Robust Feature Selection by Filled Function and Fisher Score

Javad Hamidzadeh (Hamidzadehj@gmail.com)
Sadjad University
Mahsa kelidari
Sadjad University

Research Article

Keywords: Feature selection, High-dimensional data, Filled function, Fisher score, Filter algorithm

Posted Date: February 7th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1102788/v1

Robust Feature Selection by filled function and fisher score

Mahsa kelidari[1], Javad Hamidzadeh[2]
Faculty of Computer Engineering and Information Technology
Sadjad University
Mashhad, Iran
Ma.Kelidari117@sadjad.ac.ir[1], J_Hamidzadeh@sadjad.ac.ir[2]

Abstract

Feature selection is essential in high-dimensional data analysis and filter algorithms, and due to their simplicity and fast speed, they have increasingly been drawing attention in recent years. Retaining all features in machine learning tasks is not only inefficient but the irrelevant and redundant features may have an adverse impact on the classification accuracy rate. Feature selection is an optimization problem which aims to transform the dataset's high-dimensional space to a lower-dimensional space by utilizing the relevant and suited features. Feature selection is a time-consuming task, while it is very effective in saving the time devoted to the learning algorithm. In feature selection algorithms, filter algorithms are increasingly attractive due to their simplicity and fast speed. In this paper, we are going to introduce a supervised filter feature selection using filled function and fisher score (FFFS). Based on this criterion, we try to find a feature subset resulting in the least classification error rate. In order to prove the effectiveness of the proposed algorithm, Extensive experiments have been conducted on 20 high-dimensional real-world datasets. Experimental results reveal the superiority of the proposed algorithm to state-of-the-art algorithms in terms of minimum classification error rate. Results validated through statistical analysis indicated that the proposed algorithm is able to outperform the reference algorithms by minimizing the redundancy of the selected features. So, the selected feature subset can avoid serious negative impacts on the classification process in real-world datasets. In addition, this paper proves the ability of the proposed algorithm in selecting the most relevant features for classification tasks by applying different noise rates to the datasets. According to the experiments, the FFFS is less affected by noisy attributes in comparison with other algorithms. Thus, it is a reasonable solution in handling noise and avoiding serious negative impacts on the classification error rate in real-world datasets.

Keywords

Feature selection; High-dimensional data; Filled function; Fisher score; Filter algorithm.

1. Introduction

Feature selection problem is a challenging and computationally expensive process, especially when dealing with high dimensional datasets [1]. So, in recent years, considerable attention has been paid to high dimensional data, such as Bioinformatics [2], brain tumor diagnosis [3], image retrieval [4], exploiting the ensemble paradigm for genomic data [5], emotion recognition [6], biomedical corpora [7], privacy-preserving [8], detecting malware apps [9], etc.

Feature selection becomes an important task in machine learning which shows how to assess the quality of the selected subset for the next steps. So, searching for the best feature set requires a criterion to compare the subsets and find the best one, because it can be effective in the performance of high dimensional data classification [10, 11].

With the advancement of information technology over the last decade, many new applications have emerged as hubs for information gathering and dissemination. So, data velocity and variety, have brought many challenging issues to the digital processing of raw data [12, 13].

In recent years, with the rapid advances of computer and database technologies, there are many sources of data that can be used to solve various problems. Then, optimum usage from big and complex data sources becomes the focus of many research areas. To mitigate this problem, one of the potential ways is to reduce the amount of data with the feature selection technique [14, 15].

This enormous data can contain a lot of irrelevant, noisy, and redundant information which may degrade the performance of learning algorithms. So, not only the large quantities of high dimensional data are not profitable, but also, they have brought great challenges [16-18].

Feature selection is a very important preprocessing or knowledge discovery tool for data analysis, and it has been applied to many domains and also used as an important technique for high-dimensional data which has shown its promising ability to enhance a variety of other learning tasks, such as data classification document analysis, and image processing and video processing, to name a few [22, 32, 33].

Generally, features can be categorized as: relevant, irrelevant, or redundant. An optimal feature set is a description of the original data with relevant and non-redundant features that most contribute to learning accuracy [19, 20].

One of the most important challenges in high-dimensional data analysis is the limitation in time, memory, and computational cost [18, 21]. It is desirable to reduce high data dimensionality for many machine learning tasks due to the curse of dimensionality [12, 15, 22].

Using a proper set of features leads to the reduction of model complexity of classifiers and it also causes faster convergence in the optimum results, thus, creating an improved performance of classifiers [17, 23, 24]. On the other hand, failing to remove the redundant features does not cause information problems, but raises the computational cost and increases the response time of the system. In addition, due to storage space constraints, it results in storing a lot of non-useful information along with useful data [25, 26].

According to the accessibility of class label data, feature selection can be partitioned into 3 categories [27]. When class labels are accessible, it is called supervised feature selection [28]; however, if data consists of the samples and the features without any information about the natural grouping of the data, it is called unsupervised feature selection [9, 29-31]. In the end, the integration of a small amount of labeled data into unlabeled data as additional information, such as an unsupervised algorithm with additional information, is called semi-supervised feature selection [32-34].

The most basic technique searches all possible subsets and selects the subset that minimizes the model error. However, this exhaustive search is not computationally effective, particularly in situations where a large number of features exist. So, several feature selection algorithms were created [19, 35].

Depending on how to generate feature subsets, feature selection algorithms can be divided into three categories: Wrappers, Filters, and Hybrid or Embedded [10, 11, 25]. The filter algorithms are independent of any classification scheme, so when the features get evolved, there is no need to evolve the pattern as well. Obviously, Filter algorithms select key features using intrinsic properties of data, but under particular conditions, they could give the optimal set of features for a given classifier [23, 36]. A big challenge in filter algorithms is the difficulty to design a fitness function. So, there is less work on filter algorithms than wrapper algorithms [1, 35].

In this work, we focused on the filter model based on filled function with fisher fitness function, and aim to develop a new feature selection algorithm which can effectively remove irrelevant, noisy, and redundant features. To this end, the proposed algorithm first computes the relevancy of each feature and then computes correlation values between features. This results in avoiding higher correlated features.

For clarity, the main contributions of this paper are summarized as follows:

1- It is the first time that the efficient filled function algorithm is used on a feature selection problem.
2- The fisher score is used as a fitness function to rank the features.
3- Efficiently selects informative features and reaches convergence in a reasonable time.

4- The filled function algorithm for the feature selection problem is defined as a supervised feature selection problem maximizing the performance of the learning algorithm and minimizing the feature set size.
5- Experimental results on a wide variety of high-dimensional datasets have shown the superiority of the proposed algorithm in achieving the minimum classification error rate.
6- The efficiency and effectiveness of the filled function algorithm are demonstrated through extensive comparisons with other algorithms using 20 real-world high dimensional datasets.

The rest of this paper is organized as follows: In Section 2, the related literature of filter feature selection algorithms is reviewed. The proposed algorithm is explained thoroughly in Section 3. Section 4 reports the experimental results and analyses the performances of different feature selection algorithms. Finally, Section 5 contains the conclusions and future works.

2. Related Works

Filter algorithms select the most discriminative features through the character of data. So, it means, the intrinsic properties of the data are used for feature selection, without the need of using any classification algorithm. The main advantages of the filter algorithms are their speed and scalability [1, 37].

Within the filter model, each feature can evaluate individually or through feature subsets, then this kind of algorithms can be further categorized into two groups: univariate and multivariate [38].

Univariate algorithms evaluate each feature without attention to other features and make a rank list based on the efficacy of each feature according to the criterion. Then, the features greater than a threshold value are selected. Simplicity and light computational cost of feature ranking, make it as widespread ways in feature selection [39].

However, this algorithm may not be efficient to select the best features. An issue is a contention that the redundant subset might be optimal in certain datasets. Because little attention has been devoted to the impact of correlation between the features [40], each feature may not be rated on its own independently, but it can play an effective role in the selected feature set due to its correlation with other features [39, 41].

Meanwhile, multivariate algorithms evaluate features together with a relevance estimation of each feature for obtaining a feature subset. In order to evaluate features in a multivariate algorithm, an algorithm based on maximizing the ratio of between-class scatter and within-class scatter was proposed and called linear discriminant

analysis. But it has some issues in calculating the inverse matrix of within-class scatter [42, 43].

The graph-based feature selection algorithm is another multivariate filter algorithm, which constructs the graph by preserving the geometrical structure in constructing the neighborhood graph, then selects the relevant features [44].

In recent years, plenty of attention has been received for meta-heuristic algorithms for feature selection, like ant colony optimization (ACO) [45], genetic algorithm (GA) [46], gray wolf optimization (GWO) [47], crow search algorithm (CSA) [48], cuckoo optimization algorithm (COA) [13], bat optimization algorithm [49] and simulated annealing (SA) [50], these algorithms did not perform well in reducing the features due to their random behavior, low convergence rate and entrapment in local optima [1].

None of the above works achieved the balance between the classification accuracy rate and the feature reduction rate. To tackle the above-mentioned problem, this paper proposes a novel filter feature selection algorithm, in which the relevant features are selected by using the filled function. So, the main aim of this paper is to provide an algorithm that has a reduced classification error rate and optimum reduction of the number of features.

3. Background

In this section, the filled function algorithm is described in section 3.1, and the definition of fisher score is briefly reviewed in part 3.2.

3.1 Filled Function Algorithm

In high-volume data, despite the curse of dimensionality, it is important to have an optimal subspace of the feature set. Nowadays, studies on global optimization for feature selection problems have been significantly accelerated. The optimal subspace caused less error rate in classification and low computing time.

The filled function algorithm has been introduced by Ge in 1990 [51]. The general form of the filled function is as follows in Eq. (1):

$$\begin{aligned} (P)\ & \min\ f(x) \\ & s.t.\quad g_i(x) \le 0,\ i = 1,...,m; x \in X, \end{aligned} \tag{1}$$

where $f: X \to R, g_i: X \to R, i = 1, \dots, m$ and $X$is a box. Let x^* be a local minimizer of the problem(P). The Filled function is an auxiliary function with satisfactory for global optimization results. To optimize the problem, the filled function converts the local minima points of the original problem to the local maximum point. This maximum point is a better initial point for the next iteration of problem-solving. In

other words, at pointx^*, that can be further minimized to get a point, say $\overline{x}$, of $f(x)$ lower than the $f(x^*)$ when x^* is not a global minimizer. Then the minimization of $f(x)$ is restarted from the point $\overline{x}$. The process is then repeated until a global minimizer of $f(x)$ is obtained.

The filled function algorithm has been introduced in [51]. The global optimization problem (F) is considered in the following form in Eq.(2):

$$\min f(x) \quad (2)$$

$$\text{s.t. } x \in \Omega$$

where $\Omega = [l, u] \subset Rn$ is a closed bounded set and f (x) is a continuous and differentiable function with a finite number of local minima in Ω[52].

Firstly, a flatten function has been applied in the objective function to eliminate numerous local minima just as Sui and Liu do in literature [53] and [54]. Flatten function was introduced by Wang [55], which has the following form in Eq.(3) and Eq.(4):

$$s(x, x^*) = f(x^*) + \frac{1}{2}\{1 - sign[f(x) - f(x^*)]\}[f(x) - f(x^*)] \quad (3)$$

i.e. (4)

$$s(x, x^*) = \begin{cases} f(x^*), f(x) \geq f(x^*) \\ f(x), f(x) < f(x^*) \end{cases}$$

where *f (x)* is the original function and x^* is its current local minimum. By using it, all those local minima equal to and worse than x^* will be eliminated. Thus, many locally optimal solutions will be eliminated, which makes the algorithm focus its attention only on the better local optimal solutions than the current local optimal solution x^* and thus makes the global minimum finding much easier. For the properties and the detailed information, please refer to [55].

Let Gm and Lm represent the set of global minima and local minima of the original function f (x) respectively.

Definition 1 Assume x^* is a local minimum of f (x), if p(x, x^*) satisfies the following conditions, p(x, x^*) is referred to as a filled function of the original function f (x) at x^*:

(1) x^* is a strictly local maximum of p(x, x^*).

(2) For p(x, x^*), none of the points in $\Omega 1 = \{x | f(x) \geq f(x^*), x \in \Omega \backslash x^*\}$ is a stationary point.

(3) If $x^* \in$ Lm but $x^* \notin$ Gm, then in $\Omega 2 = \{x | f(x) < f(x^*), x \in \Omega\}$, p(x, x^*) has a local minimum.

The filled function without any adjustable parameter at the current minimum x^* is constructed by the following formula [52].

$$p(x,x^*) = \frac{1}{1+\|x-x^*\|} - \alpha[s(x^*,x^*) - s(x,x^*)]^{\frac{1}{2}} - [s(x^*,x^*) - s(x,x^*)]^2 \quad (5)$$

where α is a constant, whose value is the reciprocal of the order of magnitude of $s(x^*,x^*)$. Note that after the current local minimum x^* was founded, the search will focus on Ω_2. accordingly, $p(x, x^*)$ is a continuously differentiable function in Ω_2. In fact, in region $\Omega 1$ and $\Omega 2$, $p(x,x^*) = \frac{1}{1+\|x-x^*\|} - \alpha[f(x^*) - f(x)]^{\frac{1}{2}} - [f(x^*) - f(x)]^2$ is continuous and differentiable, respectively.

Now, we will prove that the proposed $p(x, x^*)$, i.e.(5), is a filled function that satisfies the conditions in Definition 1.

Theorem 1 *The current local minimum x^* of $f(x)$ is a strictly local maximum of the proposed function $p(x, x^*)$.*

Proof x^* is a local minimum of the original function $f(x)$, so there exists a small positive real number $\delta > 0$ satisfying $f(x^*) \leq f(x)$ for $\forall x \in N(x^*, \delta)$. Thus, in this neighborhood $N(x*, \delta)$, we have $s(x, x^*) = f(x^*)$.

Hence, for any $x \in N(x^*, \delta)$, $x \quad = x^*$, we have

$$p(x,x^*) = \frac{1}{1+\|x-x^*\|}$$

Then,

$$p(x,x^*) - p(x^*,x^*) = \frac{1}{1+\|x-x^*\|} - \frac{1}{1+\|x^*-x^*\|} = \frac{1}{1+\|x-x^*\|} - 1 < 0$$

Hence, x^* is a strict local maximum of the designed $p(x, x^*)$.

Theorem 2 *Assume that x^* is the current local minimum of the original function $f(x)$, then in region $\Omega 1 = \{x | f(x) \geq f(x^*), x \in \Omega x^*\}$, for $p(x, x^*)$, none of the points is a stationary point.*

Proof For $\forall x \in \Omega 1$, $s(x, x^*) = f(x^*)$, i.e. $s(x, x^*) - s(x^*,x^*) = 0$.

$$p(x, x*) = \frac{1}{1+\|x-x^*\|}$$

Then,

$$\nabla p(x,x^*) = \frac{-1}{(1+\|x-x^*\|)^2} \cdot \frac{x-x^*}{\|x-x^*\|} \neq 0, x \neq x_1^*$$

Thus, in region Ω_1, for $p(x, x^*)$, none of the points is a stationary point.

Theorem 3 *Suppose that $x^* \in Lm$ but $x^* \notin Gm$, then there exists a local minimum of $p(x, x^*)$ in $\Omega_2 = \{x | f(x) < f(x^*), x \in \Omega\}$.*

Proof Let $\partial\Omega_2 = \{x | f(x) = f(x^*), x \in \Omega\}$ be the bound of Ω_2. Then $\Omega_2 = \partial\Omega_2 + \Omega_2 = \{x | f(x) \leq f(x^*), x \in \Omega\}$ is the closure of Ω_2, which is bounded. Because $x^* \notin Gm$, there is at least one point in Ω_2. Hence, $\overline{\Omega_2}$ is a nonempty bounded closed set. Because $p(x, x^*)$ is continuous and differentiable on $\overline{\Omega_2}$there must exist a minimum $\hat{x}$ of $p(x, x^*)$ on $\overline{\Omega_2}$, i.e. $\nabla p(x, x^*)=0$. From Theorem 2, we know that for $p(x, x^*)$, none of the points in the set $\partial\Omega 2 = \{x | f(x) = f(x^*), x \in \Omega\}$ is a stationary point. So this minimum of $p(x, x^*)$ must be in Ω_2. This minimum is at least a local minimum of $p(x, x^*)$ in Ω_2.

Assume that x^* is the current local minimum of original function $f(x)$, and the filled function at this point is $p(x, x^*)$. Note that x^* is a local maximum of $p(x, x^*)$ (Definition 1). To minimize $p(x, x^*)$, it is necessary to select an initial point near x^*. In the existing filled function algorithms, a widely used algorithm to generate the initial points is as follows in Eq.(6) [52]:

$$x_i = x^* + \delta e_i \tag{6}$$

where δ is a small step size and e_i represents the i-th coordinate axis direction. However, if the local minima of $p(x, x^*)$ are not near in the coordinate directions, it is possible that $p(x, x^*)$ cannot obtain a good local minimum by using these initial points and its local minimum found may not enter a deeper valley of the $f(x)$. This may result in the failure of the algorithm [52].

We take a two-dimensional problem $f(x) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 - x_1x_2 + x_2^2$ as an example to illustrate this. Assume $x*$ is its current minimum, the landscape of the proposed filled function $p(x, x^*)$ is displayed in figure 1a [52].

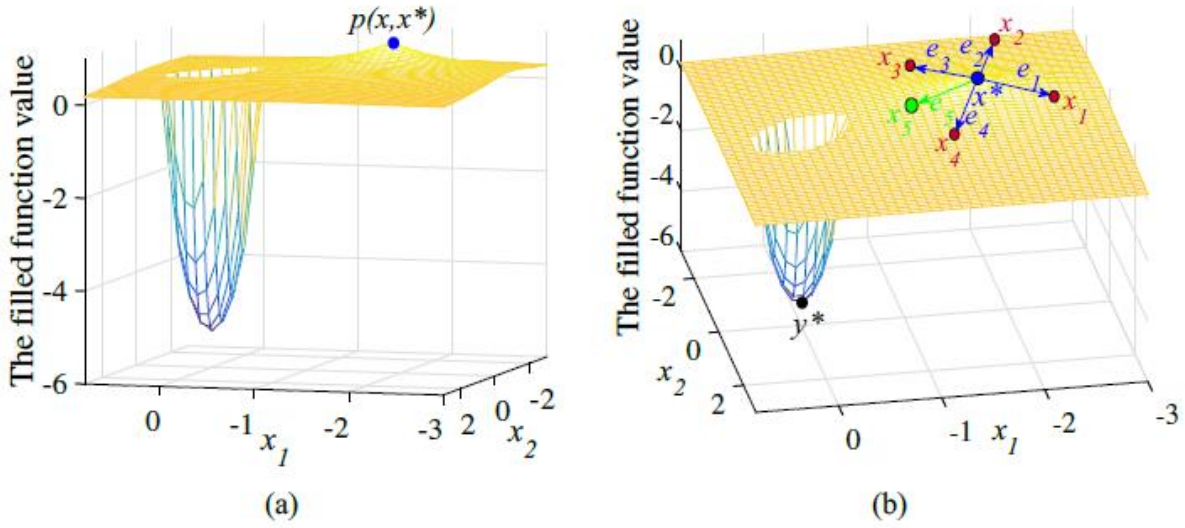


Figure 1:The illustration of p(x, x*) constructed at x*[52]

Let the blue solid point denote x^* in figure 1b, where figure1b is a top view of figure 1a. We can see from figure 1b that when $x1$ to $x4$ generated by Eq.(6) as shown in red circles are used as initial points, the local minimum y^* does not locate near these points. It is very difficult or we have to take much computation cost to find the local minimum y^* from these initial points. That is to say, the initial points generated by Eq.(6) may not be good initial points. Instead, if we choose x_5 as an initial point, we may easily find y^*. Thus, x_5 may be a good initial point. Based on this observation, it is necessary to design a scheme to adaptively select the initial points instead of using a fixed manner to choose the initial points. To achieve this goal, a new algorithm to generate the initial points from x^* along with adaptively changing both directions (not fixed to axis directions) and the number of directions is developed. The details are as follows [52]:

Step 1. The coordinate axis directions e_i and $-e_i$ are taken as the initial directions to generate initial points according to Eq.(6), where $\delta = 0.001$. Let $N = 2$. The number
of the initial points is $P_{num} = N * dim$, where dim is the number of dimensions of the problem.

Step 2. If we cannot find a local minimum of $p(x, x^*)$ by using these initial points, let $N = N + 1$. Generate $P_{num} = N * dim$ evenly distributed directions d_i for $i = 1, 2, \cdots,$ P_{num} and P_{num} initial points $x_i = x_1^i 1 + \delta d_i$ for $i = 1, 2, \cdots, P_{num}$.

Step 3. Repeat Step 2 until we find a local minimum or N reaches a pre-assigned threshold.

To elaborate on this strategy, we use a two-dimensional problem as an example to illustrate the idea of generating initial points in Fig. 2. The blue point x^* is the current minimum of $f(x)$ and y^* is a local minimum of the filled function [52]. So, we take the coordinate axis directions $d1 \sim d4$ shown by the red dotted arrows to generate

the initial points $x1 \sim x4$ as shown in Fig. 2a. We can see from figure 2a that none of the four initial points is near to y^*. It is difficult to find y^* from these initial points. In this case, we adaptively choose the uniformly distributed directions $d1 \sim d6$ and generate new initial points $x1 \sim x6$ as shown in figure 2b. Subsequently, $x5$ is near to y^* and by using it as the initial point, we can easily find y^* as shown in figure 2b [52].

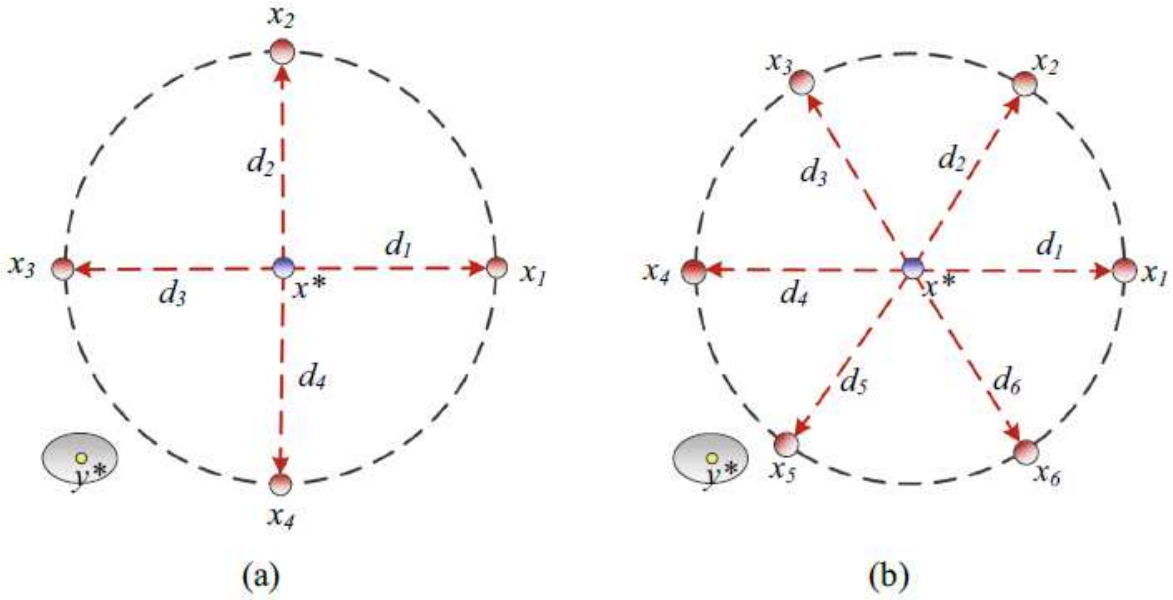


Figure 2: The illustration of the adaptive algorithm to generate initial points. a N=2 b N=3[52]

3.2 Fisher Score

Some supervised feature selection algorithms, evaluate features according to some quality creations that quantify the relevance of features individually. The selection task can be efficiently accomplished using the Fisher score based on features ranking, which leads to a suboptimal subset of features [56].

Consider the classification of C classes. Given n_i training samples $\{x_1^i, x_2^i, \dots, x_n^i\}$ for each class i, (i =1 , 2 , … , C). The a priori probability of class i is estimated by Eq. (7):

$$P_i = \frac{n_i}{\sum_{i=1}^{C} n_i} \tag{7}$$

The class means μ_i are estimated by Eq. (8):

$$\hat{\mu}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_j^i \tag{8}$$

and the gross mean μ is estimated by Eq. (9):

$$\hat{\mu}_i = \sum_{i=1}^{C} P_i \hat{\mu}_i \tag{9}$$

The sample covariance matrix $\hat{S}_i$ of the class is estimated by Eq. (10):

$$\hat{S}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \left(x_j^i - \hat{\mu}_i\right) \left(x_j^i - \hat{\mu}_i\right)^T \tag{10}$$

The within-class scatter matrix and between-class scatter matrix are estimated by Eq. (11) and Eq. (12):

$$S_w = \sum_{i=1}^{C} P_i \widehat{S_i} \tag{11}$$

$$S_b = \sum_{i=1}^{C} P_i (\hat{\mu}_i - \mu_i)(\hat{\mu}_i - \mu_i)^T \tag{12}$$

The class separability of a feature set can then be measured by Eq. (13):

$$J_F = trace(S_w^{-1} S_b) \tag{13}$$

This measure serves as a good criterion for feature subset selection and has shown superior performance in many practical problems. However, its calculation for a large number of features is computationally expensive. Instead, the Fisher criterion for one single feature has been prevalently used to select discriminant features. For the *k*th feature, it is calculated by Eq. (14):

$$Fisher(k) = \frac{S_b^{(k)}}{S_w^{(k)}} \tag{14}$$

Where $S_b^{(k)}$ and $S_w^{(k)}$ are the *k*th diagonal element of S_b and S_w , respectively, and can also be calculated from the data of a single feature.

For feature pre-selection, we calculate the Fisher criterion of each feature, order the features in decreasing order of criterion values, and simply select the features of maximum values, while the features with very small fisher values are abandoned. Though the single-feature fisher criterion does not consider the joint separability of multiple features, it is able to retain all discriminant features by only removing irrelevant/noisy features, for which the fisher criterion is nearly zero.

4. Proposed Algorithm for Feature Selection

In this section, we propose a multivariate filter-based feature selection algorithm with filled function and Fisher score. As regards the nature of the feature selection issues, the solutions are limited to the binary space values but the real-world datasets have continuous space. So, to solve the feature selections problem, the continuous (free position) must be transformed to their corresponding binary solutions.

For example, if the value of a bit is equal to 1, then its corresponding feature is selected into the feature subset; on the contrary, 0 indicates vice versa. Figure 3 shows it clearly. It means that the second and fourth features are selected for the feature subset.

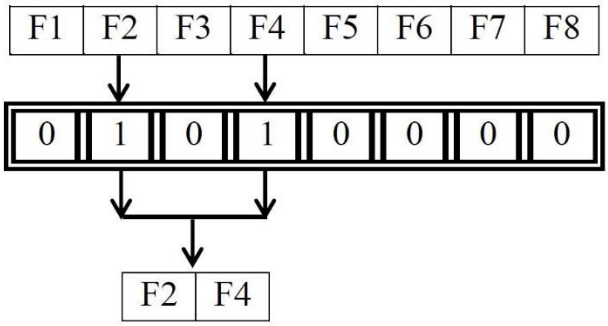


Figure 3: Example of selected features in each sample

The framework of the FFFS algorithm is shown in Algorithm 1. In iteration *t* of the purposed algorithm, there are three phases: Firstly, to reach the best feature set, it minimizes the problem to get a local minimum x^* by using an existing local optimization algorithm. Next, we adopt a flatten function $s(x, x_t^*)$ which can eliminate the local minima equal to and worse than the current minimum x_t^*. This will make the search for a global minimum much easier. Also, the Fisher score function helps rank the results to make the selection easier. Finally, we construct a filled function $p(x, x_t^*)$ at x_t^*, and then optimize it from the initial points adaptively selected. If we could not find an acceptable feature set to x_t^*, we will use the widening strategy. Otherwise, we will use this better feature set as a new initial point i.e. x_{t+1}^0, in $(t+1)th$ iteration to minimize the original function. The above phases are repeated until a global minimum of the original function is found.

Initialization steps:

Step1:	Choose an initial point $x_1^0 \in \Omega$ for the original problem.
Step2:	Set the iteration number t=1.
Step3:	**S**et the initial value of n=2, then the number of initial directions p_{num}=2*dim. ‖ dim is the number of dimensions of the problem.
Step4:	Set the threshold of N, N_{max}
Main Steps:	
Step1:	Minimize the original problem (F) from x_t^0 by applying an existing local search algorithm to get local minimum x_t^*
Step2:	Eliminate the local minima inferior to x_t^* by using flatten function $s(x,x_t^*)$ at x_t^*.
Step3:	Sort the result by fisher score
Step4:	Construct $p(x,x^*) = \frac{1}{1+\|x-x_t^*\|} - \alpha[s(x_t^*,x_t^*) - s(x,x_t^*)]^{\frac{1}{2}} - [s(x_t^*,x_t^*)] - s(x,x_t^*)^2$ at x_t^*;
Step5:	Generate the search direction d_i, i=1, 2, …, P_{num} where P_{num}=N*dim and set j=1.
Step6:	If j<P_{num}, go to step 7, otherwise, go to step 9;
Step7:	Generate the initial point for $p(x, x_t^*)$ from $\tilde{x}_t^* = x_t^* + \delta d_j$. Minimize $p(x, x_t^*)$ from $\tilde{x}_t^*$to get a local minimum p^* by adopting a local search algorithm. If p* is not in region Ω, let j=j+1, then go back to step 6, otherwise, go to step 8;
Step8:	If $f(p^*)<f(x_t^*)$, set t=t+1, $x_t^0 = p^*$and N=2, then go back to step1; otherwise, let j-j+1 and go back to step 6;
Step9:	Set N=N+1
Step10:	If N<N_{max}, then go back to step 5, otherwise, go to step 11;
Step11:	Implement the narrow valley widening strategy on $s(x, x_t^*)$ to get a better minimum x_t^* in this valley.
Step12:	If the local minimum cannot be improved, terminate the process and take x_t^* as the global minimum of the original problem; otherwise, let N=2 and go back to step2.

Algorithm 1: Pseudo FFFS algorithm

In algorithm 1, β is a small positive number, $\alpha \in [0,1]$, and B is a large positive number. Parameter d is the number of data set features.

To summarize, first, the labeled data are mapped into two hypersphere regions for target and non-target objects. This mapping process is considered nonlinear programming. The problem is solved by employing the Fisher score for finding a global minimizer. The global minimizer is considered as a boundary that fits the target class. In the end, a one-class classifier to detect target class members is obtained.

In the next section, experimental results are explained.

5. Experimental Results

In this section, in order to measure the efficiency of the proposed algorithm, the evaluation of the experimental results will be discussed. The experiments were performed on various high-dimensional datasets compared with the proposed FFFS algorithm against 3 other state-of-the-art feature selection algorithms. At first, in section 5.1 the used datasets are introduced concisely. Next, in section 5.2, the experimental and parameter setups are discussed. Finally, in section 5.3 the performance of the proposed algorithm against other relevant algorithms is evaluated.

5.1 Datasets

The results presented in the experiments are based on the 15 real-world datasets from the UCI machine learning repository and the 5 medical datasets [57, 58]. These 20 datasets have different numbers of attributes and instances, as shown in table 1. In table 1, the number of data samples, features, and classes for each dataset are given. In all datasets, the missing value is replaced with the median of that feature.

Table 1: Experimental datasets [57, 58]

No.	Dataset	number of data samples	number of features	number of classes
1	Internet Advertisements (AD)	3279	1585	2
2	Dexter	300	20000	2
3	Gisette	13500	5000	2
4	DrivFace	606	6400	4
5	CNAE-9	1080	857	9
6	Gene expression cancer RNA-Seq	801	20531	5
7	Madelon	2000	449	151
8	Arcene	900	10000	2
9	Secom	1567	589	612
10	Semeion	1593	265	2
11	Daily and Sports Activities	9120	5625	19
12	DOROTHEA	1950	100000	2
13	P53 Mutants	16772	5409	2
14	ISOLET	7797	617	26
15	MICROMASS	931	1300	20
16	Brain_Tumor1	90	5920	5

17	DLBCL	77	5469	2
18	Leukemia1	72	5327	3
19	Prostate_Tumor	102	10509	2
20	SRBCT	83	2308	4

5.2 Compared Algorithms

The comparison results obtained by the proposed algorithm with the classification problems, including a feature selection algorithm based on Chebyshev distance outlier detection called NFR-Relief [59], a feature selection algorithm combining maximal information entropy and the maximal information coefficient called mMIE-mMIC [60], and a fully-connected network called NeuralFS [61]. All algorithms have been tested on 20 datasets in terms of classification error rate and F-MEASURE performance measures.

5.3 Experimental and Parameters Setup

The experiment was conducted on the Intel core i3 computing platform with 4GB RAM, 2.13 GHz Frequency. The programming environment is MATLAB R2014a in Windows 10 operating system.

Several experiments were carried out to demonstrate the efficiency and effectiveness of the FFFS algorithm. For executing the FFFS algorithm, all experiments use the same parameters for initialization and evolution. The experiments are illustrated as:

Table 2: Parameter initializations

β	10^{-9}
B	10^{9}
c	10^{4}
j	1
α	1
b	1
a	1
n	2

To be fair, all comparison algorithms used the parameters reported in the corresponding original work.

To verify the performance of the proposed algorithm, the 3-fold cross-validation is used to generate the three equal portions in the dataset. Based on this, 70% of the training partition is used for training and the left 30% is used for the fitness function validation.

5.4 Evolution Metric

The error rate (ERR), the F-MEASURE and computational time are used as the criteria for comparing the proposed algorithm and the three other state-of-art algorithms.

5.4.1 ERR

The classification process should present the correct label for each sample. To evaluate the performance of the data classification, the error rate is calculated by Eq. (15):

$$ERR = \frac{\sum_{i=1}^{n} F(y, \hat{y})}{n} \tag{15}$$

where n is the total number of samples, y is the true label and $\hat{y}$ is the label presented by classification and also $F(y, \hat{y})$ is the function which returns zero, if y is equal to $\hat{y}$, otherwise returns one. It is clear that smaller values of ERR mean better classification results.

5.4.2 F-MEASURE

In order to analyze the quality of the classification results obtained by the algorithm, the F-MEASURE is used as the weighted harmonic mean of the precision (positive predictive value) and recall (sensitivity) of the test.

In the classification results there are 4 kinds of recognition all the time **:**

1) TP: true positive
2) TN: true negative
3) FP: false positive
4) FN: false negative

Given two arbitrary variables Precision and recall, the F-MEASURE is defined as Eq. (16) and (17):

$$\text{Precision} = \frac{TP}{TP + FP} \tag{16}$$

$$\text{recall} = \frac{TP}{TP + NP} \tag{17}$$

F-Measure is defined as Eq. (18):

$$\text{F} - \text{Measure} = 2 \times \frac{Precision \times Recal}{Precision + Recal} \tag{18}$$

F-Measure is a good parameter for evaluating the quality of classification and it also describes the weighted average between Precision and Recall. For an ideal classification algorithm, this value is 1 and in the worst case is 0.

5.4.3 Wilcoxon Rank Test

The Wilcoxon statistical test is also used to find about the proposed algorithm more precisely. It is a Non-parametric statistical test implemented to determine statistical significance between algorithms [62, 63]. The Wilcoxon test is conducted at 5% significance level to verify whether there is a statistical difference between obtaining results of error rate. In other words, if p-value<0.05, then it indicates the proposed approach has a significant difference [64].

5.4.4 Robustness Test

In this paper, we intentionally generate Gaussian noises artificially and include them in the original dataset. In this way, it is possible to analyze the FFFS capability to select the main relevant features instead of noisy data. The generation of Gaussian noise is defined as Eq. (19) [65, 66]:

$$p(x) = \frac{1}{\delta\sqrt{2\pi}} e^{-(x-\mu)^2/2\delta^2} \tag{19}$$

Where μ represents the mean and σ the standard deviation. In this paper, we use a Gaussian noise with μ = 0 and with large values of standard deviation.

The features of the original datasets are transformed into noisy ones. The transformation was performed by adding or subtracting generated noise to the original features [66]. 10% noise was added at the first step, then it was increased to 40% noise. After that, the performance of the proposed algorithm was compared to three other algorithms.

5.5 Result Interpretation

In all reported tables the third column shows the results of the original data and also in all figures the result of this column is marked with yellow. The last column belongs to the proposed algorithm and the red marks in all figures belong to the FFFS algorithm. The best results on each dataset are marked as bold.

5.5.1 Error Rate

This experiment is presented here to intuitively show the rate of error in classification. In table 3 the results obtained from applying the above algorithms are

shown. For each algorithm, the average classification error rate (ERR) by k-NN is shown and also standard deviation is reported.

Table 3: Classification error rate and standard deviation comparing result (Note that rank of each algorithm is enclosed in the parenthesis and the best results are given in bold)

ERR (mean±STD %)						
No.	**Dataset**	**All features**	**NFR-Relief [59]**	**mMIE-mMIC [60]**	**NeuralFS [61]**	**FFFS**
1	D1	50.65±1.22 (5)	25.91±0.52 (4)	21.12±1.24 (3)	**12.34±0.19 (1)**	17.62±0.25 (2)
2	D2	54.96±1.95 (5)	28.95±1.32 (4)	26.74±0.63 (3)	23.57±1.02 (2)	**18.68±0.32 (1)**
3	D3	42.64±1.62 (5)	22.34±0.50 (3)	18.55±0.75 (2)	25.65±0.74 (4)	**16.65±0.65 (1)**
4	D4	54.67±1.59 (5)	**22.96±0.98 (1)**	23.82±1.41 (2)	24.98±1.12 (4)	24.41±0.29 (3)
5	D5	51.59±2.13 (5)	20.59±0.75 (2)	**15.65±0.25 (1)**	25.32±1.23 (4)	22.95±0.95 (3)
6	D6	49.84±1.94 (5)	22.34±1.02 (4)	20.54±0.75 (3)	19.63±1.32 (2)	**14.37±0.79 (1)**
7	D7	50.48±0.95 (5)	24.75±0.97 (4)	22.98±1.54 (3)	**22.13±0.87 (1)**	22.84±0.98 (2)
8	D8	53.67±1.84 (5)	20.95±1.78 (3)	22.65±1.55 (4)	19.67±0.22 (2)	**18.52±2.86 (1)**
9	D9	45.84±0.97 (5)	25.67±1.56 (4)	21.21±1.47 (3)	20.95±0.98 (2)	**16.86±0.62 (1)**
10	D10	48.51±1.11 (5)	18.23±0.99 (2)	19.72±0.73 (3)	**16.54±0.27 (1)**	20.65±0.43 (4)
11	D11	53.62±1.32 (5)	19.21±0.85 (4)	17.32±2.01 (3)	16.98±0.97 (2)	**15.62±0.62 (1)**
12	D12	43.75±1.74 (5)	19.64±1.61 (4)	18.62±0.92 (2)	19.62±1.32 (3)	**16.43±0.21 (1)**
13	D13	49.95±0.95 (5)	20.56±1.36 (3)	19.74±1.62 (2)	20.95±1.23 (4)	**17.68±0.54 (1)**
14	D14	51.59±2.13 (5)	21.39±0.45 (2)	**21.35±0.37 (1)**	24.02±0.87 (4)	21.77±0.33 (3)
15	D15	47.45±1.57 (5)	20.64±1.22 (3)	21.59±1.37 (4)	19.39±1.15 (2)	**19.32±0.65 (1)**
16	D16	49.74±2.25 (5)	25.38±1.11 (3)	**20.71±0.06 (1)**	25.27±1.05 (4)	21.85±0.72 (2)
17	D17	52.18±0.51 (5)	25.24±0.35 (3)	26.31±0.74 (4)	**23.85±0.81 (1)**	24.42±1.08 (2)
18	D18	46.66±0.34 (5)	18.75±0.77 (2)	21.34±0.54 (3)	**18.74±0.45 (1)**	22.39±0.17 (4)
19	D19	56.32±1.36 (5)	21.42±0.74 (4)	**18.19±1.00 (1)**	21.39±1.67 (3)	18.36±0.85 (2)
20	D20	47.55±0.97 (5)	23.15±0.82 (3)	24.69±1.37 (4)	21.34±0.81 (2)	**20.20±0.74 (1)**
	Average rank	5.00 (5)	3.10 (4)	2.60 (3)	2.45 (2)	**1.85 (1)**

Table 3 shows that the proposed algorithm outperformed the NFR-Relief, mMIE-mMIC, and NeuralFS algorithms in all datasets in terms of accuracy. The best results are indicated by bold values. As can be seen from Table 3, ERR on 10 datasets shows that FFFS outperforms other algorithms on D2, D3, D6, D8, D9, D11, D12, D13, D15, and D20. According to the values of ERR given in the last row, the best one is obtained by the proposed algorithm (1.85) and the worst is by NFR-Relief (3.10).

Table 4: P-value of the Wilcoxon rank-sum test for the classification error rate based on FFFS and three other feature selection algorithms (p-values >=0.5 are in bold)

No.	**FFFS vs.**	**NFR-Relief [59]**	**mMIE-mMIC [60]**	**NeuralFS [61]**
1	D1	5.65E-05	3.42E-03	**2.71E-01**
2	D2	3.15E-05	3.71E-04	3.12E-04
3	D3	6.15E-09	4.62E-07	4.25E-05
4	D4	**3.72E-01**	5.62E-08	7.62E-09
5	D5	3.66E-11	**2.01E-02**	6.21E-07
6	D6	2.83E-13	7.62E-10	5.62E-12

7	D7	7.11E-07	9.15E-05	**2.65E-01**
8	D8	4.34E-13	5.35E-13	3.21E-12
9	D9	8.61E-12	5.36E-11	6.77E-08
10	D10	1.27E-04	9.01E-04	**0.96E-02**
11	D11	9.42-E13	3.02E-14	7.23E-11
12	D12	6.58E-10	4.25E-11	5.08E-09
13	D13	5.26E-07	6.63E-08	6.06E-07
14	D14	1.67E-09	**5.62E-01**	9.00E-10
15	D15	7.71E-12	1.58E-09	1.36E-11
16	D16	3.21E-13	**7.75E-02**	8.22E-13
17	D17	1.68E-08	4.62E-06	**8.36E-01**
18	D18	2.84E-06	7.62E-05	0.77E-03
19	D19	2.26E-13	**3.32E-01**	8.20E-12
20	D20	4.18E-12	6.15E-11	2.61E-10

P-values in table 4 show that the proposed algorithm can significantly outperform other algorithms, it can be observed that the FFFS algorithm has very low p values for most datasets.

From Table 4, it can be observed that ReliefF-RBGSA has very low p values for all datasets when compared with Relief-BGSA

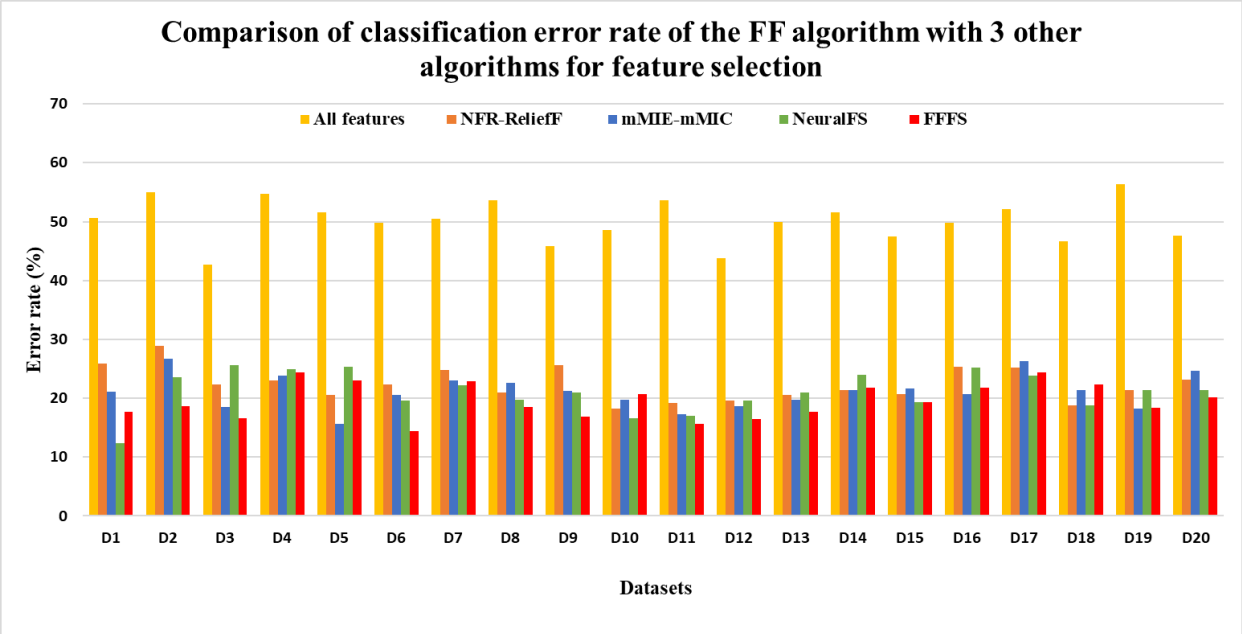


Figure 4: Classification error rate

According to table 3, considering the average scores in the datasets, the FFFS algorithm eventually shows the best performance in minimum error rate. Comparing the error rate of different algorithms, the proposed algorithm shows a lower error rate than other algorithms in most datasets. In terms of ERR, the proposed FFFS

algorithm achieves the best performance on the D2, D3, D6, D8, D9, D11, D12, D13, D15, and D20 datasets.

Following the proposed solution, the NeuralFS algorithm was ranked second in the error rate. This algorithm has the best performance in D1, D7, D10, D17 andD18 datasets.

The summary of the results presented in tables 3 and 4 and figure 4 are as follows:

1. The results obtained by almost all the algorithms are better than the original dataset for classification.
2. ERR illustrates that the FFFS performs better than other feature selection algorithms on most of the datasets.
3. By using feature selection algorithms in all datasets, the ERR was reduced to more than half of all the feature columns.
4. Concerning standard deviations, the FFFS algorithm in most of the datasets achieved more robust results against other comparison algorithms.
5. The more the standard deviation indicates the decrement of the robustness of the algorithm, it reduces the reliability of the algorithm.
6. Although the mMIE-mMIC algorithm has complex formulation and high calculations, it failed to reach a significant reduction in error rates and robustness.
7. While the proposed algorithm shows an appropriate and acceptable performance in all the datasets, the mMIE-mMIC algorithm has better performance in big datasets and despite the mMIE-mMIC algorithm, the NeuralFS algorithm has the lowest error rate in small datasets.
8. Table 4 compares the results of the FFFS with NFR-Relief, mMIE-mMIC, and NeuralFS algorithm, where bold values represent p-value >= 0.05.
9. According to a normative analysis of Wilcoxon test results, the outcome of the proposed algorithm is meaningfully distinguishable from others. Considering error, the FFFS algorithm performs significantly better than the competing algorithms.

5.5.2 F-MEASURE

Tables 5 display the average F-MEASURE obtained by different feature selection algorithms on all the datasets. For better comparison, we multiplied the numbers by 100.

Table 5: The F-MEASURE rate and standard deviation comparing results (Note that rank of each algorithm is enclosed in the parenthesis and the best results are given in bold)

F-MEASURE (mean±STD %)						
No.	**Dataset**	**All features**	**NFR-Relief [59]**	**mMIE-mMIC [60]**	**NeuralFS [61]**	**FFFS**
1	D1	53.62±2.35 (5)	67.58±1.20 (4)	71.36±1.29 (2)	70.68±1.32 (3)	**74.65±1.02 (1)**
2	D2	55.29±2.19 (5)	70.25±1.35 (3)	68.23±0.65 (4)	**72.12±0.45 (1)**	71.62±0.56 (2)
3	D3	45.95±1.98 (5)	**69.22±1.22 (1)**	66.02±1.30 (3)	66.27±0.42 (4)	68.16±0.39 (2)
4	D4	52.65±1.56 (5)	63.97±1.57 (4)	65.65±1.75 (3)	**70.52±0.67 (1)**	68.55±0.23 (2)
5	D5	43.78±0.99 (5)	**58.78±1.42 (1)**	57.59±0.95 (4)	58.00±0.84 (3)	58.62±0.96 (2)
6	D6	53.82±1.54 (5)	68.51±0.99 (4)	71.62±0.81 (3)	**78.94±0.65 (1)**	75.35±0.20 (2)
7	D7	45.66±1.83 (5)	65.63±0.62 (3)	65.49±0.56 (4)	**70.45±1.11 (1)**	68.49±0.58 (2)
8	D8	52.12±2.55 (5)	59.02±0.35 (4)	59.65±0.98 (3)	60.27±0.59 (2)	**63.65±0.13 (1)**
9	D9	51.65±1.95 (5)	58.32±2.01 (4)	61.82±1.25 (2)	**62.34±0.92 (1)**	61.71±0.45 (3)
10	D10	50.68±2.94 (5)	63.75±1.60 (4)	67.61±1.02 (2)	**69.85±0.75 (1)**	63.96±0.27 (3)
11	D11	49.55±1.33 (5)	69.92±1.44 (2)	**72.81±1.31 (1)**	63.45±1.05 (4)	64.37±0.33 (3)
12	D12	47.82±1.80 (5)	60.65±2.23 (4)	61.74±1.67 (3)	65.38±1.52 (2)	**67.63±1.30 (1)**
13	D13	50.73±2.62 (5)	63.22±1.00 (4)	65.16±0.96 (3)	**70.35±0.54 (1)**	68.83±0.71 (2)
14	D14	56.68±1.95 (5)	62.51±1.85 (4)	**70.58±0.81 (1)**	68.96±0.99 (2)	64.94±1.23 (3)
15	D15	54.31±1.67 (5)	64.26±0.96 (3)	58.32±0.75 (4)	67.91±0.86 (2)	**71.49±0.53 (1)**
16	D16	52.12±2.49 (5)	**70.85±0.35 (1)**	69.42±0.78 (2)	68.60±0.80 (4)	68.64±0.02 (3)
17	D17	50.95±1.73 (5)	64.00±1.02 (4)	**71.32±1.11 (1)**	64.94±1.22 (3)	70.13±0.54 (2)
18	D18	56.34±1.85 (5)	63.17±0.55 (4)	64.64±0.59 (3)	**68.62±0.99 (1)**	68.50±0.52 (2)
19	D19	48.91±2.21 (5)	**58.58±0.98 (1)**	54.67±0.48 (4)	58.00±0.45 (2)	56.27±0.71 (3)
20	D20	57.75±1.56 (5)	65.54±0.69 (4)	68.53±0.51 (3)	**72.32±1.03 (1)**	71.92±1.21 (2)
	Average rank	5.00 (5)	3.15 (4)	2.75 (3)	**2.00 (1)**	2.11 (2)

Table 5 shows that the proposed algorithm was ranked second with a negligible difference against the NFR-Relief, mMIE-mMIC, and NeuralFS algorithms in most datasets in terms of F-MEASURE rate. The best results are indicated by bold values.

Table 6: P-value of the Wilcoxon rank-sum test for the F-MEASURE rate based on FFFS and tree other feature selection algorithms (p-values >=0.5 are in bold)

No.	**FFFS vs.**	**NFR-Relief [59]**	**mMIE-mMIC [60]**	**NeuralFS [61]**
1	D1	6.07E-06	1.62E-04	3.81E-04
2	D2	4.52E-06	6.15E-04	**4.92E-01**
3	D3	**1.74E-01**	5.24E-05	5.044E-05
4	D4	3.85E-07	8.45E-06	**2.01E-02**
5	D5	**3.55E-02**	6.74E-03	1.37E-07
6	D6	4.74E-03	2.34E-05	**1.93E-02**
7	D7	1.51E-07	5.53E-06	**7.62E-01**
8	D8	4.62E-05	2.68E-03	4.95E-04
9	D9	5.38E-07	7.16E-03	8.13E-03
10	D10	8.72E-04	**4.94E-02**	**3.57E-02**
11	D11	**3.17E-02**	**2.57E-01**	8.35E-09
12	D12	6.43E-09	1.70E-07	3.24E-02
13	D13	4.35E-12	6.24E-11	5.74E-10
14	D14	8.76E-08	**3.13E-02**	1.24E-07
15	D15	1.27E-07	8.64E-05	**4.08E-02**

16	D16	**0.18E-02**	7.62E-04	7.25E-06
17	D17	8.22E-10	**2.62E-02**	9.68E-12
18	D18	2.18E-10	7.75E-09	3.48E-08
19	D19	**4.83E-01**	4.61E-06	9.15E-05
20	D20	9.34E-11	7.62E-11	6.82E-10

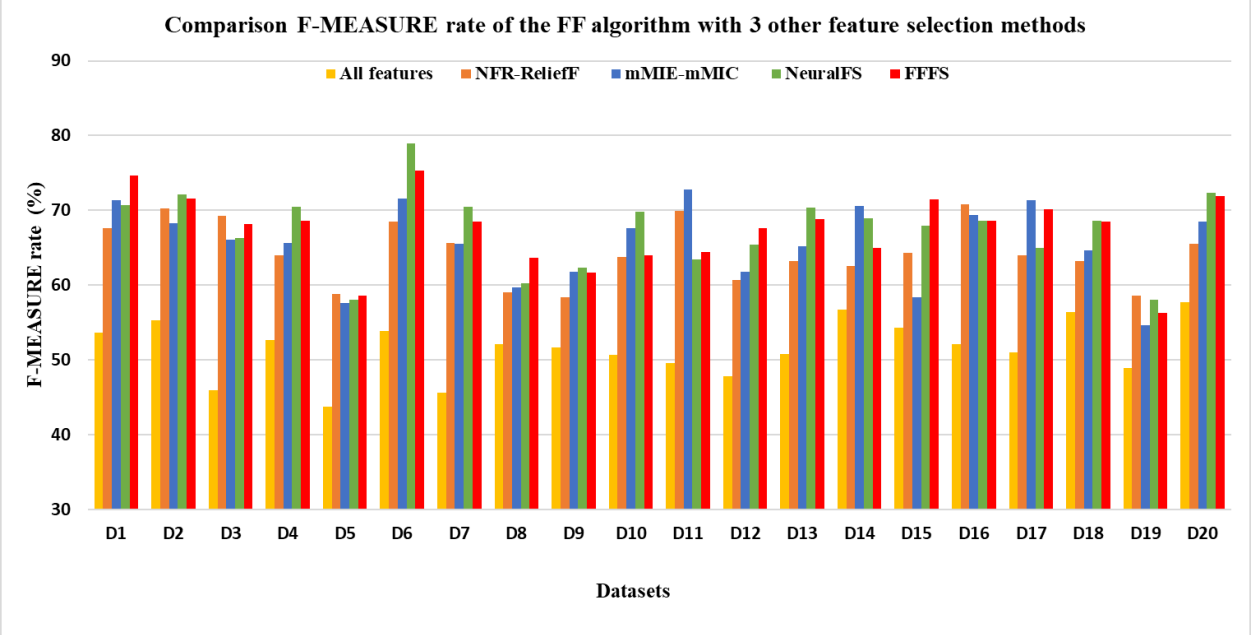


Figure 5: F-MEASURE rate

According to table 5, considering the average scores in the datasets, the NeuralFS algorithm had the best performance in the maximum F-MEASURE rate. However, the proposed algorithm was ranked second with a negligible difference. In terms of F-MEASURE, the proposed FFFS algorithm achieves the best performance on the D1, D8, D12, and D15 datasets and in the rest of the datasets, it usually ranks second.

The summary of the results presented in table 5 and table 6 and figure 5 is as follows:

1. The results obtained by almost all the algorithms are better than the original dataset for classification.
2. F-MEASURE illustrates that the proposed algorithm almost had satisfactory results against NFR-Relief and mMIE-mMIC algorithms.
3. The accuracy of the experiment is significant in the NeuralFS and FFFS algorithms, which means that the features selected by these algorithms are more suitable for classification.
4. The low standard deviation in the results of the proposed algorithm shows that is a robust and reliable algorithm for feature selection.
5. Compared algorithms usually have a higher standard deviation than the proposed algorithm, which reduces their robustness.
6. The Wilcoxon test measure shows the superiority of the proposed algorithm on the NFR-Relief, mMIE-mMIC algorithm, but the FFFS algorithm had similar performance compared to the NeuralFS algorithm.

5.5.3 **Time Consuming**

Table 7 indicated the average CPU time measured in seconds per run, taken by the FFFS algorithm and three other algorithms. This measure indicates the speed of the algorithms in selecting features from a given dataset. The results are graphically shown in figure 6.

Table 7: Comparison between FFFS with other algorithms based on time (sec)

TIME CONSUMING						
No.	**Dataset**	**All features**	**NFR-Relief [59]**	**mMIE-mMIC [60]**	**NeuralFS [61]**	**FFFS**
1	D1	572 (5)	275 (2)	321 (3)	346 (4)	**270 (1)**
2	D2	436 (5)	**241 (1)**	252 (2)	279 (4)	266 (3)
3	D3	456 (5)	**190 (1)**	315 (4)	302 (3)	201 (2)
4	D4	401 (5)	243 (2)	**242 (1)**	286 (4)	246 (3)
5	D5	393 (5)	248 (4)	212 (3)	**196 (1)**	211 (2)
6	D6	580 (5)	**298 (1)**	342 (3)	365 (4)	300 (2)
7	D7	482 (5)	307 (2)	322 (3)	349 (4)	**265 (1)**
8	D8	472 (5)	**264 (1)**	289 (4)	285 (3)	280 (2)
9	D9	512 (5)	326 (2)	360 (3)	370 (4)	**312 (1)**
10	D10	301 (5)	**212 (1)**	242 (2)	286 (4)	246 (3)
11	D11	562 (5)	**273 (1)**	329 (4)	301 (3)	280 (2)
12	D12	626 (5)	**305 (1)**	361 (4)	329 (2)	342 (3)
13	D13	764 (5)	**521 (1)**	598 (3)	627 (4)	535 (2)
14	D14	575 (5)	422 (3)	**376 (1)**	455 (4)	378 (2)
15	D15	323 (5)	236 (2)	275 (3)	305 (4)	**218 (1)**
16	D16	677 (5)	**196 (1)**	255 (2)	394 (4)	286 (3)
17	D17	532 (5)	327 (3)	242 (2)	**225 (1)**	359 (4)
18	D18	480 (5)	**243 (1)**	267 (2)	298 (3)	313 (4)
19	D19	672 (5)	291 (2)	344 (3)	378 (4)	**262 (1)**
20	D20	533 (5)	**201 (1)**	257 (2)	293 (4)	235 (3)
	Average rank	5.00 (5)	**1.65 (1)**	2.70 (3)	3.40 (4)	2.25 (2)

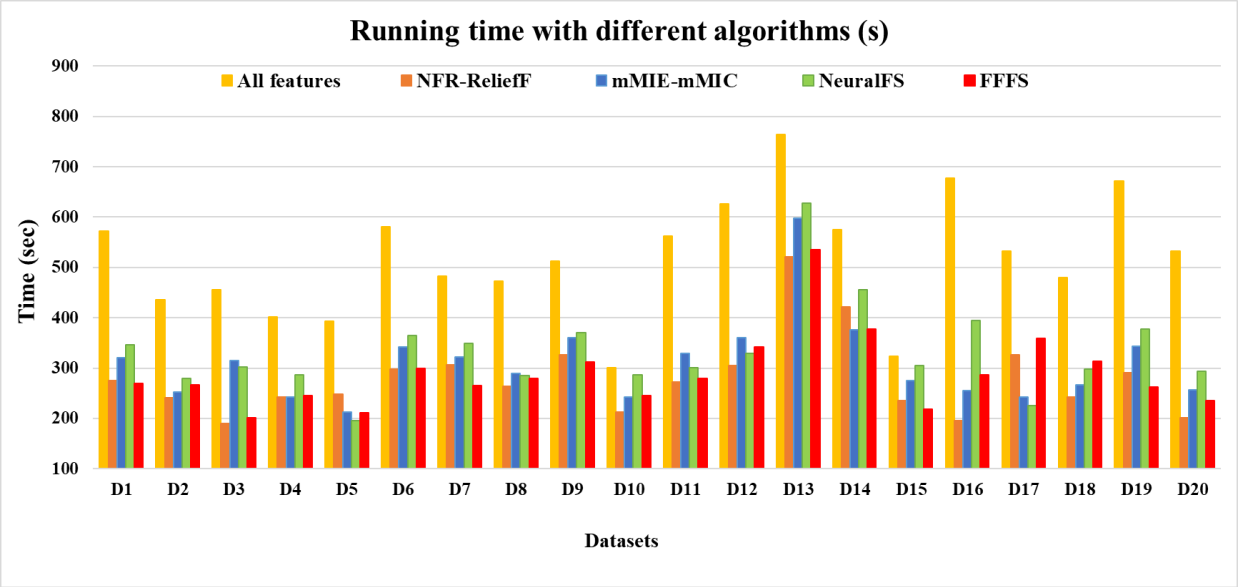


Figure 6: Time consuming

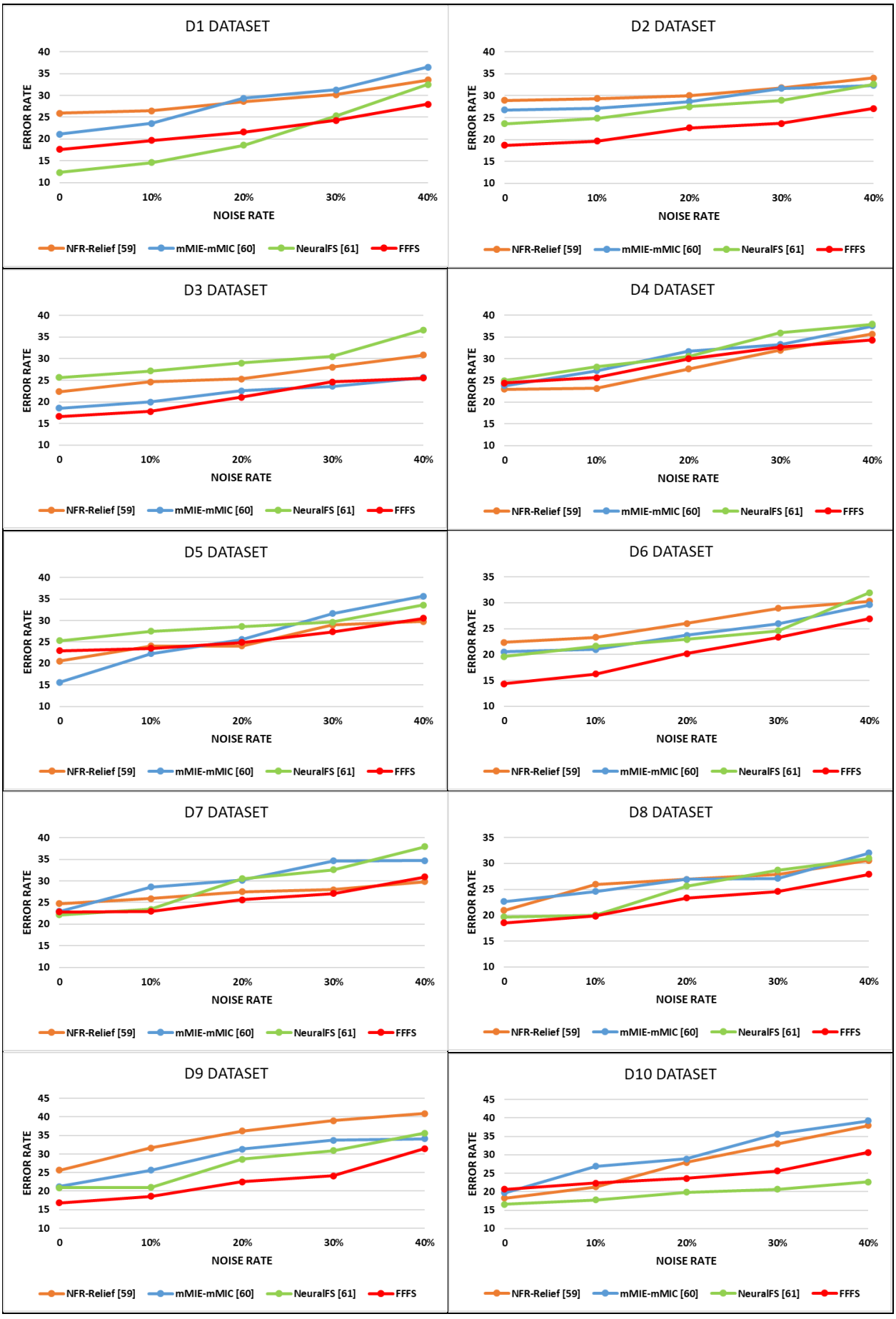
The summary of the results presented in table 7 and figure 6 is as follows:

1. The time consumed in the NFR-Relief algorithm was the least and the proposed algorithms have achieved second place in the least time-consuming.
2. The processing time of the FFFS algorithm according to the first rank in the lowest error rate is quite logical and acceptable.
3. The mMIE-mMIC and NeuralFS algorithms, due to their complexity, could not have a good execution time.

5.5.4 Noise Robustness

In order to show the robustness of the proposed algorithm, some noise was added to all datasets since real-world data always have noise.

Figure 7 depicts the convergence performance of the proposed algorithm in comparison with the other three algorithms with different noise rates in datasets. The horizontal axis (x-axis) indicates noise level, while the vertical (y-axis) represents the error rate percentage. The results shown in figure 7 indicate that the proposed algorithm outperforms the other algorithms. The most resistant algorithm (the green one) is the FFFS algorithm, which gets the lowest increment in error rate by increasing the noise rate. It means that the proposed algorithm has selected better features for classification.


D1 DATASET
D2 DATASET
D3 DATASET
D4 DATASET
D5 DATASET
D6 DATASET
D7 DATASET
D8 DATASET
D9 DATASET
D10 DATASET
ERROR RATE
NOISE RATE
0
10%
20%
30%
40%
NFR-Relief [59]
mMIE-mMIC [60]
NeuralFS [61]
FFFS

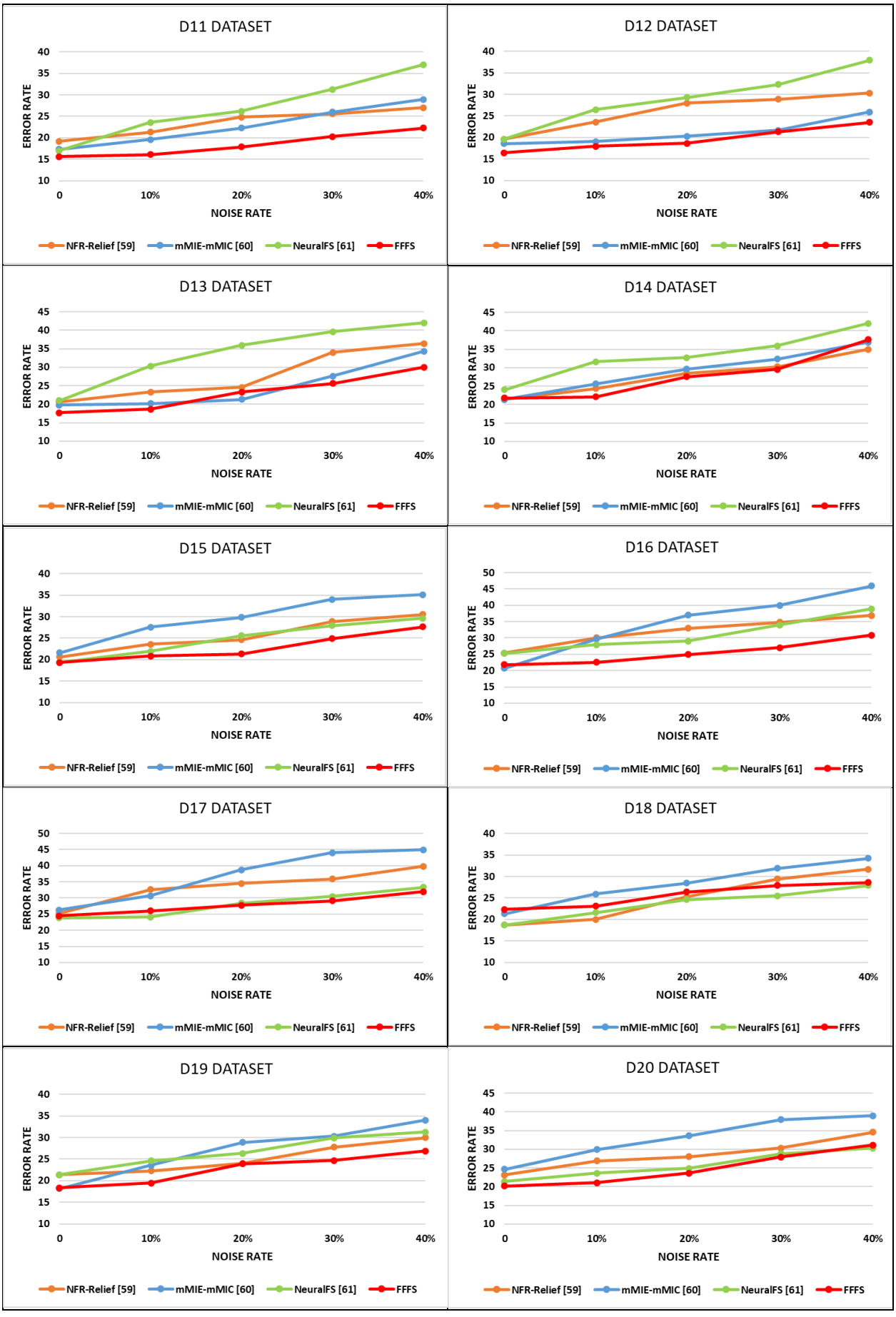

D11 DATASET
D12 DATASET
D13 DATASET
D14 DATASET
D15 DATASET
D16 DATASET
D17 DATASET
D18 DATASET
D19 DATASET
D20 DATASET
ERROR RATE
NOISE RATE
0
10%
20%
30%
40%
NFR-Relief [59]
mMIE-mMIC [60]
NeuralFS [61]
FFFS

Figure 7: Comparing FFFS performance with three other algorithms against noise

The summary of the results presented in figure 7 is as follows:

1- Experimental results on datasets demonstrate that FFFS has more robustness for datasets with noisy features.

2- The mMIE-mMIC algorithm, due to its complexity, could not have good robustness against noise in small datasets and it is more suitable for larger datasets. But, in most cases, the proposed algorithm performs better against noise and its error rate increase difference is less than the mMIE-mMIC algorithm.

3- The slope of the NeuralFS line in the graphs, indicates that it is suitable for small datasets. However, it still does not have the noise robustness as much as the FFFS algorithm.

4- NFR-Relief algorithm has a steady upward trend in most datasets. This algorithm works regardless of the dataset size and can be used when we are unaware of the nature of the dataset.

6. Conclusion and Future Works

Ever-evolving frontier technologies make huge and complex datasets. So, feature selection algorithms are necessary to refine the information and restrict them only to the relevant and useful features which are important to machine learning tasks.

In this study, a new algorithm is presented for feature selection according to the filled function and the Fisher score. The performance of the proposed algorithm was assessed on 15 datasets from the UCI data repository and 5 medical datasets. Furthermore, the proposed algorithm was compared against state-of-the-art supervised filter feature selection algorithms that are NFR-Relief, mMIE-mMIC, and NeuralFS algorithms.

The use of a filled function alongside the Fisher score to design a filter feature selection algorithm is a well-established trend and as evident from this work which brings fruitful results.

The experiments revealed that although most of the results obtained by the proposed algorithm do not significantly improve the results obtained by the other supervised feature selection algorithms, the FFFS algorithm in general obtained better results regarding the other compared algorithms, and it was able to select features subsets with low redundancy. A better feature set that was selected by the FFFS algorithm,

caused the classification error rate to improve since some redundant features were eliminated. It should be noted that the proposed algorithm could obtain more robust against noise increment results than other comparing algorithms due to selecting better features.

Speeding up the proposed algorithm by applying different alternatives, such as the use of random algorithms, or the intent to propose a new search algorithm to tune the thresholds in different filter algorithms, and perform extensive experimentations using sensitivity analysis can be studied in the future.

Compliance with ethical standards

Conflict of interest: The authors declare that they have no conflict of interest.

Ethical approval: This article does not contain any studies with animals performed by any of the authors.

Informed consent was obtained from all individual participants included in the study.

References

1. Nguyen, B.H., B. Xue, And M. Zhang, A Survey On Swarm Intelligence Approaches To Feature Selection In Data Mining. Swarm And Evolutionary Computation, 2020. 54: P. 100663.
.2 Chen, Z., Et Al., Feature Selection May Improve Deep Neural Networks For The Bioinformatics Problems. Bioinformatics, 2020. 36(5): P. 1542-1552.
.3 Toğaçar, M., Z. Cömert, And B. Ergen, Classification Of Brain Mri Using Hyper Column Technique With Convolutional Neural Network And Feature Selection Method. Expert Systems With Applications, 2020. 149: P. 113274.
.4 Kushwaha, P. And R.R.J.I. Welekar, Feature Selection For Image Retrieval Based On Genetic Algorithm. 2016. 4(2): P. 16-21.
.5 Pes, B., N. Dessì, And M. Angioni, Exploiting The Ensemble Paradigm For Stable Feature Selection: A Case Study On High-Dimensional Genomic Data. Information Fusion, 2017. 35: P. 132-147.
.6 Atkinson, J. And D.J.E.S.W.A. Campos, Improving Bci-Based Emotion Recognition By Combining Eeg Feature Selection And Kernel Classifiers. 2016. 47: P. 35-41.
.7 Yadav, S., A. Ekbal, And S. Saha, Feature Selection For Entity Extraction From Multiple Biomedical Corpora: A Pso-Based Approach. Soft Computing, 2018. 22(20): P. 6881-6904.
.8 Kashani, S.M.Z. And J. Hamidzadeh, Feature Selection By Using Privacy-Preserving Of Recommendation Systems Based On Collaborative Filtering And Mutual Trust In Social Networks. Soft Computing, 2019: P. 1-16.
.9 Mahindru, A. And A. Sangal, Semidroid: A Behavioral Malware Detector Based On Unsupervised Machine Learning Techniques Using Feature Selection Approaches. International Journal Of Machine Learning And Cybernetics, 2020: P. 1-43.
.10 Manbari, Z., F. Akhlaghiantab, And C. Salavati, Hybrid Fast Unsupervised Feature Selection For High-Dimensional Data. Expert Systems With Applications, 2019. 124: P. 97-118.
.11 Bommert, A., Et Al., Benchmark For Filter Methods For Feature Selection In High-Dimensional Classification Data. Computational Statistics & Data Analysis, 2020. 143: P. 106839.
.12 Hamidzadeh, J., R. Monsefi, And H.S. Yazdi, Irahc: Instance Reduction Algorithm Using Hyperrectangle Clustering. Pattern Recognition, 2015. 48(5): P. 1878-1889.
.13 Kelidari, M. And J. Hamidzadeh, Feature Selection By Using Chaotic Cuckoo Optimization Algorithm With Levy Flight, Opposition-Based Learning And Disruption Operator. Soft Computing, 2021. 25(4): P. 2911-2933.

.14 Song, X.-F., Et Al., Variable-Size Cooperative Coevolutionary Particle Swarm Optimization For Feature Selection On High-Dimensional Data. Ieee Transactions On Evolutionary Computation, 2020.
.15 Ghosh, M., Et Al., A Wrapper-Filter Feature Selection Technique Based On Ant Colony Optimization. Neural Computing And Applications, 2019: P. 1-19.
.16 Araújo, A.F., V.O. Antonino, And K.L. Ponce-Guevara, Self-Organizing Subspace Clustering For High-Dimensional And Multi-View Data. Neural Networks, 2020. 130: P. 253-268.
.17 Dai, J., Et Al., Fast Feature Selection For Interval-Valued Data Through Kernel Density Estimation Entropy. International Journal Of Machine Learning And Cybernetics, 2020.
.18 Li, Y., Et Al., A Novel Feature Learning Framework For High-Dimensional Data Classification. International Journal Of Machine Learning And Cybernetics, 2020: P. 1-15.
.19 Hancer, E., B. Xue, And M. Zhang, A Survey On Feature Selection Approaches For Clustering. Artificial Intelligence Review, 2020. 53(6): P. 4519-4545.
.20 Tiwari, S.R. And K.K. Rana, Feature Selection In Big Data: Trends And Challenges, In Data Science And Intelligent Applications. 2021, Springer. P. 83-98.
.21 Rao, H., Et Al., Feature Selection Based On Artificial Bee Colony And Gradient Boosting Decision Tree. Applied Soft Computing, 2019. 74: P. 634-642.
.22 Kelidari, M. And H. Javad, Feature Selection By Using Chaotic Cuckoo Optimization Algorithm With Levy Flight, Opposition-Based Learning And Disruption Operator. Soft Computing, 2020: P. 1-23.
.23 Li, Y., T. Li, And H. Liu, Recent Advances In Feature Selection And Its Applications. Knowledge And Information Systems, 2017: P. 1-27.
.24 Da Silva, D.L., L.M. Seijas, And C.J. Bastos-Filho, Artificial Bee Colony Optimization For Feature Selection Of Traffic Sign Recognition. International Journal Of Swarm Intelligence Research (Ijsir), 2017. 8(2): P. 50-66.
.25 Ramírez-Gallego, S., Et Al., A Survey On Data Preprocessing For Data Stream Mining: Current Status And Future Directions. Neurocomputing, 2017. 239: P. 39-57.
.26 Lee, J. And D.-W. Kim, Efficient Multi-Label Feature Selection Using Entropy-Based Label Selection. Entropy, 2016. 18(11): P. 405.
.27 Solorio-Fernández ,S., J.A. Carrasco-Ochoa, And J.F. Martínez-Trinidad, A Review Of Unsupervised Feature Selection Methods. Artificial Intelligence Review, 2019: P. 1-42.
.28 Zare, M., M. Eftekhari, And G. Aghamollaei, Supervised Feature Selection Via Matrix Factorization Based On Singular Value Decomposition. Chemometrics And Intelligent Laboratory Systems, 2019. 185: P. 105-113.
.29 Manbari, Z., F. Akhlaghiantab, And C.J.E.S.W.A. Salavati, Hybrid Fast Unsupervised Feature Selection For High-Dimensional Data. 2019. 124 :P. 97-118.
.30 Kakisim, A.G. And I.J.E.S.W.A. Sogukpinar, Unsupervised Binary Feature Construction Method For Networked Data. 2019. 121: P. 256-265.
.31 Shang, R., Et Al., Unsupervised Feature Selection Based On Self-Representation Sparse Regression And Local Similarity Preserving. 2017: P. 1-14.
.32 Liu, K., Et Al., Rough Set Based Semi-Supervised Feature Selection Via Ensemble Selector. 2019. 165: P. 282-296.
.33 Luo, T., Et Al., Semi-Supervised Feature Selection Via Insensitive Sparse Regression With Application To Video Semantic Recognition. Ieee Transactions On Knowledge And Data Engineering, 2018. 30(10): P. 1943-1956.
.34 Yang, X.-K., Et Al., Semi-Supervised Minimum Redundancy Maximum Relevance Feature Selection For Audio Classification. Multimedia Tools And Applications, 2018. 77(1): P. 713-739.
.35 Solorio-Fernández, S., J.A. Carrasco-Ochoa, And J.F. Martínez-Trinidad, A Review Of Unsupervised Feature Selection Methods. Artificial Intelligence Review, 2020. 53(2): P. 907-948.
.36 Liaghat, S., E.G.J.I.J.O.M.L. Mansoori, And Cybernetics, Filter-Based Unsupervised Feature Selection Using Hilbert–Schmidt Independence Criterion. 2018: P. 1-16.
.37 Cekik, R. And A.K. Uysal, A Novel Filter Feature Selection Method Using Rough Set For Short Text Data. Expert Systems With Applications, 2020. 160: P. 113691.
.38 Labani, M., Et Al., A Novel Multivariate Filter Method For Feature Selection In Text Classification Problems. Engineering Applications Of Artificial Intelligence, 2018. 70: P. 25-37.
.39 Lazar, C ,.Et Al., A Survey On Filter Techniques For Feature Selection In Gene Expression Microarray Analysis. 2012. 9(4): P. 1106-1119.
.40 Cilia, N.D., Et Al., A Ranking-Based Feature Selection Approach For Handwritten Character Recognition. 2019. 121: P. 77-86.
.41 Chandrashekar, G., F.J.C. Sahin, And E. Engineering, A Survey On Feature Selection Methods. 2014. 40(1): P. 16-28.
.42 Fukunaga, K., Introduction To Statistical Pattern Recognition. 2013: Elsevier.

.43 Miao, J. And L.J.P.C.S. Niu, A Survey On Feature Selection. 2016. 91: P. 919-926.
.44 Zhu, R., F. Dornaika, And Y.J.N.N. Ruichek, Learning A Discriminant Graph-Based Embedding With Feature Selection For Image Categorization. 2019. 111: P. 35-46.
.45 Ma, W., Et Al., A Two-Stage Hybrid Ant Colony Optimization For High-Dimensional Feature Selection. Pattern Recognition, 2021: P. 107933.
.46 Zhou, Y., Et Al., A Problem-Specific Non-Dominated Sorting Genetic Algorithm For Supervised Feature Selection. Information Sciences, 2021. 547: P. 841-859.
.47 Hussain ,K., Et Al., An Effcient Hybrid Sine-Cosine Harris Hawks Optimization For Low And High-Dimensional Feature Selection. Expert Systems With Applications, 2021: P. 114778.
.48 Ouadfel, S. And M. Abd Elaziz, Enhanced Crow Search Algorithm For Feature Selection. Expert Systems With Applications, 2020. 159: P. 113572.
.49 Sharma, H. And R. Agarwal, Updated Frequency-Based Bat Algorithm (Ufbba) For Feature Selection And Vote Classifier In Predicting Heart Disease, In Advances In Computer, Communication And Computational Sciences. 2021, Springer. P. 449-460.
.50 Mandal, A.K., Et Al., An Approach Of Feature Subset Selection Using Simulated Quantum Annealing, In Data Management, Analytics And Innovation. 2021, Springer. P. 133-146.
.51 Renpu, G.J.M.P., A Filled Function Method For Finding A Global Minimizer Of A Function Of Several Variables. 1990. 46(1-3): P. 191-204.
.52 Wu, X., Y. Wang, And N. Fan, A New Filled Function Method Based On Adaptive Search Direction And Valley Widening For Global Optimization. Applied Intelligence, 2021: P. 1-21.
.53 Sui, X., Y. Wang, And J. Liu, A New Filled Function Method Combining Auxiliary Function For Global Optimization. Pacific Journal Of Optimization, 2019. 15(1): P. 23-44.
.54 Liu, J., Et Al., A Filled Flatten Function Method Based On Basin Deepening And Adaptive Initial Point For Global Optimization. International Journal Of Pattern Recognition And Artificial Intelligence, 2020. 34(04): P. 2059011.
.55 Wang, Y.-P. And D.-L. Liu, A Global Optimization Evolutionary Algorithm And Its Convergence Based On A Smooth Scheme And Line Search. Chinese Journal Of Computers-Chinese Edition-, 2006. 29(4): P. 670.
.56 Verma, A.K., I. Saini, And B.S. Saini, A New Bat Optimization Algorithm Based Feature Selection Method For Electrocardiogram Heartbeat Classification Using Empirical Wavelet Transform And Fisher Ratio. International Journal Of Machine Learning And Cybernetics, 2020.
.57 Lichman, M., Uci Machine Learning Repository. University Of California, School Of Information And Computer Science, Irvine, Ca (2013). 2017.
.58 Statnikov, A., Et Al., A Comprehensive Evaluation Of Multicategory Classification Methods For Microarray Gene Expression Cancer Diagnosis. Bioinformatics, 2005. 21(5): P. 631-643.
.59 Munirathinam, D.R. And M. Ranganadhan, A New Improved Filter-Based Feature Selection Model For High-Dimensional Data. The Journal Of Supercomputing, 2020. 76(8): P. 5745-5762.
.60 Zheng, K., Et Al., Feature Subset Selection Combining Maximal Information Entropy And Maximal Information Coefficient. Applied Intelligence, 2020. 50(2): P. 487-501.
.61 Huang, Y., Et Al., Supervised Feature Selection Through Deep Neural Networks With Pairwise Connected Structure. Knowledge-Based Systems, 2020. 204: P. 106202.
.62 Sun, H., Et Al., Feature Selection Combining Filter And Wrapper Methods For Motor-Imagery Based Brain–Computer Interfaces. International Journal Of Neural Systems, 2021: P. 2150040.
.63 Salesi, S., G. Cosma, And M. Mavrovouniotis, Taga: Tabu Asexual Genetic Algorithm Embedded In A Filter/Filter Feature Selection Approach For High-Dimensional Data. Information Sciences, 2021. 565: P. 105-127.
.64 Effrosynidis, D. And A. Arampatzis, An Evaluation Of Feature Selection Methods For Environmental Data. Ecological Informatics, 2021. 61: P.101224 .
.65 Li, W., Et Al., Embedded Feature Selection Based On Relevance Vector Machines With An Approximated Marginal Likelihood And Its Industrial Application. Ieee Transactions On Systems, Man, And Cybernetics: Systems, 2021.
.66 Jesus, J., A. Canuto, And D. Araujo, An Exploratory Analysis Of Data Noisy Scenarios In A Pareto-Front Based Dynamic Feature Selection Method. Applied Soft Computing, 2021. 100: P. 106951.