A HYBRID FEATURE SUBSET SELECTION METHOD BASED ON GRASP AND RELIEF

by
BUSE NUR KARATEPE

Submitted to the Sabanci Universtiy Faculty of Natural Science
in partial fulfilment of
the requirements for the degree of Master of Science

Sabancı University
June 2023

A HYBRID FEATURE SUBSET SELECTION METHOD BASED ON GRASP AND RELIEF

Approved by:

Assoc. Prof. Kemal Kılıç .....
(Thesis Supervisor)

Prof. Abdullah Daşcı .....

Asst. Prof. Mustafa Hekimoğlu .....

Date of Approval: June 25 , 2023

ABSTRACT

A HYBRID FEATURE SUBSET SELECTION METHOD BASED ON GRASP AND RELIEF

BUSE NUR KARATEPE

DATA SCIENCE M.S. THESIS, JUNE 2023

Thesis Supervisor: Prof. Kemal Kılıç

Keywords: Feature subset selection, Relief, GRASP, filters, wrappers

The abundance of complex, high-dimensional data in various fields has amplified the necessity for effective feature selection strategies. This thesis proposes an innovative hybrid Feature Subset Selection technique to identify and retain the most valuable features, thereby enhancing model interpretability and combating the curse of dimensionality. Our method uniquely merges the computational efficiency of filter techniques and the precision of wrapper methods, explicitly combining the metaheuristic algorithm Greedy Randomized Adaptive Search Procedure (GRASP) and a reputable feature filtering algorithm Relief. The process initiates with a comprehensive exploration of feature combinations, subsequently applying various filter techniques, amongst which Relief exhibited superior performance. Additionally, Relief was integrated into the construction and improvement stages of GRASP.

Experiments are conducted by checking the average 30 runs of K-Nearest Neighbors scores and time. The results underscore the potency of the hybrid approach, significantly improving model performance and demonstrating the potential of integrating filter and wrapper methods for efficient feature selection in high-dimensional datasets. This contribution allows us to maximize the accuracy of the machine-learning model while minimizing the time dedicated to feature selection.

ÖZET

"GRASP VE RELIEF TEMELLI BIR HIBRIT ÖZELLIK ALT KÜMESI SEÇIMI"

BUSE NUR KARATEPE

VERI BILIMI YÜKSEK LİSANS TEZİ, HAZIRAN 2023

Tez Danışmanı: Kemal Kılıç

Anahtar Kelimeler: Özellik alt kümesi seçimi, Relief, GRASP, filtreleme, wrapper

Çeşitli alanlardaki karmaşık, yüksek boyutlu verilerin bolluğu, etkili özellik seçme stratejilerine olan ihtiyacı artırmıştır. Bu tez, en değerli özellikleri belirleyip korumak, böylece modelin yorumlanabilirliğini geliştirmek ve boyutluluk lanetiyle mücadele etmek için yenilikçi bir hibrit Özellik Alt Kümesi Seçimi tekniği önermektedir. Yöntemimiz, metasezgisel algoritma Açgözlü Rastgele Uyarlanabilir Arama Prosedürü (GRASP) ile saygın bir özellik filtreleme algoritması Relief'i açıkça birleştirerek, filtre tekniklerinin hesaplama verimliliğini ve sarma yöntemlerinin kesinliğini benzersiz bir şekilde birleştirir. Süreç, özellik kombinasyonlarının kapsamlı bir şekilde araştırılmasıyla başlar ve ardından Relief'in üstün performans sergilediği çeşitli filtre teknikleri uygulanır. Ek olarak Relief, GRASP'ın inşaat ve iyileştirme aşamalarına entegre edildi.

Deneyler, K-En Yakın Komşular puanlarının ve süresinin ortalama 30 kez kontrol edilmesiyle gerçekleştirilir. Sonuçlar, model performansını önemli ölçüde artıran ve yüksek boyutlu veri kümelerinde verimli özellik seçimi için filtre ve sarmalayıcı yöntemleri entegre etme potansiyelini ortaya koyan hibrit yaklaşımın gücünün altını çiziyor. Bu katkı, özellik seçimine ayrılan süreyi en aza indirirken makine öğrenimi modelinin doğruluğunu en üst düzeye çıkarmamıza olanak tanır.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Assoc. Prof. Kemal Kılıç. His unwavering belief in my potential, combined with his insightful critiques, patience, and expert guidance, have been instrumental throughout this research journey.

Heartfelt appreciation goes to my committee members, Prof. Abdullah Daşcı and Asst. Prof. Mustafa Hekimoglu, for their invaluable feedback and thought-provoking inquiries. Their contributions have significantly enriched this work.

My journey was made more enjoyable and enlightening through the camaraderie and intellectual discussions with my fellow researchers and friends. Special recognition goes to my roommate, Buse Çarık, and my friends Atacan Tütüncüoglu and Murat Sarp Koçak, for standing by me during challenging times and for their unfaltering support and encouragement.

To my family, your unwavering faith, patience, and love have been my rock. This achievement would not have been possible without your steadfast encouragement.

The Data Science master's program at Sabanci University has equipped me with the requisite skills to conduct this research effectively. Serving as a Teaching Assistant in the Computational Thinking and Data Analysis courses not only honed my teaching skills but also molded my thought process, which significantly contributed to the development of this thesis.

This journey has also been a personal test, completed while relocating to the Netherlands, balancing regular work, and dedicating every spare moment and weekend to research. It was unquestionably an immense challenge, and I am proud of the determination and hard work I invested. Therefore, I extend thanks to myself for persevering, staying focused, and persisting during difficult times. This accomplishment is a testament to my commitment and dedication, for which I am profoundly grateful.

Buse Nur Karatepe
Haziran 2023

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

PCA: Principal Component Analysis

GRASP: Greedy Randomized Adaptive Search Procedure

DT: Decision Tree

KNN: K-Nearest Neighbors

LR: Logistic Regression

NB: Naive Bayes

GA: Genetic Algorithm

PSO: Particle Swarm Optimization

ACO: Ant Colony Optimization

SA: Simulated Annealing

GP: Genetic Programming

TS: Tabu Search

RFE: Recursive Feature Elimination

PCS: Promising Candidate Solutions

RFS: Remaining Features Set

TempPCS: Temporary Promising Candidate Solutions

ConstPCS: Constructed Promising Candidate Solutions

RCL: Restricted Candidate List

UpdatedPCS: Updated Promising Candidate Solutions

ImprRCL: Improved Restricted Candidate List

1. INTRODUCTION

1.1 Motivation and Objective

In today's digital age, the accessibility of data mining has greatly improved thanks to advancements in technology. As a result, we can now work with larger and more complex datasets at a reduced cost. This trend is particularly evident in fields such as bioinformatics and text mining, where expansive datasets with numerous dimensions have become increasingly common. Given the vast amount of information contained within these datasets, it has become essential to develop effective feature selection techniques. Feature selection plays a critical role in identifying the most relevant and informative features, enabling us to extract valuable insights from the data. The exploration of feature selection techniques has been ongoing since the 1990s in the fields of statistics and machine learning, and it remains a fundamental topic in the realm of data mining.

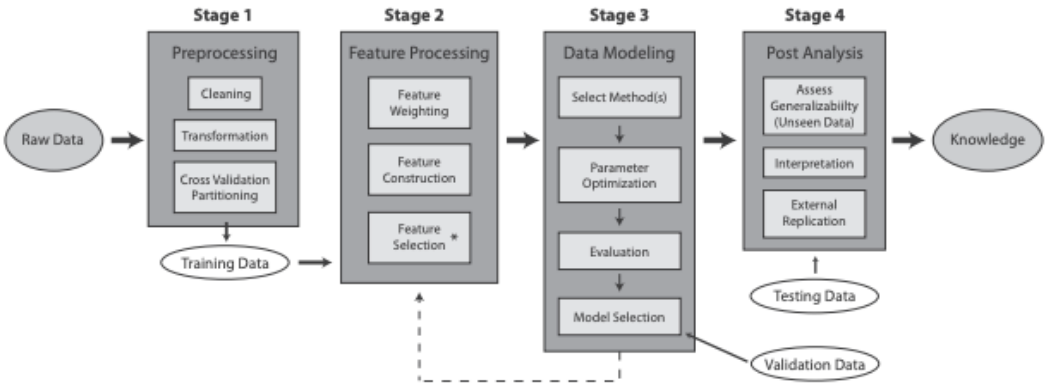


Figure 1.1 Pipeline for data science projects. Source (Urbanowicz et al., 2018)

The primary motivation of this study is dual-pronged: reducing the time dedicated to feature selection and enhancing the resultant score of the machine learning model.

Numerous methods exist for feature selection, which will be elaborated in the succeeding chapters. The significance of feature selection lies in the fact that not all features in a dataset contribute equally to the outcome. Indeed, the removal of redundant features can yield advantages in cost and predictive performance.

1.1.1 Do we need all the features?

To explain the motivation behind our study, we conducted an experiment that engaged all possible feature combinations. Given a dataset with n features, there can be $(2^n) - 1$ unique combination generated, excluding only the null subset. To see whether we need all the feature combinations, we used a classification task and its performance.

To illustrate, we refer to the Iris dataset, renowned in machine learning for its simplicity and clear features. This dataset comprises four features: sepal length, sepal width, petal length, and petal width, which are used to classify three species of Iris flowers - Setosa, Versicolour, and Virginica (Iris, 2007). We utilized a cleaned version of this dataset, which served as the basis for our experimental procedures Table 1.1.

In the context of our experiment, [0] denotes sepal length, [1] corresponds to sepal width, [2] represents petal length, and [3] indicates petal width. The aim of our analysis was to predict the Iris flower species based on these features.

Table 1.1 Iris dataset (Iris, 2007)

sepal length	**sepal width**	**petal length**	**petal width**	**target**	**classification**
-0.90	1.02	-1.34	-1.32	0	setosa
-1.14	-0.13	-1.34	-1.32	0	setosa
-1.39	0.33	-1.40	-1.32	0	setosa
1.40	0.33	0.54	0.26	1	versicolor
0.67	0.33	0.42	0.40	1	versicolor
1.28	0.10	0.65	0.40	1	versicolor
0.55	0.56	1.27	1.71	2	virginica
-0.05	-0.82	0.76	0.92	2	virginica
1.52	-0.13	1.22	1.19	2	virginica

Each combination was tested across four different machine learning models, namely: Decision Tree (DT), K-Nearest Neighbors (KNN), Logistic Regression (LR), and Naive Bayes (NB). A 10-fold cross-validation method was employed for each model to ensure reliable performance estimation. The scores for each combination and

model are documented in Table 1.2. This approach enabled us to identify the relative importance of each feature while simultaneously selecting the optimal feature subset and machine learning model.

Feature Combination	DT	KNN	LG	NB
[0]	0.6933	0.6533	0.7533	0.7267
[1]	0.4800	0.5200	0.5667	0.5667
[2]	0.9333	0.9533	0.9533	0.9533
[3]	0.9533	0.9600	0.9600	0.9533
[0, 1]	0.6467	0.7600	0.8067	0.7933
[0, 2]	0.9400	0.9400	0.9600	0.9133
[0, 3]	0.9267	0.9600	0.9467	0.9533
[1, 2]	0.9200	0.9533	0.9533	0.9133
[1, 3]	0.9267	0.9533	0.9533	0.9400
[2, 3]	0.9467	0.9667	0.9600	0.9600
[0, 1, 2]	0.9467	0.9467	0.9533	0.8800
[0, 1, 3]	0.9467	0.9533	0.9600	0.9333
[0, 2, 3]	0.9533	0.9600	0.9667	0.9600
[1, 2, 3]	**0.9600**	**0.9667**	**0.9733**	**0.9667**
[0, 1, 2, 3]	**0.9600**	**0.9667**	**0.9733**	0.9533

Table 1.2 Classification results for all feature combinations

Table 1.2 details the performance of the four models (DT, KNN, LR, and NB) when trained on all possible combinations of these four features. It's worth noting that the highest accuracy is achieved with the feature subset [1, 2, 3], i.e., sepal width, petal length, and petal width. This implies that these three features are the most critical for accurately predicting the Iris flower species. Furthermore, these results indicate that employing fewer features can achieve the same or sometimes better predictive performance. More features do not necessarily yield better results. Considering the computational power required, achieving high performance with fewer features is often a more desirable outcome. This experiment underscores the significance of effective feature selection in machine learning.

1.1.2 Why Feature Selection, not Construction?

Feature weighting, feature construction, and feature selection are three distinct techniques employed in the field of machine learning to enhance the effectiveness of models by focusing on relevant features and improving their representation.

Feature weighting involves assigning different weights to individual features within a

dataset. It aims to emphasize the importance of certain features while downplaying the impact of others. This technique can be based on statistical measures, domain knowledge, or machine learning algorithms. By assigning appropriate weights to features, the model can give more importance to the informative ones, potentially improving its performance.

Feature construction, on the other hand, entails creating new features from existing ones. It involves transforming and combining existing features to generate more informative representations of the data. Techniques such as Principal Component Analysis (PCA) or polynomial feature construction are employed to derive new features that capture additional information and patterns. Feature construction aims to enhance the model's ability to understand complex relationships within the data.

Feature selection involves identifying the most relevant subset of features from a larger feature space. It aims to eliminate redundant or irrelevant features that may hinder model performance. Feature selection can be performed using various methods, including statistical tests, wrapper methods, and embedded methods. By selecting a subset of features that have the most significant impact on the target variable, feature selection reduces dimensionality, improves interpretability, and may enhance model performance by mitigating the effects of overfitting.

In summary, feature weighting assigns weights to existing features to emphasize their importance, feature construction creates new features by transforming and combining existing ones to provide more informative representations, and feature selection focuses on identifying the most relevant subset of features from a larger feature space. These techniques play vital roles in feature engineering, allowing machine learning models to leverage the most informative and discriminative features, resulting in improved performance and interpretability. In this thesis, our primary objective is to minimize the number of features in data science problems. The presence of irrelevant or redundant features can lead to the curse of dimensionality, negatively impacting the model's performance. As highlighted by (Blumer, Ehrenfeucht, Haussler & others, 1987), Occam's Razor principle guides us towards simpler models to prevent overfitting and enhance generalization. Hence, our focus is on feature selection as the chosen approach. We opt for feature selection over feature weighting and feature construction for several reasons. Firstly, feature weighting alone does not reduce the number of features; it only assigns different weights to them. Secondly, feature construction, although capable of creating new features, runs the risk of diluting the original feature meanings by generating new representations. By prioritizing feature selection, we can effectively strike a balance between preserving

the meaningful information contained within the original features and reducing the complexity of the model. This approach enables us to construct more efficient and interpretable machine learning models that generalize well to unseen data.

1.1.3 Three Feature Subset Selection techniques

Fature subset selection techniques primarily fall into three categories: filter methods, wrapper methods, and embedded methods. Filter methods involve ranking features based on specific criteria before selecting the top ones. These criteria might include the correlation with the output variable, mutual information, or chi-square statistics. Filter methods are simple and fast, as they evaluate each feature independently, thus offering scalability for high-dimensional datasets. However, this independence assumption might lead to sub-optimal feature subsets since potential feature interactions still need to be considered. On the other hand, wrapper methods adopt a search strategy for potential feature combinations and evaluate their worth using a specific machine learning algorithm's performance. Unlike filter methods, these methods account for feature interactions and dependencies, providing better performance in many cases. Common strategies for wrapper methods include forward selection, backward elimination, and recursive feature elimination. Nevertheless, wrapper methods are computationally expensive and prone to overfitting, particularly with high-dimensional datasets due to their exhaustive search nature. The third approach, embedded methods, seeks to combine the best of both worlds. These methods perform feature selection in the process of model training and are usually specific to certain learning algorithms. They are less computationally expensive than wrapper methods while still capturing feature interactions as part of their design.

1.2 Problem Definition and Summary of the Results

The problem at hand involves the reduction of feature dimensionality while maintaining high accuracy in predictive modeling tasks. Feature selection is a vital step in machine learning that helps in mitigating the curse of dimensionality, improving model interpretability, and reducing training times.

In this study, we have implemented a novel hybrid approach that combines the strengths of the Greedy Randomized Adaptive Search Procedure (GRASP) and Relief-based feature selection methodologies. Our aim is to address the problem of minimizing features while retaining high prediction accuracy.

From our results, we find that this GRASP and Relief-based hybrid model offers improved accuracy compared to models that do not employ any feature selection. The reduced feature set not only streamlined our model but also contributed to a relatively better performance.

However, it's essential to highlight that, despite its improvements, our hybrid approach has not outperformed certain state-of-the-art models such as ITMO and GA. This observation invites further exploration and fine-tuning of our approach to extract its full potential.

1.3 Contributions

This paper proposes A Hybrid Feature Subset Selection (FSS) based on the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic and filter-method approach Relief. The GRASP algorithm has been widely recognized for its ability to address high-dimensional datasets rapidly. It is a randomized algorithm which is introduced by integrating it with Relief, a renowned feature selection algorithm. We aim to optimize feature selection and deliver better results, especially for high dimensional datasets. Our objective is to leverage the computational efficiency of filter methods to quickly eliminate the most irrelevant features, subsequently utilizing the precision of wrapper methods to optimize the feature subset. This integrated approach aims to enhance the feature selection process, enabling the model to more accurately capture the underlying function of the target concept.

Our study commenced with an exhaustive search of all feature combinations, evaluating the accuracy results for each. Given the (2n̂)-1 potential feature combinations (where n is the number of features), our goal was to identify the subset that yielded the highest accuracy. Once this initial stage was complete, we shifted our focus to various filtering techniques. To establish a comprehensive comparison basis for the Relief filter, we experimented with widely-used FSS methods such as Variance Threshold, Selecting K Best, and Principal Component Analysis (PCA).

Upon selecting the filter, we proceeded to experimental trials. Each feature subset was input into the model both with and without the application of the filter, allowing us to compare results across these different conditions. This experiment was conducted for four different models, namely Decision Tree (DT), K-Nearest Neighbors (KNN), Logistic Regression (LR), and Naive Bayes (NB). To ensure robustness in our findings, we ran each model 30 times and computed the average accuracy result. The analysis of the accuracy scores revealed that using the Relief filter led to improved performance, validating our hybrid FSS approach. This demonstrates the potential of integrating filter and wrapper methods for effective feature selection in high-dimensional datasets.

1.4 Structure of the Thesis

In Section 2, a thorough literature review is presented, encompassing the relevant research on Feature Subset Selection, Relief, and GRASP. This literature review serves as a foundation for the current study, offering a contextual framework and identifying the research gaps that our work aims to address.

Section 3 transitions into the Methodology section, where we introduce our novel model, titled "A Hybrid Feature Subset Selection Based on GRASP and Relief." This section delves into the detailed explanation of the model's components and explains how it synergizes the GRASP and Relief methodologies. By integrating these approaches, our model aims to bridge the identified research gaps and provide an innovative solution. Each step of the algorithm design and implementation is thoroughly explained to provide a clear understanding of the model's functioning.

Moving forward to Section 4, we designed the empirical component of our research. This section begins by introducing the data used in the study, providing insights into its nature and characteristics. We then proceed to describe the experiments conducted to finalize our algorithm, which involved exploring various filtering techniques to optimize the performance of our model.

Lastly, in Section 5, we present our key findings and conduct a comprehensive comparison between our proposed model, "A Hybrid Feature Subset Selection Based on GRASP and Relief" and other state-of-the-art FSS models. We highlight the strengths and limitations of our model and discuss avenues for future work and further research.

2. LITERATURE REVIEW

This chapter delves into the extensive body of literature that underpins our study. We start by exploring the concept of Feature Selection, its objectives, formulations, and primary methodologies in Section 2.1. Next, we focus on the Relief method discussed in Section 2.2, emphasizing its importance as a multivariate filter method that effectively deals with feature interactions.. Finally, Section 2.3 examines the Greedy Randomized Adaptive Search Procedure (GRASP), a promising metaheuristic technique that can potentially alleviate the computational strain of feature selection while ensuring reliable performance.

2.1 Prior Literature Review on Feature Subset Selection

2.1.1 Understanding Feature Subset Selection

Feature Subset Selection (FSS) is a vital practice in machine learning and data mining, featuring prominently in the literature due to its potential to enhance model performance, simplify model interpretation, and conserve computational resources by discarding irrelevant features (Guyon, Weston, Barnhill & Vapnik, 2003). Despite its significance, FSS presents a challenging NP-hard problem, necessitating the employment of heuristic and metaheuristic techniques to combat its computational complexity (Kohavi & John, 1997).

In a mathematical context, suppose we have a dataset D consisting of n features $f_1, f_2, \ldots, f_n$. The aim of Feature Subset Selection (FSS) is to identify a subset S from these n features. This subset is selected such that it maximizes the predictive

performance of a model trained on S. More formally, if we denote a quality criterion by Q, the FSS problem can be defined as:

$$S^* = \text{argmax}\, Q(S), \quad \text{where } S \subseteq \{f_1, f_2, \ldots, f_n\}$$

The solution to this optimization problem typically falls into three categories: filter methods, wrapper methods, and embedded methods.

1. **Filter methods** evaluate the relevance of features based on their inherent characteristics. They use metrics such as mutual information, chi-square, or correlation coefficient. While they are computationally efficient, these methods might not capture the interdependencies between features. However, certain multivariate filter methods, like Relief, are exceptions. These methods consider multiple features simultaneously, thereby accounting for their interdependence. Urbanowicz et al. (2018) highlight the efficiency of Relief, particularly its manageable time complexity and robust software capabilities Urbanowicz et al. (2018).

2. **Wrapper methods** assess the relevance of feature subsets based on the predictive performance of a specific machine learning algorithm. They evaluate each subset by training a model on it and using the model's performance as an indicator of the subset's quality. Because they account for the bias of the algorithm and interactions between features, wrapper methods tend to outperform filter methods. However, this superior performance comes at a cost: higher computational intensity. Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS) are widely used examples of wrapper methods Kittler (1978). However, these methods can be computationally demanding, particularly for high-dimensional datasets. To mitigate this, we propose using a randomized algorithm: the Greedy Randomized Adaptive Search Procedure (GRASP). As outlined by Feo and Resende (1995), GRASP is a metaheuristic approach designed for efficient exploration of the search space. It strikes a balance between exploration and exploitation, making it a promising candidate for high-dimensional feature selection Feo & Resende (1995).

3. **Embedded methods** integrate feature selection within the model training process. They take advantage of the learning algorithm's ability to evaluate feature importance, thereby balancing performance and computational efficiency. Examples of embedded methods include decision trees and regularized regression models Breiman (2001)."

2.1.2 Use of Metaheuristics in Feature Subset Selection

Metaheuristics, derived from the combination of "heuristic" and "meta," are generalized approaches that can be applied to various decision-making problems. (Gandomi & Yang, 2013) discussed the general principles and benefits of using metaheuristics, such as their ability to address complex optimization problems, navigate large solution spaces, and strike a balance between exploration and exploitation.

Metaheuristics serve as umbrella heuristics capable of solving a wide range of problems directly or with minor modifications. There are two strategies for finding solutions: intensification, which focuses on promising areas based on past searches, and diversification, which explores the entire space to avoid settling on sub-optimal solutions too quickly. A well-designed metaheuristic strikes a balance between these aspects to deliver good feasible solutions. Some notable examples of metaheuristic methods used in the past include: Genetic Algorithm (GA) (Goldberg, 1989), Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995), Ant Colony Optimization (ACO) (Dorigo & Di Caro, 1999), Simulated Annealing (SA) (Kirkpatrick, Gelatt Jr & Vecchi, 1983), Genetic Programming (GP) (Koza, 1992), and Tabu Search (TS) (Glover, 1989).

In the context of feature selection, applying metaheuristics offers an efficient and effective means of navigating the vast solution space. Although they may not guarantee to find the optimal solution due to the NP-hard nature of the feature selection problem, metaheuristics excel at discovering near-optimal or high-performing feature subsets. They leverage evolutionary computation, swarm intelligence, and simulated annealing techniques to explore different feature combinations and evaluate their performance. This enables the identification of feature subsets that maximize predictive accuracy, minimize complexity, and enhance the generalization capabilities of machine learning models. Additionally, metaheuristic algorithms exhibit adaptability to diverse dataset types, can handle high-dimensional feature spaces, and overcome the limitations of traditional feature selection methods. The utilization of metaheuristics in feature subset selection has demonstrated promising results in various domains, including bioinformatics, image processing, and financial prediction. By offering a powerful and flexible approach to address complex optimization problems, metaheuristics provide valuable insights for improving model performance and decision-making in feature selection tasks. It is within this context that our current study integrates the GRASP metaheuristic and Relief filter method to optimize feature selection.

2.2 Filtering with Relief

2.2.1 What is Relief?

The Relief algorithm is a feature selection method used to identify the most relevant features in a dataset. It was originally introduced by (Kira & Rendell, 1992) and has since been widely adopted in various fields, including bioinformatics and machine learning.

The primary objective of the Relief algorithm is to capture feature interactions, which are often crucial in understanding the underlying patterns in the data. Unlike traditional univariate filter methods that consider each feature independently, Relief evaluates the relevance of features by taking into account the interactions between features and their impact on the target variable.

The algorithm operates by assigning a weight or score to each feature based on its ability to discriminate between instances of different classes. It does this by comparing the feature values of the nearest neighboring instances, where the nearest neighbors are determined based on their Euclidean distance. The algorithm considers both the features' similarity to instances of the same class (relevance) and their difference from instances of different classes (redundancy).

By iteratively updating the feature weights for each instance in the dataset, Relief captures the relationships between features and their relevance to the target variable. The final feature scores obtained from the Relief algorithm can be used to rank the features and select the most informative subset for further analysis or model construction.

2.2.2 Application of Relief in Literature

There are several notable applications of the Relief algorithm in the literature. Urbanowicz et al. (2018) provide an introduction and comprehensive review of Relief-based feature selection, highlighting its applications in various domains including bioinformatics, genetics, and medicine (Urbanowicz et al., 2018). Wang et al. (2006)

Algorithm 1 Relief Algorithm

```
procedure RELIEF(dataset, m)
    Initialize a weight vector W with zeros
    for i = 1 to m do
        Randomly select an instance R from the dataset
        Find nearest hit H and nearest miss M for R
        for each feature F in R do
            W[F] = W[F] − diff(F, R, H)/m + diff(F, R, M)/m
        end for
    end for
    Return W
end procedure
```

apply the Relief algorithm for gene selection from microarray data to improve cancer classification accuracy (Wang, Li, Wang & Pan, 2006). Fang et al. (2018) utilize the Relief algorithm in combination with rank correlation coefficient to identify growth-driving genes in biological systems (Fang, Guo, Zhao, Guo & Zhao, 2018). Deekshit et al. (2019) employ the Relief algorithm as a feature selection technique to develop an intelligent diagnostic tool for the identification of bacterial pathogens (Deekshit, Kumar, Sadasivuni & Sadasivuni, 2019). These studies demonstrate the versatility and effectiveness of the Relief algorithm in various domains, showcasing its potential for feature selection and data analysis.

2.2.3 Different Relief Models

Relief, ReliefF, and RReliefF are three algorithms used for feature selection, a critical step in the process of building an accurate and efficient machine learning model. Let's delve a bit deeper into each of these:

1) Relief: The Relief algorithm is a feature selection technique designed to work with small-to-moderate-sized datasets. Introduced by (Kira & Rendell, 1992), Relief is capable of detecting features that are conditionally dependent, meaning it can identify features that are useful for predicting a certain class only in conjunction with another feature. The algorithm works by repeatedly selecting an instance randomly and then finding its nearest neighbors from the same and different classes. It then updates the weights of the features depending on how their values differentiate the selected instance from its neighbors. However, Relief tends to be sensitive to noise and works best with two-class problems.

2) ReliefF: Recognizing the limitations of the Relief algorithm, especially its inability to deal efficiently with multi-class and noisy datasets, (Kononenko, 1994) developed an extended version known as ReliefF. ReliefF considers multiple nearest neighbors instead of just one, which makes it more robust against noise. It also handles incomplete and missing data, as well as multi-class problems more efficiently.

3) RReliefF: This is a variant of ReliefF specifically designed to handle regression problems, where the output variable is a real number instead of a class label. It estimates feature relevancies for continuous-valued outputs and works similarly to ReliefF, but the difference in their methods lies in the way the output difference is calculated in the weight updating process. RReliefF uses the difference of the target variable of the selected instance and its nearest neighbors, instead of their class labels.

2.2.4 Application of Relief in this paper

The Relief algorithm is highly advantageous due to its capability to handle both categorical and continuous features, its robustness against noisy data, and its efficiency in managing high-dimensional datasets. However, it is important to consider certain factors that can influence the algorithm's performance, such as the selection of the number of neighbors and the specific distance metric employed.

In the broader context, the Relief algorithm has proven to be a valuable approach to feature selection, particularly in capturing feature interactions, and has been widely applied across various domains to enhance the performance and interpretability of machine learning models. In this paper, we utilize the Relief algorithm as our initial feature selection method and combine it with the GRASP metaheuristic.

Furthermore, this paper conducts a comprehensive analysis by examining different Relief models and comparing them with other filtering techniques such as Variance Threshold, Select K Best, and PCA. Through our investigation, we observed the effectiveness of the Relief algorithm and subsequently chose to proceed with it as our primary filtering method.

2.3 GRASP

2.3.1 What is GRASP?

Search algorithms, particularly those used for combinatorial optimization problems, are designed to explore the solution space efficiently to find optimal or near-optimal solutions. A prime example of this class of algorithms is the Greedy Randomized Adaptive Search Procedure (GRASP), introduced by Feo and Resende (Feo & Resende, 1989). GRASP is a metaheuristic algorithm that combines the principles of greedy algorithms and randomness to explore various regions of the solution space and escape local optima, which can be beneficial when tackling complex optimization problems.

GRASP operates in a two-stage iterative process involving a construction phase and a local search phase.

1) Construction Phase: The construction phase aims to generate a feasible solution. Unlike traditional greedy algorithms that might deterministically select the best candidate at each step, GRASP introduces an element of randomness in this phase. It builds the solution iteratively by adding an element selected randomly from a Restricted Candidate List (RCL). The RCL comprises high-quality elements, which are determined based on some greedy function criteria, but not necessarily the best ones. This random selection ensures a diversified exploration of the solution space.

The pseudocode for the construction phase is as follows:

Algorithm 2 Construction Phase of GRASP Algorithm

```
1: procedure ConstructionPhase
2:     S ← empty
3:     while S is not a complete solution do
4:         Construct a Restricted Candidate List (RCL)
5:         Choose element e from RCL at random
6:         Add e to solution S
7:     end while
8: end procedure
```

2) Improvement Phase: Once a feasible solution is constructed, the local search phase commences. This phase seeks to enhance the solution generated in the construction phase by exploring its vicinity. A neighborhood of the current solution, which contains solutions similar to it, is examined, and the algorithm moves to a better solution if one is available. This step is iteratively repeated until no further improvement can be made, yielding a locally optimal solution.

The pseudocode for the local search phase is as follows:

Algorithm 3 Improvement Phase of GRASP Algorithm

```
1: procedure IMPROVEMENT PHASE
2:     while there exists a solution S' in the neighborhood of S that is better than S do
3:         Move from solution S to the better solution S'
4:     end while
5: end procedure
```

These two phases are iteratively repeated, with each repetition producing a locally optimal solution. After numerous iterations, the best solution across all iterations is chosen as the final output of the GRASP algorithm.

2.3.2 Application of GRASP in Literature

The Greedy Randomized Adaptive Search Procedure (GRASP) has found wide-ranging applications in the literature due to its ability to provide high-quality solutions for combinatorial optimization problems. In the realm of operations research, GRASP has been applied to various scheduling problems, such as job-shop, flow-shop, and project scheduling (Festa & Pardalos, 2002). It has also been successfully utilized for problems related to logistics, including vehicle routing, traveling salesman, and facility location problems (Resende & Ribeiro, 2003). In bioinformatics, GRASP has been adopted for sequence alignment, gene recognition, and DNA fragment assembly (Ribeiro & Resende, 2007). Moreover, its applications extend to telecommunication design, such as network design and frequency assignment (Resende & Werneck, 2004). As a metaheuristic, GRASP offers a flexible and robust framework for problem-solving, which has led to its widespread usage across a multitude of fields.

2.3.3 Application of GRASP in FSS

The application of the GRASP algorithm for feature subset selection has been studied extensively in the literature. Esseghir and Idoumghar (Esseghir & Idoumghar, 2010) proposed a GRASP-based algorithm for feature selection in high-dimensional data. They used GRASP's robust and adaptive properties to iteratively build solutions and improve upon them, focusing on both convergence speed and quality of solutions. In each iteration, a Restricted Candidate List (RCL) was dynamically generated based on feature relevancy measures, ensuring the inclusion of the most relevant features and enhancing the algorithm's search space exploration. The local search phase aimed to refine the solutions, using the '2-opt' local search method to swap features in and out of the current solution based on their contribution to the classifier's accuracy.

Abreu et al. (Abreu, de Carvalho & Lorena, 2011) applied a similar methodology that combined GRASP with a hybrid filter-wrapper approach for feature subset selection in high-dimensional datasets. Their work focused on optimizing the selected features' relevancy and redundancy. In the construction phase, features were added to the solution based on the correlation between the features and the target class (relevance) and the correlation between the features themselves (redundancy). By controlling the size of the RCL, they could manage the trade-off between exploration and exploitation in the search process. On the other hand, their local search phase aimed to fine-tune the solution, using a forward-backward heuristic to add or remove features while maintaining the solution's quality.

Another exciting work by Smiti and Elouedi (Smiti & Elouedi, 2015) presented a GRASP-based approach for feature selection in clustering high-dimensional categorical data. They combined GRASP with a k-medoids-based local search for efficient exploration and exploitation of the search space. The construction phase involved creating an initial solution by selecting subsets of features that minimize the within-cluster dissimilarity. The local search phase, in turn, was responsible for refining the initial solution, aiming to improve the clustering quality by exploring the neighborhood of the current solution. Their approach demonstrated the potential of GRASP for feature selection in unsupervised learning contexts.

2.3.4 Application of GRASP in this paper

The application of the Greedy Randomized Adaptive Search Procedure (GRASP) in this study is focused on feature selection in high-dimensional data. The approach integrates a novel filtering method during the construction phase and applies a one-in, one-out neighborhood definition during the improvement phase.

In the construction phase, the process commences with the initialization of dataframes to store the results of each run and the accuracy of various feature subsets. Subsets of features are created from the input data, and for each, the Relief algorithm is implemented. The filtering process introduced in this stage terminates the loop if the minimum weight of any feature falls below t_1 threshold.

Following the filtering, the selected features are sorted by weight and used to train a K-Nearest Neighbors (KNN) model. The choice of KNN is justified by its lack of inherent feature weighting, making it an ideal classifier for this purpose. The mean cross-validation score is computed and stored for each run.

Instead of selecting the absolute best feature, the model randomly selects a top feature from the dataframe after iterating through all subsets of a specific size. This approach aids in maintaining diversity. To ensure pursuit of promising feature subsets, a check is implemented that discards the current subset if its KNN accuracy score is less than the previous subset's score. The output of the construction phase is a list of feature subsets, along with their corresponding details and performance metrics.

In the improvement phase, the feature set selected during the construction phase and its corresponding KNN accuracy are identified. The selected and non-selected features are ascertained by comparing with the full list of features. For each selected feature, a list of 'neighbors' is generated. A neighbor is defined as a feature obtained by replacing a selected feature with a non-selected one. Each 'neighbor' is trained on a KNN model, and the mean cross-validation score is calculated and stored. The scores are then sorted in descending order of accuracy. This phase aims to improve the solution found in the construction phase by exploring the neighborhood of the current solution.

In summary, the GRASP-based approach proposed in this study presents an approximate solution to the feature selection problem, offering the potential to identify a global optimum through randomization and local search strategies. More details can be found in the Section 4: Methodology.

3. METHODOLOGY

3.1 Algorithm Design

This section presents our innovative adaptation of the Greedy Randomized Adaptive Search Procedure (GRASP) to tackle feature selection in high-dimensional data. Our method uniquely integrates a filtering approach using Relief during the construction phase, and employs the one-in, one-out neighborhood definition during the improvement phase.

3.1.1 Construction Phase

The Construction Phase is the foundation of the algorithm, primarily responsible for building a promising set of candidate solutions called Restricted Candidate List (RCL). The process begins by constructing K RCLs, where K represents the total iterations planned for this phase. Each iteration starts by initializing a solution set, which is an empty set for the RCL. Simultaneously, the K-nearest neighbors (KNN) accuracy for this empty set is also initialized to zero, providing a performance baseline for feature addition.

The algorithm then determines the remaining features set (RFS) by subtracting the currently selected features in the RCL from the total set of features. For each feature in the RFS, a temporary promising candidate solution (TempRCL) is generated. This TempRCL is a potential solution where the given feature is added to the current RCL.

The Relief Score for each TempRCL is then computed. This score is a measure

indicating the worth or importance of each feature within the TempRCL. TempRCLs with an average Relief Score above a predefined threshold (t1) are retained in a separate set, referred to as the "SET of TempRCLs".

The algorithm then evaluates the performance of each TempRCL within this set. This is accomplished by calculating the KNN accuracy for each TempRCL. The TempRCLs are then ranked according to their KNN accuracies, and the top 't2' are selected for further consideration.

The algorithm randomly selects one TempRCL from this top set, and this selected TempRCL is used to update the current RCL. This loop of operations continues until adding a new feature to the RCL no longer improves the KNN accuracy. The algorithm ensures the most beneficial features are included in the RCL by this approach.

After completing each iteration, the RCL becomes a part of the RCL set only if its KNN accuracy is above a predefined threshold (t3). This phase's output is a Restricted Candidate List (RCL) containing a collection of promising constructed candidate solutions (ConstRCLs).

3.1.2 Improvement Phase

The Improvement Phase's primary role is to refine and optimize the RCLs obtained from the previous Construction Phase. This phase starts by taking the RCL as input. The algorithm then iterates through each ConstRCL within the RCL.

For each ConstRCL, the algorithm generates a neighboring solution. This is done by making minor modifications to the ConstRCL, such as the addition of one unselected feature and removal of a selected feature. The algorithm then calculates the KNN accuracy of this neighboring solution to evaluate its performance.

If the KNN accuracy of this neighboring solution exceeds the current maximum observed KNN accuracy, this neighbor is considered a better solution. As such, the algorithm updates the current RCL (now referred to as Updated Promising Candidate Solutions or UpdatedRCL), and the maximum KNN accuracy is also updated to reflect this improvement. This iterative process continues until the KNN accuracy no longer improves, which suggests that a local optimum solution is achieved.

The end output of the Improvement Phase and the overall algorithm is a set of refined

Algorithm 4 Construction Phase

Construct K RCL
for i=1 to K **do**
    Initialize NewFeatureIsBeneficial = True;
        s=1
    while NewFeatureIsBeneficial==True **do**
        Initialize $RCL_{(i)}$(s) = {};
            $KNN_{acc}RCL_{(i)}$(s) = 0
            RFS = All Features - $RCL_{(i)}$(s)
        for j=1 to C ← Cardinality of RFS **do**
            Generate a TempRCL = $RCL_{(i)}$(s+j)
            Calculate Relief Score of TempRCL
            if Relief Score among the features > threshold t_1 **then**
                Add TempRCL to "SET of TempRCLs"
            end if
        end for
        for each TempRCL in "SET of TempRCLs" **do**
            Calculate KNN accuracy
        end for
        Rank and choose the top *t2* TempRCL based on KNN accuracy
        Choose randomly one of the *t2* "SET of TempRCLs", say $TempRCL^*$
        s=s+1
        $RCL_{(i)}$(s) = $TempRCL^*$
        if K $KNN_{acc}RCL_{(i)}$(s) < $KNN_{acc}RCL_{(i)}$(s-1) **then**
            NewFeatureIsBeneficial = False
        end if
    end while
    Add $RCL_{(i)}$(s-1) to RCL Set
end for
Output ConstRCL
if KNN_{acc}ConstRCL > threshold t_3 **then**
    Add ConstRCL to "SET of ConstRCL"
end if
Output the "SET of ConstRCL"

and optimized RCLs. These optimized RCLs represent the subsets of features that are most beneficial in enhancing the predictive model's performance.

In conclusion, our adaptation of GRASP offers an approximate solution to the feature selection problem. It selects a local optimum at each stage and potentially finds a global optimum through its randomization and local search strategies.

Algorithm 5 Improvement Phase

```
Get "SET of ConstRCL" from the Construction Phase
for each i in "SET of ConstRCL" do
    Initialize UpdatedPCS = CFS_(i)
                max KNNacc = 0
                LocalOptimaIsEnsured = False
    while LocalOptimaIsEnsured == False do
        Create one neighbor of CFS_(i)
        Check KNN accuracy of that neighbor
        if KNN accuracy of the neighbor > Max KNNacc then
            UpdatedRCL = neighbor
            Update Max KNNacc with KNN accuracy of the neighbor
        else
            LocalOptimaIsEnsured = True
        end if
    end while
end for
Output "ImprRCL" and choose the top t3 based on KNN accuracy
```

3.1.3 Iterations

Table 3.1 GRASP - Iteration 1

selected columns	Relief weights	weights Avg
[0]	[0.0836]	0.0836
[1]	[0.0391]	0.0391
[2]	[0.4456]	0.4456
[3]	[0.5292]	0.5292
[4]	[0.0093]	0.0093
[5]	[0.0106]	0.0106

Table 3.2 GRASP - Iteration 1 - Filtering

selected columns	Relief weights	weights Avg
[0]	[0.0836]	0.0836
[1]	[0.0391]	0.0391
[2]	[0.4456]	0.4456
[3]	[0.5292]	0.5292
~~[4]~~	~~[0.0093]~~	~~0.0093~~
[5]	[0.0106]	0.0106

Table 3.3 GRASP - Iteration 1 - Choosing

selected columns	Relief weights	weights Avg	KNN Acc	RANK
[0]	**[0.0836]**	**0.0836**	**0.7000**	**3**
[1]	[0.0391]	0.0391	0.4917	4
[2]	[0.4456]	0.4456	0.9333	2
[3]	[0.5292]	0.5292	0.9583	1
[5]	[0.0106]	0.0106	0.3833	5

Table 3.4 GRASP - Iteration 2

selected columns	Relief weights	weights Avg	KNN Acc	RANK
[0, 1]	**[0.3986, 0.2279]**	**0.3133**	**0.7917**	**3**
[0, 2]	[0.1309, 0.5843]	0.3576	0.9333	1
[0, 3]	[0.2520, 0.5884]	0.4202	0.9250	2
[0, 4]	[0.1515, 0.1272]	0.1393	0.6750	4
[0, 5]	[0.1563, 0.1259]	0.1411	0.6750	4

Table 3.5 GRASP - Iteration 3

selected columns	Relief weights	weights Avg	KNN Acc	RANK
[0, 1, 2]	**[0.3611, 0.4167, 0.7355]**	**0.5044**	**0.8333**	**4**
[0, 2, 3]	[0.2557, 0.5877, 0.6200]	0.4891	0.9417	1
[0, 2, 4]	[0.2326, 0.6377, 0.1400]	0.3369	0.8917	2
[0, 2, 5]	[0.1805, 0.6474, 0.1700]	0.3332	0.8833	3

Table 3.6 GRASP - Iteration 4

selected columns	Relief weights	weights Avg	KNN Acc	RANK
[0, 1, 2, 3]	[0.2617, 0.7343, 0.5930, ...]	0.6009	0.9333	1
[0, 1, 2, 4]	[0.3041, 0.6422, 0.7410, ...]	0.4741	0.8500	3
[0, 1, 2, 5]	**[0.3114, 0.5962, 0.7200, ...]**	**0.4617**	**0.8583**	**2**

Table 3.7 GRASP - Final Output

Number of Features	Selected Columns	Time	KNN Acc
4	[0, 1, 2, 5]	16.451s	0.8583

Table 3.8 GRASP - All Iterations

selected columns	**Relief weights**	**weights Avg**	**KNN Acc**	**RANK**
[0]	[0.0836]	0.0836	0.7000	3
[1]	[0.0391]	0.0391	0.4917	4
[2]	[0.4456]	0.4456	0.9333	2
[3]	[0.5292]	0.5292	0.9583	1
[5]	[0.0106]	0.0106	0.3833	5
[0, 1]	[0.3986, 0.2279]	0.3133	0.7917	3
[0, 2]	[0.1309, 0.5843]	0.3576	0.9333	1
[0, 3]	[0.2520, 0.5884]	0.4202	0.9250	2
[0, 4]	[0.1515, 0.1272]	0.1393	0.6750	4
[0, 5]	[0.1563, 0.1259]	0.1411	0.6750	4
[0, 1, 2]	[0.3611, 0.4167, ...]	0.5044	0.8333	4
[0, 2, 3]	[0.2557, 0.5877, ...]	0.4891	0.9417	1
[0, 2, 4]	[0.2326, 0.6377, ...]	0.3370	0.8917	2
[0, 2, 5]	[0.1805, 0.6474, ...]	0.3332	0.8833	3
[0, 1, 2, 3]	[0.2617, 0.7343, ...]	0.6009	0.9333	1
[0, 1, 2, 4]	[0.3041, 0.6422, ...]	0.4741	0.8500	3
[0, 1, 2, 5]	[0.3114, 0.5962, ...]	0.4617	0.8583	2

4. EMPIRICAL SETTING AND DATA

This chapter encompasses the details of the experiments executed throughout this research and provides a comprehensive view of the results obtained.

4.1 Dataset

We delve into the specifics of the dataset archive in this section, followed by an explanation of the process of dataset creation.

4.1.1 Dataset Archive

Table 4.1 Datasets

Dataset Name	Source	Number of feature	Instance
Iris	(Iris, 2007)	4	150
Kidney	(Kidney, 2015)	12	158
Wine	(Wine, 2007)	13	178
Breast Cancer	(Cancer, 2017)	60	208
Connectionist Bench	(Connectionist, 2014)	60	207

4.1.2 Dataset creation and preprocessing

This part is about how we prepared our dataset for testing our model. We wanted to see if our model could identify and eliminate features that aren't contributing any

useful information. Our goal was to see if these obviously non-informative features would be recognized and excluded by our model, demonstrating its effectiveness in feature selection. As it mentioned in the first chapter, we conducted all the experiments on Iris dataset first (Iris, 2007). We load the cleaned version of this data as described in the preceding section and shown in Table 4.2. This dataset forms the backbone of our subsequent experimental procedures.

Table 4.2 Raw data

sepal length	**sepal width**	**petal length**	**petal width**	**target**
5.1	3.5	1.4	0.2	0
4.9	3.0	1.4	0.2	0
4.7	3.2	1.3	0.2	0
4.6	3.1	1.5	0.2	0
5.0	3.6	1.4	0.2	0

To ensure a well-conditioned dataset, the next step involves scaling the data. We achieve this by applying a StandardScaler to our dataset, a process that standardizes features by removing the mean and scaling to unit variance. The outcome of this scaling operation can be seen in Table 4.3. This pre-processing step is crucial when dealing with features that have varying scales as it brings them to a common scale without distorting the differences in the range of values or losing information.

Table 4.3 Scaled data

sepal length	**sepal width**	**petal length**	**petal width**	**target**
-0.9007	1.0190	-1.3402	-1.3154	0
-1.1430	-0.1320	-1.3402	-1.3154	0
-1.3854	0.3284	-1.3971	-1.3154	0
-1.5065	0.0982	-1.2834	-1.3154	0
-1.0218	1.2492	-1.3402	-1.3154	0

Further, we enrich our dataset by introducing new "dummy" features with uniformly distributed values between 0 and 1. The number of these added features equals half the original feature set. In addition, we incorporate two more columns named "Zeros(0)" and "Ones(1)" to observe their impact on the output and test the model's ability to identify these dummy features. The data with added features is shown in Table 4.4.

Upon completion of these preprocessing steps, the dataset is finally ready for subsequent modeling stages, thus providing a robust foundation for our analysis.

4.2 Experiment on Relief

sepal length	sepal width	petal length	petal width	Dummy1	Dummy2	Z	0	target
-0.9007	1.0190	-1.3402	-1.3154	0.7664	0.4988	0	1	0
-1.1430	-0.1320	-1.3402	-1.3154	0.3282	-0.2579	0	1	0
-1.3854	0.3284	-1.3971	-1.3154	-0.3758	-0.6868	0	1	0
-1.5065	0.0982	-1.2834	-1.3154	0.0403	0.5182	0	1	0
-1.0218	1.2492	-1.3402	-1.3154	-0.5211	0.8204	0	1	0

Table 4.4 Data with added dummy features

In this experiment, our aim was to assess the performance of the Relief feature selection method in our machine learning pipeline. To establish a baseline, we initially created a function, withoutFilter, which performed model training without any feature selection. We executed this function for a total of 30 runs and captured the performance metrics for Decision Trees (DT), K-Nearest Neighbors (KNN), Logistic Regression (LG), and Naive Bayes (NB) models, as well as the run time for each iteration. Without any feature selection, the average performance across 30 runs indicated that the models performed reasonably well. The result was as follows in Table 4.5:

DT	KNN	LG	NB	SelectedCols	NumOfCols	Excluded	time
0.96	0.95	0.96	0.95	[0, 1, 2, 3]	4	[]	0.279s
0.95	0.95	0.96	0.95	[0, 1, 2, 3]	4	[]	0.266s
0.95	0.95	0.96	0.95	[0, 1, 2, 3]	4	[]	0.272s
...	...	...	...	...	...	...	...
0.95	0.95	0.96	0.95	[0, 1, 2, 3]	4	[]	0.195s

Table 4.5 Results withoutFilter

Our primary evaluation metric was the effectiveness of the Relief feature selection. To assess this, we implemented Relief to score our features. The Relief algorithm, renowned for its robustness in feature selection (Urbanowicz et al., 2018), measures the value of features by assessing how their values differentiate between instances in close proximity within the feature space. On the selection of a random instance, Relief identifies the closest instance from the same class, as well as the nearest from the opposite class. The scores of features are then updated by incrementing (or decrementing) them based on the variance in feature values within the same (or opposite) class instances. This score is a reflection of how well the feature distinguishes between classes.

Upon application of Relief, we preserved 80% of the highest scoring features and eliminated the remaining 20%. In the specific case of the Iris dataset, the number of features was reduced from four to three. We carried out another set of 30 model training runs using the selected features, while capturing the same performance

metrics as in the previous trials. The results are presented in Table 4.6:

DT	**KNN**	**LG**	**NB**	**SelectedCols**	**NumOfCols**	**Excluded**	**time**
0.96	0.96	0.96	0.96	[1, 2, 3]	3	[0]	0.213s
0.96	0.96	0.96	0.96	[1, 2, 3]	3	[0]	0.728s
0.96	0.96	0.96	0.96	[1, 2, 3]	3	[0]	0.858s
...	...	...	...	...	...	...	...
0.96	0.96	0.96	0.96	[1, 2, 3]	3	[0]	0.269s

Table 4.6 Results with Relief

The comparative analysis between models trained with all features and those trained with features selected by Relief, as depicted in Table 4.7, reveals a subtle improvement in model performance across DT, KNN, LG, and NB subsequent to feature selection. Additionally, there was negligible variation in the run times, indicating that the application of Relief for feature selection could potentially enhance model performance without imposing substantial computational costs.

	DT	KNN	LG	NB	time
Without Filter (Avg)	0.9576	0.9533	**0.9600**	0.9533	0.2656s
Relief (3 features, Avg)	**0.9600**	**0.9600**	**0.9600**	**0.9667**	0.2694s

Table 4.7 Comparison of Classification Results: No Filter vs Relief Filter

These results demonstrate the effectiveness of Relief in improving model performance. By removing irrelevant or redundant features, Relief makes the models more efficient and less prone to overfitting, thereby enhancing their predictive performance.

4.3 Experiment on different filtering methods

In this section, we employed various widely used filtering methods for feature selection and compared their performance with that of the Relief method and shown in Table 4.8.

This table presents the average performance metrics for Decision Trees (DT), k-Nearest Neighbors (KNN), Logistic Regression (LG), Naive Bayes (NB), and the average run time for each method over 30 runs. Five feature selection methods were

	DT	KNN	LG	NB	time
withoutFilterAvg	0.9558	0.9533	0.9600	0.9533	0.2557s
varianceThresholdAvg	0.9556	0.9533	0.9600	0.9533	0.2995s
SelectKBestAvg	0.9533	0.9600	0.9600	0.9600	0.3094s
PCAAvg	0.9298	0.9533	0.9600	0.9333	0.2540s
Relief(3-features)Avg	**0.9600**	**0.9600**	**0.9600**	**0.9667**	0.3107s

Table 4.8 Comparison of Performance for Different Feature Filtering Methods

compared: no filtering (withoutFilter), Variance Threshold (varianceThreshold), SelectKBest, Principal Component Analysis (PCA), and Relief.

It is observed that the Relief method, which retains only the top three features, outperforms or matches the other methods in all models. Specifically, it achieved the highest average scores in DT, KNN, LG, and NB models. Moreover, it matched the best score in Logistic Regression models. All this is achieved while keeping the run time competitive, suggesting that the Relief feature selection method can effectively improve model performance without significantly impacting the computational cost, proving to be an effective feature selection method.

4.4 Experiment on Relief with dummy features

The next phase of our experimental process entailed assessing Relief's performance on a dummy dataset. The primary focus of this assessment was the accuracy scores and computational time necessary for model execution, varying in accordance with the `numOfColumns` parameter of Relief.

numOfColumns	selectedColumns	DT	KNN	LG	NB	time
4	[0, 1, 2, 3]	0.93	0.93	0.95	0.95	0.207s

Table 4.9 Baseline Model Performance with Selected Columns

Upon establishing the baseline results, we augmented the dataset with additional dummy features. Table 4.10 presents the performance outcomes of the models utilizing this augmented dataset:

Table 4.10 demonstrates that the Relief feature selection algorithm effectively pinpoints the relevant features as [0, 1, 2, 3]. Our next step is to advance our experiments on the augmented dataset, assigning the `numberOfColumns` parameter a value

numOfColumns	**selectedColumns**	**DT**	**KNN**	**LG**	**NB**	**time**
1	[3]	0.95	0.96	0.96	0.96	0.136s
2	[2, 3]	0.94	0.96	0.95	0.95	0.490s
3	[1, 2, 3]	0.93	0.95	0.94	0.95	0.653s
4	**[0, 1, 2, 3]**	**0.93**	**0.93**	**0.95**	**0.95**	**0.161s**
5	[0, 1, 5, 2, 3]	0.93	0.96	0.95	0.96	0.330s
6	[4, 3, 2, 5, 0, 1]	0.92	0.93	0.94	0.96	0.246s

Table 4.10 Performance with Dummy Dataset

that corresponds to the raw data column count, which, in this context, is 4.

5. RESULTS

In this section, we present the results of our experiments, which highlight the performance of various feature selection models. We begin by providing a comprehensive overview of the comparison models used in our study, including the Genetic Algorithms (GA) and ITMO FS. These models encompass a range of feature selection techniques and libraries that have been widely employed in the field of machine learning.

5.1 Summary of what we did

To gain insights into the impact of feature selection, we first analyze the results obtained from a dummy dataset based on the Iris dataset. This involves examining the features selected by each model and their corresponding accuracy. Through this analysis, we aim to understand the importance of feature optimization and how different models prioritize and select relevant features.

As we move forward, our focus shifts to the evaluation of accuracy and time scores across five different datasets. We start by exploring the performance of the '*Baseline*' model. In this context, '*Baseline'* represents a scenario where no feature elimination is performed during the model building process.

Following that, we investigate the performance of our hybrid approach, termed our Model. This approach combines the GRASP algorithm with Relief filtering, creating an innovative feature selection method. We also examine the results obtained using the Genetic Algorithms (GA) library, which is a popular feature selection library based on evolutionary algorithms.

Lastly, we present the outcomes achieved using the ITMO_FS library. Developed by ITMO University, the ITMO_FS library offers a wide range of advanced feature

selection algorithms.

By comparing the performance of these models, our goal is to assess the effectiveness of different feature selection techniques in terms of accuracy and computational time. This analysis is expected to provide valuable insights into the utility of each model and its suitability for different datasets and classification tasks.

In conclusion, our results section offers a comprehensive evaluation of various feature selection models. This allows us to identify the most effective approaches for feature optimization. We believe that this knowledge will contribute to advancements in the field of feature selection and aid in improving the accuracy and efficiency of machine learning models.

5.2 Brief Information About Comparison Models

We now turn our attention to an examination of the comparison models employed in this study:

5.2.1 ITMO FS

In our feature selection methodology, a key component is the ITMO_FS library (FS, 2020). The library, developed by ITMO University, is a rich source of feature selection algorithms that provides researchers and practitioners with versatile and cutting-edge tools for data analysis.

From this diverse array of feature selection methods available in the ITMO_FS library, we chose to experiment with the Model-based Optimization Selection (MOS) algorithm. Our choice was motivated by our interest in exploring the potential of embedded feature selection methods in the context of our study. Embedded methods, such as MOS, simultaneously conduct feature selection and model learning, a quality that often results in a good balance of performance and computational efficiency.

The MOS algorithm embodies the principle of model-based optimization. It takes advantage of the predictive performance of a machine learning model to guide the

feature selection process. By evaluating the performance of different subsets of features iteratively and gauging their impact on the model's accuracy, MOS is capable of identifying an optimal subset of features. This approach facilitates selection of features that not only enhance model accuracy but also reduce computational complexity by minimizing the feature set.

The effectiveness and working principle of MOS are detailed in several scholarly articles. The paper by (Fu, Wu, Zong & Yi, 2020) elucidates the optimization process of MOS and provides empirical evidence demonstrating its effectiveness. Additionally, (Pilnenskiy & Smetannikov, 2020) conducts a comparative analysis of various feature selection algorithms, including MOS, illuminating their applicability in diverse data analysis tasks. These references serve to underline the reasons for our selection of the ITMO_FS library and MOS for our study.

In summary, by integrating the ITMO_FS library, particularly the MOS algorithm, into our feature selection approach, we aim to incorporate an advanced, efficient method that optimizes classification accuracy and reduces feature dimensionality. This method will be compared with our hybrid model, *GRASP & Relief*, to provide valuable insights into the relative effectiveness of different feature selection techniques in the context of our classification tasks.

5.2.2 Genetic Algoritms

Genetic Algorithms (GA) are a popular search procedure that can be utilized for feature selection in machine learning tasks. In the context of feature selection, GA operates based on the principles of natural selection and genetics, mimicking the evolutionary process observed in biological systems. The goal of GA is to identify a subset of features that optimizes the performance of a machine learning model.

One specific implementation of GA for feature selection is the "sklearn-genetic" library Genetic (2016). This library provides a Python implementation of Genetic Algorithms and integrates seamlessly with the scikit-learn ecosystem, making it convenient to incorporate GA-based feature selection into existing machine learning pipelines.

The advantage of using Genetic Algorithms for feature selection lies in their ability to explore a vast search space efficiently and effectively. GA employs mechanisms such as crossover, mutation, and selection to generate new feature combinations

iteratively. This iterative process gradually converges towards a subset of features that maximizes the performance of the machine learning model.

However, it is important to note that Genetic Algorithms can be computationally expensive due to the need to evaluate the fitness of multiple feature subsets. Therefore, the time complexity of GA-based feature selection should be considered in relation to the dataset size and available computational resources.

By leveraging the capabilities of Genetic Algorithms, researchers can effectively explore and identify relevant features that contribute to the predictive performance of machine learning models.

5.3 Performance of our model

In Table 5.2, we present the results obtained with the dummy-added dataset. Initially, the Iris dataset had 4 features, but we expanded it to 8 features. We then applied four different Feature Selection and Search (FSS) models and recorded the selected features for each model. The 'ALL' model represents a scenario where no feature elimination was performed, resulting in the fastest execution time. On the other hand, the our model and 'Genetic Algorithm' models exhibited promising performance in identifying the most relevant features and achieving high accuracy. These models selected only one feature ('3') and produced an accuracy of 0.9583. The 'ITMO' model also demonstrated effective elimination, resulting in a subset of 5 selected features ('0, 1, 2, 3, 7'). Although it had a slightly lower accuracy of 0.9333, it still showcased efficient feature selection.

Model	**Num of Features**	**Selected Features**	**Accuracy**	**Time(s)**
Baseline	8	[0, 1, 2, 3, 4, 5, 6, 7]	0.9333	0.041
Our Model	1	[3]	**0.9583**	25.204
Genetic Algorithm	1	[3]	**0.9583**	2.604
ITMO	5	[0, 1, 2, 3, 7]	0.9333	18.325

Table 5.1 Model performance on Iris dataset

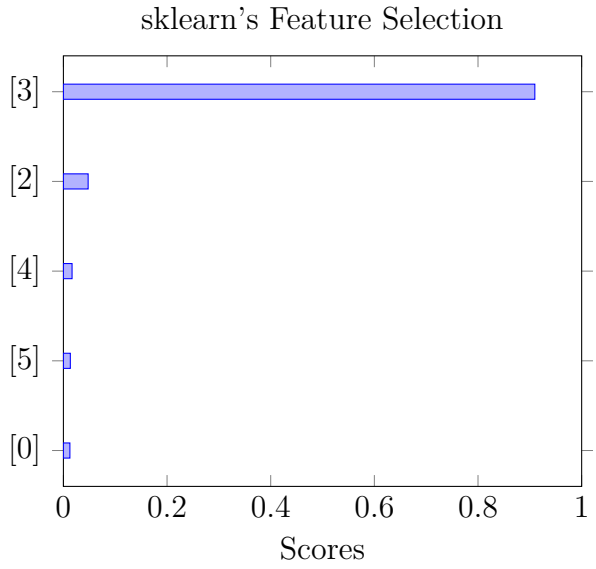
5.3.1 Comparison with Sklearn's Feature Selection Method

To place our findings in a wider perspective, we drew a comparison between the Relief feature selection method and a popular alternative, Recursive Feature Elimination (RFE), offered by the sklearn library (Iris, 2007).

RFE is an influential feature selection method that operates by fitting a model and progressively eliminating the least consequential features. It continues this iterative process until the desired number of features remains. By ranking features based on their *coef* or *feature importances* attributes, RFE seeks to minimize dependencies and collinearity in the model.

In the context of our experiment, we employed the *DecisionTreeClassifier* as our estimator, owing to its ability to rank features based on their *feature importances*. The findings from this comparative study are depicted in the subsequent figure:

As it can be ssen from the graph, the most importtant feature for this dataset is [3]. And both GRASP & Releif and GA found that and eliminate all the other ones.



5.4 Comparison on other datasets

		Accuracy				Time(s)			
Dataset	**Features**	**Baseline**	**OurModel**	**GA**	**ITMO**	**Baseline**	**OurModel**	**GA**	**ITMO**
Iris	4	0.9333	**0.9583**	**0.9583**	0.9333	0.033	3.214	1.983	3.952
Kidney	12	0.8103	**0.8417**	0.8340	0.8103	0.036	10.881	5.282	9.066
Wine	13	0.9581	**0.9933**	**0.9933**	0.9586	0.031	30.119	7.472	5.231
Breast Cancer	30	0.9649	0.9670	**0.9826**	0.9648	0.068	113.214	14.692	80.261
Connectionist	60	0.8081	0.8673	**0.8978**	0.8456	0.042	243.608	18.505	120.067

The table presents the performance metrics of different datasets with varying numbers of features, evaluated using four feature selection models: Baseline, GRASP, GA (Genetic Algorithm), and ITMO. The metrics considered are accuracy and execution time.

For the Iris dataset, which initially had 4 features, all four models achieved high accuracy, with the GRASP and GA models obtaining the highest accuracy of 0.9583. The execution time was relatively low for all models, with the *Baseline* model being the fastest at 0.033 seconds.

The Kidney dataset, also with 12 features, demonstrated lower overall accuracy compared to the previous datasets. The GRASP model achieved the highest accuracy of 0.8417, while the other models had lower accuracies. The execution times were presented in seconds, with the *Baseline* model being the fastest at 0.036 seconds.

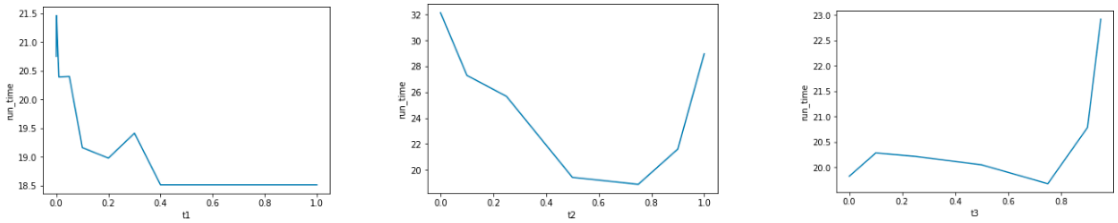
The Wine dataset, with 12 features, exhibited similar trends. The GRASP and GA models achieved the highest accuracy of 0.9933, while the *Baseline* model showed a slightly lower accuracy of 0.9581. The execution time varied, with the GRASP model taking the longest at 30.119 seconds.

The Breast Cancer dataset, with 30 features, showcased high accuracies for all models, with the GA model achieving the highest accuracy of 0.9826.

Finally, the Connectionist dataset, with 60 features, displayed varied accuracies for the different models. The GRASP model achieved the highest accuracy of 0.8978, while the other models had slightly lower accuracies. The execution times were presented in seconds, with the GRASP model taking the longest at execution time at 243.608 seconds.

Overall, the table provides a comparison of accuracy and execution time for different feature selection models across multiple datasets. It allows for an evaluation of the effectiveness of each model in terms of accuracy and the trade-off with execution time. Even though the Genetic Algorithm mostly outperforms thanks to its fine-tuned features, our model also had some promising results.

5.5 Hyperparameter Tuning



(a) t1 vs time (b) t2 vs time (c) t3 vs time

Figure 5.1 Comparison of t1, t2, and t3 parameters vs time

Hyperparameter tuning is a crucial aspect of developing robust machine learning models. This process involves optimizing the parameters of the model that are not learned from the data, in order to improve its performance. For our Relief-based Greedy Randomized Adaptive Search Procedure model, the hyperparameters of interest include ‘t1‘, ‘t2‘, ‘t3‘, and ‘K‘.

The ‘t1‘ parameter, which controls relief score, was tuned over a range from 0 to 1, in increments of 0.1. After an exhaustive search over this range, we found that a ‘t1‘ value of 0.01 yielded the most optimal model performance.

The ‘t2‘ parameter, which controls KNN acuracy score in construction phase, was similarly explored, over a range from 0 to 1, at specified intervals (0.1, 0.25, 0.5, 0.75, 0.9, and 1). Through this process, we discovered that the model performance was highest when ‘t2‘ was set to 0.5.

We followed a similar approach for the ‘t3‘ parameter, which controls KNN acuracy score in improvement phase, exploring the same range and intervals as ‘t2‘. Like ‘t2‘, we found the optimal value for ‘t3‘ to be 0.5.

Finally, we tuned the ‘K‘ parameter, which controls number of iterations. We investigated a series of possible values: 1, 5, 10, 15, 30, 50, and 100. Upon conducting this search, we found that ‘K = 10‘ yielded the best results for our model.

By methodically exploring these hyperparameters, we were able to fine-tune our Relief based GRASP model to achieve an optimal balance of performance and computational efficiency. Future work may further refine these parameters, or investigate additional hyperparameters to enhance the model’s performance.

5.6 Comparative Analysis with other GRASP Models

To ascertain the effectiveness of our proposed Relief based GRASP model, we engaged in a comparative analysis against other notable GRASP-based models. In particular, we focused on FCGRASP and SAGRASP models.

The FCGRASP model was introduced by Bermejo, Gámez, and Puerta in 2011 (Bermejo, Gámez & Puerta, 2011). It's a unique search method for feature selection in high-dimensional data, that substantially reduces the number of wrapper evaluations. This is achieved by alternating between filter and wrapper evaluations during the feature selection process. It's a two-step iterative algorithm that constructs a solution and then improves it in each iteration. In the construction phase, a lighter filter measure is used for the evaluation process while a more costly wrapper is utilized for improving the constructed solution.

Meanwhile, SAGRASP is a model developed by Moshki et al. (Moshki, Kabiri & Mohebalhojeh, 2015). It incorporates a modified version of the Simulated Annealing (SA) algorithm into the GRASP framework. This helps in escaping local minimums, further reducing wrapper evaluations per iteration, and offering implicit control over the trade-off between solution length and precision.

Our comparative analysis offers revealing insights into the efficacy and efficiency of the considered models. In terms of precision, our model registers a score of 0.883. Interestingly, this precision outperforms the SAGRASP model, which stands at 0.873. However, when juxtaposed with the FCGRASP model that boasts a precision of 0.895, our model doesn't reach the benchmark. This margin, while not vast, draws attention to potential areas for enhancement and fine-tuning in our model.

Table 5.2 Comparative Analysis on Connectionist data

	Precision			**Time(s)**		
Dataset	**Our Model**	**SAGRASP**	**FCGRASP**	**Our Model**	**SAGRASP**	**FCGRASP**
Connectionist	0.883	0.873	0.895	248	149	2054

Shifting the lens to time efficiency, our model's performance demonstrates both strengths and areas of improvement. It completes its execution in 248 seconds. While this is notably faster than the FCGRASP model, which takes a substantial 2054 seconds, it still lags behind the swift execution of the SAGRASP model at just 149 seconds. This differential in execution time underscores the balance that needs

to be struck between precision and computational efficiency, especially in real-world applications where time is often of the essence.

The above findings underscore the multifaceted nature of performance metrics in model assessment. While our model shows promise, especially in precision relative to the SAGRASP model, there's a clear avenue for refining its time efficiency. Such comparative analyses are not just an end in themselves; they serve as catalysts, directing efforts towards continuous innovation and development. In the ever-evolving realm of GRASP-based feature selection in high-dimensional data, it's pivotal to iterate and evolve. Embracing feedback, both from quantitative metrics and comparative benchmarks, is the cornerstone for future advancements in the field.

5.7 Future Work

While the current results are promising, there are several directions for future exploration. Firstly, additional datasets could be used to test the robustness of our model. Testing our approach on a variety of data types and domains can provide a better understanding of its applicability and potential limitations.

Secondly, implementing sampling techniques during our experiments could bring further insights. By varying the subsets of data our model is trained and evaluated on, we can gain deeper knowledge of its stability and consistency.

Thirdly, we aim to apply our model to regression problems. Currently, our focus has been mainly on classification tasks. Exploring regression problems would allow us to understand the model's performance in different predictive modeling contexts.

Lastly, we aim to compare our method with more state-of-the-art methods. By doing this, we can further situate our work in the current research landscape, identify the areas where our model excels, and recognize the ones where improvements can be made.

By following these directions in future research, we aim to continuously improve and refine our model, contributing to the progress in the field.

BIBLIOGRAPHY

Abreu, G., de Carvalho, A. C. P. L. F., & Lorena, A. C. (2011). A grasp algorithm for fast hybrid (filter-wrapper) feature subset selection in high-dimensional datasets. *Pattern Recognition Letters*, *32*(16), 2138–2146.

Bermejo, P., Gámez, J. A., & Puerta, J. M. (2011). A grasp algorithm for fast hybrid (filter-wrapper) feature subset selection in high-dimensional datasets. *Pattern Recognition Letters*, *32*(5), 701–711.

Blumer, A., Ehrenfeucht, A., Haussler, D., et al. (1987). Occam's razor. *Information Processing Letters*, *24*(6), 377–380.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Cancer, S. L. B. (2017). Scikit-learn: Machine learning in python.

Connectionist (2014). Uci connectionistbench. UCI Machine Learning Repository. https://doi.org/10.24432/C5T01Q.

Deekshit, V. K., Kumar, M. V. N. A., Sadasivuni, M. K., & Sadasivuni, R. (2019). Development of an intelligent diagnostic tool for the identification of bacterial pathogens using machine learning techniques. *Biomedical Signal Processing and Control*, *48*, 14–22.

Dorigo, M. & Di Caro, G. (1999). Ant colony optimization: A new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation*, (pp. 1470–1477).

Esseghir, M. & Idoumghar, L. (2010). Grasp for feature selection. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, (pp. 55–60). IEEE.

Fang, Y., Guo, Y., Zhao, Y., Guo, Y., & Zhao, H. (2018). A relief-based feature selection method combined with rank correlation coefficient to identify growth-driving genes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *15*(4), 1207–1218.

Feo, T. A. & Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, *6*(2), 109–133.

Feo, T. A. & Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, *8*(2), 67–71.

Festa, P. & Pardalos, P. M. (2002). Grasp: An annotated bibliography. *Computers & Operations Research*, *1*(3), 231–244.

FS, I. (2020). Itmo-fs. `https://pypi.org/project/ITMO-FS/`.

Fu, G.-H., Wu, Y.-J., Zong, M.-J., & Yi, L.-Z. (2020). Feature selection and classification by minimizing overlap degree for class-imbalanced data in metabolomics. *Chemometrics and Intelligent Laboratory Systems*, *196*, 103906.

Gandomi, A. H. & Yang, X.-S. (2013). Metaheuristic algorithms in modeling and optimization. In *Metaheuristic Algorithms in Modeling and Optimization* (pp. 1–12). Elsevier.

Genetic, S. L. (2016). sklearn-genetic. `https://pypi.org/project/sklearn-genetic/`.

Glover, F. (1989). Tabu search - part i. *ORSA Journal on Computing*, *1*(3), 190–206.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine*

Learning. Addison-Wesley Professional.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2003). Gene selection for cancer classification using support vector machines. *Machine Learning*, *46*(1-3), 389–422.

Iris, S. L. (2007). Scikit-learn: Machine learning in python.

Kennedy, J. & Eberhart, R. C. (1995). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, 1942–1948.

Kidney, C. (2015). $\text{Chronic}_{K}idney_{D}isease.UCIMachineLearningRepository.DOI: https://doi.org/10.24432/C5G020$.

Kira, K. & Rendell, L. (1992). A practical approach to feature selection. *In Proceedings of the Ninth International Workshop on Machine Learning*, 249–256.

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

Kittler, J. (1978). Feature set search algorithms, 41–60.

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

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of relief. *European conference on machine learning*, *784*, 171–182.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Moshki, M., Kabiri, P., & Mohebalhojeh, A. (2015). Scalable feature selection in high-dimensional data based on grasp. *Applied Artificial Intelligence*, *29*, 283–296.

Pilnenskiy, N. & Smetannikov, I. (2020). Feature selection algorithms as one of the python data analytical tools. *Future Internet*, *12*, 54.

Resende, M. G. C. & Ribeiro, C. C. (2003). Greedy randomized adaptive search procedures. In *Handbook of Metaheuristics*, (pp. 219–249). Springer.

Resende, M. G. C. & Werneck, R. F. (2004). A hybrid heuristic for the p-median problem. *Journal of Heuristics*, *10*(1), 59–88.

Ribeiro, C. C. & Resende, M. G. C. (2007). Grasp: Greedy randomized adaptive search procedures. In *Handbook of Heuristics*, (pp. 587–608). Springer.

Smiti, A. & Elouedi, Z. (2015). Feature selection for high-dimensional data clustering. *Applied Intelligence*, *42*(1), 147–163.

Urbanowicz, R. J., Meeker, M., La Cava, W., Olson, R. S., & Moore, J. H. (2018). Relief-based feature selection: Introduction and review. *Journal of Biomedical Informatics*, *85*, 189–203.

Wang, D., Li, H., Wang, J., & Pan, Y. (2006). Gene selection from microarray data for cancer classification—a machine learning approach. *Computational Biology and Chemistry*, *30*(6), 487–496.

Wine, S. L. (2007). Scikit-learn: Machine learning in python.