Parallel feature subset selection wrappers using k-means classifier

NIKOLAOS PAPAIOANNOU[1], ALKIVIADIS TSIMPIRIS[1], CHRISTOS TALAGOZIS[1], LEONIDAS FRAGIDIS[2], ATHANASIOS ANGEIOPLASTIS[1], SOTIRIOS TSAKIRIDIS[1], DIMITRIOS VARSAMIS[1]

[1]Department of Computer, Informatics and Telecommunications Engineering, International Hellenic University, Serres, GREECE
[2]Department of Management Science and Technology, International Hellenic University, Kavala, GREECE

Abstract: - In a world where the volume of data is constantly increasing, the implementation time of various processes increases significantly. Therefore, the proper management and the effort to reduce the dimensions of the datasets are considered imperative. Feature selection can reduce the size of the datasets by keeping a smaller subset, while improving the accuracy of the classification. The main purpose of this paper is to propose and examine the efficiency of parallel feature selection wrappers based on k-means classifier. The simple k-means algorithm and a parallel version of it are used. Different parallelization variants of feature subset selection (fss) are presented and their accuracy and computation time are also evaluated on four different datasets. The comparison is performed among different parallelization variations and the serial implementation of fss with the k-means clustering algorithm. Finally, the results of the research are presented, highlighting the importance of parallelization in reducing the execution time of the proposed algorithms.

Key-Words: - Classification, Feature selection, k-means, Parallelization, Wrappers

Received: May 9, 2022. Revised: January 17, 2023. Accepted: February 11, 2023. Published: March 9, 2023.

1 Introduction

Nowadays, the amount of data is constantly increasing. Considering big data, the evolution of technology and the internet, it is perceptible that the amount of data is growing rapidly year by year, as well as the diversity of the sources which produce it. Consequently, there are systems and problems that contain thousands of variables, which are likely to lead to millions, or even billions of relationships with each other. Furthermore, management and processing this large amount of data, turns into a challenge for data mining researchers and engineers.

Data mining is the process of extracting useful information, such as dependencies and patterns from large amounts of data. More specifically, it helps to draw useful conclusions and information from seemingly unrelated data. Data mining consists of a set of important steps, such as data cleaning and formatting, database creation and maintenance, data visualization, using algorithms and features selection. Various methods are used to draw conclusions from the data mining process. Some of these methods are related to association, prediction, classification, clustering analysis, decision trees and neural networks, [1]. Although all these methods are extremely powerful and useful tools in the process of data mining, it is very important to correctly distinguish which method will be chosen to be used, depending on the specificity of each problem and the information that is desired to be extracted.

Clustering is an unsupervised method of pattern recognition used in machine learning. The main goal of this process is to better understand the underlying information and structure of the dataset, depending on the similarities between the data. This goal is accomplished by dividing the data into smaller subsets (classes) according to the similarity of those belonging to the same class and the differences from those belonging to the other classes. Deep insights of the dataset is not the primary concern of clustering, but it emphasizes of data classification and considering that it is an unsupervised method, it turns it into a very useful preparatory stage before a deeper further analysis.

One of the most common clustering algorithms which is widely used is k-means, [2]. k-means is an unsupervised learning algorithm that defines the computer how to group different objects based on their attributes-features. Like any other algorithm,

k-means has its pros and cons. From one point of view, it is easy to implement and to define subsets from the original complex datasets without any other preprocess. Most of the times it was found out that k means algorithm is faster than hierarchical clustering and also very efficient in terms of computational cost which is O(K*n*d) for the first iteration (assignment) and O(n*d) while updating the centroids, [3]. On the other hand, although k-means and other clustering methods, [4], are great at initially separating the data into clusters, it depends heavily on the location of the points, rather than the actual relation of the data itself. Furthermore, it can give significantly different results, depending on the number of clusters that are set (k), which increases the importance of the initial choice of the number of clusters(k). Last but not least, it is also sensitive to the outliers.

Outliers and big data in general, could reduce the efficiency and therefore the reliability of the k-means algorithm, [5]. These kinds of datasets may contain a large amount of unrelated and redundant information, which are likely to affect the performance of the classification algorithm. In order to avoid these problems, feature selection methods have been used, [6], [7], [8].

Feature selection is often used as a preparatory stage of machine learning or in conjunction with it, [9]. The selection of the useful features is an important step, in order to increase the efficiency of the clustering and the classification algorithms. Especially, when the analysis concerns problems of classification and/or regression of many observable features, feature selection becomes an important stage of analysis, which can reduce the size of the dataset and the computational cost, by removing irrelevant and unnecessary data like outliers, but also by improving accuracy, as well as understanding deeper the structure of the model or the data, [10]. Therefore, feature selection is considered necessary when dealing with big data cases. The purpose of feature selection algorithms is to create a subset of features from the initial one (the whole dataset), based on some criteria. Feature selection methods are divided into three main categories. Filter methods, wrapper methods and embedded methods. Filter methods depending on the general characteristics of the data, without including any learning algorithm. These methods use some dependency measures such as mutual information, [11], conditional mutual information, [12], Pearson's correlation, [13], in order to examine the relationships between the features and to create the best possibly feature subset. These methods are relatively resistant to overfitting, but they often do not provide the optimal subset of features. However, due to the fact that the computational cost of these methods is much lower compared to the others, it is widely used by researchers when dealing with big data. Wrapper methods, [14], [15], require a predefined learning algorithm and uses its performance to evaluate and determine which features will be selected. Thus, for each new subset of features, the wrapper method needs to satisfy a predefined hypothesis or classifier. It tends to find features that fit better to the predefined learning algorithm, which results in very good learning performance, but at a higher computational cost. Although this method is really efficient and achieves high performance, its computational cost makes it prohibitive for large scale datasets (big data), because the classifier needs to be trained plenty of times. The embedded methods, [16], [17], [18], combine the advantages of both filter and wrapper methods. In this method, the algorithm learns which features contribute best to the accuracy of the model, during the process of the creation of the model. Although they usually have lower computational cost than wrapper methods, they are much slower than filter methods and in addition, the selected features highly depend on the learning machine.

In this paper, a wrapper feature selection method will be used. As it was mentioned earlier, wrapper methods lead to really good learning performances, but at high computational cost. Parallelization can reduce the computational cost, without compromising the efficiency of the methods.Recently, in feature selection, different parallelization techniques have been proposed and have been used for high dimensional cases involving large amount of data, such as cancer classification, [19], and wind forecasting, [20]. In order to overcome the limitations related to the increase of computation time while using feature selection methods, different approaches have been proposed. Such examples are, a novel hybrid parallel implementation that uses MPI and OpenMP for feature selection on distributed-memory clusters, [21], a parallel heterogeneous ensemble feature selection based on multi-core architectures (apply parallelism to multi-core in CPU along with GPU), [22], and a framework QCFS which exploiting the capabilities of modern CPU's, by executing in parallel, on four cores, the feature selection process, [23]. Finally, a feature selection library has been proposed, [24], in order to help with the acceleration of the feature selection process. This library includes seven methods, which follow a hybrid MPI/multithreaded approach. In this paper, different variations of parallelization of feature selection algorithms based on k-means will be examined, in order to reduce computational cost, to quantify the reduction in time resulting from each method and finally, to select the most effective one. The rest of the paper is organized as follows: In chapter 2 basic elements of theory, as well as the methodology followed are presented. In Chapter 3, the Datasets

are presented, in Chapters 4, 5 a simulation study for various datasets and a case study are performed respectively, while chapter 7 presents the conclusions.

2 Methodology

The main purpose of this paper is to compare different parallelized variations of a wrapper feature selection algorithm (FSS) with each other and in serial implementations, both in terms of efficiency and execution time. As classifier, k-means algorithm and its variant are selected, while as evaluation functions two similarity metrics (CRI and Accuracy) are used.

2.1 Classification algorithms

2.1.1 *k-means*

k-means, [25], is a simple unsupervised learning algorithm, which is widely used in clustering problems. Given a fixed a priori number of k clusters, the algorithm sets k centroids, one for each cluster. Then, the data objects of a given dataset are divided into smaller classes, satisfying a minimum distance criterion. Specifically, for each object, the distance with all centroids is calculated and the objects are classified in the centroid class with which the respective object had the minimum distance. Then, by calculating the mean of each class, the new centroids emerge. The process is then repeated, until there are no more changes in the resulting centroids.

2.1.2 *k-meansRA*

k-meansRA clustering method, [26], is similar to the *k-means* clustering method, but with a significant difference at the first step. In contrast with the *k-means* clustering method, which select randomly a defined number of k centroids from the whole dataset and then the whole process is applied, *k-meansRA* assigns the patterns of the dataset randomly to k classes and the centroids are calculated after the initial random assignments. The rest of the process is similar to the simple *k-means* algorithm and the best solution results from the set for which the sum of the distances of the patterns from the respective centroids is minimized. Although the differences between the methods are very small and differ only in the first step, according to [26] the *k-meansRA* method performs better and at a lower computational cost compared to the others.

2.2 Classification efficiency

In order to measure the classification performance, different metrics have been widely used. Accuracy, CRI (corrected rand index), [27], log-loss, [28], and AUC-ROC, [29], are some of the most popular metrics. In this paper, Accuracy and CRI will be used.

2.2.1 CRI

The Corrected Rand Index (CRI) is a standard metric for the classification performance. With this metric, two partitions can be compared and quantify the similarity between them. Let's suppose that

$$\text{CRI} = \frac{\sum_{i=1}^{R}\sum_{j=1}^{C}\binom{k_{ij}}{2} - \binom{K}{2}^{-1}\sum_{i=1}^{R}\binom{k_{i.}}{2}\sum_{j=1}^{C}\binom{k_{.j}}{2}}{\frac{1}{2}\left[\sum_{i=1}^{R}\binom{k_{i.}}{2} + \sum_{j=1}^{C}\binom{k_{.j}}{2}\right] - \binom{K}{2}^{-1}\sum_{i=1}^{R}\binom{k_{i.}}{2}\sum_{j=1}^{C}\binom{k_{.j}}{2}},$$

where k_{ij} is the number of objects in the i-cluster of the first partition and the j-cluster of the second, $k_{i.}$ is the number of objects in the i-cluster of the first partition and $k_{.j}$ is the number of objects in the j-cluster of the second partition. CRI ranges from -1 to 1, where 1 indicates exact agreement of the two partitions, values near zero indicate random agreement and negative values indicate disagreement. In our computations of CRI, the first partition regards the known classes.

The range of the values that CRI can take is from -1 to 1. A CRI value equal to 1 means an absolute match between the two partitions in terms of classification, while -1 indicates a complete disagreement. Values close to zero show randomness.

2.2.2 Accuracy

Accuracy is a widely used metric for evaluating classification performance. It is a really simple metric and specifically it is the ratio of number of the correct predictions to the total number of samples.

$$accuracy = \frac{C}{T}$$

where C is the number of correct predictions and T is the total number of the initial samples. For binary classification, it can easily be transformed to a fraction from confusion matrix terms

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

where:

- TP (True Positive) represents the number of true positive instances,
- FP (False Positive) represents the number of false positive instances,
- FN (False Negative) represents the number of false negative instances and
- TN (True Negative) represents the number of true negative instances.

2.3 FSS

In order to reduce the dimensionality and increase the efficiency of the clustering, a wrapper feature selection is used. Forward Sequential Selection (FSS), [30], [31], is a widely used algorithm that can reduce the process time. The subset of features is built in progressively, adding one feature to each "circle". First, the feature which has the maximum similarity with the class is found, based on the evaluation function (CRI or Accuracy). Then, in each circle, the feature for which the evaluation function value is better from both all the other features and of the maximum value of the previous circle is added. The algorithm stops when the termination criterion is satisfied, given a predefined threshold. When the value of the evaluation function does not increase enough with the addition of a new feature, or even more, if it reduces its performance, then the algorithm stops and as an optimal feature subset is selected the one from the previous circle of the algorithm. The FSS algorithm reads as follows:

1. Set $l = 1$ and find the best single feature clustering, i.e., $S_l = \{q_{(l)}\}$, where $q_{(l)} = \arg\max_{q_i} \text{CRI}(\{q_i\})$ and $i = 1, \ldots, d$.
2. For $l > 1$, compute clusterings for the feature subsets $S_l^i = S_{l-1} \cup q_i$, where $q_i \notin S_{l-1}$ and find the one with the largest CRI, i.e., $q_{(l)} = \arg\max_{q_i \notin S_{l-1}} \text{CRI}(S_l^i)$.
3. If $T = \dfrac{\text{CRI}(S_l^{(l)}) - \text{CRI}(S_{l-1})}{\text{CRI}(S_{l-1})} > threshold$ (here 0.001), then $S_l = S_l^{(l)}$, $l = l + 1$, and go to step 2, otherwise stop.

2.4 Parallelization

To reduce the time of the aforementioned process, parallelization variations are presented and performed, using different parallelization functions of matlab such as *parfor* and *spmd*.The aim is to find which parallelization method (depending on which part of the algorithm it was applied to) is the most efficient compared to others, in terms of lower computational cost. The parallelization techniques are performed both to the feature selection process and also to the *k-means* classifier.

2.4.1 MATLAB parallelization functions

In this paper the *parfor* and the *spmd* of Matlab commands will be used. *Spmd* is a parallel region, while *parfor* is a parallel for loop. The *spmd* command offers more flexibility, because it can be used both in loops as well as it can operate on distributed arrays and vectors. *Spmd* allows a piece of code to be defined and run simultaneously on multiple workers, while *parfor* is a parallelization technique for loops with several restrictions.

2.4.2 Parallelization variations

• **fssPkmeansRA** In this variant, the entire FSS process is parallelized using the *parfor* matlab command and as classifier, the serial *k-meansRA* clustering algorithm is used.

1. Set $l = 1$, parallelize the loop and find the best single feature clustering, using the serial *k-meansRA* as classifier, i.e., $S_l = \{q_{(l)}\}$, where $q_{(l)} = \arg\max_{q_i} \text{CRI}(\{q_i\})$ and $i = 1, \ldots, d$.
2. For $l > 1$, parallelize the loop and compute clusterings using *k-meansRA* as classifier for the feature subsets $S_l^i = S_{l-1} \cup q_i$, where $q_i \notin S_{l-1}$ and find the one with the largest CRI, i.e., $q_{(l)} = \arg\max_{q_i \notin S_{l-1}} \text{CRI}(S_l^i)$.
3. If $T = \dfrac{\text{CRI}(S_l^{(l)}) - \text{CRI}(S_{l-1})}{\text{CRI}(S_{l-1})} > threshold$ (here 0.001), then $S_l = S_l^{(l)}$, $l = l + 1$, and go to step 2, otherwise stop.

• **fsskmeansRAP** As fsskmeansRAP, the serial FSS algorithm is mentioned, which uses as classifier the parallelized *k-meansRA* with the *parfor* command.

1. Set $l = 1$. By using the serial loop, parallelize the *k-meansRA* classifier (here, with the parfor command for MATLAB) and find the best single feature clustering, i.e., $S_l = \{q_{(l)}\}$, where $q_{(l)} = \arg\max_{q_i} \text{CRI}(\{q_i\})$ and $i = 1, \ldots, d$.
2. For $l > 1$, use the serial loop and compute clusterings using parallel *k-meansRA* as classifier (here, with the *parfor* command for MATLAB) for the feature subsets $S_l^i = S_{l-1} \cup q_i$, where $q_i \notin S_{l-1}$ and find the one with the largest CRI, i.e., $q_{(l)} = \arg\max_{q_i \notin S_{l-1}} \text{CRI}(S_l^i)$.
3. If $T = \dfrac{\text{CRI}(S_l^{(l)}) - \text{CRI}(S_{l-1})}{\text{CRI}(S_{l-1})} > threshold$ (here 0.001), then $S_l = S_l^{(l)}$, $l = l + 1$, and go to step 2, otherwise stop.

• **fsskmeansRASPMD** The fsskmeansRASPMD is similar to fsskmeansRAP, but the *spmd* command is used instead of parallelizing the *k-meansRA* clustering algorithm with the *parfor* command.

1. Set $l = 1$. By using the serial loop, parallelize the *k-meansRA* classifier (here, with the *spmd* command for MATLAB) and find the best single feature clustering, i.e., $S_l = \{q_{(l)}\}$, where $q_{(l)} = \arg\max_{q_i} \text{CRI}(\{q_i\})$ and $i = 1, \ldots, d$.

2. For $l > 1$, use the serial loop and compute clusterings using parallel *k-meansRA* as classifier (here, with the *spmd* command for MATLAB) for the feature subsets $S_l^i = S_{l-1} \cup q_i$, where $q_i \notin S_{l-1}$ and find the one with the largest CRI, i.e., $q_{(l)} = \arg\max_{q_i \notin S_{l-1}} \text{CRI}(S_l^i)$.

3. If $T = \dfrac{\text{CRI}(S_l^{(l)}) - \text{CRI}(S_{l-1})}{\text{CRI}(S_{l-1})} > threshold$ (here 0.001), then $S_l = S_l^{(l)}$, $l = l + 1$, and go to step 2, otherwise stop.

• **fsskmeans** fsskmeans is the serial implementation of FSS process, using the simple serial k-means clustering algorithm as classifier, [32]. This variation is used for comparison with the examined parallel variations.

1. Set $l = 1$ and find the best single feature clustering using *k-means* as classifier, i.e., $S_l = \{q_{(l)}\}$, where $q_{(l)} = \arg\max_{q_i} \text{CRI}(\{q_i\})$ and $i = 1, \ldots, d$.

2. For $l > 1$, compute clusterings using *k-means* as classifier for the feature subsets $S_l^i = S_{l-1} \cup q_i$, where $q_i \notin S_{l-1}$ and find the one with the largest CRI, i.e., $q_{(l)} = \arg\max_{q_i \notin S_{l-1}} \text{CRI}(S_l^i)$.

3. If $T = \dfrac{\text{CRI}(S_l^{(l)}) - \text{CRI}(S_{l-1})}{\text{CRI}(S_{l-1})} > threshold$ (here 0.001), then $S_l = S_l^{(l)}$, $l = l + 1$, and go to step 2, otherwise stop.

• **fsskmeansRA** Similar to fsskmeans, fsskmeansRA is the serial implementation of the FSS process, using as classifier a variation of k-means clustering algorithm, the *k-meansRA*.

1. Set $l = 1$ and find the best single feature clustering using *k-meansRA* as classifier, i.e., $S_l = \{q_{(l)}\}$, where $q_{(l)} = \arg\max_{q_i} \text{CRI}(\{q_i\})$ and $i = 1, \ldots, d$.

2. For $l > 1$, compute clusterings using *k-meansRA* as classifier for the feature subsets $S_l^i = S_{l-1} \cup q_i$, where $q_i \notin S_{l-1}$ and find the one with the largest CRI, i.e., $q_{(l)} = \arg\max_{q_i \notin S_{l-1}} \text{CRI}(S_l^i)$.

3. If $T = \dfrac{\text{CRI}(S_l^{(l)}) - \text{CRI}(S_{l-1})}{\text{CRI}(S_{l-1})} > threshold$ (here 0.001), then $S_l = S_l^{(l)}$, $l = l + 1$, and go to step 2, otherwise stop.

3 Dataset

This study is applied to four different datasets. The datasets are derived from regression systems from [12]. The study is performed for binary classification for Datasets 1,2,3 and for five classes for Dataset 4. The variable y gives the classes after discretization. For Datasets 1,2,3, each element of the variable y goes to the first class if its value is below zero and respectively all the other elements whose values are equal to or greater than zero, go to the second class. The features f_i are random variables. The systems are generated for 33,50 and 100 features of 10000 samples each. For Dataset 4 (Mackey Glass differential equation), the system is generated for 365 features of 1000 samples each and concerns the classification to five different classes (regimes).

3.1 Dataset 1

The Dataset 1 resulting from

$$y = \kappa y_1 + (1 - \kappa) y_2,$$

where y_1 and y_2 are:

$$\begin{aligned} y_1 &= \beta_1 f_1 + \beta_2 f_2 + e_1, \\ y_2 &= \beta_3 f_3 + \beta_4 f_4 + \beta_5 f_5 + e_2 \end{aligned}$$

where κ is a weighted parameter ($\kappa = 0.5$), $f_i = 1, ..., 5$, are the predictors of y_1 and y_2 and consequently are relevant features to the class variable C, while e_1 and e_2 are standard normal random variables.The coefficients are $\beta_1 = -3$, $\beta_2 = 2$, $\beta_3 = 3$, $\beta_4 = 2$ and $\beta_5 = -4$. This dataset is generated for a total of 33,50 and 100 features. Beyond the five independent features the rest are irrelevant to C, while they are identically correlated to each other. The features are drawn from a multivariate normal distribution with zero mean and unit variance for each feature component and cross-correlation r = 0.5 for all feature pairs.

3.2 Dataset 2

$$y = \beta_1 f_1 + \beta_2 f_2 + \beta_3 f_3 + \beta_4 f_4 + e,$$

where:

$$\begin{aligned} f_1 &= x_1, \\ f_2 &= x_2, \\ f_3 &= \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3, \\ f_4 &= \alpha_4 x_1^2 + \alpha_5 x_2^2 \end{aligned}$$

where: f_i, $i = 1, ..., 4$ are the predictors of the class variable y. The features (f_i) are related to x_i, $i = 1, 2, 3$ which are random independent standard normal variables. The coefficients are $\alpha_1 = 0.2$, $\alpha_2 = 0.3$, $\alpha_3 = 2$, $\alpha_4 = 0.1$, $\alpha_5 = 0.1$, $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 0.2$, $\beta_4 = 0.3$. This dataset is also generated for a total of 33,50 and 100 features. Beyond the four independent features, the rest are irrelevant to C.

3.3 Dataset 3

$$y = \beta_1 f_1 + \beta_2 f_2 + \beta_3 f_3 + \beta_4 f_4 + \beta_b f_5 + \beta_6 f_6 + e,$$

where:

$$\begin{aligned} f_1 &= x_1, \\ f_2 &= x_2, \\ f_3 &= x_1 x_2, \\ f_4 &= x_3, \\ f_5 &= x_4^2, \\ f_6 &= x_1 x_5 \end{aligned}$$

The class variable y consists of six features f_i, $i = 1, ..., 6$ which are functionally related to x_i, $i = 1, ..., 4$ (random independent standard variables) and x_5 which is independent to the rest of x_i and follows the distribution of the product of two random variables (product distribution). The β_i are setted as $\frac{1}{\sigma_i}$, $i = 1, ..., 6$ where σ_i is the standard deviation of f_i. Consequently, all β_i contribute the same to y.

3.4 Dataset 4

$$\dot{x} = \frac{dx}{dt} = \frac{0.2x(t-\Delta)}{1 + [x(t-\Delta)]^{10}} - 0.1x(t)$$

The dataset 4 is is based on the Mackey-Glass delay differential equation chaotic system. Different measures computed on different regimes, in order to configure the classes to be identified. For Δ equals to 17, the system becomes chaotic and as the Δ increases, so does the complexity. In terms of complexity, two similar settings are considered. One for Δ=120,140,...,200 and the second one for Δ= 110,130,...,190. Each regime is derived for 200 samples of the time series length of 1000. 365 different measures are computed on each time series, which are considered as the features of this dataset. The features have different relations with each other. Some are strongly correlated, while others are irrelevant. For this dataset, the true relevant features for the classification of the five dynamical regimes aren't known. The challenge in this dataset is to select the optimal feature subset that discriminame better the five chaotic regimes.

4 Results

In this paper,three Datasets (1, 2, 3) are generated for 33,50 and 100 features of 10000 samples each, while Dataset 4 generated for 365 different metrics (features) of 1000 samples. Based on these Datasets, different parallelized variations of FSS are compared in terms of classification accuracy and execution time. For Datasets 1, 2 and 3 the binary classification is examined, while for Dataset 4, the study is performed for five classes. During the feature selection process the CRI used as an evaluation function, as it is more robust. When the optimal subset of features has been selected, the performance of the algorithm is evaluated with the accuracy metric. The study is performed for 10 iterations and the results correspond to the average of execution time and accuracy.

The hardware that was used for the evaluation of the algorithms, was a computing system with 12th Gen Intel(R) Core(TM) i7-12700 2.10GHz and 16GB RAM. The operating system was Windows 11 Pro and the SSD was 1000GB.

4.1 Results for Dataset 1

In Table 1 the results of the examined variations for a total of 33,50 and 100 features are illoustrated. The processes repeated for 10 iterations. f refer the selected features, t corresponds to the average execution time, while ac refer to the average values of the Accuracy similarity measure.

Table 1: Results for Dataset 1

Number of Features	33			50			100		
	f	t	ac	f	t	ac	f	t	ac
fsskmeans	5,1	60.61	0.64	5,1	92.51	0.64	5,1	180.63	0.64
fsskmeansRA	5,1	23.43	0.64	5,1	35.37	0.64	5,1	67.62	0.64
fssPkmeansRA	5,1	2.95	0.64	5,1	4.30	0.64	5,1	7.70	0.64
fsskmeansRAP	5,1	26.03	0.64	5,1	36.86	0.64	5,1	76.49	0.64
fsskmeansRASPMD	5,1	31.88	0.64	5,1	47.65	0.64	5,1	92.29	0.64

The results for Dataset 1 (Table 1) suggest that all the examined variations are equally reliable in terms of the selected features and the accuracy. For all the examined sets of features (33,50,100), all methods select features 1 and 5 as the optimal subset. For this Dataset, only the features, f_i, i=1,...,5 are relevant to the class variable, while the rest of the features that are included in the Dataset, are irrelevant to the class variable. The methods are also equal in classification accuracy, as they all achieve about 64%. However, there is a significant difference at the execution time. The fssPkmeansRA is significantly faster than the other examined methods. Specifically, on average, when the initial dataset consists of 33,50 or 100 features, it is respectively 7.9, 8.2 and 8.8 times faster than the second one (fsskmeansRA). Additionally, compared to the simple serial implementation of fss

using the simple k-means as classifier (fsskmeans), the fssPkmeansRA is on average 21.8 times faster. It is worth mentioning that, when comparing the two serial methods, the fsskmeansRA employing *k-meansRA* as classifier, is considerably quicker than the one using the conventional *k-means* as classifier (fsskmeans). In particular, when the Dataset consists of 33 features, fsskmeans needs 60.61 seconds on average, in order to stop the execution of the algorithm, while fsskmeansRA needs only 23.43 seconds. The same seems to apply when the Dataset consists of 50 or 100 features, where fsskmeans needs 92.51 and 180.63 seconds respectively, while fsskmeansRA needs 35.37 and 67.62 seconds respectively. Finally, it is observed that for this dataset, the serial implementation which uses *k-meansRA* as classifier, is faster than the parallelized one with the *spmd* matlab command (fsskmeansRASPMD). In terms of execution time, the parallelized method fsskmeansRASPMD, had the second worst performance, needed on average 57.27 seconds in order to complete the optimal subset.

4.2 Results for Dataset 2

In Table 2 the results of the examined variations for a total of 33,50 and 100 features are illoustrated. The processes repeated for 10 iterations. f refer the selected features, t corresponds to the average execution time, while ac refer to the average values of the Accuracy similarity measure.

Table 2: Results for Dataset 2

Number of Features	33			50			100		
	f	t	ac	f	t	ac	f	t	ac
fsskmeans	2	49.06	0.74	2	70.72	0.74	2	148.11	0.74
fsskmeansRA	2	17.36	0.74	2	24.42	0.74	2	52.30	0.74
fssPkmeansRA	2	2.78	0.74	2	3.29	0.74	2	6.58	0.74
fsskmeansRAP	2	15.54	0.74	2	20.15	0.74	2	42.05	0.74
fsskmeansRASPMD	2	22.18	0.74	2	30.57	0.74	2	63.17	0.74

For Dataset 2 (Table 2), the same conclusions as for Dataset 1 are derived. The Datasets consist of 33, 50 and 100 features. Only the features f_i, i=1,..,4, are relevant to the class variable for this system, while all the other features are irrelevant. The optimal subsets consist of only one feature and specifically the f_2, which is the same for all the examined methods. It is observed that all the methods under study have the same accuracy, which is on average 74%, but their execution time differs significantly (Table 2). The fssPkmeansRA is the fastest one, which needed on average 2.78, 3.29 and 6.58 seconds to complete the optimal subsets, when the Dataset consists of 33, 50 and 100 features respectively. fssPkmeansRA is about 6 times faster than fsskmeansRAP, which is the second fastest, while from the slowest one (fsskmeans), the fssPkmeansRA is about 20 times faster. The serial FSS method which uses the parallelized *k-meansRA* as classifier (fsskmeansRAP), is slightly faster than the corresponding serial one (fsskmeansRA). When the Dataset consists of 33 features, fsskmeansRAP is 1.82 seconds faster than the serial fsskmeansRA. The same seems to apply when the Dataset consists of 50 and 100 features, where fsskmeansRAP is faster than the serial fsskmeansRA, 4.27 and 10.25 seconds respectively. Comparing the two serial methods fsskmeans and fsskmeansRA, similar to Dataset 1, the fsskmeansRA is about 2.82, 2.89 and 2.83 times faster than the serial one (fsskmeans), when the Dataset consists of 33, 50 and 100 features respectively. The serial FSS method, which utilize the parallelized with the *spmd* method *k-meansRA* as classifier, although it is equal to the other methods in terms of accuracy, it is the second worse in terms of execution time, needed 38.64 seconds on average in order to complete the feature selection process.

4.3 Results for Dataset 3

Dataset 3 is considered a little more complicated than the others. Similarly to Datasets 1 and 2, the true relevant features with the class variable are known (f_i, i=1,..,6). It is observed that contrary to Datasets 1 and 2, the optimal feature subsets obtained from the examined methods are different. Specifically, the methods that utilize *k-meansRA* as classifier, select two features for the optimal subset, particularly f_4 and f_2, which are both true relevant features to the class variable (Table 3). On the contrary, the serial FSS using the simple *k-means* as classifier (fsskmeans), selects the feature f_4 for the optimal subset, but also adds the feature f_{21}, which is random normal variable and irrelevant to the class variable y. The accuracy of the methods that use *k-meansRA* as classifier is about 67%, while for fsskmeans is about 64%. Similar to the aforementioned examples (Sections 4.1 and 4.2), the fssPkmeansRA is the fastest method and specifically, it is on average 4.43 times faster than the second fastest (fsskmeansRAP) and 11.4 times faster than the serial implementation of fss, using the simple *k-means* as classifier (fsskmeans). The fsskmeans had the worst performance both in terms of accuracy and execution time.In order to complete the feature selection process, fsskmeans needed 43.73, 65.45 and 150.20 seconds on average, when the Datasets consist of 33, 50 and 100 features respectively. The fsskmeansRAP method which uses the MATLAB *parfor* command in order to parallelize the *k-meansRA* classifier, is shown to be faster than the corresponding one that uses the *spmd* MATLAB command (fsskmeansRASPMD), regardless the size of the Dataset. Worth noting that for this Dataset (but also for Dataset 4 and Case study), the fsskmeans algorithm was adding more features to

the optimal subset, but we stopped it to two, in order to compare the execution time between the algorithms. Specifically, for this Dataset, fsskmeans normally added also the features f_7 and f_{12} to the optimal subset, which are also random normal variables and irrelevant to the class variable y.

In Table 3 the results of the examined variations for a total of 33,50 and 100 features are illoustrated. The processes repeated for 10 iterations. f refer the selected features, t corresponds to the average execution time, while ac refer to the average values of the Accuracy similarity measure.

Table 3: Results for Dataset 3

Number of Features	33			50			100		
	f	t	ac	f	t	ac	f	t	ac
fsskmeans	4,21	43.73	0.64	4,21	65.45	0.64	4,21	150.20	0.64
fsskmeansRA	4,2	24.78	0.67	4,2	36.34	0.67	4,2	89.58	0.67
fssPkmeansRA	4,2	4.87	0.67	4,2	5.55	0.67	4,2	11.28	0.67
fsskmeansRAP	4,2	18.26	0.67	4,2	25.90	0.67	4,2	55.05	0.67
fsskmeansRASPMD	4,2	29.25	0.67	4,2	43.07	0.67	4,2	96.41	0.67

4.4 Results for Dataset 4

For this dataset, the classification problem is to discriminate five highly complex regimes. In Table 4 the results of the examined variations for the Mackey-Glass delay differential equation chaotic system are presented. f refer to the selected features, t corresponds to the average execution time and ac refer to the average values of the similarity measure Accuracy. For each case we consider five different regimes. For Δ_1 case we considered the values of $\Delta = 110, 130, ..., 190$ and for Δ_2 we considered $\Delta = 120, 140, ..., 200$.

Table 4: Results for Dataset 4

Setting	Δ_1			Δ_2		
	f	t	ac	f	t	ac
fsskmeans	54,71	108.81	0.76	54,74	107.58	0.75
fsskmeansRA	54,71	51.53	0.76	54,74	48.57	0.75
fssPkmeansRA	54,71	7.68	0.76	54,74	7.07	0.75
fsskmeansRAP	54,71	64.98	0.76	54,74	60.16	0.75
fsskmeansRASPMD	54,71	110.92	0.76	54,74	102.80	0.75

For this data set, the actual relevant features for the classification of the five dynamic regimes are unknown, in contrast to Datasets 1,2 and 3. However the results (Table 4) are quite similar to the previous Datasets in terms of accuracy and computational time. The fastest method is fssPkmeansRA, which needed on average 7.68 seconds to select the optimal feature subset for the setting Δ_1 (five regimes for $\Delta = 110, 130, ..., 190$) and 7.07 seconds for the setting Δ_2 ($\Delta = 120, 140, ..., 200$). It is also faster than fsskmeansRA (second fastest) 6.70 and 6.87 times on average, for the settings Δ_1 and Δ_2 respectively. Compared to the serial implementation fsskmeans, fssPkmeansRA is faster 14.17 times for the setting Δ_1 and 15.21 times for Δ_2. In terms of Accuracy, all algorithms are equal and achieve an average classification accuracy of 76% for Δ_1 and 75% for Δ_2. For the setting Δ_2, the common features selected by the algorithms in all iterations are those presented in Table 4, while depending on the iteration, the feature (each algorithm is repeated 10 times and different results can occur) f_{51} is selected two of ten times against f_{54}. It was also observed that fsskmeans, which uses the simple *k-means* as classifier, added a lot more features to the optimal subset compared to the other methods using *k-meansRA* as classifier. We stopped the algorithm when the optimal subset included two features, in order to compare the execution time with the other examined algorithms. The extra features that fsskmeans added to the optimal subset, lead to an extra computational cost, but it also seems to achieve better classification accuracy, since the Accuracy becomes slightly better than the other methods (0.77 for Δ_1 and 0.79 for Δ_2). The optimal feature subset that best discriminates the five chaotic regimes is unknown, and the algorithms selected a feature set that best served this purpose.

5 Case study

In order to evaluate the performance of the examined methods, a study in real world dataset was carried out. The study was performed to the Human Activity Recognition using Smartphones Dataset, which carried out with a group of 30 volunteers within an age range of 19-48 years and each person performed six activities such as walking, walking upstairs, walking downstairs, sitting, standing, laying, wearing a smartphone on the waist. The Dataset consists of 561 features and 10299 instances. The true relevant features with the class variable aren't known. The study was performed similarly as before, but for real world data. Similar to Dataset 4, the problem doesn't concern binary classification (6 classes here) and the Accuracy was performed in order to evaluate the classification accuracy of the different examined methods. The results of the examined variations for the Human Activity Recognition Using Smartphones Dataset are presented in Table 5. f refer to the selected features, t corresponds to the average execution time, while ac refer to the average values of the similarity measure Accuracy.

The fssPkmeansRA achieved the highest accuracy (about 77%), while it was needed only 66.05 seconds in order to complete the optimal subset. Comparing with the methods that use *k-meansRA* as classifier, fssPkmeansRA is about 2.16, 6.40 and 8.14 times faster than fsskmeansRAP, fsskmeansRA and

Table 5: Results of Case Study

	f	t	ac
fsskmeans	10,42	737.56	0.71
fsskmeansRA	10,57	422.90	0.77
fssPkmeansRA	10,57	66.05	0.77
fsskmeansRAP	10,57	142.84	0.77
fsskmeansRASPMD	10,57	538.28	0.77

fsskmeansRASPMD respectively. The simple serial implementation fsskmeans had the worst classification accuracy (about 71%). Compared to the serial fsskmeans, fssPkmeansRA was 11.17 times faster. It is also observed that the methods that use *k-meansRA* as classifier, select only different features compared to fsskmeans. Specifically fsskmeansRA, fssPkmeansRA, fsskmeansRAP and fsskmeansRASPMD select the features f_{10} and f_{57} for the optimal subset, while the serial implementation fsskmeans selects the features f_{10} and f_{42}. It is worth noting that normally the fsskmeans selected five or more features for the optimal subset (f_{10},f_{42},f_{53},f_{147},f_{133}). Similarly to the Dataset 3 and the Dataset 4, the serial fsskmeans algorithm was stopped when two features were selected, since only a slight improvement was observed in terms of accuracy and in order to compare it with the other methods, in terms of the execution time. The serial FSS method utilizing parallelized *k-meansRA* with the MATLAB *spmd* command, needed on average 538.28 seconds in order to complete the feature selection process, while achieving about 77% accuracy. Compared to the serial FSS method using the *parfor* MATLAB command in order to parallelize the *k-meansRA* classifier, it is observed that fsskmeansRASPMD is about 3.77 times slower, without improving the accuracy.

In a study conducted in 2015, [33], for the same dataset, the authors achieved classification accuracy of 60% in 582.1 seconds using serial implementation of k-means. As shown in Table 5, fssPkmeansRA achieve 77% classification accuracy, while it only takes 66.05 seconds to run, which as mentioned in their paper, is really important in real time activity monitoring, because it requires the model to be built dynamically from the captured data.

6 Discussion

For Datasets with known functionally relevant features to the class variable such as Dataset 1, Dataset 2 and Dataset 3, the fssPkmeansRA was faster than both the serial implementations and the other examined parallelized methods, while the classification accuracy was equal or even better. Such an example concerns Dataset 3, where taking into account that based on the design of the system, only the f_1,...,f_6 features are important and related to the class variable, it was observed that the serial implementation selected only f_4 and included feature f_{21} to the optimal subset of features. This could lead to really bad conclusions, since for Dataset 3, beyond features f_1,...,f_6, all the other features are random normal variables and irrelevant to the class variable y. It is therefore understood that the fssPkmeansRA, contrary to the other examined methods, could dramatically reduce the processing time, without compromising the efficiency.

For datasets where the actual relevant features are unknown, the serial implementation fsskmeans, appears slightly more accurate in terms of classification accuracy, when more features were added to the optimal subset (but at a high computational cost) and less accurate when the comparison is made for the same size of the optimal subset.

For all the examined Datasets, the fsskmeansRASPMD that use *spmd* MATLAB command in order to parallelize the *k-meansRA* classifier, was slower even from the corresponding serial one (fsskmeansRA), which indicates that this method might not be appropriate for these systems.

In general, the proposed fssPkmeansRA variant, where the whole feature selection process is parallelized, appeared to be the fastest feature selection method compared to the rest of the examined methods, but also with high accuracy rates, proving that it could also be used in on-line processes for reducing the dimensional space in clustering and classification problems.

7 Conclusion

In this work, different parallelization variations of feature selection algorithms using *k-means* as classifier were proposed and examined, in order to evaluate the significance of the parallelization process in terms of computational cost and classification accuracy, as well as to emphasize on the importance of the methods and the tools that will be chosen for the parallelization to be occured.Specifically, different methods such as fsskmeans and fsskmeansRA was employed, in order to examine the importance of the choice of classifier, since fsskmeans uses the simple *k-means* and fsskmeansRA uses the *k-meansRA*. The fssPkmeansRA and fsskmeansRAP were compared, in order to examine the significance of selecting which part of the algorithm to parallelize, since in the fssPkmeansRA the whole feature selection process was parallelized, while in fsskmeansRAP only the classifier was parallelized. The fsskmeansRASPMD was employed for comparing different parallelization MATLAB techniques such as *parfor* and *spmd*, when all the other parameters remained the

same. The results suggest that in order to maximize accuracy and minimize the execution time, it is crucial to determine carefully the parameters for both the classifier to be used and the part of the algorithm that will be parallelized. At least for the Datasets that used in this study, the fssPkmeansRA considered as the most reliable both in terms of accuracy and execution time. In the fssPkmeansRA the whole feature selection process was parallelized using the *parfor* MATLAB command and the *k-meansRA* was employed as classifier.

Further research would involve a different kind of parallelization, not only on the CPU but also on the GPU. A comparison of execution time and accuracy with other feature selection methods, using different classifiers, could also be performed to further examine the importance of parallelization.

8 Declarations

All authors declare that they have no conflicts of interest.

References:

[1] S. Mittal, M. Shuja, and M. Zaman, "A review of data mining literature," *IJCSIS*, vol. 14, pp. 437–442, 2016.

[2] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[3] M. Capo, A. Pérez, and J. Lozano, "An efficient k-means algorithm for massive data," 2016.

[4] M. Omran, A. Engelbrecht, and A. Salman, "An overview of clustering methods," *Intell. Data Anal.*, vol. 11, pp. 583–605, 2007.

[5] X.-D. Wang, R.-C. Chen, F. Yan, Z.-Q. Zeng, and C.-Q. Hong, "Fast adaptive k-means subspace clustering for high-dimensional data," *IEEE Access*, vol. 7, pp. 42639–42651, 2019.

[6] R. Chen, C. Dewi, S. Huang, and R. Caraka, "Selecting critical features for data classification based on machine learning methods," *Journal Of Big Data*, vol. 7, p. 26, 2020.

[7] R. Shang, J. Chang, L. Jiao, and Y. Xue, "Unsupervised feature selection based on self-representation sparse regression and local similarity preserving," *International Journal of Machine Learning and Cybernetics*, vol. 10, 2019.

[8] S. Shekhar, N. Hoque, and K. Bhattacharyya, "Pknn-mifs: A parallel knn classifier over an optimal subset of feature," *Intelligent Systems with Applications*, vol. 14, p. 200073, 2022.

[9] I. Guyon and A. Elisseeff, *An Introduction to Feature Extraction*, vol. 207, pp. 1–25. 2008.

[10] C. C. Aggarwal and C. K. Reddy, eds., *Data Clustering: Algorithms and Applications*. CRC Press, 2014.

[11] C. Ding and H. Peng, "Peng, h.: Minimum redundancy feature selection from microarray gene expression data. journal of bioinformatics and computational biology 3(2), 185-205," *Journal of bioinformatics and computational biology*, vol. 3, pp. 185–205, 2005.

[12] A. Tsimpiris, I. Vlachos, and D. Kugiumtzis, "Nearest neighbor estimate of conditional mutual information in feature selection," *Expert Systems with Applications*, vol. 39, p. 12697–12708, 2012.

[13] H. Chen and X. Chang, "Photovoltaic power prediction of lstm model based on pearson feature selection," *Energy Reports*, vol. 7, pp. 1047–1054, 2021. 2021 International Conference on Energy Engineering and Power Systems.

[14] J. Maldonado, M. Riff, and B. Neveu, "A review of recent approaches on wrapper feature selection for intrusion detection," *Expert Systems with Applications*, vol. 198, p. 116822, 2022.

[15] K. Bouzoubaa, Y. Taher, and B. Nsiri, "Predicting dos-ddos attacks: Review and evaluation study of feature selection methods based on wrapper process," *International Journal of Advanced Computer Science and Applications*, vol. 12, 2021.

[16] X. Zhang, G. Wu, Z. Dong, and C. Crawford, "Embedded feature-selection support vector machine for driving pattern recognition," *Journal of the Franklin Institute*, vol. 352, no. 2, pp. 669–685, 2015. Special Issue on Control and Estimation of Electrified vehicles.

[17] M. Zhu and J. Song, "An embedded backward feature selection method for mclp classification algorithm," *Procedia Computer Science*, vol. 17, pp. 1047–1054, 2013. First International Conference on Information Technology and Quantitative Management.

[18] N. Mahendran and P. Vincent, "A deep learning framework with an embedded-based feature selection approach for the early detection of the alzheimer's disease," *Computers in Biology and Medicine*, vol. 141, p. 105056, 2022.

[19] L. Venkataramana, S. Jacob, and R. Ramadoss, "A parallel multilevel feature selection algorithm for improved cancer classification," *Journal of Parallel and Distributed Computing*, vol. 138, 2019.

[20] Q. L., J. W., and H. Z., "A wind speed interval forecasting system based on constrained lower upper bound estimation and parallel feature selection," *Knowledge-Based Systems*, vol. 231, p. 107435, 2021.

[21] J. González-Domínguez, V. Bolón-Canedo, B. Freire Castro, and J. Touriño, "Parallel feature selection for distributed-memory clusters," *Information Sciences*, vol. 496, 2019.

[22] N. Hijazi, H. Faris, and I. Aljarah, "A parallel metaheuristic approach for ensemble feature selection based on multi-core architectures," *Expert Systems with Applications*, vol. 182, p. 115290, 2021.

[23] H. Kizilöz and A. Deniz, "An evolutionary parallel multiobjective feature selection framework," *Computers and Industrial Engineering*, vol. 159, p. 107481, 2021.

[24] B. Beceiro, J. González-Domínguez, and J. Touriño, "Parallel-fst: A feature selection library for multicore clusters," *Journal of Parallel and Distributed Computing*, vol. 169, pp. 106–116, 2022.

[25] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," vol. 1, pp. 281–297, 1967.

[26] D. Varsamis and A. Tsimpiris, "Parallel implementations of k-means in matlab," *Contemporary Engineering Sciences*, vol. 13, pp. 359–366, 2020.

[27] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, 1985.

[28] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[29] J. Hanley and B. Mcneil, "The meaning and use of the area under a receiver operating characteristic (roc) curve," *Radiology*, vol. 143, pp. 29–36, 1982.

[30] K. Fu, J.-C. Simon, A. Checroun, C. Roche, E. Coffman, and J. Eve, "Sequential methods in pattern recognition and machine learning," *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences, Série A*, 1971.

[31] D. W. Aha and R. L. Bankert, *A Comparative Evaluation of Sequential Feature Selection Algorithms*, pp. 199–206. New York, NY: Springer New York, 1996.

[32] A. Tsimpiris and D. Kugiumtzis, "Feature selection for classification of oscillating time series," *Expert Systems*, vol. 29, no. 5, pp. 456 – 477, 2012.

[33] M. Yamin and G. Chetty, "Intelligent human activity recognition scheme for e health applications," *Malaysian Journal of Computer Science*, 2015.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.