Journal of Artificial Intelligence Research 75 (2022) 953-983 Submitted 06/2022; published 11/2022

Initialization of Feature Selection Search for Classification

Maria Luque-Rodriguez MLUQUE@UCO.ES
Jose Molina-Baena I42MOBAJ@UCO.ES
Alfonso Jimenez-Vilchez I52JIVIA@UCO.ES
Dept. of Computer Science and Numerical Analysis, Universidad de Córdoba (Spain)

Antonio Arauzo-Azofra ARAUZO@UCO.ES
Area of Project Engineering, Univerdad de Córdoba (Spain),
Campus de Rabanales, Cordoba 14014 Spain

Abstract

Selecting the best features in a data set improves accuracy and efficiency of classifiers in a learning process. Data sets generally have more features than necessary, some of them being irrelevant or redundant to others. For this reason, numerous feature selection methods have been developed, in which different evaluation functions and measures are applied. This paper proposes the systematic application of individual feature evaluation methods to initialize search-based feature subset selection methods. An exhaustive review of the starting methods used by genetic algorithms from 2014 to 2020 has been carried out. Subsequently, an in-depth empirical study has been carried out evaluating the proposal for different search-based feature selection methods (Sequential forward and backward selection, Las Vegas filter and wrapper, Simulated Annealing and Genetic Algorithms). Since the computation time is reduced and the classification accuracy with the selected features is improved, the initialization of feature selection proposed in this work is proved to be worth considering while designing any feature selection algorithms.

1. Introduction

Inside the field of Pattern Recognition, the task of a classifier is to use a feature vector to assign an object to a category (Duda, Hart, & Stork, 2000). A supervised classification learning algorithm generates classifiers from a table of training vectors whose category is known. However, sometimes these vectors have more features than those really needed. Feature selection (or attribute reduction) is a technique used to choose a subset of the available features that allows us to obtain acceptable results, sometimes even better results. This speeds up the learning process by using less features.

Theoretically, if we knew the complete statistical distribution, the more features used the better results would be obtained. However, in practical learning scenarios, it might be better to use a feature subset (Kohavi & John, 1997). The process of feature selection in any classification problem is crucial since it allows us to eliminate: those features that may mislead us (the so-called noise features), those features that do not provide much information (irrelevant features) and those that include repeated information (redundant features)(Tang, Alelyani, & Liu, 2014).

Sometimes, if we have a large number of initial features to analyze, the algorithms that carry out the learning process may have memory or time consumption problems. They can even turn to be inapplicable. Besides, the use of feature selection functions may improve

intelligibility and reduce the data acquisition and handling costs. Due to all these advantages, feature selection has become a widely used technique in classification. As a result of this, several methods have been developed (Jović, Brkić, & Bogunović, 2015; Thangavel & Pethalakshmi, 2009), with diverse recent applications, for example, in intrusion detection system (Aljawarneh, Aldwairi, & Yassein, 2018), in urban short-term travel speed prediction (Chen, Wang, Zhang, Wei, Xu, Huang, & Cai, 2018) or in text classification problems (Labani, Moradi, Ahmadizar, & Jalili, 2018).

The problem of feature selection may be seen as a search problem in the power set of available features set (Blum & Langley, 1997)(Kohavi, 1994). The aim is to find a feature subset that allows us to improve a learning process in any way. In order to characterize all feature selection methods, we can distinguish their building blocks using the modularization shown in figure 1 (Arauzo-Azofra, Benitez, & Castro, 2008). This modularization allows to create new methods by combination and to focus research on a specific module. This paper focus on the initialization part of the search method.

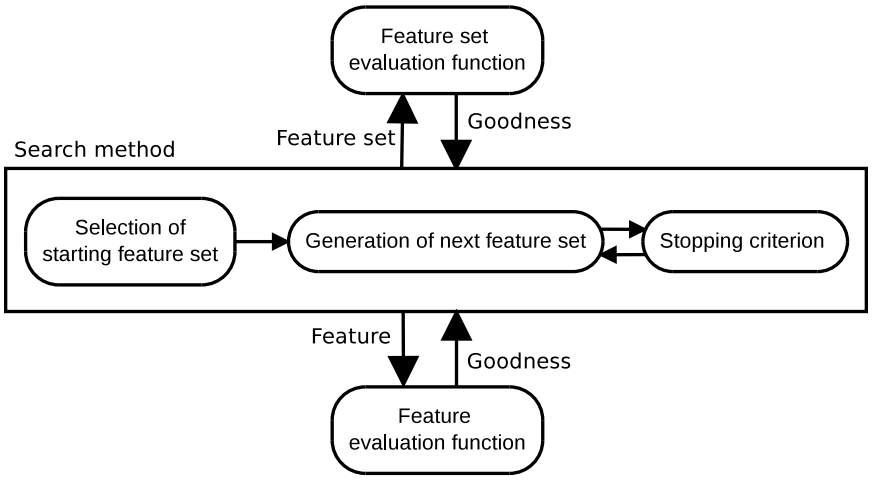

Figure 1: Feature selection modularized.

Feature selection algorithms can be divided in two main types. The first one is formed by those methods that are based on a search strategy over the search space of all possible feature sets together with a feature set evaluation measure, which are commonly named feature subset selection methods (to emphasize that they are working with sets). The other type is formed by the methods that evaluate all features individually and then apply some cutting criteria to decide which features are selected and which are not. On the one hand, feature subset selection methods are superior to those based on individual evaluation because, as they can consider inter-dependencies among features, they achieve better results. On the other hand, individual feature selection methods are much faster and easier to configure (Schiffner, Bischl, Lang, Richter, Jones, Probst, Pfisterer, Gallo, Kirchhoff, Kühn, et al., 2016). These are probably the reasons why they are so widely used.

Feature subset selection methods are slower because the search space is large (2^n, being n the number of features). For this reason, any improvement on the search can be profitable. Focusing on the *selection of starting feature set* block, this paper explores the hybridization of both types of feature selection methods, by using individual feature selection methods

as a starting point for the search strategy of feature subset selection methods. Similar hybridizations have been proposed in several papers (Hsu, Hsieh, & Lu, 2011; Oreski & Oreski, 2014; Jadhav, He, & Jenkins, 2018; Singh & Selvakumar, 2015) without the modular view. However, by using this modular view, this paper presents many novel hybridizations by combination and it empirically evaluates them to obtain more general conclusions.

With the hypothesis that these combined methods can perform faster —as they may avoid exploring some parts of the space— and provide better features —by being focused on a more concrete area of the space—, we compare traditional methods and the ones implementing the starting strategy. This study helps us to assess how suitable individual evaluation methods are to start feature subset selection methods.

This paper is organized as follows. In section 2, some feature selection methods are described by their building blocks. Section 3 reviews initialization strategies recently used in genetic algorithms. Section 4 describes in detail the proposed starting methods. Section 5 describes the empirical methodology proposed. Finally, Sections 6 and 7 describe, respectively, the results and the conclusions obtained.

2. Feature Selection Methods

From the two approaches to feature selection that we propose to hybridize, first, we revise those methods that evaluate the features individually. Next, we study the methods that use feature set evaluation.

2.1 Individual Feature Selection Methods

Individual feature selection methods are composed of two parts that are run sequentially. The first part is a measure that assigns an assessment of its relevance to each feature. The second part is analogous to a direct search. It decides which set of features is selected by just using those individual feature evaluations. Usually this is performed with a cutting criterion as a limit for the number of features included. We describe here the commonly used proposals for each part that will be used later in the proposed hybridization.

The main advantage of these methods is that, by using simple measures, the assessment speed of the relevant features is rapid. On the contrary, as the main disadvantage, they cannot properly consider the redundancy between features, so there may be two relevant features selected while excluding another that is more relevant.

2.1.1 Measures of Individual Feature Utility

The following five individual measures are usually considered to score the features in these methods:

- **Mutual Information (info)** measures the quantity of information that one feature gives about the class, based on Shannon's information theory (Vergara & Estévez, 2014).

$$I(F, C) = H(C) - H(C|F) \tag{1}$$

- **Gain Ratio (gain)** is defined as the ratio between information gain and the entropy of the feature. In this way, this measure avoids favoring features with more values.

This measure is also used inside C4.5 algorithm (Quinlan, 1993).

$$\text{Gain ratio} = \frac{I(F,C)}{H(F)} \tag{2}$$

- **Gini index (gini)** can be seen as the probability of two randomly chosen instances having a different class. It was used by Breiman to generate classification trees (Breiman, Friedman, Olshen, & Stone, 2017).

$$\text{Gini index} = \sum_{i,j \in C; i \neq j} p(i|F)p(j|F) \tag{3}$$

- **Relief-F (reli)** is an extension of Relief (Kononenko, 1994). It can handle discrete and continuous attributes, as well as null values. Despite evaluating individual features, Relief takes into account the relation among features. This is probably the reason behind Relief-F great performance and why it has become very well known and commonly used in feature selection.
- **Relevance (rele)** is a measure that discriminates between attributes on the basis of their potential value in the formation of decision rules (Demšar, Curk, Erjavec, Črt Gorup, Hočevar, Milutinovič, Možina, Polajnar, Toplak, Starič, Štajdohar, Umek, Žagar, Žbontar, Žitnik, & Zupan, 2013).

2.1.2 Cutting Criteria

In this paper, the previous measures are combined with the following two typical cutting criteria:

- **Fixed number (n)** simply selects a given number of a features. Obviously, the selected features will be the ones with the greater evaluation.
- **Fraction (p)** selects a fraction, given as a percentage, of the total number of available features.

2.2 Feature Subset Selection Methods

Feature subset selection methods employ measures that evaluate complete sets of features. With the information provided by evaluating sets, we can handle the interrelations between features, thus avoiding the redundancy. This is impossible to do by using only individual feature measurements. These methods usually provide more accurate results. However, their main drawback is that assessing all possible subsets of features is usually not feasible or very expensive. For this reason, heuristic searches applied in these methods lead only to sub-optimal results.

2.2.1 Measures of Feature Set Utility

Feature set measures are functions that, given a training data set ($T \in \top$, where $\top$ is the set of every possible training set) and a feature subset $S \subset P(F)$ ($P(F)$ denotes the powerset of F, being F the set of all features), return a valuation of the relevance of those features.

$$\text{Evaluation function} : P(F) \times \top \rightarrow \mathbb{R} \tag{4}$$

Three well known feature set measures are:

- **Inconsistent examples** measure uses an inconsistency rate that is computed by grouping all examples (patterns) with the same values in all the selected features. For each group, assuming that the class with the largest number of examples is the correct class of each group, the number of examples with a different class is counted (these are the inconsistent examples) (Arauzo-Azofra et al., 2008). The rate is computed dividing the sum of these counts by the number of examples in the data set, as seen in the equation:

$$\text{Inconsistency} = \frac{\text{Number of inconsistent examples}}{\text{Number of examples}} \tag{5}$$

  In order to establish the relation between the consistency and inconsistency and since each one is defined in the interval [0,1], we define the consistency as:

$$\text{Consistency} = 1 - \text{Inconsistency} \tag{6}$$

- **Mutual information**, as the individual mutual information measure, identifies the difference between the class entropy and the class entropy conditioned to the knowledge provided by, in this case, the whole set of features to evaluate (Vergara & Estévez, 2014):

$$I(C, S) = H(C) - H(C|S) \tag{7}$$

  The ideal scenario would be finding the smallest set of features that fully determines C, this means I(C,S) = H(C), but it is not always possible.

- **Wrapper approach** measure uses the learning algorithm to evaluate whether a feature set is good (Kohavi & John, 1997). It uses a quality measure of the learning algorithm performance with the selected features. One of the advantages of this measure is that the feature selection algorithm performs the feature evaluation in the same context in which it will be applied and thus it takes into account the possible bias of the learning algorithm.

2.2.2 Search Methods

The search strategies for feature sets are diverse. In this paper, several different search strategies are selected to show the effect of initialization in diverse contexts: sequential, probabilistic and meta-heuristic searches.

The feature selection search space can be represented in a Hasse diagram. For example, a five feature (a, b, c, d, e) space is illustrated in figure 2. Each level represents all the feature sets of a given size and the lines connect sets that only differ in one feature (adding a feature downwards and removing upwards). The path followed by an example execution

of the search method is represented in colors. We use a light background colour to represent all evaluated sets. They are numerated in the evaluation order. The arrows and borders with the same darker colour represent the sets that are selected as the following state.

As representative of the sequential searches, the Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS) methods (Pudil, Novovičová, & Kittler, 1994) are selected (they are also known as Step Forward/Backward Selection). The former starts from an empty set of features and it adds the feature that improves the selection the most. The later conducts the search in the opposite direction, removing the feature that improves or reduces the least.

In figure 2 (red lines), the example run of SFS evaluates all the feature sets ({a}, {b}, {c}, {d} and {e}), choosing the set that obtains the best result ({c}). Then it evaluates all the feature sets including one more feature ({a, c}, {b, c}, {c, d} and {c, e}). This process is repeated until the established stopping criterion is reached, in the example, up to the final set {a, b, c, e}.

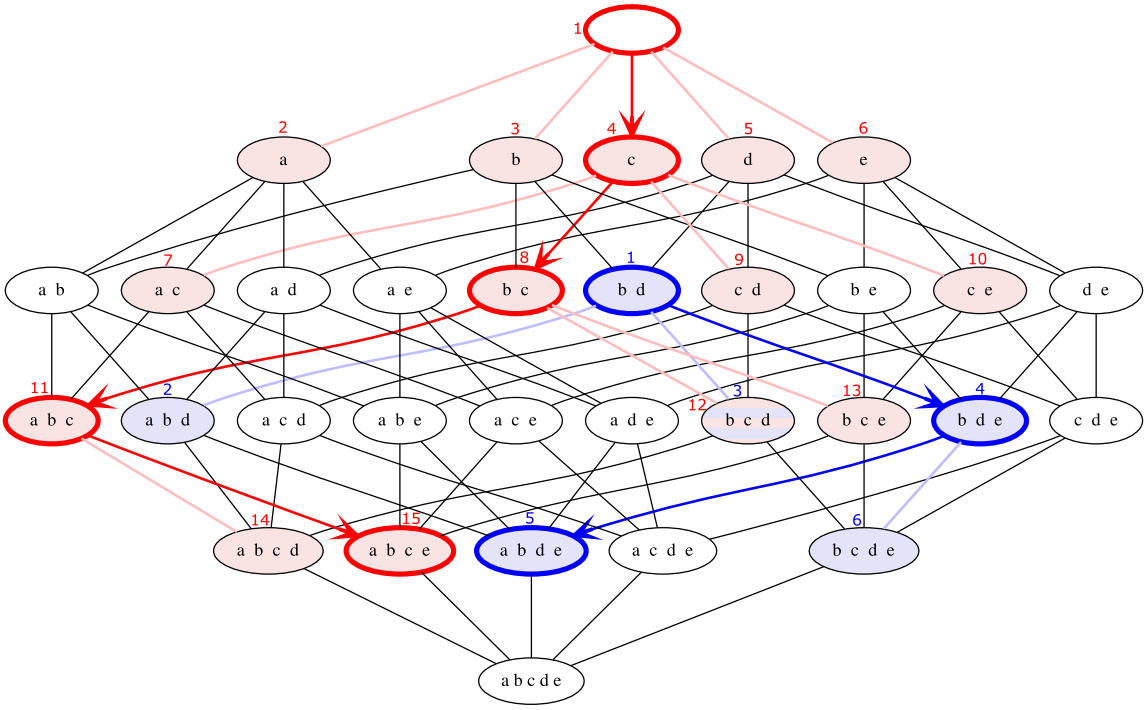

Figure 2: Example of search with: in red, classic SFS (Sequential Fordward Selection) algorithm; in blue, SFSwS (with Start).

Algorithm 1 shows the pseudo-code of SFS. Let $J(S)$ be an evaluation measure to be optimized (maximized in these algorithms) and $\overline{J}$ its supremum value. For clarity, in all pseudo-codes, we omit the code to keep, update and return the best set (the one with the maximum $J(S)$ value and less features in case of tie) found in the search.

Algorithm 1 SFS (Sequential Forward Selection)

$S_0 \leftarrow \varnothing$; $k \leftarrow 0$ ▷ Start with the empty set
while $S_k \neq F$ and $J(S_k) < \overline{J}$ **do**
  $x^+ \leftarrow \underset{x \notin S_k}{\operatorname{argmax}}\ J(S_k \cup \{x\})$ ▷ Select the next best feature
  $S_{k+1} \leftarrow S_k \cup \{x^+\}$; $k \leftarrow k + 1$ ▷ Update

In the opposite direction, SBS (Algorithm 2) starts from a set with all the features and eliminates those that improve or less degrade the valuation provided by the evaluation function. An example run is shown in figure 3.

Algorithm 2 SBS (Sequential Backward Selection)

$S_0 \leftarrow F$; $k \leftarrow 0$ ▷ Start with the full set of features
while $S_k \neq \varnothing$ **do**
  $x^- \leftarrow \underset{x \in S_k}{\operatorname{argmax}}\ J(S_k - \{x\})$ ▷ Find the least contributing selected feature
  $S_{k+1} \leftarrow S_k - \{x^-\}$; $k \leftarrow k + 1$ ▷ Update

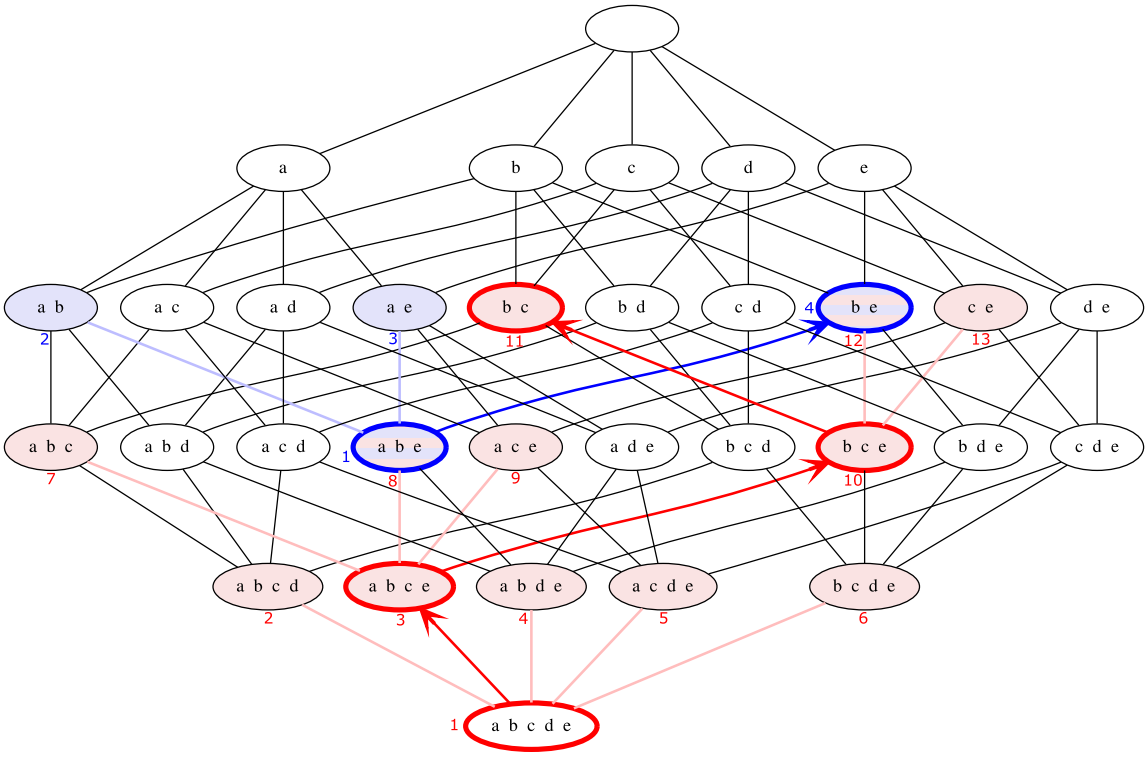

Figure 3: Example of search with: in red, classic SBS (Sequential Backward Selection) algorithm; in blue, SBSwS (with Start).

Probabilistic search algorithms follow some criterion that depends on a random component. As representatives of probabilistic algorithms, we have chosen Las Vegas Filter (LVF), which is a filter probabilistic feature selection algorithm designed for monotonic evaluation measures (Nandi, 2011); and Las Vegas Wrapper (LVW), which is similar but useful with non-monotonic measures, as when applying the wrapper approach (Liu & Setiono, 1996).

Algorithm 3 LVF (Las Vegas Filter)

1: $S_0 \leftarrow F$ ▷ Start with the full set of features
2: **for** $k \leftarrow 0$ to iterations **do**
3: $C \leftarrow \text{randomSubset}(F, \text{maxSize} = |S_k|)$
4: **if** $J(C) > J(S_k)$ or ($J(C) >$ threshold and $|C| < |S_k|$) **then**
5: $S_{k+1} \leftarrow C$

LVF (Algorithm 3) consists of randomly exploring sets with the same or smaller number of features than the best one found so far, finishing after a given number of iterations. It starts with all the features and the goal is to find the smallest subset with an evaluation over a given threshold.

Algorithm 4 LVW (Las Vegas Wrapper)

1: **for** $k \leftarrow 0$ to iterations **do**
2: $S_k \leftarrow \text{randomSubset}(F)$ ▷ Start with a random set of features

LVW (Algorithm 4) randomly evaluates sets of features and it simply obtains the one with the best value.

As a representation of the meta-heuristic algorithms, we have used Simulated Annealing (SA) (Kirkpatrick, Gelatt, & Vecchi, 1983) and Genetic Algorithm (GA) (Tan, Fu, Zhang, & Bourgeois, 2008).

Algorithm 5 SA (Simulated Annealing)

1: $t \leftarrow t_0$ ▷ Initial temperature
2: $S_0 \leftarrow \text{randomSubset}(F)$; $k \leftarrow 0$
3: **while** $t >= t_f$ **do** ▷ t_f, final temperature
4: **for** $i \leftarrow 0$ to n **do** ▷ n, number of neighbours
5: $C \leftarrow \text{generateRandomNeighbour}\ (S_k)$
6: $\lambda \leftarrow J(C) - J(S_k)$
7: **if** $\lambda > 0$ or $U(0,1) > e^{-\lambda/t}$ **then** ▷ $U(0,1)$, uniform random number in $[0,1[$
8: $S_{k+1} \leftarrow C$; $k \leftarrow k+1$
9: $t \leftarrow t_0/(k+1)$ ▷ Cauchy Annealing

SA (Algorithm 5) starts with a random set of features obtained from the complete set and at each step the algorithm generates a neighbour of the current set. The neighbour will be a set where the state of a random feature is swapped (selected if non selected and vice-versa). The transition to the new state is carried out according to the following rules: if the generated set is better than the current best one, it is directly accepted, otherwise, the generated set would be accepted with a certain probability. This probability is a function

of the difference of evaluation measures (energy of the states) and the temperature. The higher the temperature the more likely to accept solutions and explore. It is repeated until a certain established computational time is reached.

Algorithm 6 GA (Genetic Algorithm)

```
1: Initialize P_0                          ▷ P is a list of feature sets {S_0, ..., S_i, ...}
2: for k ← 0 to generations do
3:     Evaluate P_k
4:     Select P_{k+1} from P_k
5:     Cross and mutate P_{k+1}
```

GA (Algorithm 6) starts from a population of feature sets, randomly generated, and evolves towards the best one. In each generation, each feature set (chromosome) of the current population is evaluated (using an objective function) and those with better results are reproduced and relatively bad solutions die to be replaced by others, which present features of good solutions in the new population. Finally, the chromosomes selected to be part of the new population are altered using crossover and mutation operators.

3. Initialization Strategies Used in Genetic Algorithms for Feature Selection

In the genetic algorithms (GAs), the initialization is more relevant than in other search methods because, instead of one, several start points are chosen when the population is initialized. For this reason, in order to get an idea of which are the most used methods, this section revises the initialization strategies used in recent proposals of GAs for feature selection.

The methodology followed, in order to have a wide and representative review, was to carry out a bibliographic search in Google-scholar starting from 2014 until 2020 with the following search criteria: `allintitle: "Feature Selection" "Genetic Algorithm"`. Subsequently, to guarantee a minimum level of quality, the obtained documents have been filtered to consider only those published in journals included in Scimago or in CORE-level conferences. After this filtering, 216 papers have been analyzed.

We have excluded 38 publications for falling in one of these categories:

- Undefined: Some publications focus on the application area and do not fully describe the FS algorithm applied.
- Off-topic: Feature selection is performed outside of the GA.
- Spam: Despite having filtered by research publisher, we have found publications that are either nonsense text that mimic real contributions or very poor automatic translations that become unintelligible.
- Without access: A few publications to which we did not have access as they exceeded a reasonable budget for this review. The authors of these publications were contacted through Researchgate but no replies were received.

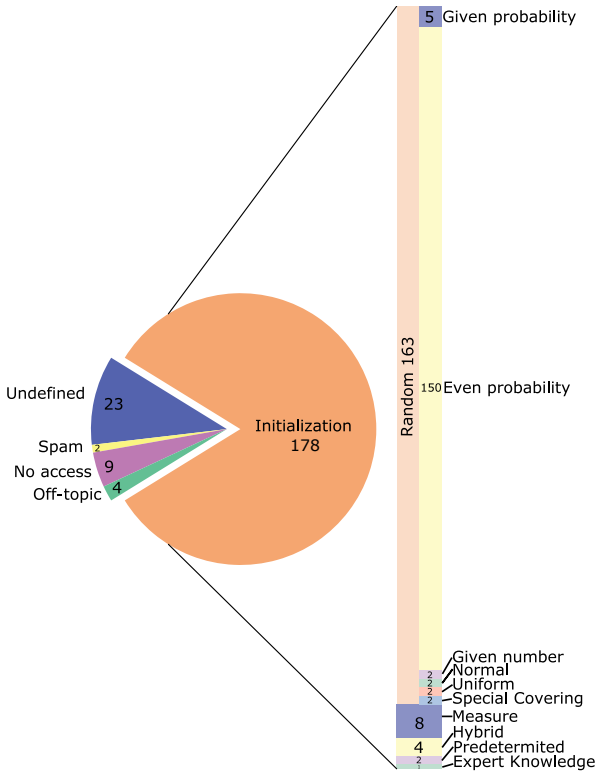

Figure 4: Literature review results classified in categories

In Figure 4, the distribution of the analyzed papers can be seen. The pie chart shows the percentage of the reviewed papers and the categorization of the excluded papers whereas the bar graphic provides the distribution of the reviewed ones according to its initializing strategy.

The discovered initialization methods can be divided into the following categories:

- **Random:** the individuals of the initial population are generated by some stochastic process. No other information is considered, thus all the features are equally likely to be chosen. However, the size of the feature sets selected depend on the sampling mechanism applied. The search space representing all possible feature sets have a much higher density of intermediate set sizes. Figure 5 shows this distribution illustrated with red circles and the size distribution of the sets sampled by each random initialization. Considering this, the motivation for using each sampling mechanism is commented below.

  The great bulk of the revised proposals (163 articles) belong to this category. We have subdivided it into several subcategories related to the different used probability distributions:

  - **Given probability**: each feature has a certain probability to be included in the initialization solution. It can be given as a parameter or established in the proposal. The population uses binary encoding feature sets as individuals (strings

of 0 or 1 values are generated, where a 1 represents a selected feature). For each feature, a 1 gets assigned with the given probability.

Only five papers from our review have explicitly applied this generic approach (Jing, 2014; Anirudha, Kannan, & Patil, 2014; Szenkovits, Meszlényi, Buza, Gaskó, Lung, & Suciu, 2017; Gupta & Purohit, 2017; Ahn & Hur, 2020).

The given probability regulates the number of features selected in the initial individuals. This allows to tune it according to the knowledge domain of each learning scenario. For example, a lower p allows to search for smaller sets of features, more adequate for a specific learning algorithm. The size of the feature sets generated with this method follows the binomial distribution $B(n, p)$ (a discrete probability distribution that counts the number of successes in a sequence of n independent Bernoulli trials with a fixed probability p of success). With this distribution, the expected average size of feature sets is $p \cdot n$, being n the total number of features. Figure 5 shows two examples (for $p = 0.1$ and 0.5) of the distribution of feature set sizes.

– **Even probability**: a special case of the given probability category in which the probability of including a feature is the same as not including it ($p = 0.5$). It was named Bitstring-uniform procedure by (Kallel & Schoenauer, 1997).

  As this is the direct interpretation of choosing a bit randomly, most of the revised papers (163) use this approach. We have included, in this category, several papers which do not explicitly state the probability but, they either use binary coding and initialize randomly, such as (Raman, Somu, Kirthivasan, Liscano, & Sriram, 2017), or they implement a standard GA. As clear examples, Shukla, Singh, and Vardhan (2019) and Ge, Zhang, Liu, and Sun (2019) describe the initialization in detail.

  Using even probability mimics the search space, so this is the method that samples all the sets in the whole search space with the same probability.

– **Given number**: the initial population gets filled with random sets of a specific number of features. Our review has found only two proposals using this approach. Vieira, Garcia, Pabón, Cota, de Souza, Ueyama, and Pessin (2020) test three options: initializing with sets containing the 20% of all available features, the same with the 80%, and the even probability strategy. In their specific problem, the 20% strategy provides significantly better results. In addition, Sahu, Dehuri, and Jagadev (2017) group the features in k clusters and randomly take one from each cluster. In this way, they reduce redundancy among the selected features.

  Using a given number of features focuses on a much reduced search space. Thus the adequate size can be chosen for each learning algorithm.

– **Normal**: the size of initial feature sets follows a normal distribution $N(\mu, \sigma^2)$.

  Ma and Xia (2017a) and Ma and Xia (2017b) explore the space using a tribe. Each tribe is a GA initialized with individuals whose size follows the Normal distribution. In this way, each GA explores a part of the search space.

  In the given probability strategy, when reducing probability, the size gets reduced but there is a strong peak with most sets getting near the expected size (see

figure 5). However, by using the Normal strategy the same reduction can be attained while controlling the dispersion at the same time.

- **Uniform**: all feature set sizes have the same probability. Named uniform covering procedure (Kallel & Schoenauer, 1997), it first gets a random density of 1's (uniformly in $[0, 1]$) for each bit string, then it chooses each bit to be 1 with this specific probability. Therefore, it leads to a continuous probability distribution of set sizes.

  Only two proposals have applied this initialization: (Priya & Sivaraj, 2017) and (Rahmadani, Dongoran, Zarlis, et al., 2018).

  Uniform method samples the space in a way that may fit well to the goal of detecting how many features should be used.

- **Homogeneous block**: creates blocks of contiguous 1's (Kallel & Schoenauer, 1997). It might be useful in problems where the features were sorted in some way where adjacent features provide complementary information. No papers have been found using this approach in our review.

- **Special coverings**: initialize covering parts of the search space with the initial population in some pseudo-randomized way.

  Two interesting ideas have been found in our review. One is running cellular automata, one automaton for each feature, in parallel, to create one individual in each state (Das, Pati, & Ghosh, 2020). The other applies chaos maps to assign a probability (real number in $[0, 1]$) to each gen representing a feature and, then, uses a random generator to select the feature with that chaotic probability (Tahir, Tubaishat, Al-Obeidat, Shah, Halim, & Waqas, 2020). Both proposals lack of a clear explanation about the motivation for the specific covering procedure designed. Nevertheless, they seem good ideas that provide good results and probably deserve deeper research.

- **Measure:** this category includes all methods using some measure over individual features to initialize the population.

  The most versatile approach of this category assigns a probability, depending on the measure, to each feature to generate individuals randomly. For example, Jiang, Chin, Wang, Qu, and Tsui (2017) randomly generate excellent initial individuals by selecting features using the probability from a t-test over the multiple linear regression coefficients. One third of the population is generated in this way and the rest uses an even probability. Likewise, Park, Park, Kim, and Lee (2020) create clusters of features which are assigned to different populations. Each population gets initialized with individuals whose features are selected with a probability proportional to their entropy in its cluster. Finally, Wang, Zhang, Bai, and Mao (2018) apply F-score, Gain ratio and the Pearson correlation coefficient to rank features and, then, as cut-off method, they add the features to a learning algorithm until it stops improving. This leads to three selected feature sets, which are included in the initial population. The rest of the individuals are generated with a probability for each feature proportional to the results of the first phase.

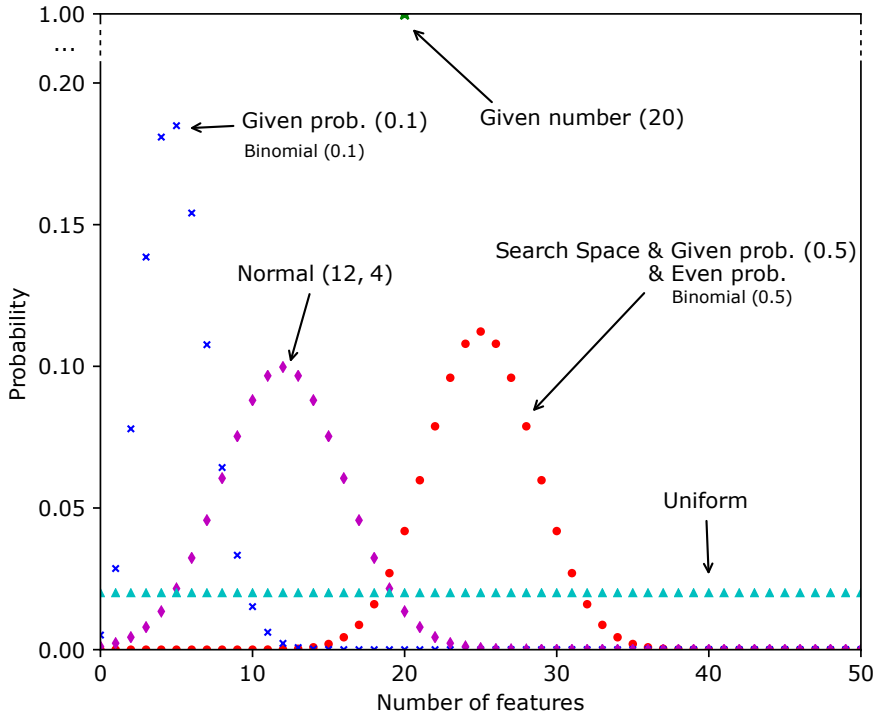

Figure 5: Distribution of feature set size for each random initialization identified (in a search space with 50 features)

In contrast, another approach is to use the measures just to set one individual. Mao, Zhang, and Fan (2016) use a filter approach with F-score, Relief and Gain ratio as measures to generate the first individual of the initial population (we understand that using the union of all features selected) and the rest are randomly generated (even probability considered by default).

Another interesting idea is to fixate the selection of some important features. (Bai, Fan, Zhang, Xu, & Zhang, 2017) use T-test as a measure to rank features. Then they force the selection of the top features (according to a parameter) in all initial individuals. The rest of the features are randomly selected in each individual.

The following three proposals perform a previous filter reducing the features that will be considered. In this way, the initial population and the whole GA only use those reduced feature set. They are not exactly initialization strategies and they could also fit well in the next category (hybrids) but we mention them in this category because well-known individual measures are used in the initialization. To begin with, Oreski and Oreski (2014) take the union of the top features evaluated by Information Gain, Gain Ratio, Gini Index and Correlation measures, together with some features selected by an expert. Then the initial population is generated randomly with a given probability. Similarly, Jadhav et al. (2018) apply the Information Gain measure to rank the features of the initial data set and subsequently use only those top features. In the case of (Singh & Selvakumar, 2015), they use the measures Information Gain, OneR attribute evaluation, Gain Ratio, Chi-square and Relief.

- **Hybrids:** in this category, other optimization methods are used to start the candidate population of the GA.

  For example, Bian, Peng, Wang, and Zhang (2016) use a Tent map from chaos theory to perform a previous search to initialize the population. Another approaches propose to generate the initial population of the GA by combining the SFS and SBS search methods to select the best initial features (Homsapaya & Sornil, 2017, 2018). Finally, a remarkable approach (Lin, Wang, Xiao, Huang, & Wang, 2015) uses the groups created by Fast Correlation Based Filter (FCBF). Each individual contains one randomly chosen feature from each group. In this way, the initial solutions contain less redundant features.

- **Predetermined:** some methods set the selection of features without conditions or randomness.

  Chen, Lin, Tang, and Xia (2016) initialize all individuals with all features selected (all bits set to 1 in the binary coding). It seems reasonable because it optimizes other learning parameters encoded in the individuals at the same time (individuals are varied). Similarly, Nguyen, Liew, Tran, Pham, and Nguyen (2014) initialize just one individual to ensure that the option of selecting all features is evaluated. The remaining individuals are initialized randomly.

- **Expert knowledge:** an expert in the problem domain creates an initial population with crucial features.

  For example, the intrusion detection system proposed in (Gharaee & Hosseinvand, 2016) generates the initial population from the results of previous investigations in each type of attack, using the most important features in each chromosome for each class of attack.

4. Proposed Modular Selection of Starting Feature Sets

The proposal is to evaluate the use of individual feature selection methods embedded in search based feature subset selection methods. This is defining a general hybrid approach combining two concrete types of feature selection methods, so it fits in the hybrid category of the previous classification. Individual methods implement an evaluation function that analyzes all the features and afterwards, a number of them are selected according to some established criterion. These selected features will be used as the feature set to start the search based feature selection.

Most of the search methods in conventional feature selection start with a given set. For example, the empty set in *SFS (Sequential Forward Selection)*, the set of all features in *SBS (Sequential Backward Selection)*, or a set of random features in *SA (Simulated Annealing*). See table 1.

For all the search methods that start with a single feature set, the size is determined by the cutting criterion. The two cutting criteria considered in this paper make it fixed as in *Given number* methods. In the case of the GA, where several sets are needed, there is a random sampling from two sets with an effect similar to the methods in *Given probability*.

Method	Efficiency	Start Set
SFS	$O(n^2)$	$\emptyset$
SBS	$O(n^2)$	The set with all features
LVF	$O(k)$	The set with all features
LVW	$O(k)$	A random feature set
SA	$O(n * k)$	A random feature set
GA	$O(k)$	A set of random feature sets

Table 1: Properties of the well known search methods in feature selection

The proposal can reduce the computing time by setting the search on a shorter path (as an example, see figure 2). Moreover, the accuracy of feature selection and the reduction (in terms of a lower number of selected features) may improve, by placing the search method in a more interesting point of the search space.

Figure 6, shows an schema of how the starting method works inside the modularization given by figure 1. Each of the features in {a, b, c, d, e} gets evaluated with an evaluation function of individual features. Subsequently, the features that have exceeded the cutting criterion are preselected ({a, c, d}). With these features, the search method initiates the search process. It is not a filtering process, the search method can also use the features initially dismissed by the starting method.

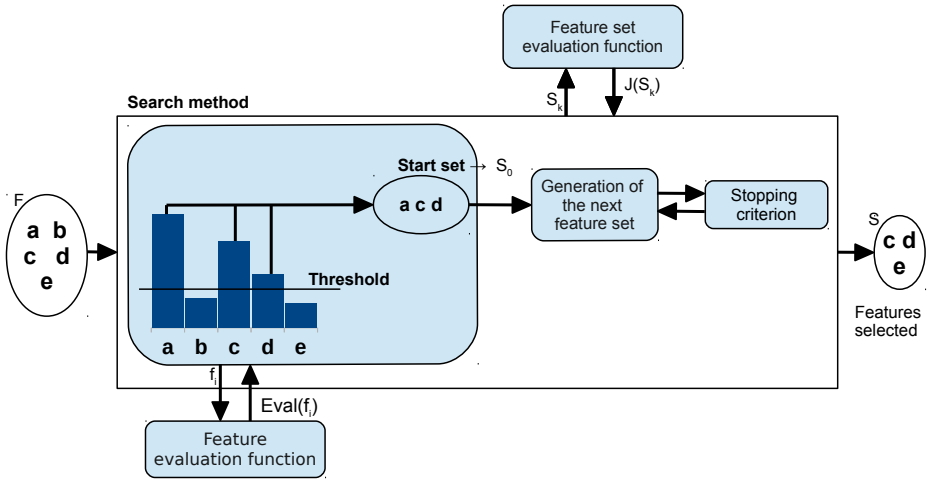

Figure 6: Starting method inside modularized feature selection

The flexibility of the modularized proposal allows any FS method to be used as a starting method. This paper focuses on individual feature selection methods because they are the fastest and, as a result, the most suitable for this hybridization.

In the following sections we will discuss the implications of fixing a starting set for each method.

4.1 Sequential Search with Start (SFSwS and SBSwS)

The classic SFS and SBS algorithms start from the empty set or the complete set of features and they add or remove features, respectively. Their respective hybrid FS methods with the start module will be denoted as SFSwS and SBSwS henceforth. See figures 2 and 3 for an illustration of an example search.

Since these proposed methods are reducing the search path and, as a result, they are reducing the number of feature set evaluations, a reduction of the computing time is expected. We expect the number of features selected with SFSwS to increase because it starts with a minimum, while we expect a reduction with SBSwS for the opposite reason. As both of them preselect a series of good features, they may match or improve the accuracy.

4.2 Probabilistic Search with Start (LVFwS and LVWwS)

The version of the probabilistic algorithms with the starting approach are denoted LVFwS and LVWwS. Figure 7 shows an example run of LVF. Note that it moves randomly exploring ignoring any relation among sets but it cannot increase the size of the current feature set.

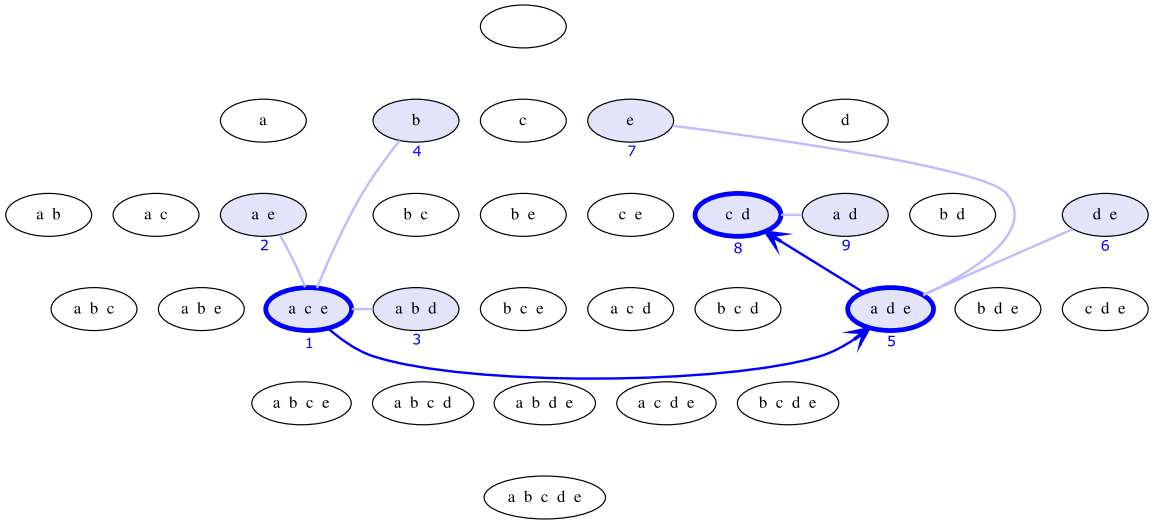

Figure 7: Example of search LVF algorithm with Start (Las Vegas Filter with Start).

Since, in both probabilistic algorithms, the number of evaluations is established by a parameter and kept the same with the a priori start method, our hypothesis is that: a) running time will be similar; b) the number of features will get reduced in LVFwS, as it imposes a maximum size, and kept equal for LVWwS; c) the accuracy will improve.

4.3 Simulated Annealing with Start (SAwS)

Tough not illustrated, an example search is easy to be imagined over figure 3. SAwS moves finding neighbours (adding or removing features) following relation lines as SFS and SBS do but randomly and in both senses.

As the probabilistic methods, SA has a fixed number of evaluations thus running time will be similar. We expect the start method to reduce the number of features selected or the accuracy achieved with the start method.

4.4 Genetic Algorithm with Start (GAwS)

To improve the classical algorithm, our proposal applies the start method to the initialization of the population. In contrast to other proposals, GAwS follows the modularized approach, so that any FS method can be used to initialize the population, not only those based on individual measures. In order to ensure that the initial population is diverse, the initial sets take features from the preselected features (Q) and the non-preselected ($\overline{Q}$), if necessary. Therefore, we have applied a probability to the pre-selected Q as well as the non-preselected features. These probabilities are assigned as shown in figure 8. The parameter P_{desired} establishes the probability that regulates the expected number of features to be selected. Equally important, $P_{\max}$ regulates when to start taking features from the non-preselected to ensure diversity.

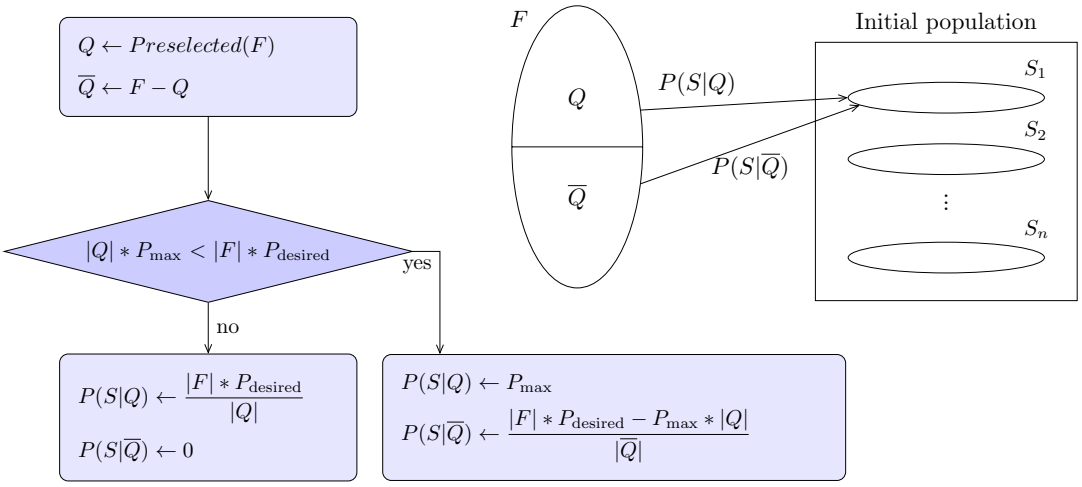

Figure 8: Initialization of the genetic algorithm population with the starting modularized approach (GAwS)

In this approach, since the number evaluations is fixed, we expect a similar running time as in a genetic algorithm with a simple random start. However, since we are doing a pre-selection of good features, we expect to obtain better results in reduction and accuracy.

5. Materials and Methods

In this section, we provide a detailed description of the experimental methodology.

5.1 Experimental Design

The aim is to compare each classic feature subset selection method presented with its corresponding method started using individual feature selection.

The dependent variables to evaluate the results of feature selection are:

- The accuracy rate of classification (*Acc*). The data sets we have chosen are mostly balanced, which means the number of cases in each class is approximately the same. In this case, we choose accuracy as the metric to measure the performance due to its simplicity.
- The number of selected features (*Nof*).
- The time spent in feature selection (*FSTime*).

In order to get a reliable estimate of these variables, every experiment has been performed using 10-fold stratified cross-validation. For each experiment, we have taken the mean and standard deviation of the ten folds.

In these experiments, there are several factors:

1. A starting method, with sub-factors:
   - Cutting criterion
   - Evaluation function of individual features
2. A feature subset selection method, with sub-factors:
   - A search method
   - Evaluation function of a set of features
3. A learning algorithm that generates the classifier.
4. A classification problem represented in a data set (*Data set*).

The design of the global experiment is complete, in order to subsequently be able to study the interactions of all factors, alone or together. This means that all the possible combinations among the factors are tested. However, there are some exceptions with the Wrapper measure. It has not been tested with the largest data sets (taking the size as the product of the number of features by the number of examples): Adult, Anneal, Audiology, Car, Ionosphere, led24, Mushrooms, Soybean, Splice, Vehicle, Wdbc, Yeast, and Yeast-class-RPR.

5.2 Data Sets

In order to include a wide range of classification problems, the following publicly available repositories were explored seeking for representative problems with diverse properties (discrete and continuous data, different number of classes, features, examples, and unknown values): UCI (Newman & Merz, 1998), OPENML (Demšar et al., 2013) and a dataset called *Parity3+3* generated artificially. Finally, 27 data sets were chosen. They are listed along with their main properties in Table 2:

- **Data set** column shows the name by which data sets are known.
- **Ex.** is the number of examples (tuples) in the data set.
- **Feat.** is the number of features.
- **Type** of features: Discr. (all are discrete), Cont. (all are continuous) or Mixed (both types).
- **Cl.** is the number of classes.

Data set	Examples	Feat.	Type	Cl.	Source
Appendicitis	106	7	Continuous	2	OpenML
Australian	690	14	Continuous	2	UCI
Bands	365	19	Continuous	2	UCI
Breast-cancer	286	9	Mixed	2	UCI
Cleveland (Heart Dis.)	297	13	Continuous	5	UCI
Contraceptive	1473	9	Continuous	3	UCI
Echocardiogram	131	10	Mixed	2	UCI
Glass	214	9	Continuous	6	UCI
Statlog Heart	270	13	Continuous	2	UCI
Hepatitis	155	19	Continuous	2	UCI
Horse-colic	368	26	Mixed	2	UCI
House-votes84	435	16	Discrete	2	UCI
Ionosphere	351	32	Continuous	2	UCI
Labor-relations	57	16	Mixed	2	UCI
Lung-cancer	32	56	Discrete	3	UCI
Parity3+3	500	12	Discrete	2	Artificial
Pima	768	8	Continuous	2	UCI
Primary-tumor	339	17	Discrete	21	UCI
Promoters	106	57	Discrete	2	UCI
SAheart	462	9	Mixed	2	OpenML
Soybean	307	35	Discrete	19	UCI
Tic-tac-toe Endgame	958	9	Discrete	2	UCI
Vowel	990	10	Continuous	11	OpenML
Wine	178	13	Continuous	3	UCI
Wisconsin	683	9	Continuous	2	UCI
Yeast	1484	8	Continuous	10	UCI
Zoo	101	16	Discrete	7	UCI

Table 2: Data sets used in the experimentation

5.3 Classifiers

In order to estimate the quality of the feature selection process executed by each method, the experiments are performed in a full learning environment for classification problems.

A set of well known methods have been considered. These methods have been chosen to cover each category to which the most used methods belong. They are: Naive Bayes (NBayes), a simple probabilistic classifier; the K Nearest Neighbor (kNN), an algorithm based on the assumption that closer examples belong to the same class; C4.5 (C45), a decision tree based classification; Multi-Layer Perceptron (ANN), an artificial neural network; and Support Vector Machines (SVM).

5.4 Development and Running Environment

The feature selection methods have been programmed in Python. The software used for learning methods has been Orange (Demšar et al., 2013) component-based data mining software, except for artificial neural networks, where SNNS (U. of Stuttgart,1995) was used, integrated in Orange with OrangeSNNS package.

Experiments have run on a cluster of 8 nodes with Intel Xeon E5420 CPU 2.50GHz processor and 2 nodes with Intel Xeon E5630 CPU 2.53GHz, under Ubuntu 16.04 GNU/Linux operating system.

5.5 Parameters and Data Transformations

All evaluation functions are parameter free except Relief-F. For this measure, the number of neighbours to search was set to 6, and the number of instances to sample was set to 100.

Some of the learning algorithms require parameter fitting. In the case of kNN, k was set to 15 after testing that this value worked reasonably well on all data sets used. The multi-layer perceptron used has one layer trained during 250 cycles with a propagation value of 0.1. For SVM we used *Orange.SVMLearnerEasy* method to fit parameters to each case automatically.

Besides, consistency and information measures require discrete valued features. For this reason, after some preliminary tests with equal frequency and equal width discretization methods, the later with six intervals was chosen. This is only applied for feature selection. Then learning algorithms get the features with the original data.

5.5.1 Parameters of the Search Methods

The SFS and SBS search methods are completely determined and, therefore, have no parameters to set. The two probabilistic algorithms of the Las Vegas type are similar. Its main parameter is the number of limit assessments, which in both has been set to 1000 feature sets. LVF needs to limit the reduction allowed for the relevance of the features and it has been set at 1%.

In metauristic methods (SA and GA) the first restriction we have used in their parameters is that they perform 1000 evaluations.

In SA it is necessary to set:

- Initial temperature: We use $T_0 = (-v/-\ln(\phi))(S_0)$, which allows a probability ϕ of accepting a solution that is v by one worse than the initial solution S_0. We take $v = 0.3$, and $\phi = 0.3$.

- Generation of neighbours: The generation of new neighbours is done by adding or eliminating features of the current set. 20 neighbours are generated in each cooling, that is, for each temperature value.

- Cooling scheme: We have used the Cauchy one ($T = T_0/(1+i)$). 50 coolings are carried out.

In GA it is necessary to set:

- A generational type, with a population of 40 individuals and 50 generations.

- Simple one point crossover, on the binary representation of the set of selected features. Crossover probability: 0.6.

- The mutation adds or removes one feature. Mutation probability: 0.001.

- For the initialization method, to calculate the probability of preselecting features in the GA, after conducting some preliminary experiments, the parameters that obtained good results are the following:

  - $P_{desired} = 0.5$
  - $P_{max} = 0.8$

6. Experimental Results

This section presents the results divided into two sections. First, the best starting method (an individual measure with a cutting criterion) and its parameters are established for each FS search. Then, the results comparing the classical search methods with its started versions are examined.

6.1 Starting Method Setup

The starting method has two parts:

- The evaluation function of individual features.

- The cutting criterion. To carry out a selection of the chosen parameters, we have taken into account the best results obtained in (Arauzo-Azofra, Aznarte, & Benítez, 2011). The parameters tested in each of the cutting criteria are as follows:

  - **Fixed number (denoted as n-n)** $n \in \{9, 13, 17\}$
  - **Fraction (denoted as p-p)** $p \in 0.2, 0.5, 0.8\}$

As we have five individual measures and six cutting criterion possibilities, there are a total of thirty options. Table 3 shows the best performing starting method for each feature selection method. In order to obtain this, we have used 3 independent data sets from the rest of the selected sets for experimentation. One of a small size (appendicitis), another of a medium size (lung-cancer) and another of a large size (horse-colic) as shown in the table 2. The best parameter and measure combination is chosen according to its classification accuracy in the average ranking among the three data sets.

Table 3: Best parameter and individual measure for each search method

Search	Parameters	Individual measure
SFSwS	n-9	reli
SBSwS	n-9	gain
LVFwS	n-9	info
LVWwS	n-9	info
SAwS	p-0.2	reli
GAwS	n-13	gini

6.2 Comparisons Between Classical Methods and Started Methods

Our goal is to test the hypothesis that using the started search methods proposed is better in some scenarios. In order to compare the performance of selected features, a classification learning algorithm needs to be trained and evaluated. All these comparisons use the average of ten fold cross-validation to get an stable and confident result. As an example, figure 4 details the first comparison using SFS search with IE measure and evaluated with ANN. The Wilcoxon test will determine if the differences found are significant and this is what the following tables summarize.

This section deals with a series of comparisons between the classical and the started methods, as described above. The effectiveness of feature selection is evaluated with five classifiers (ANN, C4.5., KNN, Naive-Bayes and SVM). The search methods are combined with the three set measures described in section 2.2: Inconsistent examples (IE), Mutual Information (Inf) and Wrapper (Wra). Therefore, for each search, there are a total of fifteen possible scenarios to evaluate the started approach in three aspects: the accuracy, the reduction in the number of selected features and feature selection time. Feature selection reductions and feature selection times do not vary with the classifier, except for the wrapper measure. This leaves us with tables 5-10 that show the results of all Wilcoxon test confronting classical versus started search methods. The tables indicate which is better, the classical method or the method with a starting set, for each aspect: the best accuracy, highest reduction and fastest computing. The differences are indicated if they are intuitive (p-value < 0.2) and boldfaced if significant (p-value < 0.1).

These results lead to the following diagnosis. First, for the SFS algorithms (table 5), as expected (see section 4.1), the started method is faster but it increases the number of selected features. The started methods improve their counterparts in classification accuracy when using the faster set measures (IE and Inf) but not with the wrapper approach. However,

	SFS			SFSwS		
Dataset	**Acc**	**NoF**	**FSTime(s)**	**Acc**	**NoF**	**FSTime(s)**
Australian	0.845	8.6	1.127	0.864	11.0	0.174
Bands	0.603	6.2	0.661	0.661	9.2	0.040
Breast-cancer	0.633	8.0	0.609	0.674	9.0	0.022
Cleveland (Heart Disease)	0.576	7.0	0.312	0.566	10.9	0.051
Contraceptive	0.555	9.0	0.905	0.555	9.0	0.080
Echocardiogram	0.908	4.1	0.208	0.900	9.0	0.010
Glass	0.585	5.9	0.157	0.560	9.0	0.011
Statlog Heart	0.781	7.1	0.278	0.833	10.0	0.031
Hepatitis	0.812	5.2	0.810	0.736	10.2	0.049
House-votes84	0.952	10.1	1.048	0.947	12.4	0.188
Ionosphere	0.917	5.1	2.727	0.889	9.2	0.064
Labor-relations	0.873	3.5	0.154	0.917	9.0	0.007
Parity3+3	1.000	4.5	0.160	1.000	9.0	0.037
Pima	0.768	7.5	0.403	0.767	8.0	0.037
Primary-tumor	0.439	15.8	0.592	0.439	15.8	0.295
Promoters	0.802	4.1	0.227	0.895	9.0	0.025
SAheart	0.684	6.2	0.527	0.736	9.0	0.027
Soybean	0.814	10.2	1.049	0.886	15.9	1.022
Tic-tac-toe Endgame	0.952	8.0	0.651	0.984	9.0	0.039
Vowel	0.656	7.2	2.104	0.688	9.1	0.092
Wine	0.944	4.0	0.381	0.978	9.0	0.022
Wisconsin	0.963	5.1	0.395	0.972	9.0	0.037
Yeast	0.578	7.7	1.463	0.586	8.0	0.089
Zoo	0.940	4.9	0.056	0.950	9.8	0.009

Table 4: Comparison of SFS search without and with start method, selecting the best IE measure value and evaluating the feature selection cross-validating with an ANN (summarized in the first line of Table 5)

Table 5: Wilcoxon test comparing SFS and SFS started

Meas.	Cla.	Best Acc.	p-value	Hightest Reduc.	p-value	Fastest	p-value
IE	ANN	**started**	**0.027**				
IE	C4.5	started	0.110				
IE	KNN	**started**	**0.042**	**classic**	**0.000**	**started**	**0.000**
IE	NBayes	—	0.615				
IE	SVM	**started**	**0.004**				
Inf	ANN	started	0.108				
Inf	C4.5	**started**	**0.003**				
Inf	KNN	—	0.926	**classic**	**0.001**	**started**	**0.000**
Inf	NBayes	—	0.896				
Inf	SVM	**started**	**0.014**				
Wra	ANN	—	0.555			**started**	**0.001**
Wra	C4.5	—	0.961			**started**	**0.000**
Wra	KNN	—	0.251	**classic**	**0.000**	**started**	**0.000**
Wra	NBayes	**classic**	**0.008**			**started**	**0.000**
Wra	SVM	started	0.177			**started**	**0.000**

Table 6: Wilcoxon test comparing SBS and SBS started

Meas.	Cla.	Best Acc.	p-value	Hightest Reduc.	p-value	Fastest	p-value
IE	ANN	—	0.476				
IE	C4.5	—	0.676				
IE	KNN	classic	0.116	—	0.395	**started**	**0.000**
IE	NBayes	—	0.281				
IE	SVM	—	0.856				
Inf	ANN	—	0.339				
Inf	C4.5	—	0.639				
Inf	KNN	classic	0.117	—	0.500	**started**	**0.000**
Inf	NBayes	—	0.259				
Inf	SVM	—-	0.810				
Wra	ANN	**classic**	**0.042**	**started**	**0.001**	**started**	**0.000**
Wra	C4.5	—	0.910	**started**	**0.000**	**started**	**0.000**
Wra	KNN	—	0.751	**started**	**0.001**	**started**	**0.000**
Wra	NBayes	**classic**	**0.053**	**started**	**0.000**	**started**	**0.000**
Wra	SVM	—	0.685	**started**	**0.004**	**started**	**0.001**

Table 7: Wilcoxon test comparing LVF and LVF started

Meas.	Cla.	Best Acc.	p-value	Hightest Reduc.	p-value	Fastest	p-value
IE	ANN	—	0.615				
IE	C4.5	—	0.465				
IE	KNN	—	0.751	classic	0.192	**started**	**0.000**
IE	NBayes	—	0.751				
IE	SVM	—	0.931				
Inf	ANN	started	0.185				
Inf	C4.5	**classic**	**0.024**				
Inf	KNN	—	0.587	—	0.937	**started**	**0.000**
Inf	NBayes	—	0.986				
Inf	SVM	—	0.951				

Table 8: Wilcoxon test comparing LVW and LVW started

Meas.	Cla.	Best Acc.	p-value	Hightest Reduc.	p-value	Fastest	p-value
Wra	ANN	**started**	**0.009**	—	0.531	**started**	**0.037**
Wra	C4.5	—	0.276	—	0.772	started	0.137
Wra	KNN	—	0.543	—	0.738	**started**	**0.092**
Wra	NBayes	—	0.444	started	0.166	—	0.360
Wra	SVM	—	0.614	**classic**	**0.065**	—	0.931

Table 9: Wilcoxon test comparing SA and SA started

Meas.	Cla.	Best Acc.	p-value	Hightest Reduc.	p-value	Fastest	p-value
IE	ANN	—	0.983				
IE	C4.5	**started**	**0.036**				
IE	KNN	—	0.227	—	0.246	**started**	**0.000**
IE	NBayes	—	0.422				
IE	SVM	—	0.681				
Inf	ANN	—	0.906				
Inf	C4.5	—	0.896				
Inf	KNN	—	0.867	—	0.894	**started**	**0.000**
Inf	NBayes	—	0.401				
Inf	SVM	—	0.904				
Wra	ANN	—	0.931	—	0.135		
Wra	C4.5	—	0.944	**classic**	**0.039**		
Wra	KNN	—	0.594	classic	0.118	**started**	**0.000**
Wra	NBayes	—	0.396	—	0.975		
Wra	SVM	—	0.538	—	0.904		

Table 10: Wilcoxon test comparing GA and GA started

Meas.	Cla.	Best Acc.	p-value	Hightest Reduc.	p-value	Fastest	p-value
IE	ANN	—	0.267				
IE	C4.5	started	0.199				
IE	KNN	—	0.615	**classic**	**0.003**	**classic**	**0.000**
IE	NBayes	**started**	**0.025**				
IE	SVM	started	0.181				
Inf	ANN	**started**	**0.073**				
Inf	C4.5	started	0.809				
Inf	KNN	**started**	**0.013**	**classic**	**0.000**	**classic**	**0.000**
Inf	NBayes	—	0.717				
Inf	SVM	—	0.485				
Wra	ANN	**started**	**0.019**	**classic**	**0.001**	—	0.330
Wra	C4.5	—	0.689	**classic**	**0.002**	—	0.670
Wra	KNN	—	0.478	**classic**	**0.003**	—	0.808
Wra	NBayes	—	0.751	**classic**	**0.001**	classic	0.144
Wra	SVM	started	0.110	**classic**	**0.001**	**classic**	**0.064**

except for the advantage of the classic method with naive Bayes, no significant differences have been found using the wrapper measure.

In the case of the SBS algorithm (table 6), contrary to SFS, the started version does not improve classification accuracy. Nevertheless, this has happened by applying greater reductions in a faster process, as expected.

As far as the LVF and LVW algorithms are concerned (tables 7 and 8), the results contradict our hypotheses. We expected a similar computing time (see section 4.2) but the started version of LVF has proved significantly faster. This may be because, although the number of feature sets evaluated is the same, by starting with a good smaller set, the average size of sets evaluated is smaller and the evaluation of smaller sets is faster. In reduction, we expected it to be greater for the started method. However, it seems that the search process avoids reducing the sets more and the results are very similar. About accuracy, there are disparities in results and only a few scenarios have shown significant differences. In the wrapper approach with the artificial neural network, the started approach has been significantly better, while the opposite happened with the mutual information measure in a C4.5 learning process.

As for the SA algorithm (table 9), the accuracy has improved only in one scenario but the started method has been faster in all cases. On the contrary, it has not improved the reduction of features. Since the parameter of the starting method used is a percentage of 20% of the number of features (table 3), it may be difficult to improve the reduction with respect to the classical methods with a such a small percentage of features selected with the start method.

In the case of the GA algorithm (table 10), at the cost of an inferior reduction and slower computation, the started method has beaten to its classical counterpart in accuracy.

Considering the results with all search methods, the starting methods do not excel when they are requested to prune features to select. This happens in SBS and LVF where the direction of the search is towards the reduction of features. However, they seem to work well selecting an initial set to grow, like SFS and GA do.

Finally, these results support thinking that the starting methods are an interesting approach to include in the feature selection toolbox of machine learning classification because they have been proven to provide advantages in several scenarios. Tables 5-10 may serve as a guide to know when to use an starting method.

7. Conclusions and Future Work

The context of search based feature selection for classification problems has been revised and structured in a modular definition of the process. Inside this modular structure, we have focused on the initialization. This paper characterizes all the initialization methods by their semantics and their distribution of the feature set size.

After revising the most relevant papers from 2014 to 2020 that deal with the initialization of genetic algorithms for feature selection, we have identified the relative use of each initialization strategy in literature. This has revealed that the most used methods are just the simplest ones and little attention has been set on the performance of the initialization. It can be concluded that most studies uses inferior performance initiation strategies.

This paper proposed the use of individual feature selection methods as the starting method for the search involved in feature subset selection methods. It has been systematically tested over several well known feature selection methods (SFS, SBS, LVF, LVW, SA and GA) on 27 classification problems and evaluated with five learning algorithms.

After the evaluation, we can conclude that the accuracy achieved has clearly improved in most of the scenarios using SFS and GA, while computing time spent is reduced when using the starting methods. In contrast, the results on the reduction of the number of features selected are mixed, when using inconsistent example measures, the number of features seems to grow using started methods, while when using wrapper and mutual information measures, the largest reduction of selected features is often carried out by some started search methods.

We hope that this paper will open new opportunities for researching improvements on the initialization of feature selection methods. The modularized proposal allows to use any feature selection method as the starting mechanism of a feature search. We believe this may lead to many different proposals in a well organized development frame that eases comparison.

Acknowledgments

This research is partially supported by projects P18TP5168 funded by Junta de Andalucía (Spain), PID2020-118224RB-I00 and PID2020-115832GB-I00, funded by Ministerio de Ciencia e Innovación (Spain).

References

Ahn, G., & Hur, S. (2020). Efficient genetic algorithm for feature selection for early time series classification. *Computers & Industrial Engineering*, *142*, 106345.

Aljawarneh, S., Aldwairi, M., & Yassein, M. B. (2018). Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*, *25*, 152–160.

Anirudha, R., Kannan, R., & Patil, N. (2014). Genetic algorithm based wrapper feature selection on hybrid prediction model for analysis of high dimensional data. In *Industrial and Information Systems (ICIIS), 2014 9th International Conference on*, pp. 1–6. IEEE.

Arauzo-Azofra, A., Aznarte, J. L., & Benítez, J. M. (2011). Empirical study of feature selection methods based on individual feature evaluation for classification problems. *Expert Systems with Applications*, *38*(7), 8170 – 8177.

Arauzo-Azofra, A., Benitez, J. M., & Castro, J. L. (2008). Consistency measures for feature selection. *Journal of Intelligent Information Systems*, *30*(3), 273–292.

Bai, J., Fan, Z., Zhang, L., Xu, X., & Zhang, Z. (2017). Classification of methicillin-resistant and methicillin-susceptible staphylococcus aureus using an improved genetic algorithm for feature selection based on mass spectra. In *Proceedings of the 9th International Conference on Bioinformatics and Biomedical Technology*, pp. 57–63. ACM.

Bian, J., Peng, X.-g., Wang, Y., & Zhang, H. (2016). An efficient cost-sensitive feature selection using chaos genetic algorithm for class imbalance problem. *Mathematical Problems in Engineering*, *2016*.

Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, *97*(1-2), 245–271.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (2017). *Classification and regression trees*. Routledge.

Chen, L., Wang, S., Zhang, Y.-H., Wei, L., Xu, X., Huang, T., & Cai, Y.-D. (2018). Prediction of nitrated tyrosine residues in protein sequences by extreme learning machine and feature selection methods. *Combinatorial chemistry & high throughput screening*, *21*(6), 393–402.

Chen, Z., Lin, T., Tang, N., & Xia, X. (2016). A parallel genetic algorithm based feature selection and parameter optimization for support vector machine. *Scientific Programming*, *2016*.

Das, A. K., Pati, S. K., & Ghosh, A. (2020). Relevant feature selection and ensemble classifier design using bi-objective genetic algorithm. *Knowledge and Information Systems*, *62*(2), 423–455.

Demšar, J., Curk, T., Erjavec, A., Črt Gorup, Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., & Zupan, B. (2013). Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, *14*, 2349–2353.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern Classification (2Nd Edition)*. Wiley-Interscience.

Ge, J., Zhang, X., Liu, G., & Sun, Y. (2019). A novel feature selection algorithm based on artificial bee colony algorithm and genetic algorithm. In *2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*, pp. 131–135. IEEE.

Gharaee, H., & Hosseinvand, H. (2016). A new feature selection ids based on genetic algorithm and svm. In *Telecommunications (IST), 2016 8th International Symposium on*, pp. 139–144. IEEE.

Gupta, A., & Purohit, A. (2017). Rgap: A rough set, genetic algorithm and particle swarm optimization based feature selection approach. *International Journal of Computer Applications*, *161*(6).

Homsapaya, K., & Sornil, O. (2017). Improving floating search feature selection using genetic algorithm. *Journal of ICT Research and Applications*, *11*(3), 299–317.

Homsapaya, K., & Sornil, O. (2018). Modified floating search feature selection based on genetic algorithm. In *MATEC Web of Conferences*, Vol. 164, p. 01023. EDP Sciences.

Hsu, H.-H., Hsieh, C.-W., & Lu, M.-D. (2011). Hybrid feature selection by combining filters and wrappers. *Expert Systems with Applications*, *38*(7), 8144–8150.

Jadhav, S., He, H., & Jenkins, K. (2018). Information gain directed genetic algorithm wrapper feature selection for credit rating. *Applied Soft Computing*, *69*, 541–553.

Jiang, S., Chin, K.-S., Wang, L., Qu, G., & Tsui, K. L. (2017). Modified genetic algorithm-based feature selection combined with pre-trained deep neural network for demand forecasting in outpatient department. *Expert Systems with Applications*, *82*, 216–230.

Jing, S.-Y. (2014). A hybrid genetic algorithm for feature subset selection in rough set theory. *Soft Computing*, *18*(7), 1373–1382.

Jović, A., Brkić, K., & Bogunović, N. (2015). A review of feature selection methods with applications. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1200–1205. IEEE.

Kallel, L., & Schoenauer, M. (1997). Alternative random initialization in genetic algorithms. In *Proceedings of the 7 th International Conference on Genetic Algorithms*, pp. 268–275. Morgan Kaufmann.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *SCIENCE*, *220*(4598), 671–680.

Kohavi, R. (1994). Feature subset selection as search with probabilistic estimates. In *AAAI fall symposium on relevance*, Vol. 224.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*, 273–324.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of relief. In *Proceedings of the European Conference on Machine Learning on Machine Learning*, ECML-94, pp. 171–182, Secaucus, NJ, USA. Springer-Verlag New York, Inc.

Labani, M., Moradi, P., Ahmadizar, F., & Jalili, M. (2018). A novel multivariate filter method for feature selection in text classification problems. *Engineering Applications of Artificial Intelligence*, *70*, 25–37.

Lin, X., Wang, X., Xiao, N., Huang, X., & Wang, J. (2015). A feature selection method based on feature grouping and genetic algorithm. In *International Conference on Intelligent Science and Big Data Engineering*, pp. 150–158. Springer.

Liu, H., & Setiono, R. (1996). Feature selection and classification - a probabilistic wrapper approach. In *in Proceedings of the 9th International Conference on Industrial and Engineering Applications of AI and ES*, pp. 419–424.

Ma, B., & Xia, Y. (2017a). A genetic algorithm based feature selection for binary phenotype prediction using structural brain magnetic resonance imaging. In *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pp. 172–177. IEEE.

Ma, B., & Xia, Y. (2017b). A tribe competition-based genetic algorithm for feature selection in pattern classification. *Applied Soft Computing*, *58*, 328–338.

Mao, Y., Zhang, Z., & Fan, D. (2016). Hybrid feature selection based on improved genetic algorithm for stock prediction. In *Digital Home (ICDH), 2016 6th International Conference on*, pp. 215–220. IEEE.

Nandi, G. (2011). An enhanced approach to las vegas filter (lvf) feature selection algorithm. In *2011 2nd National Conference on Emerging Trends and Applications in Computer Science*, pp. 1–3. IEEE.

Newman, C. B. D., & Merz, C. (1998). UCI repository of machine learning databases..

Nguyen, T. T., Liew, A. W.-C., Tran, M. T., Pham, X. C., & Nguyen, M. P. (2014). A novel genetic algorithm approach for simultaneous feature and classifier selection in multi classifier system. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pp. 1698–1705. IEEE.

Oreski, S., & Oreski, G. (2014). Genetic algorithm-based heuristic for feature selection in credit risk assessment. *Expert systems with applications*, *41*(4), 2052–2064.

Park, J., Park, M.-W., Kim, D.-W., & Lee, J. (2020). Multi-population genetic algorithm for multilabel feature selection based on label complementary communication. *Entropy*, *22*(8), 876.

Priya, R. D., & Sivaraj, R. (2017). Dynamic genetic algorithm-based feature selection and incomplete value imputation for microarray classification. *Current Science*, *112*(1), 126–131.

Pudil, P., Novovičová, J., & Kittler, J. (1994). Floating search methods in feature selection. *Pattern recognition letters*, *15*(11), 1119–1125.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rahmadani, S., Dongoran, A., Zarlis, M., et al. (2018). Comparison of naive bayes and decision tree on feature selection using genetic algorithm for classification problem. In *Journal of Physics: Conference Series*, Vol. 978, p. 012087. IOP Publishing.

Raman, M. G., Somu, N., Kirthivasan, K., Liscano, R., & Sriram, V. S. (2017). An efficient intrusion detection system based on hypergraph-genetic algorithm for parameter optimization and feature selection in support vector machine. *Knowledge-Based Systems*, *134*, 1–12.

Sahu, B., Dehuri, S., & Jagadev, A. K. (2017). An ensemble model using genetic algorithm for feature selection and rule mining using apriori and fp-growth from cancer microarray data. *International Journal of Applied Engineering Research*, *12*(10), 2391–2408.

Schiffner, J., Bischl, B., Lang, M., Richter, J., Jones, Z. M., Probst, P., Pfisterer, F., Gallo, M., Kirchhoff, D., Kühn, T., et al. (2016). mlr tutorial. *arXiv preprint*, *Online*.

Shukla, A. K., Singh, P., & Vardhan, M. (2019). A new hybrid feature subset selection framework based on binary genetic algorithm and information theory. *International Journal of Computational Intelligence and Applications*, *18*(03), 1950020.

Singh, S., & Selvakumar, S. (2015). A hybrid feature subset selection by combining filters and genetic algorithm. In *Computing, Communication & Automation (ICCCA), 2015 International Conference on*, pp. 283–289. IEEE.

Szenkovits, A., Meszlényi, R., Buza, K., Gaskó, N., Lung, R. I., & Suciu, M. (2017). Feature selection with a genetic algorithm for classification of brain imaging data. *Advances in Feature Selection for Data and Pattern Recognition*, *138*, 185.

Tahir, M., Tubaishat, A., Al-Obeidat, F., Shah, B., Halim, Z., & Waqas, M. (2020). A novel binary chaotic genetic algorithm for feature selection and its utility in affective computing and healthcare. *Neural Computing and Applications*, *Online*, 1–22.

Tan, F., Fu, X., Zhang, Y., & Bourgeois, A. G. (2008). A genetic algorithm-based method for feature subset selection. *Soft Computing*, *12*(2), 111–120.

Tang, J., Alelyani, S., & Liu, H. (2014). Feature Selection for Classification: A Review. In *Data Classification*, Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, pp. 37–64. Chapman and Hall/CRC.

Thangavel, K., & Pethalakshmi, A. (2009). Dimensionality reduction based on rough set theory: A review. *Applied Soft Computing*, *9*(1), 1 – 12.

Vergara, J. R., & Estévez, P. A. (2014). A review of feature selection methods based on mutual information. *Neural computing and applications*, *24*(1), 175–186.

Vieira, A. C., Garcia, G., Pabón, R. E., Cota, L. P., de Souza, P., Ueyama, J., & Pessin, G. (2020). Improving flood forecasting through feature selection by a genetic algorithm–experiments based on real data from an amazon rainforest river. *Earth Science Informatics*, *Online*, 1–14.

Wang, D., Zhang, Z., Bai, R., & Mao, Y. (2018). A hybrid system with filter approach and multiple population genetic algorithm for feature selection in credit scoring. *Journal of Computational and Applied Mathematics*, *329*, 307–321.