelDenClu, and PF-RelDenBi. Experimental evaluation on both synthetic and real-world datasets has demonstrated their effectiveness in various pattern recognition tasks. Nevertheless, several opportunities for further development and exploration remain. The following directions outline possible avenues for future work.

1. Extension of MIDI for other information-theoretic applications: MIDI provides a fast and reliable estimate of the dependence between two variables by computing mutual information normalized by the minimum of their entropies. This allows MIDI to detect non-monotonous relationships. In future work, we aim to extend the underlying histogram-based estimation scheme to other applications that require efficient estimation of mutual information or entropy.

2. Supervised feature selection using MIDI: While MIDI has been used in this thesis for unsupervised feature selection, its potential in supervised settings remains unexplored. Future research could investigate its applications to supervised feature selection tasks, where class-label information is available.

3. Parallelization of biclustering algorithms: The biclustering methods proposed in this thesis rely on pairwise feature analysis, which can become computationally expensive in high-dimensional spaces. To improve scalability and efficiency, we

plan to develop parallelized versions of these algorithms, making them more suitable for large-scale data.

4. Ensemble approaches for general clustering tasks: The primary objective of the proposed biclustering methods is to discover feature relationships within subsets of observations. These methods perform best when clusters are defined by distinct feature relationships, rather than spatial proximity alone. To handle a wider variety of clustering scenarios, future work may involve designing ensemble methods that combine relation-based biclustering with distance-based or density-based clustering techniques.

5. Incorporating fuzzy membership values: Currently, the biclustering methods assign hard inclusion or exclusion of observations and features within biclusters. A natural extension is to integrate fuzzy set theory to assign degrees of membership, enabling soft clustering and better capturing uncertainty and overlapping structures in the data.

6. Causal inference using relation-based biclustering: The usefulness of relation-based biclustering as a preliminary step for causal analysis has been demonstrated in the context of COVID-19 datasets. In future, we intend to design a generalized framework that integrates these methods into causal inference pipelines for broader application domains.

In summary, the exploration of non-linear and non-monotonous relationships in data remains an open area for investigation. Continued research in this direction holds the potential to uncover novel patterns and generate deeper insights across diverse application areas.

Bibliography

[1] J. Friedenberg and G. Silverman, *Cognitive Science: An Introduction to the Study of Mind*. SAGE Publications, 2015.

[2] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[3] K. Murphy, *Probabilistic Machine Learning: Advanced Topics*. Adaptive Computation and Machine Learning series, MIT Press, 2023.

[4] J. L. Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *The American Statistician*, vol. 42, no. 1, pp. 59–66, 1988.

[5] P. Schober, C. Boer, and L. A. Schwarte, "Correlation coefficients: appropriate use and interpretation," *Anesthesia & analgesia*, vol. 126, no. 5, pp. 1763–1768, 2018.

[6] A. Rényi, "On measures of dependence," *Acta Mathematica Academiae Scientiarum Hungarica*, vol. 10, no. 3-4, pp. 441–451, 1959.

[7] T. F. Móri and G. J. Székely, "Four simple axioms of dependence measures," *Metrika*, vol. 82, no. 1, pp. 1–16, 2019.

[8] R. Schilling, *Measures, Integrals and Martingales*. Measures, Integrals and Martingales, Cambridge University Press, 2017.

[9] G. Bontempi and M. Flauder, "From dependency to causality: A machine learning approach," *Journal of Machine Learning Research*, vol. 16, no. 74, pp. 2437–2457, 2015.

[10] N. Altman and M. Krzywinski, "Association, correlation and causation," *Nature Methods*, vol. 12, pp. 899–900, 2015.

[11] J. Pearl, *Causality*. Cambridge university press, 2009.

[12] M. Ebrahimi Warkiani and M. H. Moattar, "A comprehensive survey on recent feature selection methods for mixed data: Challenges, solutions and future directions," *Neurocomputing*, vol. 623, p. 129372, 2025.

[13] L. Morán-Fernández, V. Bólon-Canedo, and A. Alonso-Betanzos, "How important is data quality? Best classifiers vs best features," *Neurocomputing*, vol. 470, pp. 365–375, 2022.

[14] A. Tanay, R. Sharan, and R. Shamir, "Biclustering algorithms: A survey," *Handbook of Computational Molecular Biology*, vol. 9, no. 1-20, pp. 122–124, 2005.

[15] R. Vidal, "Subspace clustering," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2011.

[16] E. N. Castanho, H. Aidos, and S. C. Madeira, "Biclustering data analysis: a comprehensive survey," *Briefings in Bioinformatics*, vol. 25, no. 4, p. bbae342, 2024.

[17] H. Wang, Y. Song, W. Chen, Z. Luo, C. Li, and T. Li, "A Survey of Co-Clustering," *ACM Trans. Knowl. Discov. Data*, vol. 18, no. 9, pp. 1–28, 2024.

[18] R. Henriques, C. Antunes, and S. C. Madeira, "A structured view on pattern mining-based biclustering," *Pattern Recognition*, vol. 48, no. 12, pp. 3941–3958, 2015.

[19] B. Wang, L. Yao, J. Hu, and H. Li, "A new algorithm for convex biclustering and its extension to the compositional data," *Statistics in Biosciences*, vol. 15, no. 1, pp. 193–216, 2023.

[20] K. Kailing, H.-P. Kriegel, and P. Kröger, "Density-connected subspace clustering for high-dimensional data," in *Proc. SIAM International Conference on Data Mining (SDM'04)*, vol. 4, pp. 246–256, 2004.

[21] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," *Information Sciences*, vol. 622, pp. 178–210, 2023.

[22] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN," *ACM Transactions on Database Systems*, vol. 42, no. 3, pp. 1–21, 2017.

[23] M. D. Noronha, R. Henriques, S. C. Madeira, and L. E. Zárate, "Impact of metrics on biclustering solution and quality: A review," *Pattern Recognition*, vol. 127, p. 108612, 2022.

[24] L. Fisher and J. W. V. Ness, "Admissible clustering procedures," *Biometrika*, vol. 58, no. 1, pp. 91–104, 1971.

[25] L. Teng and L. Chan, "Discovering biclusters by iteratively sorting with weighted correlation coefficient in gene expression data," *Journal of Signal Processing Systems*, vol. 50, pp. 267–280, 2008.

[26] J. Xie, A. Ma, Y. Zhang, B. Liu, S. Cao, C. Wang, J. Xu, C. Zhang, and Q. Ma, "QUBIC2: a novel and robust biclustering algorithm for analyses and interpretation of large-scale RNA-Seq data," *Bioinformatics*, vol. 36, no. 4, pp. 1143–1149, 2019.

[27] B. Pontes, R. Giráldez, and J. S. Aguilar-Ruiz, "Biclustering on expression data: A review," *Journal of Biomedical Informatics*, vol. 57, pp. 163 – 180, 2015.

[28] N. Jain and C. A. Murthy, "A new estimate of mutual information based measure of dependence between two variables: properties and fast implementation," *International Journal of Machine Learning and Cybernetics*, vol. 7, pp. 857–875, 2016.

[29] N. Jain and S. Ghosh, "An unsupervised band selection method for hyperspectral images using mutual information based dependence index," in *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium, Kuala Lumpur, Malaysia*, pp. 783–786, 2022.

[30] N. Jain and C. A. Murthy, "Connectedness-based subspace clustering," *Knowledge and Information Systems*, vol. 58, no. 1, pp. 9–34, 2019.

[31] N. Jain, S. Ghosh, and C. A. Murthy, "RelDenClu: A Relative Density based Biclustering Method for identifying non-linear feature relations," *arXiv:Computer Vision and Pattern Recognition, Preprint, arXiv:1811.04661*, 2021.

[32] N. Jain and S. Ghosh, "Machine Learning Method to Discover the Novel Association of Vaccine and Vitamin A Supplement on Covid 19 Infection: A Biclustering Approach," in *2021 IEEE 18th India Council International Conference (INDICON), Guwahati, India*, pp. 1–6, 2021.

[33] N. Jain, S. Ghosh, and A. Ghosh, "A parameter free relative density based biclustering method for identifying non-linear feature relations," *Heliyon*, vol. 10, p. e34736, 2024.

[34] S. Ross, *Introductory Statistics*. Academic Press, 2017.

[35] P. Mitra, C. A. Murthy, and S. K. Pal, "Unsupervised feature selection using feature similarity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 301–312, 2002.

[36] C. Spearman, "The proof and measurement of association between two things," *International Journal of Epidemiology*, vol. 39, no. 5, pp. 1137–1150, 2010.

[37] W. H. Kruskal, "Ordinal measures of association," *Journal of the American Statistical Association*, vol. 53, no. 284, pp. 814–861, 1958.

[38] H. Abdi, "The Kendall rank correlation coefficient," *Encyclopedia of measurement and statistics*, vol. 2, pp. 508–510, 2007.

[39] S. Chatterjee, "A new coefficient of correlation," *Journal of the American Statistical Association*, vol. 116, no. 536, pp. 2009–2022, 2021.

[40] K. Fukumizu, A. Gretton, G. Lanckriet, B. Schölkopf, and B. K. Sriperumbudur, "Kernel Choice and Classifiability for RKHS Embeddings of Probability

Distributions," in *Advances in Neural Information Processing Systems* (Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, eds.), vol. 22, pp. 1750–1758, Curran Associates, Inc., 2009.

[41] A. Gretton, R. Herbrich, A. Smola, O. Bousquet, B. Schölkopf, and A. Hyvärinen, "Kernel methods for measuring independence," *Journal of Machine Learning Research*, vol. 6, no. 12, pp. 1750 – 1758, 2005.

[42] T. Wang, X. Dai, and Y. Liu, "Learning with Hilbert–Schmidt independence criterion: A review and new perspectives," *Knowledge-Based Systems*, vol. 234, p. 107567, 2021.

[43] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring Statistical Dependence with Hilbert-Schmidt Norms," in *Algorithmic Learning Theory*, pp. 63–77, Springer Berlin Heidelberg, 2005.

[44] D. Sejdinovic, B. Sriperumbudur, A. Gretton, and K. Fukumizu, "Equivalence of distance-based and RKHS-based statistics in hypothesis testing," *The Annals of Statistics*, vol. 41, no. 5, pp. 2263–2291, 2013.

[45] R. Gray, *Entropy and Information Theory*. Springer New York, 2013.

[46] T. van Erven and P. Harremos, "Rényi Divergence and Kullback-Leibler Divergence," *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 3797–3820, 2014.

[47] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. 25, pp. 723–773, 2012.

[48] B. Póczos, Z. Ghahramani, and J. Schneider, "Copula-based kernel dependency measures," in *Proceedings of the 29th International Conference on Machine Learning*, ICML'12, (Madison, WI, USA), pp. 1635–1642, Omnipress, 2012.

[49] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *Journal of machine learning research*, vol. 2, pp. 67–93, 2001.

[50] G. J. Székely and M. L. Rizzo, "Energy statistics: A class of statistics based on distances," *Journal of Statistical Planning and Inference*, vol. 143, no. 8, pp. 1249–1272, 2013.

[51] G. J. SzéKely and M. L. Rizzo, "Brownian distance covariance," *Annals of Applied Statistics*, vol. 3, no. 4, pp. 1236–1265, 2009.

[52] G. J. Székely and M. L. Rizzo, *The energy of data and distance correlation*. Chapman and Hall/CRC, 2023.

[53] D. N. Reshef, Y. A. Reshef, P. C. Sabeti, and M. Mitzenmacher, "An empirical study of the maximal and total information coefficients and leading measures of dependence," *The Annals of Applied Statistics*, vol. 12, no. 1, pp. 123 – 155, 2018.

[54] R. Heller, Y. Heller, and M. Gorfine, "A consistent multivariate test of association based on ranks of distances," *Biometrika*, vol. 100, pp. 503–510, 12 2012.

[55] C. R. Rao, *Karl Pearson Chi-Square Test The Dawn of Statistical Inference*, pp. 9–24. Birkhäuser Boston, 2002.

[56] R. Ash, *Information Theory*. Dover Books on Mathematics, Dover Publications, 2012.

[57] T. Cover and J. Thomas, *Elements of Information Theory*. Wiley, 2012.

[58] P. Beale, *Statistical Mechanics*. Academic Press, 2011.

[59] T. O. Kvålseth, "Entropy and Correlation: Some Comments," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 17, no. 3, pp. 517–519, 1987.

[60] Y. Y. Yao, "Information-theoretic measures for knowledge discovery and data mining," in *Entropy Measures, Maximum Entropy Principle and Emerging Applications*, vol. 119 of *Studies in Fuzziness and Soft Computing*, pp. 115–136, Springer Berlin Heidelberg, 2003.

[61] T. O. Kvålseth, "On Normalized Mutual Information: Measure Derivations and Properties," *Entropy*, vol. 19, no. 11, 2017.

[62] A. MartÍnez-UsÓ, F. Pla, J. M. Sotoca, and P. GarcÍa-Sevilla, "Clustering-based hyperspectral band selection using information measures," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 12, pp. 4158–4171, 2007.

[63] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, "Detecting novel associations in large data sets," *Science*, vol. 16, pp. 1518–1524, December 2011.

[64] M. Bouchard, A.-L. Jousselme, and P.-E. Doré, "A proof for the positive definiteness of the jaccard index matrix," *International Journal of Approximate Reasoning*, vol. 54, no. 5, pp. 615–626, 2013.

[65] S. Temma, M. Sugii, and H. Matsuno, "The Document Similarity Index based on the Jaccard Distance for Mail Filtering," in *2019 34th International Tech-*

nical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), pp. 1–4, 2019.

[66] A. Carass, S. Roy, A. Gherman, J. C. Reinhold, A. Jesson, T. Arbel, O. Maier, H. Handels, M. Ghafoorian, B. Platel, *et al.*, "Evaluating white matter lesion segmentations with refined Sørensen-Dice analysis," *Scientific reports*, vol. 10, no. 1, p. 8242, 2020.

[67] D. Müller, I. Soto-Rey, and F. Kramer, "Towards a guideline for evaluation metrics in medical image segmentation," *BMC Research Notes*, vol. 15, no. 1, p. 210, 2022.

[68] S. Karthik Viswanath, L. Naveen, R. Ananda, S. Rajalakshmi, and S. Angel Deborah, "Abstractive Text Summarizer: A Comparative Study on Dot Product Attention and Cosine Similarity," in *2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pp. 1–8, 2021.

[69] Q. Li, H. Kaneko, and L. Meng, "A cosine similarity-based token subsampling method for vision transformer in cloud computing," *Neural Computing and Applications*, vol. 37, no. 4, pp. 2627–2639, 2025.

[70] N. El Aboudi and L. Benhlima, "Review on wrapper feature selection approaches," in *2016 international conference on engineering & MIS (ICEMIS)*, pp. 1–5, IEEE, 2016.

[71] M. Cherrington, F. Thabtah, J. Lu, and Q. Xu, "Feature Selection: Filter Methods Performance Challenges," in *2019 International Conference on Computer and Information Sciences (ICCIS)*, pp. 1–4, 2019.

[72] S. Solorio-Fernández, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "A review of unsupervised feature selection methods," *Artificial Intelligence Review*, vol. 53, pp. 907–948, 2020.

[73] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM computing surveys (CSUR)*, vol. 50, no. 6, 2017.

[74] A. J. Ferreira and M. A. Figueiredo, "An unsupervised approach to feature discretization and selection," *Pattern Recognition*, vol. 45, no. 9, pp. 3048–3060, 2012.

[75] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on communications*, vol. 28, no. 1, pp. 84–95, 2003.

[76] Y. Li, B.-L. Lu, and Z.-F. Wu, "Hierarchical fuzzy filter method for unsupervised feature selection," *Journal of Intelligent & Fuzzy Systems*, vol. 18, no. 2, p. 157169, 2007.

[77] R. Varshavsky, A. Gottlieb, M. Linial, and D. Horn, "Novel unsupervised feature filtering of biological data," *Bioinformatics*, vol. 22, no. 14, pp. e507–e513, 2006.

[78] M. Banerjee and N. R. Pal, "Feature selection with SVD entropy: Some modification and extension," *Information Sciences*, vol. 264, pp. 118–134, 2014.

[79] S. Wang, W. Pedrycz, Q. Zhu, and W. Zhu, "Unsupervised feature selection via maximum projection and minimum redundancy," *Knowledge-Based Systems*, vol. 75, pp. 19–29, 2015.

[80] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[81] Z. A. Zhao and H. Liu, *Spectral Feature Selection for Data Mining*. Chapman & Hall/CRC, 2018.

[82] W. Zhou, C. Wu, Y. Yi, and G. Luo, "Structure preserving non-negative feature self-representation for unsupervised feature selection," *IEEE Access*, vol. 5, pp. 8792–8803, 2017.

[83] C. Tang, X. Zhu, J. Chen, P. Wang, X. Liu, and J. Tian, "Robust graph regularized unsupervised feature selection," *Expert Systems with Applications*, vol. 96, pp. 64–76, 2018.

[84] Y. Shi, J. Miao, Z. Wang, P. Zhang, and L. Niu, "Feature selection with $\ell_{2,1-2}$ regularization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4967–4982, 2018.

[85] X. Wang, X. Zhang, Z. Zeng, Q. Wu, and J. Zhang, "Unsupervised spectral feature selection with l1-norm graph," *Neurocomputing*, vol. 200, pp. 47–54, 2016.

[86] S. Karami, F. Saberi-Movahed, P. Tiwari, P. Marttinen, and S. Vahdati, "Unsupervised feature selection based on variance–covariance subspace distance," *Neural Networks*, vol. 166, pp. 188–203, 2023.

[87] P. Dhal and C. Azad, "A comprehensive survey on feature selection in the various fields of machine learning," *Applied Intelligence*, vol. 52, p. 4543–4581, 2021.

[88] S. Chormunge and S. Jena, "Correlation based feature selection with clustering for high dimensional data," *Journal of Electrical Systems and Information Technology*, vol. 5, no. 3, pp. 542–549, 2018.

[89] D. Paul, R. Su, M. Romain, V. Sébastien, V. Pierre, and G. Isabelle, "Feature selection for outcome prediction in oesophageal cancer using genetic algorithm and random forest classifier," *Computerized Medical Imaging and Graphics*, vol. 60, pp. 42–49, 2017.

[90] H. Chamlal, A. Benzmane, and T. Ouaderhman, "Elastic net-based high dimensional data selection for regression," *Expert Systems with Applications*, vol. 244, p. 122958, 2024.

[91] R. Das, G. Kasieczka, and D. Shih, "Feature selection with distance correlation," *Physical Review D*, vol. 109, no. 5, p. 054009, 2024.

[92] G. Roffo, S. Melzi, U. Castellani, A. Vinciarelli, and M. Cristani, "Infinite feature selection: A graph-based feature filtering approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 12, pp. 4396–4410, 2021.

[93] Z. Lin and F. Han, "On boosting the power of chatterjee's rank correlation," *Biometrika*, vol. 110, no. 2, pp. 283–299, 2022.

[94] S. C. Madeira and A. L. Oliveira, "Biclustering algorithms for biological data analysis: a survey," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 1, no. 1, pp. 24–45, 2004.

[95] K. Sim, V. Gopalkrishnan, A. Zimek, and G. Cong, "A survey on enhanced subspace clustering," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 332–397, 2013.

[96] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 1, pp. 1–58, 2009.

[97] J. A. Hartigan, "Direct clustering of a data matrix," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 123–129, 1972.

[98] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra, "Minimum sum-squared residue co-clustering of gene expression data," in *Proceedings of the 2004 SIAM international conference on data mining*, pp. 114–125, SIAM, 2004.

[99] Q. Huang, J. Yang, X. Feng, A. W.-C. Liew, and X. Li, "Automated trading point forecasting based on bicluster mining and fuzzy inference," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 2, pp. 259–272, 2020.

[100] Q. Huang, D. Tao, X. Li, L. Jin, and G. Wei, "Exploiting local coherent patterns for unsupervised feature ranking," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 6, pp. 1471–1482, 2011.

[101] Q. Huang, T. Wang, D. Tao, and X. Li, "Biclustering learning of trading rules," *IEEE Transactions on Cybernetics*, vol. 45, no. 10, pp. 2287–2298, 2015.

[102] L. Parsons, E. Haque, and H. Liu, "Subspace Clustering for High Dimensional Data: A Review," *ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets*, vol. 6, no. 1, pp. 90–105, 2004.

[103] F. Liu, Y. Yang, X. S. Xu, and M. Yuan, "MESBC: A novel mutually exclusive spectral biclustering method for cancer subtyping," *Computational Biology and Chemistry*, vol. 109, p. 108009, 2024.

[104] P. Carmona-Saez, R. D. Pascual-Marqui, F. Tirado, J. M. Carazo, and A. Pascual-Montano, "Biclustering of gene expression data by non-smooth non-negative matrix factorization," *BMC Bioinformatics*, vol. 7, no. 1, p. 78, 2006.

[105] A. José-García, J. Handl, W. Gómez-Flores, and M. Garza-Fabre, "An evolutionary many-objective approach to multiview clustering using feature and relational data," *Applied Soft Computing*, vol. 108, pp. 1568–4946, 2021.

[106] T. Yun and G.-S. Yi, "Biclustering for the comprehensive search of correlated gene expression patterns using clustered seed expansion," *BMC Genomics*, vol. 14, 2013.

[107] G. Getz, E. Levine, and E. Domany, "Coupled two-way clustering analysis of gene microarray data," *Proceedings of the National Academy of Sciences*, vol. 97, no. 22, pp. 12079–12084, 2000.

[108] H. A. Ahmed, P. Mahanta, D. K. Bhattacharyya, and J. K. Kalita, "Shifting-and-scaling correlation based biclustering algorithm," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 6, pp. 1239–1252, 2014.

[109] A. Tanay, R. Sharan, and R. Shamir, "Discovering statistically significant biclusters in gene expression data," *Bioinformatics*, vol. 18, pp. S136–S144, 2002.

[110] A. O. Laura Lazzeroni, "Plaid models for gene expression data," *Statistica Sinica*, vol. 12, no. 1, pp. 61–86, 2002.

[111] Q. Huang, X. Huang, Z. Kong, X. Li, and D. Tao, "Bi-phase evolutionary searching for biclusters in gene expression data," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 803–814, 2019.

[112] O. Maatouk, E. Ayari, H. Bouziri, and W. Ayadi, "Bobea: A bi-objective biclustering evolutionary algorithm for genome-wide association analysis," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 344–347, 2022.

[113] R. Galindo-Hernández, K. Rodríguez-Vázquez, E. Galán-Vásquez, and C. I. Hernández Castellanos, "Online-adjusted evolutionary biclustering algorithm to identify significant modules in gene expression data," *Briefings in Bioinformatics*, vol. 26, no. 1, p. bbae681, 2025.

[114] P. Swathypriyadharsini and K. Premalatha, "Hybrid Cuckoo Search with Clonal Selection for Triclustering Gene Expression Data of Breast Cancer," *IETE Journal of Research*, vol. 69, no. 5, pp. 2328–2336, 2023.

[115] S. Mitra and H. Banka, "Multi-objective evolutionary biclustering of gene expression data," *Pattern Recognition*, vol. 39, no. 12, pp. 2464–2477, 2006.

[116] F. Divina and J. S. Aguilar-Ruiz, "Biclustering of expression data with evolutionary computation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 5, pp. 590–602, 2006.

[117] C. A. Gallo, J. A. Carballido, and I. Ponzoni, "BiHEA: A Hybrid Evolutionary Approach for Microarray Biclustering," in *Advances in Bioinformatics and Computational Biology: 4th Brazilian Symposium on Bioinformatics*, pp. 36–47, Springer Berlin Heidelberg, 2009.

[118] K. Seridi, L. Jourdan, and E. G. Talbi, "Multi-objective evolutionary algorithm for biclustering in microarrays data," in *2011 IEEE Congress of Evolutionary Computation (CEC)*, pp. 2593–2599, 2011.

[119] J. Sun and Q. Huang, "Two stages biclustering with three populations," *Biomedical Signal Processing and Control*, vol. 79, p. 104182, 2023.

[120] A. José-García, J. Jacques, V. Sobanski, and C. Dhaenens, "Metaheuristic biclustering algorithms: From state-of-the-art to future opportunities," *ACM Comput. Surv.*, vol. 56, no. 3, pp. 1–38, 2023.

[121] X. Peng, J. Feng, J. T. Zhou, Y. Lei, and S. Yan, "Deep subspace clustering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 12, pp. 5509–5521, 2020.

[122] L. Zhou, B. Xiao, X. Liu, J. Zhou, E. R. Hancock, *et al.*, "Latent distribution preserving deep subspace clustering," in *28th International Joint Conference on Artificial Intelligence*, pp. 4440 – 4446, 2019.

[123] P. Zhou, Y. Hou, and J. Feng, "Deep adversarial subspace clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1596–1604, 2018.

[124] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, "Deep subspace clustering networks," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, pp. 23–32, Curran Associates, Inc., 2017.

[125] G. Moise, J. Sander, and M. Ester, "Robust projected clustering," *Knowledge and Information Systems*, vol. 14, no. 3, pp. 273–298, 2008.

[126] H.-P. Kriegel, P. Kroger, M. Renz, and S. Wurst, "A generic framework for efficient subspace clustering of high-dimensional data," in *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, pp. 250–257, IEEE Computer Society, 2005.

[127] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," *SIGMOD Record*, vol. 28, no. 2, pp. 61–72, 1999.

[128] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," *SIGMOD Record*, vol. 27, no. 2, pp. 94–105, 1998.

[129] S. K. Solanki and J. T. Patel, "A survey on association rule mining," in *2015 Fifth International Conference on Advanced Computing & Communication Technologies*, pp. 212–216, 2015.

[130] A. K. H. Tung, X. Xu, and B. C. Ooi, "CURLER: Finding and Visualizing Nonlinear Correlation Clusters," in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pp. 467–478, 2005.

[131] M. Greenacre, P. J. Groenen, T. Hastie, A. I. D'Enza, A. Markos, and E. Tuzhilina, "Principal component analysis," *Nature Reviews Methods Primers*, vol. 2, no. 1, p. 100, 2022.

[132] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley-India, 2006.

[133] S.-K. Ng, "Recent developments in expectation-maximization methods for analyzing complex data," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 5, no. 6, pp. 415–431, 2013.

[134] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition.* Springer Series in Statistics, Springer New York, 2009.

[135] Y. Ren, C. Domeniconi, G. Zhang, and G. Yu, "A weighted adaptive mean shift clustering algorithm," in *Proceedings of the 2014 SIAM International Conference on Data Mining*, pp. 794–802, SIAM, 2014.

[136] M. Matabuena, J. C. Vidal, O. H. M. Padilla, and D. Sejdinovic, "Kernel biclustering algorithm in Hilbert spaces," *Advances in Data Analysis and Classification*, pp. 1–42, 2025.

[137] Y. Cheng and G. M. Church, "Biclustering of expression data," in *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pp. 93–103, 2000.

[138] Y. Chen, C. Lei, C. Li, H. Ma, and N. H. and, "Robust convex biclustering with a tuning-free method," *Journal of Applied Statistics*, vol. 52, no. 2, pp. 271–286, 2025.

[139] S. Hochreiter, U. Bodenhofer, M. Heusel, A. Mayr, A. Mitterecker, A. Kasim, T. Khamiakova, S. Van Sanden, D. Lin, W. Talloen, L. Bijnens, H. W. H. Göhlmann, Z. Shkedy, and D.-A. Clevert, "FABIA: factor analysis for bicluster acquisition," *Bioinformatics*, vol. 26, no. 12, pp. 1520–1527, 2010.

[140] J. S. Aguilar-Ruiz, "Shifting and scaling patterns from gene expression data," *Bioinformatics*, vol. 21, no. 20, pp. 3840–3845, 2005.

[141] H. A. Ahmed, P. Mahanta, D. K. Bhattacharyya, and J. K. Kalita, "Shifting-and-scaling correlation based biclustering algorithm," *IEEE/ACM Transactions*

on Computational Biology and Bioinformatics, vol. 11, no. 6, pp. 1239–1252, 2014.

[142] S. Bergmann, J. Ihmels, and N. Barkai, "Iterative signature algorithm for the analysis of large-scale gene expression," *Physical Review. E*, vol. 67, no. 3, p. 031902, 2003.

[143] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini, "Discovering local structure in gene expression data: The order-preserving submatrix problem," *Journal of Computational Biology*, vol. 10, no. 3-4, pp. 373–384, 2003.

[144] G. Li, Q. Ma, H. Tang, A. H. Paterson, and Y. Xu, "QUBIC: a qualitative biclustering algorithm for analyses of gene expression data," *Nucleic Acids Research*, vol. 37, no. 15, p. e101, 2009.

[145] Z. Wang, G. Li, R. W. Robinson, and X. Huang, "UniBic: Sequential row-based biclustering algorithm for analysis of gene expression data," *Scientific Reports*, vol. 6, p. 23466, 2016.

[146] X. Liu, T. Yu, X. Zhao, C. Long, R. Han, Z. Su, and G. Li, "ARBic: an all-round biclustering algorithm for analyzing gene expression data," *NAR Genomics and Bioinformatics*, vol. 5, no. 1, p. lqad009, 2023.

[147] J. Li, Q. Mei, C. Yang, N. Zhu, and G. Li, "Transbic: bucket trend-preserving biclustering for finding local and interpretable expression patterns," *Briefings in Bioinformatics*, vol. 26, no. 1, p. bbaf050, 2025.

[148] X. He and L. Moreira-Matias, "Robust Continuous Co-Clustering," *arXiv preprint arXiv:1802.05036*, 2018.

[149] F. Alqadah, C. Reddy, J. Hu, and H. Alqadah, "Biclustering neighborhood-based collaborative filtering method for top-n recommender systems," *Knowledge and Information Systems*, vol. 44, pp. 475–491, 2014.

[150] D. Ienco, R. G. Pensa, and R. Meo, "Parameter-free hierarchical co-clustering by n-ary splits," in *Machine Learning and Knowledge Discovery in Databases* (W. Buntine, M. Grobelnik, D. Mladenić, and J. Shawe-Taylor, eds.), pp. 580–595, Springer Berlin Heidelberg, 2009.

[151] A. G. Dalgleish, "The relevance of non-linear mathematics (chaos theory) to the treatment of cancer, the role of the immune response and the potential for vaccines," *QJM: An International Journal of Medicine*, vol. 92, no. 6, pp. 347–359, 1999.

[152] R. M. A. Urbach, *Footprints of chaos in the markets: analyzing non-linear time series in financial markets and other real systems.* Prentice Hall, London, 2000.

[153] A. Dionisio, R. Menezes, and D. A. Mendes, "Entropy-based independence test," *Nonlinear Dynamics*, vol. 44, no. 1, pp. 351–357, 2006.

[154] G. A. Darbellay and D. Wuertz, "The entropy as a tool for analysing statistical dependences in financial time series," *Physica A: Statistical Mechanics and its Applications*, vol. 287, no. 3-4, pp. 429 – 439, 2000.

[155] C. W. Granger, E. Maasoumi, and J. Racine, "A dependence metric for possibly nonlinear processes," *Journal of Time Series Analysis*, vol. 25, no. 5, pp. 649–669, 2004.

[156] A. Dionisio, R. Menezes, and D. A. Mendes, "Mutual information: a measure of dependency for nonlinear time series," *Physica A: Statistical Mechanics and its Applications*, vol. 344, no. 1, pp. 326–329, 2004.

[157] A. Kraskov and P. Grassberger, "MIC: Mutual Information based hierarchical Clustering," in *Information Theory and Statistical Learning*, ch. 5, pp. 101–123, Springer U.S., 2009.

[158] N. Simon and R. Tibshirani, "Comment on 'Detecting Novel Associations In Large Data Sets' by Reshef Et Al, Science Dec 16, 2011," *arXiv:Statistics.Methodology, Preprint, arXiv:1401.7645*, 2014.

[159] D. Pál, B. Póczos, and C. Szepesvári, "Estimation of Renyi Entropy and Mutual Information Based on Generalized Nearest-Neighbor Graphs," in *Advances in Neural Information Processing Systems (NIPS 2010)*, pp. 1849–1857, Curran Associates, Inc., 2010.

[160] E. Schaffernicht and H.-M. Gross, "Weighted mutual information for feature selection," in *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part II 21*, pp. 181–188, Springer, 2011.

[161] A. Dey, S. Ghosh, and A. Ghosh, "Band elimination for dimensionality reduction of hyperspectral images using mutual information," in *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, pp. 2055–2058, 2020.

[162] H. Kantz, J. Kurths, and G. Mayer-Kress, *Nonlinear Analysis of Physiological Data*. Springer Berlin Heidelberg, 2012.

[163] G. Lugosi and A. Nobel, "Consistency of data-driven histogram methods for density estimation and classification.," *The Annals of Statistics*, vol. 24, no. 2, pp. 687–706, 1996.

[164] M. J. B. Appel and R. P. Russo, "The connectivity of a graph on uniform points on $[0,1]^d$," *Statistics & Probability Letters*, vol. 60, no. 4, pp. 351–357, 2002.

[165] D. Darling, "On a class of problems related to the random division of an interval," *The Annals of Mathematical Statistics*, vol. 24, pp. 239–253, 1953.

[166] P. Deheuvels, "Strong limit theorems for maximal spacings from a general univariate distribution," *The Annals of Probability*, vol. 12, no. 4, pp. 1181–1193, 1984.

[167] V. Vapnik, *The Nature of Statistical Learning Theory*. Information Science and Statistics, Springer New York, 2013.

[168] I. Csiszar and J. Korner, *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Orlando, FL, USA: Academic Press, Inc., 1982.

[169] I. Bairamov, A. Berred, and A. Stepanov, "Limit results for ordered uniform spacings," *Statistical Papers*, vol. 51, no. 1, pp. 227–240, 2010.

[170] A. Datta, S. Ghosh, and A. Ghosh, "Clustering based band selection for hyperspectral images," in *2012 International Conference on Communications, Devices and Intelligent Systems (CODIS)*, pp. 101–104, 2012.

[171] A. Datta, S. Ghosh, and A. Ghosh, "PCA, Kernel PCA and Dimensionality Reduction in Hyperspectral Images," in *Advances in Principal Component Analysis: Research and Development*, pp. 19–46, Springer Singapore, 2018.

[172] D. Yang and W. Bao, "Group Lasso-Based Band Selection for Hyperspectral Image Classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, pp. 2438–2442, 2017.

[173] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, "220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3," Sep 2015.

[174] M. Grana, M. A. Veganzons, and B. Ayerdi, "Hyperspectral remote sensing scenes," *Grupo de Inteligencia Computacional (GIC)*, 2020.

[175] I.-C. Yeh, "Default of credit card clients." UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C55S3H.

[176] V. Realinho, M. V. Martins, J. Machado, and L. Baptista, "Predict Students' Dropout and Academic Success." UCI Machine Learning Repository, 2021. DOI: https://doi.org/10.24432/C5MC89.

[177] R. Bock, "MAGIC Gamma Telescope." UCI Machine Learning Repository, 2007. DOI: https://doi.org/10.24432/C52C8B.

[178] W. Wolberg, "Breast Cancer Wisconsin (Original)." UCI Machine Learning Repository, 1990. DOI: https://doi.org/10.24432/C5HP4Z.

[179] V. Arzamasov, "Electrical Grid Stability Simulated Data." UCI Machine Learning Repository, 2018. DOI: https://doi.org/10.24432/C5PG66.

[180] H.-P. Kriegel and A. Zimek, "Subspace clustering, ensemble clustering, alternative clustering, multiview clustering: what can we learn from each other?," in *Proceedings of the ACM SIGKDD Workshop MultiClust*, 2010.

[181] R. F. Ling, "A probability theory of cluster analysis," *Journal of the American Statistical Association*, vol. 68, no. 341, pp. 159–164, 1973.

[182] D. P. Mandal and C. A. Murthy, "Selection of alpha for alpha-hull in $\mathbf{R}^2$," *Pattern Recognition*, vol. 30, no. 10, pp. 1759 – 1767, 1997.

[183] A. C. Müller, S. Nowozin, and C. H. Lampert, *Information Theoretic Clustering Using Minimum Spanning Trees*, pp. 205–215. Springer Berlin Heidelberg, 2012.

[184] J. M. Steele and T. L. Snyder, "Worst-case growth rates of some classical problems of combinatorial optimization," *SIAM Journal on Computing*, vol. 18, no. 2, pp. 278–287, 1989.

[185] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[186] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pp. 89–98, ACM, 2003.

[187] P. Orzechowski, A. Pańszczyk, X. Huang, and J. H. Moore, "runibic: a Bioconductor package for parallel row-based biclustering of gene expression data," *Bioinformatics*, vol. 34, no. 24, pp. 4302–4304, 2018.

[188] M. Hassani, M. Hansen, E. Müller, I. Assent, S. Günnemann, T. Jansen, and T. Seidl, "subspace: Interface to OpenSubspace," 2015. R package [Online], https://cran.r-project.org/web/packages/subspace/index.html.

[189] J. K. Gupta, S. Singh, and N. K. Verma, "MTBA: MATLAB Toolbox for Biclustering Analysis," in *IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions, IIT Kanpur, India*, pp. 94–97, IEEE, 2013.

[190] Y. Liu, X. Li, Y. Feng, L. Zhao, and W. Z. and, "Representativeness and redundancy-based band selection for hyperspectral image classification‘," *International Journal of Remote Sensing*, vol. 42, no. 9, pp. 3534–3562, 2021.

[191] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[192] J. E. Chacón and A. I. Rastrojo, "Minimum adjusted Rand index for two clusterings of a given size," *Advances in Data Analysis and Classification*, vol. 17, no. 1, p. 125–133, 2023.

[193] The World Bank, "World development indicators," 2015. https://datacatalog.worldbank.org/dataset/world-development-indicators.

[194] S. Mukherjee, "COVID-19 Data: Adding World Development Indicators," 2020. https://www.kaggle.com/code/sambitmukherjee/.

[195] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for DNA microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[196] E. Dong, H. Du, and L. Gardner, "An interactive web-based dashboard to track covid-19 in real time," *The Lancet Infectious Diseases*, vol. 20, no. 5, pp. 533–534, 2020.

[197] E. Mathieu, H. Ritchie, L. Rodés-Guirao, C. Appel, C. Giattino, J. Hasell, B. Macdonald, S. Dattani, D. Beltekian, E. Ortiz-Ospina, and M. Roser, "Coronavirus Pandemic (COVID-19)," *Our World in Data*, 2020. https://ourworldindata.org/coronavirus.

[198] World Health Organization, "Immunization, vaccines and biologicals," 2020. https://immunizationdata.who.int/.

[199] R. Sorensen, R. Barber, D. Pigott, A. Carter, C. Spencer, S. Ostroff, R. Reiner, C. Abbafati, C. Adolph, A. Allorant, *et al.*, "Variation in the COVID-19 infection-fatality ratio by age, time, and geography during the pre-vaccine era: A systematic analysis," *The Lancet*, vol. 399, no. 10334, pp. 1469–1488, 2022.

[200] S. S. Chakrabarti, U. Kaur, A. Singh, S. Chakrabarti, M. Krishnatreya, B. K. Agrawal, A. Mittal, A. Singh, R. Khanna, I. S. Gambhir, K. Jin, and S. Chakrabarti, "Of Cross-immunity, Herd Immunity and Country-specific Plans: Experiences from COVID-19 in India," *Aging and disease*, vol. 11, no. 6, pp. 1339–1344, 2020.

[201] N. Curtis, A. Sparrow, T. A. Ghebreyesus, and M. G. Netea, "Considering BCG vaccination to reduce the impact of COVID-19," *The Lancet*, vol. 395, no. 10236, pp. 1545–1546, 2020.

[202] New Scientist, "Will the spread of covid-19 be affected by changing seasons?," 2020. https://www.newscientist.com/article/2239380-will-the-spread-of-covid-19-be-affected-by-changing-seasons/.

[203] C. D. Rickett, K. J. Maschhoff, and S. R. Sukumar, "Does Tetanus vaccination contribute to reduced severity of the COVID-19 infection?," *Medical Hypotheses*, vol. 146, p. 110395, 2021.

[204] C. Xu, W. Xu, and K. Jing, "Fast algorithms for singular value decomposition and the inverse of nearly low-rank matrices," *National Science Review*, vol. 10, no. 6, p. nwad083, 2023.

Namita Jain
04/06/2025