Tel-Aviv University

Raymond and Beverly Sackler Faculty of Exact Sciences

The Blavatnik School of Computer Science

A feature ranking algorithm for clustering medical data

Thesis submitted in partial fulfillment of graduate requirements for

The degree "Master of Sciences" in Tel-Aviv University

School of Computer Science

By

Eran Shpigelman

Under the supervision of

Prof. Ron Shamir

February 2024

Acknowledgement

I would like to express my sincere appreciation to the people who accompanied and supported me during this period and helped me accomplish this milestone.

Foremost, I wish to extend my deep thanks to my outstanding supervisor Prof. Ron Shamir. I am honored to have been advised by a world-renowned scientist like Ron, who walked me through my first steps in scientific research, with great patience and understanding on the one hand, and uncompromising professionalism on the other hand. I also thank Ron for pushing me to present my work in domestic and international conferences and to publish it in scientific journals.

Besides the scientific work, Ron's support, both on a personal and professional level, has made it possible for me to reach this milestone while being recruited for reserve duty. Ron served as an example for the responsibility the academia has for the greater good of the people of Israel, and I'm grateful for having such a role model.

Secondly, I want to thank my lab members that helped me and contributed to this work, especially to Nimrod Rappoport, Omer Noy and Hagai Levi whose advice inspired different parts of this research. I would also like to thank my collaborators in a previous research project from the Tel Aviv Sourasky Medical Center, especially Dr. Aviram Hochstadt and Pros. Yan Topilsky. Our joint work ultimately led to the research presented here.

I wish to thank all my lab mates – Dan C., Lianrong, David, Tom, Hagai, Hadar, Dan F., Naama, Ron Sa., Maya, Tal, Assaf, Tammy and Yahely, for being part of this journey through fascinating discussion and great friendship. I would also like to thank the amazing Gilit Zohar-Oren and Shlomit Hillel for their administrative help.

I would like to express my gratitude for the financial support I have received during my studies: the Edmond J. Safra Center for Bioinformatics at Tel Aviv University, Israel Science Foundation (Grant No. 3165/19), within the Israel Precision Medicine Partnership program, and a grant from the Tel Aviv University Center for AI and Data Science (TAD).

Lastly, I thank my family who inspired me to follow my interests and have always been supportive of my choices. My mother Mazal, my brothers Aviv and Baruch and my love and fiancé Ran.

Abstract

The availability of electronic medical records (EMR) data has advanced dramatically in recent years, and a growing number of studies apply algorithmic and machine learning methods to these data. A key method is clustering, used for a variety of purposes, including finding unknown subtypes of known diseases. The abundance and redundancy of information in EMR data raises the need to identify and rank the features that are most relevant for the clustering task. This is needed both for time and space efficiency reasons as well as for improving the interpretability of the results.

Here we propose FRIGATE (Feature Ranking In clustering using GAme ThEory), an algorithm that ranks features by their importance to clustering, incorporating the concepts of Shapley value and Multiplicative Weights. FRIGATE is an ensemble feature ranking algorithm, which derives the importance of features from multiple clustering solutions on sub-groups of features. For each clustering solution a small group of features is ranked in a Shapley-like framework, and multiplicative weights are applied to limit the randomness of their choice. It can handle both categorical and continuous features, a common need in medical data. FRIGATE outperforms previously suggested ensemble ranking algorithms on simulated and real medical data, both in solution quality and in speed.

Table of Contents

1. Introduction

In the past decade and a half, medical systems around the world underwent a major digitization revolution [1]. As a result, most of the personal medical information is now stored electronically, transforming the way medical research is conducted. Today, it is possible to conduct research on the population level, on a large number of patients and using multiple types of medical features. It is also possible to follow a group of patients over time and learn from the changes in their medical conditions. Although medical data sharing has been slow [2], the number of clinical data sets available to researchers is growing [3]. Such resources include data sets that span a large range of clinical data types, such as MIMIC [4], [5], and some even offer a combination of genomic and medical information, e.g., the UK BioBank [6].

Medical data have some special characteristics that make them challenging to work with. Firstly, some of them are of great magnitude. For example, in MIMIC-III alone there is information of 46,520 patients, with 753 different lab tests and 14567 different ICD-9 codes [4]. Another challenge that is closely related to the large number of features is the data incompleteness. Medical data typically have high percentage of missing values even for frequently taken measurements [7].

These challenges and the availability of the data create an opportunity for collaboration between clinicians and computer scientists in order to answer medical questions using computational tools. More specifically, we see a growing number of machine-learning applications on medical data [8] alongside development of computational tools dedicated to analyzing medical information [9], [10]. Some of the computational tools in use are from supervised machine-learning, where there are certain patient labels that the researchers want to learn (e.g., sick and healthy). A lot of the research in this direction is focused on building predictive models for early detection of diseases [11], while other creative uses include predicting the levels of infection in the population during the Covid-19 pandemic [12]. Another type of machine-learning models is unsupervised. In this case labels are not available, and the researchers are trying to find latent patient groups that are not yet known. These problems are usually referred to as clustering problems, and in the medical domain the clusters are used in discovery of new subgroups of known diseases [13]–[15] and in clinical image segmentation [16]. Our research is focused on this type of problems.

In the generic clustering problem the input is a set of m vectors in R^n. The vectors are called *samples*, and the coordinates are called *features*. The goal is to partition the samples into groups called *clusters*

so that the average intra-cluster distance between samples is as small as possible and the average inter-cluster distance is as large as possible [17]–[19]. These two objectives pull solutions in different directions, since increasing the number of clusters would decrease both the first and the second objective. In some versions of the problem the number of clusters is given and then the first objective suffices. Most exact formulations of clustering are NP-hard, and heuristics are often used. In the analysis of electronic medical records (EMR), a sample corresponds to a patient or an inpatient's record, where features can be demographic information (age, gender, previous diseases, etc.), results of measurements (temperature, blood pressure, etc.) or lab results (white blood cells count, cholesterol level, etc.). There are multiple clustering algorithms that use different properties of the data to create the clusters. We will focus on a family of algorithms that is usually referred to as *partitioning based* or *moving centers*, which include the widely used k-means algorithm. These algorithms usually specify the initial number of clusters and iteratively reallocate samples among clusters until convergence based on a pre-defined criterion [20].

One challenge in clustering problems is the choice of the number of clusters k. While some algorithms determine k [21], others require k as an input [22]. That means that the user needs to define in advance the number of latent groups that are to be found in the data. This is a hard task, as the properties of the data are often inherently unknown. Also, different values of k can potentially result in significantly different cluster structures, or equivalently, different possibilities of partitioning the data, with no known absolute truth. For example, in the medical domain we can have a large data set of patients that when using $k = 2$ the data will be partitioned in accordance with sex, but higher values of k can partition the data into subsets corresponding to different medical conditions, which without additional knowledge will be hard to label.

Another challenge is interpretability of the results, in particular in medical research. When finding new clusters in the data, we want to be able to understand the most important features that create them, in order to obtain clinical insights. When dealing with large data sets with possibly thousands of features, this is challenging. Also, running the algorithms on huge data sets is computationally expensive and not always feasible. For these reasons, feature selection algorithms were proposed [23]. Such algorithms seek the most important features for the clustering task. Our goal here is the development of such an algorithm that ranks all the features according to their importance to the clustering task, and is meant to specifically fit medical data.

Many medical databases, as well as genomic data and others, contain a large number of features. One way to deal with the high dimensional data is dimension reduction methods [24]. However, the effect of individual features, which is crucial for medical insights, will be obscured. Another option is to use feature selection algorithms, which choose a subset of features that will create a sub matrix with "good" clusters. There are several feature selection methods for clustering algorithms [23], [25]. Typically, they are divided into two main groups: filters and wrappers. The filter methods determine the importance of features based on internal properties of the data, without performing any clustering in the process. This makes them very efficient. The wrapper methods, on the other hand, use a specific clustering algorithm and iteratively produce a clustering solution using a subset of features that according to some evaluation criterion are the most relevant for the clustering task. They tend to be more accurate than the filter methods [23]. There are also some hybrid methods that combine the filter and the wrapper ideas [23].

In recent years ensemble feature ranking algorithms were also suggested. These methods create an ensemble of clustering solutions on subsets of features and then use some metric to evaluate the contribution of each feature [26]. These methods were shown to perform better than filter and wrapper methods, including on medical data sets [26]–[28]. Ensemble methods can also be used for choosing a subset of important features, in contrast to ranking the full set of features [29]. However, here, as we aim to work with medical information, we prefer to lose as little information as possible and prefer to evaluate and rank the full set of features. For these reasons we chose to develop a new algorithm within the ensemble ranking framework.

There are three ensemble ranking algorithms that we refer to in this paper: FRMV [27] is a relatively flexible framework that iteratively ranks the features according to a pre-defined relevance measure (e.g., linear correlation). FRCM [28], designed for genomic data, uses a consensus clustering solution before evaluating the features. FRSD [26] uses silhouette score [30], a known criterion of clusters quality, to rank the features. All of them will be described in detail below.

Here we introduce a new algorithm called FRIGATE (feature ranking in clustering using game theory), which uses two concepts from game theory. The first is motivated by Shapley value, a measure of the contribution of every player to the group in a cooperative game [31], [32]. In our case the players are the features and the "game" is clustering. Shapley values are widely used for feature evaluation in classification models [32] and so far were not used in clustering for feature selection or ranking. The second is Multiplicative Weights (MW) [33], a framework to improve the selection of players in an

iterative process such that players are selected from a distribution based on their performance so far. In FRIGATE we use MW to guide the choice of features for each clustering solution and avoid the choice of features that proved to be insignificant. All previously presented ensemble algorithms choose subsets of features at random. To the best of our knowledge this is the first time that MW is adapted to feature selection for clustering.

The thesis is organized as follows: We first present relevant background including clustering methods, extant algorithms of ensemble feature ranking for clustering and the game theory concepts that are incorporated in FRIGATE. Next, in the Methods section, we present in detail the FRIGATE algorithm and describe the construction of simulated data. In the Results section we demonstrate a run of FRIGATE, show the process of choosing the model hyper-parameters and measure the performance of FRIGATE and the extant algorithms, both on simulated and on real data. We conclude with a discussion of the results.

2. Background

In this chapter we describe algorithms and computational methods that will be used in the thesis.

2.1 k-means

k-means [22] is one of the most widely used clustering algorithms. It receives as an input the number k of clusters and the data matrix, and assumes features are continuous. The algorithm first chooses cluster centers (centroids) and assigns each sample to the cluster with the closest centroid using a distance metric. Then, the center of mass of each cluster is set as the new centroid, and the process is repeated until convergence. The initial centroids are either chosen at random or with k-means++ [34], which chooses samples from the cohort based on a probabilistic model of their contribution to the intra cluster distance and meant to speed up the convergence. The algorithm is heuristic, and returns a local optimum only. It is usually run several times and the solution with the smallest overall distance to centroids is chosen. Here we used the Euclidean distance metric:

$$d(x, y) = \sum_{i=1}^{n}(x_i - y_i)^2 \qquad (1)$$

and k-means as implemented in Scikit learn [35] with 100 k-means++ initializations in each run.

2.2 k-modes

k-modes [36] is a clustering algorithm for categorical data, namely, feature values are discrete (two or more) categories. Other than that, the algorithm is identical to k-means. The initial centroids are chosen with Cao method [37] which is similar to k-means++. The Hamming distance is used as the distance metric:

$$d(x,y) = \sum_{i=1}^{n} \delta(x_i, y_i) \tag{2}$$

$$where \quad \delta(x_i, y_i) = \begin{cases} 1 & if\ x_i = y_i \\ 0 & else \end{cases} \tag{3}$$

Here we used the k-modes implementation in [38].

2.3 k-prototypes

k-prototypes [39] is an algorithm that clusters mixed data, i.e. data with both continuous and categorical features. This is a common case in many applications, including the medical domain. For example, patient medical records typically contain continuous features, such as weight, height, age and lab tests like blood count, as well as some categorical features like sex, diagnosis, and past diseases. The iterative process of k-prototypes is identical to k-means, with an adjusted distance function (see below), and the initial centroids choice is based on the categorical features with Cao method [37] similarly to k-modes. k-prototypes was reported as one of the best performers in a recent benchmark of mixed-data clustering algorithms [40]. The distance metric that we used is:

$$d(x,y) = \sum_{i=1}^{p} (x_i - y_i)^2 + \gamma \sum_{i=p+1}^{m} \delta(x_i, y_i) \tag{4}$$

Where $x_1, \dots x_p$ are numerical variables, $x_{p+1}, \dots x_m$ are categorical variables, and δ is the Hamming distance function. The γ factor determines the relative contribution of the categorical features in comparison to the continuous features, where $\gamma = 1$ means equal contribution, $\gamma > 1$ gives higher value to the categorical features and $\gamma < 1$ gives higher value to the continuous features. Here we used a k-prototype implementation in [38].

2.4 The silhouette score

The silhouette score [30] is a common intrinsic measure for the quality of a clustering solution, often used when the real labels are unknown. The score is calculated separately for each sample and

measures how well a sample belongs to the cluster it is assigned to, versus the next nearest cluster. For some sample i, define:

$$s(i) = \frac{b(i)-a(i)}{\max(a(i),b(i))} \quad (5)$$

Where $a(i)$ is the average distance of i to all other samples in its cluster, and $b(i)$ is the minimal average distance between i to samples of another cluster. The total silhouette score of the clustering solution is $\sum_i s(i)$, summed over all samples. Here we used silhouette as implemented in Scikit learn [35].

2.5 The Elbow method

The Elbow method seeks to determine the optimal number of clusters in a dataset. It works by calculating the differences in some function v for the quality of clusters as the number of clusters changes. For a solution with i clusters, let $v(i)$ be the sum of total within-cluster sum of squares. As increasing the number of clusters decreases the value of $v(i)$, the optimal i is chosen as the value at which a major change is observed, and for a larger i the changes are minor. Visually, this is an "elbow" point in the graph of $v(i)$ vs. i. As suggested in [41] the value chosen is the one that brings to a maximum the function:

$$v[i+1] + v[i-1] - 2v[i] \quad (6)$$

2.6 Adjusted Rand Index

The Adjusted Rand Index (ARI) measures the similarity between two clustering solutions [42]. It is used, for example, to test for the similarity between a clustering solution and the true labels. ARI is based on Rand Index (RI) [43]. Given a clustering solution I and true labels L we define four relation types between a pair of samples:

1. The two samples are in the same cluster in I and in L.
2. The two samples are in the different clusters in I and in L.
3. The two samples are in the same cluster in I but in different clusters in L.
4. The two samples are in different cluster in I but in the same clusters in L.

We define a as the number of pairs of type 1, b the number of pairs of type 2, c the number of pairs of type 3, d the number of pairs of type 4. Then RI is:

$$RI = \frac{a+b}{a+b+c+d} \quad (7)$$

RI scores range between 0 and 1, where $RI = 1$ means a perfect match between I and L.

RI gets closer to 1 when the number of clusters increases [44], and does not consider the cluster sizes. For example, if 90% of samples have the same label in L, and in I all the samples are in the same cluster, then the RI score will be 0.9, a seemingly very high score, although the clustering solution does not reflect the true labels. ARI tries to overcome these problems by adjusting RI for chance grouping of samples, by calculating the expected number of pairs appearing in the same cluster in L and I given the cluster sizes, under the hypergeometric assumption:

$$E\left(\Sigma_{i,l}\binom{n_{il}}{2}\right) = \Sigma_i\binom{n_{i\cdot}}{2}\Sigma_l\binom{n_{\cdot l}}{2}/\binom{n}{2} \quad (8)$$

where n_{il} represents the number of samples that are in cluster i in I and in cluster l in L, with a total of n samples. The $\cdot$ stands for a row/column sum, which accounts for a cluster size. For example, $n_{i\cdot}$ is the size of cluster i in I. Similarly, the expected values can be calculated for every entry in Table 1 (details are found in [42]).

Cluster in L / Cluster in I	L_1	L_2	…	L_x	Sum
I_1	n_{11}	n_{12}	…	n_{1x}	$n_{1\cdot}$
I_2	n_{21}	n_{22}	…	n_{2x}	$n_{2\cdot}$
. . .	. . .	. . .		. . .	. . .
I_y	n_{y1}	n_{y2}	…	n_{yx}	$n_{y\cdot}$
Sum	$n_{\cdot 1}$	$n_{\cdot 2}$	…	$n_{\cdot x}$	n

Table 1. Illustration of the ARI setting. A value n_{il} represents the numbers of samples that are in the i^{th} cluster in I and the l^{th} cluster in L. The total number of samples in n. The ARI adjusts the RI for chance grouping of the samples given the sums $n_{1\cdot}, \ldots, n_{y\cdot}$ and $n_{\cdot 1}, \ldots, n_{\cdot x}$.

Using a general formulation of:

$$\frac{Index - Expected\_Index}{Maximum\_Index - Expected\_Index} \quad (9)$$

ARI is defined to be:

$$ARI = \frac{\sum_{i,l}\binom{n_{il}}{2} - \frac{\left[\sum_i \binom{n_{i\cdot}}{2} \sum_l \binom{n_{\cdot l}}{2}\right]}{\binom{n}{2}}}{\frac{1}{2}\left[\sum_i \binom{n_{i\cdot}}{2} + \sum_l \binom{n_{\cdot l}}{2}\right] - \frac{\left[\sum_i \binom{n_{i\cdot}}{2} \sum_l \binom{n_{\cdot l}}{2}\right]}{\binom{n}{2}}} \quad (10)$$

The maximum ARI is 1, which means a perfect match, but unlike RI, ARI can also take negative values with the bound of -1, where 0 means random clusters ("expected"), and negative values means less accurate than random assignment. Here we used ARI as implemented in Scikit learn [35] with a lower bound of -0.5.

2.7 FRMV

Feature ranking from multiple views (FRMV) [27] is an ensemble feature ranking algorithm that ranks the features according to their relevance to a clustering solution. The algorithm is presented formally below (Algorithm 1). It performs T iteration, where in each iteration a subset of q features is selected with replacement, a clustering solution is obtained for it, a relevance score is computed for each feature, and the participating features are ranked according to their score. The final feature ranking is done according to the average rank. It was not specified in [27] how to rank repeating features and no implementation was provided. The relevance measure is a function $g: \mathbb{R}^m \mathrm{x} \mathbb{N}^m \to \mathbb{R}$ where m is the number of samples in the cohort. It receives a vector d_r of values of some feature d and a vector of cluster membership I, corresponding to a clustering solution, and returns a numeric score that reflects how well feature v fits solution I. The framework allows the use of any clustering algorithm and any relevance measure. In [27], Hong et al. used k-means as the clustering algorithm and tested two relevance measures: the linear correlation coefficient and symmetrical uncertainty. As the authors reported that the results of the two measures were comparable, we used the linear correlation coefficient:

$$\rho(f_r, I) = \frac{cov(d_r, I)}{\sigma(d_r)\sigma(I)} \quad (11)$$

The fraction of features f that were chosen per iteration was $\frac{1}{2}$ and the number of iterations performed was $T = 100$.

Algorithm 1: FRMV – with k-means

Input: $\boldsymbol{A}$ – mxn matrix of m samples and n features; $\boldsymbol{g}$ - relevance measure; $\boldsymbol{k}$ - number of clusters; $\boldsymbol{T}$ - number of iterations; $\boldsymbol{f}$ - fraction of the features to use in each iteration

Output: $\boldsymbol{R}$ – list of the features ordered by importance for clustering

for $t \leftarrow 1$ to T
  $S \leftarrow$ a set of $q = \lceil f \cdot n \rceil$ features randomly chosen with replacement
  $A^{(t)} \leftarrow$ a matrix derived from A of size mxq with columns corresponding to S
  Perform k-means on $A^{(t)}$
  $I^{(t)} \leftarrow$ labels of the clustering solution
  For each feature in S compute its relevance to $I^{(t)}$ according to relevance measure g
  $r^{(t)} \leftarrow$ ranks of features in S according to their relevance
end
$R \leftarrow$ array of length n with the average rank of each feature
Return an order of the features sorted in decreasing order according to R

Some parts of the algorithm are not clear. Sampling features with replacement raises two major issues: first, features that are chosen several times in an iteration gain excessive influence on the clustering solution. This can affect the clustering solutions and may potentially create a large variation in the final ranking of the results (see Results). This issue may be fixed by a large number of iterations, such that the excessive influence will average over all features, but there is no clear suggestion how to pick the number of iterations. Another issue is the calculation of the linear correlation coefficient. I in Equation 9 represents a categorical vector of cluster membership, i.e., the clusters are numbered $1, \ldots, k$ and $I_j = l$ if sample j belongs to cluster l. Therefore, calculating its standard deviation and covariate with a continuous vector depends on the numbering of the clusters. No implementation of FRMV was provided in [27]. We implemented it as described in [27] and tested by other publications [26], [28]. In our implantation of it we calculated the correlation using the arbitrary numbering of the clusters, which affects the results. Also, in each iteration we ran the clustering algorithm on the full subset of features, including repeating features. However, the ranking was performed on set of unique features with no repetition, and therefore the number of ranked features and the highest rank might vary among iterations.

2.8 FRCM

Feature ranking based on the consensus matrix (FRCM) [28] is an ensemble feature ranking algorithm that was originally designed for genomic data. It is described in Algorithm 2. Unlike FRMV, in FRCM the production of clustering solution and the evaluation of features are separated. The algorithm performs T iterations, where in each a subset of q features is selected and a clustering solution is obtained for them, using k-means. The number of clusters k is selected in each iteration uniformly at random from the range $\{2, K_{\max}\}$ where $K_{max} = \min\left(\left\lceil\sqrt{m}\right\rceil, K_v\right)$, where m is the number of samples and K_v is provided as input. For each solution an mXm co-association matrix $M^{(t)}$ is computed, where $M_{i,j} = 1$ if the two samples are in the same cluster and 0 otherwise. In their paper [28] Zhang et al. used $T = 200$, $q = \left\lceil\frac{n}{2}\right\rceil$ and $K_v = 20$.

After T co-association matrices $M^{(1)}, \ldots, M^{(T)}$ were generated, a *consensus matrix* M is created as an average of all the co-association matrices:

$$M = \frac{1}{T}\sum_t M^{(t)} \tag{12}$$

In addition, for each feature d, an *affinity matrix* is calculated as:

$$A_{i,j}^{(d)} = \sqrt{\left(1 - \frac{\left(x_{i,d} - x_{j,d}\right)^2}{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2}\right)} \tag{13}$$

Where x_i and x_j are vectors of samples. $A_{i,j}^{(d)}$ measures how influential feature d is on the distance between two samples: the larger the influence, the smaller $A_{i,j}^{(d)}$ is. So, it can be viewed as the distance between the two samples in terms of the feature d.

Lastly, the score of each feature is calculated as:

$$z^d = ARI_{mm}\left(M, A^{(d)}\right) = \frac{(s_0 - s_3)}{0.5(s_1 + s_2) - s_3} \tag{14}$$

Where $s_0 = \sum_{i,j,i\neq j}\frac{M_{ij}A_{ij}}{2}$, $s_1 = \sum_{i,j,i\neq j}\frac{M_{ij}}{2}$, $s_2 = \sum_{i,j,i\neq j}\frac{A_{ij}}{2}$, $s_3 = \sum_{i,j,i\neq j}\frac{2s_1 s_2 M_{ij}}{m(m-1)}$

ARI_{mm} is a measure for real-valued matrix similarity inspired by the Adjusted Rand Index, which measures similarity between clustering solutions. ARI_{mm} has similar properties to ARI (described above) with higher values indicating higher similarity. Hence, for two samples i, j in the same cluster, if feature

d is important we expect $A_{i,j}^{(d)}$ to be high, since the relative difference between $x_{i,d}$ and $x_{j,d}$ is expected to be small. Finally, features are ranked according to z. No implementation of FRCM was provided in [28]. We implemented it as described in [28] and tested by another publication [26].

Algorithm 2: FRCM

Input: $\boldsymbol{A}$ - mxn matrix of m samples and n features; $\boldsymbol{K_v}$ - maximum number of clusters;
$\boldsymbol{T}$ - number of iterations; q – number of features selected per iteration
Output: $\boldsymbol{R}$ – list of the n features ordered by importance for clustering

$K_{max} \leftarrow \min(\sqrt{m}, K_v)$
for $t \leftarrow$1 to T
  $S \leftarrow$ a set of randomly chosen q features
  $A^{(t)} \leftarrow$ the sub matrix of size mxq of A induced by S
  $k_t \leftarrow$ number of clusters randomly chosen from $\{2,\ldots, K_{max}\}$
  Perform k-means on $A^{(t)}$
  $I^{(t)} \leftarrow$labels of the clustering solution
  $M^{(t)} \leftarrow$co-association matrix corresponding to the clustering solution
end
$M \leftarrow \frac{1}{T}\sum_t M^{(t)}$
$z \leftarrow n$ long list of scores, initialized to 0
For each feature d
  $A^{(d)} \leftarrow$ affinity matrix of size mxm. See Equation 13.
  $z^{(d)} \leftarrow ARI_{mm}(M, A^{(d)})$. See Equation 14.
end
Return an order of the features sorted in decreasing order of $z^{(d)}$.

2.9 FRSD

The FRSD algorithm (Feature ranking based on silhouette decomposition) [26], has a similar flow to FRMV and FRCM. In each iteration the algorithm randomly chooses a subset of the features, produces a clustering solution using k-means and ranks the selected features based on a specific criterion. k-means is run for all possible values of k between 2 and $K_{max} = \min(\lceil\sqrt{m}\rceil, K_v)$. FRSD performs T iterations for each possible k. Hence, the overall number of k-means runs is $(K_{max} - 1) \cdot T$. In [26] Yu et al. used $T = 200$, subsampled $f = 0.06$ of the features for each clustering solution and set $K_v = 20$.

As the name suggests, FRSD uses silhouette score to rank the features. Let S be the silhouette score for a clustering solution C. Let $q = f \cdot n$ be the number of features in the solutions. Given a participating

feature j its values are shuffled, and the silhouette of the solution C is recomputed and denoted by S_j. Let $\delta_j = S - S_j$. The higher δ_j is, the more important feature j was to the solution. Let r_j be the ranking of the features, $r_j \in \{1,2,\dots,q\}$ according to δ_j in ascending order, where the feature j with highest value of δ_j is ranked q, and the lowest ranked 1. The final score of feature j is:

$$s_j = \frac{r_j}{q} w \tag{15}$$

where $w = \frac{1+S}{2}$ is a weight meant to increase the importance of solutions with high value of S. That way clustering solutions with low silhouette score, obtained when many non-informative features are selected, will have a weaker effect on the final result. The final rank is based on the average score of each feature over all iterations for all values of k. The complete procedure is shown in Algorithm 3, which returns an array of features in decreasing order of importance.

Algorithm 3: FRSD

Input: $\boldsymbol{A}$ - mxn matrix of m samples and n features; $\boldsymbol{K_v}$– maximum number of clusters;
$\boldsymbol{T}$ – number of iterations**;** $\boldsymbol{f}$ – fraction of n to use in each iteration
Output: $\boldsymbol{R}$ **–** list of the n features ranked in decreasing order of importance for clustering

$scores \leftarrow$ array of length n for keeping score of each feature, initialized to 0s
$K_{max} \leftarrow \min(\sqrt{m}, K_v)$
for k in $\{2,\dots,K_{max}\}$
    $scores^{(k)} \leftarrow$ array of length n for keeping score of each feature, initialized to 0s
    $counts^{(k)} \leftarrow$ array of length n for counting the times each feature is selected, initialized to 0s
    for $t \leftarrow$1 to T
        $h \leftarrow$ a set of $q = \lceil f \cdot n \rceil$ randomly chosen features
        $A^{(t)} \leftarrow$ the sub matrix of size mxq of A with columns corresponding to h
        Perform k-means on $A^{(t)}$
        $I^{(t)} \leftarrow$labels of the k-means solution
        $S^{(t)} \leftarrow$ silhouette score of the solution on $A^{(t)}$ and $I^{(t)}$
        $w^{(t)} \leftarrow \frac{1+S^{(t)}}{2}$
        $\delta \leftarrow$ list of length q initialized to 0s
        for v in h
            $counts[v] \mathrel{+}= 1$
            $\hat{v} \leftarrow$ Shuffled version of v
            $A_v^{(t)} \leftarrow$ a matrix identical to $A^{(t)}$ with $\hat{v}$ instead of v
            $S_v^{(t)} \leftarrow$ the silhouette score for $A_v^{(t)}$ and $I^{(t)}$
            $\delta[v] \leftarrow S^{(t)} - S_v^{(t)}$
        End

$r \leftarrow$ a list of the ranks of h according to δ, where higher δ accounts for higher r.

for v in h: $scores^{(k)}[v] += \frac{r[v]}{q} w^{(t)}$

end

$scores += scores^{(k)} / counts^{(k)}$

end

$scores \leftarrow \frac{scores}{K_{max}-1}$

Return an order of the features sorted in decreasing order of $scores$.

No implementation of FRSD was provided in [26]. We implemented it as described in [26] and used silhouette as implemented in Scikit learn [35].

2.10 Shapley values

In the theory of cooperative games in game theory, there is a set N of players who can form coalitions. Each coalition $S \subset N$ has a value $g(S)$. According to the Shapley theory [31] the contribution of player i to group $S \cup \{i\}$ is defined as:

$$g(S \cup \{i\}) - g(S) \tag{16}$$

and the Shapley value of player i is a weighted average of its contributions over all possible Ss, i.e.:

$$\sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n-|S|-1)!}{n!} [g(S \cup \{i\}) - g(S)] \tag{17}$$

It is widely used in supervised learning to measure the contribution of a feature to a prediction model, where for efficiency reasons it is usually evaluated using random permutations instead of enumerating all possible groups S [32]. To the best of our knowledge, Shapley values were not used to date in feature selection for clustering.

2.11 Multiplicative Weights

Multiplicative Weights (MW) is an algorithmic update method used in game theory and algorithm design. The motivation of MW [33] is to improve the decisions we make along the iterations by gradually favoring decisions that were proven to be right so far. In our case the decisions are the features selected and we use the Hedge update rule that was suggested by Arora et al. [33]:

$$w_i^{(t+1)} = w_i^{(t)} \cdot exp(-\eta m_i^t) \tag{18}$$

Where w_i^t is the weight of feature i at the t-th iteration, $\eta \leq 1$ is a constant parameter and m_i^t is the cost of feature i at iteration t. m_i^t is a value in the range $[-1,1]$ that reflects how good decision i was in iteration t, where higher positive values correspond to worse decisions and negative values correspond to good decisions that warrant an award instead of a cost. A common practice, also used in our implementation, is to use non-negative values only. The choice of values for m_i^t depends on the implementation.

3. Methods

3.1 The FRIGATE algorithm

FRIGATE is a new ensemble feature ranking algorithm, which uses the Shapley value concept to find the most valuable features for clustering based on multiple runs of k-means (or k-prototypes/k-modes) algorithm.

To use the Shapley values in our context, the players are the features. We assume the number of clusters k is given, and use the total distance of samples to their cluster centroids as the objective function g:

$$g(S) = \sum_{j=1}^{k} \sum_{x \epsilon C_j} d^S\left(x, y_j\right) \quad (19)$$

Here S is a set of features, k is the number of clusters, C_j is the set of samples included in cluster j and y_j is the centroid of cluster j. d^S is the distance function on the sample vectors restricted to the coordinates in S. We call $g(S)$ the *solution score*.

d^S and the clustering algorithm that we use will depend on the data types in S. If all the features are continuous then we use k-means for clustering and the Euclidean distance. When we have a mixture of categorical and continuous features, we will use k-prototype for clustering, and the corresponding distance function (Equation 4). If we have only categorical features, k-modes is used for clustering with d as the Hamming distance. We used k-means as implemented in Scikit learn [35] with 100 k-means++ initializations in each run.

Algorithm 4 presents the procedure for continuous features. In iteration t, the algorithm selects at random a subset of features and performs k-means on the corresponding submatrix $A^{(t)}$. Once a solution has been obtained, we calculate the contribution of feature i as the difference between that

solution's score and the score obtained by the same clustering on the submatrix $A^{(t)}$ in which the values of feature i were randomly shuffled among the samples, keeping the rest unchanged. For the final ranking we use the average scores of the features.

Algorithm 4: FRIGATE

Input: $\boldsymbol{A}$ - mxn matrix of m samples and n features; $\boldsymbol{k}$ - number of clusters; $\boldsymbol{T}$ - number of iterations; $\boldsymbol{f}$ - fraction of features to use in each iteration

Output: $\boldsymbol{R}$ **–** list of the m features ordered by importance for clustering

1 $scores \leftarrow$ array of length n for keeping score of each feature, initialized to 0s
2 $counts \leftarrow$ array of length n for counting the times each feature is selected, initialized to 0s
3 **for** $t \leftarrow 1$ to T
4 $h \leftarrow$ a set of $q = \lceil f \cdot n \rceil$ randomly chosen features
5 $A^{(t)} \leftarrow$ a matrix of size mxq with columns corresponding to h
6 Perform k-means on $A^{(t)}$
7 $I \leftarrow$ labels of the clustering solution
8 $g(h) \leftarrow$solution score of $A^{(t)}$ and I
9 **for** v in h
10 $\hat{v} \leftarrow$ Shuffled version of v
11 $A_v^{(t)} \leftarrow$ a matrix identical to $A^{(t)}$ except having $\hat{v}$ instead of v
12 $g_v \leftarrow$ solution score of $A_v^{(t)}$ and I
13 $scores[v] \leftarrow scores[v] + (g_v - g(h))$
14 $counts[v] \leftarrow counts[v] + 1$
15 **end**
16 **end**
17 $scores \leftarrow scores/counts$
18 return the features sorted in decreasing order of $scores$

Note that FRSD can also be seen as a type of a Shapley-like algorithm with a function g that uses the silhouette. However, a main difference is that FRIGATE does not rank the features on every iteration and accumulates the ranks for the final score, as in FRSD and FRMV, but instead summarizes the raw scores. That way poor clustering solutions that are based on non-informative features will have large $g(h)$ values (line 8 in Algorithm 4) as well as large g_v values (line 12). This will limit the ability of these features to receive high scores, as they are calculated by subtracting the distance after shuffling the values of a feature from the original distance (line 13 in Algorithm 4). Thanks to these properties of $g(h)$ and g_v, we do not need to use an additional factor, as FRSD does with silhouette, to assess the quality of the clusters. It also reduces the number of calculations and improves the efficiency of the algorithm.

We now discuss runtime complexity, referring only to k-means for simplicity. The runtime of k-means is $O(m \cdot q \cdot k \cdot c)$ for m samples, q features, k clusters and up to c iterations. We sample in each FRIGATE iteration $q = f \cdot n$ features. For each k-means run we perform i initializations. Therefore, the runtime of the k-means executions in each iteration of FRIGATE is $O(mqkci)$. Other than k-means runs, in each iteration we shuffle the values of q features over the full cohort in $O(m)$ for each feature and recalculate the solution score d_v in $O(m)$. The overall runtime of an iteration is $O(mqkci + mq) = O(mqkci)$. Hence, the additional actions to test the contribution of each feature do not increase the asymptotic runtime. We perform T iterations, so the overall runtime is $O(mTqkci)$. As $q = f \cdot n$ with constant f we can write the runtime as: $O(mTnkci)$.

3.2 The FRIGATE-MW algorithm

MW offers a smarter way to choose the features in FRIGATE for each clustering solution instead of choosing them randomly. Algorithm 5 shows the version of FRIGATE that uses MW for continuous features, which we call FRIGATE-MW.

We define an n-long array L so that $L(i) = \frac{i-1}{n-1}$. At each iteration we rank the features by their scores so far and use the ranks and L to determine $m_i^{(t)}$ (see chapter 2.11). If the rank of feature i at iteration t is r then $m_i^{(t)} = L[r]$. The weights of features that were not selected in the iteration remain unchanged. For the next iteration we select features from distribution $\boldsymbol{p^{(t)}} = \{w_1^{(t)}/\Phi^{(t)}, \dots, w_N^{(t)}/\Phi^{(t)}\}$ where $\Phi^{(t)} = \sum_i w_i^{(t)}$ is the sum of weights at the t-th iteration. To the best of our knowledge, this is the first use of MW in feature selection for clustering.

Algorithm 5: FRIGATE-MW

Input: $\boldsymbol{A}$ - mxn matrix of m samples and n features; $\boldsymbol{k}$ - number of clusters**;** $\boldsymbol{T}$ - number of iterations; $\boldsymbol{f}$ - fraction of features to use in each iteration; $\boldsymbol{\eta}$ – a Multiplicative Weights parameter

Output: $\boldsymbol{R}$ **–** list of the m features ordered by importance for clustering

$scores \leftarrow$ array of length n for keeping score of each feature, initialized to 0s

$counts \leftarrow$ array of length n for counting the times each feature is selected, initialized to 0s

$weights \leftarrow$ array of length n for keeping the weight of each feature, initialized to 1s

$L \leftarrow$ a static array of length n for the costs used in Multiplicative Weights. $L = [0, \frac{1}{n-1}, \frac{2}{n-1} \dots, \frac{(n-2)}{n-1}, 1]$

for $t \leftarrow 1$ to T

  $P \leftarrow weights/sum(weights)$

  $h \leftarrow$ a set of $q = \lceil f \cdot n \rceil$ features chosen from the distribution P

$A^{(t)}$ ← a sub matrix of size mxq of A with columns corresponding to h
Perform k-means on $A^{(t)}$
I ← labels of the clustering solution
$g(h)$←the solution score of $A^{(t)}$ and I
for v in h
    $\hat{v}$ ← Shuffled version of v
    $A_v^{(t)}$ ← a matrix identical to $A^{(t)}$ except having $\hat{v}$ instead of v
    g_v←the solution score of $A_v^{(t)}$ and I
    $scores[v] \leftarrow scores[v] + (g_v - g(h))$
    $counts[v] \leftarrow counts[v] + 1$
end
$ranks \leftarrow sort(scores/\ counts)$ // rank the features based on the scores so far
for v in h
    r ← rank of v in $ranks$
    $weights[v] = weights[v] \cdot \exp(-\eta \cdot L[r])$ // update the weight of v according to eq. 14
end
$scores \leftarrow scores/counts$
return the features sorted in decreasing order of $scores$

In each iteration we update the weights of the q participating features in constant time for each feature and sort the array of weights in $O(n \cdot \log(n))$. The overhead of MW for each iteration is thus $O(n \cdot \log(n) + q) = O(n \cdot \log(n))$, since $q < n$. The total runtime of each iteration in FRIGATE-MW is: $O(mqkci + n \cdot \log(n))$. Therefore, the total runtime is: $O(mTqkci + Tn \cdot \log(n)) = O\big(Tn(mkci + \log(n))\big)$. Altogether, the increase in the runtime over FRIGATE is not major. However, note that in FRIGATE-MW the iterations cannot be programmed to run in parallel, in contrast to FRIGATE.

For both variations of the algorithm we used $T = 2n$ and $f = 0.1$, and for FRIGATE-MW we used $\eta = 0.5$. For a detailed description of the parameter choice see Supplementary 1.

3.3 Simulation

We performed simulations in order to test the algorithms in situations where the true clustering and the informative features are known. The simulations were along the same lines of those described in [26]. The parameters of the simulation are:

- k – number of clusters

- c – number of samples in each cluster
- α – number of informative features
- β – number of non-informative features
- μ – distribution parameter
- σ – correlation coefficient between features

Simulating continuous data: For each cluster j, we construct c vectors of length $n = \alpha + \beta$ from multivariate normal distribution, where α features are sampled from a normal distribution with mean of $j \cdot \mu$ for $j \epsilon [0, \dots, k-1]$. The other β features are sampled from a normal distribution with mean 0 for all clusters and therefore represent the non-informative features. Thus, the mean vector of a sample in the j^{th} cluster is: $\mu_j = [(j \cdot \mu)_{\alpha \text{x} 1}, 0_{\beta \text{x} 1}]$.

Next, we define a covariance matrix, parameterized by σ, used to create correlations between the different features. The covariance matrix Σ is identical for all clusters:

$$\Sigma = (1-\sigma) \cdot I_{n \text{x} n} + \sigma \cdot 1_{n \text{x} 1} \cdot 1_{n \text{x} 1}^T \quad (20)$$

The nx$(k \cdot c)$ data matrix A then undergoes z-score normalization for each feature. This step is needed when working with many data types, especially in the medical domain as the values of different features can be of different magnitude.

Simulating mixed data: To build a simulation of mixed data we add three more parameters:

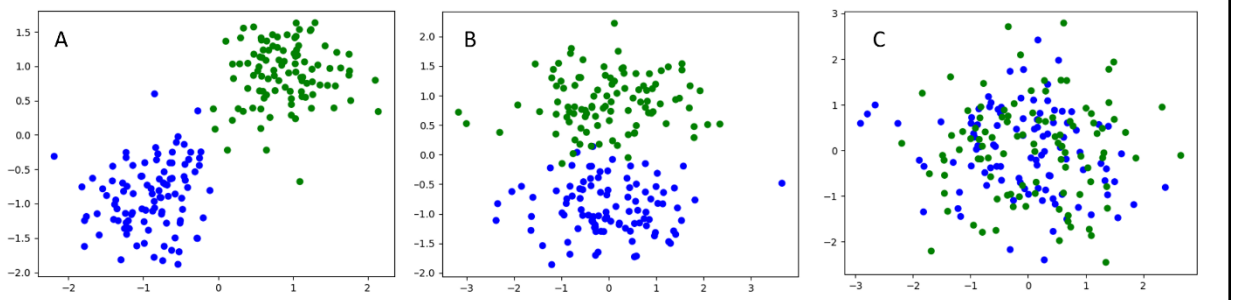
- $\alpha_{categorical}$ – number of informative categorical features
- $\beta_{categorical}$ – number of non-informative categorical features
- p – probability of choosing the right category

We assume that the categorical features have k categories, labeled $\{0,1, \dots, k-1\}$. For the informative features of a sample in the j^{th} cluster, we choose the value j with probability p and a value from $\{0, \dots, k-1\} \backslash \{j\}$ with probability $1-p$ where the value is chosen uniformly at random. For the non-informative features we choose a random value uniformly from $\{0, \dots, k-1\}$. The simulation of the continuous features is done as described before, and we concatenate the two matrixes into a single input matrix. In our simulation we used $p = 0.95$.

3.4 Demonstration of FRIGATE

For better understanding of the FRIGATE process, we demonstrate it graphically. We simulated data as described in section 3.3, with two continuous features, two clusters ($k = 2$), and 100 samples in each cluster, and simulation parameters $\mu = 4$, $\sigma = 0$. Figure 1 shows the data, where each axis is a feature and the samples are colored by cluster membership. We simulated three scenarios:

A. Both features are informative for the clustering solution (Figure 1A).
B. Only one feature is informative (Figure 1B)
C. Both features are not informative (Figure 1C).



D

Scenario	Solution Score	Scores after shuffling values of F1		Scores after shuffling values of F2	
		Solution	Feature	Solution	Feature
A	81.76	395.24	313.48	368.09	286.33
B	237.28	237.79	0.51	541.33	304.05
C	256.48	374.07	114.59	407.04	150.56

Figure 1. Illustrations of simulations with two clusters. In the simulation, there are 100 samples in each cluster, $\mu = 4$, $\sigma = 0$, and two features. A-C: features are represented by the axes. Each color represents a different cluster. A: both features are informative for clustering, B: only the feature represented by the y axis is informative, C: both features are not informative. D: Demonstration of FRIGATE iteration on the data of A-C. Solution score refers to line 8 in Algorithm 4, feature's score refers to line 13 in Algorithm 4. F1 is represented by the x-axis in Figure1 A-C and F2 is represented by the y-axis. We can see that the informative features received higher scores than the non-informative ones.

Next, we performed an iteration of the FRIGATE algorithm, using the centroids obtained from the clustering solution on the two features, to show the differences in scores in each scenario (Figure 1D):

A. When the two features were informative, the solution score (line 8 in Algorithm 4) was 81.76, and the scores of the features (line 13 in Algorithm 4) were 313.48 and 286.33. Both feature scores are high, and the difference can result from the randomness in shuffling the values (line

10 in Algorithm 4) or from the simulation that might have produced one feature that is more informative than the other.

B. When only one feature was informative, the solution score was 237.28, and the feature scores were 0.51 for the non-informative features and 304.05 for the informative feature.

C. When the two features were non-informative, the solution score was 256.48, and the feature scores were 117.59 and 150.57.

In Figure 2 we demonstrate graphically the iteration for scenario B. Figure 2A shows the results of k-means clustering of the data (line 6 in Algorithm 4), with a solution score of 237.28. Figure 2B shows the data after shuffling the values of the non-informative feature (line 10 in Algorithm 4). The shuffled data has an almost identical solution score of 237.79 (line 12 in Algorithm 4) and a feature score of 0.51. Figure 2C shows the sample locations after shuffling the values of the informative feature, which gives a new solution score of 541.33 and a feature score of 304.05.

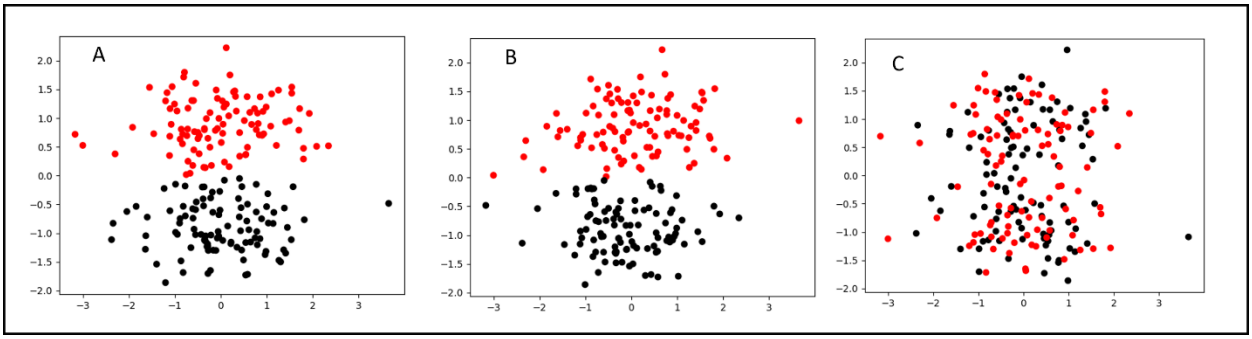


Figure 2. Illustrations of the different steps of FRIGATE for scenario B that is shown in Figure 1B, where one feature is informative (y axis) and one is non-informative (x axis). A – a clustering solution of the data (line 6 in Algorithm 4) colored by clusters labels. The solution score is 237.28 (line 8 in Algorithm 4). B- Results of shuffling the x coordinates, representing the non-informative feature (lines 10-13 in Algorithm 4). The solution score is similar to A and the feature's score is 0.51. C- Results of shuffling the y coordinates, representing the informative feature. An increase in the solution score led to a feature's score of 304.5.

The illustrations of scenarios A and C are given in Supplementary 2. In all scenarios the informative features scored much higher than the non-informative ones. Notice that the differences in scores are due to the initial solution score of each scenario – the poor results of scenario 3 already produced a relatively high solution score, so the ability of any feature to score high is limited.

3.5 Evaluation measures

When applied to a real dataset, each algorithm produces a ranking of the features. In our tests the truly informative features were unknown but the "true" clustering is known. We therefore applied the following procedure from [26]–[28] to evaluate the results. We ran k-means on the subset of the data

containing only the j top ranked features. The clustering produced was compared to the true labels available for the dataset using the Adjusted Rand Index (ARI) [42]. The process was repeated with increasing values of j, for $j\epsilon[1,N]$ for N number of features. The rationale was that a better feature ranking will manifest a high ARI for smaller values of j, as it puts the most informative features at the top. The process was repeated ten times per algorithm.

The above measure gives a value for the top j features, and a separate value for each j. We developed two new scores that summarize the measure across all values of j, while giving higher weight to the features that rank higher.

Suppose M feature ranking algorithms are compared on the same dataset. For each j, we compute the ARI of each algorithm on the top j features that it selected, and rank the algorithms based on their scores, from 1 for the top performer to M. For simplicity of the description, we assume there are no ties. The *weighted rank* of algorithm a is defined as:

$$WR(a) = \frac{2}{N(N+1)} \sum_j (N-j+1) * \left(\frac{M-rank(a,j)+1}{M}\right) \quad (21)$$

Here $rank(a,j)$ is the rank of algorithm a on the top j features. Hence, the second factor in the sum ranges from 1 for the top ranked algorithm to $1/M$ for the worst ranked, and the first factor gives a different weight to each j, from N for the first feature to 1 for the last ranked. The factor $\frac{2}{N(N+1)}$ rescales the total sum to $[0,1]$.

The WR measure is relative and depends on the set of algorithms tested. We introduce a second measure for a single algorithm. The algorithm's ARI score is computed for each top j features and weighted as above. The *weighted ARI* of algorithm a is defined as:

$$WARI(a) = \frac{2}{N(N+1)} \sum_j (N-j+1) * ARI(a,j) \quad (22)$$

where $ARI(a,j)$ is the ARI of algorithm a on the top j features. Hence, the range of the score is [-1,1] and higher scores are better.

Both scores can be generalized to handle ties and also situations where not all values of j are tested, e.g., when there are too many features.

4. Results

4.1 Algorithms Performance

We measured the performance of FRMV, FRSD, FRCM, FRIGATE and FRIGATE-MW on simulated and real data, including four genomic and seven EMR datasets. The number of clusters k in FRIGATE and for FRMV was chosen with the elbow method that we implemented as suggested in [41].

4.1.1 Simulated Data

We simulated data with 200 samples and 100 features of which 20 are informative, divided into two or four equal-sized clusters ($k = \{2,4\}$), mean distances $\mu = \{0.5,1,2,4\}$ and feature correlation levels $\sigma = \{0,0.05,0.2,0.5\}$. We ran the algorithms on data with and without z-score normalization. The *accurate recognition rate* is defined as the fraction of informative features in the top 20 ranked features. Results for $k = 4$ with $\mu = \{0.5,1\}$ are shown in Table 2, and the other cases are found in Supplementary 4. In all cases, the elbow method chose $k = 2$. On normalized data FRCM performed best, and FRIGATE-MW second. On non-normalized data FRIGATE-MW was best. FRMV scored poorly in all cases. FRSD scored poorly in most normalized scenarios, while in most non-normalized scenarios it scored high. We can also see that in general smaller values of μ and k account for harder cases, and normalized data is more challenging than non-normalized data. The FRIGATE variations and FRCM are affected by the correlation levels, where high levels of correlation cause a drop in performance. We can see the major drop in performance of these algorithms for $\sigma \geq 0.2$. FRSD and to some extant FRMV show opposite behavior, where extreme levels of correlation lead to improved results. This is counter-intuitive, as high correlation levels are expected to cause higher similarities between all features, including pairs of informative and non-informative ones. FRSD and FRMV are also more affected by the structure of the data (k, μ, normalized. See Supplementary 4) in comparison to FRIGATE and FRCM (see Discussion).

It is worth mentioning that as 20% of the features were informative, a score below 0.2 accounts for performance worse than random ordering of features. FRMV repeatedly scored below 0.2, FRSD scored low for most of the normalized cases with low correlation levels, and FRIGATE scored below random levels in the extreme correlation setting. FRCM is the only algorithm that rarely dropped significantly below random levels (Supplementary 4).

parameters	$\mu = 0.5$ $\sigma = 0$	$\mu = 0.5$ $\sigma = 0.05$	$\mu = 0.5$ $\sigma = 0.2$	$\mu = 0.5$ $\sigma = 0.5$	$\mu = 1$ $\sigma = 0$	$\mu = 1$ $\sigma = 0.05$	$\mu = 1$ $\sigma = 0.2$	$\mu = 1$ $\sigma = 0.5$
normalized								
FRIGATE	0.98 ± 0.03	0.91 ± 0.07	0.46 ± 0.17	0.09 ± 0.08	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	0.97 ± 0.05	0.09 ± 0.08
FRIGATE-MW	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	0.62 ± 0.33	0.19 ± 0.15	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	0.01 ± 0.02
FRCM	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{0.72 \pm 0.15}$	0.35 ± 0.18	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{0.99 \pm 0.03}$
FRSD	0.06 ± 0.04	0.06 ± 0.04	0.11 ± 0.06	$\mathbf{0.38 \pm 0.17}$	0.01 ± 0.02	0 ± 0	0.04 ± 0.05	0.32 ± 0.08
FRMV	0.13 ± 0.16	0.13 ± 0.13	0.25 ± 0.16	0.16 ± 0.12	0.05 ± 0.1	0.03 ± 0.03	0.09 ± 0.12	0.06 ± 0.16
non-normalized								
FRIGATE	0.99 ± 0.02	0.98 ± 0.03	0.76 ± 0.12	0.7 ± 0.15	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$
FRIGATE-MW	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0.02}$	$\mathbf{0.94 \pm 0.08}$	0.31 ± 0.14	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$
FRCM	$\mathbf{1 \pm 0}$	0.99 ± 0.02	0.82 ± 0.1	0.45 ± 0.2	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$
FRSD	0.79 ± 0.06	0.74 ± 0.11	0.77 ± 0.07	$\mathbf{0.89 \pm 0.06}$	$\mathbf{1 \pm 0.02}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$
FRMV	0.2 ± 0.19	0.12 ± 0.2	0.23 ± 0.2	0.11 ± 0.08	0.02 ± 0.06	0.02 ± 0.04	0.13 ± 0.2	0.1 ± 0.15

Table 2. Performance on simulated data, with $k = 4$. In **bold** are the top performers.

4.1.2 Real Data

We tested the five algorithms on 11 real genomic and EMR datasets from different sources for which a known clustering was available or created by us. The datasets are described in Table 3.

Figure 3 shows the performance of the algorithms on four genomic databases [45]–[48] (datasets 1-4 in Table 3). These datasets were used in a benchmark of clustering [49]. They have a large number of features and a modest number of samples (about two orders of magnitude lower). Note that here we do know the true clustering but we do not know which and how many features are informative, but it is expected that many features do not carry information relevant to the clustering. In all cases the value chosen by the elbow method for FRIGATE and FRMV was $k = 2$.

The performance of both variations of FRIGATE and FRSD was comparable and generally good, reaching maximum ARI of 0.35-0.7 already with less than 100 features in most cases. FRSD performed markedly better than the other methods on dataset 3 (Figure 3C). FRCM performed poorly in most cases, with slow gradual increase in ARI. FRMV performed better than the others on dataset 2 (Figure 3B), and its results had a wide variance across repetitions in most cases. It is worth mentioning that the description of the FRMV algorithm in [27] was not clear, especially calculating linear correlation between continuous

features and categorical cluster membership. This, as well as sampling features with replacement, can potentially create major variability between different runs of the algorithm.

No.	Source	Domain	Data Name	# of clusters	# of samples	# of features	Data type
1	[46], [49]	Genomic	Bredel-2005	3	50 (31,14,5)	1739	Continuous
2	[45], [49]	Genomic	Armstrong-2002-v2	3	72 (24,20,28)	2194	Continuous
3	[47], [49]	Genomic	Tomlins-2006	5	50 (27,20,32,13,12)	2315	Continuous
4	[48], [49]	Genomic	Nutt-2003-v1	4	50 (14,7,14,15)	1377	Continuous
5	MIMIC-III [4], [5]	EMR	Young cancer patients	2	161 (122,39)	70	Continuous
6	MIMIC-III [4], [5]	EMR	Young healthy patients	2	110 (84,26)	47	Continuous
7	MIMIC-III [4], [5]	EMR	Newborns	2	5286 (1534,3752)	29	Continuous
8	[50], [51]	EMR	Heart failure	2	169 (68,101)	77	Continuous
9	eICU [52], [53]	EMR	Intubated patients	2	441 (136, 305)	157 (87, 70)	Mixed
10	eICU [52], [53]	EMR	Short stay at ICU	2	570 (487, 83)	79 (59, 20)	Mixed
11	eICU [52], [53]	EMR	Young patients	2	232 (138, 94)	86 (72, 14)	Mixed

Table 3. Details of the real data sets used for the performance benchmark. The numbers in parentheses in column "# of samples" are the sizes of the clusters, and in the column "# of features" are the number of continuous and categorical features, respectively.

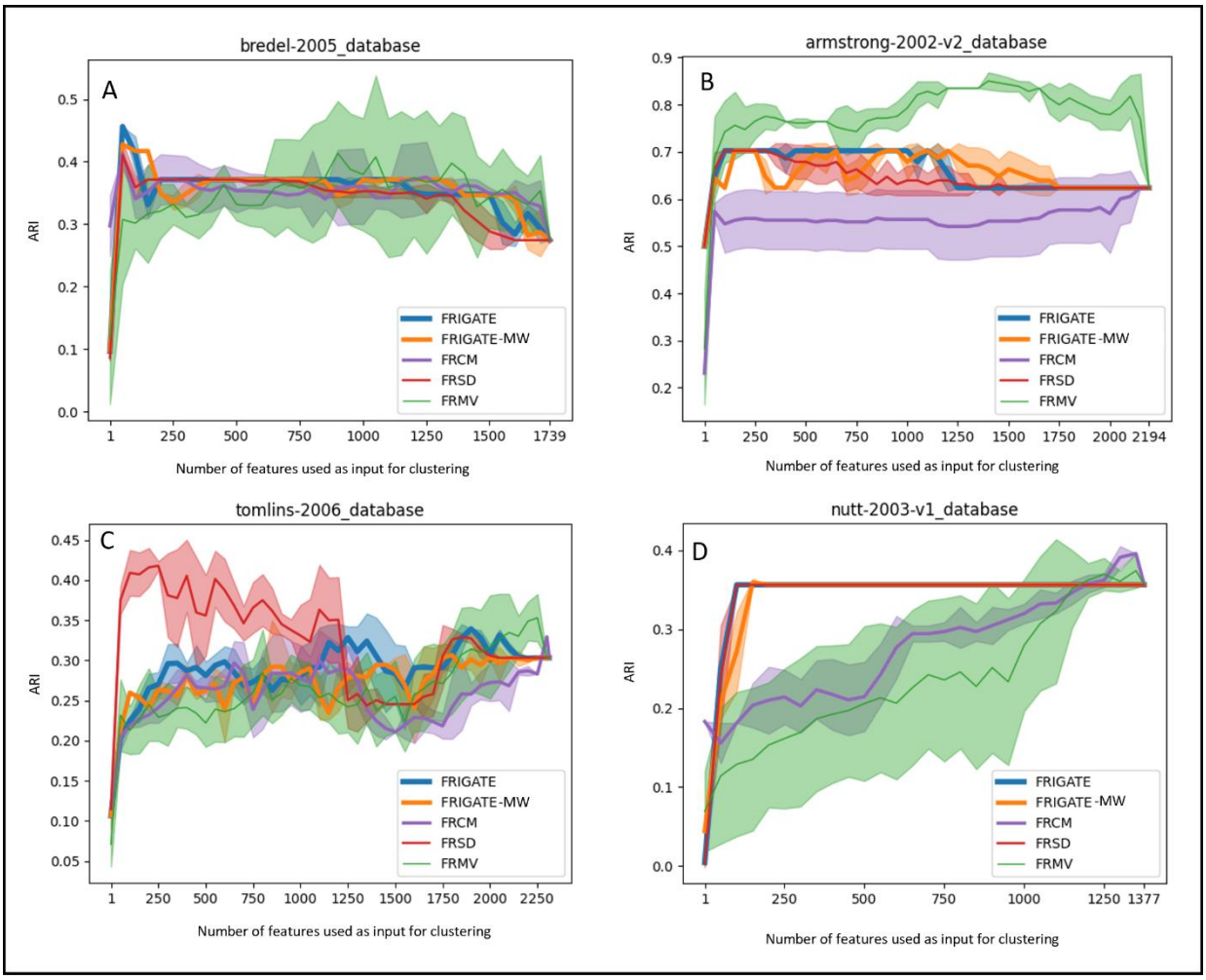


Figure 3. Performance of the tested algorithms on genomic datasets. The ranking produced by each algorithm was used to cluster the data with a growing number of features. The Y axis is the ARI score compared to the known clustering. The results are average of ten runs. The light-colored sleeve around each plot is ± 1 std. A-D for datasets 1-4 in Table 3, respectively.

We created three EMR datasets from the MIMIC-III repository [4], [5] and three from the eICU repository [52], [53], both downloaded from PhysioNet [3] (datasets 5-7, 9-11 in Table 3). The input features used were continuous, containing lab tests ("labs"), age and length of stay in the hospital (days in MIMIC and minutes in eICU) and Apache score in eICU. For each lab, we included only the first measurement that was available for the patient during the ICU stay. For each patient we included data from a single ICU stay. For the MIMIC datasets ICD-9 diagnosis codes were extracted per ICU stay and used for labeling the patients. For the eICU datasets, diagnoses and Apache score parameters were used as categorical variables and for labeling. Labs that were missing in >70% of the cohort were removed. To remove potential outliers, we z-scored each continuous measurement across the cohort, and removed patients that had any lab with $|z - score| \geq 3$. We then applied the Iterative Imputer as implemented

in [35] to the raw data to complete missing data and performed z-score normalization. The MIMIC cohorts that we constructed were:

1. Dataset 5 – patients that had a cancer ICD-9 diagnosis, aged 18-40. The data were divided into two clusters by length of stay: 122 patients who were discharged alive and spent less than 18 days in ICU, and 39 patients who either died during the ICU stay or stayed 18 days or more at the ICU. 70 features were recorded.
2. Dataset 6 – "healthy" patients: individuals aged 20-30 who did not have ICD-9 diagnosis of cancer, benign tumors, hypertension, cardiac disease, endocrine related disease, or hepatitis and stayed up to one day at ICU. They were divided into two clusters by sex: 84 males and 26 females. Here 47 features were recorded.
3. Datasets 7 – Newborns divided into two clusters: 1534 with jaundice and 3752 without jaundice, with 29 features.

The results on these datasets are shown in Figure 4A-C and summarized in Table 4. For Dataset 5 (Figure 4A), when using up to 50% of the ranked features FRIGATE performance was best. With over 50% of features FRCM results were comparable. For Dataset 6 (Figure 4B) FRCM was best followed by FRIGATE. FRMV performed comparably to FRIGATE and FRSD performed worst. For Dataset 7 (Figure 4C) with up to 50% of features FRCM performed best. With 50% or more of the ranked features the results of FRIGATE and FRMV were comparable to FRCM or better. FRSD was the worst performer.

Dataset 8 consists of heart failure patients from Zigong Fourth People's Hospital [50], [51], also extracted from PhysioNet. This cohort was divided into two age groups: 68 patients of ages 29-49 and 101 patients of ages 89-100. We had 77 features in this cohort after removing features with >30% missing data, and used the Iterative Imputer for missing data. The results are shown in Figure 4D and Table 4. Here FRSD performed comparably to FRIGATE and even slightly better in some thresholds, where FRMV and FRCM performed much worse, with especially poor results in the first 40% of features. A full comparison among the results is found in Supplementary 5.

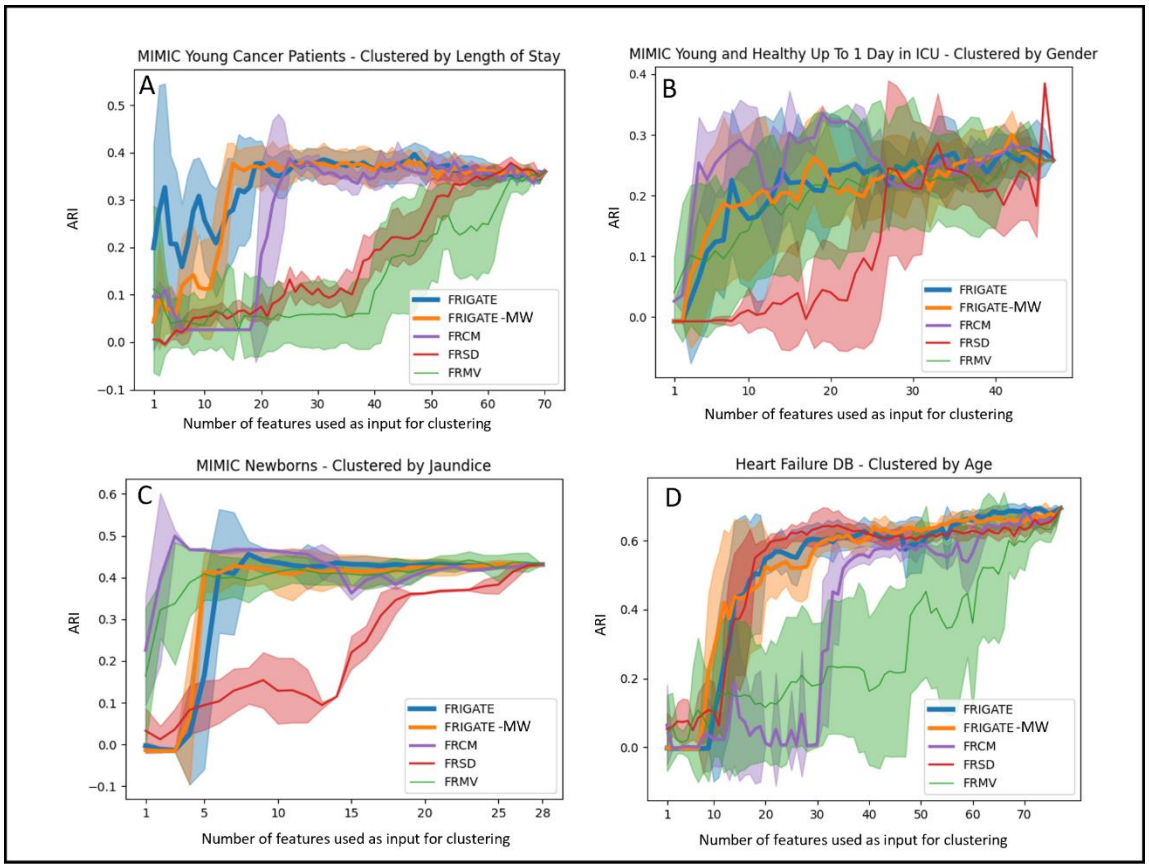


Figure 4. A-D: Performance on datasets 5-8 respectively. See Figure 3 for caption details.

Algorithm	Dataset 5		Dataset 6		Dataset 7		Dataset 8	
	ARI of top $\lceil 25\% \rceil$ features	ARI of top $\lceil 50\% \rceil$ features	ARI of top $\lceil 25\% \rceil$ features	ARI of top $\lceil 50\% \rceil$ features	ARI of top $\lceil 25\% \rceil$ features	ARI of top $\lceil 50\% \rceil$ features	ARI of top $\lceil 25\% \rceil$ features	ARI of top $\lceil 50\% \rceil$ features
FRIGATE	0.328 ± 0.105	0.372 ± 0.029	0.182 ± 0.149	0.237 ± 0.092	0.409 ± 0.146	$\mathbf{0.435 \pm 0.014}$	0.547 ± 0.045	$\mathbf{0.627 \pm 0.042}$
FRIGATE-MW	$\mathbf{0.37 \pm 0.042}$	$\mathbf{0.378 \pm 0.031}$	$\mathbf{0.21 \pm 0.119}$	0.198 ± 0.038	0.427 ± 0.042	0.417 ± 0.038	0.524 ± 0.103	0.601 ± 0.035
FRMV	$0.053 \pm 0.082^{*\dagger}$	$0.053 \pm 0.053^{*\dagger}$	0.181 ± 0.116	0.214 ± 0.104	0.401 ± 0.047	$0.41 \pm 0.027^{*}$	$0.116 \pm 0.154^{*\dagger}$	$0.221 \pm 0.233^{*\dagger}$
FRSD	$0.058 \pm 0.013^{*\dagger}$	$0.107 \pm 0.049^{*\dagger}$	$0.006 \pm 0.022^{*\dagger}$	$0.097 \pm 0.149^{*}$	$0.129 \pm 0.053^{*\dagger}$	$0.115 \pm 0.0^{*\dagger}$	$\mathbf{0.573 \pm 0.029}$	0.625 ± 0.025
FRCM	$0.026 \pm 0.0^{*\dagger}$	$0.35 \pm 0.027^{\dagger}$	$\mathbf{0.21 \pm 0.068}$	$\mathbf{0.291 \pm 0.054}^{\dagger}$	$\mathbf{0.465 \pm 0.007}$	0.422 ± 0.034	$0.012 \pm 0.004^{*\dagger}$	$0.559 \pm 0.024^{*\dagger}$

Table 4. Performance on Dataset 5-8. In **bold** are the top performers.

* - significant difference from FRIGATE, † - significant difference from FRIGATE-MW

The eICU cohorts that we constructed included Caucasian patients admitted directly to ICU with sex labels:

1. Dataset 9 – intubated patients aged 70 and above were divided according to status at discharge of “Alive”, 305 patients, and “Expired”, 136 patients. 87 continuous and 70 categorical features that had a value in at least 1% of the cohort were used.

2. Dataset 10 – patients who stayed up to one day in ICU, separated by age groups: 487 patients aged 18 to 80, and 83 patients aged 80 or older. 59 continuous and 20 categorical features that had a value in at least 5% of the cohort were used.
3. Dataset 11 – patients aged 18-30 separated by length of stay: 138 who stayed over 4.5 days (>6500 minutes) or expired, and 94 who stayed 4.5 days or less and were discharged alive. 72 continuous and 14 categorical features that had a value in at least 5% of the cohort were used.

The results for the eICU datasets are shown in Figure 5. Figures 5A, 5C, 5E compare all algorithms using the continuous features only. The same trends are observed – both versions of FRIAGTE and FRCM perform best, FRMV has a large variance in results and FRSD performs poorly.

We next used these datasets to test the ability to improve the results by adding categorical features. We tested different values of γ and looked for a change in the ARI of the full set of features in comparison to only using the continuous features (results not shown). A change in ARI means a different composition of the clusters caused by the categorical features. For $\gamma < 5$ in most cases there was no change in the composition of the clusters, and $\gamma > 6$ lead to a major decrease in ARI. We therefore chose $\gamma = 6$ in all cases. In most datasets we do not see an improvement, and in some cases more features were needed to reach high values of ARI. Overall, the categorical features did not improve the solution. Interestingly, in dataset 10 (Figure 5D) adding the categorical variables harmed the performance of FRIGATE-MW more than that of FRIAGATE.

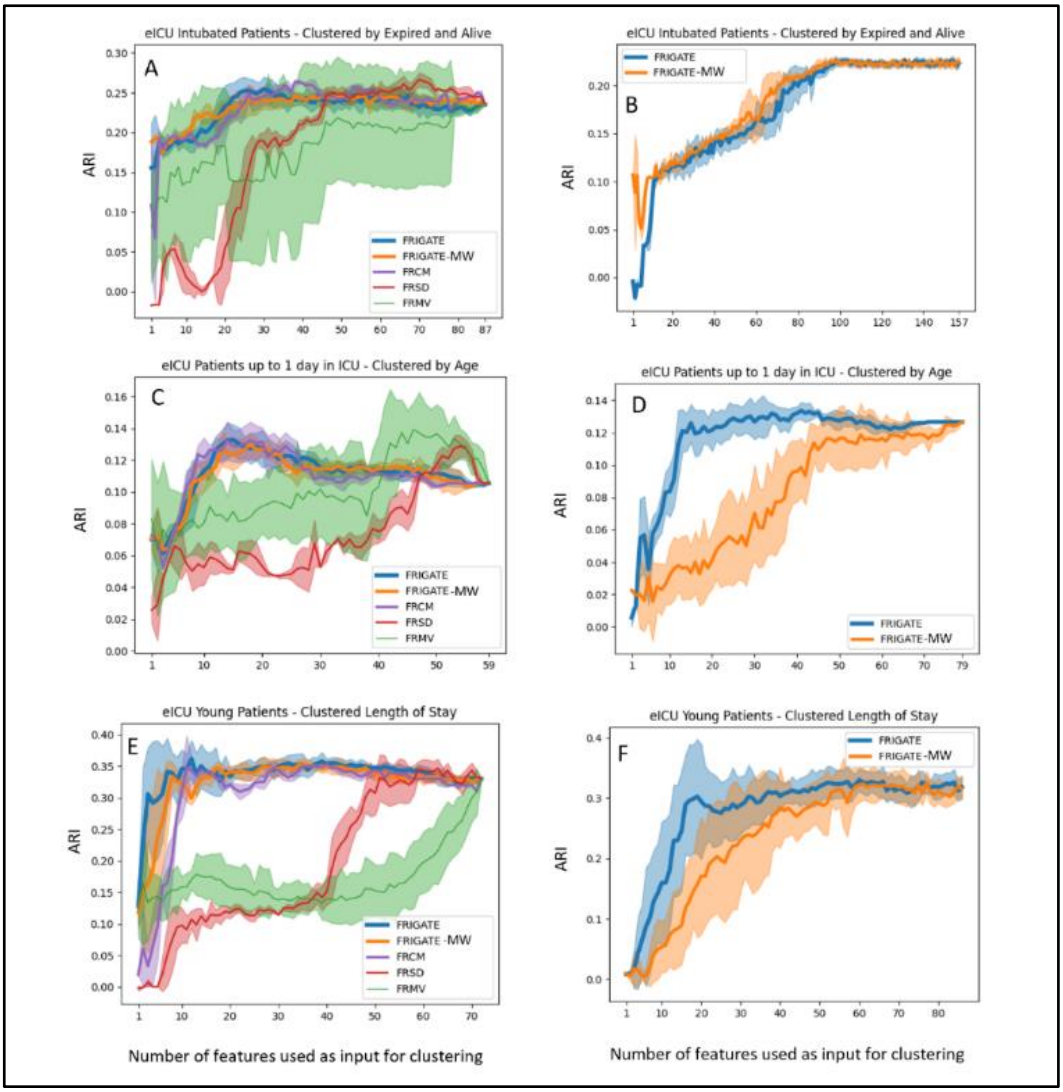


Figure 5. Performance on datasets 9-12 from the eICU repository. See Figure 3 captions for details. A-B: dataset 9, C-D: dataset 10, E-F: dataset 11. A, C, E show results when using only the continuous features of the datasets, and B, D, F show results for the mixed data.

In Table 5 we show the weighted rank (WR) and weighted ARI ($WARI$) scores of all algorithms for datasets 1-11. Apart from dataset 7 with the *WARI*, a variant of FRIGTAE is among the top two algorithms in all cases. In terms of *WR*, FRIGATE was best in the 4 cases and second in 4, and FRIGATE-MW was best in 1 and second in 6. In terms of WARI, FRIGATE was best in 4, second in 2, FRIGATE-MW best in 2 and second in 5 cases. FRCM was best in 3 and second in one case for both measures.

Dataset	1	2	3	4*	5	6*	7*	8	9	10	11
Weighted rank											
FRIGATE	**0.727**	0.662	0.678	**0.748**	**0.881**	0.613	0.657	0.649	0.668	0.667	**0.933**
FRIGATE-MW	0.699	0.582	0.565	0.735	0.807	0.606	0.522	0.682	**0.794**	0.751	0.752
FRMV	0.430	**0.951**	0.345	0.245	0.334	0.527	0.596	0.411	0.301	0.517	0.383
FRCM	0.514	0.195	0.468	0.429	0.519	**0.854**	**0.825**	0.358	0.789	**0.782**	0.625
FRSD	0.544	0.541	**0.879**	0.721	0.405	0.255	0.297	**0.800**	0.414	0.226	0.266
Weighted ARI											
FRIGATE	**0.347**	0.659	0.268	**0.312**	**0.314**	0.175	0.287	0.411	0.214	0.103	**0.327**
FRIGATE-MW	**0.347**	0.646	0.255	0.307	0.277	0.178	0.296	0.424	**0.223**	0.106	0.308
FRMV	0.321	**0.737**	0.236	0.179	0.085	0.166	0.367	0.185	0.161	0.089	0.152
FRCM	0.346	0.529	0.245	0.227	0.201	**0.236**	**0.412**	0.218	0.216	**0.108**	0.279
FRSD	0.339	0.641	**0.334**	0.311	0.105	0.058	0.143	**0.432**	0.126	0.058	0.128

Figure 5. Weighted rank and weighted ARI for the tested algorithms in datasets 1-11. In **bold** is the top performer for the dataset, underlined is the second best. * - Datasets where the top two performing algorithms were different for the two evaluation metrics.

4.2 Clinical Significance – Test Case

We wished to evaluate the clinical relevance of the leading chosen features to the target labels. We chose to focus on Dataset 6 as there is evidence for sex-based differences in lab tests [54]. We chose the twelve features that were available in both cohorts and according to [54] fulfil:

$$\frac{abs(x_{male}-x_{female})}{\max(x_{male},x_{female})} \geq 0.1 \qquad (23)$$

where x_i is the mean value of feature x for sex i [54]. We call these the top features. A ranked list of all features according to FRIGATE and FRIGATE-MW and the top features are in Supplementary 6.

We performed a hypergeometric test between the 12 top ranked features according to FRIGATE and the top features from [54], and similarly for FRIGATE-MW. For FRIGATE-MW, six of the top ranked features were also top features in [54] giving a significant p-value of 0.034. For FRIGATE, five of the top twelve features were common with the top features of [54], which accounts to a non-significant p-value of 0.136.

We also calculated the p-value of the minimum hypergeometric score (mHG), as used in the DRIM algorithm [55], for calculating the significance without determining in advance the threshold for the hypergeometric test and accounting for multiple testing. For FRIGATE the mHG was obtained for 13 features, with p-value of 0.07. For FRIGATE-MW the threshold was 10 features with p-value of 0.01.

It is important to remember that [54] refers to seemingly healthy individuals, while Dataset 6 comprised of patients who were in the ICU for up to one day, and some stayed overnight. That means that although the patients were young and did not require a major intervention, they still suffered from some medical condition. Indeed, the top feature in both versions of FRIGATE was "days in hospital" (more females stayed overnight, details not shown), which might suggest some correlation between the clusters and the medical condition, together with the correlation with sex.

4.3 Runtime comparison

Table 6 shows the runtimes on Databases 1-8 for the tested algorithms. The FRIGATE variants are slower on the genomic Datasets 1-4, which have many features and a few samples, but fast on the EMR datasets, which have less features. FRCM runs faster on Datasets 1-4, but when the number of samples grows its runtime increases sharply (Database 7).

The behavior of FRIGATE can be explained by the choice to set the number iterations depending on the number of features. However, this is a tunable parameter with a trade off with f, the number of features included per iteration (see Supplementary 1). FRCM, on the other hand, has a set number of iterations, and produces an mxm matrix for each feature, which is expensive both in runtime and in space. Note also the slowdown of FRSD on Dataset 7, which has thousands of patients.

Dataset no. /Algorithm	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
FRIGATE	1967.9 ± 28.2	4739.7 ± 80.2	5309.5 ± 136.3	1336.7 ± 21.4	49.8 ± 0.2	32.5 ± 0.1	286.8 ± 2.7	132.9 ± 1.1
FRIGATE-MW	2854.0 ± 11.9	4903.2 ± 45.6	5736.0 ± 35.084	1898.3 ± 6.4	40.9 ± 0.8	29.2 ± 1.2	296.3 ± 10.5	118.8 ± 1.3
FRCM	500.2 ± 6.2	1188.9 ± 14.3	2457.3 ± 31.4	$411.7.2 \pm 3.7$	229.6 ± 2.5	111.2 ± 1.3	74215.4 ± 938.4	268.6 ± 1.8
FRSD	1159.6 ± 6.0	1898.3 ± 8.2	2808.4 ± 6.2	998.2 ± 11.1	1450.9 ± 12.3	922.8 ± 30.9	39716.9 ± 256.0	1484.0 ± 15.9
FRMV	101.0 ± 0.6	123.7 ± 0.6	133.6 ± 1.2	82.8 ± 0.7	21.6 ± 0.2	19.8 ± 0.3	653.9 ± 3.3	47.3 ± 1.0

Table 6. Runtime in seconds for Datasets 1-8. Results are mean±STD of five runs for Datasets 1-4 and of three runs for Datasets 5-8 (the number of repetitions was reduced as the total runtime was large).

5. Discussion

We presented here FRIGATE, a new ensemble feature ranking algorithm for clustering, focusing on clustering of medical data. The need for such an algorithm arises from the growing volume of EMR data available to researchers [1]–[6], [52], [53]. Clustering, an unsupervised learning, is highly relevant for medical research, especially in recognition of disease subtypes [13]–[15].

When dealing with medical records researchers are facing two main challenges. The first challenge is the large number of features in patients' medical files. That includes lab tests, imaging results, diagnosis codes, past diseases etc. The second challenge is the need to be able to explain analysis results, which is of major importance in medical research as the ultimate goal is to derive clinically meaningful insights. These two challenges affect the ability to produce meaningful clusters: too many features will potentially mask the truly relevant ones, and in extreme cases require dimensionality reduction that will harm the explainability. Another issue that these challenges bring is computing power – while we do not want to lose information, running machine-learning algorithms on huge datasets can be expensive in runtime and space complexity.

The need for producing explainable clusters of patients and the challenges that it brings were our motivation for creating a new feature selection algorithm for clustering. FRIGATE is a feature ranking algorithm that uses the Shapley value concept to evaluate the importance of features, unattached to a specific clustering solution. In FRIGATE we perform iterations, in each we randomly choose a subset of features and produce a clustering solution on it, using k-means or a similar clustering algorithm. We use the objective function of the algorithm, which is the total distance to cluster centroids, as our *solution score*. Then, in a Shapley-like manner we eliminate the contribution of each feature separately by shuffling its values across all samples, and recalculate the solution score. The difference between the original solution score and the one obtained after shuffling is the *feature score* for the iteration. The total score of each feature is the average feature score, and is used to rank the full set of features. As FRIGATE uses the solution score for evaluating the features, using the clustering algorithms k-modes for categorical data and k-prototypes for mixed data allows FRIGATE to handle different data types, which is highly relevant in the medical domain.

In another variation of FRIGATE, that we call FRIGATE-MW, we use the MW reweighting method for choosing the features for each iteration – instead of at random. With MW, we update weights for the features in each iteration, where badly performing features will be less likely to be chosen. The idea is

that features that were proven non-informative are less relevant for the process, and focusing on evaluating the more important features will make the algorithm more efficient.

The process we used in FRIGATE, of producing multiple clustering solutions on subsets of features, is called ensemble feature ranking. Three other algorithms that follow such process were proposed in the past: FRMV, FRCM and FRSD [26]–[28]. All of them are based on multiple clustering solutions obtained with k-means. The code for the three algorithms was not provided by their authors, and we publish here our implementation to their codes, as well as for the two variations of FRIGATE, in the FRIGATE github repository: https://github.com/Shamir-Lab/FRIGATE.

Compared to the other algorithms, FRIGATE is the only one that incorporates categorical and mixed data features. This is also the first use of MW within the feature ranking for clustering framework and the first explicit use of Shapley values for unsupervised feature selection (although FRSD can also be seen as Shapley-like algorithm, see Background and Methods).

For choosing the hyper-parameters of FRIGATE and comparing the performance of the different algorithms we created simulated data built as presented in [26]. In the simulated data the informative features are known and we can measure the fraction of informative features among the top ranked features, a metric we call *accurate recognition rate*.

One parameter we compared between the algorithms is k, the number of clusters provided for the algorithms for producing the multiple clustering solutions. While FRCM and FRSD do not require k as an input and they average their results over multiple values of k, FRMV and FRIGATE require k as an input. Setting k is a challenging task, as in real data this parameter is inherently unknown by the nature of unsupervised learning. For FRIGATE and FRMV we used the elbow method, as described by [41], to set k. FRCM and FRIGATE seem unaffected by the value of k, and were able to achieve high accurate recognition rates on a wide range of k, some over four times larger than the real k. FRMV performed poorly constantly. FRSD seems highly affected by different values of k. As mentioned, FRSD does not receive k as an input, but including high values of k harms its performance. More interestingly, the best performance was obtained for $k = \{2,3\}$ when the real k was four. We cannot fully explain this phenomenon, but we can assume that FRSD is more affected by the data structure than the other algorithms and its behavior is not stable. This is also supported by the simulation results and its performance on real data that will be discussed shortly. On the other hand, FRIGATE and FRCM seem to be able to recognize features that create a separation in the data, even when a wrong k is provided.

Intuitively, a feature that separates the data into x clusters, can divide the data to any $k < x$ by union of clusters. Similarly, when $k > x$, redundant clusters can be formed, but if samples that came from different distributions fall into different clusters, the algorithm may be able to recognize the informative features. Given the robustness of FRIGATE, we would like to test in future research the ability to waive the required input k. FRSD and FRCM are averaging their results over different values of k, but this method is currently not relevant for FRIGATE, as the solution score is affected by to the number of clusters and averaging over different values of k will probably be biased. Therefore, a new method is needed.

In section 2.7 we mentioned some unclear parts of the FRMV algorithm, in particular the calculation of linear correlation coefficient between a continuous features' vector and the cluster membership vector, which depends on the arbitrary numbering of the clusters. When testing its performance, FRMV performed the worst in most cases with constantly high variation between different runs on the same data. For these reasons we will not discuss its results deeply.

The simulation results helped understanding the differences between the algorithms. We tested a variety of simulation parameters that created a wide range of complexities of the data. In all simulations 20% of the features were informative. That means that random arrangement of the features should result in an accurate recognition rate of 0.2. Generally, the performance on $k = 4$ was better than $k = 2$. It can be explained by the simulation setup, which affects the distance between clusters in space, especially in the non-normalized case. The distance between the centroids of two adjacent clusters is constant, so for k=4 there is a pair of clusters for which the distance between their centroids is twice that of the two clusters when k=2, and it is easier to recognize important features when the centroids are farther apart. As expected, performance improved for larger values of μ, which account for better separated clusters. The σ parameter reflects the level of correlations between features. As expected, higher correlation levels usually accounted for worse results. The best performer on the simulation data was FRCM, where its lowest accurate recognition rate was 0.14 for normalized data, with $k = 2$, $\mu = 0.5$ and $\sigma = 0.5$, an unrealistically high correlation level (see Supplementary 3). Its second worst value was 0.19 for the same parameters on non-normalized data, a comparable value to random assignment, and the rest of the results were significantly above random. In comparison, FRIGATE rates < 0.2 several times when $\sigma = 0.5$, and it seems to be more affected by the correlation levels than the other algorithms. For FRSD performance was sharply lower on normalized data in comparison to non-normalized. That was observed even in cases where FRIGATE and FRCM performed perfectly, while FRSD

had accurate recognition rate of 0 (e.g., $k = 2, \mu = 2, \sigma = 0$). However, FRSD seems unaffected by the correlation levels, and in some cases even significantly improved its performance on high levels of correlations (e.g., for $k = 2, \mu = 2, \sigma = 0.5$ the rate was 0.55). FRMV presented similar trends to FRSD. It is again an unexpected behavior by FRMV and FRSD with no clear explanation. Generally, higher correlation levels in our simulation framework means that the informative and non-informative features behave more similarly. So intuitively, it should be harder to set them apart. As mentioned, FRSD was more affected by the data structure than by the correlation levels. Our hypothesis is that enforcing such high levels of correlation between all features shaped the data so that the differences between features are better captured by the changes in the silhouette score, which is incorporated in FRSD. This should be further addressed in future research, both for understanding the behavior of FRSD as well as for understanding the effect of high levels of correlation on the structure of the simulated data.

In all cases where the algorithms had extremely low accurate recognition rates, below 0.2, that means that the informative features were notably absent from the top of the list - meaning that they were especially recognized as non-informative. When testing it and looking at the bottom 20 features of FRIGATE on 10 simulation runs with: $k = 4,\ \mu = 2,\ \sigma = 0.05$ and normalized data, $68 \pm 24\%$ of the informative features were in the bottom 20%. This suggests that not only that FRIGATE did not recognize the informative features, but high levels of correlation make the algorithm recognize the informative features as the most non-informative. Although these levels of correlation are unrealistic, the behavior of the algorithm is interesting and not understood at this point. Further research is needed to understand why the distance to centroids, which is objective function used by FRIGATE, was affected more dramatically for non-informative features when the correlation levels between all features were high.

We also compared the performance of the algorithms on genomic and EMR real data. While genomic data for clustering was available and used by another publication [28], quality EMR data for clustering was harder to obtain. Most of the EMR data we used was based on the publicly available MIMIC-III [4], [5] and eICU [52], [53], two very large datasets from American ICU facilities. The composition of patients in these data is highly heterogenic and deriving from them a cohort with coherent clusters was a challenging task. We based the clusters and their labels on demographics like age and gender, past diseases like cancer, and outcomes like length of stay in the ICU and mortality. On the real datasets, where no truly informative features were known, we used an alternative evaluation method where we measured the ARI levels for increasing number of ranked features, starting with a single feature, the top

ranked, and finishing with the full set of features. For a more rigorous comparison of the algorithms' performances, we developed two new evaluation matrices: *weighted rank,* which scores the algorithms based on their relative performance compared to the other tested algorithms; and *weighted ARI,* which scores an algorithm based on its ARI scores, irrespective of other algorithms.

The performance of the algorithms on the real data was different than on the simulated data. Generally, the FRIGATE variations performed best or comparable to the best performer in the majority of cases. On the genomic data FRSD performed better than FRCM, and FRMV was inferior, with the exception of one case where it was the best performer. On the medical data FRCM performed better than FRSD. This leads to several important conclusions. First, FRIGATE performs constantly well on real data. Secondly, although FRSD and FRMV performed relatively poorly on the simulated data, they can perform well in some real cases. Therefore, we conclude that the simulation is limited in its ability to represent real data.

Our runtime tests showed that FRIGATE is overall preferable. It is slower on datasets with large number of features and small number of samples, but for datasets with a large number of samples, FRIGATE is much faster than FRSD and FRCM, with a prohibitively high runtime by FRCM. In conclusion, FRIGATE is the only algorithm that performed constantly well, and had an acceptable runtime in all cases.

When comparing the performance of FRIGATE and FRIGATE-MW, we see different behaviors on simulated and real data. On simulated data, the two algorithms performed comparably, but when a difference was observed it was usually in favor of FRIGATE-MW. This suggests that MW has the potential to improve random selection of features in unsupervised tasks. On real data the difference between the variants was observed mostly on mixed data, where FRIGATE performed better than FRIGATE-MW. However, although the algorithm was designed to work with mixed data, including categorical features did not improve the results in comparison to using only the continuous features.

Currently, we cannot conclude if MW can significantly improve the feature selection process on real data, and which FRIGATE variation is preferable. Future work should evaluate the possible contribution of MW to the ensemble framework, and more specifically, broaden the options for cost functions, which are a key factor in MW. Also, more research is needed to understand the reasons that lead FRIGATE-MW to perform worse than FRIGATE, especially on mixed data cases.

We showed on a test case that the selected features by the FRIGATE algorithms are clinically relevant. This suggests that FRIGATE can be utilized not only as a feature selection algorithm, but also for assessing the importance of features in the data even on relatively small cohorts. This should be tested

in future research, and potentially compared to other feature importance methods like AMSD [40], [56], which can analyze feature importance for a given clustering result.

Our study has several limitations. We compared FRIGATE to three other algorithms whose code was not available. That means that their reported performance here is based solely on our implementation. This is mostly relevant to the runtime comparison. Other implementations might be able to achieve improved runtime to some of the tested algorithms.

Another limitation is FRIGATE's sub optimal results on mixed data. One issue is the choice of γ, the weight parameter for categorical features in the k-prototype clustering algorithm that we used. It was shown in the past [15], including in the original paper presenting k-prototype [39], that is not clear how to choose γ. Here we tried to evaluate γ in simulated datasets, with different ratios of continuous and categorical features. $\gamma = 1$ was consistently preferred, but in our tests on real data the best performance was achieved for γ=6, and other values were preferred in different publications [15], [36]. A proper choice of γ should be a main focus for future research.

A key limitation in the evaluation of EMR data was the validity of the clusters that we produced. Heterogenous cohorts like these of MIMIC and eICU may contain multiple overlapping subgroups, which may confound clustering attempts and their evaluation. Including mixed data where both the categorical and continuous features are relevant, was another challenging task. In our tests, the categorical features did not have a significant effect on the results, and in some cases their inclusion harmed the results. Also, all the datasets that we generated were partitioned into two clusters. More work is needed to create medical datasets appropriate for clustering from publicly available sources, that will include mixed data and account for more complicated cases with a larger number of clusters.

As discussed above, we used the elbow method for choosing k, the number of clusters. In all runs of both simulated and real data, the choice was $k = 2$ even when the real number of clusters was higher. This is aligned with a previous publication [15]. Although we showed on simulated data that FRIGATE is unaffected by using k that is different than the actual number of clusters, there is a need for a reliable method to choose k. This is also relevant to another limitation that was mentioned before: the simulated data are not fully representative of real data. It suggests that on real, more complex data than what we used here, using a suboptimal k will harm the performance of the algorithm. This should be investigated in future research.

6. References

[1] H. Atasoy, B. N. Greenwood, and J. S. Mccullough, "The digitization of patient care: a review of the effects of electronic health records on health care quality and utilization," *Annual Review of Public Health Annu. Rev. Public Health*, vol. 13, no. 1, pp. 487–500, 2019, doi: 10.1146/annurev-publhealth.

[2] B. Fecher, S. Friesike, and M. Hebing, "What drives academic data sharing?," *PLoS One*, vol. 10, no. 2, Feb. 2015, doi: 10.1371/journal.pone.0118053.

[3] A. Goldberger *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation [Online]*, vol. 101, no. 23, pp. e215–e220, 2000.

[4] A. Johnson, T. Pollard, and R. Mark, " MIMIC-III Clinical Database (version 1.4)," PhysioNet.

[5] A. E. W. Johnson *et al.*, "MIMIC-III, a freely accessible critical care database," *Sci Data*, vol. 3, May 2016, doi: 10.1038/sdata.2016.35.

[6] C. Sudlow *et al.*, "UK Biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age," *PLoS Med*, vol. 12, no. 3, Mar. 2015, doi: 10.1371/journal.pmed.1001779.

[7] Y. Luo, "Evaluating the state of the art in missing data imputation for clinical data," *Brief Bioinform*, vol. 23, no. 1, Jan. 2022, doi: 10.1093/bib/bbab489.

[8] A. Garg and V. Mago, "Role of machine learning in medical research: A survey," *Computer Science Review*, vol. 40. Elsevier Ireland Ltd, May 01, 2021. doi: 10.1016/j.cosrev.2021.100370.

[9] M. M. Papathanasiou, M. Onel, I. Nascu, and E. N. Pistikopoulos, "Chapter 6 - Computational tools in the assistance of personalized healthcare," *Computer Aided Chemical Engineering*, vol. 42, pp. 139–206, 2018.

[10] S. Khanmohammadi, N. Adibeig, and S. Shanehbandy, "An improved overlapping k-means clustering method for medical applications," *Expert Syst Appl*, vol. 67, pp. 12–18, Jan. 2017, doi: 10.1016/j.eswa.2016.09.025.

[11] G. Battineni, G. G. Sagaro, N. Chinatalapudi, and F. Amenta, "Applications of machine learning predictive models in the chronic disease diagnosis," *Journal of Personalized Medicine*, vol. 10, no. 2. MDPI AG, Jun. 01, 2020. doi: 10.3390/jpm10020021.

[12] P. Wang, X. Zheng, J. Li, and B. Zhu, "Prediction of epidemic trends in COVID-19 with logistic model and machine learning technics," *Chaos Solitons Fractals*, vol. 139, Oct. 2020, doi: 10.1016/j.chaos.2020.110058.

[13] Y. Wang *et al.*, "Unsupervised machine learning for the discovery of latent disease clusters and patient subgroups using electronic health records," *J Biomed Inform*, vol. 102, Feb. 2020, doi: 10.1016/j.jbi.2019.103364.

[14] G. Tosto, S. E. Monsell, S. E. Hawes, G. Bruno, and R. Mayeux, “Progression of extrapyramidal signs in Alzheimer’s disease: clinical and neuropathological correlates,” *Journal of Alzheimer’s Disease*, vol. 49, no. 4, pp. 1085–1093, Jan. 2016, doi: 10.3233/JAD-150244.

[15] E. Shpigelman *et al.*, “Clustering of clinical and echocardiographic phenotypes of covid-19 patients,” *Sci Rep*, vol. 13, no. 1, p. 8832, Dec. 2023, doi: 10.1038/s41598-023-35449-1.

[16] H. P. Ng, S. H. Ong, K. W. C. Foong, P. S. Goh, and W. L. Nowinski, “Medical image segmentation using k-means clustering and improved watershed algorithm,” in *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, 2006, pp. 61–65. doi: 10.1109/ssiai.2006.1633722.

[17] M. R. Anderberg, *Cluster analysis for applications: probability and mathematical statistics: a series of monographs and textbooks*, vol. 19. Academic press, 2014.

[18] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall Inc., 1988.

[19] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis.* John Wiley & Sons, 2009.

[20] T. S. Madhulatha, “An overview on clustering methods,” *arXiv preprint arXiv:1205.1117*, May 2012.

[21] M. J. Li, M. K. Ng, Y. M. Cheung, and J. Z. Huang, “Agglomerative fuzzy K-Means clustering algorithm with selection of number of clusters,” *IEEE Trans Knowl Data Eng*, vol. 20, no. 11, pp. 1519–1534, Nov. 2008, doi: 10.1109/TKDE.2008.88.

[22] J. McQueen, “Some methods for classification and analysis of multivariate observations,” *Proc. 5th Berkeley Symp. Math. Statist. Prob.*, vol. 1, pp. 281–297, 1967.

[23] S. Solorio-Fernández, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, “A review of unsupervised feature selection methods,” *Artif Intell Rev*, vol. 53, no. 2, pp. 907–948, Feb. 2020, doi: 10.1007/s10462-019-09682-y.

[24] K. Y. Yeung and W. L. Ruzzo, “Principal component analysis for clustering gene expression data,” *Bioinformatics*, vol. 17, no. 9, pp. 763–774, 2001.

[25] J. G. Dy and C. E. Brodley, “Feature selection for unsupervised learning,” *Journal of Machine Learning Research*, vol. 5, pp. 845–889, 2004.

[26] J. Yu, H. Zhong, and S. B. Kim, “An ensemble feature ranking algorithm for clustering analysis,” *J Classif*, vol. 37, no. 2, pp. 462–489, Jul. 2020, doi: 10.1007/s00357-019-09330-8.

[27] Y. Hong, S. Kwong, Y. Chang, and Q. Ren, “Consensus unsupervised feature ranking from multiple views,” *Pattern Recognit Lett*, vol. 29, no. 5, pp. 595–602, Apr. 2008, doi: 10.1016/j.patrec.2007.11.012.

[28] S. Zhang, H. S. Wong, Y. Shen, and D. Xie, “A new unsupervised feature ranking method for gene expression data based on consensus affinity,” *IEEE/ACM Trans Comput Biol Bioinform*, vol. 9, no. 4, pp. 1257–1263, 2012, doi: 10.1109/TCBB.2012.34.

[29] D. Guan, W. Yuan, Y. K. Lee, K. Najeebullah, and M. K. Rasel, “A review of ensemble learning based feature selection,” *IETE Technical Review (Institution of Electronics and Telecommunication Engineers, India)*, vol. 31, no. 3. Medknow Publications, pp. 190–198, 2014. doi: 10.1080/02564602.2014.906859.

[30] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *J Comput Appl Math*, vol. 20, pp. 53–65, 1987.

[31] Shapley Loid S., “A value for n-person games,” *Contributions to the Theory of Games*, pp. 307–317, 1953.

[32] S. Mukund and A. Najmi, “The many Shapley values for model explanation,” *International conference on machine learning*, 2020.

[33] S. Arora, E. Hazan, and S. Kale, “The multiplicative weights update method: a meta-algorithm and applications,” *Theory of Computing*, vol. 8, no. 1, pp. 121–164, 2012, doi: 10.4086/toc.2012.v008a006.

[34] D. Arthur and S. Vassilvitskii, “k-means++: the advantages of careful seeding,” in *SODA ’07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.

[35] F. Pedregosa, V. Michel, and O. Grisel, “Scikit-learn: machine learning in python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] Z. Huang, “A fast clustering algorithm to cluster very large categorical data sets in data mining,” *Dmdk*, vol. 3, no. 8, pp. 34–39, May 1997.

[37] F. Cao, J. Liang, and L. Bai, “A new initialization method for categorical data clustering,” *Expert Syst Appl*, vol. 36, no. 7, pp. 10223–10228, Sep. 2009, doi: 10.1016/j.eswa.2009.01.060.

[38] N. J. de Vos, “kmodes categorical clustering library.” Accessed: Jul. 10, 2023. [Online]. Available: https://github.com/nicodv/kmodes

[39] Z. Huang, “Extensions to the k-means algorithm for clustering large data sets with categorical values,” *Data Min Knowl Discov*, vol. 12, pp. 283–304, 1998, doi: https://doi.org/10.1023/A:1009769707641.

[40] G. Preud’homme *et al.*, “Head-to-head comparison of clustering methods for heterogeneous data: a simulation-driven benchmark,” *Sci Rep*, vol. 11, no. 1, Dec. 2021, doi: 10.1038/s41598-021-83340-8.

[41] N. Rappoport and R. Shamir, “Multi-omic and multi-view clustering algorithms: review and cancer benchmark,” *Nucleic Acids Res*, vol. 46, no. 20, pp. 10546–10562, Nov. 2018, doi: 10.1093/nar/gky889.

[42] L. Hubert and P. Arabie, “Comparing Partitions,” *Journal of Classification 2, 193–218 ().*, vol. 2, pp. 193–218, Dec. 1985.

[43] W. M. Rand, “Objective criteria for the evaluation of clustering methods,” *J Am Stat Assoc*, vol. 66, no. 336, pp. 846–850, 1971.

[44] S. Douglas, “Properties of the Hubert-Arable adjusted Rand index,” *Psychological Methods* , vol. 9, no. 3, pp. 386–396, 2004.

[45] S. A. Armstrong *et al.*, “MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia,” *Nat Genet*, vol. 30, no. 1, pp. 41–47, 2002, doi: 10.1038/ng765.

[46] M. Bredel *et al.*, “Functional network analysis reveals extended gliomagenesis pathway maps and three novel MYC-interacting genes in human gliomas,” *Cancer Res*, vol. 65, no. 19, pp. 8679–8689, Oct. 2005, doi: 10.1158/0008-5472.CAN-05-1204.

[47] S. A. Tomlins *et al.*, “Integrative molecular concept modeling of prostate cancer progression,” *Nat Genet*, vol. 39, no. 1, pp. 41–51, Jan. 2007, doi: 10.1038/ng1935.

[48] C. L. Nutt *et al.*, “Gene expression-based classification of malignant gliomas correlates better with survival than histological classification,” *Cancer Res*, vol. 63, no. 7, pp. 1602–1607, 2003.

[49] M. C. P. de Souto, I. G. Costa, D. S. A. de Araujo, T. B. Ludermir, and A. Schliep, “Clustering cancer gene expression data: A comparative study,” *BMC Bioinformatics*, vol. 9, Nov. 2008, doi: 10.1186/1471-2105-9-497.

[50] Z. Zhang *et al.*, “Hospitalized patients with heart failure: integrating electronic healthcare records and external outcome data (version 1.3),” PhysioNet.

[51] Z. Zhang *et al.*, “Electronic healthcare records and external outcome data for hospitalized patients with heart failure,” *Sci Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1038/s41597-021-00835-9.

[52] T. Pollard, A. Johnson, J. Raffa, L. A. Celi, O. Badawi, and R. Mark, “eICU collaborative research database (version 2.0),” *PhysioNet*, 2019, doi: https://doi.org/10.13026/C2WM1R.

[53] T. J. Pollard, A. E. W. Johnson, J. D. Raffa, L. A. Celi, R. G. Mark, and O. Badawi, “The eICU collaborative research database, a freely available multi-center database for critical care research,” *Sci Data*, vol. 5, Sep. 2018, doi: 10.1038/sdata.2018.178.

[54] N. M. Cohen *et al.*, “Personalized lab test models to quantify disease potentials in healthy individuals,” *Nat Med*, vol. 27, no. 9, pp. 1582–1591, Sep. 2021, doi: 10.1038/s41591-021-01468-6.

[55] E. Eden, D. Lipson, S. Yogev, and Z. Yakhini, “Discovering motifs in ranked lists of DNA sequences,” *PLoS Comput Biol*, vol. 3, no. 3, pp. 0508–0522, 2007, doi: 10.1371/journal.pcbi.0030039.

[56] P. C. Austin, “An introduction to propensity score methods for reducing the effects of confounding in observational studies,” *Multivariate Behav Res*, vol. 46, no. 3, pp. 399–424, May 2011, doi: 10.1080/00273171.2011.568786.

7. Supplementary 1 – Parameter choice

7.1 Parameter choice for FRIGATE-MW

FRIGATE-MW has three parameters: $\boldsymbol{T}$, the number of iterations, $\boldsymbol{f}$, the fraction of the features used in each iteration, and $\boldsymbol{\eta}$, the MW parameter. If the input is mixed data there is also $\boldsymbol{\gamma}$, the K-prototypes parameter. We seek values of T and f that are small in order to increase the efficiency of the algorithm, but still produce good results.

For testing the performance of FRIGATE-MW with different hyper-parameters, we built a simulation as described in chapter 3.3, with the following parameters: $k = 4$ clusters of size $c = 50$ each, $\alpha = 20$ informative features and $\beta = 80$ non-informative ones. The statistical parameters were $\mu = 1$ and $\sigma = 0.05$. The value of σ was chosen to produce similar correlations to those observed in real data (see details in Supplementary 3).

Figure S1 shows the average accurate recognition rate results of 10 simulations for different values of f, T and η. As expected, more iterations and larger f produce better results. We can see that for 200 iterations the results are perfect for all values of η with $f \geq 0.1$, and the results for 100 iterations are a close second (perfect results for η=0.25 and 0.5, and 0.995 for 1). Increasing η from 0.25 to 1 improves the results. For the same parameters of the simulation but $\mu = 4$ (which causes the clusters to be more distinct) we observed similar results (Figure S2).

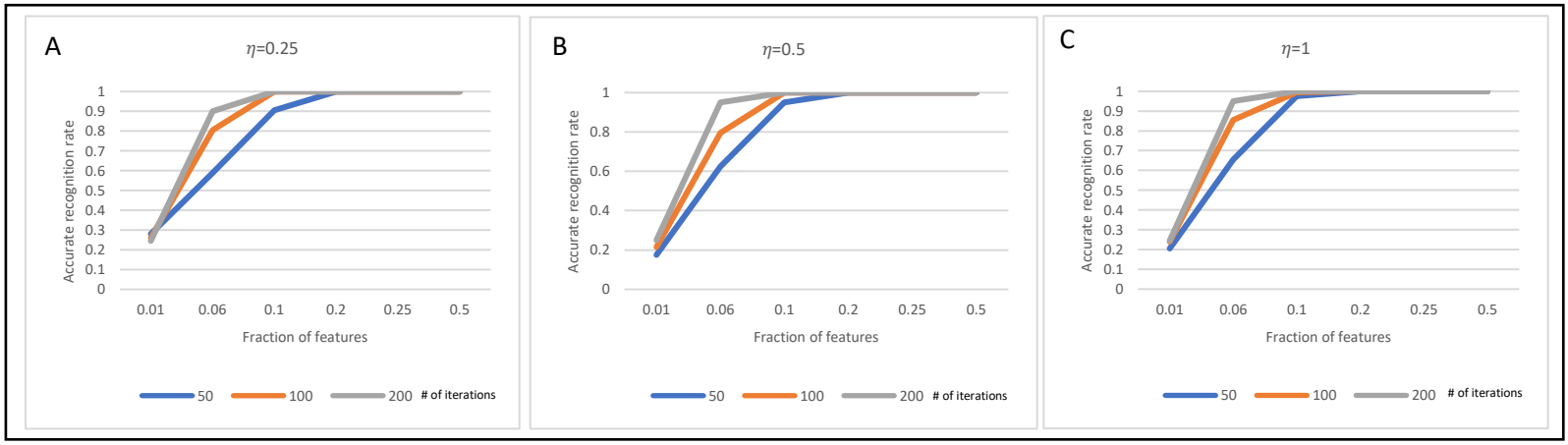


Figure S1. Performance of FRIGATE-MW on simulated data for different hyper-parameters. The simulation had 4 clusters, 50 samples in each cluster, and 100 features of which 20 were informative. The differences between the clusters were set by $\mu = 1, \sigma = 0.05$. The graphs show the accurate recognition rate of the algorithm, averaged over 10 runs. Plot color: number of iterations. X axis: fractions of features participating in each iteration. A, B and C present results for values of $\eta = 0.25$, 0.5 and 1 respectively.

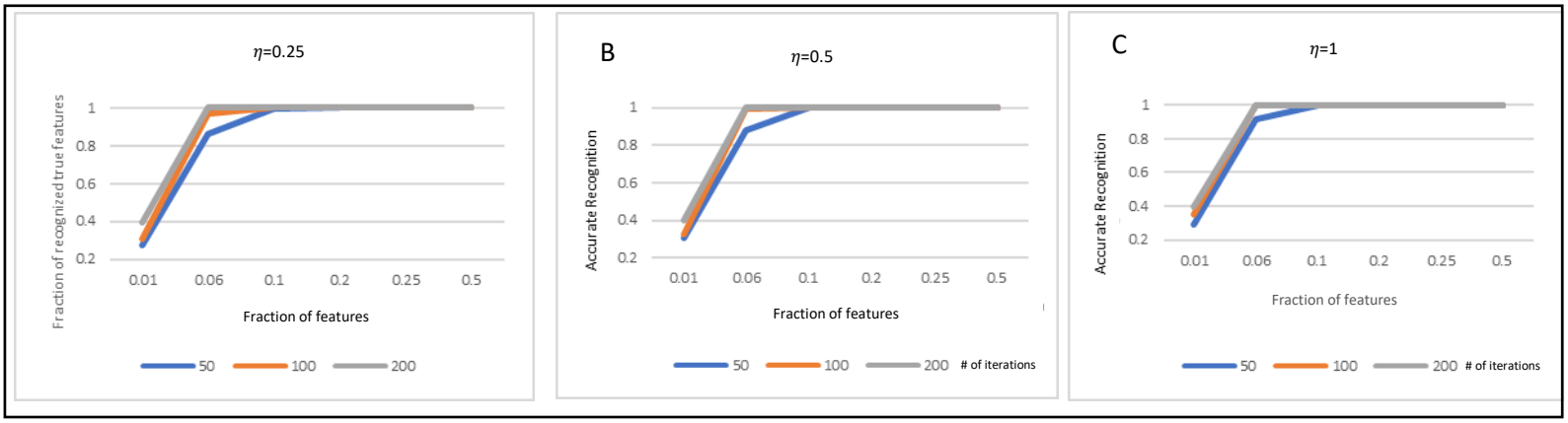


Figure S2. Performance of FRIGATE-MW with different hyper-parameters. The simulation setup and presented results are as in Figure S1, but here we used $\mu = 4$.

We also tested the scenario of two clusters with 100 samples in each cluster ($k = 2, c = 100$) and kept the rest of the parameters unchanged (Figure S3). The trends are similar to Figure S1, but we can see that the overall scores are slightly worse. Still, the combination with $f = 0.2$, $\eta \geq 0.5$ and $T = 200$ achieved perfect results. A similar simulation with $\mu = 4$ achived high results in all scenarios (Figure S4). In another simulation that used values similar to [26]: $k = 4,\ c = 50\ and\ n = 950$ the results were perfect for all parameters tested. We therefore chose to fix the hyper-parameters to $f = 0.1$, and $T = 2 \cdot |V|$.

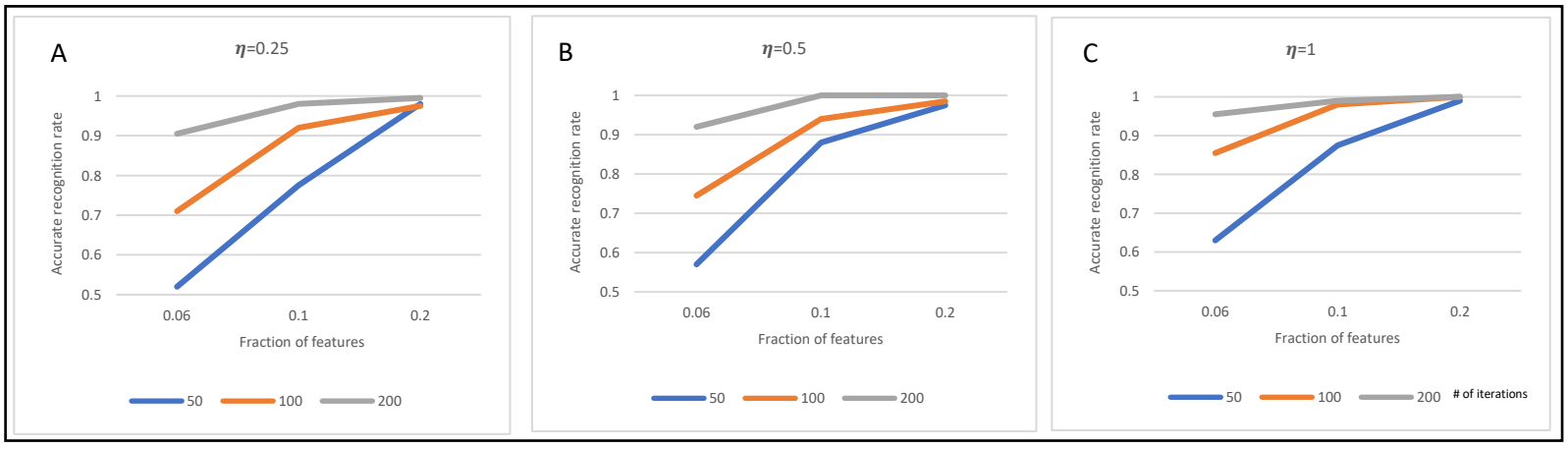


Figure S3. Performance of FRIGATE-MW on simulated data for different hyper-parameters. The simulation setup and presented results are as in Figure S1, but here there were 2 clusters and 100 samples in each cluster.

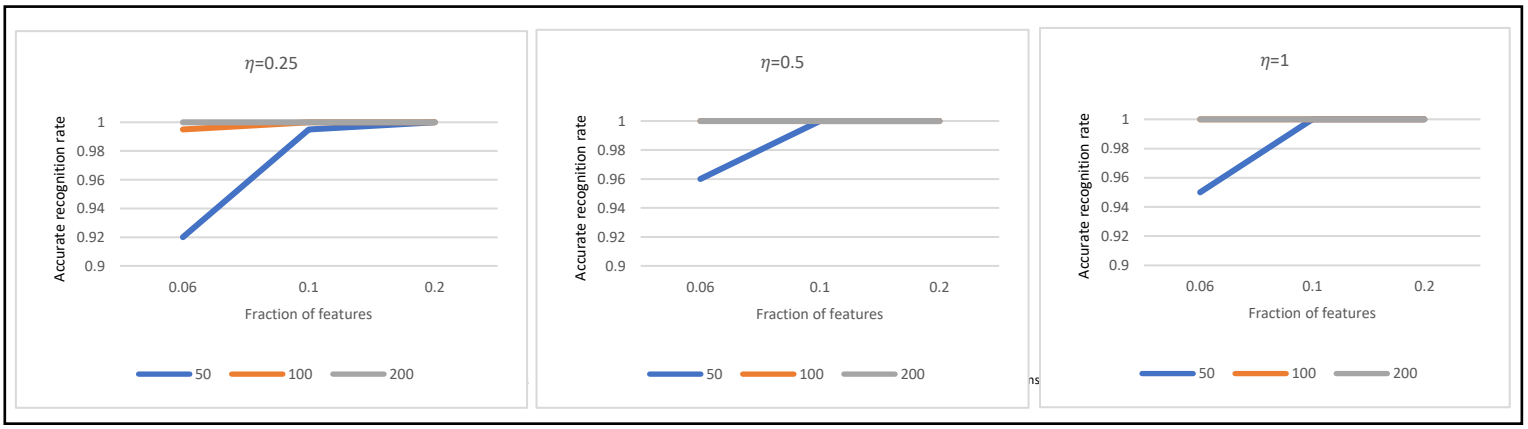


Figure S4. Performance of FRIGATE-MW on simulated data for different hyper-parameters. The simulation setup and presented results are as in Figure S3, but here we used $\mu = 4$..

7.2 Parameter choice for FRIGATE

The results of FRIAGTE in the same simulations are shown in Figure S5 for $k = 2, 4$ and $\mu = 1,4$. We can see that the algorithm without MW improves more slowly with the number of iterations and is in general inferior. This can be seen especially in the hardest case of $\mu = 1; k = 2$. Based on these results we chose as default values $f = 0.1, T = 2 \cdot |V|$ for the non-MW algorithm as well.

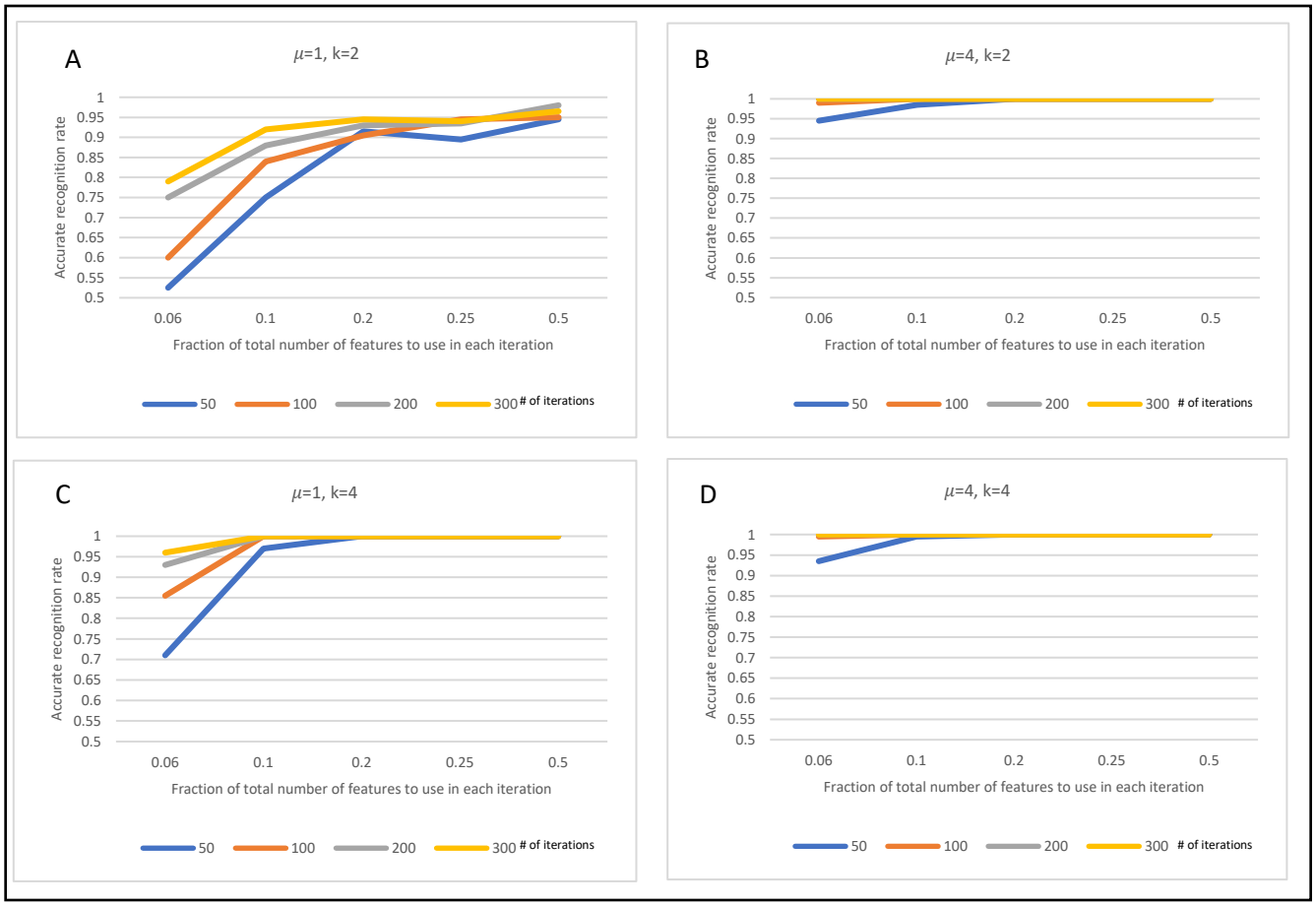


Figure S5. Performance of FRIGATE on simulated data for different hyper-parameters. Simulation parameters: σ=0.05, total number of samples=200, 100 feature of which 20 are informative. A: $\mu = 1, k = 2$. B: $\mu = 4, k = 2$. C: $\mu = 1, k = 4$. D: $\mu = 4, k = 4$.

Next, we wanted to determine the value of the weight parameter γ in the k-prototypes algorithm, when using mixed data. We performed simulations with 100 continuous features of which 20 are informative ($\alpha = 20, \beta = 80$), $\sigma = 0.05$ and 200 samples equally divided between clusters. The other parameters were tested in multiple options: $k = \{2,4\}$ (accounted for $c = 100$ and $c = 50$ respectively), $\mu = \{1,4\}$, {100 categorical features of them 20 informative ($\alpha_{categorical} = 20, \beta_{categorical} = 80$), 50 categorical features of them 10 informative ($\alpha_{categorical} = 10,\ \beta_{categorical} = 40$)}. We ran each scenario 10 times and for each type of features calculated the accurate recognition rate in the top thirty or forty features, depending on the total number of informative features.

The average results of FRIGATE and FRIGATE-MW for $\mu = 1$ and $k = 2$ are shown in Tables S2-S5. We can see that in all cases $\gamma = 1$ performed perfectly and $\gamma = 2$ near perfectly in both scenarios, unrelated to the proportion of continuous and categorical features. As expected, extreme values of γ, which favor

one type of features over the other, were detrimental. For $k = 4$ and $\mu = 4$ the results were nearly perfect in all scenarios and thus not informative (results not shown).

Table S2- FRIGATE-MW	γ			
	0.5	**1**	**2**	**4**
Continuous Features	1	1	0.99	0.845
Categorical Features	1	1	1	1

Table S3- FRIGATE-WW	γ			
	0.5	**1**	**2**	**4**
Continuous Features	1	1	0.975	0.38
Categorical Features	0.82	1	1	1

Table S4- FRIGATE	γ			
	0.5	**1**	**2**	**4**
Continuous Features	1	1	0.96	0.595
Categorical Features	0.99	1	1	1

Table S5- FRIGATE	γ			
	0.5	**1**	**2**	**4**
Continuous Features	1	1	1	0.93
Categorical Features	0.97	1	1	1

Tables S2-S5. Average accurate recognition rates in 10 simulation runs, for mixed data by FRIGATE and FRIGATE-MW for different values of the weight parameter γ in the k-prototypes algorithm. Simulation parameters: $\sigma = 0.05$, c_j = 100, $\alpha = 20$, $\beta = 80$, $\mu = 1$, $k = 2$. Tables 3,5: $\alpha_{categorical} = 20$, $\beta_{categorical} = 80$. Tables 4,6: $\alpha_{categorical} = 10$, $\beta_{categorical} = 40$.

The results of this simulation suggest that $\gamma = 1$ is the favorable value. However, this may not hold for all cases. We can see that for $\gamma > 2$ the results are poor, regardless of the proportion between continuous and categorical features, in contradiction with pervious publications [15]. More research is needed regarding the choice of γ.

8. Supplementary 2 - Illustration of a FRIGATE iteration

Scenario 1 – two informative features

Figure S6A shows the results of k-means clustering of the simulated data with two informative features (line 6 in Algorithm 4), with a solution score of 81.76. Figure S6B shows the data after shuffling the values of the x-coordinate feature (line 10 in Algorithm 4). The new solution score is 395.24 (line 12 in Algorithm 4) and a feature score of 313.48. Figure S6C shows the data after shuffling the values of the y-coordinate feature, which led to a new solution score of 368.09 and a feature score of 286.33.

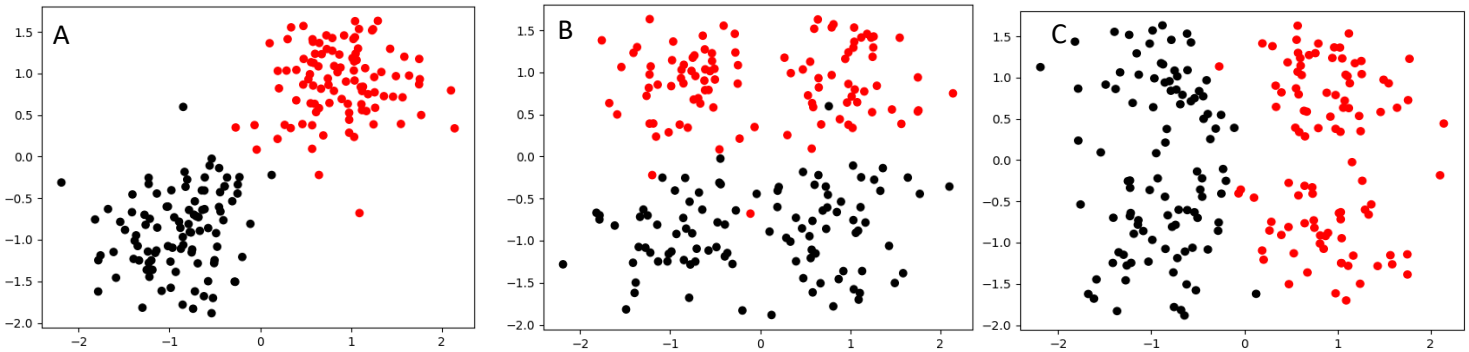


Figure S6. Illustration of the different steps of FRIGATE for scenario 1 of two informative features. A – A clustering solution of the data (line 6 in Algorithm 4) colored by clusters labels. The solution score is 81.76 (line 8 in Algorithm 4). B- Results of shuffling the x-coordinate (lines 10-13 in Algorithm 4). The solution score increased dramatically to 286.33. C- Results of shuffling the y-coordinate feature. We see again a great change from S1A with a solution score of 313.48.

Scenario 3 – two non-informative features

Figure S7A shows the results of k-means clustering of the data where both features are non-informative. The solution score is 256.48. Figure S7B shows the data after shuffling the values of y-coordinate feature. The new solution score is 374.07 (line 12 in Algorithm 4) and a feature score of 114.59. Figure S7C shows the data after shuffling the values of the x-coordinate feature, which led to a solution score of 407.04 and feature score of 150.56.

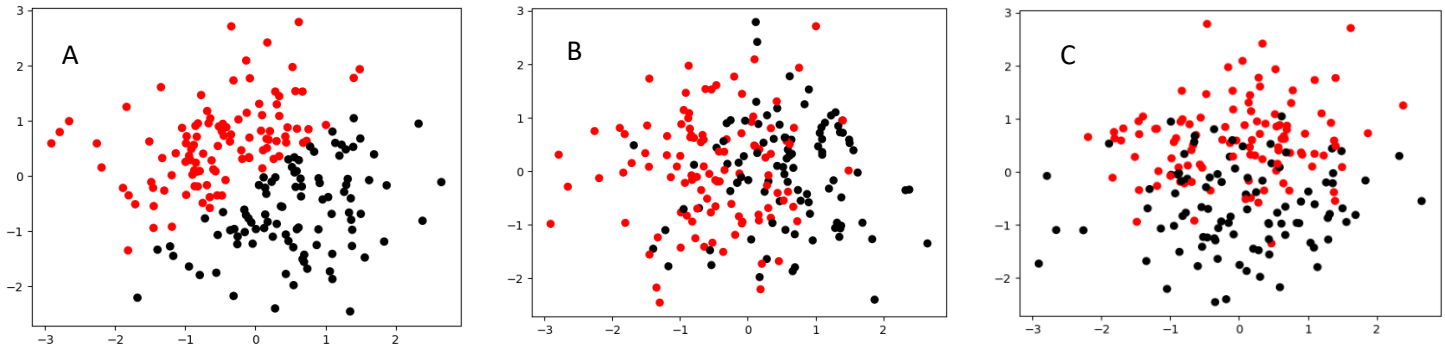


Figure S7. Illustration of the different steps of FRIGATE for scenario 3 of two non-informative features. A – A clustering solution of the data (line 6 in Algorithm 4) colored by clusters labels. The solution score is 256.48 (line 8 in Algorithm 4). B- Results of shuffling the x-coordinate feature (lines 10-13 in Algorithm 4). The solution score changed to 150.56. C- Results of shuffling the y-coordinate feature. The solution score is similar to Figure S2B, 114.59.

9. Supplementary 3 - Choosing the value of σ

We tested the average Pearson correlation between the different features when constructing the simulation using different values of σ. We tested three scenarios and the average results of 10 runs for $\mu = 2$ are presented in Figure S8:

- Scenario 1 - 200 samples divided into 4 clusters ($k = 4$) with 20 informative features and 80 non-informative.
- Scenario 2 – same as scenario 1 with 2 clusters ($k = 2$).
- Scenario 3 – the simulation suggested in [26]: 200 samples, 4 clusters ($k = 4$), 50 informative features, 950 non informative features.

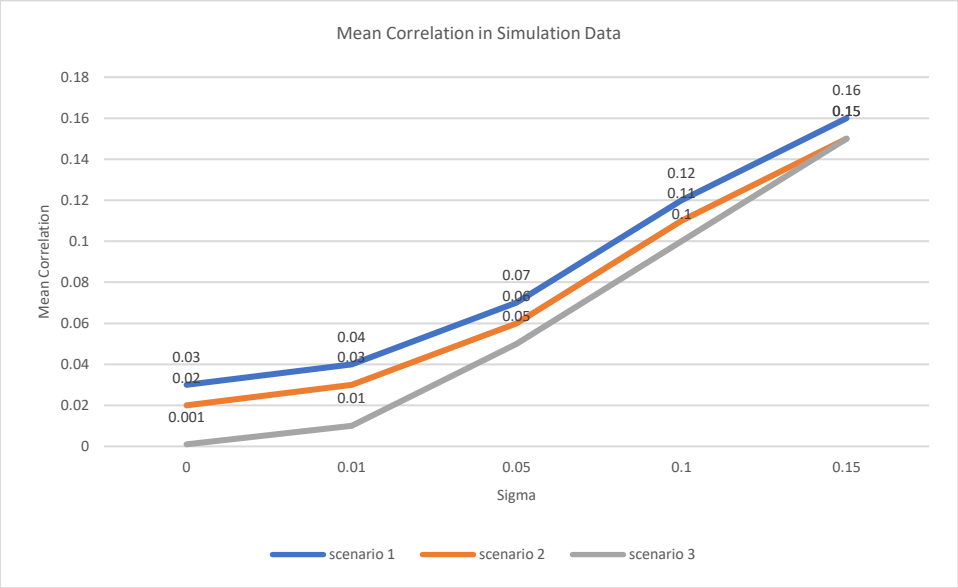


Figure S8. Testing the average Pearson correlation (X axis) in the data in different simulation scenarios, with different values of σ (Y axis). Description of the scenarios is found above.

For $\mu = \{1,4\}$ we received nearly identical results (not shown).

Next, we tested the average Pearson correlation in real data sets. The four genomic cancer datasets as used in [49] and presented in the Results section (Datasets 1-4) have an average Pearson correlation of 0.07 with STD of 0.03. The value of σ that achieved similar correlation in the data is 0.05, so other than the simple case of $\sigma = 0$ we desided to test also $\sigma = 0.05$, and $\sigma = 0.2$ as a case of extreme correlation. We also tested the highly unrealistic $\sigma = 0.5$ since it was used in [26].

10. Supplementary 4 – simulation results

Simulated data with 200 samples and 100 features of which 20 are informative, divided into two or four equal-sized clusters, and tested $\mu = \{0.5,1,2,4\}$ and $\sigma = \{0,0.05,0.2,0.5\}$, with and without z-score normalization. Results of the accurate recognition rate, supplement to Table 2.

$k = 4$, normalized	$\mu = 2$ $\sigma = 0$	$\mu = 2$ $\sigma = 0.05$	$\mu = 2$ $\sigma = 0.2$	$\mu = 2$ $\sigma = 0.5$	$\mu = 4$ $\sigma = 0$	$\mu = 4$ $\sigma = 0.05$	$\mu = 4$ $\sigma = 0.2$	$\mu = 4$ $\sigma = 0.5$
FRIGATE	1 ± 0	1 ± 0	1 ± 0.02	0.06 ± 0.1	1 ± 0	1 ± 0	1 ± 0	0.1 ± 0.12
FRIGATE-MW	1 ± 0	1 ± 0	1 ± 0	0 ± 0	1 ± 0	1 ± 0	1 ± 0	0.3 ± 0.46
FRCM	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0
FRSD	0 ± 0	0 ± 0	0.14 ± 0.1	0.66 ± 0.12	0.05 ± 0.06	0.05 ± 0.04	0.25 ± 0.1	0.66 ± 0.11
FRMV	0 ± 0	0 ± 0	0.04 ± 0.11	0.43 ± 0.3	0 ± 0	0 ± 0	0 ± 0	0.24 ± 0.31

Table 6. Performance on simulated data, with $k = 4$ and z-score normalized data.

$k = 4$, Non-normalized	$\mu = 2$ $\sigma = 0$	$\mu = 2$ $\sigma = 0.05$	$\mu = 2$ $\sigma = 0.2$	$\mu = 2$ $\sigma = 0.5$	$\mu = 4$ $\sigma = 0$	$\mu = 4$ $\sigma = 0.05$	$\mu = 4$ $\sigma = 0.2$	$\mu = 4$ $\sigma = 0.5$
FRIGATE	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0
FRIGATE-MW	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0
FRCM	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0
FRSD	1 ± 0	1 ± 0.02	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0
FRMV	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0

Table 7. Performance on simulated data, with $k = 4$ and non-normalized data.

$k = 2$, normalized	$\mu = 0.5$ $\sigma = 0$	$\mu = 0.5$ $\sigma = 0.05$	$\mu = 0.5$ $\sigma = 0.2$	$\mu = 0.5$ $\sigma = 0.5$	$\mu = 1$ $\sigma = 0$	$\mu = 1$ $\sigma = 0.05$	$\mu = 1$ $\sigma = 0.2$	$\mu = 1$ $\sigma = 0.5$
FRIGATE	0.34 ± 0.09	0.39 ± 0.11	0.29 ± 0.09	0.15 ± 0.07	0.96 ± 0.04	0.89 ± 0.06	0.46 ± 0.11	0.16 ± 0.11
FRIGATE-MW	0.38 ± 0.11	0.37 ± 0.18	0.22 ± 0.1	0.14 ± 0.1	1 ± 0.02	1 ± 0	0.56 ± 0.22	0.15 ± 0.2
FRCM	0.49 ± 0.08	0.42 ± 0.11	0.23 ± 0.08	0.14 ± 0.08	1 ± 0.02	0.98 ± 0.02	0.66 ± 0.11	0.29 ± 0.16
FRSD	0.17 ± 0.07	0.15 ± 0.05	0.23 ± 0.08	0.26 ± 0.08	0.09 ± 0.05	0.12 ± 0.07	0.15 ± 0.08	0.37 ± 0.1
FRMV	0.19 ± 0.11	0.19 ± 0.07	0.18 ± 0.07	0.22 ± 0.12	0.1 ± 0.12	0.14 ± 0.13	0.12 ± 0.09	0.26 ± 0.19

Table 8. Performance on simulated data, with $k = 2$ and z-score normalized data.

$k = 2$, normalized	$\mu = 2$ $\sigma = 0$	$\mu = 2$ $\sigma = 0.05$	$\mu = 2$ $\sigma = 0.2$	$\mu = 2$ $\sigma = 0.5$	$\mu = 4$ $\sigma = 0$	$\mu = 4$ $\sigma = 0.05$	$\mu = 4$ $\sigma = 0.2$	$\mu = 4$ $\sigma = 0.5$
FRIGATE	1 ± 0	1 ± 0	0.94 ± 0.07	0.11 ± 0.14	1 ± 0	1 ± 0	1 ± 0	0.21 ± 0.17
FRIGATE-MW	1 ± 0	1 ± 0	1 ± 0	0.24 ± 0.39	1 ± 0	1 ± 0	1 ± 0	0.1 ± 0.3
FRCM	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0
FRSD	0 ± 0	0 ± 0	0.07 ± 0.07	0.55 ± 0.1	0.26 ± 0.07	0.28 ± 0.07	0.35 ± 0.11	0.37 ± 0.08
FRMV	0.25 ± 0.27	0.19 ± 0.25	0.06 ± 0.15	0.17 ± 0.16	0.21 ± 0.27	0.09 ± 0.15	0.01 ± 0.02	0.24 ± 0.32

Table 9. Performance on simulated data, with $k = 2$ and z-score normalized data.

$k=2$, Non-normalized	$\mu=0.5$ $\sigma=0$	$\mu=0.5$ $\sigma=0.05$	$\mu=0.5$ $\sigma=0.2$	$\mu=0.5$ $\sigma=0.5$	$\mu=1$ $\sigma=0$	$\mu=1$ $\sigma=0.05$	$\mu=1$ $\sigma=0.2$	$\mu=1$ $\sigma=0.5$
FRIGATE	0.41 ± 0.1	0.37 ± 0.1	0.27 ± 0.11	0.3 ± 0.12	0.96 ± 0.04	0.93 ± 0.04	0.82 ± 0.09	0.54 ± 0.14
FRIGATE-MW	0.45 ± 0.12	0.43 ± 0.13	0.33 ± 0.11	0.32 ± 0.11	0.99 ± 0.02	1 ± 0.02	0.89 ± 0.18	0.79 ± 0.13
FRCM	0.51 ± 0.15	0.38 ± 0.11	0.25 ± 0.07	0.19 ± 0.06	1 ± 0	0.99 ± 0.02	0.75 ± 0.14	0.42 ± 0.1
FRSD	0.32 ± 0.06	0.36 ± 0.08	0.32 ± 0.1	0.37 ± 0.1	0.74 ± 0.08	0.72 ± 0.09	0.73 ± 0.04	0.8 ± 0.07
FRMV	0.18 ± 0.09	0.22 ± 0.12	0.21 ± 0.09	0.2 ± 0.09	0.14 ± 0.17	0.18 ± 0.25	0.28 ± 0.2	0.23 ± 0.17

Table 10. Performance on simulated data, with $k = 2$ and non-normalized data.

$k=2$, Non-normalized	$\mu=2$ $\sigma=0$	$\mu=2$ $\sigma=0.05$	$\mu=2$ $\sigma=0.2$	$\mu=2$ $\sigma=0.5$	$\mu=4$ $\sigma=0$	$\mu=4$ $\sigma=0.05$	$\mu=4$ $\sigma=0.2$	$\mu=4$ $\sigma=0.5$
FRIGATE	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0
FRIGATE-MW	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0
FRCM	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0
FRSD	0.98 ± 0.02	0.99 ± 0.02	0.99 ± 0.02	1 ± 0.02	1 ± 0	1 ± 0	1 ± 0	1 ± 0
FRMV	0.12 ± 0.21	0.33 ± 0.32	0.05 ± 0.1	0.19 ± 0.27	0 ± 0	0 ± 0	0 ± 0	0 ± 0

Table 11. Performance on simulated data, with $k = 2$ and non-normalized data.

11. Supplementary 5 – Significance levels

P-values for the differences in performances of FRIGATE, FRIGATE-MW, FRMV, FRCM and FRSD of Datasets 5-7 in Table 3. The # column shows the ranking of ARI scores where 1 is best. See Table 4 for the scores. In case of ties in the ARI we used the STD as a tiebreaker, where lower STD accounts for a better rank.

	#	FRIGATE-MW	FRMV	FRSD	FRCM
FRIGATE	2	0.256	4.66e-06	2.164e-07	3.764e-08
FRIGATE-MW	1	-	3.763e-09	1.308e-14	1.066e-15
FRMV	4	-	-	0.856	0.328
FRSD	3	-	-	-	3.611e-07
FRCM	5	-	-	-	-

Table S12. T-test p-values of ARI scores for $\lceil 25\% \rceil$ features of Dataset 5.

	#	FRIGATE-MW	FRMV	FRSD	FRCM
FRIGATE	2	0.66	2.117e-12	1.771e-11	0.0961
FRIGATE-MW	1	-	2.031e-12	1.65e-11	0.0451
FRMV	5	-	-	0.029	5.446e-12
FRSD	4	-	-	-	5.575e-11
FRCM	3	-	-	-	-

Table S13. T-test p-values of ARI scores for $\lceil 50\% \rceil$ features of Dataset 5.

	#	FRIGATE-MW	FRMV	FRSD	FRCM
FRIGATE	3	0.648	0.987	0.002	0.595
FRIGATE-MW	2	-	0.588	4.564e-05	1
FRMV	4	-	-	1.836e-04	0.504
FRSD	5	-	-	-	4.218e-08
FRCM	1	-	-	-	-

Table S14. T-test p-values of ARI scores for $\lceil 25\% \rceil$ features of Dataset 6.

	#	FRIGATE-MW	FRMV	FRSD	FRCM
FRIGATE	2	0.231	0.607	0.021	0.127
FRIGATE-MW	4	-	0.653	0.052	3.07e-04
FRMV	3	-	-	0.057	0.052
FRSD	5	-	-	-	0.001
FRCM	1	-	-	-	-

Table S15. T-test p-values of ARI scores for $\lceil 50\% \rceil$ features of Dataset 6.

	#	FRIGATE-MW	FRMV	FRSD	FRCM
FRIGATE	3	0.712	0.871	2.09e-05	0.241
FRIGATE-MW	2	-	0.209	4.39e-11	0.011
FRMV	4	-	-	4.173e-10	4.72e-04
FRSD	5	-	-	-	1.073e-13
FRCM	1	-	-	-	-

Table S16. T-test p-values of ARI scores for $\lceil 25\% \rceil$ features of Dataset 7.

	#	FRIGATE-MW	FRMV	FRSD	FRCM
FRIGATE	1	0.177	0.018	1.232e-23	0.278
FRIGATE-MW	3	-	0.641	1.809e-15	0.76
FRMV	4	-	-	6.575e-18	0.394
FRSD	5	-	-	-	1.918e-16
FRCM	2	-	-	-	-

Table S17. T-test p-values of ARI scores for $\lceil 50\% \rceil$ features of Dataset 7.

	#	FRIGATE-MW	FRMV	FRSD	FRCM
FRIGATE	2	0.526	1.031e-07	0.142	1.572e-18
FRIGATE-MW	3	-	1.66e-06	0.165	5.947e-12
FRMV	4	-	-	3.059e-08	0.468
FRSD	1	-	-	-	2.904e-22
FRCM	5	-	-	-	-

Table S18. T-test p-values of ARI scores for $\lceil 25\% \rceil$ features of Dataset 8.

	#	FRIGATE-MW	FRMV	FRSD	FRCM
FRIGATE	1	0.15	3.753e-05	0.898	3.125e-04
FRIGATE-MW	3	-	7.479e-05	0.095	0.006
FRMV	5	-	-	3.53e-05	2.41e-04
FRSD	2	-	-	-	1.076e-05
FRCM	4	-	-	-	-

Table S19. T-test p-values of ARI scores for $\lceil 50\% \rceil$ features of Dataset 8.

12. Supplementary 6 - Clinical significance

Full results of the ranked features by FRIGATE and FRIGATE-MW that were also top features found in [54]. See chapter 4.2 for details.

Rank	Feature	Mendelson Cohen et al.
1	days in hospital	☒
2	Calculated Bicarbonate, Whole Blood	☒
3	Hemoglobin	☑
4	Hematocrit	☑
5	Platelet Count	☑
6	MCH	☒
7	Alanine Aminotransferase (ALT)	☑
8	Asparate Aminotransferase (AST)	☒
9	Bilirubin, Total	☑
10	Red Blood Cells	☑
11	Fibrinogen, Functional	☒
12	MCHC	☒
13	Calculated Total CO2	☒
14	Anion Gap	☒
15	MCV	☒
16	Bicarbonate	☒
17	Alkaline Phosphatase	☑
18	Age	☒
19	pCO2	☒
20	Neutrophils	☒
21	Lymphocytes	☒
22	pH	☒
23	Base Excess	☒
24	PT	☒
25	Glucose	☒
26	INR(PT)	☒
27	Chloride	☒
28	pO2	☒
29	RDW	☒
30	Sodium, Whole Blood	☒
31	pH	☒
32	Ethanol	☒
33	Specific Gravity	☒
34	Creatinine	☑
35	Phosphate	☒
36	White Blood Cells	☒
37	Basophils	☑
38	Potassium, Whole Blood	☒
39	Urea Nitrogen	☑
40	Monocytes	☑
41	Calcium, Total	☒
42	Lactate	☒
43	Eosinophils	☑
44	PTT	☒
45	Magnesium	☒
46	Amylase	☒
47	Lipase	☒

Table S20. Average ranks of FRIGATE-MW for Dataset 6. The bold line represents $\lceil 25\% \rceil$ of features cutoff. ☑: Top features in [54], ☒: The rest.

Rank	Feature	Mendelson Cohen et al.
1	days in hospital	☒
2	Platelet Count	☑
3	Calculated Bicarbonate, Whole Blood	☒
4	Hematocrit	☑
5	pH	☒
6	MCH	☒
7	Age	☒
8	Hemoglobin	☑
9	Neutrophils	☒
10	Red Blood Cells	☑
11	INR(PT)	☒
12	Bilirubin, Total	☑
13	Alanine Aminotransferase (ALT)	☑
14	Asparate Aminotransferase (AST)	☒
15	pH	☒
16	Lymphocytes	☒
17	Fibrinogen, Functional	☒
18	MCHC	☒
19	MCV	☒
20	Anion Gap	☒
21	Calculated Total CO2	☒
22	Alkaline Phosphatase	☑
23	RDW	☒
24	PT	☒
25	Glucose	☒
26	Specific Gravity	☒
27	Urea Nitrogen	☑
28	Bicarbonate	☒
29	pCO2	☒
30	Basophils	☑
31	Base Excess	☒
32	Ethanol	☒
33	White Blood Cells	☒
34	Chloride	☒
35	Creatinine	☑
36	pO2	☒
37	Sodium, Whole Blood	☒
38	Potassium, Whole Blood	☒
39	Monocytes	☑
40	Eosinophils	☑
41	Phosphate	☒
42	PTT	☒
43	Calcium, Total	☒
44	Magnesium	☒
45	Lactate	☒
46	Amylase	☒
47	Lipase	☒

Table S21. Average ranks of FRIGATE for Dataset 6. The bold line represents $\lceil 25\% \rceil$ of features cutoff. ☑: Top features in [54], ☒: The rest.

13. Supplementary 7 – The effect of k in each algorithm

FRIGATE requires determining the number of clusters k in advance, unlike FRCM and FRSD, which test a prescribed range of k. Generally, it is preferred not to determine the number of clusters in advance. However, all the previously suggested ensemble algorithms are based on k-means, which requires k as an input. As it is commonly done, we suggest trying different k values and choosing the value using some criterion, e.g., the "elbow" method [41]. This is the approach we use for FRIGATE.

To test the effect of k in prior algorithms, we simulated data with four clusters ($k = 4$), fifty samples in each cluster ($c = 50$), and 100 features, of which 20 are informative ($\alpha = 20$, $\beta = 80$), when the data is z-score normalized, and measures the accurate recognition rate. We repeated the run 10 times for each case and present in Figure S9 the mean results. We ran FRCM and FRSD with a range $[min\_k, max\_k]$ where min_k of {2,3,4}, and $max\_k = \{4,8,12,15\}$, (recall that for FRCM and FRSD the default range is [2:15] as $k\_max = \min(\lceil\sqrt{200}\rceil, 20) = 15$). We can see that generally an increase of max_k results in a sharp decrease in the results of FRSD. Interestingly, using $min\_k < k$ had a positive effect: using a range of [2:4], when $k = 4$, produced better results than using [3,4] or [4,4]. Similar results were obtained for simulation with $\mu = 4$ (Figure S11).

It is worth mentioning that in FRSD the larger the range - the more iterations the algorithm performs in total ($T = 200$ for each tested value of k). To see if the total number of iterations has a major effect on the results, we also tested the performance of FRSD when the range $[k, k]$ was given as input for different k values, and the total number of iterations per k was set to 2800 (the number of iterations for the full default range of k), and the results are shown in Figure S10 for $\mu = 2$ (see Figure S12 for $\mu = 4$). We can see that FRSD performs best for k=3 and k=2, although the real number of clusters is four. Higher values of k harm the ability of the algorithm to find any relevant features. This helps explaining the results in Figure S9, where we see a drop in performance when large values of k are included and affecting the results.

FRCM, on the other hand, has a fixed number of iterations and samples a value of k for each iteration from the same range as FRSD. Interestingly, when performing the same analyses for FRCM, the algorithm produced perfect results for all cases (results not shown).

Unlike FRSD and FRCM, FRIGATE and FRMV require k as an input. Hence, we tested how different values of input k affect the results of FRIGATE and FRMV on simulated data with the same parameters used for FRSD and FRCM. We tested the algorithms in the same simulation, using as input values of k {2, 3, 4, 8,

12, 15}. The results of FRIAGTE were perfect in all scenarios, which means it was unaffected by the input k. FRMV had poor results in all scenarios (results not shown).

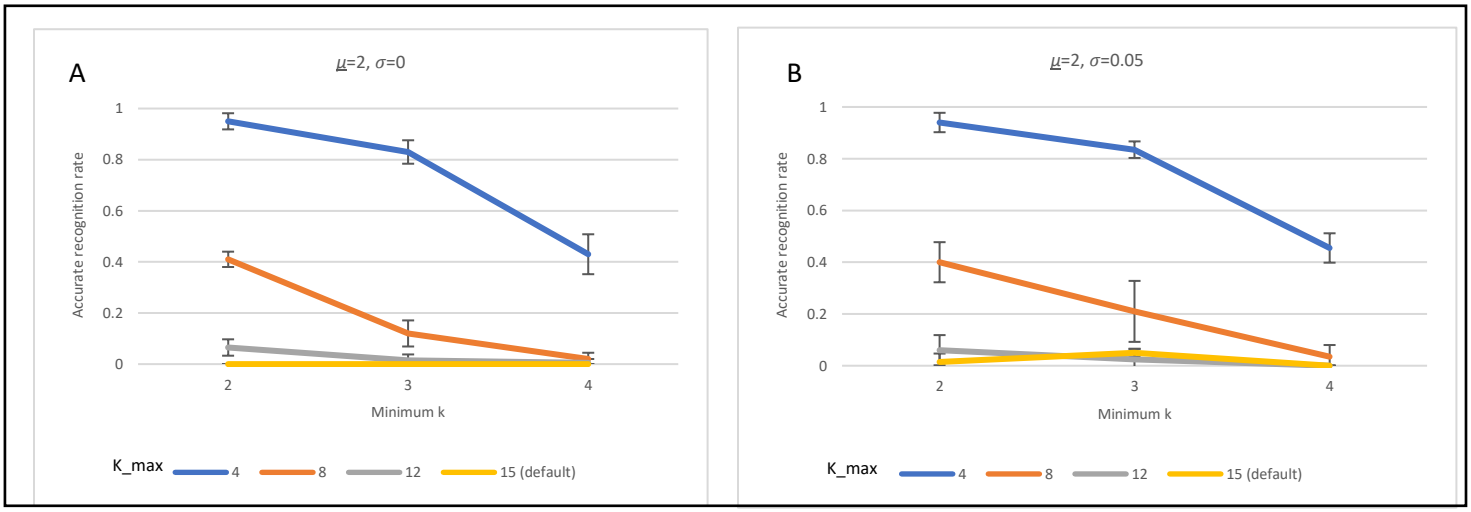


Figure S9. Performance of FRSD with different ranges for $[k\_min, k\_max]$. The simulation parameters are 4 clusters, 50 samples in each cluster, 100 features - of which 20 are informative, $\mu = 2$. The results show accurate recognition rate of FRSD over 10 runs. A: $\sigma = 0$, B: $\sigma = 0.05$.

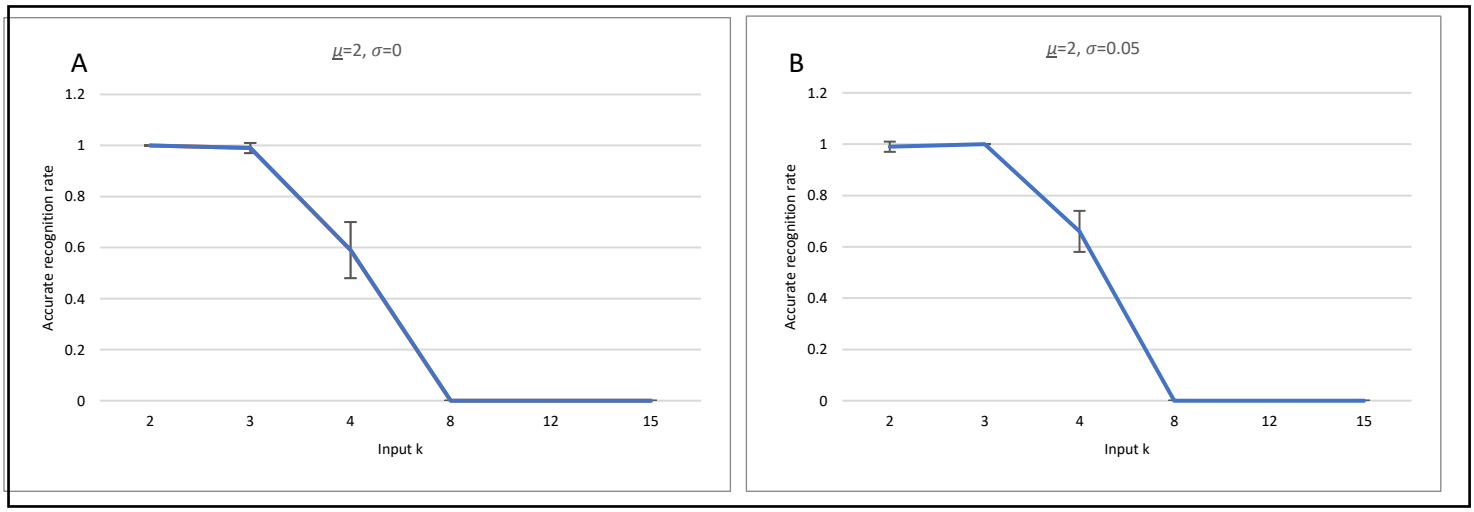


Figure S10. Performance of FRSD when a single number of clusters is tested, the true number is 4, and 2800 iterations are performed per each number of clusters tested. The rest of the simulation parameters are as in Figure 7. A: $\sigma = 0$, B: $\sigma = 0.05$.

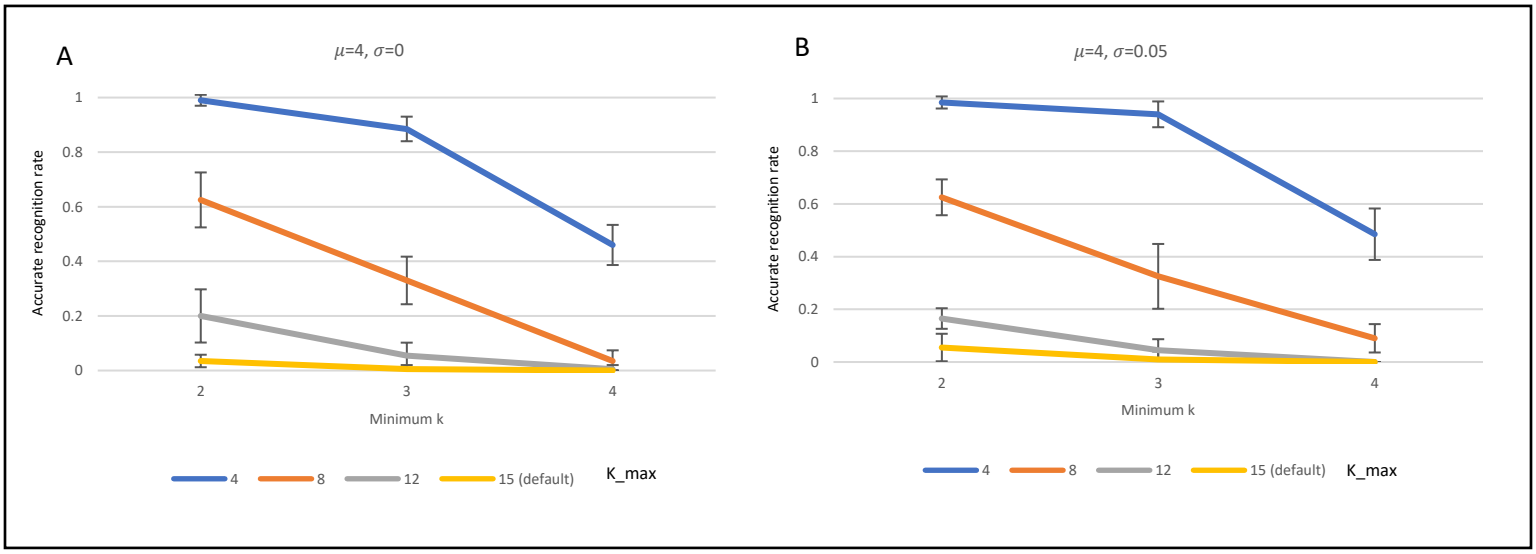


Figure S11. Performance of FRSD with different values used as max_x. The simulation has 4 clusters, 50 samples in each cluster, 100 features - 20 of which are informative for clustering, $\mu = 4$ and $\sigma = 0$ or $\sigma = 0.05$ for A and B respectively. We measured the accurate recognition rate and we present here the mean of 10 runs of FRSD on the range $[min\_k, max\_k]$ for different values of min_x and max_x.

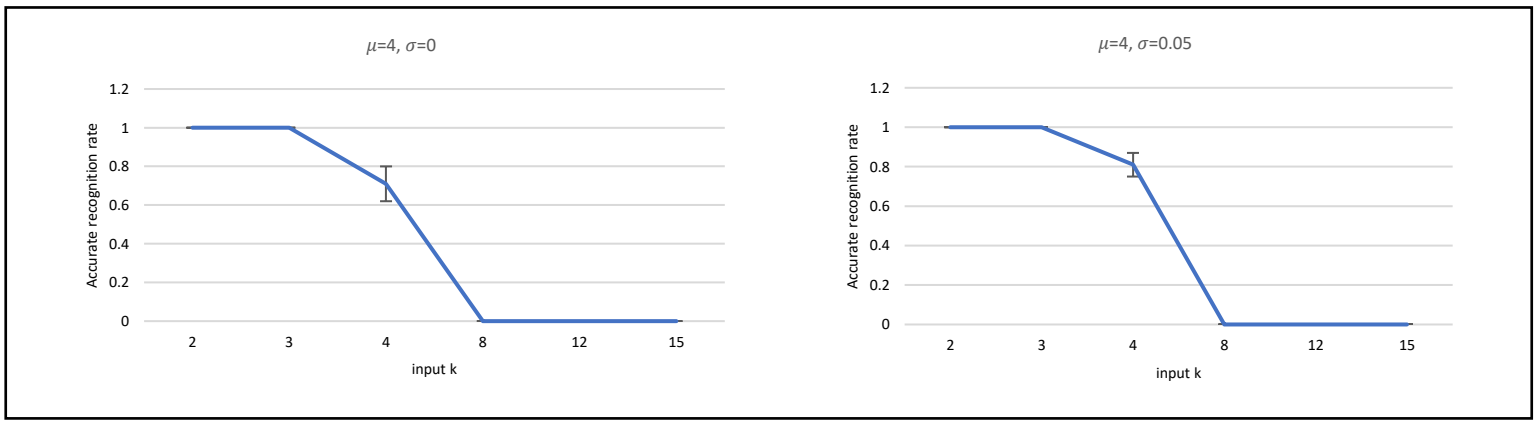


Figure S12. Performance of FRSD with a single value of k and 2800 iterations. The simulation parameters are as in Figure S6. A: $\sigma = 0$, B: $\sigma = 0.05$.

תקציר

הזמינות של נתוני רשומות רפואיות ממוחשבות השתפרה באופן ניכר בשנים האחרונות, ומספר הולך וגדל של מחקרים עושים שימוש באלגוריתמיקה ולמידת מכונה על נתונים מסוג זה. שיטה מרכזית בתחום זה היא קיבוץ (clustering), בה נעשה שימוש למטרות שונות, כולל זיהוי תתי סוגים חדשים של מחלות מוכרות. השפע והיתירות של המידע הקיים ברשומות הרפואיות מעלה את הצורך לזהות את המשתנים המשמעותיים ביותר לפעולת הקיבוץ ולדרגם. זיהוי המשתנים החשובים ביותר משמעותי גם משיקולי זמן ריצה וחיסכון בזיכרון, וגם לשם שיפור יכולת ההבנה ופירוש התוצאות.

אנחנו מציגים כאן את FRIGATE (Feature Ranking In clustering using GAme ThEory), אלגוריתם אשר מדרג את המשתנים בהתאם לחשיבתם לקיבוץ, ומשלב מושגים מתורת המשחקים: ערכי שפלי (Shapley) ומשקולות כפליים (MW ,Multiplicative Weights). FRIGATE הוא אלגוריתם דירוג משתנים בשיטת המכלול (ensemble feature ranking algorithm) אשר קובע את חשיבות המשתנים על בסיס פתרונות קיבוץ רבים על תתי-קבוצות של משתנים. עבור כל פתרון קיבוץ, קבוצה מצומצמת של משתנים מדורגת באופן דומה לחישוב ערכי שפלי, ונעשה שימוש ב-MW במטרה לייעל את הבחירה של תת-הקבוצה הבאה של המשתנים. נראה שזהו אלגוריתם דירוג המשתנים הראשון שמאפשר ניתוח של נתונים רציפים ובדידים יחד, שהוא צורך ממשי בניתוח מידע רפואי. FRIGATE הראה ביצועים טובים יותר הן מבחינת איכות התוצאות והן מבחינת זמן ריצה, בהשוואה לאלגוריתמי דירוג בשיטת המכלול שהוצגו בעבר, הן על קלטים של נתוני סימולציה והן על נתוני אמת רפואיים.

אוניברסיטת תל אביב

הפקולטה למדעים מדוייקים ע"ש ריימונד ובברלי סאקלר

בית הספר למדעי המחשב ע"ש בלווטניק

אלגוריתם דירוג משתנים עבור קיבוץ של נתונים רפואיים

חיבור זה הוגש כעבודת גמר לתואר 'מוסמך אוניברסיטה' באוניברסיטת תל אביב

בבית הספר למדעי המחשב

על ידי

ערן שפיגלמן

בהנחיית

פרופסור רון שמיר

פברואר 2024