JULY 2019

M.Sc. in ELECTRONICS AND COMPUTER ENGINEERING

SALEEM ISMAEL SADEQ

REPUBLIC OF TURKEY
GAZİANTEP UNIVERSITY
GRADUATE SCHOOL OF
NATURAL & APPLIED SCIENCES

SUBSPACE CONSTRAINT CLUSTERING FOR SEMI-SUPERVISED SPARSE DATA

M.Sc. THESIS

IN

ELECTRONICS AND COMPUTER ENGINEERING

BY

SALEEM ISMAEL SADEQ

JULY 2019

Subspace Constraint Clustering for Semi- Supervised Sparse Data

M.Sc. Thesis

in

Electronics and Computer Engineering

Gaziantep University

Supervisor

Prof. Dr. Gölge ÖGÜCÜ YETKIN

By

Saleem Ismael SADEQ

July 2019

REPUBLIC OF TURKEY
GAZİANTEP UNIVERSITY
GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES
ELECTRONICS AND COMPUTER ENGINEERING DEPARTMENT

Name of the thesis: Subspace Constraint Clustering for Semi- Supervised Sparse Data

Name of the student: Saleem Ismael Sadeq

Exam date: 04.07.2019

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. A. Necmeddin YAZICI
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. ERGUN ERÇELEBİ
Head of Department

This is to certify that we have read this thesis and that in our consensus opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science

Prof. Dr. Gölge ÖGÜCÜ YETKİN
Supervisor

Examining Committee Members:	Signature
Prof. Dr. Gölge ÖGÜCÜ YETKİN	…….
Assist. Prof. Dr. Mohammed MADI	……
Assist. Prof. Dr. Nurdal WATSUJİ	…….

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Saleem Ismael SADEQ

ABSTRACT

SUBSPACE CONSTRAINT CLUSTERING FOR SEMI- SUPERVISED SPARSE DATA

SADEQ, Saleem Ismael
M.Sc. Thesis in Electronics and Computer Eng.
Supervisor: Prof. Dr. Gölge ÖGÜCÜ YETKIN
July 2019
71 pages

The advancement in computational processes has brought about high-dimensional data. Processing and analyzing this type of data requires special attention as several persistent problems are often faced. Data clustering/grouping has been widely used in data mining and machine learning applications, as it provides an idea on the underlying knowledge presented by the datasets. *k*-means is commonly used in normal clustering applications due to its simplicity and effectiveness; yet, it usually fails with high-dimensional datasets. This is mainly caused by the curse of dimensionality and possible noise within the dataset. Dimensionality reduction can be applied to alleviate this problem on both space and subspace levels. Nevertheless, subspace dimensionality reduction has attracted more attention in recent studies due to its robustness and high performance. Thus, in this research, it has been decided to investigate soft subspace clustering for sparse data. Four algorithms have been chosen, based on their mechanism orientation, to compare their performance with *k*-means in clustering hard-to-cluster datasets. MATLAB 2018b was used to develop a Graphical User Interface (GUI) for running all clustering processes and producing both numerical and visual results. It has been found that the performance of *k*-means can be dramatically improved through incorporating soft subspace computing, and particularly, ReliefF algorithm, in this study, could improve the performance to more than 50% in some cases.

Keywords: Clustering, dimensionality reduction, *k*-means, soft-subspace constraint, sparse data

ÖZET

UZAY ALTI KISITLI KÜMELEME İÇİN YARI DENETİMLİ SEYREK VERİLER

SADEQ, Saleem Ismael
Yüksek Lisans Tezi, Elektronik ve Bilgisayar Müh. Bölümü
Tez Yöneticisi:: Prof. Dr. Gölge ÖGÜCÜ YETKIN
Temmuz 2019, 71 sayfa

Kompütasyonel süreçlerde kaydedilen ilerleme yüksek boyutlu verileri meydana getirmiştir. Bu tür verilerin işlenmesi ve analiz edilmesi ise ısrarla tekrarlanan birçok problemle sıkça karşılaşıldığından özel dikkat gerektirir. Veri kümeleme/gruplama, veri kümeleri tarafından sunulan temel bilgiler hakkında bir fikir verdiğinden, veri madenciliği ve makine öğrenimi uygulamalarında yaygın olarak kullanılmaktadır. *k*-ortalama (*k*-means) algoritması, basitliği ve etkinliği nedeniyle normal kümeleme uygulamalarında yaygın olarak kullanılır; ancak, genellikle yüksek boyutlu veri kümelerinde başarısız olur. Buna temel olarak veri setindeki "curse of dimensionality" yani boyutla birlikte gelen problemler ve de olası gürültü neden olur. Bu problemi gidermek için, hem uzay hem de alt uzay seviyelerinde, boyut azaltma/indirgeme uygulanabilir. Bununla birlikte, alt uzay boyut azaltma, son zamanlardaki çalışmalarda, sağlamlığı ve yüksek performansı nedeniyle dikkatleri üzerine daha fazla çekmiştir. Bu nedenle, bu araştırmada, seyrek veriler için esnek alt uzay kümelemesinin incelenmesine karar verilmiştir. Mekanizma yönelimlerine bağlı olarak, kümelenmesi zor olan veri kümelerinin kümelenmelerini *k*-ortalama performansı ile karşılaştırmak için dört algoritma seçilmiştir. Tüm kümeleme işlemlerini çalıştırmak ve hem sayısal hem de görsel sonuçlar üretmek için bir Grafik Kullanıcı Arabirimi(GUI), MATLAB2018b kullanılarak geliştirildi. *k*-ortalama performansının bilhassa ReliefF algoritması olmak üzere esnek alt uzay hesaplama dahil edilerek önemli ölçüde yükseltilebileceği ve bu çalışmamızda ise bazı durumlarda performansı %50'den daha fazla artırabildiği bulunmuştur.

Anahtar Kelimeler: Kümeleme, boyut azaltma/indirgeme, *k*-means/*k*-ortalama, esnek alt uzay kısıtlaması, seyrek veri

ACKNOWLEDGMENT

I wish to express my deepest thanks and gratitude to my supervisor **Prof. Dr. Gölge Ögücü Yetkin** for her guidance, advice, encouragements and insight throughout the researchI would also like to thank my parents and my family for giving me courage, as well as being patient with me, and supporting me throughout my life.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

LIST OF SYMBOLS/ABBREVIATIONS

CBL	Cell-based clustering
CLIQUE	Clustering in quest
COSA	Clustering on subsets of attribute
CLTree	Clustering tree
CFS	Constraint Feature Selection
DOC	Density-based optimal clustering
ENCLUS	Entropy clustering
FINDIT	Fast and Intelligent Subspace Clustering Algorithm using Dimension Voting
GUI	Graphical User Interface
IFS	Infinite Feature Selection
LAC	Local Adaptive Clustering
PROCLUS	Projected clustering
SYNCLUS	Synthesized clustering
2-D	Two-dimensional
UDFS	Unsupervised Discriminative Feature Selection

CHAPTER 1

INTRODUCTION

1.1 General

The rapid development of information systems and their applications has expanded both the amount of data produced and needed. Recently, massive amounts of data are being used in many applications ranging from scientific to leisure uses. This type of data is usually required by complex systems, which have rapidly increased both the amount of the required data and the complexity of its analysis. Any acquired data in its raw format is generally neither efficient nor compatible with the operational process of modern systems. The main reason for such implication is both time and accuracy of the targeted outcomes. Raw acquired data, in most circumstances, undergoes a series of preparation, filtering, classification and clustering processes prior to the final analysis. Such processes assist in both mining the right data as well as producing efficient outcomes. In many applications, data can be represented by a matrix that provides both data samples and the intrinsic attributes of these samples [1]. This data presentation can be extremely complicated due to the high dimensionality [2]. In data analysis, dimensionality poses a serious limitation to both time and storage capability, which hinder the efficiency of the outcomes [3].

Contemporary trends in data analysis such as machine learning have emerged as a response to obstinate needs for analysis accuracy and efficiency. Labeling data samples and discovering distinguished patterns within them have been applied since a very long time ago [4]. Human beings, for various applications, practiced the initial attempts manually. However, the tremendous amount of data and the essential complexity dictates the use of a machine for this kind of purpose. Recently, with the advances in machine learning and data mining techniques, classifying data patterns and clustering similar data have become possible even without prior knowledge of data labels. This is usually done by retrieving the common underlying knowledge that generally regarded unsupervised learning process [5]. In data clustering application,

datasets are partitioned into groups referred to by clusters. These clusters are essentially composed based on the similarity of data objects; meaning that similar objects are gathered in one group. The idea of clustering similar objects requires a mechanism for measuring the similarity/dissimilarity of data objects to decide whether they belong to a specific class or to another one. In addition, in some cases, such as sparse data, which is recognized by its essential intermittent recording fashion (the existence of gaps in data samples) [6], prior to applying any measuring process, the main concern is guided towards space or subspace reduction to alleviate the caused problem [7]. It is possible to come across with, in data mining and machine learning applications, data samples, which are both sparse and high-dimensional. In this case, the sparsity occurs due to the intermittent acquisition process, such as in many sensor-collected data, while the dimensionality is composed by the existence of many features describing the data samples [8]. Both characteristics, sparsity, and high-dimensionality, impose restrictions on both process and analyzing data samples. Sparsity imposes the lack of data objects that deteriorates the efficiency of analysis algorithms, which becomes worse when high-dimensionality is a matter of concern [9]. Dimensionality reduction can be achieved through the deployment of dimensions themselves as knowledge in the clustering process. Clustering is generally classified as an unsupervised learning method; however, when dimensions or features are used for gaining some insight on dimensionality reduction, the process becomes semi-supervised in its essence [10]. This semi-supervised mechanism has recently been applied to clustering of high-dimensional or sparse data to produce better outcomes.

Features deployment can generally be categorized into two main branches: feature extraction and selection. The idea of feature extraction is based on retrieving some features from the dimensions themselves to reduce the dimensionality and perform the clustering process in a more efficient way. On the other hand, it is possible to measure the relevance of feature or a set of features to achieve the required reduction, and in this case, the mechanism is called feature selection. Both methods require some sort of knowledge to obtain satisfying outcomes. In reality, it is nearly impossible to manually label large datasets. Instead, analysts use both labeled and unlabeled datasets in the processing, which refers back to the same idea of semi-supervised learning essence [11]. As stated earlier, features can be used for dimensionality reduction. However, when deploying feature extraction or feature selection directly to high-dimensional data, the mechanism may stumble in the dimensionality curse, as there

are too many dimensions irrelevant but close in values. Recently, the concept of subspace clustering has been proposed as a response to the sparse data clustering problem [12]. The main idea here is to avoid the focus from considering the entire space, due to the high-dimensionality, and apply the clustering concept on subspace-based. In this case, the clustering procedure is achieved by clustering in subspace instead of the whole space; this obviously is a way much easier and more efficient than considering the entire dense dimension. It is possible using subspace clustering to assign a specific weight for each dimension in the cluster, which refers to the degree of importance of the dimension and based on this, similar dimensions can be clustered together [13, 14].

In subspace clustering application, it is possible to maximize the performance of an algorithm by minimizing its objective function. The objective function here refers to the possible error during the updating process of dimensional weights assignment, which continues until convergence [15]. This research area is considered an up-to-date field and requires more investigation from different perspectives.

1.2 Research Problem

The application of the common clustering algorithm to sparse data can be both cumbersome and inefficient, especially if the considered data samples are high-dimensional. The essence of missing data objects due to sparsity deteriorate the performance of the algorithms and increase the required processing time [9]. Deploying feature selection can be used as a solution for alleviating the high-dimensionality problem. However, when both sparsity and high-dimensionality are a matter of concern, the outcomes are far from acceptable. In such cases, clustering data in the whole space with scattered high-dimensional data objects poses a real challenge. The challenge is composed by the curse of dimensionality, the big space with little data objects and applicability of the classical available methods [12].

1.3 Research objectives

The main objective of this research is to improve the clustering performance using the concept of subspace clustering of semi-supervised sparse data based on the knowledge available in the data feature. This objective is divided into the following sub-objectives:

1. To review and investigate the high-dimensional and sparse data clustering literature and narrow-down the possible applicable algorithms.
2. To investigate and select soft subspace clustering algorithms, which can be used for subspace constraint clustering.
3. To implement a MATLAB code for the selected algorithms and test their performances using real dataset.
4. To benchmark the outcomes produced from the selected algorithms with the performance of *k*-means and prove their applicability.

1.4 Research methodology

The research is mainly applied using four different algorithms applicable to subspace clustering process or claimed to be applicable to it. These algorithms are listed below:

- Classical *k*-means.
- Constraint Feature Selection (CFS).
- Infinite Feature Selection (IFS).
- ReliefF Selection.
- Unsupervised Discriminative Feature Selection (UDFS).

These algorithms are used for reducing the subspace dimensionality with some datasets that are hard to cluster. The used datasets are retrieved from the UCI Machine Learning Repository and are properly cited in Chapter 4.

The used development simulation environment is MATLAB 2018b, which is installed on an Asus Notebook, Intel(R) Core (TM) i7, 64-bit CPU @ 3.4 GHz and 16 GB RAM. The developed code is presented in a Graphical User Interface (GUI) format for convenience use and further modification. All developed and quoted codes have been properly appended in Appendix A.

1.5 Thesis organizations

The thesis is composed of five chapters, which are organized as follows:

- **Introduction:** Has presented the introductory part of the research where the general required concepts are briefly presented and explained. In addition, the

formulation of the research idea, motivation, main problems, objectives, and methodology have also been given in this chapter.

- **Background:** This chapter addresses the concept of clustering in a relatively deep sense. It also provides a description of the clustering problem in sparse datasets and possible ways for overcoming this kind of problem. In addition, the chapter categorizes subspace dimensionality reduction and presented possible available solutions.
- **Literature Review:** This chapter review the most relevant works have been investigated in the field of soft subspace clustering. It also addresses different mathematical models presented in these studies along with their optimization way. Further, the chapter addresses combining some feature selection methods for improving subspace reduction ability.
- **Methodology:** This chapter presented the materials and methods used in this research. The datasets used in the research are addressed within this chapter as well as the used simulation software, and the way of development. Moreover, it elaborates the steps of the deployed algorithms and presents their brief mathematical concept.
- **Results and Discussion:** This chapter presented the obtained results from the selected algorithms, and addresses their results in the way it justifies why they are produced and the possible reasons for their failure if any.
- **Conclusion and Future Work:** This chapter provides the concluded remarks from this research. It also addresses possible ways from improvement and directions for future research in the same field.

CHAPTER 2

LITERATURE REVIEW

2.1 Background

The main goal of clustering applications is to divide and group a dataset into subsets (clusters) based on some similar characteristics. These characteristics are usually obtained from the attributes of the dataset itself. In data mining and machine learning methods, clustering is generally regarded as unsupervised, meaning that the attributes of the dataset cannot exactly provide a solid relationship to conduct the clustering process, rather than the inherited structure of the dataset is investigated to achieve the possible grouping [16]. Although, this is usually the case, deploying an observation or a characteristic in the dataset for the clustering process have been adopted in a number of machine learning studies [17].

The concept of clustering is widely applicable to almost any dataset which data samples can be classified into groups. However, in some cases, high-dimensionality of datasets makes it inefficient, due to the accuracy and time, to produce the expected clustering outcomes. Essentially, a dot or a vector in a multidimensional plane can represent data samples in a dataset. The distances between these vectors determine the dimensionality from one side and the applicability of the clustering process form the other side [18]. As stated in Chapter 1, the recent developments in technology and its applications have made datasets more complex in many aspects, especially in number samples and dimensionality, which is directly related to the attributes describing the dataset. In this regard, it is important to mention that one of the reflections or impacts of high-dimensionality of data is commonly recognized as data sparsity. The main issue to be tackled in the case of high-dimensionality or data sparsity is maintaining clustering performance in terms of both processing time and accuracy. In addition, one of the common problems with high-dimensional sparse data is generated due to the meaningless interpretation of distance measures, which is of distance measures

difference due to the nearly equal distances in high-dimensional data.

Traditionally, in order for clustering algorithms to work properly, they need to include all dimensions of the samples of the dataset. This is done to obtain the needed knowledge for the unsupervised learning process. Nevertheless, when considering high-dimensional datasets, it is likely that many of the available dimensions are either repetitive or irrelevant to the analysis. In addition, considering all available dimensions may hinder the outcomes and may smear some possible clusters due to the generated noise [20]. Therefore, researchers have investigated this specific problem to come up with some considerable solutions. One of the deployed solutions, partially successful, is feature selection, which enhances the outcomes of the clustering process, considering data sparsity. The main idea of feature selection is eliminating the redundant and/or irrelevant dimensions. Even though feature selection relatively succeeds in many applications, the main drawback is the hidden or overlapped clusters in the datasets. As an improvement proposal for such a drawback is the localizing the clustering procedure by deploying the subspace clustering mechanism [21, 22].

2.2 Sparse data

In many data acquisition applications, such as sensor-collected, the presence of data samples or objects relies on many factors. Among the relevant factors to this work is the amount and characteristics of the acquired data. Here, it is possible to have a data collection pattern with a lot of zeros as a sensor output. This missing can be caused by justified or unjustified reasons [9]; yet, the consequence of having zeros in data collection is the formation of data sparsity. This scenario is commonly realized by the deployment of various reading sensors or actuators to collect data from the same source based on different circumstances [23]. In this case, the presence of zeros in the data collection is an essential part of the data itself and can not be neglected. Nevertheless, the complication created the plethora of zeros is the composition of rare data points (sparse) with an actual value other than zero [24]. Unlike dense data, which is characterized by plenty of different recording for the same situation, for instance, the traffic movement on a busy road, over a certain period, sparse data is characterized by the existence of rare points with actual readings other than zero [25]. In modern applications, data is collected from many sources with excessively large amounts and high-dimension, as these dimensions are at extreme importance for recognition

purpose. Thus, with a large number of data points which have a number of dimensions, the sparsity problem becomes a real problem for data analysis due to both computation and sparsity complications [26]. This problem is less serious in supervised learning algorithms as the outcomes depend directly on the data points themselves where the principal component plays a major role in such cases [27]. However, in unsupervised learning algorithms, such as the classical clustering *k*-means, the validation of the results can not be directly achieved, and the expectation of erroneous outcomes is quite high [25].

2.2.1 Sparse data problem in machine learning

In machine learning applications, the performance of most algorithms, especially classification and clustering, dictates the presence of actual data with values other than zero [9]. This problem becomes more obvious when unsupervised algorithms are considered, as the accurate prediction tends to fail due to the data pattern [28]. Therefore, some complementing techniques have been proposed to handle the sparsity issue as a preprocessing stage, such as imputation or deletion [9]. In imputation, the mean value of all presented values is calculated and then used for replacing all zeros in the dataset. In this case, the presence of zero is compensated by the mean to overcome the sparsity issue. However, this may work well for datasets with small amount of entries [29]. Otherwise, imputation tends to fail when a large amount of entries is considered, especially when the large deviation is expected to occur based on the available data points' values [9]. On the other hand, the deletion method applies the total elimination of zeros from the dataset, which potentially removes a real recorded observation from the dataset and creates a false impression in the obtained analysis results [9], [26].

The sparsity problem can be also tackled by some algorithms as a part of their internal mechanism. There are native algorithms that can efficiently handle the presence of zeros in a dataset, depending on the type of application, without the need of a preprocessing or external compensation. Some of the common examples for such algorithms are decision tree algorithms. However, usually, these algorithms are not opted for clustering applications [26].

In clustering applications, the sparsity is generally tackled by distance matrices, where the improvement is made leveraging the dissimilarity and based on it decide the best

match for the data point. Some of the proposed algorithms have already incorporated the Jaccard Index and Pearson Similarity Coefficient to cope with the sparsity problem [9]. Jaccard Index is used due to its ability to measure the occurrence of the data points other their absence, which is suitable for sparse data applications [30]. Both standard deviation and covariance are applied in Pearson similarity coefficient to estimate the likelihood of cluster matching. These two statistical factors largely assist in coping with the sparsity data clustering problem [31].

2.3 Subspace and sparse data

The challenge of having sparse data with high-dimensionality is a common obstacle in modern data mining and machine learning applications. Obviously, this kind of challenge is very serious as it deteriorates the performance of unsupervised clustering process, which hinders the ultimate analysis. Thus, this issue dictates some sort of preprocessing or improved clustering mechanism. Among the proposed methods is subspace clustering, which refers to deploying a number of subspaces in sparse data rather than considering the whole space at once [32]. As it can be understood for the name, subspace clustering deploys a different subset of dimensions to cluster objects similar to each other in that specific subset. However, the task is not as straightforward and easy as it may sound; determining subspaces in sparse data and the objects belonging to these subspaces pose a challenge in machine learning applications [33]. Arguably, subspace clustering for high-dimensional data is the most effective way developed so far. Achieving such a clustering requires sub-dividing space into a set of subspaces to reduce the dimensionality of data and then obtain the union of these subspaces for the final clustering process, as shown in Figure 2.1 [34]. Somehow similar to the traditional clustering methods, the membership objects of a union of subspace clustering is found via the similarities with other objects, but considering only subspace instead of the whole space. Unlike in traditional clustering, in subspace clustering, the overlapping of subspace classes is allowed. Nevertheless, determining these subspaces themselves is the main challenge in subspace clustering. For this reason, as it will be discussed in Chapter 3, it is commonly stated that subspace clustering requires search and evaluation methods. Doing so is generally categorized into two main groups: hard subspace and soft subspace clustering. Nonetheless, some researchers categorize all subspace clustering into four broad categories: algebraic, clustering-based, iterative, spectral, and statistical [35].

It is important to mention that subspace clustering is categorized in a hierarchal way based on subspace search methods as well as cluster evaluation methods, as seen in Figure 2.2. It is possible to deploy any applicable subspaces search method that discovers all possible subspace for clustering purpose. However, the main problem encounters this kind of scenario is the feasibility of the outcomes as subsets search by itself is a cumbersome and challenging task. Thus, the level of the complexity of the algorithm is determined based on the sophistication of the used search method, which in turns expands the hierarchal view to a number of levels. Additionally, the features evaluated for clustering must be considered on a subspace level, or in a more precise term locally measured, which also allows the expansion of the hierarchal categorization [34].

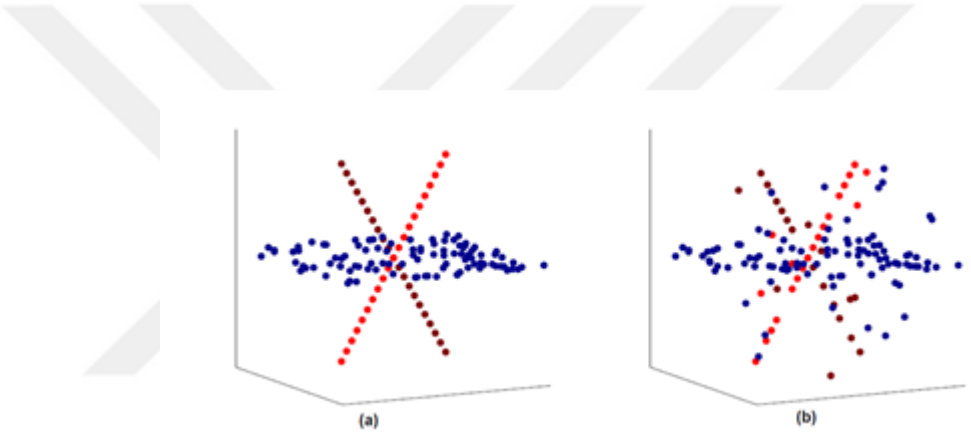

Figure 2.1 (a) Exact clustering, (b) subspace clustering in terms of unions [34]

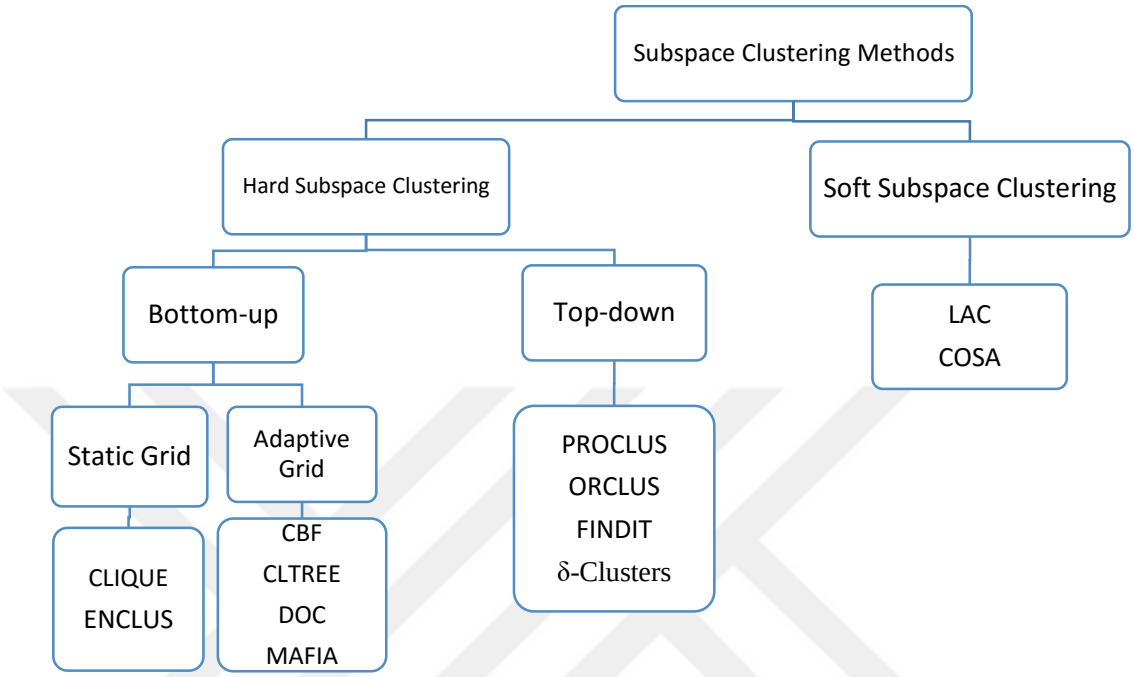

Figure 2.2 Subspace clustering hierarchy

2.3.1 Hard Subspace Clustering

In this method of subspace clustering, it is aimed at discovering the precise subspaces, which cover all possible clusters. Essentially, this category is sub-divided into two broad searching categories, namely bottom-up and top-down [35]. The following subsections are not exhaustively reviewing all methods, as there are a lot, mostly similar, under the same category. However, the presented methods generally cover the main used ideas.

2.3.1.1 Bottom-up Methods

Bottom-up is one of the hard clustering approaches that represent the second hierarchy level of the categorization. Generally, conducting subspace search using bottom-up composes some essential steps, which are represented by:

- **Intervals Generation and Identification**: This is the initial step where the whole space is divided into a number of intervals. These intervals are then used for determining the local density level for each of them.

- **Two-Dimensional Cells Identification:** Once intervals and their densities have been determined, dense cells are determined within these intervals based on only two-dimensional (2D) at first.
- **Three-Dimensional Cells Identification*:*** Once intervals, their densities, and dense cells in 2D have been determined, the interaction between these dense cells and dense intervals are used for identifying dense cells within these intervals but on different dimensional base, starting from three-dimensional ending by *k*-dimensional, where *k* represents the number of dimensions.
- **Merging Adjacent Dense Cells*:*** This is the concluding step where subspaces with the same dimensions are used to merge all adjacent dense cells, which ultimately form the required clusters.

These steps are not exactly straightforward and merging dense cells may lead to the creation of two or more clusters that are supposed to be one cluster. In addition, applying this method usually ends up with overlapped clusters. Thus, acquiring useful knowledge from these clusters is dependent on varies factors such as the choice of the grid size as well as the controlling parameters of the deployed density threshold. The main process in the bottom-up subspace clustering method is generating intervals to ultimately produce the dimensional subspace or *grid*. For achieving this task, two main methods are followed, namely CLIQUE and ENCLUS. These two methods deploy a fixed-size grid to divide each space into intervals [36]. Moreover, it is possible to deploy an adaptive grid determined by data-driven scenarios, which sets the exact intervals within a subspace. These adaptive approaches include Cell-Based Clustering (CBF), CLTREE, DOC, and MAFIA. While both CBF and MAFIA use histograms for analyzing the cells within the subspaces, the strategy followed by CLTree is based on a decision tree for finding the optimal cutpoints, and DOC deploys a random search process guided by the maximum interval and minimum instances number per cluster [37]. The following points describe each approach in bottom-up strategy:

A. CLIQUE: The name CLIQUE is coming after the use of (CLustering In QUEst) initially presented by Agrawal et al. in 1998 [38]. This algorithm is considered one of the oldest subspace clustering approaches. As a static grid bottom-up approach, CLIQUE performs subspace clustering by identifying cells densities and then merging the neighbor cells in local clusters. Initially, the dense subspaces are identified and dense intervals within these subspaces are sorted. The dense areas, apart from the

dataset, are defined as coverage; this coverage is then used for determining which subspaces to keep, the ones with greatest densities, and the rest of subspace are trimmed. CLIQUE identifies dense cells or grid units regarding their adjacency within the kept subspaces, deploying a depth-first search scenario. Another scenario is used, usually, greedy growth, to form clusters by merging these cells together. The mechanism of cluster creation is initiated at the randomly selected dense cell and then greedily growing to form a maximal region. This process is applied to each dimension, and the union of merged cells keeps progressing until it covers the whole cluster. During the course of this action, it is inevitable to generate redundant regions. These regions are repeatedly removed where the redundant regions with the smallest areas are eliminated until no further elimination is possible. To form the final clustering form, CLIQUE uses Disjunctive Normal Form (DNF) to define the hyper-rectangular clusters. Using CLIQUE, the clustering process may generate disjoint or overlapped clusters where more than one cluster can share the same instances. Nevertheless, tuning the input parameters is essentially impractical for a specific dataset. In addition, trimming subspace may lead to the loss of effective clusters [39].

B. ENCLUS: This algorithm is considered a variant of CLIQUE algorithm. The main difference between CLIQUE and ENCLUS is that ENCLUS deploys entropy for speculating the density, where the name came from (ENtropy CLUStering). The general intuition made in the case of ENCLUS is that subspaces with clusters typically have lower entropy than those with no clusters. By measuring the entropy of both coverage and density, ENCLUS identifies possible high coverage and density. The main idea is that the entropy decreases by the increase of both coverage and density. In addition, the correlation is deployed by ENCLUS as a measure of clusterability. The correlation is measured by interest, which is calculated as the difference of entropies for the sum of a set of dimensions and multi-dimensional distribution. In this case, the larger the difference value, the higher the correlation will be. Essentially, the main methodology followed by CLIQUE for mining dense subspace is also used by ENCLUS, which agrees in outlined steps of the bottom-up approach. However, trimming the insignificant intervals is achieved by deploying some of the entropy properties, namely the downward and upward closures to identify the subspaces with minimal correlation. The mechanism followed by ENCLUS here is measuring interest gain, which allows determining subspaces with entropy value above a certain

threshold, □, and interest gain. This procedure resembles the determination of coverage in CLIQUE, and the exact same scenario can be used for the rest of the clustering procedure or any other possible mechanism can be applied. ENCLUS is similar to CLIQUE in the sense that both require setting up the grid size, as it is static, and this size affects the performance of both algorithms. Almost the same running time of both algorithms can be observed for the same datasets. In addition, subspace scalability issues appear with both CLIQUE and ENCLUS, yet the use of entropy helps in reducing the total irrelevant clusters but fail in improving the loss of some meaningful clusters [40].

C. Cell-based Clustering: This method is known in its abbreviated form by CBF. The key development idea of CBF is based on combating scalability issues that deteriorate other bottom-up approaches, such as CLIQUE and ENCLUS. The increment of cells resulting from dimensionality increase is solved in CBF by deploying an algorithm that continuously inspecting a dimension's maximum and minimum values in order to create the optimal partitioning. In addition, scalability is partially addressed by CBF, considering a large number of instances in a dataset. The problem here is tackled by the deployment of an improved data structure, filtering-based index, for storing cells, which allows faster access. Notwithstanding, CBF suffers from its sensitivity to section threshold, which determines the frequency of intervals in the same dimension, and to cell threshold which identifies the data points; minimal density in an interval. Mostly, all other advantages provided by bottom-up approaches are available with CBF, yet there is a slight deterioration in the clustering performance with CBF compared to other methods, however, the processing time is generally faster [41].

D. CLTree: Similar to CBF, CLTree deploys a different methodology for identifying the most suitable subspaces in a dataset. However, unlike CBF, CLTree uses a special decision tree for achieving this task. For dividing each subspace and determining the dense regions within intervals, CLTree deploys an algorithm based on decision trees. Being a bottom-up approach, CLTree follows the same steps mentioned earlier in achieving the clustering process, that is, dividing the subspace into intervals, individually evaluating each interval, identifying intervals with dense areas, and deploying the dense ones for the rest of the clustering process. In CLTree, The boundaries of intervals are used for splitting the deployed decision tree; the decision here is made based on the density, which is measured by some sort of gain criteria. It

is important to mention that CLTree requires two main parameters as inputs, which are minimum cells number in a region that makes it insignificant and can be trimmed, and the minimum density relatively compared between adjacent regions. This procedure is done prior to merging adjacent regions to create a cluster. Discarding insignificant regions from the decision tree is a crucial step and it dramatically affects the produced results if not considered carefully. The way CLTree clusters a dataset follows finding clusters in the form hyper-rectangular areas within subspaces. The used decision tree may become complex in terms of time where it may reach to a quadratic function. However, scaling the algorithm linearly in terms of both dataset instances number and the obtained intervals is possible [42].

E. Density-based Optimal Clustering (DOC): DOC projective clustering is considered a hybrid approach, which combines both bottom-up and top-down approaches. Essentially, DOC considers a projective cluster, and it defines an optimal projective cluster in a mathematical way for this purpose. The projective cluster in DOC is established by determining a strong clustering tendency of a subset of instances C towards a subset of intervals D. In DOC, this is formed as a pair (C, D), and in order to create the aimed clustering process, DOC must identify the optimal pairs (C, D). Here, identifying the optimal pairs is done by generating a small subset S, by any valid means, for example, random sampling. Instances in both C and S are compared and the absolute result of their difference should be less than or equal of the fixed-length, ω, which is a fixed subspace length provided by the user. This procedure is repetitively done, and both C and S are continuously obtained, using random sampling or any alternative method, until the optimal result is achieved. The minimum number of instances which can create a class, α, is also needed for the operational process of DOC. The minimum allowed density for a cluster is then determined by both parameters α and ω. It is also suggested that a trade-off should be made between the number of intervals and subspaces. For this specific purpose ano,ther parameter is used, which is usually referred to by β. This balancing process is at extreme importance for DOC, as the time complexity grows exponentially with respect to the intervals number and linearly with the cells number [34].

F. MAFIA: Like ENCLUS, MAFIA is another variant of CLIQUE. However, unlike CLIQUE, MAFIA improves the performance of the outcomes by deploying an adaptive grid size. This grid size is obtained from the data distribution of a given

dataset. The general steps used by bottom-up methods are also used by MAFIA where initially the whole space is divided into a set of subspaces. The minimum required number of intervals in MAFIA is determined by deploying a histogram. The same idea used in CLIQUE of merging adjacent dense cells into larger regions, which form a cluster, later on, is also used in MAFIA. Nevertheless, space partitioning is done based on data distribution in this case, which forms the adaptive nature of the algorithm and allows more accurate results to be produced. MAFIA requires thresholds for both area density and adjacency merging. These thresholds will allow merging adjacent regions within a specified threshold value; these merged regions form the clusters later on. It is important to mention that MAFIA is sensitive to these threshold values. In the usual operational process, MAFIA deploys a default grid as seed, where the adaptive grid is then formed based on it. Like CLIQUE and ENCLUS, MAFIA can operate based on different input scenarios, such as large datasets, different sizes, and shapes of subspaces, and it can also produce overlapping clusters as well. However, unlike

CLIQUE and ENCLUS, MAFIA outperforms these two algorithms in terms of time complexity, yet, it is crucial to mention that the time complexity of MAFIA also exponentially grows when the number of intervals increases [44].

2.3.1.2 Top-Down Subspace Methods

Top-down methods for subspace clustering are considered iterative methods, meaning that they initiate their procedure by outlining an approximate view of the clusters that cover the whole space. These clusters initially start with equally weighted subspaces. Later, every subspace will be given a certain weight in a relationship with each cluster. The clustering process keeps updating the assigned weights by iteration, as it is an iterative approach. One of the disadvantages of top-down approaches is the expensive time complexity of the iterations, which has been improved by integrating some sampling techniques into the methods. Unlike bottom-up approaches, top-down approaches create clusters that accept dataset samples in only one class. Both subspaces size, as well as the total number of clusters, is considered as critical parameters for all top-down approaches. It is possible to deploy the sample size as a parameter, which enhances the quality of the output, in top-down approaches, if the used technique is based on sampling [45]. Similar to bottom-up approaches, top-down has a number of commonly used algorithms, which are FINDIT, ORCLUS, PROCLUS, and δ-Clusters.

A. FINDIT: This algorithm refers to Fast and Intelligent Subspace Clustering Algorithm using Dimension Voting [46]. FINDIT is a top-down approach that deploys the essence of voting in its operation. The concept here relies on a certain distance measure referred to by Feature-Oriented Distance (FOD). FINDIT identifies instances that agree on features with a specified threshold value, ϵ. In high-dimensional data, instances may get a close match with several features but not all of them. The initial step in FINDIT is achieved by choosing two small sets, randomly created, to be clusters' medoids; this stage is called the sampling stage. In the next stage, FINDIT uses the generated medoids for calculating FOD to determine the correlation, which allows the creation of clusters. The threshold is then iteratively increased until the final clustering shape comes to a stable state. This stage is called clthe uster-forming stage. Finally, FINDIT allocates the dataset instances to the formed mediods considering all identified subspaces. Generally, two main parameters are used by FINDIT, namely the minimum permissible instances number per cluster, and the minimum permissible distance between any two clusters. One of the main disadvantages of FINDIT is the iterative nature of finding the optimal, which increases the time complexity of the algorithm. It is possible in many applications which use FINDIT to find a proper sampling technique integrated with the algorithm to improve its usability and running time quality [47].

B. ORCLUS: This algorithm uses the correlations contained or generated as an inter-attribute in a given dataset. Initially, ORCLUS outlines clusters arbitrary then it assigns cells to the closest cluster in an iterative way; this stage is referred to as clustering assignment. The next stage followed by ORCLUS is achieved by identifying subspaces associated with the generated clusters. This is mathematically obtained by choosing the eigenvectors with the smallest eigenvalues after calculating the covariance matrix for each cluster. This produces some clusters that are close to each other and can be merged together. The mathematical computation of ORCLUS creates a high complexity, which makes the algorithm deploy some sort of sampling scenario to enhance the performance. Nevertheless, possible missing small clusters are inevitable in this case [48].

C. PROCLUS: PROjected CLUStering (PROCLUS) algorithm was the oldest top-down approach. PROCLUS uses similar principles of those deployed by ORCLUS.

The main idea of dataset sampling, setting up a number of mediods, and then iteratively improving the clustering process is also applied in PROCLUS. A suitable greedy algorithm is used for choosing a number of possible mediods. These mediods should be relatively not close to each other. During this stage, PROCLUS makes sure that, at least, one class is representing each cluster. Generally, this phase is referred to by clustering initialization. One the initialization step has been concluded, PROCLUS deploys a number of randomly selected mediods, k, and iterate over the reduced dataset to evaluate whether the clustering process is improved or not. a specific parameter is used to maintain the quality of the clustering process, namely the average subspace dimensionality, l. Each medoid has to associate k l features. Identifying the targeted subspaces for each medoid is followed by assigning data points to these medoids; this is done using Manhattan segmental distance. The process is concluded by eliminating any medoid with a minimal number of data points. This stage is referred to by the iteration phase. The final stage in PROCLUS is achieved by refining the created clusters where the assignment of data points may be shuffled to come up with better quality clusters. This stage is called the refinement phase. PROCLUS requires working with equally divided subspaces and with the present average number of features. Overlapping in PROCLUS is a case to consider, as each set of data points are ultimately associated with one cluster. The running time of PROCLUS is usually little faster than CLIQUE and its variant ENCLUS considering large datasets. Similar to iterative approaches, PROCLUS is sensitive to the input parameters, which affect the quality of the clustering process. In addition, medoids elimination may completely discard some useful clusters in the process [49].

D. δ-Clusters: This algorithm deploys the coherence between instances subset and attributes subset. The idea is based on calculating this coherence and using it as a distance measure for clustering purpose. The relationship the can be observed with coherent data samples is used for generating a proper cluster, as it is possible to discover instance based on their coherence. It is possible to use any valid coherence measure; however, some researchers have used PearsonR for such purpose [50]. Using δ-clusters, cluster seeds are initially created, then the algorithm attempts iteratively to improve these clusters using a random shuffling technique, which tries different data samples and attributes to get a better clustering quality. This process stops when the iterative process produces no further improvements. Two main parameters are needed

for the operational process of δ-clusters algorithms, namely the size of the cluster and the total number of considered clusters. It is important to mention that the time complexity of δ-clusters depends on these two parameters. In addition, while implementing the δ-clusters method, it is recommended to maintain pre-knowledge or some sort of speculation of cluster size, as it makes difference with large datasets. Unlike other top-down approaches, δ-clusters deploys the coherence as a measure, which makes it a potential method for a number of applications, especially in the field of medicine [51].

Despite the diversity of clustering approaches, stumbling with unsatisfied clustering outcomes is a common problem, especially when very dense or sparse datasets are considered. Nowadays, with many different sophisticated systems being in use, the complexity of mined datasets has become generally very high; thus, avoiding the use of dimensional data makes current systems very impractical. Meaning that any enhancement solution should consider the use of high dimensional datasets, as they are the commonly produced output in many modern systems. In this regard, the advancement in computerized processes has enabled the deployment of soft methods that relatively have a lower implementation cost as well a more efficient clustering outcome. Mainly, for this reason, researchers have put forward different soft clustering enhancement methods, especially in the past two decades. Soft subspace clustering is one of the recent trends deployed as a solution for the persistent high dimensionality problem. The method itself may use different algorithms aiming at reducing the dimensionality and achieving the desired outcomes.

2.4 Related Works

Dimensionally high datasets are treated in special ways when clustering is a matter of interest. When traditional clustering methods, such as *k*-means, do not produce the expected clustering outcomes due to data sparsity, a number of different methods can be applied to improve the performance in this case. As stated in the previous chapter, BACKGROUND chapter, it is possible to reduce the dimensionality using different methods, where the choice is applicable based on the type of datasets as well as the aimed performance, in terms of quality and time complexity. Nevertheless, reviewing each and every dimensionality reduction method would require hundreds if not thousands of pages. Thus, the reviewed literature in this thesis is restricted to soft

subspace clustering, as it is used in the experimental for improving k-means outcomes.

Prior to the year 2000, almost all subspace clustering approaches appeared in the literature were represented by hard subspace clustering methods [18][19]. This is true except for very few attempts, one of which is the work of DeSarbo et al. in 1984 [54]. In their work, the author came up with the aid of what is called SYNthesized CLUStering (SYNCLUS). In SYNCLUS, the classical *k*-means was modified by integrating dimensional weight. The algorithm initializes its process by obtaining *k* clusters, using the classical *k*-means, using a set of starting weights. Next, the algorithm deploys an optimization function, given in Equation (2.1), to produce the updated weights (optimal values).

$$C_i^2 = \frac{\left[\sum_{j=1}^{J}\sum_{j'=1}^{J}\delta_{jj'}d_{jj'}^{2(i)}\right]^2}{\sum_{j=1}^{J}\sum_{j'=1}^{J}\delta_{jj'}^2\sum_{j=1}^{J}\sum_{j'=1}^{J}d_{jj'}^{4(i)}} \tag{2.1}$$

where,

C_i^2 is the sum of squares, or the mean-squared error in a basic sense,

$d_{jj'}^{2(i)}$ is the weighted squared dissimilarity for the objects *j* and *j'* in the *i*th dimension.

$\delta_{jj'}$ Is a positive quantity used as a controlling parameter calculated for the objects *j* and *j'* in the *i*th dimension.

SYNCLUS keeps iterating until C_i^2 is minimized as much as possible, which means the algorithm has reached its optimal performance, and no further change in the clustering process is possible for a given dataset. SYNCLUS was tested and compared to both hierarchal clustering analysis and *k*-means by the developers. Generally, for the deployed dataset, SYNCLUS outperformed the two other algorithms [54]. Nevertheless, the time complexity of SYNCLUS is relatively high; thus, it cannot be used for complex datasets with a large number of samples.

De Soete, using hierarchal clustering, had also proposed soft subspace clustering. The main idea presented by the author is the deployment of a hierarchal structure for dimensional weight assignment [55]. The main issue with such studies is usually the time complexity of the algorithm, which tends to grow exponentially for large datasets.

The idea of De Soete was later extended by Makarenov and Legendre in [56] to produce an enhanced k-means application suitable for sparse datasets. The authors deployed a weight vector, which was assigned to its corresponding dissimilarities of two data points. Using these weights, an iterative procedure was applied to optimize the cost function and reach the optimal clustering performance. This can be represented in Equation 2.2 [56].

$$H(W) = \sum_{k=1}^{K}\left(\sum_{i,j=1}^{n_k} d_{ij}^2 / n_k\right) \quad (2.2)$$

where,

dij is the distance between two data points represented by the (ith, jth) pair,

K is the total number of clusters nk is the number of data points in the kth cluster.The authors had also used the numerical data presented in [55] for their analysis. The authors could show good results compared to the ones obtained in [55]; however, the algorithm deployed very complicated optimization procedure that made its time complexity very high and processing time very slows. This implies that the algorithm in its developed state cannot be used for large datasets. It is important to mention that the work of Makarenov and Legendre was one of the earliest after the year 2000. This work was followed by the work of Modha and Spangler [57], which also deployed a method for automated weighting for the classical *k*-means clustering. Initially, classical *k*-means is applied to determine the weights for each feature, then weight assignment optimization is achieved using Fisher's ratio, *Q*. This ratio has mainly to do with the average distortion of within-cluster and between-cluster. The main problem with this application is deciding the initial set of weights, or the weight vector, which will be improved later. Since the initial step is randomly generated and there is no guarantee that an optimal set can be chosen, the iterative process can extend in its running time and make the processing time considerably slow.

The next following years have witnessed a new trend in terms of dimensionality reduction, where soft subspace methods have emerged as a competitor to the classical techniques hard subspace reduction technique. The work of Domeniconi et al. presented an improvement for the classical *k*-means by incorporating a locally adaptive clustering mechanism or LAC in short. In their method, the authors presented a way

for assigning dimensionality weight to produce subspace reduction. The idea is based on the fact that the formed clusters should be relevant to data points as well as some metrics for weighted distance. When there is a dataset, *G*, has a space with *N* dimensionality, a *weighted cluster*, *C*, can be defined as a data points, *D*, subset, which has weights in a vector format given as $\mathbf{w} = (w_1, w_2, \ldots, w_N)$. Now, associating each cluster to its own weighted vector provides a good idea on how data points in this subset are correlated to the cluster. This is measured by relative average distance and can be iteratively updated. The partitioning of data points, Y, with respect to the weighted clusters produces a set of *k* clusters where their centers are represented by c_j, where $j = 1, \ldots, k$. based on this, the partitioning of *G* to set follows Equation 2.3 [52].

$$G_j = \left\{ Y \middle| \sqrt{\left(\sum_{i=1}^{N} w_{ij}\left(y_i - c_{ji}\right)^2\right)} < \sqrt{\left(\sum_{i=1}^{N} w_{li}\left(y_i - c_{li}\right)^2\right)}, \forall\, l \neq j \right\} \tag{2.3}$$

The most important in for LAC to work is to minimize the error generated within this partitioning process, which is given in Equation 2.4 [52].

$$E = \sum_{j=1}^{k} \sum_{j=1}^{k} w_{ji} \times exp^{-Y_{ji}} \tag{2.4}$$

According to Equation (3.4), both centers and weights will be optimum if the error, *E*, is minimized. The authors designed their developed LAC on these bases and empirically obtained influence parameter, *h*, to minimize the error effect. Thus, they developed the dimensional weight applied in this work as given in Equation 2.5 [52].

$$w_{ji} = \frac{e^{hY_{ji}}}{\sum_{l=1}^{N} \sqrt{e^{-2hY_{jl}}}} \tag{2.5}$$

Using these equations, the authors deployed four different datasets, three of which are simulated, and one real dataset, to test the developed LAC algorithm. The results obtained from these datasets were compared by the performance of PROCLUS, classical *k*-means, and DOC. Generally, the performance of LAC outweighed all considered algorithms in terms of error rate, except for the real dataset case, where PROCLUS recorded a slightly better performance.

This work and relevant literature had stimulated other researchers to investigate clustering in the same direction. Within a short time next to the development of LAC, Friedman, and Meulman, in 2004, proposed and developed COSA, which is based on

improving the classical *k*-means clustering to suite sparse or high-dimensional datasets [53]. COSA relies on the local assignment of weights, which represents a soft subspace reduction methodology. To elaborate on this process, Friedman and Meulman published an extensive work elaborating the operational mechanism of COSA. In principle, there is an objective function, resembles the objective function of LAC but with a different scenario, that should be optimized to produce the desired clustering quality. This formula is given in Equation 2.6.

$$J_{COSA} = \sum_{i=1}^{N} \left\{ \frac{1}{\sqrt{N}} \sum_{j \in knn(i)} D_{ij}[w_j] + \lambda \sum_{k=1}^{n} w_{ki} \log(w_{ki}) \right\} \quad (2.6)$$

where

D_{ij} is the measure of dissimilarity, or distance, for a pair (i, j) that represents a data point,

$knn(i)$ is the set of nearest objects to i obtained from D_{ij},

w is the assigned dimensional weight, which changes based on ith feature and lth cluster,

λ is the incentive strength controlling parameter, which is a positive quantity.

According to Friedman and Meulman, minimizing Equation (2.6) requires a number of steps. Initially, a weighted distance matrix is built to represent data points. Next, the nearest neighbors are found based on the obtained distance matrix to achieve a hierarchal clustering process. This process then takes place on an iterative basis until no further change can be achieved in the clustering. The authors generated their own simulated datasets and tested COSA algorithm using these datasets along with some real datasets. Essentially, the performance produced by COSA was superior back then; however, the main problem with the algorithm is time complexity.

Huang et al. presented an automated *k*-means clustering algorithm for dimensionality reduction. Their work used relatively the same principles of initially deploying the classical *k*-means then iteratively updating the clustering process based on an optimization formula. Essentially, the proposed improvement here is called *W-k*-means referring to weighted *k*-means, and it was achieved based on minimizing the objective function, which is given in Equation 2.7 [58]:

$$J_{WKM} = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{j=1}^{m} p_{i,l} w_j^{\beta} \, d(x_{i,j}, z_{l,j}) \quad (2.7)$$

where,

d is the dissimilarity measure calculated between data point *i* and the medoid of the cluster *l* for the *j*th variable,

$p_{i,l}$ is a positive integer variable represents the location of data point *i* with respect to cluster *l*,

w_j^{β} is the jth feature weight and β is the weight controlling parameter.

Equation (2.7) is used for improving the classical *k*-means clustering process by iteratively the given objective function. The authors applied their *W*-*k*-means algorithm using synthesized datasets and obtained improved outcomes compared to the classical *k*-means using the same datasets.

Most of the previously presented attempts to stumble with time complexity issues due to the multiple loops and excessive iterations. Jing et al. proposed an entropy-based weighting *k*-means, which is considered computationally less complex than previous soft subspace reduction in clustering. The authors extended the objective function of *k*-means, such as the one given in Equation (2.7), to include the entropy of dimensional weight, which provides an idea about the density of the subspace. This objective function is presented in Equation 2.8 [19].

$$J_{EWKM} = \sum_{l}^{k} \left[\sum_{j=1}^{n} \sum_{i=1}^{m} p_{lj}\, w_{lj,} d\left(z_{l,j} - x_{i,j}\right)^2 + \gamma \sum_{i=1}^{m} w_{lj} \log w_{lj} \right] \quad (2.8)$$

where,

d, *p*, *x*, and *w* previously defined,

γ is a controlling parameter that manages the incentive strength for the clustering process with respect to the ithe increment of dimensions.

The second part of Equation (2.8) represents the effect of the included entropy on the objective function. The objective function provided in Equation (2.8) has to be minimized in order for the clustering process to be optimized. Generally, the running time of this algorithm is approximately *O*(*hmnk*) where *h* is the number of needed clustering iterations. This means the running time of the algorithm grows linearly with respect to the number iterations, data points, dimensions, and clusters. The performance of the algorithm outweighed number of other algorithms such as COSA,

LAC, PROCLUS, SCAD1 and the classical *k*-means using the same dataset [19].

Gan and Wu presented a subspace clustering based on Zangwill's convergence theorem and they called it Fuzzy Subspace Clustering (FSC). FSC is a recursive algorithm, which randomly initiates a set of clusters with centers, *k*, and then uses these *k* points to initialize the fuzzy subspace, *Z*. The weighted values, *W*, are then found using *Z* and given data points using dissimilarity measure. Finally, the partition *U* can be found from the determined values of *Z* and *W*. FSC deploys an objective function in the form given in Equation 2.9 [59].

$$J_{FSC} = \varepsilon \sum_{j=1}^{k} \sum_{h=1}^{d} w_{jh}^{\alpha} + \sum_{j=1}^{k} \sum_{i=1}^{n} u_{ji} \sum_{h=1}^{d} w_{jh}^{\alpha} \left(x_{ih} - z_{jh}\right)^2 \tag{2.9}$$

where,

$\mathcal{E}$ is a small positive number,

δ is a positive integer number assigned as a fuzzy index,

FSC updates the weights based on Equation 2.10.

$$w_{lj} = \frac{1}{\sum_{t=1}^{d} \left[\frac{\sum_{i=1}^{n} u_{il}(x_{ij} - z_{lj})^2 + \varepsilon}{\sum_{i=1}^{n} u_{il}(x_{it} - z_{lt})^2 + \varepsilon}\right]^{1/(\delta - 1)}} \tag{2.10}$$

The general time complexity of the objective function here has a linear approximation with respect to the number of iterations and set clusters. This makes the approach computationally expensive if the dataset is very large.

Deng et al. proposed Enhanced Soft Subspace Clustering (ESSC) based on incorporating between-cluster separation and between-class information into the objective functions of existing methods, such as *k*-means or any modified soft subspace clustering objective function. The authors proposed incorporating the said details into entropy-based methods, such as the one reported in [10]. Thus, the objective function for their own work shall be a modified version that includes two extra parts, as given in Equation 2.11 [60].

$$J_{ESSC} = \sum_{i=1}^{C} \sum_{j=1}^{N} u_{ij}^{m} \sum_{k=1}^{D} w_{ik} \left(x_{jk} - v_{ik}\right)^2 + \gamma \sum_{i}^{C} \sum_{k=1}^{D} w_{ik} \, lnw_{ik} -$$

$$\eta \sum_{i=1}^{C} \left(\sum_{i=1}^{C} u_{ij}^{m}\right) \sum_{k=1}^{D} w_{ik} \left(v_{ik} - v_{0k}\right)^2 \tag{2.11}$$

where,

C is the number of clusters,

D is the number of features,

N is the number of data samples,

U is the fuzzy partition matrix, which is represented by $[u_{ij}]_{C\times N}$,

V is the centroids matrix, which is represented by $[v_1, \dots, v_C]$,

W is the weights matrix, which is represented by $[w_1, \dots, w_C]$,

γ is the entropy influence controlling parameter, $\gamma > 0$,

η is the weighting between-cluster separation controlling parameter, $\eta > 0$.

The weights in this method are updated according to Equation 2.12.

$$w_{ik} = \frac{exp\left(-\frac{\sigma_{ik}}{\gamma}\right)}{\sum_{k'=1}^{D} exp\left(-\frac{\sigma_{ik'}}{\gamma}\right)} \tag{2.12}$$

where,

$$\delta_{ik} = \sum_{j=1}^{N} u_{ij}^{m} \left(x_{jk} - v_{ik}\right)^2 - \eta \sum_{j=1}^{N} u_{ij}^{m} (v_{ik} - v_{ok})^2 \tag{2.13}$$

The enhanced version of entropy-based weighted subspace clustering initially sets the values of control parameters, then it randomly initiates centroids matrix and then begins calculating the weighting dimensional. For next iterations, the weights will be updated by the formula given in 2.12. There must be a certain iteration threshold for terminating the process. The authors provided an extensive set of parameters values tested for classification purpose. They also deployed synthetic dataset and compared the performance of their algorithm with different soft and hard subspace clustering algorithms. In addition, different datasets extracted from UCI were deployed for benchmarking purpose. In most cases, their algorithm outweighed the performance of other relevant algorithms [60].

Boongoen et al. proposed a filtering approach for soft subspace clustering for sparse datasets. Their idea is based on reducing the computation complexity and lowering the running time of the clustering operation. In addition, this filtering technique can be extended to a number of clustering techniques, other than the classical *k*-means. This algorithm deploys data reliability measure to initiate the clustering process. This is

done by generating a dimensionality-degree matrix. The presented reliability-based *k*-means (R-KM) uses the same classical *k*-means objective function, but it integrates with it the association matrix to update the weights. Initially, a set of medoids, *Z*, is randomly chosen, then the weights, *W*, are estimated based on Equation 2.14 [61].

$$w_{lj} = \frac{AS_{ij}^{\alpha}}{\sum_{t=1}^{d} AS_{it}^{\alpha}} \tag{2.14}$$

where,

α is number of the nearest neighbors,

AS_{ij}^{α} is the association or similarity matrix of data point dimension,

For the next iterations, *W* calculates the set of clusters, *C*, as provided in Equation 2.15.

$$w_{lj} = \frac{MA_{ij}^{\alpha}}{\sum_{t=1}^{d} MA_{it}^{\alpha}} \tag{2.15}$$

where,

MA_{ij}^{α} is the measure of the associativity of the *j*th dimension that has a minimum sharing characteristic for all C_l members.

This algorithm showed varying results with respect to the performance. Some of the algorithms outweighed its performance in some dataset, while it performed better compared to a number of other algorithms. The authors suggested that the algorithm can be dramatically improved by tuning some of the parameters as well as incorporating some sampling technique [61].

De Amorim and Mirkin [62] presented another improvement for the modification made for *k*-means in the work of Huang et al. [58]. The authors suggested a method for combating the effect of noise on the performance of the clustering procedure. For this specific reason, Minkowski metric was incorporated in dissimilarity calculation to improve the updating step of dimensional weights. The objective function suggested by De Amorim and Mirkin includes Minkowski metric instead of the effect of weights only. Thus, Equation (2.7) was modified in their work to suit the applied metric; the produced objective function is given

in Equation 2.16.

$$J_{MKM} = \sum_{l=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{M} p_{i,l} w_j^{\beta} \left| x_{i,j} - z_{l,j} \right|^{\beta} \quad (2.16)$$

The weights are updated also by incorporating Minkowiski metric formula to account for the new weights, as provided in Equation 2.17.

$$w_j = \frac{1}{\sum_{j=1}^{M} \left[D_{v\beta} / D_{u\beta} \right]^{1/(\beta-1)}} \quad (2.17)$$

where,

$$D_{v\beta} = \sum_{l=1}^{K} \sum_{i=1}^{M} p_{i,l} \left| x_{i,j} - z_{l,j} \right|^{\beta}.$$

The author had extensively tested the version of their algorithm and compared it with a number of other algorithms. In all presented cases in their work, the algorithm provided superior performance.

Chitsaz and Jahromi had also proposed an idea for alleviating the effect of clustering noise [63]. The authors proposed the use of fuzzy-based subspace clustering process that incorporates a noise detection stage. Essentially, their study addressed clustering outliers' problem, which is a serious issue in sparse or high-dimensional data. In addition, they also considered the quality of clustering interpretation within their development. Accordingly, the authors modified the objective function of the classical *k*-means as provided in Equation 2.18.

$$J_{FSSC-ND} = \sum_{i=1}^{C} \sum_{j=1}^{N} u_{ij}^{m} \sum_{k=1}^{D} w_{ik} \left(x_{jk} - v_{ik} \right)^2 + \sum_{j=1}^{N} \delta^2 \left(1 - \sum_{i=1}^{C} u_{ij} \right)^m + \gamma \sum_{i=1}^{C} \sum_{k=1}^{D} w_{ik} \ln w_{ik} \quad (2.18)$$

In this method, the weights are updated using the same scenario of Equation (2.12). However, one more step is used for minimizing this objective function, which is the updated centroids, as given in Equation 2.19.

$$v_{ik} = \frac{\sum_{j=1}^{N} u_{ij}^{m} x_{jk}}{\sum_{j=1}^{N} u_{ij}^{m}} \quad (2.19)$$

The authors deployed different datasets, synthetic and real to benchmark their algorithm. The algorithm, in most case, showed a better performance compared to some soft subspace clustering algorithms such as entropy-based, LAC, FSC, FCM, and NC.

A very useful survey on soft subspace clustering algorithms is provided in the work of Deng et al. [64]. The authors addressed a lot of classification aspects in their study. They also categorized the methods based on their orientation, performance, flexibility, etc. Most of the works within the past decade or so lay within enhancing available method an attempting to overcome some of the drawbacks of the developed ideas. However, some research attempts incorporated new research methodologies such as particle swarm, which replaced the Minkowski metric, as given in Equation (2.16), by the nature of particle swarms' behavior [65]. More recently, the concept of semi-supervised subspace clustering algorithms has emerged as a promising tool for improving clustering behavior. Such methods deploy some selection methodology in order to benefit from data attributes in reducing the density of sparse data. This process creates a sort of semi-supervised learning scheme, which can be incorporated with available algorithms to enhance the overall quality of the process [66].

In the most recent studies, authors have generally proposed incorporating a different/number of stages to improve the outcomes of the soft subspace clustering of sparse data; the works of Xu et al. [67] and Yang et al. [68] represent a good example for this methodology. Xu et al. proposed integrating Enhanced Soft Subspace Clustering (ESSC) and Sparse Learning (SL) to tackle any interpretability issues caused by lengthy rules required for sparse datasets. The proposed deploys ESSC to establish the antecedents and sparse subspaces needed for fuzzy rules. To minimize the number of fuzzy rules parameters, the authors used SL as an optimization maneuver. The authors used 10 different medical datasets to test the applicability of the proposed method. It was found that in most cases their method performed slightly better than other ESSC methods, which implies the possibility of further investigation and usability [67]. In a similar way, Yang et al. applied a three-stage framework for solving the dimensionality problem within certain subspaces. In their work, a specific matrix, affinity matrix, was initially extracted from datasets using some representation techniques, composing the first stage. In the second stage, the graph-based transformation and optimization process was applied to the affinity matrix to reduce erroneous clustering results. Finally, spectral clustering was performed on the refined affinity matrix after the second stage to establish a set of subspaces. The authors had tested their method on both synthetic and real datasets, mostly image-based samples and prove the enhancement in a general sense [68].

Bang et al. proposed a clustering algorithm called PUMA that deploys multi-attribute weights for solving both high-dimensionality and inaccurate attributes' weighting problems. The authors proposed that the attribute subspaces are created initially by measuring the co-occurrence probability of each attribute within all dimensions. Once these subspaces are constructed, the algorithm composes the sub-clustering process with respect to the constructed subspaces based on a parallel process considering each computing node. Finally, the composed sub-clusters have to be iteratively merged using a hierarchical clustering method, which measures various scale clusters to do this process. In this work, both synthetic and real datasets were used for testing the performance of the algorithm. The authors used a 24-node Hadoop cluster, which they implemented for algorithm testing purpose. According to the produced outcomes, PUMA performed much better than relevant algorithms for the same deployed dataset [69].

2.5 Used Algorithms

In this study, five different algorithms have been used for clustering purpose. The first one is classical *k*-means clustering method, which is not listed along with other algorithms, as it is commonly used. The rest of the algorithms are used for providing the semi-supervised nature for the clustering process.

2.5.1 Classical *k*-Means

k-means is a very popular clustering algorithm used in machine learning applications. The algorithm is based on vector quantization to divide the data objects *d* into a certain number of clusters *C*. To categorize *d* in their relevant class *c*, *k*-means the cluster *c* with the nearest mean to *d*. In this research, the base-case where both subspace and dimensionality reduction are set to null is realized by the classical *k*-means. Comparing the outcomes of other algorithms, which deploy both subspace and dimensionality reductions will provide means for the final judgment. The classical *k*-means applied in this research follows the algorithm given in Table 2.1 [81].

Table 2.1 Classical *k*-means algorithm

Input:	k a number set for clusters D a lifting ratio set
Output	K clusters
1 **Begin**	
2	$Randomly\ select\ k\ objects\ from\ D\ to\ compose\ the\ inital\ cluster\ centroids$
3	do
4	$create\ K\ clusters\ by\ assigning\ each\ object\ to\ the\ closest\ centroid$ using the mean value of the objects in each cluster
5	$update\ the\ cluster\ means$
6	until no change
7 **End**	

2.5.2 Infinite Feature Selection

Incorporating unsupervised soft subspace clustering procedure by feature selection algorithm can also be done using Infinite Feature Selection (IFS) algorithm. This algorithm was proposed by Roffo et al. [77]. Essentially, the algorithm seems suitable for this research's application as it focuses on the main dense features within the dataset. The algorithm is provided in Table 2.2.

Table 2.2 IFS algorithm

Input:	$F = \{f^{(1)}, \dots, f^{(N)}\}$ α
Output	$\check{s}$ energy scores, for each feature
1 **Begin**	
2	for $i = 1$ to N do begin
3	for j = 1 to N do begin
4	$\sigma_{ij} = max(std(f^{(i)}), std(f^{(j)}))$
5	$c_{ij} = 1 - \lvert Spearman(f^{(i)}, f^{(j)}) \rvert$
6	$A(i,j) = \alpha\sigma_{ij} + (1-\alpha)c_{ij}$
7	End
8	end
9	$r = {}^{0.9}/_{\rho(A)}$
10	$\check{S} = (\mathbf{I} - rA)^{-1} - \mathbf{I}$
11	$\check{s} = \check{S}\mathbf{e}$
12	return $\check{s}$

22 End

2.5.3 Constraint Feature Selection

Constraint Feature Selection (CFS) algorithm is a training-based feature selection algorithm that can be used for semi-supervised sparse data reduction, especially combined with some soft subspace reduction technique [78]. The algorithm is given in Table 2.3.

Table 2.3 CFS algorithm

Input: $S(F_1, \dots, F_N, C)$
δ
Output S_{best}

1 **begin**
2 for $i = 1$ to N do begin
3 calculate $SU_{i,c}$ for F_i
4 if $(SU_{i,c} \geq \delta)$
5 $append\ F_i\ to\ S'_{list}$
6 end
7 order S'_{list} in descending $SU_{i,c}$ value
8 $F_q = getFirstElement(S'_{list})$
9 do begin
10 $F_q = getNextElement(S'_{list}, F_p)$
11 if$(F_p <>$ NULL$)$
12 Do begin
13 $F'_q = F_q$
14 if $(SU_{p.q} \geq SU_{q.c})$
15 remove F_q from S'_{list}
16 $F_q = getNextElement(S'_{list}, F'_q)$
17 else $F_q = \text{getNextElement}(S'_{list}, F_p)$
18 end until $(F_q = \text{NULL})$
19 $F_q = getNextElement(S'_{list}, F_p)$
20 end until $(F_q = \text{NULL})$
21 $S_{best} = S'_{list}$
22 end

2.5.4 ReliefF Selection Algorithm

ReliefF algorithm was developed by Kononenko [79] as a robust algorithm for overcoming a number of issues such as noisy data presence. In this research, the

speculation has been made that ReliefF may assist in improving the quality of clustering outliers and noise effect reduction. The algorithm is not very complicated and presents updates itself based on dimensional density, especially in sparse datasets. The generic concept of ReliefF is presented as an algorithm in Table 2.4.

Table 2.4 ReliefF algorithm

Input: for all instances x_i initiate A, C

Output W quality attributes vector

1 **Begin**
2 $W[A] = 0.0$
3 for $i = 1$ to N do begin
4 randomly pick an instance x_i
5 find nearest neighbors of y_i, k
6 for j = 1 to M do begin
7 if $C_j <> class(x_i)$
8 find k nearest $M_j(C_j)$
9 end
10 for $i = 1$ to N
11

$$W[A] = W[A] - \frac{\sum_{j=1}^{K} \text{diff}\,(A, x_i, y_i)}{NK} + \sum_{C \neq class(x_i)} \left[\frac{P(C)}{1 - P(class(x_i))}\right] \frac{\sum_{j=1}^{K} \text{diff}\left(A, x_i, M_j(C)\right)}{NK}$$

22 End

2.5.5 Unsupervised Discriminative Feature Selection

The fifth and last deployed algorithm in this study is unsupervised Discriminative Feature Selection (UDFS) which is proposed by Yang et al. [80]. This algorithm can be used for semi-supervised clustering process. The algorithm is claimed to be powerful in its performance, especially in high-dimensional data, which requires a higher computation cost with many of the available algorithms. The adopted algorithm used for UDFS is given in Table 2.5.

Table 2.5 UDFS algorithm

Input: Training set x_i λ, γ

Output optimized weights vector W

1 **begin**
3 for $i = 1$ to N do begin
4 $B_i = \left(\widetilde{X}_i^T \widetilde{X}_i^T + \lambda I\right)^{-1}$

5 $\mathrm{M_i} = \mathrm{S_i H_{k+1} B_i H_{k+1} S_i^T}$

6 End

7 $$M = X\left(\sum_{i=1}^{n} M_i\right)X^T$$

8 set t $= 0$ and initialize $D_0 \in \mathbb{R}^{d\times d}$ as an identity matrix

9 do

11 $P_t = M + \gamma D_t$

12 $W_t = [p_1, \dots, p_c]$ where $p_1, \dots, p_c$ are the eigenvectors of P_t corresponding to the first c smallest eigenvalues

13 Update the diagonal matrix D_{t+1} as

$$D_{t+1} = \begin{bmatrix} \frac{1}{2\|w_t^1\|_2} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \frac{1}{2\|w_t^d\|_2} \end{bmatrix}$$

14 $t = t + 1$

22 until convergence

23 Sort each feature $f_i|_{i=1}^{d}$ according to $\|w_t^i\|_2 i$n descending order and select the top $-$ ranked ones.

CHAPTER 3

METHODOLOGY

3.1 Datasets and Simulation Environment

3.1.1 Datasets

Four different datasets have been used in this study. The datasets are extracted from the UCI Machine Learning Repository. The selection of the datasets is made in a way that they should be somehow sparse and difficult to achieve the clustering process using them. In fact, the selected datasets are not commonly used for clustering applications; rather, they have often been deployed in classification applications. In this case, it would be easy to evaluate the outcomes based on real datasets, which are hard to cluster. The summary of the used datasets is provided in Table 3.1.

3.1.1.1 Fertility Dataset

This dataset was donated to the UCI Machine Learning Repository in 2013 by Gil and Girela. The dataset is considered multivariate with 100 instances from 100 volunteers provided human semen for sample analysis according to WHO 2010 criteria. These 100 instances are characterized by 10 attributes or features. The considered features are as follow [70]:

Season: where the analysis took place along the four season of the year and different weights were assigned for each season (Winter = -1; Spring = -0.33; Summer = 0.33; Fall = 1).

Age: the age was recorded at the time of analysis and it was within the range 18 through 36. The weighted values assigned to these attributes are (0, 1).

Childish Diseases: the availability of some of the early age diseases were considered while taking the samples, such as chickenpox, measles, mumps, polio, etc. This feature was characterized by either available (yes = 0) or not available (no = 1). **Accident or Serious Trauma**: this feature refers to the availability of any traumatic or serious accidents the volunteer have had. The availability is characterized by (yes = 0) and the unavailability is characterized by (no = 1).

Surgical Intervention: this feature also refers to the availability of any surgical intervention history. The availability is characterized by (yes = 0) and the unavailability is characterized by (no = 1).

High Fever in the Last Year: this feature refers to the time of high fever incidents within the year precedes sample acquisition time if any. There are three different options characterizing this feature, namely less than three months (-1), more than three months (0), and no (1).

Frequency of Alcohol Consumption: this feature is assigned five different options ranging from (0, 1) as follows: 1) several times a day, 2) every day, 3) several times a week, 4) once a week, and 5) hardly ever or never.

Smoking Habit: this feature is characterized by three options: never = -1, occasional = 0, and daily = 1.

Number of Hours Spent Sitting per Day: this feature characterizes the number of hours spent sitting per day (in 16 hours) and the assigned weight to ranges from 0 to 1.

Output: this feature provides the diagnostic nature of the volunteer by either normal (N) or altered (O).

3.1.1.2 Immunotherapy Dataset

This dataset was donated to the UCI Machine Learning Repository by Khozeimeh and Layegh in 2018. The dataset is related to the treatment of wart in 90 patients by immunotherapy. The dataset is considered univariate and it's composed of 8 features. These features are as follows [71, 72]:

Response to Treatment: this feature describes the state of the patient in terms of his/her response to treatment using the applied immunotherapy method. The assigned values are Yes = 0 and No = 1.

Gender: this feature provides the gender of the patient which is characterized by Male = 0 and Female = 1. All samples are composed of 41 Men and 49 Women.

Age: this feature provides the age range of patients; the range within this dataset is from 15 through 56 years.

Time Elapsed before Treatment: this feature refers to the time preceding the application of immunotherapy to the addressed patients in months. The options here range from 0 through 12.

Number of Warts: this feature provides the total available warts on the body of the patient, and it ranges from 1 through 19.

Types of Wart: this feature describes the types of available warts. There are three types for the whole sample: 1-Common, 2-Plantar, and 3-Both.

Surface Area of the Warts: this feature provides the total surface area of available warts in the whole body of every patient in mm^2. This feature weight ranges from 6 mm^2 through 900 mm^2.

Induration Diameter or Initial Test: this feature provides the diameter, in mm, of the induration of the wart at the beginning of the treatment using the immunotherapy. The weight of the feature here ranges from 5 through 70 mms.

3.1.1.3 Cryotherapy Dataset

This dataset was donated to the UCI Machine Learning Repository by Khozeimeh and Layegh in 2018. The dataset is related to the treatment of wart in 90 patients by cryotherapy. The dataset is considered univariate and it's composed of 7 features; these are same as those of immunotherapy ones except for the last feature "Induration Diameter or Initial Test", which is omitted for cryotherapy's case [72, 73].

3.1.1.4 Wholesale Dataset

This dataset was donated to the UCI Machine Learning Repository by Cardoso in 2014. The dataset is related to the annual spending in Monetary Units (MU) for some clients of a wholesale distributor. The dataset is multivariate with integer attributes characteristics. The total number of features is 8, which are given as following [74]:

Fresh: this feature states the annual spending on fresh products by MU.

Milk: this feature states the annual spending on milk products by MU.

Grocery: this feature states the annual spending on grocery products by MU.

Frozen: this feature states the annual spending on frozen products by MU.

Detergents and Paper: this feature states the annual spending on detergents and paper products by MU.

Channel: this feature describes the used channel by the client. The feature is characterized by Horeca (Hotel/Restaurant/Cafe) or Retail channel, which is nominal.

Region: this feature provides information on the regional customers, which is also nominal and considers Lisnon, Oporto or Other regions.

Table 0.1 Summary of the used datasets

No.	Name	Instances #	Features #	Features Type	Field
1	*Fertility*	100	10	Real	Health Science
2	*Immunotherapy*	90	8	Numeric	Health Science
3	*Cryotherapy*	90	7	Numeric	Health Science
4	*Wholesale*	440	8	Integer	Business

3.1.2 Simulation Environment and Tool

The deployed simulation environment is Asus Notebook, Intel(R) Core (TM) i7, 64-bit CPU @ 3.4 GHz and 16 GB RAM. The used simulation software is MATLAB 2018b, where all codes are developed and written using it.

3.1.3 Evaluation Procedure and Metrics

In this work, the term *clusterability* refers to the ability of the deployed algorithm to properly cluster the dataset. This procedure is rather relative and it has different evaluation metrics, depending on the application being used. In order to come with a percentage outcome of the performance, the *purity* has been deployed as a metric for the produced clusters. This metric provides a measure for the external clustering quality (clusterability), which evaluates the internal clustering essence. Purity can be best described as a metric for inspecting to what extent every cluster is containing a unique class of data [75]. Assuming that the deployed algorithm is clustering the data objects into a number of clusters M with the total number of data objects D, where each formed $m \in M$ and each data object $d \in D$, and at the same time these data objects $d \in C$ (some data classes), the purity P can be calculated as [75]:

$$P = \frac{1}{D}\sum_{m \in M} max_{c \in C} |m \cap c| \tag{3.1}$$

Basically, the purity here is normalized and maximally can go up to 1.0, and it has been deployed as a floored percentage to a convenient comparison.

3.1.4 Experimental Setup

The used algorithms were all coded in MATLAB and a GUI was created for the convenient loading. A simple preprocessing procedure was applied prior to clustering the datasets by any algorithm. The aim of the preprocessing to alleviate any data deviation by normalizing the numeric data points. Let a data point or object d is available in the dataset, a map p is set to establish the normalization process, in the range [0, 1], as follows [76]:

$$\bar{d} = \frac{d - \min(p)}{\max(p) - \min(p)} \tag{3.2}$$

No further preprocessing was applied in order to ensure the clarity of the judgment of the deployed algorithms' performance. In addition, time complexity, algorithm speed, etc. were not quoted as metrics for the evaluation process, as the main purpose is not getting involved with detailed algorithms' analysis, rather, the judgment based on a numerical performance outcome satisfies the needed objective.

CHAPTER 4

RESULTS AND DISCUSSION

The presented algorithms have been implemented in MATLAB 2018b, and a useful Graphical User Interface (GUI) incorporated both the output clustering view, datasets loading and the performance of each algorithm. The developed GUI is shown in Figure 4.1. The designed GUI is very adaptable can accommodate any number of datasets. In addition, it possible to incorporate more algorithms by rearranging the components of the GUI. The datasets have been transformed into matrices and loaded to the same MATLAB folder, where all codes reside. The selection of datasets is very straightforward, where there is a list of all available datasets, and the user can choose any of them by clicking on it to activate the choice. The results are represented by the clusterability in a floored percentage displayed by analog gauges for each algorithm. In addition, a view of each algorithm's clustering processing is shown in 2-D plots for each case. Counters for the number of data points in each main clusters have been assigned for each algorithm. Finally, a summary for dataset details is given at the center top of the GUI.

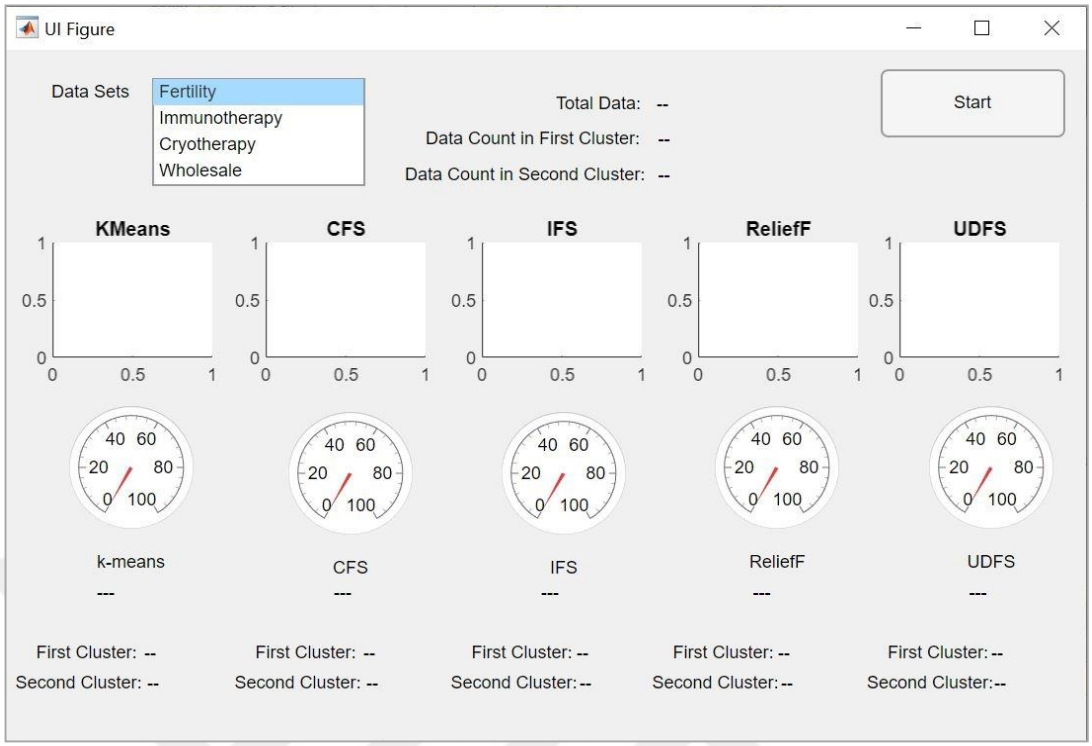

Figure 4.1 Main view of the developed GUI

The obtained results from the *fertility* dataset are shown in Figure 4.2. It is clear from the results that the ReliefF algorithm outperforms all other algorithms where its clustering ability (measured by purity as given in Chapter 3) reaches 64%. Compared to all algorithms, *k*-means has the lowest performs, which proves that, in many cases, the classical *k*-means is an inadequate clustering tool when the dimensionality of a matter of concern.

The same procedure has been applied to the *Immunotherapy* dataset and the obtained results are shown in Figure 4.3. Interestingly, the performances of all algorithms, except UDFS, are 44%. This is mainly due to the essence of the data samples' distribution within the dataset.

Analyzing the third dataset, *Cryotherapy* dataset produces the best performance by ReliefF, where the clustering ability reaches 76%. In addition, IFS produces a considerable result, in this case, 69%, compared to other algorithms. While the performance of the *k*-means algorithm remains below 50%, as in the previous datasets. The detailed results are shown in Figure 4.4.

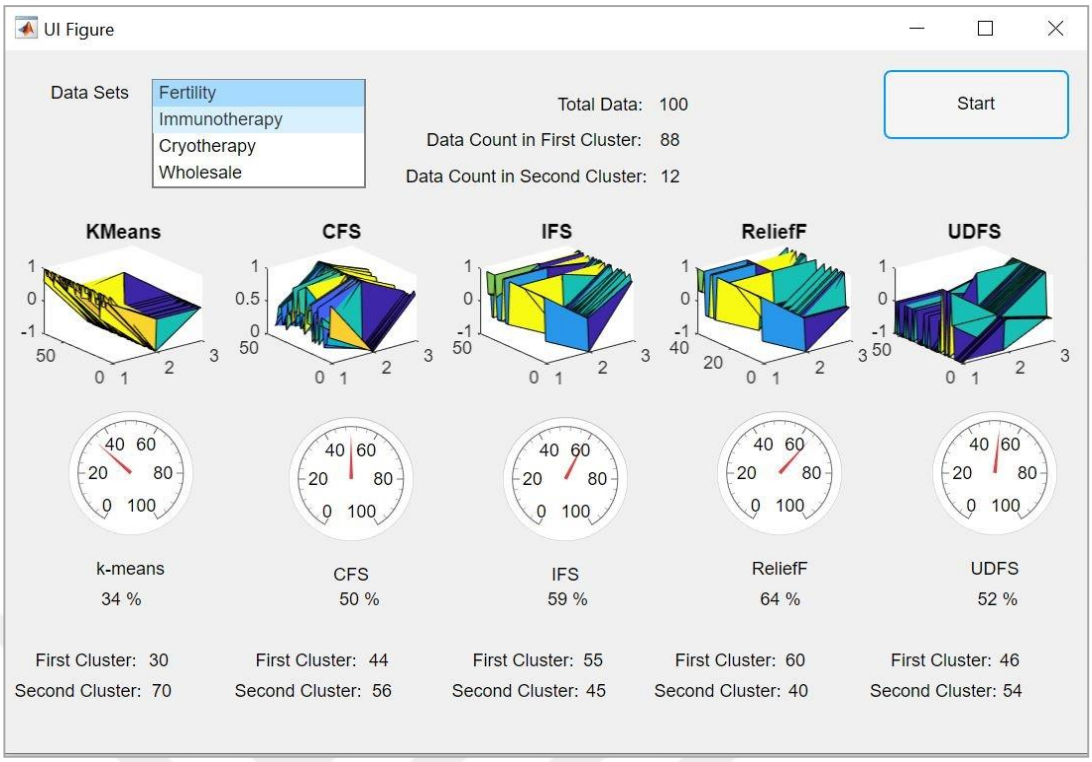

Figure 4.2 Clustering results of the *fertility* dataset

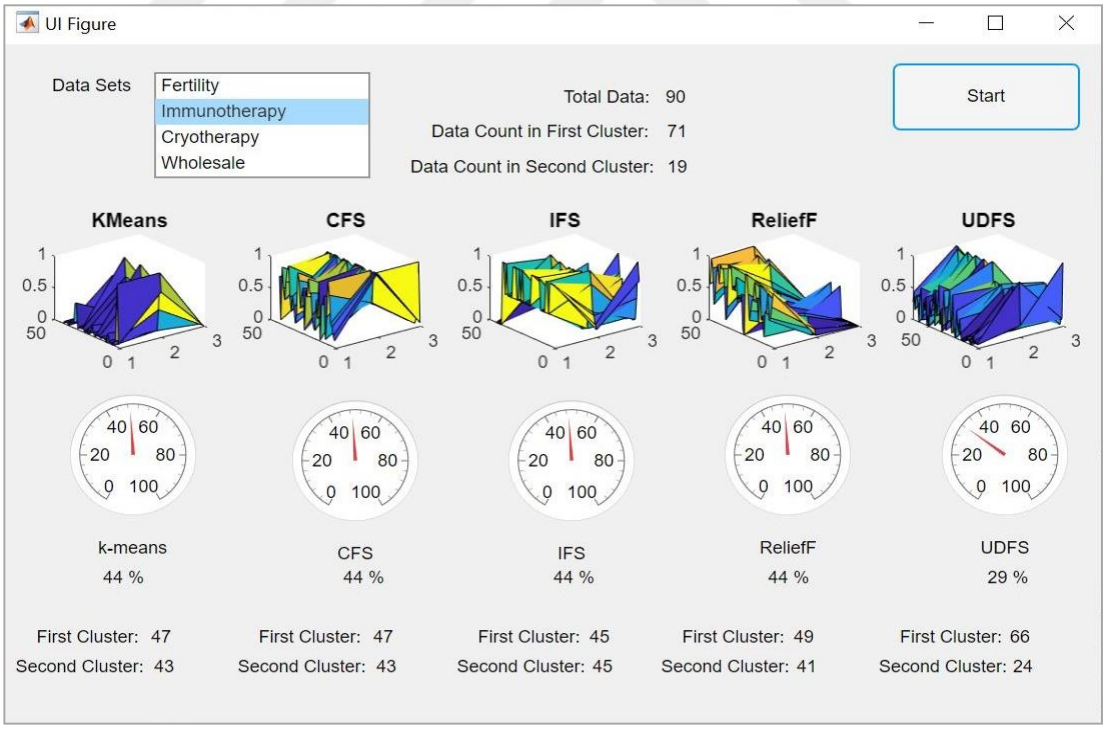

Figure 4.3 Clustering results of the Immunotherapy dataset

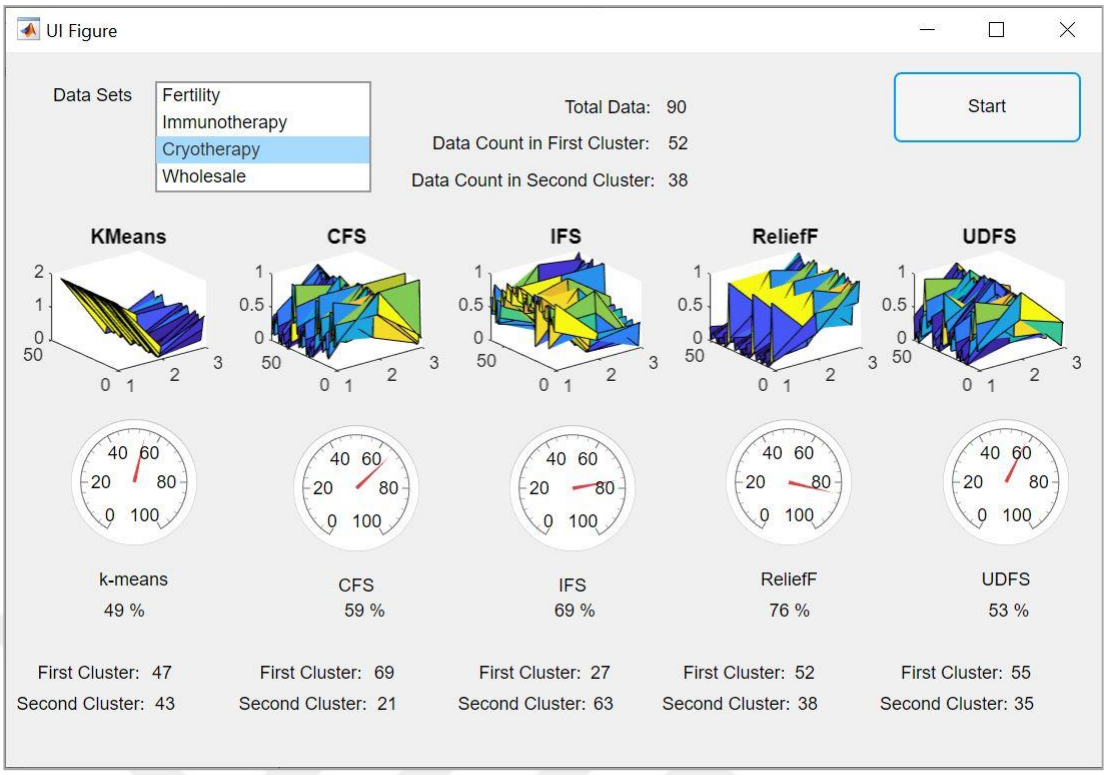

Figure 4.4 Clustering results of the Cryotherapy dataset

The last used dataset, *Wholesale* dataset, is considered one of the hardest due to the distribution of data samples as well as the number of data points. The best clustering performance is achieved by the CFS algorithm, which is 55%, and a very close performance has been achieved by the ReliefF algorithm, which is 53%. Although these results are not considered high in terms of clustering studies. However, using such datasets, this performance is considered a very good improvement for *k*-means. This is obviously clear from the obtained results by classical *k*-means, which is 3% or in other sense, approaching zero. To summarize all produced results, Table 4.1 provides the analysis details for the used datasets and clustering algorithms. The best performance is highlighted by the bold font for the recognition purpose.

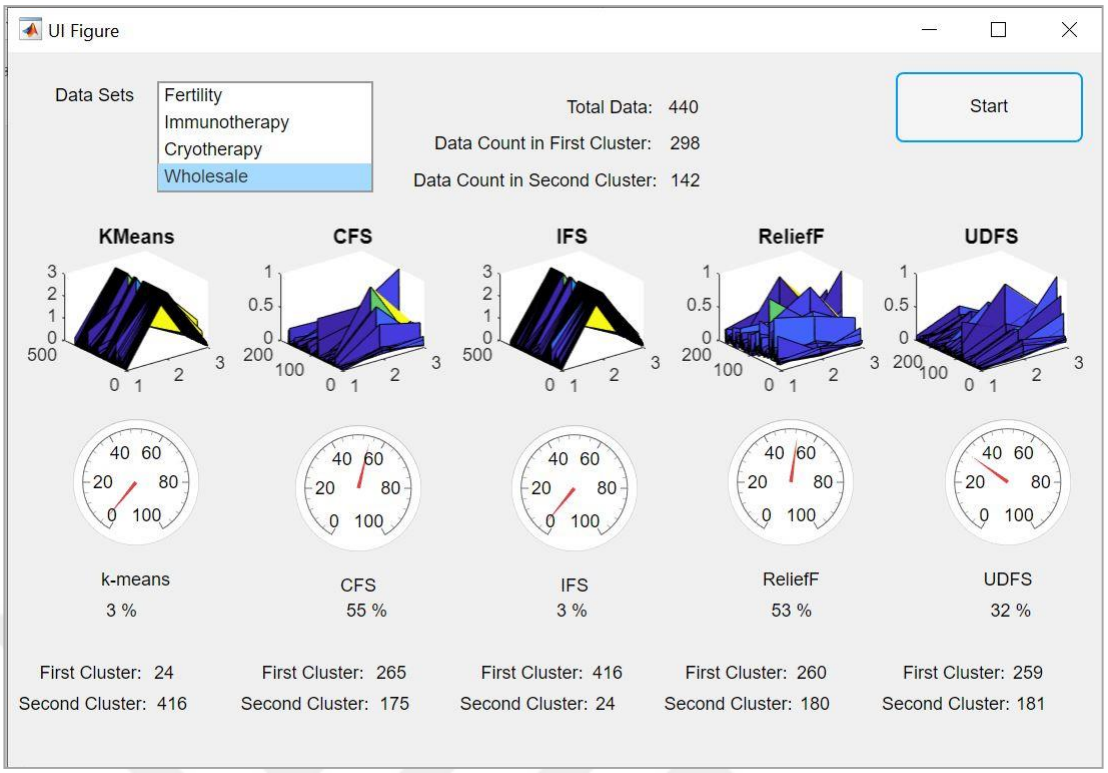

Figure 4.5 Clustering results of the Wholesale dataset

Table 4.1 Summary of algorithms' performances using selected datasets

Algorithm / Dataset	*k*-means	CFS	IFS	ReliefF	UDFS
Fertility	34	50	59	**64**	52
Immunotherapy	44	44	44	44	29
Cryotherapy	49	59	69	**76**	53
Wholesale	3	**55**	3	53	32

CHAPTER 5

CONCLUSION AND FUTURE WORK

This study has presented an investigation of subspace clustering for sparse data, where classical clustering methods are neither convenient nor efficient. The issue of using classical clustering methods, such as k-means, has been demonstrated in both theoretical and practical ways. Even though there are two main methods for improving classical clustering methods' performance in high-dimensional data clustering, it was decided to proceed with soft subspace clustering, as it is one of the promising mechanisms, which have been extensively investigated. Among the many methods applied in soft subspace reduction, four different algorithms where chosen based on their mechanism orientation to be incorporated to k-means for improving its performance. A developed GUI, using MATLAB 2018b, was used to evaluate the performance of the selected algorithms. The evaluation was based on real four datasets, which are uneasy to cluster, retrieved from the UCI Machine Learning Repository. Generally, the algorithms could perform better than *k*-means in all cases, except for one case due to the nature of the dataset, which proves that it is possible to improve the performance of the classical clustering using soft subspace dimensionality reduction, even for unusual datasets.

The tested algorithms have also shown that, in most cases, ReliefF algorithm could achieve the best outcomes, even though it was not close to optimal. The main issue with not achieving high results is the nature of the dataset, as the main aim was not guided towards producing an optimal solution; rather, it was aimed at enhancing the performance of *k*-means by incorporating these algorithms. Essentially, this aim has been achieved, and generally, above 50% improvement of the performance of *k*-means has been recorded.

One of the main limitations of this study is the lack of extensive testing results for different datasets, especially for clustering purpose. In fact, this can be a direction for a future study where the deployed algorithms can be tested using different synthesized and real datasets suitable for clustering purpose. In addition, the selection of algorithms can also be changed or more algorithms can be added for further classification purpose. It is also possible, for future studies, to deploy different levels of soft subspace dimensionality reduction techniques and investigate both the performance and running time. Moreover, recent trends in machine learning, such as deep-learning, swarming behavior as well as kernels can be deployed for exploring better clustering results.

REFERENCES

[1] Eldén, L. (2007). Vectors and Matrices in Data Mining and Pattern Recognition. *Matrix methods in data mining and pattern recognization. Society for Industrial and Applied Mathematics, Philadelphia*.

[2] Ghinita, G., Tao, Y., & Kalnis, P. (2008, April). On the anonymization of sparse high-dimensional data. In *IEEE 24th International Conference on Data Engineering* pp. 715-724.

[3] Donoho, D. L. (2000). High-dimensional data analysis: The curses and blessings of dimensionality. *AMS math challenges lecture*, *1*(2000), 32.

[4] Prendeki, J. (2018). The Curse of Big Data Labeling and Three Ways to Solve It. AWS. [Online]. Available at https://aws.amazon.com/blogs/apn/the-curse-of-big-data-labeling-and-three-ways-to-solve-it/ Accessed on 07.09, 2019.

[5] Moltzau, A. (2019). Advancements in Semi-Supervised Learning with Unsupervised Data Augmentation: Why is it important for the field of artificial intelligence? Towards Data Science. [Online]. https://towardsdatascience.com/advancements-in-semi-supervised-learning-with-unsupervised-data-augmentation-fc1fc0be3182 Accessed on 07.09.2019

[6] Schafer, J. L. (1997). *Analysis of Incomplete Multivariate Data*. Chapman and Hall/CRC.

[7] Elhamifar, E., & Vidal, R. (2009, June). Sparse Subspace Clustering. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* pp. 2790-2797.

[8] Steinbach, M., Ertöz, L., & Kumar, V. (2004). The challenges of clustering high dimensional data. In *New Directions in Statistical Physics* pp. 273-309. Springer, Berlin, Heidelberg.

[9] Gugger, M. (2012). Clustering High-dimensional Sparse Data. Bachelor Thesis, University of Zurich. Available at: https://www.merlin.uzh.ch/contributionDocument/download/3895 Accessed on 07.10, 2019.

[10] Zhang, D., Zhou, Z. H., & Chen, S. (2007, April). Semi-supervised dimensionality reduction. In Proceedings of the 2007 SIAM International

Conference on Data Mining pp. 629-634 . *Society for Industrial and Applied Mathematics.*

[11] Sedhai, S., & Sun, A. (2018). Semi-supervised spam detection in Twitter stream. IEEE Transactions on Computational Social Systems, **5(1)**, 169-175.

[12] Li, B., Lu, H., Zhang, Y., Lin, Z., & Wu, W. (2018). Subspace Clustering under Complex Noise. *IEEE Transactions on Circuits and Systems for Video Technology.*

[13] Kriegel, H. P., Kröger, P., & Zimek, A. (2009). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD),* **3(1),** 1.

[14] Günnemann, S., Kremer, H., & Seidl, T. (2010, April). Subspace clustering for uncertain data. In Proceedings of the 2010 SIAM International Conference on Data Mining (pp. 385-396). *Society for Industrial and Applied Mathematics*.

[15] Wang, J., Wang, S., Chung, F., & Deng, Z. (2013). Fuzzy partition based soft subspace clustering and its applications in high dimensional data. *Information Sciences,* **246**, 133-154.

[16] Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR),* **31(3),** 264-323.

[17] Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

[18] Parsons, L., Haque, E., & Liu, H. (2004). Subspace clustering for high dimensional data: a review. *Acm Sigkdd Explorations Newsletter*, **6(1)**, 90-105.

[19] Jing, L., Ng, M. K., & Huang, J. Z. (2007). An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Transactions on knowledge and data engineering,* **19(8).**

[20] Wang, Y. X., & Xu, H. (2016). Noisy sparse subspace clustering. *The Journal of Machine Learning Research,* **17(1),** 320-360.

[21] Jain, A. K. (2010). Data clustering: *50 years beyond K-means. Pattern recognition letters,* **31(8),** 651-666.

[22] Elhamifar, E., & Vidal, R. (2013). Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence,* **35(11),** 2765-2781.

[23] Salah, A., Rogovschi, N., & Nadif, M. (2016, May). Model-based co-clustering for high dimensional sparse data. In *Artificial Intelligence and Statistics* (pp. 866-874).

[24] Suresh, S., Saraswathi, S., & Sundararajan, N. (2010). Performance enhancement of extreme learning machine for multi-category sparse data classification problems. *Engineering Applications of Artificial Intelligence*, ***23*(7)**, 1149-1157.

[25] Hämäläinen, J., Kärkkäinen, T., & Rossi, T. (2018). Scalable robust clustering method for large and sparse data. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. ESANN.

[26] Steinbach, M., Ertöz, L., & Kumar, V. (2004). The challenges of clustering high dimensional data. In *New directions in statistical physics* pp. 273-309. Springer, Berlin, Heidelberg.

[27] Bissmark, J., & Wärnling, O. (2017). The Sparse Data Problem Within Classification Algorithms: The Effect of Sparse Data on the Naïve Bayes Algorithm. Bachelor Thesis.

[28] Zhang, R., & Lu, Z. (2016, July). Large Scale Sparse Clustering. In *IJCAI* pp. 2336-2342.

[29] Hussain, S. F., & Haris, M. (2019). A k-means based co-clustering (kCC) algorithm for sparse, high dimensional data. *Expert Systems with Applications*, *118*, 20-34.

[30] Bindi, F., Manzini, R., Pareschi, A., & Regattieri, A. (2007). Similarity coefficients and clustering techniques for the correlated assignment problem in warehousing systems. In *Proceedings of 19th International Conference on Production Research*.

[31] Sandhya, N., Lalitha, Y. S., Govardhan, A., & Anuradha, K. (2008). Analysis of similarity measures for text clustering. *CSC Journals*, 2.

[32] Kailing, K., Kriegel, H. P., & Kröger, P. (2004, April). Density-connected subspace clustering for high-dimensional data. In Proceedings of the 2004 SIAM International Conference on Data Mining pp. 246-256. *Society for Industrial and Applied Mathematics*.

[33] Elhamifar, E., & Vidal, R. (2009, June). Sparse subspace clustering. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. *IEEE Conference on* pp. 2790-2797.

[34] Zhu, W., Lu, J., & Zhou, J. (2018). Nonlinear subspace clustering for image clustering. Pattern Recognition Letters, 107, 131-136.

[35] Parsons, L., Haque, E., & Liu, H. (2004). Subspace clustering for high dimensional data: a review. *Acm Sigkdd Explorations Newsletter,* **6(1),** 90-105.

[36] Goyal, P., Kumari, S., Singh, S., Kishore, V., Balasubramaniam, S. S., & Goyal, N. (2016, October). A Parallel Framework for Grid-Based Bottom-Up Subspace Clustering. *In Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on* pp. 331-340.

[37] Nagesh, H., Goil, S., & Choudhary, A. (2001, April). Adaptive grids for clustering massive data sets. In Proceedings of the 2001 SIAM International Conference on Data Mining pp. 1-17. *Society for Industrial and Applied Mathematics.*

[38] Agrawal, R., Gehrke, J., Gunopulos, D., & Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications ACM. **Vol. 27(2),** pp. 94-105).

[39] Qian, W. N., & Zhou, A. Y. (2002). Analyzing popular clustering algorithms from different viewpoints. *Journal of software,* **13(8),** 1382-1394.

[40] Cheng, C. H., Fu, A. W., & Zhang, Y. (1999, August). Entropy-based subspace clustering for mining numerical data. In Proceedings of the fifth *ACM SIGKDD international conference on Knowledge discovery and data mining ACM*.pp. 84-93.

[41] Chang, J. W., & Jin, D. S. (2002, March). A new cell-based clustering method for large, high-dimensional data in data mining applications. In Proceedings of the *2002 ACM symposium on Applied computing ACM*. pp. 503-507.

[42] Liu, B., Xia, Y., & Yu, P. S. (2000, November). Clustering through decision tree construction. In Proceedings of the ninth international conference on *Information and knowledge management ACM*. pp. 20-29.

[43] Jahirabadkar, S., & Kulkarni, P. (2013). Clustering for High Dimensional Data: Density based Subspace Clustering Algorithms. International Journal of *Computer Applications*, **63(20).**

[44] Goil, S., Nagesh, H., & Choudhary, A. (1999, June). MAFIA: Efficient and scalable subspace clustering for very large data sets. In Proceedings of the 5th

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining ACM. pp. 443-452

[45] Hwang, J. J., Whang, K. Y., Moon, Y. S., & Lee, B. S. (2004). A top-down approach for density-based clustering using multidimensional indexes. *Journal of Systems and Software*, **73(1),** 169-180.

[46] Woo, K. G., Lee, J. H., Kim, M. H., & Lee, Y. J. (2004). FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, **46(4),** 255-271.

[47] Pham, D. S., Arandjelović, O., & Venkatesh, S. (2016, July). Achieving stable subspace clustering by post-processing generic clustering results. *In Neural Networks (IJCNN), 2016 International Joint Conference on* pp. 2390-2396.

[48] Aggarwal, C. C., & Yu, P. S. (2000). Finding generalized projected clusters in high dimensional spaces *ACM*. **Vol 29**(2), pp. 70-81

[49] Aggarwal, C. C., Wolf, J. L., Yu, P. S., Procopiuc, C., & Park, J. S. (1999, June). Fast algorithms for projected clustering. *In ACM SIGMoD Record* ACM. **Vol. 28(2)**, pp. 61-72).

[50] Yang, J., Wang, W., Wang, H., & Yu, P. (2002, February). d-clusters: Capturing subspace correlation in a large data set. In icde (p. 0517). *IEEE*.

[51] Böhm, C., Kailing, K., Kröger, P., & Zimek, A. (2004, June). Computing clusters of correlation connected objects. In Proceedings of the 2004 *ACM SIGMOD international conference on Management of data ACM* pp. 455-466.

[52] Domeniconi, C., Papadopoulos, D., Gunopulos, D., & Ma, S. (2004, April). Subspace clustering of high dimensional data. In Proceedings of the 2004 *SIAM international conference on data mining* pp. 517-521. *Society for Industrial and Applied Mathematics.*

[53] Friedman, J. H., & Meulman, J. J. (2004). Clustering objects on subsets of attributes (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **66(4),** 815-849.

[54] DeSarbo, W. S., Carroll, J. D., Clark, L. A., & Green, P. E. (1984). Synthesized clustering: A method for amalgamating alternative clustering bases with differential weighting of variables. *Psychometrika*, **49(1),** 57-78.

[55] De Soete, G. (1988). OVWTRE: A program for optimal variable weighting for ultrametric and additive tree fitting. *Journal of Classification***, 5(1),** 101-104.

[56] Makarenkov, V., & Legendre, P. (2001). Optimal variable weighting for ultrametric and additive trees and K-means partitioning: *Methods and software. Journal of Classification,* **18(2),** 245-271.

[57] Modha, D. S., & Spangler, W. S. (2003). *Feature weighting in k-means clustering. Machine learning,* **52(3),** 217-237.

[58] Huang, J. Z., Ng, M. K., Rong, H., & Li, Z. (2005). Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **27(5),** 657-668.

[59] Gan, G., & Wu, J. (2008). *A convergence theorem for the fuzzy subspace clustering (FSC) algorithm. Pattern Recognition,* **41(6),** 1939-1947.

[60] Deng, Z., Choi, K. S., Chung, F. L., & Wang, S. (2010). *Enhanced soft subspace clustering integrating within-cluster and between-cluster information. Pattern Recognition***, 43(3),** 767-781.

[61] Boongoen, T., Shang, C., Iam-On, N., & Shen, Q. (2011). Extending data reliability measure to a filter approach for soft subspace clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* **41(6),** 1705-1714.

[62] De Amorim, R. C., & Mirkin, B. (2012). Minkowski metric, feature weighting and anomalous cluster initializing in K-Means clustering. *Pattern Recognition,* **45(3),** 1061-1075.

[63] Chitsaz, E., & Jahromi, M. Z. (2016). A novel soft subspace clustering algorithm with noise detection for high dimensional datasets. *Soft Computing,* **20(11),** 4463-4472.

[64] Deng, Z., Choi, K. S., Jiang, Y., Wang, J., & Wang, S. (2016). A survey on soft subspace clustering. *Information sciences*, **348**, 84-106.

[65] Li, Y., Liang, X., Lu, Y., & Jiao, L. (2017, November). Soft subspace clustering using QPSOSC algorithm. *2017 IEEE Symposium Series on Computational Intelligence (SSCI),* (pp. 1-8).

[66] Harikumar, S., & Akhil, A. S. (2018). Semi supervised approach towards subspace clustering. *Journal of Intelligent & Fuzzy Systems,* **34(3),** 1619-1629.

[67] Xu, P., Deng, Z., Cui, C., Zhang, T., Choi, K. S., Suhang, G., Wang, J., & Wang, S. (2019). Concise Fuzzy System Modeling Integrating Soft Subspace Clustering and Sparse Learning. *IEEE Transactions on Fuzzy Systems*.

[68] Yang, S., Zhu, W., & Zhu, Y. (2019). Three-Stage Subspace Clustering Framework with Graph-Based Transformation and Optimization. *arXiv preprint arXiv:1905.01145*.

[69] Pang, N., Zhang, J., Zhang, C., Qin, X., & Cai, J. (2019). PUMA: Parallel Subspace Clustering of Categorical Data using Multi-attribute Weights. *Expert Systems with Applications*, **126**, 233-245.

[70] https://archive.ics.uci.edu/ml/datasets/Fertility Accessed on 22.12.2018

[71] https://archive.ics.uci.edu/ml/datasets/Immunotherapy+Dataset(Accessed on 22.12.2018.

[72] Khozeimeh, F., Alizadehsani, R., Roshanzamir, M., Khosravi, A., Layegh, P., & Nahavandi, S. (2017). An expert system for selecting wart treatment method. *Computers in biology and medicine*, *81*, 167-175.

[73] https://archive.ics.uci.edu/ml/datasets/Cryotherapy+Dataset+ Accessed on 22.12.2018.

[74] https://archive.ics.uci.edu/ml/datasets/wholesale+customers(Accessed on 22.12.2018.

[75] Amigó, E., Gonzalo, J., Artiles, J., & Verdejo, F. (2009). A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, ***12*(4)**, 461-486.

[76] Xiong, Z., & Zhang, Z. (2016, January). A Data Preprocessing Method Applied to Cluster Analysis on Stock Data by Kmeans. In *2016 International Conference on Intelligent Control and Computer Application (ICCA 2016)*. Atlantis Press.

[77] Roffo, G., Melzi, S., & Cristani, M. (2015). Infinite feature selection. In Proceedings of the *IEEE International Conference on Computer Vision* pp. 4202-4210.

[78] Hindawi, M., Allab, K., & Benabdeslem, K. (2011). Constraint selection-based semi-supervised feature selection. *IEEE 11th International Conference on Data Mining (ICDM)*, pp. 1080-1085.

[79] Kononenko, I. (1994, April). Estimating attributes: analysis and extensions of RELIEF. *In European conference on machine learning*. Springer, Berlin, Heidelberg. pp. 171-182.

[80] Yang, Y., Shen, H. T., Ma, Z., Huang, Z., & Zhou, X. (2011, July). l2, 1-norm regularized discriminative feature selection for unsupervised learning. *In IJCAI*

proceedings-international joint conference on artificial intelligence **Vol. 22(1)** p. 1589.

[81] Zhou, P. Y., & Chan, K. C. (2014, May). A model-based multivariate time series clustering algorithm. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, Cham pp. 805-817.

APPENDEX

```
classdef gui < matlab.apps.AppBase
    % app components properties
    properties (Access = public)
        a1                      matlab.ui.control.UIAxes
        a2                      matlab.ui.control.UIAxes
        a3                      matlab.ui.control.UIAxes
        a4                      matlab.ui.control.UIAxes
        a5                      matlab.ui.control.UIAxes
        lbl21                   matlab.ui.control.Label
        lbl22                   matlab.ui.control.Label
        lbl31                   matlab.ui.control.Label
        lbl32                   matlab.ui.control.Label
        lbl41                   matlab.ui.control.Label
        lbl42                   matlab.ui.control.Label
        lbl51                   matlab.ui.control.Label
        lbl52                   matlab.ui.control.Label

        UIFigure                matlab.ui.Figure
        StartButton             matlab.ui.control.Button
        kmeansLabel             matlab.ui.control.Label
        g1                      matlab.ui.control.Gauge
        CFSLabel                matlab.ui.control.Label
        g2                      matlab.ui.control.Gauge
        IFSLabel                matlab.ui.control.Label
        g3                      matlab.ui.control.Gauge
        ReliefFLabel            matlab.ui.control.Label
        g4                      matlab.ui.control.Gauge
        UDFSLabel               matlab.ui.control.Label
        g5                      matlab.ui.control.Gauge
        l1                      matlab.ui.control.Label
        l2                      matlab.ui.control.Label
        l3                      matlab.ui.control.Label
        l4                      matlab.ui.control.Label
        l5                      matlab.ui.control.Label
        aaaa                    matlab.ui.control.Label
        l1_3                    matlab.ui.control.Label
        l1_4                    matlab.ui.control.Label
        u1                      matlab.ui.control.Label
        u2                      matlab.ui.control.Label
        u3                      matlab.ui.control.Label
        l1_8                    matlab.ui.control.Label
        l1_9                    matlab.ui.control.Label
        l1_10                   matlab.ui.control.L
```

```
l1_11                   matlab.ui.control.Label
        l1_12                   matlab.ui.control.Label
        l1_13                   matlab.ui.control.Label
        l1_14                   matlab.ui.control.Label
        l1_15                   matlab.ui.control.Label
        l1_16                   matlab.ui.control.Label
        l1_17                   matlab.ui.control.Label
        lbl11                   matlab.ui.control.Label
        lbl12                   matlab.ui.control.Label

        DataSetsListBoxLabel    matlab.ui.control.Label
        DataSetsListBox         matlab.ui.control.ListBox
    end

    properties (Access = public)
        y; % Description
        y2;
        y3;
        y4;
       deger;
    end

    methods (Access = public)

        function  main(app)
            a1='Fertility';
            a1=convertCharsToStrings(a1);
            a2='Immunotherapy';
            a2=convertCharsToStrings(a2);
            a3='Cryotherapy';
            a3=convertCharsToStrings(a3);
            a4='Wholesale';
            a4=convertCharsToStrings(a4);
            %dizi=rand(100,2);
            %plot(app.a1,dizi(:,1:1),dizi(:,2:2),'*g');
            app.y=load('veri_v1.mat');
            app.y2=load('veri_v2.mat');
            app.y3=load('veri_v3.mat');
            app.y4=load('veri_v4.mat');

            f=app.deger;
            if size(f,1)==0
            a=a1;
            else
            a=convertCharsToStrings(f);
            end

            X=app.y.veri_v1;
            X2=app.y2.veri_v2;
            X3=app.y3.veri_v3;
            X4=app.y4.veri_v4;
```

```
            if a==a1
                app.u1.Text=cellstr(sprintf('100'));
                app.u2.Text=cellstr(sprintf('88'));
                app.u3.Text=cellstr(sprintf('12'));
                veri=[X(:,6:8) X(:,10:11)];
                [dogru,c1,c2]=kmeans(veri,400);
                app.g1.Value=dogru;
                app.l1.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl11.Text=cellstr(sprintf(' %d
',size(c2,1)));
                app.lbl12.Text=cellstr(sprintf(' %d
',size(c1,1)));

                ax = app.a1;
                s = surf(ax,c1(:,2:4));
                zoom(ax,'on')

                cfsgelen=cfs(X(:,1:9));
                veri=[X(:,cfsgelen(7):cfsgelen(7))
X(:,cfsgelen(8):cfsgelen(8)) X(:,cfsgelen(9):cfsgelen(9))
X(:,10:11)];
                [dogru,c1,c2]=kmeans(veri,400);
                app.g2.Value=dogru;
                app.l2.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl21.Text=cellstr(sprintf(' %d
',size(c1,1)));
                app.lbl22.Text=cellstr(sprintf(' %d
',size(c2,1)));

                ax = app.a2;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')

                IIcfsgelen=llcfs(X(:,1:9));
                veri=[X(:,IIcfsgelen(1):IIcfsgelen(1))
X(:,IIcfsgelen(2):IIcfsgelen(2))
X(:,IIcfsgelen(3):IIcfsgelen(3)) X(:,10:11)];
                [dogru,c1,c2]=kmeans(veri,400);
                app.g3.Value=dogru;
                app.l3.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl31.Text=cellstr(sprintf(' %d
',size(c2,1)));
                app.lbl32.Text=cellstr(sprintf(' %d
',size(c1,1)));

                ax = app.a3;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')
```

```
gelenreliefF=reliefF(X(:,1:9),X(:,10:10),3);

veri=[X(:,gelenreliefF(1):gelenreliefF(1))
X(:,gelenreliefF(2):gelenreliefF(2))
X(:,gelenreliefF(3):gelenreliefF(3)) X(:,10:11)];
                [dogru,c1,c2]=kmeans(veri,400);
                app.g4.Value=dogru;
                app.l4.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl41.Text=cellstr(sprintf(' %d
',size(c2,1)));
                app.lbl42.Text=cellstr(sprintf(' %d
',size(c1,1)));

                ax = app.a4;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')

                udfsgelen=UDFS(X(:,1:9),2);
                veri=[X(:,udfsgelen(7):udfsgelen(7))
X(:,udfsgelen(8):udfsgelen(8))
X(:,udfsgelen(9):udfsgelen(9)) X(:,10:11)];
                [dogru,c1,c2]=kmeans(veri,400);
                app.g5.Value=dogru;
                app.l5.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl51.Text=cellstr(sprintf(' %d
',size(c2,1)));
                app.lbl52.Text=cellstr(sprintf(' %d
',size(c1,1)));

                ax = app.a5;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')

            elseif a==a2

                app.u1.Text=cellstr(sprintf('90'));
                app.u2.Text=cellstr(sprintf('71'));
                app.u3.Text=cellstr(sprintf('19'));

                veri=[X2(:,5:7) X2(:,8:9)];
                [dogru,c1,c2]=kmeans(veri,20);
                app.g1.Value=dogru*100/size(veri,1);
                app.l1.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl11.Text=cellstr(sprintf(' %d
',size(c2,1)));
```

```
                app.lbl12.Text=cellstr(sprintf(' %d
',size(c1,1)));

                ax = app.a1;
                s = surf(ax,c1(:,2:4));
                zoom(ax,'on')


                cfsgelen=cfs(X2(:,1:7));
                veri=[X2(:,cfsgelen(5):cfsgelen(5))
X2(:,cfsgelen(6):cfsgelen(6))
X2(:,cfsgelen(7):cfsgelen(7)) X2(:,8:9)];
                [dogru,c1,c2]=kmeans(veri,20);
                app.g2.Value=dogru*100/size(veri,1);
                app.l2.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl21.Text=cellstr(sprintf(' %d
',size(c2,1)));
                app.lbl22.Text=cellstr(sprintf(' %d
',size(c1,1)));

                ax = app.a2;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')
                IIcfsgelen=llcfs(X2(:,1:7));
                veri=[X2(:,IIcfsgelen(1):IIcfsgelen(1))
X2(:,IIcfsgelen(2):IIcfsgelen(2))
X2(:,IIcfsgelen(3):IIcfsgelen(3)) X2(:,8:9)];
                [dogru,c1,c2]=kmeans(veri,20);
                app.g3.Value=dogru*100/size(veri,1);
                app.l3.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl31.Text=cellstr(sprintf(' %d
',size(c1,1)));
                app.lbl32.Text=cellstr(sprintf(' %d
',size(c2,1)));

                ax = app.a3;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')


gelenreliefF=reliefF(X2(:,1:7),X2(:,8:8),3);

veri=[X2(:,gelenreliefF(1):gelenreliefF(1))
X2(:,gelenreliefF(2):gelenreliefF(2))
X2(:,gelenreliefF(3):gelenreliefF(3)) X2(:,8:9)];
                [dogru,c1,c2]=kmeans(veri,20);
                app.g4.Value=dogru*100/size(veri,1);
                app.l4.Text=cellstr(sprintf(' %.f
%%',dogru));
```

```
                app.lbl41.Text=cellstr(sprintf(' %d
',size(c1,1)));
                app.lbl42.Text=cellstr(sprintf(' %d
',size(c2,1)));

                ax = app.a4;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')

                udfsgelen=UDFS(X2(:,1:7),2);
                veri=[X2(:,udfsgelen(5):udfsgelen(5))
X2(:,udfsgelen(6):udfsgelen(6))
X2(:,udfsgelen(7):udfsgelen(7)) X2(:,8:9)];
                [dogru,c1,c2]=kmeans(veri,20);
                app.g5.Value=dogru*100/size(veri,1);
                app.l5.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl51.Text=cellstr(sprintf(' %d
',size(c1,1)));
                app.lbl52.Text=cellstr(sprintf(' %d
',size(c2,1)));

                ax = app.a5;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')

            elseif a==a3

                 app.u1.Text=cellstr(sprintf('90'));
                app.u2.Text=cellstr(sprintf('52'));
                app.u3.Text=cellstr(sprintf('38'));

                veri=[X3(:,1:3) X3(:,7:8)];
                [dogru,c1,c2]=kmeans(veri,20);
                app.g1.Value=dogru*100/size(veri,1);
                app.l1.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl11.Text=cellstr(sprintf(' %d
',size(c2,1)));
                app.lbl12.Text=cellstr(sprintf(' %d
',size(c1,1)));

                ax = app.a1;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')

                cfsgelen=cfs(X3(:,1:6));
                veri=[X3(:,cfsgelen(4):cfsgelen(4))
X3(:,cfsgelen(5):cfsgelen(5))
X3(:,cfsgelen(6):cfsgelen(6)) X3(:,7:8)];
                [dogru,c1,c2]=kmeans(veri,20);
```

```
                app.g2.Value=dogru*100/size(veri,1);
                app.l2.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl21.Text=cellstr(sprintf(' %d
',size(c1,1)));
                app.lbl22.Text=cellstr(sprintf(' %d
',size(c2,1)));

                ax = app.a2;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')


                IIcfsgelen=llcfs(X3(:,1:6));
                veri=[X3(:,IIcfsgelen(1):IIcfsgelen(1))
X3(:,IIcfsgelen(2):IIcfsgelen(2))
X3(:,IIcfsgelen(3):IIcfsgelen(3)) X3(:,7:8)];
                [dogru,c1,c2]=kmeans(veri,20);
                app.g3.Value=dogru*100/size(veri,1);
                app.l3.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl31.Text=cellstr(sprintf(' %d
',size(c2,1)));
                app.lbl32.Text=cellstr(sprintf(' %d
',size(c1,1)));

                ax = app.a3;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')


gelenreliefF=reliefF(X3(:,1:6),X2(:,7:7),3);

veri=[X3(:,gelenreliefF(1):gelenreliefF(1))
X3(:,gelenreliefF(2):gelenreliefF(2))
X3(:,gelenreliefF(3):gelenreliefF(3)) X3(:,7:8)];
                [dogru,c1,c2]=kmeans(veri,20);
                app.g4.Value=dogru*100/size(veri,1);
                app.l4.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl41.Text=cellstr(sprintf(' %d
',size(c1,1)));
                app.lbl42.Text=cellstr(sprintf(' %d
',size(c2,1)));

                ax = app.a4;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')

                udfsgelen=UDFS(X3(:,1:6),2);
```

```
                veri=[X3(:,udfsgelen(4):udfsgelen(4))
X3(:,udfsgelen(5):udfsgelen(5))
X3(:,udfsgelen(6):udfsgelen(6)) X3(:,7:8)];
                [dogru,c1,c2]=kmeans(veri,20);
                app.g5.Value=dogru*100/size(veri,1);
                app.l5.Text=cellstr(sprintf(' %.f
%%',dogru));
                app.lbl51.Text=cellstr(sprintf(' %d
',size(c1,1)));
                app.lbl52.Text=cellstr(sprintf(' %d
',size(c2,1)));

                ax = app.a5;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')
            elseif a==a4
                app.u1.Text=cellstr(sprintf('440'));
                app.u2.Text=cellstr(sprintf('298'));
                app.u3.Text=cellstr(sprintf('142'));

                cfsgelen=cfs(X4(:,1:7));
                veri=[X4(:,cfsgelen(1):cfsgelen(1))
X4(:,cfsgelen(2):cfsgelen(2))
X4(:,cfsgelen(3):cfsgelen(3)) X4(:,8:9)];
                [dogru,c1,c2]=kmeans(veri,500);
                app.g1.Value=dogru*100/size(veri,1);
                app.l1.Text=cellstr(sprintf(' %.f
%%',dogru*100/size(veri,1)));
                app.lbl11.Text=cellstr(sprintf(' %d
',size(c2,1)));
                app.lbl12.Text=cellstr(sprintf(' %d
',size(c1,1)));

                 ax = app.a1;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')


                cfsgelen=cfs(X4(:,1:7));
                veri=[X4(:,cfsgelen(5):cfsgelen(5))
X4(:,cfsgelen(6):cfsgelen(6))
X4(:,cfsgelen(7):cfsgelen(7)) X4(:,8:9)];
                size(veri,1)
                [dogru,c1,c2]=kmeans(veri,400);
                app.g2.Value=dogru*100/size(veri,1);
                app.l2.Text=cellstr(sprintf(' %.f
%%',dogru*100/size(veri,1)));
                app.lbl21.Text=cellstr(sprintf(' %d
',size(c2,1)));
                app.lbl22.Text=cellstr(sprintf(' %d
',size(c1,1)));
```

```
                ax = app.a2;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')


                IIcfsgelen=llcfs(X4(:,1:7));
                veri=[X4(:,IIcfsgelen(1):IIcfsgelen(1))
X4(:,IIcfsgelen(2):IIcfsgelen(2))
X4(:,IIcfsgelen(3):IIcfsgelen(3)) X4(:,8:9)];
                [dogru,c1,c2]=kmeans(veri,400);
                app.g3.Value=dogru*100/size(veri,1);
                app.l3.Text=cellstr(sprintf(' %.f
%%',dogru*100/size(veri,1)));
                app.lbl31.Text=cellstr(sprintf(' %d
',size(c1,1)));
                app.lbl32.Text=cellstr(sprintf(' %d
',size(c2,1)));

                ax = app.a3;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')



gelenreliefF=reliefF(X3(:,1:7),X2(:,8:8),3);

veri=[X4(:,gelenreliefF(1):gelenreliefF(1))
X4(:,gelenreliefF(2):gelenreliefF(2))
X4(:,gelenreliefF(3):gelenreliefF(3)) X4(:,8:9)];
                [dogru,c1,c2]=kmeans(veri,400);
                app.g4.Value=dogru*100/size(veri,1);
                app.l4.Text=cellstr(sprintf(' %.f
%%',dogru*100/size(veri,1)));
                app.lbl41.Text=cellstr(sprintf(' %d
',size(c2,1)));
                app.lbl42.Text=cellstr(sprintf(' %d
',size(c1,1)));

                ax = app.a4;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')


                udfsgelen=UDFS(X4(:,1:7),2);
                veri=[X4(:,udfsgelen(5):udfsgelen(5))
X4(:,udfsgelen(6):udfsgelen(6))
X4(:,udfsgelen(7):udfsgelen(7)) X4(:,8:9)];
                [dogru,c1,c2]=kmeans(veri,400);
                app.g5.Value=dogru*100/size(veri,1);
```

```
                app.l5.Text=cellstr(sprintf(' %.f
%%',dogru*100/size(veri,1)));
                app.lbl51.Text=cellstr(sprintf(' %d
',size(c1,1)));
                app.lbl52.Text=cellstr(sprintf(' %d
',size(c2,1)));

                ax = app.a5;
                s = surf(ax,c1(:,1:3));
                zoom(ax,'on')

            end

        end

    end

    methods (Access = private)
        % Code that executes after component creation
        function startupFcn(app)

        end
        % Button pushed function: StartButton
        function StartButtonPushed(app, event)
            main(app);

        end
        % Value changed function: DataSetsListBox
        function DataSetsListBoxValueChanged(app, event)
            value = app.DataSetsListBox.Value;
            app.deger=value;

        end
    end
    % App initialization and construction
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [225 150 730 475];
            app.UIFigure.Name = 'UI Figure';
            % Create StartButton
            app.StartButton = uibutton(app.UIFigure,
'push');
            app.StartButton.ButtonPushedFcn =
createCallbackFcn(app, @StartButtonPushed, true);
            app.StartButton.Position = [592 417 125 47];
            app.StartButton.Text = 'Start';
            % Create kmeansLabel
            app.kmeansLabel = uilabel(app.UIFigure);
```

```
            app.kmeansLabel.HorizontalAlignment =
'center';
            app.kmeansLabel.VerticalAlignment = 'top';
            app.kmeansLabel.Position = [60 112 52 22];
            app.kmeansLabel.Text = {'k-means'; ''};
            % Create g1
            app.g1 = uigauge(app.UIFigure, 'circular');
            app.g1.Position = [44 149 84 84];
            % Create CFSLabel
            app.CFSLabel = uilabel(app.UIFigure);
            app.CFSLabel.HorizontalAlignment = 'center';
            app.CFSLabel.VerticalAlignment = 'top';
            app.CFSLabel.Position = [220 115 29 15];
            app.CFSLabel.Text = 'CFS';
            % Create g2
            app.g2 = uigauge(app.UIFigure, 'circular');
            app.g2.Position = [192 145 84 84];
            % Create IFSLabel
            app.IFSLabel = uilabel(app.UIFigure);
            app.IFSLabel.HorizontalAlignment = 'center';
            app.IFSLabel.VerticalAlignment = 'top';
            app.IFSLabel.Position = [361 108 35 22];
            app.IFSLabel.Text = 'IFS';
            % Create g3
            app.g3 = uigauge(app.UIFigure, 'circular');
            app.g3.Position = [336 145 84 84];
            % Create ReliefFLabel
            app.ReliefFLabel = uilabel(app.UIFigure);
            app.ReliefFLabel.HorizontalAlignment =
'center';
            app.ReliefFLabel.VerticalAlignment = 'top';
            app.ReliefFLabel.Position = [500.5 112 44
22];
            app.ReliefFLabel.Text = 'ReliefF';
            % Create g4
            app.g4 = uigauge(app.UIFigure, 'circular');
            app.g4.Position = [480 149 84 84];
            % Create UDFSLabel
            app.UDFSLabel = uilabel(app.UIFigure);
            app.UDFSLabel.HorizontalAlignment = 'center';
            app.UDFSLabel.VerticalAlignment = 'top';
            app.UDFSLabel.Position = [648 119 38 15];
            app.UDFSLabel.Text = 'UDFS';
            % Create g5
            app.g5 = uigauge(app.UIFigure, 'circular');
            app.g5.Position = [625 149 84 84];
            % Create l1
            app.l1 = uilabel(app.UIFigure);
            app.l1.VerticalAlignment = 'top';
            app.l1.Position = [63 98 80 15];
            app.l1.Text = '---';
```

```
            % Create l2
            app.l2 = uilabel(app.UIFigure);
            app.l2.VerticalAlignment = 'top';
            app.l2.Position = [223 98 86 15];
            app.l2.Text = '---';
            % Create l3
            app.l3 = uilabel(app.UIFigure);
            app.l3.VerticalAlignment = 'top';
            app.l3.Position = [363 98 78 15];
            app.l3.Text = '---';
            % Create l4
            app.l4 = uilabel(app.UIFigure);
            app.l4.VerticalAlignment = 'top';
            app.l4.Position = [506 98 79 15];
            app.l4.Text = '---';
            % Create l5
            app.l5 = uilabel(app.UIFigure);
            app.l5.VerticalAlignment = 'top';
            app.l5.Position = [652 98 79 15];
            app.l5.Text = '---';
            % Create aaaa
            app.aaaa = uilabel(app.UIFigure);
            app.aaaa.VerticalAlignment = 'top';
            app.aaaa.Position = [373 433 62 15];
            app.aaaa.Text = 'Total Data:';
            % Create l1_3
            app.l1_3 = uilabel(app.UIFigure);
            app.l1_3.VerticalAlignment = 'top';
            app.l1_3.Position = [285 402 157 22];
            app.l1_3.Text = 'Data Count in First
Cluster:';
            % Create l1_4
            app.l1_4 = uilabel(app.UIFigure);
            app.l1_4.VerticalAlignment = 'top';
            app.l1_4.Position = [271 377 172 22];
            app.l1_4.Text = 'Data Count in Second
Cluster:';
            % Create u1
            app.u1 = uilabel(app.UIFigure);
            app.u1.VerticalAlignment = 'top';
            app.u1.Position = [441 433 62 15];
            app.u1.Text = '--';
            % Create u2
            app.u2 = uilabel(app.UIFigure);
            app.u2.VerticalAlignment = 'top';
            app.u2.Position = [442 409 62 15];
            app.u2.Text = '--';
            % Create u3
            app.u3 = uilabel(app.UIFigure);
            app.u3.VerticalAlignment = 'top';
            app.u3.Position = [442 384 62 15];
```

```
app.u3.Text = '--';
% Create l1_8
app.l1_8 = uilabel(app.UIFigure);
app.l1_8.VerticalAlignment = 'top';
app.l1_8.Position = [22 57 80 15];
app.l1_8.Text = 'First Cluster:';
% Create l1_9
app.l1_9 = uilabel(app.UIFigure);
app.l1_9.VerticalAlignment = 'top';
app.l1_9.Position = [8 36 92 15];
app.l1_9.Text = 'Second Cluster:';
% Create l1_10
app.l1_10 = uilabel(app.UIFigure);
app.l1_10.VerticalAlignment = 'top';
app.l1_10.Position = [170 57 80 15];
app.l1_10.Text = 'First Cluster:';
% Create l1_11
app.l1_11 = uilabel(app.UIFigure);
app.l1_11.VerticalAlignment = 'top';
app.l1_11.Position = [156 36 92 15];
app.l1_11.Text = 'Second Cluster:';
% Create l1_12
app.l1_12 = uilabel(app.UIFigure);
app.l1_12.VerticalAlignment = 'top';
app.l1_12.Position = [316 57 80 15];
app.l1_12.Text = 'First Cluster:';
% Create l1_13
app.l1_13 = uilabel(app.UIFigure);
app.l1_13.VerticalAlignment = 'top';
app.l1_13.Position = [302 36 92 15];
app.l1_13.Text = 'Second Cluster:';
% Create l1_14
app.l1_14 = uilabel(app.UIFigure);
app.l1_14.VerticalAlignment = 'top';
app.l1_14.Position = [452 57 80 15];
app.l1_14.Text = 'First Cluster:';
% Create l1_15
app.l1_15 = uilabel(app.UIFigure);
app.l1_15.VerticalAlignment = 'top';
app.l1_15.Position = [438 36 92 15];
app.l1_15.Text = 'Second Cluster:';
% Create l1_16
app.l1_16 = uilabel(app.UIFigure);
app.l1_16.VerticalAlignment = 'top';
app.l1_16.Position = [597 57 80 15];
app.l1_16.Text = 'First Cluster:';
% Create l1_17
app.l1_17 = uilabel(app.UIFigure);
app.l1_17.VerticalAlignment = 'top';
app.l1_17.Position = [583 36 92 15];
app.l1_17.Text = 'Second Cluster:';
```

```
% Create lbl11
app.lbl11 = uilabel(app.UIFigure);
app.lbl11.VerticalAlignment = 'top';
app.lbl11.Position = [95 57 50 15];
app.lbl11.Text = '--';
% Create lbl12
app.lbl12 = uilabel(app.UIFigure);
app.lbl12.VerticalAlignment = 'top';
app.lbl12.Position = [97 36 50 15];
app.lbl12.Text = '--';
% Create lbl21
app.lbl21 = uilabel(app.UIFigure);
app.lbl21.VerticalAlignment = 'top';
app.lbl21.Position = [243 57 50 15];
app.lbl21.Text = '--';
% Create lbl22
app.lbl22 = uilabel(app.UIFigure);
app.lbl22.VerticalAlignment = 'top';
app.lbl22.Position = [245 36 50 15];
app.lbl22.Text = '--';
% Create lbl31
app.lbl31 = uilabel(app.UIFigure);
app.lbl31.VerticalAlignment = 'top';
app.lbl31.Position = [387 57 50 15];
app.lbl31.Text = '--';
% Create lbl32
app.lbl32 = uilabel(app.UIFigure);
app.lbl32.VerticalAlignment = 'top';
app.lbl32.Position = [389 36 50 15];
app.lbl32.Text = '--';
% Create lbl41
app.lbl41 = uilabel(app.UIFigure);
app.lbl41.VerticalAlignment = 'top';
app.lbl41.Position = [523 57 50 15];
app.lbl41.Text = '--';
% Create lbl42
app.lbl42 = uilabel(app.UIFigure);
app.lbl42.VerticalAlignment = 'top';
app.lbl42.Position = [525 36 50 15];
app.lbl42.Text = '--';
% Create lbl51
app.lbl51 = uilabel(app.UIFigure);
app.lbl51.VerticalAlignment = 'top';
app.lbl51.Position = [667 57 50 15];
app.lbl51.Text = '--';
% Create lbl52
app.lbl52 = uilabel(app.UIFigure);
app.lbl52.VerticalAlignment = 'top';
app.lbl52.Position = [669 36 50 15];
app.lbl52.Text = '--';
% Create a1
```

```
            app.a1 = uiaxes(app.UIFigure);
            title(app.a1, 'KMeans')
            app.a1.Position = [12 248 133 118];
            % Create a2
            app.a2 = uiaxes(app.UIFigure);
            title(app.a2, 'CFS')
            app.a2.Position = [156 248 133 118];
            % Create a3
            app.a3 = uiaxes(app.UIFigure);
            title(app.a3, 'IFS')
            app.a3.Position = [302 248 133 118];
            % Create a4
            app.a4 = uiaxes(app.UIFigure);
            title(app.a4, 'ReliefF')
            app.a4.Position = [448 248 133 118];
            % Create a5
            app.a5 = uiaxes(app.UIFigure);
            title(app.a5, 'UDFS')
            app.a5.Position = [584 248 133 118];
            % Create DataSetsListBoxLabel
            app.DataSetsListBoxLabel =
uilabel(app.UIFigure);
            app.DataSetsListBoxLabel.HorizontalAlignment
= 'right';
            app.DataSetsListBoxLabel.VerticalAlignment =
'top';
            app.DataSetsListBoxLabel.Position = [27 441
58 15];
            app.DataSetsListBoxLabel.Text = 'Data Sets';
            % Create DataSetsListBox
            app.DataSetsListBox =
uilistbox(app.UIFigure);
            app.DataSetsListBox.Items = {'Fertility',
'Immunotherapy', 'Cryotherapy', 'Wholesale'};
            app.DataSetsListBox.ValueChangedFcn =
createCallbackFcn(app, @DataSetsListBoxValueChanged,
true);
            app.DataSetsListBox.Position = [100 384 144
74];
            app.DataSetsListBox.Value = 'Fertility';
        end
    end
    methods (Access = public)
        % Construct app
        function app = gui
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            % Execute the startup function
            runStartupFcn(app, @startupFcn)
```

```
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```