International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 13 (2021) pp. 010-019

Received: 14 September 2020; Accepted: 25 February, 2021; Published: 20 March, 2021

An Adaptive Multi-levels Sequential Feature Selection

Knitchepon Chotchantarakun[1] and Ohm Sornil[2]

[1] Graduate School of Applied Statistics (GSAS), National Institute of Development Administration (NIDA), Serithai Road, Klong-Chan, Bangkapi, Bangkok 10240, Thailand
knitchepon@gmail.com

[2] Graduate School of Applied Statistics (GSAS), National Institute of Development Administration (NIDA), Serithai Road, Klong-Chan, Bangkapi, Bangkok 10240, Thailand
osornil@as.nida.ac.th

***Abstract*: Dealing with a large amount of data becomes a major challenge in data mining and machine learning. Feature selection is a significant preprocessing step for selecting the most informative features by removing irrelevant and redundant features, especially for the large datasets. These selected features play an important role in information searching and enhance the performance of a machine learning model such as classification and prediction. There have been several strategies proposed during the past few decades. In this study, we have proposed a new technique called Multi-levels Forward Inclusion (MLFI). The proposed algorithm consists of two parts. The first part is aimed to search for the maximum classification accuracy by applying the multi-levels forward-searching technique. The second part provides an improvement on the previous result by replacing a week feature. Hence, the idea is to apply an adaptive multi-levels forward search method and a replacement step during the feature addition without any backtracking search. However, we need to limit the level of forward-searching to maintain a lower execution time by introducing an adaptive variable called the generalization limit. We have tested our algorithm on eight UCI datasets and compare their accuracy with standard methods. MLFI shows better results than the other sequential forward floating techniques for the majority of the tested datasets.**

***Keywords*:** Classification accuracy, Data mining, Dimensionality reduction, Feature selection, Sequential search, Supervised learning.

I. Introduction

In the data analysis task, a large amount of data can be a high dimensional dataset which directly affects performance because some irrelevant and redundant features also make some contribution to the analysis. To overcome the problem, those irrelevant and redundant features should be eliminated which lead to the more effective dimensions. This data preprocessing step is called feature selection. Generally, the goal of feature selection is to determine the best subset of features for conducting statistical analysis or building a machine learning model. Feature selection assists in selecting the minimum features from the whole dataset. These features are useful for finding accurate data models. To ensure the optimal feature subset, a feature selection method has to evaluate a total of $2^n - 1$ subsets, where n is the total number of features in the dataset. Even though an exhaustive search for optimal feature subset results to an optimal solution, but it is not practical especially for a moderately large n. This type of problem is said to be an NP-hard problem, as a result, many search strategies have been proposed in the literature for suboptimal solutions. Feature selection algorithm can be classified in many different ways. The most common one can be categorized into three types which are filter approach, wrapper approach, and embedded approach [1]–[3].

Filter approach uses an independent criterion function to select the feature without depending upon the type of classifier used which leads to the simplicity of the method, whereas the interactions with classifier and feature dependencies are ignored. Filter method ranks each individual feature according to the measurement such as information, distance, or similarity. It only considers the association between the feature and the class label. The nature of this method results in a drawback that each feature is considered separately.

Wrapper approach uses the result of the classifier to determine the goodness of the given feature, therefore the selected features are dependent on the classification algorithm. This method removes the disadvantage of the filter approach by the consideration of feature dependency whereas it is more time consuming than the filter approach. The quality of the feature is directly related to the performance of the classifier.

Embedded approach searches for an optimal feature subset during the model training that is built into the classifier construction. It returns both the learned model and selected features simultaneously. The benefit of this method is that it takes less computational time than the wrapper approach. This method is also called the hybrid model. It incorporates with learning algorithm and optimized for higher accuracy. The embedded approach utilizes a filter-based technique to select highly representative features and then apply a wrapper-based technique to add candidate features. The candidate subsets are

evaluated for selecting the best ones. It does not only reduce the dimensionality of the dataset but also decreases the computational time and improves the performance. Somol, Novovicova, and Pudil [4] proposed a flexible hybrid sequential forward floating selection (hSFFS) by employing an evaluation function to filter some features and using a wrapper criterion to identify the optimal feature subset. The main benefit of this method is the ability to trade off the resulting quality with the computational cost in order to enable the wrapper-based selection in high dimensional datasets. Their experimental results show promising classification accuracy.

Mwadulo [5] has stated that a supervised feature selection normally uses in the classification problem by calculating the correlation between the feature and the class label. The supervised model aims to find an optimal feature subset that maximizes the classification accuracy. In the filter approach, to analyze the relevance and redundancy of feature-class and feature-feature respectively, need to use a model such as Euclidean distance, information measures, and Pearson correlation. For wrapper models, the classification error or accuracy rate used as the feature evaluation.

In this study, we have explored an effective way to improve classification accuracy of a machine learning model regarding sequential feature selection. In section II, we have discussed the related works. In section III, we have presented our proposed feature selection method. In section IV, we describe our implementation environment. Results and discussion have illustrated in section V. We have concluded our study with some further directions in section VI.

II. Related Works

A. Feature Selection Process

Feature selection becomes a necessary step in the data mining process because the high dimensionality and vast amount of data give rises to a challenge to the learning task. Many irrelevant features do not add much value during the learning process, hence learning models tend to become highly complicated and decrease learning accuracy. Feature selection is one effective way to identify relevant features for dimensionality reduction. However, the benefit of feature selection comes with an extra effort by trying to get an optimal subset that represents the original dataset. Jovic, Brkic, and Bogunovic [6] categorized feature selection methods into three common search strategies. Exponential algorithms evaluate subsets that grow exponentially with the feature space size, for example, Exhaustive search and Branch-and-bound. Sequential algorithms such as Sequential Forward Floating Selection (SFFS) [7] include or exclude features from the active subset sequentially. Random algorithms incorporate randomness into the search process to optimize the solution. An example of a random algorithm is Evolutionary computation algorithms using genetic or ant colony optimization.

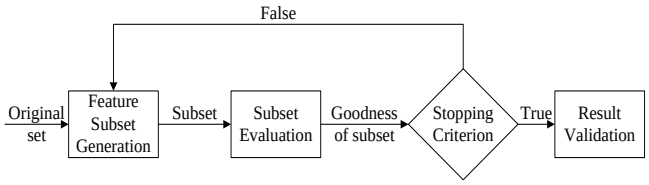

Figure 1. Feature Selection Process.

Figure. 1 shows a feature selection process. Feature subset generation produces candidate feature subsets for evaluation based on searching strategies. Subset evaluation is aimed to evaluate the subset generated from the previous procedure. Filter or wrapper method applies during this process. Stopping criterion may be based on a predefined number of selected features, the number of iterations reached, or feature addition does not produce any better subsets. The result validation is to compare results by using classifiers on each relevant attribute subset.

B. Sequential Forward Selection (SFS)

The SFS process in a forward search manner starts with an empty set and adds one feature to the selected subset during each iteration until discovering a new feature subset that maximizes the criterion function value. SFS is essential for constructing other more complex algorithms. Large datasets normally contain a lot of features whereas only some of them are significant for model training. The idea is to select a feature that gives the highest learning accuracy. Assume we have a set $Y = \{y_1, y_2, \ldots, y_D\}$, where D is the number of input dimensions. We want to find a subset $X_k = \{x_j \mid j = 1, 2, \ldots, k; x_j \in Y\}$, where $k = (0, 1, 2, \ldots, D)$, and d is the required subset size. Initialize $X_0 = \{\}$ and $k = 0$, and x^+ is an included feature where $x \in Y - X_k$. The algorithm can be explained below.

Step 1: Inclusion step.

$x^+ = \arg\max J(x_k = x)$, where $x \in Y - X_k$

$X_{k+1} = X_k + x^+$

$k = k + 1$

(Add a selected feature x^+ to the subset X_k, where x^+ is a feature that maximizes the criterion function (J).)

Step 2: Continue step 1 until d features are selected.

C. Sequential Forward Floating Selection (SFFS)

One of the most significant inventions in this area is the SFFS. This technique combines the concept of SFS with Sequential Backward Search (SBS). SBS giving it more effective than SFS by introducing the backtracking step. The SBS method starts with a full feature subset and eliminates a feature in each iteration until a predetermined criterion is satisfied. The backtracking step is the conditional step where the improvement can be made during the search process. SFFS is said to be a state-of-the-art method that is widely used in several applications. Researchers in sequential feature selection normally extend the study using SFFS as a base method to compare their results. To explain the SFFS algorithm below, let x^- be an excluded feature where $x \in X_k$.

Step 1: Inclusion step. (Apply SFS algorithm.)
Step 2: Conditional exclusion step. (This step is similar to the SBS algorithm.)
$x^- = arg\ max\ J(x_k = x)$, *where* $x \in X_k$
If $J(x_k - x^-) > J(x_{k-1})$:
$X_{k-1} = X_k - x^-$
$k = k - 1$
(Remove a feature if the resulting subset improves the performance. If $k \leq 2$ or there is no improvement, go to step 1, or else, repeat step 2.)
Step 3: Continue steps 1 and 2 until d features are selected.

After the introduction of SFFS, there are several improved versions have been proposed to obtain better performance. An adaptive version of the floating search method was presented [8]. The idea of Adaptive SFFS (ASFFS) is selecting features to add or remove more than one feature in each sequential step in order to search for a better subset. The number of search features in each step can be varied depending on the remaining features in the dataset. The result is a more thorough search with better chances to find the optimal solution by setting a higher generalization level. There are two free parameters, r_{max} and b in ASFFS which specify the generalization limit and range of the adaptive search. The parameter r specifies the number of features to be added in the forward phase or inclusion phase which is calculated adaptively. In the backward phase or exclusion phase, removes o features if it increases the performance. ASFFS is identical to SFFS if we assign $r_{max} = 1$. The suggestions for the two values are 4 and 3, respectively. The nearer the current subset size to d, the higher is the generalization limit. The reason behind this characteristic is to save time by limiting the generalization level while the current subset is still far from the desired one. The generalization level (r) increases when the number of features (k) in the current subset gets close to d until it reaches r_{max}. The ASFFS has shown better results than SFFS due to a more thorough search.

Calculation of r-value is done at the beginning of every forward and backward phase using the following conditions.

1. If $|k - d| < b$, let $r = r_{max}$
2. Else if $|k - d| < b + r_{max}$, let $r = r_{max} + b - |k - d|$
3. Else let $r = 1$

While the number of features is far away from the required subset size, r is assigned to 1 which is exactly like SFFS's procedure. When k is getting closer to d, the value for r is increasing but no more than r_{max}. Even though ASFFS has shown slightly better results than SFFS but it takes more computational time due to the complexity of the algorithm. The adaptive step leads to additional work to the SFFS structure both forward and backward directions. Elements of the current feature subset can be increased or decreased along the searching process that is another reason for a longer time required. The backtracking step explores features within the current subset only without considering the unselected features that are located outside the boundary. The generalization level can be helpful during the search only when k in the current subset is getting close to the target size, thus the detailed search concept works only when k almost reaches the end of the process.

D. Improve Forward Floating Selection (IFFS)

One remarkable improvement of SFFS is the IFFS algorithm [9], which had successfully removed the weakness of SFFS by adding an additional step to improve the criterion function value. From the fact that the best k-subset is unnecessary containing all the features from the best (k–1)-subset, therefore IFFS was introduced to solve this nesting problem. This improved step is called "replacing the weak feature" that is to check whether removing any feature in the currently selected feature subset and adding a new one at each sequential step can improve the current feature subset. IFFS can impressively solve the nesting effect of SFFS and the algorithm is simpler than ASFFS with lower computing time. IFFS yields better performance than both SFFS and ASFFS with a little longer process time than SFFS. IFFS algorithm has been explained below by applying the same variables as SFFS.

Step 1: Inclusion step. (Apply SFS algorithm.)
Step 2: Conditional exclusion step. (Apply SBS algorithm.)
Step 3: Checking if replacing the weak feature helps.
For x_i in X_k :
$X_{k-1} = X_k - x_i$
For x_j in $Y - X_{k-1}$:
$x_j = \arg\max J(x_j)$
If $J(X_{k-1} + x_j) > J(X_k)$:
$X_k = X_{k-1} + x_j$
(Generate k new subsets of k features by removing one feature and adding one feature using SFS. Calculate the J-values of k subsets. If the subset with the largest J-value gives an improvement, then replace the new subset to the current subset, and go to step 2. Otherwise, go to step 1.)
Step 4: Continue steps 1, 2 and 3 until d features are selected.

Another improved sequential search algorithm is the Sequential Deep Floating Forward Search (SDFFS) [10]. The deep searching step aims to confirm whether there exists a subset with k features being better than the current one that has been found so far by the SFS step which cannot be found in the SFFS algorithm. The experimental results show that SDFFS does not perform the best all the times for all feature's sizes. However, the results are relatively high accuracy and more stable. This leads to the overall performance is quite remarkable. One major drawback for SDFFS is the computational time that is far greater than those previous methods. SDFFS searches through the first 100 features with only slight chances to gain the accuracy especially those with low criterion function value, and thus modification of the algorithm should be concerned to make it more effective.

A recent study from Homsapaya, and Sornil [11] has introduced a floating search technique employing a genetic algorithm (GA) to improve the quality of the selected feature subset. The results show that GA improves the performance for the majority of sample datasets. Kadhum, Manaseer, and

Dalhoum [12] proposed a new model of evolutionary wrapper feature selection by applying GA to explore the space of feature combinations from a set of features that has already assigned its priority. Extreme Learning Machines (ELM) and Support Vector Machine (SVM) are considered as the classifiers based on a Chronic Kidney Disease dataset (CKD) from the UCI repository [13]. The application of the proposed model affects the classification performance by improving an accuracy rate with less computing time.

Other recent works in the feature selection domain are focusing on the application of feature selection techniques to other areas of work such as face recognition, text classification and medical science [14]–[16]. The improvement of sequential feature selection tends to focus on a non-deterministic algorithm like particle swarm optimization, genetic algorithm or deep neural network [17], [18], whereas our study concerns a deterministic algorithm.

III. Proposed Method

In this study, we focus on the wrapper approach based on a sequential selection algorithm. It uses the result of a data mining algorithm to determine how good the given subset is. During the search process, the space of possible feature subsets is defined to generate and evaluate features until we get the satisfied subset. For the sequential floating search methods, the number of features dynamically increases and decreases until the desired target is reached. The variables allow floating forward or backwards so that they can be flexibly changed without pre-setting any parameters. For this reason, a floating search is possible to be trapped at a local optimum since the best *k*-subset does not necessarily contain the best (*k*–1)-subset. Therefore, we present an alternative improvement to the floating search algorithm to remove some of its drawbacks and try to find a solution as much closer to the optimal solution as possible.

Our study attempts to explore a new sequential feature selection algorithm that produces better results than the earlier works. We have proposed a Multi-levels Forward Inclusion (MLFI) algorithm where the idea is to remove the backtracking step and modify the "replacing the weak feature" step in IFFS with the addition of an adaptive floating search. Instead of adding one feature, we add more than one feature but to a point satisfied by the generalization limited (denoted by *r*). The appropriate value of *r* depends on the specified conditions which are described in the next subsection. In every iteration, we can add features ranging from 1 to r_{max}, where r_{max} is the maximum value of *r* that we need to define. The higher *r* leads to more computing time so we need to keep *r* as a small number such as 4 or 5. The level of *r* for each iteration will use *s* as a variable. The level *s* is similar to the level *o* in ASFFS but *s* is determined dynamically according to the *r* calculation methods. MLFI considers a wider range of features that lead to a more thorough search. As a result, there are higher chances to maximize the current feature subset regarding classification accuracy. MLFI algorithm can be explained below by the flowchart in Figure 2 and algorithm 1.

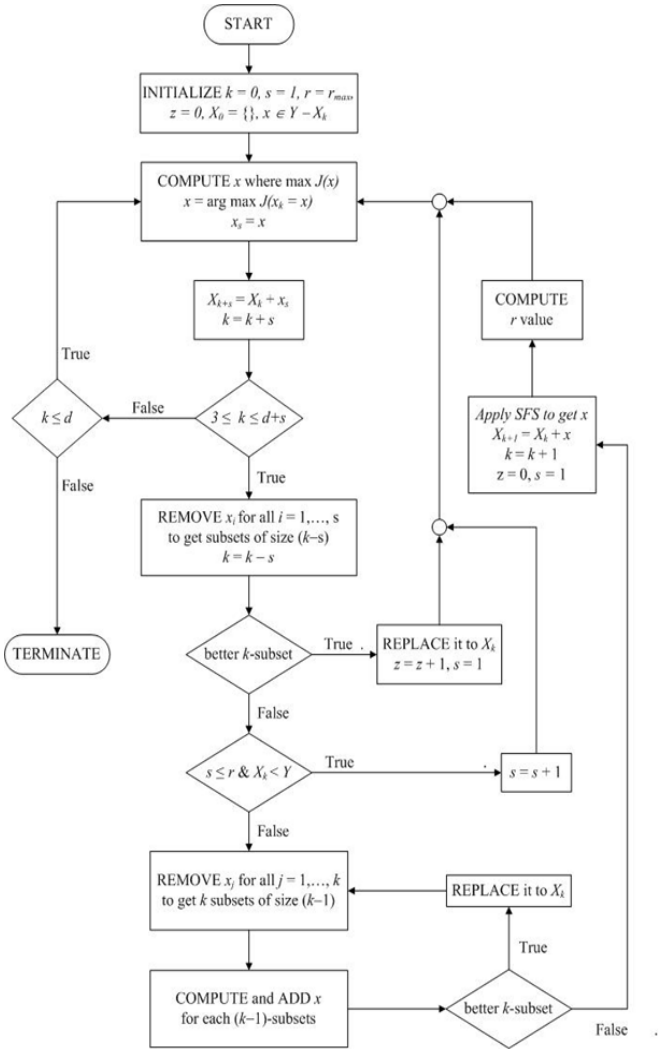

Figure 2. Flowchart of MLFI algorithm.

Step 1: Apply SFS to select one feature from the remaining feature set. Add this feature to the selected feature subset. Continue step 2 with the feature subset X_k where $k = k + 1$.

Step 2: From the selected feature subset (X_k), remove *s* features iteratively from 1 to *r*. Then, search for the best (*k*–*s*)-subset. If there is a better subset X_{k-s}, replace it to the previous X_{k-s}. Repeat steps 1 and 2 until $s > r$, then continue step 3.

Step 3: From the selected feature subset size *k*, remove 1 feature iteratively we have X_{k-1}, and use SFS to select a new feature from the remaining feature set ($Y - X_{k-1}$) for adding to each feature subset. Then calculate whether there is an improvement. If there is an improvement, replace that previous feature subset with the newly selected feature subset and repeat step 3. Otherwise, continue step 4 with the feature subset X_k.

Step 4: Compute the *r*-value, then continue step 5.

Step 5: Continue steps 1, 2, 3 and 4 until *d* features are selected.

Algorithm 1: Multi-levels Forward Inclusion (MLFI)

Input: A set of features $Y = \{y_1, y_2, \ldots, y_D\}$, where D is the number of input dimensions; J is a criterion function; d is the required subset size; r is the generalization level which is limited by r_{max}.
Output: A feature subset $X_k = \{x_j \mid j = 1, 2, \ldots, k; x_j \in Y\}$, where $k = (0, 1, 2, \ldots, d)$.
Initialize: Initialize $X_0 = \{\}$; $k = 0$; $s = 1$; $r = r_{max}$; $z = 0$.
(1) Feature Inclusion
#Find the best feature and update X_k
$x^+ = \arg\max J(x_k = x)$, where $x \in Y - X_k$
$X_{k+1} = X_k + x^+$
$k = k + 1$
$\max(X_k) = X_{k+1}$
(2) Multi-levels Forward Inclusion
#Searching for better k-subsets by multi-levels forward searching step
Repeat
x_s in X_k : #where $s = 1, \ldots, r$
$X_{k-s} = X_k - x_s$
If $J(X_{k-s}) > J(\max(X_{k-s}))$:
$\max(X_{k-s}) = X_{k-s}$
$z = z + 1$
Else
$s = s + 1$
Go to step 1
Until $s > r$
(3) Feature Replacement
#Replace a weak feature by trying to remove one feature and added one feature that maximize the criterion function
Repeat
For x_j in X_k : #where $j = 1, 2, \ldots, k$
$X_{k-1} = X_k - x_i$
For x_i in $Y - X_{k-1}$: #where $i = 1, 2, \ldots, d - (k-1)$
$x_i = \arg\max J(x_i)$
If $J(X_{k-1} + x_i) > J(X_k)$:
$X_k = X_{k-1} + x_i$
$\max(X_k) = X_k$
Until $J(X_{k-1} + x_i) \le J(X_k)$
$x^+ = \arg\max J(x_k = x)$, where $x \in Y - X_k$
$X_{k+1} = X_k + x^+$
$k = k + 1$
$\max(X_k) = X_{k+1}$
(4) Compute r-value
If $z < r_{max}$:
$r = r_{max} - z$
Else :
$r = 1$
$z = 0$
$s = 1$
(5) Termination Condition
#Terminate when $k > d$
If $k \le d$
Go to step 1
$X_k = \max(X_k)$ #for all k
Return the best individual subset X_k

A. Computation of r–Value

The generalization limit (r) needs to be carefully specified since a larger value of r results into a more thorough search and also increases the time complexity. We introduce a user-defined parametric limit r_{max} to restricting the maximum generalization level. This number can be any integer depending on how deep we need to search but normally it is only a small integer. The suggestion of r_{max} from ASFFS is 4. In our experiments, we assigned the value of r_{max} to be 5 for all tested datasets. The level s is determined dynamically according to the r calculation technique we have proposed.

1) Method I

The generalization limit can be changing adaptively depending on the number of times we have found better k-subsets. If we have found a few better subsets in the previous iteration, the next iteration we should try a deeper search and that will increase the value of r. On the other hand, if the previous iteration has found many better subsets, the next iteration may not need to go too deep that will decrease the value of r. This adaptive nature by adjusting the generalization limit automatically is aimed to save computing time. Therefore, the search should go deeper when cannot find a better subset. The application of this calculation technique leads to better performance than the previous works via our algorithm. We have selected the first 20 features from the whole dataset for the experiments. MLFI considers a wider range of features that lead to a thorough search. As a result, there are higher chances to improve the current feature subset.

Assume $r_{max} = 5$, thus $1 \le r \le 5$. Let z be the number of times the algorithm has found a better subset for that particular iteration. Thus, z is related to r by $r_{max} - z$. Adaptive determination of r is defined as follows.

1. If $z < r_{max}$, $r = r_{max} - z$
2. Else , $r = 1$

From the condition above, we can build a graph in Figure 3 that shows the value of r the first 20 features; if we have $z = \{0, 0, 2, 0, 3, 0, 2, 0, 0, 0, 3, 0, 2, 1, 0, 0, 3, 0, 4, 0\}$. The value for r decreases while z increases.

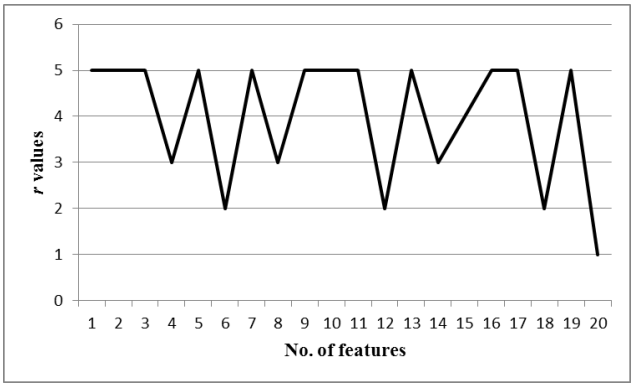

Figure 3. r-values for method I.

2) Method II

The generalization limit r increases step by step starting from 1 to r_{max} along with the feature subset sizes. The calculation can be defined by the equation below.

1. $r = \lceil k / \lceil d/r_{max} \rceil \rceil$

For example, if we let $d = 20$ and $r_{max} = 5$, thus $r = \lceil k/4 \rceil$. Now, the value for r is shown in Figure 4 for all subset size k, where $1 \le k \le 20$.

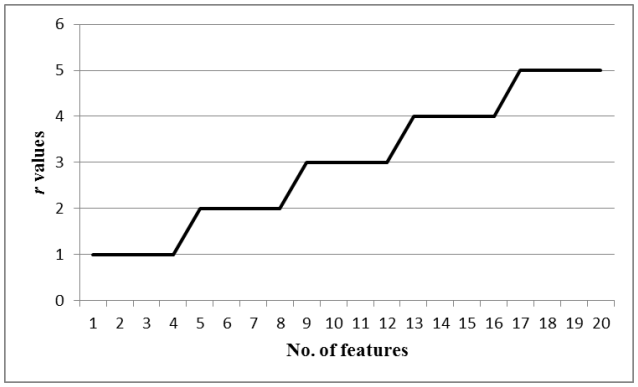

Figure 4. *r*-values for method II.

3) Method III

The idea of the third method is to avoid the solution to be trapped by local optima since it starts with the same subset as IFFS for the first half of the *k*-subsets ($r = 1$). For the second half, we increase the *r*-value until it reaches the maximum generalization limit. Then the process continues by applying r_{max} through the end of the required subset sizes. The calculation is defined below.

1. If $k <= \lfloor d / 2 \rfloor$, $r = 1$
2. Else if $\lfloor d / 2 \rfloor < k < \lfloor d / 2 \rfloor + r_{max}$, $r = r + 1$
3. Else , $r = r_{max}$

For example, let assume $d = 20$ and $r_{max} = 5$, then we have; if $k <= 10$: $r = 1$, elif $10 < k < 15$: $r = r + 1$, else: $r = 5$. Figure 5 shows an example of *r*-value for method III.

Figure 5. *r*-values for method III.

B. Demonstration by an Example using Wine Dataset

To demonstrate the MLFI algorithm we have selected the wine dataset from UCI repository for our illustration. Initially, assume we have a dataset $Y = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ with 13 features, the required subset size (d) is 20 features, and assign $r_{max} = 5$.

1) Feature Inclusion

At the beginning, assume that we apply SFS for the first 3 features, thus for $k = 1$, 2 and 3 we have $X_1 = \{6\}$, $X_2 = \{6, 10\}$ and $X_3 = \{6, 10, 2\}$ respectively. Now, the current subset of $k = 3$ is $X_3 = \{6, 10, 2\}$. This subset is the best 3-subset that has been found so far.

2) Multi-levels Forward Inclusion

Assume we continue the process up to $k = 4$, we have $X_4 = \{6, 10, 2, 7\}$ with 90.09% classification accuracy. For $k = 4$, $s = 1$, remove one feature except $x_4 = 7$ to find a better 3-subset, now we have ($\{10, 6, 7\}$, $\{2, 6, 7\}$ and $\{2, 10, 7\}$). After calculate the criterion function values J, we have found that $J(\{10, 6, 7\})$, $J(\{2, 6, 7\})$ and $J(\{2, 10, 7\})$ are not greater than $J(\{6, 10, 2\})$, therefore we continue to the next inner loop for $s = 2$ with $J(\{6, 10, 2\}) = 90.04\%$ as the best 3-subset that has been found so far.

For $k = 5$, $s = 2$, we add the forth and the fifth features into X_5 we get $\{6, 10, 2, 7, 5\}$. Remove two features to find a better 3-subset, now consider only subsets containing the feature $x_5 = \{5\}$ which are ($\{5, 6, 7\}$, $\{2, 10, 5\}$, $\{10, 5, 6\}$, $\{10, 5, 7\}$, $\{2, 5, 6\}$, $\{2, 5, 7\}$). We calculate the J values for all the combinations of 3-subset by removing any two features except x_5. We cannot find a better 3-subset, the process continues to the next inner loop for $s = 3$ with the same $J(\{6, 10, 2\})$ as the best 3-subset that has been found so far.

For $k = 6$, $s = 3$, we add the forth, the fifth and the sixth features into X_6 we get $\{6, 10, 2, 7, 5, 11\}$. Remove three features to find a better 3-subset, now consider only subsets containing the feature $x_6 = \{11\}$ which are ($\{11, 6, 7\}$, $\{10, 11, 5\}$, $\{10, 11, 6\}$, $\{10, 11, 7\}$, $\{11, 5, 7\}$, $\{2, 11, 5\}$, $\{11, 5, 6\}$, $\{2, 11, 6\}$, $\{11, 10, 2\}$, $\{2, 11, 7\}$). We calculate the J values for all the combinations of 3-subset by removing any three features except x_6. From the current feature subsets, we cannot find a better 3-subset, continue to the next inner loop for $s = 4$ with the same $J(\{6, 10, 2\})$ as the best 3-subset that has been found so far.

For $k = 7$, $s = 4$, we add the forth, the fifth, the sixth, and the seventh features into X_7 we get $\{6, 10, 2, 7, 5, 11, 0\}$. Remove four features to find a better 3-subset, now consider only subsets containing the feature $x_7 = \{0\}$ which are ($\{0, 5, 6\}$, $\{0, 10, 7\}$, $\{0, 5, 7\}$, $\{0, 10, 6\}$, $\{0, 11, 6\}$, $\{0, 11, 7\}$, $\{0, 10, 11\}$, $\{0, 6, 7\}$, $\{0, 11, 5\}$, $\{0, 2, 5\}$, $\{0, 2, 7\}$, $\{0, 2, 6\}$, $\{0, 2, 10\}$, $\{0, 2, 11\}$, $\{0, 10, 2\}$, $\{0, 10, 5\}$). We calculate the J values for all the combinations of 3-subset by removing any four features except x_7. From the current feature subsets, we can find a better 3-subset, which is $J(\{0, 11, 6\}) = 90.06\%$. Therefore, continue the next iteration for $k = 4$ and $s = 1$ with $J(\{0, 11, 6\})$ as the best 3-subset that has been found so far.

The process repeats by exploring further until an improvement cannot be made. After the adaptive search meets the condition $s > r$, the best 3-subset that we have found so far would be $X_3 = \{0, 11, 6\}$. Continue the next step in order to maximize this solution.

3) Feature Replacement

We are now continue onto the feature replacement step by removing one feature iteratively we get $\{11, 6\}$, $\{0, 6\}$ and $\{0, 11\}$, then select one feature from the remaining set using SFS we get $\{11, 6, 0\}$, $\{0, 6, 9\}$ and $\{0, 11, 6\}$. The J values for each subset are 90.06%, 91.17% and 90.06% respectively. At this point, we have found $J(\{0, 6, 9\}) = 91.17\%$ as the new best 3-subset. Replace $\{0, 11, 6\}$ with $\{0, 6, 9\}$ thus $X_3 = \{0, 6, 9\}$ is the best subset of size 3 features that we have found so far. Continue the next outer loop with $k = 4$ for $J(\{0, 6, 9, 11\}) = 92.27\%$ as the best 4-subset. We can see that the number of

features in the subset either increases or remains the same throughout the whole process. When the process continues up to 5-subset we get X_5 = {0, 6, 8, 9, 11} which is the highest accuracy which cannot be found by other sequential searching techniques.

4) Compute r-value

An adaptive determination of *r* is applied to find the value of *r* for the next iteration. In method I, there are two input variables, one is the maximum value of *r* (r_{max}), which is 5 for this particular example. The other one is *z*, which recently acquired from the multi-levels forward inclusion step. The value of *r* can be varying from 1 to 5 depending on the value of *z*. Therefore, *r* is changing adaptively on different iteration. Method II and III do not require a variable *z*, whereas they can lead to better accuracy than the method I for some datasets.

5) Termination Condition

The MLFI algorithm processes sequentially until the subset size (*k*) reaches the required subset size (*d*). The best of all feature subsets are copied into X_k and then terminate the program. This method applies the idea of adaptive search in order to explore the potential subset thoroughly, in other words, it provides a better chance to find the optimal solution via a more detailed search by adjusting the generalization limit adaptively.

IV. Experimental Setup

To compare our proposed method with other algorithms, we have developed the experimental environment similar to the previous works. The performance of the feature selection methods is usually evaluated by the machine learning model. The common models include Naïve Bayes, C4.5, SVM, Decision Tree, K-means, etc. One of the simplest and most popular classifiers is K-Nearest-Neighbors (KNN). Due to its robustness and versatility, KNN is often used in various applications such as economic forecasting, data compression and genetics which can outperform other more powerful classifiers. KNN is a supervised learning algorithm and is selected to calculate the classification accuracy in IFFS. Therefore, we use KNN to compare our performance on different algorithms based on 5-fold cross-validation. Table 1 shows eight standard datasets used in this study with various sizes from the UCI machine learning repository.

Table 1. Datasets used in the experiments.

Dataset name	No. of instances	No. of features	No. of classes
Wine	173	13	3
Online shopper	12330	17	2
Lymphography	142	18	2
Crowdsourced	10545	28	6
Ionosphere	351	34	2
Soybean	266	35	15
Spectf heart	267	44	2
Sonar	208	60	2

Python is one of the most popular languages with a rise in technologies like machine learning, artificial intelligence and data analysis. It is the selected programming language for our study which was developed using Jupyter notebook editor. Data normalization is preferred as a preprocessing step. We randomly select some instances in case the number is large and also eliminate some missing values if necessary. We have applied the same randomly selected instances to all techniques to ensure that they receive the same input.

V. Results and Discussion

In this section, we discuss our results on the MLFI algorithm compared with popular suboptimal methods which are SFS, SFFS and IFFS. Our study aims to increase classification accuracy rather than reducing the time complexity. Normally, the improvement of the earlier technique requires a more complicated algorithm which also requires higher computing time unavoidably. MLFI requires higher time than IFFS whereas they are bounded by $O(n^2)$, moreover, SFS and SFFS are bounded by $O(n)$. The size of the dataset does not affect the algorithm. The number of features shown in each graph would be either 20 or less depending on the size of the dataset. The *y*-axis is the classification accuracy and the *x*-axis is the number of features. We consider the smallest subset with the maximum accuracy as the best performance for each algorithm. Results of MLFI are selected from the *r*-value calculation methods that produce the best performance among the three methods.

Table 2. The comparison of maximum classification accuracy (%) and resulted number of selected features in parenthesis for each algorithm.

Dataset name	SFS	SFFS	IFFS	MLFI
Wine	92.82 (10)	93.38 (7)	93.38 (7)	93.38 (7)
Online shopper	90.43 (7)	90.59 (7)	90.67 (5)	90.67 (4)
Lymphography	88.00 (15)	88.76 (13)	90.14 (11)	90.81 (10)
Crowdsourced	89.46 (20)	88.98 (20)	90.13 (19)	90.42 (20)
Ionosphere	93.45 (5)	94.02 (12)	94.59 (12)	94.87 (8)
Soybean	89.10 (18)	90.23 (18)	90.23 (18)	91.73 (20)
Spectf heart	81.65 (11)	84.64 (8)	85.36 (12)	86.53 (15)
Sonar	78.56 (11)	77.44 (6)	80.88 (20)	81.76 (19)

The results in Table 2 show that the classification accuracy was noticeably enhanced by the proposed algorithm compared to the previous works. MLFI shows the best performance in all datasets since it produces either the highest in accuracy and/or lower number of feature subset. Only for the Wine dataset that SFFS, IFFS and MLFI achieve the same optimal solutions. This is due to the size of the dataset being small. In the following subsection, we show the accuracy graphs for our further discussion.

A. Wine dataset

Figure 6 shows that IFFS performs equal to or better than SFFS for all sizes except 3-subset and 5-subset. Whereas

MLFI produces better results than IFFS in general and has higher accuracy than SFFS at subsets of size 5 and 6 with 93.36 %. This is the best 5-subset and 6-subset of Wine dataset which cannot be found by the other methods. Overall MLFI performs either equal to or better than SFS, SFFS and IFFS. Thus, the MLFI algorithm produces the best results among the other techniques on this dataset.

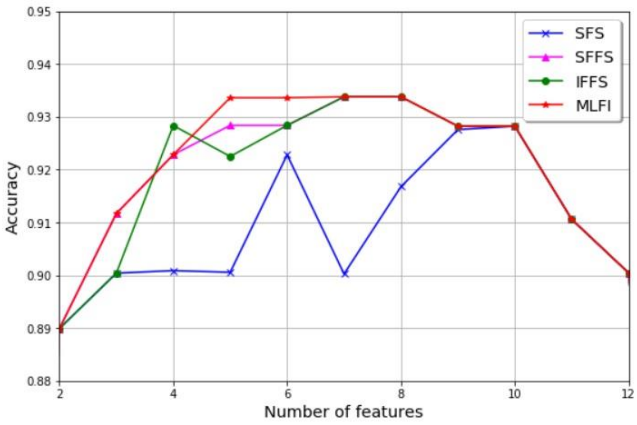

Figure 6. Accuracy graph for Wine dataset.

B. *Online shopper dataset*

In Figure 7, SFFS, IFFS and MLFI produce similar results but only IFFS and MLFI that give the maximum accuracy with 90.67%. This solution occurs at 5-subset for IFFS, whereas MLFI occurs at 4-subset. Therefore, the MLFI algorithm produces the best results regarding this dataset.

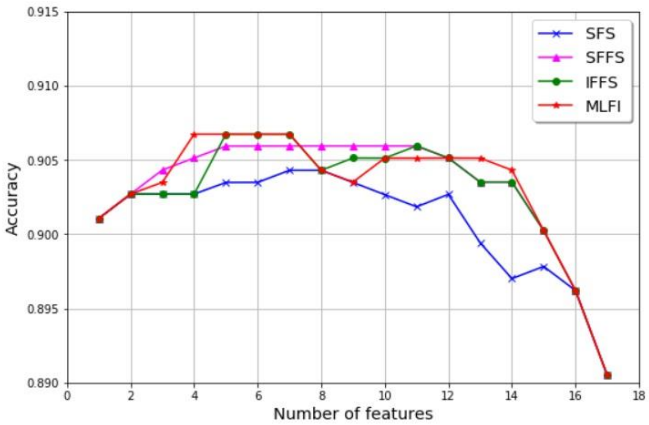

Figure 7. Accuracy graph for Online shopper dataset.

C. *Lymphography dataset*

There are 18 features for lymphography dataset. Even though the number of features is small but different combinations can lead to different performances. MLFI begins to produce the best performance from 9-subset up to 11-subset (see Figure 8). From 11-subset, MLFI gives the same results as IFFS. Overall MLFI produces the best maximum accuracy than SFS, SFFS and IFFS. Therefore, MLFI algorithm produces the best results among other techniques.

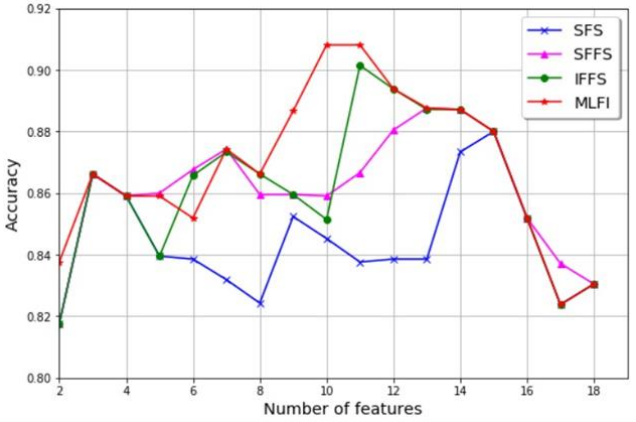

Figure 8. Accuracy graph for Lymphography dataset.

D. *Crowdsourced dataset*

All techniques produce increasing accuracy when the feature size grows bigger (see Figure 9). MLFI and IFFS produce the same results at the beginning until it reaches a subset size 11 then start to diverge. MLFI performs either equal to or better than IFFS for the rest of the feature size accept only 13-subset. SFS and SFFS have lower accuracy than MLFI for almost all the subset sizes, thus we can ignore these two methods. Overall 20 features, MLFI gives the best solution at 20-subset. The maximum solution for IFFS occurs at 19-subset which has a lower accuracy than MLFI with the same size.

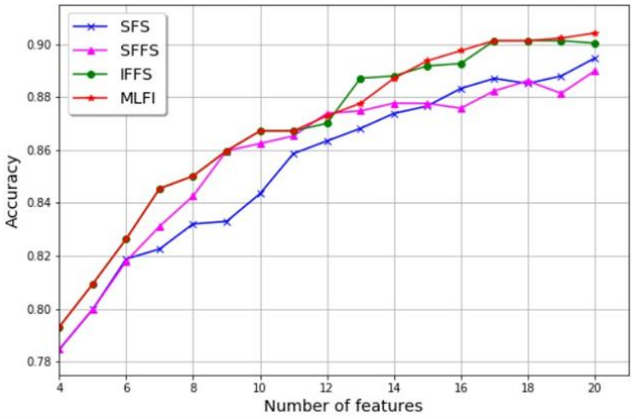

Figure 9. Accuracy graph for Crowdsourced dataset.

E. *Ionosphere dataset*

From Figure 10, MLFI produces the highest accuracy from 6-subset to 11-subset. Its performance drops a little bit at 18-subset but still close to IFFS. MLFI performs well in general and produces the best result at 8-subset with 94.87% accuracy which cannot be found by the other techniques. It is quite obvious that MLFI is the best method regarding this dataset.

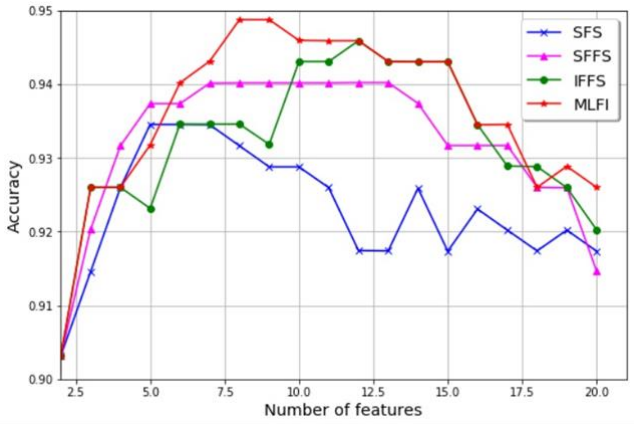

Figure 10. Accuracy graph for Ionosphere dataset.

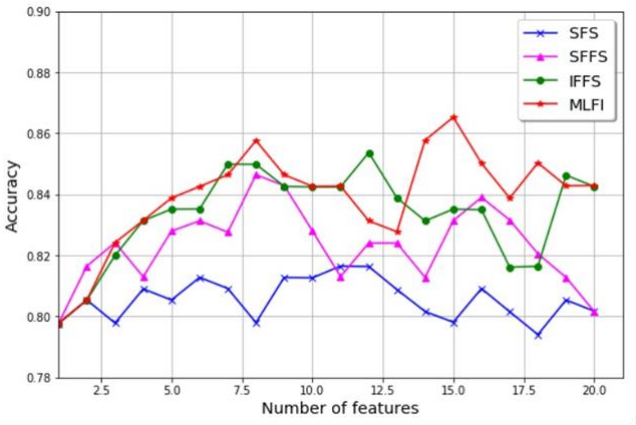

Figure 12. Accuracy graph for Spectf heart dataset.

F. Soybean dataset

From the Soybean dataset, MLFI produces exactly the same accuracy with IFFS from the beginning to 13-subset and diverges higher (see Figure 11). Its performance surpasses the other methods at 18-subset onward. IFFS performs quite poor for this dataset. From 12-subset up to 20-subset, its accuracy is even lower than SFFS accept at 19-subset. For overall 20 feature subsets, MLFI can give the best accuracy at 20-subset with 91.73% and better than the other for all subsets since 18-subset.

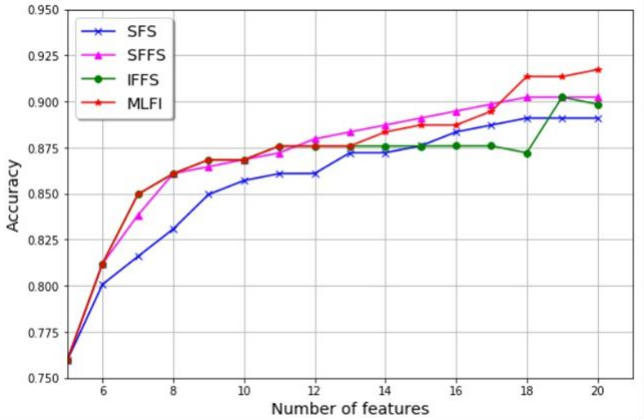

Figure 11. Accuracy graph for Soybean dataset.

G. Spectf heart dataset

For spectf heart dataset with 44 features and 267 instances, both MLFI and IFFS generally produce better results all the way through. MLFI gives lower accuracy than IFFS at 7, 12, 13, and 19 feature subsets, and also lower than SFFS only at the beginning (see Figure 12). For other subset sizes, MLFI performs better most of the times. The best solution of 86.53% accuracy for the first 20-subset is produced by MLFI at 15-subset.

H. Sonar dataset

Sonar dataset contains 60 features with 208 instances. The graph in Figure 13 shows that all techniques produce varies results from the beginning to the end. MLFI gives the same values with IFFS at the start. Both MLFI and IFFS overcome the other two techniques since the subset size 9. But for all 20 feature subsets, MLFI performs best at 19-subset with the maximum accuracy at 81.76%, whereas the best solution from IFFS is 20-subset with 80.88% accuracy.

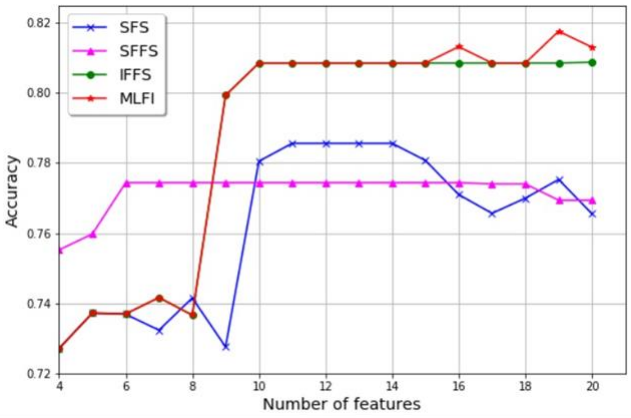

Figure 13. Accuracy graph for Sonar dataset.

The proposed algorithm based on a sequential feature selection algorithm produced effective feature subsets with higher classification accuracy on several different datasets. Our proposed algorithm can extract a more relevant and effective feature subset from the source dataset using multi-levels forward search with an adaptive generalization level technique. From the experiments, the maximum accuracies with the smallest subsets produced by our proposed method most of the times. This improvement is the result of the multi-levels forward-searching technique that leads to a more thorough search on the current feature subsets. Many powerful subsets with higher accuracy are discovered by our in-depth searching method.

VI. Conclusion

Feature selection is very important for the performance of classification in the data mining process. Our study focuses on an improvement of the early sequential feature selections. Our proposed algorithm is Multi-levels Forward Inclusion (MLFI) algorithm. We aim to develop a feature selection method that overcomes the previous works in terms of accuracy. We proposed a feature selection algorithm based on the sequential searching technique by improving the performance of SFFS. Incorporating an adaptive multi-levels forward search that maximizes a classification accuracy of feature subset with the addition of feature replacement step that is designed to solve the nesting problem. It is able to discover important subsets that cannot be discovered by SFFS or IFFS. The algorithm employs an adaptive generalization limit to indicate the level of forwarding search. The higher the limit leads to a higher chance of finding a better subset. In the experiments, we have compared our method with SFS, SFFS and IFFS. Results on the eight standard UCI datasets have shown that MLFI performs better than the other suboptimal sequential feature selection algorithms for the majority of the tested datasets. Further study can be focusing on the adjustment of generalization limit and the application of MLFI with various criterion functions.

References

[1] K.Pavya, B.Srinivasan, "Feature Selection Techniques in Data Mining: A Study", *International Journal of Scientific Development and Research (IJSDR)*, vol. 2, no. 6, pp. 594-598, 2017.

[2] J. Cai, J. Luo, S. Wang, S. Yang, "Feature selection in machine learning: A new perspective", *Neurocomputing*, pp. 70-79, 2018.

[3] K.Sutha, J. J. Tamilselvi, "A Review of Feature Selection Algorithms for Data Mining Techniques", *International Journal on Computer Science and Engineering (IJCSE)*, pp. 63-67, 2015.

[4] P. Somol, J. Novovicova, P. Pudil, "Flexible-Hybrid Sequential Floating Search in Statistical Feature Selection", *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 632-639, 2006.

[5] M. W. Mwadulo, "A Review on Feature Selection Methods For Classification Tasks", *International Journal of Computer Applications Technology and Research*, vol. 5, no. 6, pp. 395- 402, 2016.

[6] A. Jovic, K. Brkic, N. Bogunovic, "A review of feature selection methods with applications", *International Convention on Information and Communication Technology*, 2015.

[7] P. Pudil, J. Novovicova, J. Kittler, "Floating search methods in feature selection", *Pattern Recognition Letters*, pp. 1119-1125, 1994.

[8] P. Somol, P. Pudil, J. Novovicova, P. Paclik, "Adaptive floating search methods in feature selection", *Pattern Recognition Letters*, pp. 1157-1163, 1999.

[9] S. Nakariyakul, and D. P. Casasent, "An improvement on floating search algorithms for feature subset selection", *Pattern Recognition*, pp. 1932-1940, 2009.

[10] J. Lv, Q. Peng, Z. Sun, "A modified sequential deep floating search algorithm for feature selection", *International Conference on Information and Automation*, pp. 2988-2933, 2015.

[11] K. Homsapaya, O. Sornil, "Improving Floating Search Feature Selection using Genetic Algorithm", *Journal of ICT Research and Applications*, vol. 11, no. 3, pp. 299-317, 2017.

[12] M. Kadhum, S. Manaseer, A. L. A. Dalhoum, "Evaluation Feature Selection Technique on Classification by Using Evolutionary ELM Wrapper Method with Features Priorities", *Journal of Advances in Information Technology*, vol. 12, no. 1, pp.21-28, Feb. 2021.

[13] D. Dheeru, E. Karra Taniskidou, "UCI machine learning repository", http://archive.ics.uci.edu/ml, 2017.

[14] V. B. Canedo, A. A. Betanzos, "Ensembles for feature selection: A review and future trends", *Information Fusion*, vol. 52, pp. 1-12, 2019.

[15] G. Cisotto, M. Capuzzo, A. V. Guglielmi, A. Zanella, "Feature selection for gesture recognition in Internet-of-Things for healthcare", *IEEE International conference on Communication (ICC)*, Jun. 2020.

[16] R. J. S. Raj, S. J. Shobana, I. V. Pustokhina, D. A. Pustokhin, D. Bupta, K. Shankar, "Optimal Feature Selection-Based Medical Image Classification Using Deep Learning Model in Internet of Medical Things", *IEEE Access*, vol. 8, pp.58006-58017, Mar. 2020.

[17] W. Liu, J. Wang, "A Brief Survey on Nature-Inspired Metaheuristics for Feature Selection in Classification in this Decade", *Proceedings of the 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*, pp. 424-429, May 2019.

[18] R. K. Huda, H. Banka, "New efficient initialization and updating mechanisms in PSO for feature selection and classification", *Neural Computing and Applications*, vol. 32, pp. 3283-3294, 2020.

Author Biographies

K. Chotchantarakun is a Ph.D. Candidate in Computer Science from the Graduate School of Applied Statistics, National Institute of Development Administration in Thailand. He is also a lecturer from the Faculty of Information Studies, Burapha University in Thailand. He received his M.Sc. in Computer Science from Chulalongkorn University, Thailand in 2006, and B.Sc. in Computer Science 2nd class honors from Mahidol University International College, Thailand in 2003. His research of interest includes Computer Arithmetic, Data Mining, Data Science and Information Science.

O. Sornil is an associate professor from the Graduate School of Applied Statistics, National Institute of Development Administration in Thailand. He received his Ph.D. in Computer Science from the Virginia Polytechnic Institute and State University, USA in 2001, M.Sc. in Computer Science from Syracuse University, USA in 1997, M.B.A. in Finance from Mahidol University, Thailand in 2006, and B.Sc in Electrical Engineer 2nd class honors from Kasetsart University, Thailand in 1993. His research of interest includes Artificial Neural Network, Genetic Algorithm, Fuzzy Logic, Text Mining, Data Communication and Networking, Data Science, Business Analytics, Parallel Processing, and Pattern Recognition.