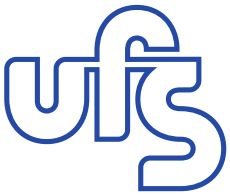


UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

A Novel approach to use Multi-Armed Bandit for Feature Selection

Proposta de Dissertação de Mestrado

Keomas da Silva Monteiro

São Cristóvão – Sergipe

2025

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Keomas da Silva Monteiro

A Novel approach to use Multi-Armed Bandit for Feature Selection

Proposta de Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): André Britto de Carvalho

São Cristóvão – Sergipe

2025

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Monteiro, Keomas da Silva.
M775n A novel approach to use Multi-Armed Bandit for feature selection / Keomas da Silva Monteiro; orientador André Britto de Carvalho. – São Cristóvão, SE, 2025.
83 f.: il.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2025.

1. Computação. 2. Aprendizado do computador. 3. Inteligência artificial. I. Carvalho, André Britto de, orient. II. Título.

CDU 004.85

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: Keomas da Silva Monteiro

Em 29 dias do mês de Agosto do ano de dois mil e vinte cinco, com início às 9hs, realizou-se na Sala de Seminários do PROCC da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Keomas da Silva Monteiro**, que desenvolveu o trabalho intitulado: **" A Novel approach to use Multi-Armed Bandit for Feature Selection",** sob a orientação do Prof. Dr. **André Britto de Carvalho .** A Sessão foi presidida pelo Prof. Dr. **André Britto de Carvalho** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. **Bruno Almeida Pimentel** (UFAL) e, em seguida, o Prof. Dr. **Renê Pereira Gusmão** (Procc/UFS). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) **Aprovado** "*(aprovado/reprovado)"*. Atendidas as exigências da Instrução Normativa 05/2019/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 04/2021/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos",* 29 de agosto de 2025.

gov.br Documento assinado digitalmente
ANDRE BRITTO DE CARVALHO
Data: 01/09/2025 14:51:14-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. André Britto de Carvalho
(PROCC/UFS)
Presidente

gov.br Documento assinado digitalmente
BRUNO ALMEIDA PIMENTEL
Data: 01/09/2025 17:25:36-0300
Verifique em https://validar.iti.gov.br

Prof. Dr Bruno Almeida Pimentel
(UFAL)
Examinador Externo

gov.br Documento assinado digitalmente
RENE PEREIRA DE GUSMAO
Data: 02/09/2025 16:13:55-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Renê Pereira Gusmão
(PROCC/UFS)
Examinador Interno

gov.br Documento assinado digitalmente
KEOMAS DA SILVA MONTEIRO
Data: 03/09/2025 07:29:17-0300
Verifique em https://validar.iti.gov.br

Keomas da Silva Monteiro
Candidato

Nem tudo que se constrói nasce do desejo de receber algo. Há obras que florescem da gratidão-silenciosa, profunda, sincera. Assim foi desta vez. Minha maior motivação foi honrar a pessoa que dia após dia é meu impulso. Agir em seu nome é um benefício à minha alma. À minha esposa N.M.

Soli Deo Gloria.

Acknowledgements

Aos meus amados pais, **Antonio** e **Monica**, sou eternamente grato. O incentivo constante aos estudos que sempre me ofertaram foi a base sólida sobre a qual construí minha jornada acadêmica. Sem a crença e o apoio inabaláveis de vocês, este momento não seria possível.

À minha querida esposa, **Nathalya Fontes** – minha amiga, confidente e amor fiel. Seu apoio incondicional, compreensão e paciência foram o porto seguro em todos os momentos. E ao nosso precioso **Olavo Augustus**, meu bebê e inspiração dia após dia, sua alegria e inocência são a maior motivação para buscar sempre o melhor.

À querida **Elbe Fontes** (vovó Elbe), que foi um suporte nos momentos mais difíceis, sem sua ajuda este trabalho não poderia ter sido escrito. Incontavéis foram as vezes quão providencial a senhora foi.

Ao Professor **André Britto**, meu orientador, expresso não apenas gratidão, mas uma profunda admiração. Sua presença em minha formação acadêmica, desde a graduação, tem sido um privilégio. Sua solicitude em cada etapa deste trabalho foram fundamentais. Sua orientação precisa e os valiosos ensinamentos que me transmitiu foram a bússola que guiou a pesquisa.

À secretaria do PROCC, agradeço imensamente pela presteza e eficiência.

A todos os professores que cruzaram meu caminho nesta jornada acadêmica, meu reconhecimento pela excelência no ensino e pela paixão em compartilhar o conhecimento.

Aos meus colegas de curso, a convivência e as trocas de experiências enriqueceram imensamente este período.

E, acima de tudo, a Deus e à Sua Providência, por guiar meus passos, conceder-me saúde, sabedoria e as oportunidades para alcançar mais este objetivo. A fé em Sua bondade foi o alicerce que sustentou toda a minha jornada.

O Senhor reina, portanto nada é por acaso
(João Calvino)

Abstract

This work explores the application of Multi-Armed Bandit (MAB) algorithms for feature selection (FS) in machine learning, aiming to address the challenges posed by high-dimensional data, such as computational complexity and overfitting. While traditional FS methods are widely used, the integration of MAB in this context remains unexplored. This research proposes novel MAB-based algorithms, specifically adapting the Epsilon-greedy (MAB-EgreedyFS) and Upper Confidence Bound (MAB-UCBFS) algorithms, to dynamically manage feature inclusion and exclusion. In this way, the feature set is formed with the aim of providing the best accuracy for the classifier in the classification task. For this, each feature's status is treated as an "arm" in the bandit problem approach that abstracts the search for the best features as the exploration-exploitation dilemma. During the process a Support Vector Machine (SVM) is applied as the classifier to evaluate the methods. A experimental set was performed, the proposed methods were evaluated in seven dataset and compared against established FS methods: SVM-RFE, extra-trees-based method, genetic algorithm-based method a ANOVA filter method. The results indicate that MAB-UCBFS consistently achieved strong performance, notably ranking as the "Best Method" for most of the evaluated data sets. While not universally superior, MAB-UCBFS demonstrated robust and competitive performance across most scenarios. Statistical analysis using Conover test heatmaps further corroborated these findings, highlighting significant differences between MAB-UCBFS and other techniques on several datasets. This study successfully validates the viability and strong performance of MAB-based algorithms, particularly MAB-UCBFS, as innovative and effective solutions for feature selection.

Keywords: multi-armed bandits, feature selection, machine learning.

List of Figures

Figure 1 – Main Types of Machine Learning 20
Figure 2 – Example of customer churn dataset 20
Figure 3 – Objective SVM 21
Figure 4 – SVM and nonlinearly 23
Figure 5 – Majority Voting enseble 24
Figure 6 – Adaboost 25
Figure 7 – General process of a filter model 29
Figure 8 – General process of a wrapper model 30
Figure 9 – General process of a embedded model 32
Figure 10 – Example of customer churn dataset 34
Figure 11 – Types of Machine Learning 39
Figure 12 – Feature set updating 45
Figure 13 – Dataset Breast Cancer Coimbra 51
Figure 14 – Breast Cancer Coimbra class distribution 52
Figure 15 – Wisconsin class distribution 53
Figure 16 – Ionoephere class distribution 54
Figure 17 – Diabetes Dataset 54
Figure 18 – Diabetes class distribution 55
Figure 19 – Mine class distribution 55
Figure 20 – Bim Detector class distribution 56
Figure 21 – Musk distribution 57
Figure 22 – Extra-trees from Scikitlearn 59
Figure 23 – SVM-RFE from Scikitlearn 60
Figure 24 – Anova from Scikitlearn 61
Figure 25 – Performance over k 64
Figure 26 – HeatMap Conover test Coimbra dataset 64
Figure 27 – Performance over k Diabetes dataset 66
Figure 28 – Heatmap Diabetes 66
Figure 29 – Performance over k Mine dataset 67
Figure 30 – Heatmap Mine dataset 68
Figure 31 – Performance over k Musk dataset 69
Figure 32 – Heatmap Musk dataset 69
Figure 33 – Performance over k Wisconsin dataset 70
Figure 34 – Heatmap Wisconsin dataset 71
Figure 35 – Performance over k Ionosphere dataset 72
Figure 36 – Heatmap Ionosphere dataset 72

Figure 37 – Performance over k BIM Detector dataset 73
Figure 38 – Heatmap BIM Detector dataset 73
Figure 39 – Cross-Validation Score Comparison Across Multiple Datasets. 74
Figure 40 – Heatmap Conover Test Results for Various Datasets 77

List of Tables

Table 1 – Type of FS Method 31
Table 2 – Feature Selection Method Categories Overview 32
Table 3 – Feature Selection Method Categories Overview 33
Table 4 – The generic search string 37
Table 5 – Number of papers returned by each database 37
Table 6 – Selection Criteria 37
Table 7 – MAB use in machine learning 38
Table 8 – Comparison of MAB-EgreedyFS and MAB-UCBFS 49
Table 9 – Datasets information 51
Table 10 – Comparison of Datasets Characteristics 58
Table 11 – Top FS method for Breast Cancer Coimbra dataset 63
Table 12 – Top FS method for Diabetes dataset 65
Table 13 – Top FS method for Mine dataset 67
Table 14 – Top FS method for Musk dataset 68
Table 15 – Top FS method for Wisconsin dataset 70
Table 16 – Top FS method for Ionosphere dataset 71
Table 17 – Top FS method for BIM Detector dataset 73
Table 18 – Comparison Table of Results 75
Table 19 – MAB-UCBFS Performance and Dataset Characteristics 76

List of Algorithms

1 Genetic Algorithm 28
2 Arm Selection in ϵ-Greedy 35
3 UCB1 36
4 MAB-EgreedyFS `best_arm()` method 47
5 MAB-EgreedyFS method 47
6 MAB-Eegredy `Reward` Function 47
7 `MAB-UCBFS best_arm()` method 48
8 `MAB-UCBFS` method 48
9 MAB-UCBFS `Reward` Function 49
10 Evaluation function 62

*

List of abbreviations and acronyms

MAB	Multi-armed Bandit
DCOMP	Departamento de Computação
UFS	Universidade Federal de Sergipe
FS	Feature Selection
SVM	Support Vector Machine
BCO	Bacterial colony optimization
ML	Machine Learning
UCB	Upper Confidence Bound
COD	Curse of Dimensionality
NLP	Natural Language Processing
AS	Active Search
IG	Information Gain
CBCD	Coimbra Breast Cancer Dataset
CMAB	Combinatorial multi-armed bandit
GBM	Gradient boosting machin
RFE	Recursive Feature Elimination
SVM-RFE	SVM - Recursive Feature Elimination
KNN	K-nearest neighbors
AI	Artificial Intelligence
DBSCAN	Density-Based Spatial Clustering of Applications with Noise

List of symbols

Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
$\in$	Pertence

Contents

1 Introduction 14
1.1 Objectives of the research 15
1.1.1 Specific objectives 16
1.1.2 Document Structured 16

2 Theoretical Background 18
2.1 Machine Learning 18
2.1.1 Support Vector Machines 20
2.1.2 Combining Models: Ensembles Methods 23
2.1.2.1 Random Forest 25
2.1.2.2 Extremely Randomized Trees 26
2.1.3 Genetic Algorithms 26
2.2 Feature Selection 28
2.3 Multi-Armed Bandits 33
2.3.1 Regret in Multi-Armed Bandits 33
2.3.2 Exploration-Exploitation in Feature Selection 34
2.3.3 Epsilon-Greedy 35
2.3.4 Upper Condence Bounds (UCB) 36
2.4 Related work 36
2.4.1 Research Questions 37
2.4.2 How Multi-armed bandit currently is associated with the Machine Learning field? 38
2.4.3 Multi-armed bandit can be applied for Feature Selection in Machine Learning? 39
2.4.4 Feature Selection Related Work 40

3 Multi-Armed bandit in Feature Selection 42
3.1 How Multi-armed bandit can be applied for Feature Selection? 42
3.1.1 The key to success 44
3.1.2 Detailed Rationale for Epsilon-Greedy and UCB 45
3.1.3 Applying Epsilon-Greedy Algorithm for Feature Selection 46
3.1.4 Applying Upper Condence Bounds Algorithm for Feature Selection . . 47
3.1.5 Comparison of MAB-EgreedyFS and MAB-UCBFS 49

4 Experimental Evaluations 50
4.1 Experiments 50

4.1.1 Datasets . . . 50
4.1.1.1 Breast Cancer Coimbra . . . 51
4.1.1.2 Breast Cancer Wisconsin . . . 52
4.1.1.3 Ionosphere . . . 52
4.1.1.4 Diabetes . . . 53
4.1.1.5 Mine . . . 54
4.1.1.6 BIM Detector . . . 56
4.1.1.7 Musk . . . 56
4.1.2 Algorithms Implementation . . . 58
4.1.2.1 Extra-trees-based . . . 59
4.1.2.2 SVM-RFE . . . 60
4.1.2.3 Filter-based ANOVA f-test . . . 60
4.1.2.4 Genetic Algorithm . . . 61
4.1.2.5 MAB-EgreedyFS and MAB-UCBFS . . . 62
4.1.3 Evaluation Metrics and Statics . . . 62
4.1.4 Evaluation by Dataset . . . 63
4.1.4.1 Feature Selection on Breast Cancer Coimbra . . . 63
4.1.4.2 Feature Selection on Diabetes . . . 64
4.1.4.3 Feature Selection on Mine . . . 66
4.1.4.4 Feature Selection on MUSK . . . 68
4.1.4.5 Feature Selection on WINSCONSIN . . . 69
4.1.4.6 Feature Selection on Ionosphere . . . 71
4.1.4.7 Feature Selection on BIM Detector . . . 72
4.1.4.8 Put all togheter . . . 74

5 Conclusion . . . 78

Bibliography . . . 80

1 Introduction

Machine learning algorithms, powered by advances in computing capabilities and vast datasets, play a pivotal role in extracting valuable insights, making predictions, and automating complex tasks across various industries. In this context a challenge that players must overcome is called the "curse of dimensionality" (COD), datasets that exhibit high dimensionality due to a massive number of features (LI et al., 2017). High-dimensional data poses unique challenges for machine learning models, leading to increased computational complexity, potential overfitting, and greater storage requirements. Tackling these challenges is crucial for ensuring the effectiveness and interpretability of machine learning models, making advancements in algorithms and techniques for handling high-dimensional data a pressing area of research in the field of machine learning.

It is possible to obtain a reduction in dimensionality using feature selection (FS) or feature extraction (FE) (DHAL; AZAD, 2022). FS is the process of finding a subset of features from the original set, whereas FE is the process of extracting valuable information to generate a new feature space, which can generally lead to better results.

However, in several fields (e.g. healthcare or finance) it is important to keep the original meaning of the data, and FS becomes more appropriate. The aim of FS is to identify the most relevant subset of features from a larger set to enhance model performance and interoperability. The key focus lies in discerning or defining the relationship between features (or attributes) and establishing the correlation or dependence between features and the class (also referred to as the outcome or dependent variable). These relationships are crucial because not all features are indispensable; some may be irrelevant or redundant, potentially leading to a significant deterioration in the performance of the classifier. Feature Selection often results in models easier to explain and interpret, suitable for fields where is necessary to know why a decision was made.

There are several types of methods for applying FS, such as statistically based (BAHASSINE et al., 2020), probability based (WAHID et al., 2020), similarity measure based (LIU et al.,

2018), evolutionary based (WANG; JING; NIU, 2017), and other methods (ADORADA et al., 2018). Each has a lot of research work.

The wide application of FS appears in some works. Wang and colleagues introduced the Bacterial Colony Optimization (BCO) method (evolution-based), a branch of AI and soft computing, for the selection of characteristics in the classification of cancer problems of microarray gene expression (WANG; JING; NIU, 2017). In (JEON; OH, 2020) a new feature selection method is proposed, using recursive feature elimination (RFE) applied with the classification models: Support Machine Vector (SVM), Random forests(RF), and Gradient boosting machine (GBM). A Hybrid-RFE was developed, which is an ensemble formed by theses models, combining their feature weighting functions. Recently, (ALFIAN et al., 2022) proposed a combination of extra trees and support vector machine techniques to predict breast cancer, showcasing competitive performance. Those works have some concerns, for example, (WANG; JING; NIU, 2017) has datasets with a low number of instances compared to the high dimensionality, such datasets may lead to overfitting, suggesting the need for more structured evaluation like cross-validation. In addition, the comparison with other machine learning models in (ALFIAN et al., 2022) used different evaluation techniques. In addition, most FS methods need the number of features to be selected as a parameter, which demands predefined domain knowledge.

However, an research area little explored in FS context is the simple but very powerful framework Multi-Armed Bandits (MAB). Multi-armed bandits (MAB) is a framework for algorithms that make decisions over time under uncertainty (SLIVKINS et al., 2019), offering a very clean, simple theoretical formulation for analyzing trade-offs between exploration and exploitation. The decision which features select to obtain the best accuracy is a problem that MAB can solve. Drawing an analogy from the casino setting, we have a row of slot machines (bandits) where a gambler must decide which to play to maximize their cumulative rewards. MAB provides an elegant solution dealing with the exploitation-exploration dilemma (SLIVKINS et al., 2019). The versatility of MAB algorithms makes them applicable in various settings, like problems in Online Advertising, Clinical Trials, Content Recommendation, Network Routing, Dynamic Pricing, and Energy Management.

MAB is still little explored in FS context. Examples of the MAB approach in this context are Active Search (AS) algorithms for Natural Language Processing (NLP), aiming to find as many positive instances as possible to train a model and construct a knowledge base (ZHU; COLES; XIE, 2020) and in (LIU et al., 2021) a combinatorial multi-armed bandit (CMAB) task.

1.1 Objectives of the research

The main objective of this work is to propose MAB-based algorithms for feature selection and validate its capability in this area of study, instigating more researches. The proposed methods are based on the Epsilon-greedy and on the Upper Condence Bounds (UCB) algorithms

(VERMOREL; MOHRI, 2005), both well-known MAB solutions. Each one deals with the trade-offs between exploration and exploitation in a clever way. Conducting an investigation and research on the application of MAB in feature selection holds substantial promise for contributing to FS task. By exploring the integration of MAB algorithms into feature selection processes, the study has the potential to offer innovative solutions to challenges associated with optimizing model performance in machine learning. MAB algorithms, known for their ability to balance exploration and exploitation, can provide valuable insights into dynamic selection and prioritizing features for improved accuracy. This research may lead to the development of more efficient and adaptive feature selection methods, enhancing the interpretability and generalization capabilities of machine learning models. Ultimately, the exploration of MAB in feature selection can contribute novel perspectives and methodologies to the broader field of machine learning research.

1.1.1 Specific objectives

The following specific objectives were defined to achieve the main objective:

1. Perform review of the literature about MAB in feature selection;
2. Validate the hypothesis that MAB-based algorithms for feature selection can be used;
3. Propose novel methods for feature selection using MAB-based algorithms variations;
4. Execute a comparative experiment between the proposed algorithms and the main types of algorithms found in the literature for feature selection;

1.1.2 Document Structured

This document is structured with chapters and sections to enhance comprehension, facilitate navigation, and promote a seamless reading experience. Each chapter is outlined as follows:

1. Chapter 1 - Introduction: This section provides initial definitions from the literature, addresses the problem statement, presents arguments and hypotheses related to the topic, outlines the objectives, and discusses the proposed methodology for achieving these objectives.
2. Chapter 2 - Theoretical Background: This section delves into the theoretical context, incorporating a literature review (systematic mapping) pertinent to the proposed theme. It encompasses the outcomes of this review, a synthesis of the studies examined. Also, elucidates fundamental concepts of machine learning, feature selection, and Multi-Armed Bandits, concluding with a focused discussion on the current state and challenges of feature selection.

3. Chapter 3 - This chapter details the innovative core of our work. It meticulously outlines the various research steps undertaken, presenting our novel MAB-based algorithms for feature selection. This chapter explain how Multi-Armed Bandit techniques are applied in novel ways to solve feature selection problems, providing the specific adaptations of Epsilon-greedy and Upper Confidence Bound (UCB) algorithms.

4. Chapter 4 - Experimental Evaluations: This section provides an evaluation to validate the effectiveness of MAB-based algorithms for feature selection. Details the experimental setup, including the datasets used, the implementation specifics of our algorithms, and the rigorous evaluation metrics and statistical analyses used. Discussion of the results, comparing our MAB approaches against established feature selection techniques across diverse datasets.

5. Chapter 5 - Conclusion: The final chapter summarizes the entire investigation. Synthesizes the key findings, discusses their implications, and provides a conclusive perspective on the topic. This section also highlights the contributions of our work to the field and suggests promising avenues for future research.

2

Theoretical Background

In this chapter fundamental concepts about machine learning and data for better understanding the work are provided. The concepts are followed by a fundamental theoretical discussion about feature selection and its application and the Multi-armed Bandit (MAB) approach and method. Finally, the chapter presents a systematic map focusing on the utilization of Multi-armed bandit algorithms in the domain of machine learning, aiming to offer a comprehensive overview of its current usage and potential.

2.1 Machine Learning

Machine Learning (ML) is a subfield of Artificial Intelligence (AI), focused in developing algorithms and models able to derive knowledge from data in order to make predictions (RASCHKA; MIRJALILI, 2019). This is achieved building algorithms that can learn patterns by themselves without being programmed explicitly. Thanks to machine learning, nowadays it is possible to have robust email spam filters, convenient text and voice recognition software, reliable web search engines, solve classification issues in medical applications, and several areas from real-world. The three main types of learning in ML is: *supervised learning*, *unsupervised learning* and *reinforcement learning*.

In supervised learning the goal is to learn a model from labeled training data that allows us to make predictions about unseen or future data. An algorithm is trained with structured data to learn patterns. The figure 2 shows an example of a dataset that could be used to train a model that will predict whether a customer will cancel their subscription, for example.The dataset has a column (or variable) that already contains the churn outcome (cancel or not cancel) for past or existing customers. The objective of the model is to learn the relationship between this outcome column and the other features (also called independent variables or predictor variables). A supervised learning task with discrete class labels, such as in this example, is

called a **classification** task. Another subcategory of supervised learning is **regression**, where the outcome signal is a continuous value. **Decision Tree**, **Support Vector Machine (SVM)**, **K-nearest neighbors (KNN)** and **Perceptron** are well-known examples of algorithms to solve classification tasks.

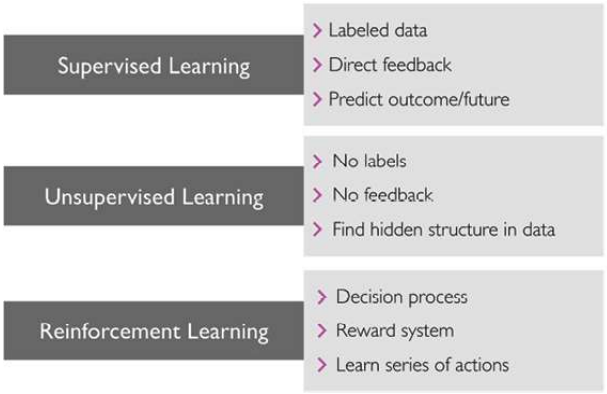
Unsupervised learning is a type of algorithm that explore the structure of our data to extract meaningful information without the guidance of a known outcome variable, the model learn patterns from the data by itself. This type of algorithm usually can detect similarities between variables or records, so it will try to group those that are very close to each other. This kind of algorithm can be used for clustering (grouping records) or dimensionality reduction (reducing the number of variables) (SO et al., 2020). **Density-Based Spatial Clustering of Applications with Noise (DBSCAN)** and **K-means** are algorithms samples widely used for clustering.

Reinforcement learning (RL) is a type of machine learning algorithm that learns how to act in a specific environment based on the feedback it receives. The goal is to develop a system (agent) that improves its performance based on interactions with the environment. The agent maximize some notion of cumulative reward over time. In other words, the agent learn a series of actions that maximizes this reward via an exploratory trial-and-error approach or deliberative planning (RASCHKA; MIRJALILI, 2019). The key components of reinforcement learning include:

- Agent: The entity that takes actions in the environment. It is the decision maker who learns to perform tasks.
- Environment: The external system with which the agent interacts. The environment responds to the actions taken by the agent and provides feedback in the form of rewards or penalties.
- State: The current situation or configuration of the environment, which the agent uses to make decisions. The state is usually represented as a set of variables that capture relevant information about the environment.
- Action: The set of possible moves or decisions that the agent can make in a given state. The actions lead to transitions from one state to another.
- Reward: A numerical signal that the environment provides to the agent after it takes an action in a particular state. The reward serves as feedback, indicating the desirability of the action.

Games and robotics are fields where RL is applied, **Monte Carlo** and **Q-learning** are RL algorithms widely used. The figure 1 shows an overview of the types of machine learning.

Figure 1 – Main Types of Machine Learning



Source: (RASCHKA; MIRJALILI, 2019)

Figure 2 – Example of customer churn dataset

Target Features

Cancel	Months since first subscription	Monthly average spent	Average number of phone calls made last month	Average number of phone calls made last quarter	Additional options
Yes	13	$70	56	63	Yes
No	2	$35	35	34	Yes
No	6	$40	46	50	Yes
Yes	16	$110	53	75	No

Source: (SO et al., 2020)

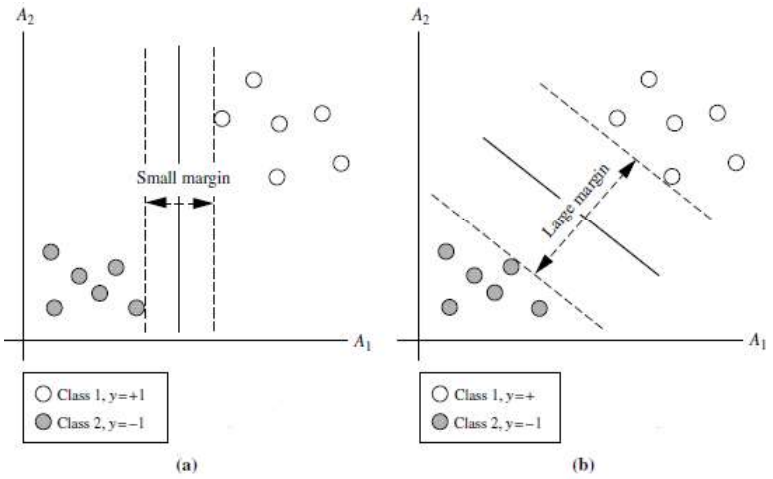
2.1.1 Support Vector Machines

Support Vector Machines (SVMs) is a method for the classification of both linear and nonlinear data. In SVMs the objective is to maximize the "margin".

Given a separating hyperplane that classifies the samples in the dataset, the margin is defined as the distance between this separating hyperplane (decision boundary) and the training samples that are closest to this hyperplane, which are the so-called support vectors.

In the figure 3 the objective of SVM is presented, given a linear classification task, to create decision boundaries with a large margin as possible, in a way opposite to one with a small.

Figure 3 – Objective SVM



Font: (HAN; KAMBER; PEI, 2012)

This provide a model with low generalization error whereas models with small margin tends to overfitting[1].

First to formulate this, let the dataset be given as the $\boldsymbol{X}$ $(x_1 \ldots x_n)$ tuple attributes and the ***class labels*** y_1, y_2 (targets to be predicted). So

$$w_0 + w_1 x_1 + w_2 x_2 = 0 \tag{2.1}$$

is a decision boundary (hyperplane) that separates the two classes (y_1, y_2) like in figure 3. As in the previous sections, $(w_0, \ldots, w_2)$ are the weights that in combination with the input attributes performs a decision boundary. $\boldsymbol{W}$ is a vector of wights. Thus, any point above the separating hyperplane satisfies

$$w_0 + w_1 x_1 + w_2 x_2 > 0 \tag{2.2}$$

In the same way, any point below the separating hyperplane satisfies the conditions.

$$w_0 + w_1 x_1 + w_2 x_2 < 0 \tag{2.3}$$

Based on that, it is possible to define the 'sides' hyperplanes of the margin as

$$\textbf{h1} : w_0 + w_1 x_1 + w_2 x_2 \geq 1 \quad for \quad y_1 = 1 \tag{2.4}$$

$$\textbf{h2} : w_0 + w_1 x_1 + w_2 x_2 \leq -1 \quad for \quad y_2 = -1 \tag{2.5}$$

[1] The model can not generalize, leading to a bad accuracy

Combining

$$y_i(w_0 + w_1x_1 + w_2x_2) \geq 1, \forall_i \tag{2.6}$$

So any tuple that falls on or above h1 belongs to class +1, and any that falls on or below h2 belongs to class -1. The training samples that fall on hyperplanes H1 or H2 (i.e., the "sides" defining the margin) are called support vectors. In figure 3, the support vectors are shown encircled with a thicker border. To find the support vectors- what means find large distance and the best hyperplane- is the training goal. This distance is given by: Let

$$w0 + \mathbf{WXpos} = 1$$

and

$$w0 + \mathbf{WXneg} = -1$$

Distance is:

$$\mathbf{W(Xpos\text{-}Xneg)} = 2$$

Where **Xpos** is (x_1, x_2) on **h1** and **Xneg** on **h2**, **W** is the weight vector. Normalizing by the norm $\|W\|$:

$$\frac{W(Xpos - Xneg)}{\|W\|} = \frac{2}{\|W\|}$$

So maximize $\frac{2}{\|W\|}$ is the real objective. This can be achieve with quadratic programming based on Lagrangian formulation Karush-Kuhn-Tucker (KKT) Han, Kamber e Pei (2012). It is in this way how a svm solve linearly separable classification tasks, this approach provides greats performances making SVM very popular. Details about the necessary math in Vapnik (1998).

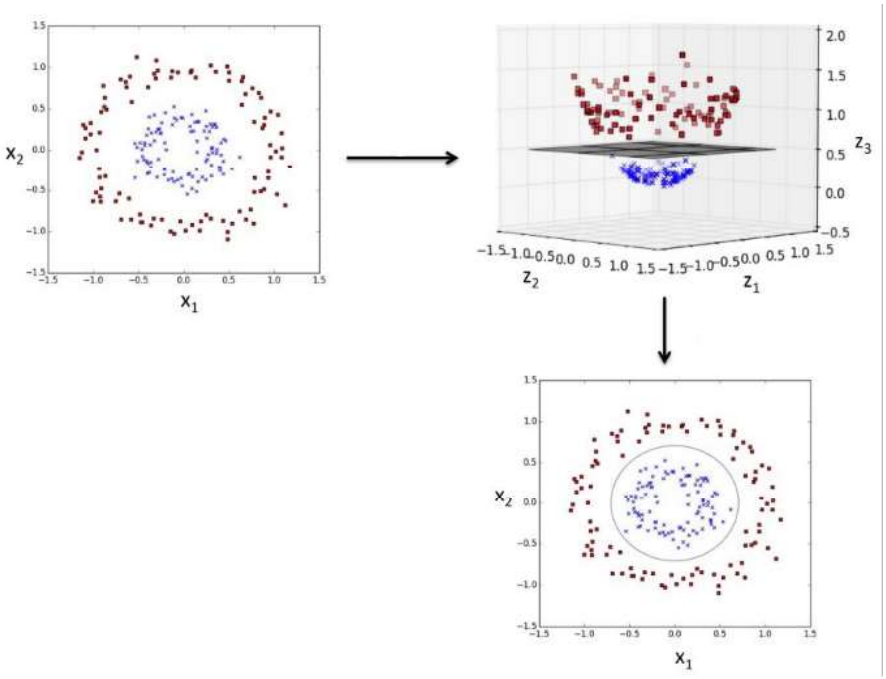
When working with data nonlinearly separable, the previous approach can be extended making SVM able to provide nonlinear decision boundary. To do that, first it is necessary transform the original input data into a higher dimension using a kernel function, after that it is possible find a hyperplane able to separate as before, the figure 4 represent that.

The maximal marginal hyperplane found in the new space corresponds to a nonlinear separating hypersurface in the original space (HAN; KAMBER; PEI, 2012). A Kernel function is a math trick that combine samples from the dataset to change their dimensionalty. For example, $K(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2)$. The principal kernel functions used are Han, Kamber e Pei (2012):

Polynomial kernel of degree *h*:	$K(x_1, x_2) = (x_1x_2)^2$
Gaussian radial basis function kernel (RBF):	$K(x_1, x_2) = e^{-\|x_1-x_2\|^2/2\sigma^2}$
Sigmoid kernel:	$K(x_1, x_2) = tanh(\kappa x_1x_2 - \delta)$

Each ***Kernel*** results in a different nonlinear classifier in (the original) input space. For example an SVM with a sigmoid kernel is equivalent to a multilayer perceptron (with no hidden layers).

Figure 4 – SVM and nonlinearly



Font: Raschka, Sebastian(2015)

Although SVM is a powerful classifier, there are no rules for determining which admissible kernel will result in the most accurate SVM, the selection of algorithms for a specific problem is a other problem to be solve in machine learning, what will be exposed in the next chapter.

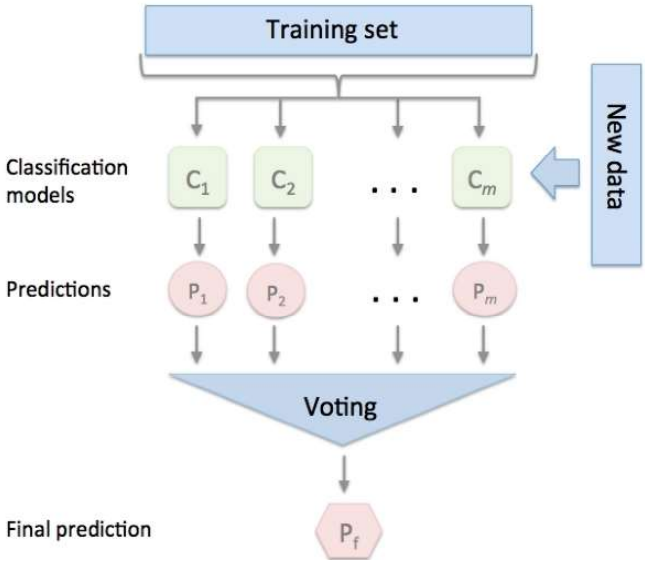
2.1.2 Combining Models: Ensembles Methods

An alternative approach for solve classification tasks, is the combination of learning models, known as Ensemble learning. The goal of ensemble is to combine differents models(SVMs, KNN, Logistic regression) into a meta-classifier that has a better performance. A ensemble tends to be more accurate than its base classifiers (it is not a rule). For example, a well known ensemble method is the majority voting. This methods simply means that the class label that will be predicted is the one with the majority vote from the base classifiers, that one received more than 50 percent of the votes. So, while with only one classifier a misclassification is the imediate output, in a ensemble is necessary that more of half from the base classifiers also outputs a wrong class. The figure 5 shows majority voting ensemble idea.

Besides this methods another popular is the Adaptaitve Boosting (AdaBoost). Adaboost is a modification on the **boosting** ensemble method. In **boosting**, the ensemble consists of very simple base classifiers, known as **weak learners** (typically decision tree stump), that have only a slight performance advantage over random guessing. The boosting procedure is summarized in the steps Raschka e Mirjalili (2019):

1. Draw a random subset of training samples d_1 without replacement from the training set $\boldsymbol{D}$

Figure 5 – Majority Voting enseble



Font: Raschka, Sebastian(2015)

to train a weak learner C_1 .

2. Draw second random training subset d_2 without replacement from the training set and add 50 percent of the samples that were previously misclassified to train a weak learner C_2.

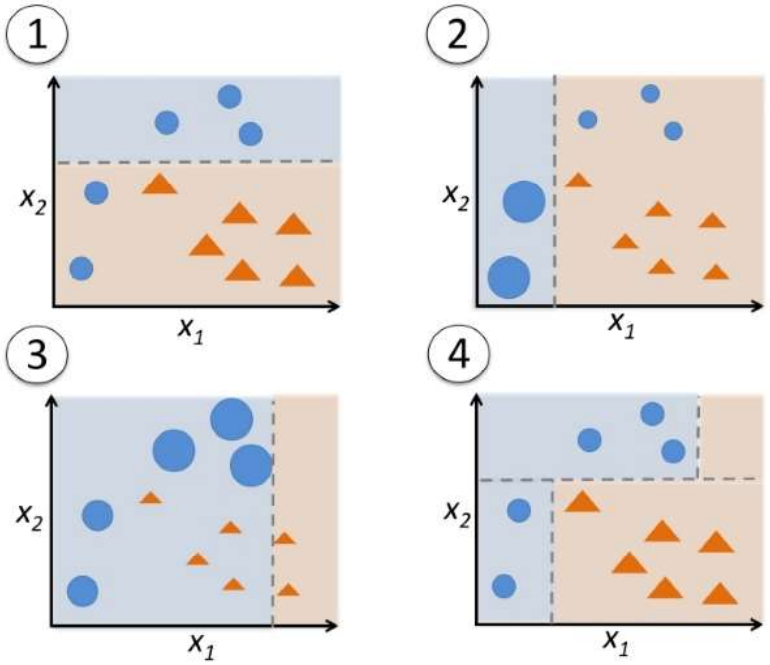
3. Find the training samples d_3 in the training set $\boldsymbol{D}$ on which C_1 and C_2 disagree to train a third weak learner C_3.

4. Combine the weak learners C_1, C_2 and C_3 via majority voting.

The goal is focus on training samples that are hard to classify, letting the weak learners subsequently learn from misclassified samples to improve the performance of the ensemble. AdaBoost has the same objective but with a different approach. It uses the complete training set to train the weak learners, and **reweights** the training samples in each iteration in the training process. The weights of samples misclassified will increase and in a opposite way, weights of samples correctly classified will be decreased. The figure 6 shows the idea behind Adaboost ensemble.

So, in the figure 6 the subfigure one represent a dataset where all samples has the same weights. The dashed line represents a weak learner trained, that classifies wrong two circles. In the next round (subfigure 2), a large weights is assign to this two misclassified samples and another weak learner classifies wrong others three circles. In the same way on the subfigure three, theses three samples receives a large update weights and another weak learner is trained. So finally, in the final step, the three weak learners trained on different reweighted training subsets are combined using a weighted majority vote, as shown in subfigure 4.

Figure 6 – Adaboost



Font: Raschka, Sebastian(2015)

This is how Adaboost works, first formulated by Schapire (1990) and widely used. However, Adaboost is also known for its tendency to overfit Raschka e Mirjalili (2019).

2.1.2.1 Random Forest

Another very popular ensemble method is the **Random forest**. That is a ensemble of decision trees classifiers, (a "forest" of decision trees). Its popularity comes from the good performance, scalability and ease for use. The random forest algorithm can be summarized in four simple steps (RASCHKA; MIRJALILI, 2019):

1. Get a random bootstrap sample of size *n* (randomly choose n samples from the training set with replacement).
2. Grow a decision tree from the bootstrap sample. At each node:
   a) Randomly select *d* features without replacement.
   b) Split the node using the feature that provides the best split according to the objective function, for instance, by maximizing the information gain.
3. Repeat the steps 1 to 2 *k* times (number of decision trees).
4. Aggregate the prediction by each tree to assign the class label by majority vote.

In other words a random forest is a set of *k* decision trees, where each tree is build with random samples from dataset and considering random features for its construction. In general

using a large k leads a better accuracy, but it is necessary dealing with the tradeoff accuracy versus computational cost. Other tradeoff that must be considered is the value for n, a large value decrease the randomness and thus leads it to overfit. On the opposite, choosing a small value for n reduces the overfit once increase the randomness. About the number of d features at each split, a reasonable default value that is used in general implementations is $\sqrt{m}$, where m is the number of features in the training set. More about the implementation of random forest in Raschka e Mirjalili (2019).

2.1.2.2 Extremely Randomized Trees

Another tree-based ensemble is the Extremely Randomized Trees (GEURTS; ERNST; WEHENKEL, 2006), often referred to as Extra-Trees, represent a novel tree-based ensemble learning method designed for both supervised classification and regression tasks. Unlike traditional decision tree algorithms that meticulously search for the optimal attribute and cut-point for splitting nodes, Extra-Trees introduce a significant degree of randomization in these choices. In its most extreme form, the algorithm constructs entirely randomized trees whose structures are independent of the learning sample's output values. The core idea is to leverage strong randomization in conjunction with ensemble averaging to achieve a more substantial reduction in variance compared to other methods, while simultaneously minimizing bias by utilizing the entire learning sample rather than bootstrap replicas. This approach not only contributes to improved accuracy but also offers notable computational efficiency due to the simplified node-splitting procedure. The Extra-trees algorithm can be summarized in the following four steps:

1. Build multiple decision trees, each using the complete training dataset.
2. At each node, randomly select a subset of attributes and then randomly choose a cut-point for each of them.
3. Pick the best split from these randomly generated options to divide the node.
4. Combine the outputs of all individual trees to form the final prediction.

2.1.3 Genetic Algorithms

In the 1950s and the 1960s several computer scientists independently studied evolutionary systems with the idea that evolution could be used as an optimization tool for engineering problems. The idea in all these systems was to evolve a population of candidate solutions to a given problem, using operators inspired by natural genetic variation and natural selection. Many computational problems require searching through a huge number of possibilities for solutions, which is right to say that evolution strategies, evolutionary programming, and genetic algorithms solve optimization problems and together form the field of evolutionary computation (MITCHELL, 1998).

Genetic algorithms (GAs) were invented by John Holland in the 1960s at the University of Michigan. Holland's GA is a method for moving from one population of "chromosomes" (e.g., strings of ones and zeros, or "bits") to a new population by using a kind of "natural selection" together with the genetics-inspired operators of crossover, mutation, and inversion. A "chromosomes" represents a candidate solution, thus population of "chromosomes" is a set of candidate solutions. Through crossover, mutation and inversion the GA processes populations of chromosomes, successively replacing one such population with another. The GA most often requires a fitness function that assigns a score (fitness) to each chromosome in the current population. The fitness of a chromosome depends on how well that chromosome solves the problem at hand.

The simplest form of genetic algorithm involves three types of operators: selection, crossover (single point), and mutation:

- **Selection**: This operator selects chromosomes in the population for reproduction. The fitter the chromosome, the more times it is likely to be selected to reproduce.
- **Crossover**: This operator randomly chooses a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring. For example, the strings 10000100 and 11111111 could be crossed over after the third locus in each to produce the two offspring 10011111 and 11100100. The crossover operator roughly mimics biological recombination between two single-chromosome (haploid) organisms.
- **Mutation**: This operator randomly flips some of the bits in a chromosome. For example, the string 00000100 might be mutated in its second position to yield 01000100. Mutation can occur at each bit position in a string with some probability, usually very small (e.g., 0.001).

Since is set a defined problem to be solved and a bit string representation for candidate solutions, a simple GA works as the algorithm 1.

Algorithm 1 Genetic Algorithm

Start with a randomly generated population of n l-bit chromosomes (candidate solutions to a problem).
Calculate the ***fitness*** $f(x)$ of each chromosome x in the population.
repeat
Select a pair of parent chromosomes from the current population.
Cross over the pair at a randomly chosen point (chosen with uniform probability) to form two offspring.
Mutate the two offspring at each locus with probability p_m, place the resulting chromosomes in the new population.
if n is odd **then**
One new population member can be discarded at random.
end if
until n offspring have been created
Replace the current population with the new population.
Go to step 2.

2.2 Feature Selection

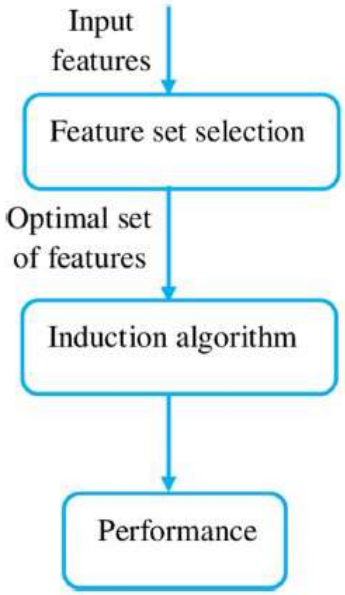
In the digital age era, every minute, billion, or even trillions of data is generated and this data turn possible to machine learning to solve problems in several areas like disease analysis or diagnosis, social platform, financial analysis, weather prediction, online educational platform, bioinformatic, and multiple security platforms. However, the fast development of data raises numerous difficulties for the successful and effective organization of data. Whenever these big data, ML, or data mining approaches are applied, a severe issue is critically arising, which is a Curse of Dimensionality (COD) (DHAL; AZAD, 2022). COD is an event that occurs when the high-dimensional data does not organize, classify, and analyze in a lower-dimensional space; practically, it occurs due to the closeness and sparsity of data. A dataset with a massive number of features in a classification task produces a model that tends to be overfitted with a bad performance, also high dimensional data significantly increases the computational complexity for the classification model and space complexity for the storage requirements. A feature is an attribute, characteristic or measurable property of an object that is being observed. For example, in the medical field, to detect skin cancer in the human body, various types of features can be used in ML algorithms such as creatinine levels in the blood, neutrophils range, lymphocytes, monocytes, eosinophils, etc.

Feature Selection (FS) is a type of process that addresses those issues, which is a dimensionality reduction technique. FS is the task of selecting the essential or relevant features so that there will be a chance of improving the performance and accuracy of the classification task. By eliminating the irrelevant features, the FS task reduces the data's dimensionality, accelerates the classification process, and accuracy or performance increase.

In literature, FS task is divided into three models: **Filter model** (figure 7), **Wrapper**

model (figure 8) and **Embedded model** (figure 9). Filter model, evaluates the quality of the feature subset, independent of any specific machine learning algorithm, selection of features is based on statistical measures or heuristics (mutual information, chi-squared), typically applied before the learning algorithm, in (WAHID et al., 2020) is shown a filter model in action. The Wrapper model uses a specific machine learning algorithm to evaluate the performance of different feature subsets, and involves a search process to find the optimal subset, in (LI; MENG, 2012) SVM - Recursive Feature Elimination (SVM-RFE) algorithm was applied as a wrapper model. In the Embedded model feature selection is integrated into the model training process, and important features are determined during the model training, such as Random Forests. The table 1 shows the characteristics of each method.

Figure 7 – General process of a filter model

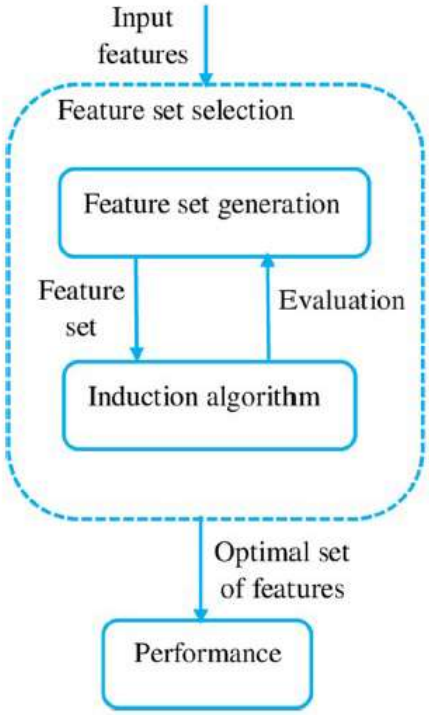


Source: (DHAL; AZAD, 2022)

There are different evaluation techniques for a feature selection method. It finds the answer to the optimal set features that could increase the accuracy of the classifier. These are the following measures which are applied in the feature subset evaluation task,

- Divergence measure: This feature assessment metric is frequently used to assess the discriminative or distinguishing ability of features. Indicates how effectively features are differentiated within classes. A feature with a greater divergence distance holds more significance than one with a lower divergence distance, highlighting its discriminative power. One of the important divergence method named as **Kullback–Leibler divergence** (COETZEE, 2005) is used in fluid mechanics, neuroscience, and machine learning.

Figure 8 – General process of a wrapper model



Source: (DHAL; AZAD, 2022)

- **Information Gain (IG) or uncertainty measure**: Information Gain (IG) serves as an uncertainty measure in the context of feature selection. When applied to feature selection, IG helps identify and prioritize features that contribute the most to reducing uncertainty or entropy in a dataset. The goal is to select features that provide the most valuable information for predicting the target variable in a machine learning task.

- **Dependency measure**: This metric attempts to quantify the degree of correlation or association between features and the class variable. Hence, it is commonly referred to as a dependency measure or correlation/association measure. In contrast to the feature evaluation process based on Information Gain, which considers how feature values change with respect to posterior and prior probabilities, our focus here is solely on understanding how features exhibit strong associations with the class. One of the important method **Hilbert-Schmidt Independence Criterion (HSIC)** (YAMADA et al., 2014) uses this dependency measure for the feature evaluation process.

- **Consistency measure**: It is the probability-based approach, that tries to find the minimal number of feature subsets that can classify the problem equivalent to when the full feature set can classify the problem. One of the best methods is the **Las Vegas algorithm** (LIU; MOTODA, 1998) uses the consistency measure for the feature evaluation process.

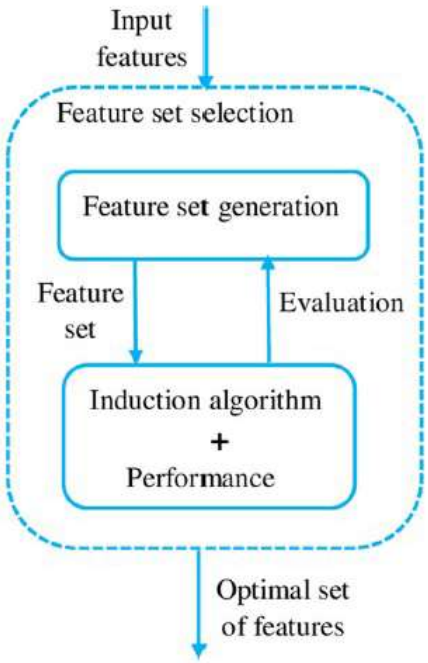
Table 1 – Type of FS Method

FS Method	Role in the FS process
Filter model	-This method goes through various statistical measures for the assignment of rank or score to each feature. – This method selects feature subsets before the learning model. – The process for selecting a feature subset is done only once in the entire building learning model's mechanism. – This method follows the feature relevance property to generate the feature subset. – The process for the generation of feature subset is independent of any ML algorithm. – This method follows the inherent property of the features. Also, it neglects the dependency among the features for the generation of feature subset. – This method doesn't remove the multicollinearity property. It should be removed alternatively by another mechanism.
Wrapper Model	– This method goes the accuracy of the model for the assignment of rank or score to each feature. – This method follows the dependency among the features for the generation of the feature subset. – For the generation of feature subset, the classifier runs multiple times. – This method goes for the generation of feature subset by considering the learning model to be applied. – This method selects the most useful features for the generation of the feature subset. – From the inference of the previous learning model, this method decides whether to add or remove the feature from the feature's subset. – This method uses a specific classifier for the evaluation of the quality of the selected features.
Embedded model	– The FS process builds in the training phase. And, this method evaluates the feature subset for the algorithm being trained. – This method uses the usefulness property of the features to generate the feature subset. – This method uses the learning mechanism for the search space. – This method uses the aggregation of the advantages of the filter and wrapper method. – This approach calculates dependency among the features very effectively. Due to the consideration of the classifier,this method selects the relevant features.

Source: (DHAL; AZAD, 2022)

- **Accuracy measure**: This measure depends on the classifier performance. In FS's whole process, the only goal is to find the optimal feature subset and get the optimal feature subset that only relies upon for the best predictive accuracy.

Figure 9 – General process of a embedded model



Source: (DHAL; AZAD, 2022)

There are several types of FS algorithm, such as: statistically based (BAHASSINE et al., 2020), probability-based (WAHID et al., 2020), similarity (LIU et al., 2018) measure-based, and Evolutionary algorithm based (WANG; JING; NIU, 2017) (table 2, table 3). But there are also other methods not included in these groups. Novel methods that approaches feature selection from abstractions of computer science as all (MAB-based, SVM-RFE).

Table 2 – Feature Selection Method Categories Overview

Category	**Description/Principle**	**Examples from Literature**
Statistical Measure Based	Assigns rank or score to each feature through various statistical measures.	Chi-Square, Analysis of variance (ANOVA).
Probability Measure Based	Identifies features that contribute most to reducing uncertainty or entropy in a dataset.	Mutual Information, Information Gain (IG).
Similarity Measure Based	Quantifies the degree of correlation or association between features and the class variable.	Fisher Score, Relief
Evolutionary Algorithm Based	Evolves a population of candidate solutions to a given problem, using operators inspired by natural genetic variation and natural selection.	Genetic Algorithm (GA), Bacterial Colony Optimization (BCO).

Source: (DHAL; AZAD, 2022)

Table 3 – Feature Selection Method Categories Overview

Category	**Pros**	**Cons/Limitations**
Statistical Measure Based	Simple, efficient for numerical features.	May not capture complex relationships.
Probability Measure Based	Quantifies information contribution.	Can be computationally intensive for large datasets.
Similarity Measure Based	Focuses on feature relatedness.	May not capture complex, nonlinear dependencies.
Evolutionary Algorithm Based	Good for global optimization, explores complex search spaces.	Can be computationally expensive, sensitive to parameter tuning.

Source: (DHAL; AZAD, 2022)

2.3 Multi-Armed Bandits

Multi-armed bandits (MAB) is a simple but very powerful framework for algorithms that make decisions over time under uncertainty (SLIVKINS et al., 2019), offering a very clean, simple theoretical formulation for analyzing trade-offs between exploration and exploitation. The term "multi-armed bandits" comes from a stylized gambling scenario in which a gambler faces several slot machines, a.k.a. one-armed bandits, that appear identical, but yield different payoffs.

In its simplest formulation (stochastic), a bandit problem consists of a set of **K** probability distributions $\langle D_1, ..., D_K \rangle$, with associated expected values $\langle \mu_1, ..., \mu_K \rangle$ and variances $\langle \sigma_1^2, ..., \sigma_K^2 \rangle$. Initially, the D_i are unknown to the player.

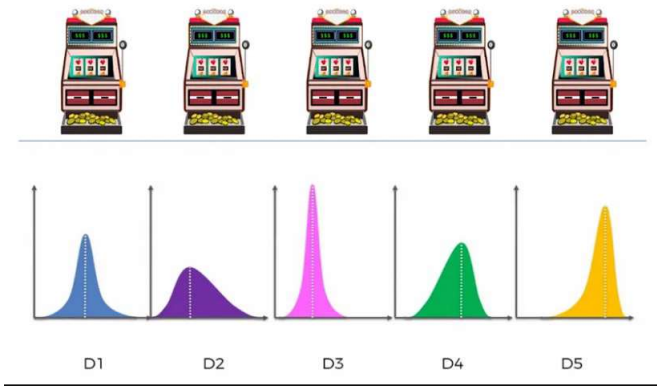
In fact, these distributions are generally interpreted as corresponding to arms on a slot machine; the player is viewed as a gambler whose goal is to collect as much money as possible by pulling these arms over many turns. At each turn, $t = 1, 2, ...$, the player selects an arm, with index $j(t)$, and receives a reward $r(t) \approx D_j(t)$. The player has a two-fold goal: on one hand, finding out which distribution has the highest expected value; on the other hand, gaining as much rewards as possible while playing. Bandit algorithms specify a strategy by which the player should choose an arm $j(t)$ at each turn. The figure 10 represents this idea. Clinical trials, on-line advertising and dynamic pricing is some examples of areas where problems can be fall in MAB.

Some well-known MAB algorithms are $\epsilon - greedy$ and Upper Condence Bounds (UCB), each one deals with the trade-offs between exploration and exploitation in a clever way and is described in the next subsections.

2.3.1 Regret in Multi-Armed Bandits

While the intuitive goal of a gambler pulling slot machine arms is to maximize cumulative reward, the theoretical performance of Multi-Armed Bandit (MAB) algorithms is often quantified by a metric called **regret**. Regret measures the difference between the total reward accumulated by the MAB algorithm over a series of T rounds and the reward that would have been obtained

Figure 10 – Example of customer churn dataset



Source: (AUSTIN, 2022)

if the algorithm had *always* chosen the optimal arm (the one with the highest true expected reward) from the very beginning. This concept is fundamental to understanding the efficiency of MAB strategies in balancing exploration and exploitation (SLIVKINS et al., 2019; AUER; CESA-BIANCHI; FISCHER, 2002).

Formally, if μ^* is the expected reward of the optimal arm and $\mu_{j(t)}$ is the expected reward of the arm chosen at round t, the cumulative regret $R(T)$ after T rounds is defined as:

$$R(T) = T\mu^* - \sum_{t=1}^{T} \mu_{j(t)}$$

The objective of a well-designed MAB algorithm is to minimize this regret, ideally achieving sub-linear regret (e.g., logarithmic regret) with respect to T. This indicates that the algorithm quickly learns to identify and exploit the best arms, with the cost of exploration diminishing over time. Algorithms like UCB are designed to achieve such optimal regret bounds (AUER; CESA-BIANCHI; FISCHER, 2002). Epsilon-Greedy, while simpler, typically exhibits linear regret in the worst case, as it continues to explore sub-optimal arms with a fixed probability (VERMOREL; MOHRI, 2005). Understanding regret provides a more formal lens through which to evaluate the efficiency of MAB strategies in balancing exploration and exploitation.

2.3.2 Exploration-Exploitation in Feature Selection

The core challenge in feature selection, is inherently analogous to the classic **exploration-exploitation dilemma** central to Multi-Armed Bandit problems. In the context of identifying an optimal feature subset, this dilemma manifests as follows:

- **Exploration:** This refers to the necessity of trying out new, untested individual features or

novel combinations of features. It involves venturing into unknown parts of the feature space, potentially including features that have not yet been evaluated, or re-evaluating features that were previously considered less important or even excluded. The primary goal of exploration is to discover potentially better feature subsets that could lead to higher model performance, even if their immediate impact is uncertain. Without sufficient exploration, the algorithm might converge prematurely to a sub-optimal feature set.

- **Exploitation:** This involves leveraging the knowledge gained from past evaluations. It means prioritizing and retaining features or feature combinations that have already demonstrated good performance (i.e., yielded high rewards, such as improved classifier accuracy). The objective of exploitation is to maximize the current performance based on the best-known feature subset. However, an over-reliance on exploitation risks missing out on truly optimal feature combinations that have not yet been sufficiently explored.

Multi-Armed Bandit algorithms provide an framework to dynamically manage this trade-off. By treating each potential action (e.g., including a specific feature, excluding a specific feature) as an "arm," MAB algorithms can intelligently allocate "pulls" (evaluations) to balance the need to discover new, effective features (exploration) with the desire to capitalize on already successful feature configurations (exploitation). This adaptive learning process is particularly advantageous in high-dimensional feature spaces where exhaustive search is computationally prohibitive and traditional heuristic methods may struggle to find global optima, as they often lack a principled mechanism for balancing these two critical aspects of search.

2.3.3 Epsilon-Greedy

The Epsilon-greedy algorithm is widely used because it is very simple, and has obvious generalizations for sequential decision problems. At each round $t = 1, 2, ..$, we generate a random number. If the generated number is greater than ϵ (epsilon), then we exploit the arm with the highest arm average until that time stamp. Otherwise, we explore a new arm(which could be the greedy arm) (algorithm 2).

Algorithm 2 Arm Selection in ϵ-Greedy

```
Generate a random number l ∈ [0, 1]
if l < ε then
    Explore: Choose an arm uniformly at random
else
    Exploit: Choose the arm with the highest average reward so far
end if
```

In an earlier empirical study,(VERMOREL; MOHRI, 2005) did not find any practical advantage to using these methods. Therefore, in our experiments, we will only consider fixed values of epsilon.

2.3.4 Upper Condence Bounds (UCB)

The UCB family of algorithms has been proposed by (AUER; CESA-BIANCHI; FISCHER, 2002), the simplest algorithm, UCB1, maintains the number of times that each arm has been played, denoted by $n_t(a)$, and the average reward $\mu_t(a)$ of an arm to choose a arm following the strategy in the algorithm 3.

Algorithm 3 UCB1

Try each arm a once
for each round t **do**
Pick arm a^* such that $a^* = \text{argmax}_a \left(\mu_t(a) + \sqrt{\frac{2\ln(t)}{n_t(a)}} \right)$
end for

The second term in the expression represent the *confidence term*. Each time a is selected, the uncertainty is presumably reduced, $n_t(a)$ increments, and, as it appears in the denominator, the uncertainty term decreases. On the other hand, each time an arm other than a is selected, t increases, but $n_t(a)$ does not; because t appears in the numerator, the uncertainty estimate increases. The use of the natural logarithm means that the increases get smaller over time; all actions will eventually be selected, but actions with lower value estimates, or that have already been selected frequently, will be selected with decreasing frequency over time. The summands in the UCB represent exploitation and exploration, respectively, and summing them up is a natural way to trade off the two.

2.4 Related work

To evaluate works in the research field, a systematic mapping following the Kitchenham protocol (KITCHENHAM, 2004) was provided. The systematic review provide insights into MAB and Features Selection (FS) in Machine Learning. Where MAB is currently used in FS, as which methods, application fields, and datasets, FS is more needed is presented following.

A Systematic mapping (SM) is a secondary study that aims to identify and classify studies related to a broad research topic (KEELE et al., 2007). Given a research topic, an SR aims to identify and highlight available research on that topic. To achieve this, this work follows Kitchenham protocol, divided in three phases: Planning the Review, Conducting the Review and Reporting the Review. In the next section we presented each one.

The objective of our systematic mapping seeks to delve into the existing literature to comprehensively assess the efficacy and applicability of multi-armed bandit algorithms in the context of feature selection. In this phase we set inclusion and exclusion criteria, define the research questions or objectives, and create a detailed protocol outlining the step-by-step procedures for conducting the review. Planning also encompasses the identification of relevant databases, search terms, and data extraction methods.

2.4.1 Research Questions

- How Multi-armed bandit currently is associated with the Machine Learning field?
- Multi-armed bandit can be applied for Feature Selection in Machine Learning?

A search string was generated to begin exploring the information within a database or online platform - ACM Digital Library, ScienceDirect and SpringerLink - the terms used were chosen iteratively based on feedback on the results, the table 4 shows the string.

Table 4 – The generic search string

(“Meta-learning” OR Metalearning) AND (“Multi-Armed-Bandit” OR “contextual-bandit”) AND (Dataset OR Metafeatures OR DATA) AND (Select OR SELECTION)

Source: Produced by authors

The search was performed during May and June 2023, and a total of 317 papers returned as a result of the execution of the search string (table 5) .

Table 5 – Number of papers returned by each database

Database	Number of papers
ACM Digital Library	59
ScienceDirect	46
SpringerLink	212

Source: Produced by authors

With the aim of selecting articles within the study area capable of answering the research questions. The selection criteria presented in table 6 were taken into account.

Table 6 – Selection Criteria

The study discusses an initiative to use multi-armed bandits in data selection, meta-learning or machine learning.

Source: Produced by authors

The following exclusion criteria were adopted to remove works from the analysis that would not contribute to answering the research questions:

- The study is an older version of another study already considered;
- The study is not a primary study;

- Unable to access the study;
- Unable to access the study;
- The study is not related to the use of Multi-Armed Bandit in the context of data selection.

In order to get a literature overview of the research field in consideration, let us answer the questions.

2.4.2 How Multi-armed bandit currently is associated with the Machine Learning field?

Multi-armed bandit (MAB) solves exploration-exploitation problems and problems like that can be found in recommender systems (CAÑAMARES; REDONDO; CASTELLS, 2019), automatic selection (SHANG et al., 2019), Active Search (ZHU; COLES; XIE, 2020) and feature selection (LIU et al., 2021).

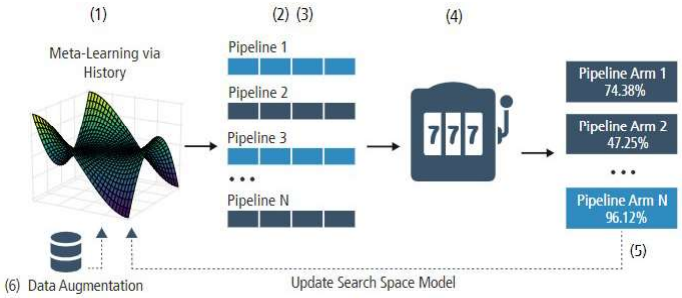
Table 7 – MAB use in machine learning

Area	**Short Definition**
Recommender Systems	Software that provides suggestions to users based on preferences, historical data, and often considering factors like similarity to other users or items. These systems aim to predict user interests to recommend suitable recommendations.
Automatic Selection	The process by which an algorithm or system automatically chooses the best option, configuration, or component from a set of possibilities, often based on predefined criteria or learned patterns, without direct human intervention at each step of the decision. This can apply to models, features, hyperparameters, etc.
Active Search	Model-agnostic and focuses on finding as many positive samples as possible within a limited labeling budget. These positives can then be used to train various models or build knowledge bases.
Feature Selection	The process of selecting a subset of relevant features (variables, predictors) from a larger set of available features for use in model construction. The goal is to improve model performance, reduce overfitting, enhance interpretability, and decrease computational cost by removing irrelevant or redundant features.

To automatically select a pipeline in a Machine Learning system, an implementation of MAB was applied in (SHANG et al., 2019). In this work an AutoML system is proposed, the system interacts with the user about an ML task and automatically selects the best pipeline for the task, including all necessary data cleaning, feature engineering, and hyperparameter tuning

steps. The core idea is that each arms corresponds to a different ML pipeline configuration. The problem is set as a sequential decision-making process. The system progressively allocates computational resources (like training on more data or evaluating different hyperparameters) to explore various pipelines, learning from the performance (e.g., accuracy or error rate) observed on a sample of data for each. This balances the "exploration" of new, potentially better pipeline configurations with the "exploitation" of pipelines that have already shown promising results. The figure 11 shows this process.

Figure 11 – Types of Machine Learning



Source: (SHANG et al., 2019)

In Natural Language Processing (NLP), relation extraction classifies a pair of words/phrases in a sentence into one of the multiple relations (negative or positive). Positive relations are called positive instances, positive data are informative for model training and yet are much rarer and more difficult to find. Active Search (AS) algorithms aims to find as many positives as possible that potentially can be used to train a model, construct a knowledge base, or for other downstream tasks. An implementation of UCB Multi-armed bandit (AUER; CESA-BIANCHI; FISCHER, 2002), was used in (ZHU; COLES; XIE, 2020) as a novel AS algorithm, exploring the dilemma exploration-explotation.

At least one result of using MAB for feature selection was found. A little explored area of research. In (LIU et al., 2021) a combinatorial multi-armed bandit (CMAB) task, includes arms as features and a super arm as a set of arms in the search process for selecting features. Iteratively generating new subsets and improving their quality.

2.4.3 Multi-armed bandit can be applied for Feature Selection in Machine Learning?

The nature of MAB suggests its ability to deal with selection and searches as described in the previous subsection. There are several heuristic search techniques in Feature Selection and these techniques try to find the optimal solution by enhancing the solution at each step based on some predefined heuristic function or the cost measure(DHAL; AZAD, 2022).

A kind of combinatorial multi-armed bandit (CMAB) task is provided in (LIU et al., 2021). But the paper explicitly states that "Traditional MAB methods cannot be directly applied in this task by simply assigning one arm for each feature" because MAB can only pull one arm at a time. It states arms as features and a super arm as a set of arms and mechanisms (generative and ranking oracles) are proposed to leads how the "super arms" are build. It moves beyond the basic MAB paradigm to handle the selection of features. Although solves the problem, it is a complex solution while a pure MAB-based algorithm can achieve great results presented in this work.

The utilization of Multi-Armed Bandit (MAB) algorithms for feature selection remains a relatively underexplored area within the realm of machine learning. While MAB algorithms have gained prominence in addressing exploration-exploitation trade-offs in reinforcement learning and online recommendation systems, their application to feature selection is a domain that has yet to be extensively investigated. The innovative idea of adapting MAB frameworks to the feature selection process holds promise in dynamically and adaptively choosing relevant features during the model training phase. By treating each feature as an "arm" and allowing the algorithm to intelligently allocate exploration and exploitation efforts, MABs have the potential to enhance the efficiency of feature selection tasks. The exploration of MABs in feature selection offers an intriguing avenue for research, with the prospect of improving model interpretability and generalization in diverse machine learning applications.

2.4.4 Feature Selection Related Work

In (WANG; JING; NIU, 2017) a Bacterial Colony Optimization (BCO) method was used -a branch within AI and soft computing, delving into a category of global optimization algorithms inspired by the principles of biological evolution - to feature selection in the classification of microarray gene expression cancer problems. Microarray gene expression have a very high dimension dataset.This work obtained good results, above 90 per cent of accuracy in some datasets. However the number of instances in each dataset was very low compared to the number of features, a *Brain Tumor* dataset used had 10.367 features and only 50 instances, this make algorithms prone to overfitting. To address this concern and ensure a more robust evaluation, a structured approach such as cross-validation should be employed to a robust evaluation.

Recently, (ALFIAN et al., 2022), proposed a combination among extra-trees and SVM technique for predicting breast cancer on the Coimbra Breast Cancer Dataset (CBCD) (PATRCIO; CARAMELO, 2018). Extra-trees algorithm is one of the examples of an embedded method to extract relevant features. Applying the Extra-trees algorithm turn SVM competitive in the CBCD classification task. However the comparison with others machine learning models was with different evaluation techniques.

Support Vector Machine - Recursive Feature Elimination (SVM-RFE) is a feature selection method that combines the principles of Support Vector Machines (SVM) with a recursive elimination process to identify the most informative features in a dataset (GUYON

et al., 2002). Recursive Feature Elimination (RFE) is a feature selection technique aiming to identify the optimal subset of features by leveraging the learned model and classification accuracy. In the conventional RFE approach, features are systematically eliminated one by one, starting with the least informative feature that leads to a reduction in "classification accuracy" when integrated into a classification model. This approach can be applied to other classification models such as random forests (RFs) and gradient boosting machines (GBMs), both of which have in-built feature evaluation mechanisms.In (JEON; OH, 2020) is proposed a new feature selection method—**Hybrid-RFE**— that is an ensemble of the feature evaluation methods of SVM-RFE, RF-RFE, and GBM-RFE, combining their feature weighting functions. The Hybrid model obtained better results than SVM-RFE, RF-RFE and GBM-RFE individually. However, comparison with other methods in the literature was not made.

In the (ELSSIED; IBRAHIM; OSMAN, 2014), a filter method was applied, filter methods are used before the application of any classification. The statical approach ANOVA F-test was used as a feature selection to identify the most relevant features (attributes) for spam classification, by determining which features show significant differences between spam and non-spam classes. The objective of ANOVA is test whether the mean values of a feature differ significantly between the spam and non-spam classes. As a statical techinique it is necessary that the features are continuous values.

3

Multi-Armed bandit in Feature Selection

In chapter 2 we presented a theoretical review of Machine Learning (ML), Feature Selection (FS), and Multi-Armed bandit(MAB). About that, in ML, FS plays a crucial role in reducing data's dimensionality and enhancing performance, in the real world FS must be present in the pipeline of machine learning applications. Multi-armed bandit algorithms deal with decisions over time under uncertainty, analyzing the trade-off exploration-exploitation, and are used for some kind of selection problems. In this chapter, we propose the application of MAB techniques in novel ways to solve feature selection problems and verify their feasibility in different datasets. For that, we will adapt the main implementations for MAB, Epsilon-greedy and Upper Confidence Bound (UCB) to the feature selection goal.

3.1 How Multi-armed bandit can be applied for Feature Selection?

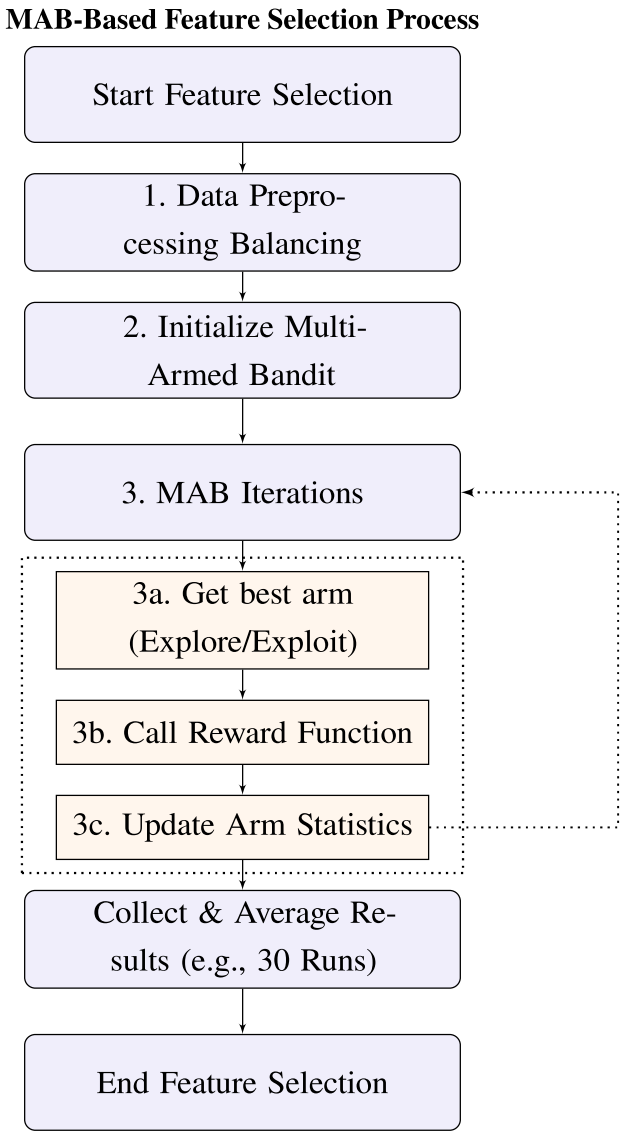
To demonstrate the use of a Multi-armed bandit for Feature Selection, was applied the *Epsilon-greedy* and Upper Condence Bounds (UCB) algorithm. Recall that the goal of MAB is to choose the right arms to maximize the reward over time (chapter 2).

In the proposed algorithms, for each feature, there is an arm representing the inclusion and another arm representing the exclusion. So, if a dataset has five features, the MAB-based algorithms will work with ten arms. During all the processing, the set of features is updated, decisions over uncertainty are made to include or exclude some feature. The decision on whether to include or not some feature is based on the reward function of an arm. If in some iteration an arm get the best reward function, and this arm represents the inclusion of a feature, the feature set is updated with this inclusion. Choosing an arm over time is the main step in MAB-proposed algorithms and each approach design how choosing, actually "how choosing an arm" means how deal with exploration-exploitation dilemma.

The reward function used by the proposed algorithms uses a SVM classifier (with a radial basis function kernel, $C = 1$, and $random_state = 1$) that is trained on the preprocessed and balanced training data, utilizing only the features currently marked as selected, the reward returned by the reward function is the accuracy score of the SVM classifier on the independent test set (hold-out 70/30) for *MAB-EgreedyFS* and the mean score of the 10-Fold CV for *MAB-UCBFS*. The Support Vector Machine (SVM) was chosen as the classifier for the reward function due to its proven robustness, strong generalization capabilities. Also and more important SVM is deterministic with fixed parameters (RBF kernel, C=1) ensuring consistency in the reward calculation across.

Diagram 3.1 presents the all the steps for a MAB-Based algorithm. In the step 1, the received dataset is normalized and the imbalanced instances are balanced. The Step 2 defines the arms for MAB, based in the dataset features. In the step 3, the box "MAB Iterations" is formed by the boxes: *3a, 3b and 3c*. Each one differs between *MAB-EgreedyFS* and *MAB-UCBFS* in implementation way. Respectively, the algorithm 5 is the "MAB Iterations" for *MAB-EgreedyFS* and the alorithm 8 for *MAB-UCBFS*. Also, the steps *3a* and *3b* are the algorithms 4 and 6 for *MAB-EgreedyFS*, while for *MAB-UCBFS* are the algorithms 7 and 9.

The algorithm iteratively updates its knowledge based on the evaluation results, continually adjusting the combinations of features. Over time, the algorithm approach ensures a balance between exploring new feature subsets and exploiting the best-performing ones. After the iterations the feature set updated represents the best combination of feature inclusions and exclusions identified through the iterative process.



3.1.1 The key to success

In the proposed algorithms, for each feature, there is an arm representing the inclusion and another arm representing the exclusion. So, if a dataset has five features, the MAB-based algorithms will work with ten arms. The key to permit implementing the "pure" MAB approach is that the same feature set is updated over iterations. This leads to the creation and experimentation of different combination of features, but a combination is formed due to the reward function of each arm by iteration. Per iteration, the selected features change in one feature with the goal to improve the accuracy of a classifier.

Figure 12 shows a representation of how the feature set is updated. The process begins by initializing the feature set as an empty collection. This means that at the very first iteration, no features are considered "selected", representing in the figure the $iteration = 0$. As the MAB algorithm proceeds through its iterations, it dynamically builds and refines this feature set. When an "inclusion" arm for a specific feature is chosen, that feature is added to the current feature

set. In the $iteration = 1$ and $iteration = 2$, two inclusion are taken, one in each iteration. Now the feature set has two features. This meaning that the classifier first was trained with one feature, and MAB decides to include one more feature in the $iteration = 2$. After being trained with two features, MAB decides to include a third feature. However, from the $iteration = 24$ to $iteration = 25$, MAB decides to remove a feature, after being trained with four in the $iteration = 24$. An "exclusion" arm for a feature was chosen. This mechanism allows the MAB to toggle the status of each feature (included/excluded) based on observed rewards, taken by a trained classifier with a feature set that is updated by iteration. Decisions made in one round influence the feature subset available for evaluation in subsequent rounds. This iterative refinement allows the MAB to learn the most effective combination of features over time, without requiring a predefined number of features. The goal is to converge on a feature set that maximizes the classifier's performance.

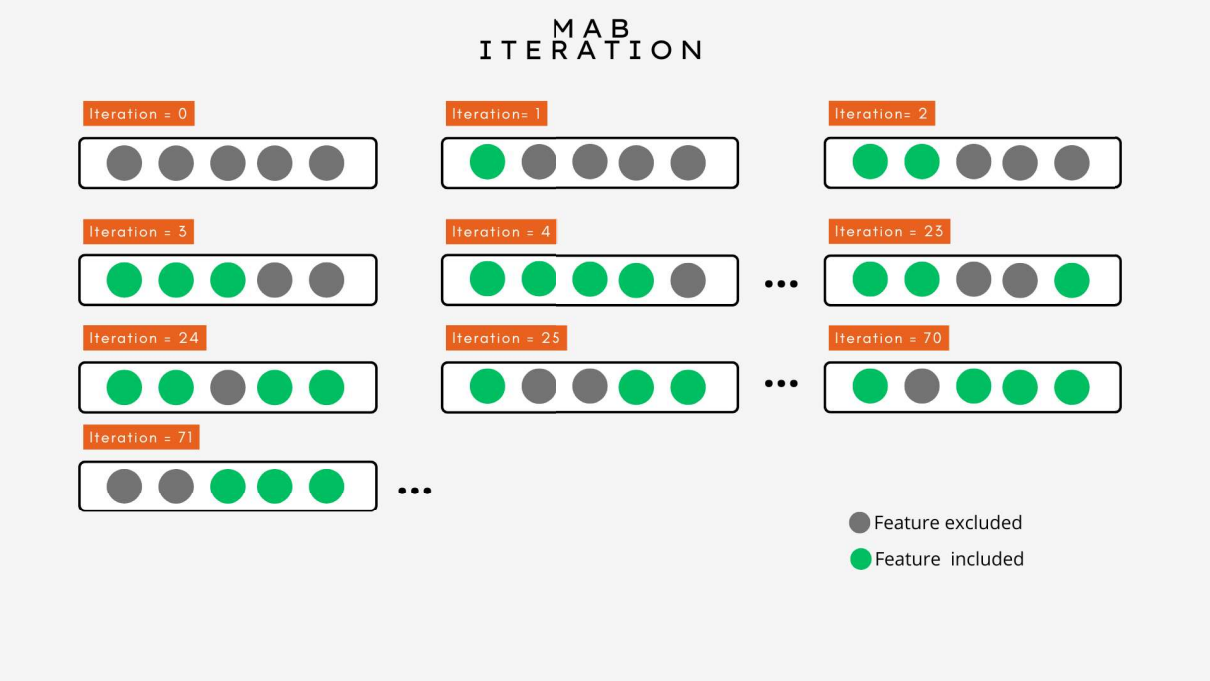


Figure 12 – Feature set updating

3.1.2 Detailed Rationale for Epsilon-Greedy and UCB

The selection of Epsilon-Greedy and Upper Confidence Bound (UCB) algorithms for adapting Multi-Armed Bandits to feature selection is rooted in their distinct yet complementary approaches to solving the exploration-exploitation dilemma. This dilemma is central to feature selection, where the algorithm must balance exploring new, potentially optimal feature combinations with exploiting those that have already demonstrated good performance.

- **Epsilon-Greedy (MAB-EgreedyFS)**: This algorithm was chosen primarily for its simplicity and ease of implementation, making it an excellent baseline for demonstrating the feasibility of a pure MAB approach to feature selection. At each iteration, Epsilon-Greedy introduces

a fixed probability (epsilon) of random exploration. This ensures that the algorithm does not get stuck in local optima by always trying new feature inclusion/exclusion actions, even if current information suggests a "best arm". The remaining $(1 - epsilon)$ probability is dedicated to exploitation, where the algorithm selects the arm (feature action) that has yielded the highest average reward so far. While straightforward, its fixed exploration rate can sometimes be a limitation, as it may explore excessively even when a near-optimal solution has been found, or insufficiently in complex search spaces. However, for an initial validation of the MAB concept in feature selection, its clear mechanism is highly valuable.

- **Upper Confidence Bound (UCB) (MAB-UCBFS)**: UCB offers a more sophisticated and principled approach to balancing exploration and exploitation compared to Epsilon-Greedy. Its selection criterion dynamically adjusts the exploration bonus based on the uncertainty associated with each arm. Arms that have been pulled fewer times, or those with higher variance in their observed rewards, receive a larger exploration bonus, making them more likely to be chosen. This "optimism in the face of uncertainty" strategy allows UCB to direct its exploration more intelligently. In the context of feature selection, UCB is particularly beneficial because the "true" performance contribution of a feature (or its inclusion/exclusion) is initially unknown. UCB systematically explores less-known feature actions while gradually converging on the most promising ones, providing a more efficient and theoretically robust search for the optimal feature subset. This dynamic balance is crucial in complex, high-dimensional feature spaces where exhaustive search is infeasible.

By implementing both Epsilon-Greedy and UCB, this research aims not only to validate the general applicability of MAB for feature selection but also to compare the effectiveness of a simpler, fixed-exploration strategy against a more adaptive, uncertainty-aware approach in this novel domain.

3.1.3 Applying Epsilon-Greedy Algorithm for Feature Selection

The algorithm 5 shows the *MAB-EgreedyFS* algorithm. The process begins by initializing a set with all features for a dataset. For each iteration of the MAB algorithm, a decision is made to explore or exploit. With a probability ϵ (epsilon) (set to 0.1 in this study), the algorithm explores by randomly selecting an "arm". Otherwise, it exploits by choosing the "best arm," which corresponds to the action (adding or removing a specific feature) that has yielded the highest average reward so far.

To choose the "best arm", for each individual arm is calculated the equation 3.1.

$$\text{arm\_score} = \frac{(\text{current\_arm\_avg} \times \text{times\_current\_arm\_has\_been\_chosen\_so\_far}) + \text{reward}}{\text{times\_current\_arm\_has\_been\_chosen\_so\_far} + 1.0} \tag{3.1}$$

Once an arm is chosen (add or remove specific feature), the reward function is called, this function updates the current feature subset by either including or excluding the feature associated with the chosen arm. Lastly, the arms statics are updated: average reward for chosen arm and increment pick count for chosen arm. During the iteration loop the configuration of the set of features is changed searching the best configuration.

Algorithm 4 MAB-EgreedyFS `best_arm()` method

return the index of the arm with the maximum average reward in arm_avg

Algorithm 5 MAB-EgreedyFS method

for i = 1 to `iterations` **do**
  Choose an Arm:
    randon_epsilon ← random uniform number between 0 and 1
  if randon_epsilon < eps **then**
    arm ← choose a random arm
  else
    arm ← best_arm()
  end if
  Call Reward Function:
    reward ← reward_func(arm, feature set)
  Update Arms Statistics:
    numerator ← (arm_avg[arm] × arm_pick[arm]) + reward
    denominator ← arm_pick[arm] + 1
    self.arm_avg[arm] ← numerator/denominator
    self.arm_pick[arm] ← arm_pick[arm] + 1
    save actual accuracy and actual feature set
end for

Algorithm 6 MAB-Eegredy `Reward` Function

function REWARD(`arm`, `feature set`)
  classifier ← SVM()
  if arm not make feature set empty **then**
    update feature set with arm
    score ← hold-out-70(classifier, feature set(arm))
    return score
  end if
end function

3.1.4 Applying Upper Condence Bounds Algorithm for Feature Selection

Based on the Upper Confidence Bound (UCB) algorithm, a reinforcement learning approach derived from the Multi-Armed Bandit (MAB) problem is developed the *MAB-UCBFS* method. In the same way of *MAB-EgreedyFS*, each "arm" in this context represents a specific

action related to a feature: either including it in the current feature subset or excluding it. The selection of an arm at each iteration is governed by the UCB formula 3.2.

$$\text{Best arm} = \arg\max_i \left(\text{arm\_avg}_i + c \times \sqrt{\frac{\ln(\text{total\_iterations})}{\text{arm\_pick}_i}} \right) \tag{3.2}$$

Where c is an exploration parameter (set to 0.2), $total\_iterations$ is the cumulative number of times any arm has been pulled, and arm_pick_i is the number of times a specific arm i has been pulled. The operator $\arg\max_i$ finds the index (i) that maximizes the expression. This formula promotes a balance between exploiting (leveraging what's known to be good) arms with high historical average rewards and exploring less-pulled arms that may possess high potential. The arm_avg_i represents the average reward obtained from pulling the arm i so far. It is a direct measure of the past performance of that arm. The higher the arm_avg_i more appealing an arm is for exploitation. Whereas the second part tends to be the exploration balance, if an arm has been pulled very few times, the denominator becomes small, making the fraction large. This results in a significant "exploration bonus" for that arm. This encourages the algorithm to try less-known options to gather more information about their true potential.

Unlike the algorithm 5, the selection of an arm at each iteration is governed by the UCB formula 3.2 and the The reward returned by the reward function is the mean score of the 10-fold CV with the SVM classifier.

Algorithm 7 `MAB-UCBFS best_arm()` method

function BEST_ARM
  Calculate for each arm i:
  $\text{Value}_i \leftarrow \text{arm\_avg}_i + c\sqrt{\frac{\ln(\text{its})}{\text{arm\_pick}_i}}$
  return the index of the arm with the maximum value
end function

Algorithm 8 `MAB-UCBFS` method

for $i = 1$ to `iterations` **do**
  Choose an Arm:
    arm ← best_arm()
  Call Reward Function:
    reward ← rewar(arm,feature set)
  Update Arms Statistics:
    num ← (arm_avg[arm] × arm_pick[arm]) + reward
    denom ← self.arm_pick[arm] + 1
    arm_avg[arm] ← num/denom
    arm_pick[arm] ← arm_pick[arm] + 1
    save actual accuracy and actual feature set
end for

Algorithm 9 MAB-UCBFS `Reward` Function

```
function REWARD(arm, feature set)
    classifier ← SVM()
    if arm not make feature set empty or make feature set with all features then
        update feature set with arm
        score ←  mean of 10-cross_validation(classifier, feature set(arm) )
        return score
    end if
end function
```

3.1.5 Comparison of MAB-EgreedyFS and MAB-UCBFS

To further highlight the differences and respective strengths of our two proposed MAB-based feature selection algorithms, MAB-EgreedyFS and MAB-UCBFS, the following table 8 provides a direct comparison of their key characteristics and operational principles.

Table 8 – Comparison of MAB-EgreedyFS and MAB-UCBFS

Feature	**MAB-EgreedyFS**	**MAB-UCBFS**
Exploration Strategy	Fixed probability (ϵ) of random exploration.	Dynamic exploration based on uncertainty (confidence bounds).
Exploitation Strategy	Selects arm with highest average reward so far.	Selects arm maximizing (average reward + exploration bonus).
Exploration-Exploitation Balance	Simple, fixed trade-off. Can be inefficient.	Principled, adaptive balance. More efficient in complex scenarios.
Computational Complexity	Generally lower due to simpler selection logic.	Slightly higher due to logarithmic calculations for confidence bounds.
Convergence	Can converge slower, or get stuck in local optima if ϵ is too small.	Tends to converge faster and more reliably to global optima.
Robustness to Initial Estimates	More sensitive to initial random choices if ϵ is small.	Less sensitive, as it actively explores uncertain arms.
Reward Function	Accuracy score from hold-out (70/30) SVM classifier.	Mean score of 10-Fold Cross-Validation with SVM classifier.
Primary Advantage	Simplicity, ease of implementation, good baseline.	More theoretically robust, efficient exploration, better long-term performance.

4

Experimental Evaluations

In chapter 3 we proposed *MAB-EgreedyFS* based on the Epsilon-greedy algorithm (VERMOREL; MOHRI, 2005) and *MAB-UCBFS* that is based on the Upper Condence Bounds (UCB), each one deals with the trade-offs between exploration and exploitation in a clever way.

In this chapter a evaluation of the MAB algorithms is presented to compare the effectiveness of feature selection, experiments were made in seven datasets presented against: extra-trees-based (ALFIAN et al., 2022), SVM-RFE (ADORADA et al., 2018), filter-based ANOVA (DHAL; AZAD, 2022) and Genetic Algorithm (GA) (BACK; FOGEL; MICHALEWICZ, 1999).

4.1 Experiments

The proposed MAB-based algorithms are evaluated against a representative algorithm of some categories. GA from **Evolutionary algorithm based**, filter-based ANOVA from **Statistical measure based** , SVM-RFE and Extra-trees from **Other methods for FS**.

The proposed MAB-based algorithms, SVM-RFE and GA are **Wrapper models**, Extra-trees-based is a **Embedded model** while ANOVA-based is a **Filter model** (see chapter 2).

The seven datasets presented in the table 9 were used in the experiments. To compare the proposed MAB methods with the principal techniques used in the literature an evaluation performance is required. The next subsections detail the datasets, algorithms implementation, metrics and statistics.

4.1.1 Datasets

The experiments are applied in three medical datasets: Breast Cancer Coimbra, Breast Cancer Wisconsin and Diabets. By the (DHAL; AZAD, 2022) medical area are the most used in

FS. The others datasets are often used to classification tasks.

To mitigate the impact of class imbalance, all the datasets were preprocessed to "under-sample" the majority class to match the cardinality of the minority class. Subsequent to balancing, all features were normalized to a scale between 0 and 1, a standard practice to ensure consistent feature contributions and optimize classifier performance.

Table 9 – Datasets information

Database	Instances	Features
Breast Cancer Coimbra (PATRCIO; CARAMELO, 2018)	116	9
Breast Cancer Wisconsin (WOLBERG WILLIAM, 1995)	569	30
Ionosphere (SIGILLITO WING, 1989)	351	34
Diabets (EARLY. . . , 2020)	400	16
Mine (SEJNOWSKI; GORMAN, 1988)	208	60
Musk (CHAPMAN; JAIN, 1994)	476	165
BIM Detector (FILHO, 2023)	1841	10

4.1.1.1 Breast Cancer Coimbra

The Coimbra dateset (PATRCIO; CARAMELO, 2018) was gathered from the Gynecology Department of the Coimbra University Hospital Center (CHUC) during the period spanning 2009 to 2013. It encompasses information from 64 women diagnosed with breast cancer and 52 healthy subjects. The dataset comprises data on **9 potential risk factors or attributes**, which were derived from routine blood analysis. All attributes(features) are **quantitative**, and a **binary dependent variable indicates the presence or absence of breast cancer**. The figure 13 shows a tabular vision of the data.

Figure 13 – Dataset Breast Cancer Coimbra

	Age	BMI	Glucose	Insulin	HOMA	Leptin	Adiponectin	Resistin	MCP.1	Classification
0	48	23.500000	70	2.707	0.467409	8.8071	9.702400	7.99585	417.114	1
1	83	20.690495	92	3.115	0.706897	8.8438	5.429285	4.06405	468.786	1
2	82	23.124670	91	4.498	1.009651	17.9393	22.432040	9.27715	554.697	1
3	68	21.367521	77	3.226	0.612725	9.8827	7.169560	12.76600	928.220	1
4	86	21.111111	92	3.549	0.805386	6.6994	4.819240	10.57635	773.920	1

Source: (PATRCIO; CARAMELO, 2018)

The figure 14 show class label balance, 104 instances were used.

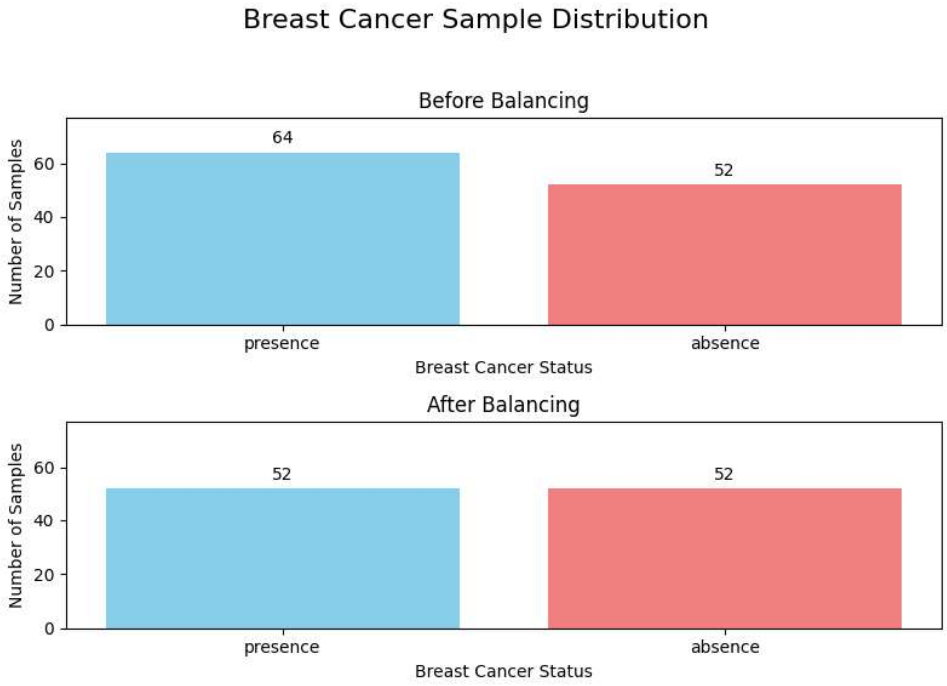


Figure 14 – Breast Cancer Coimbra class distribution

4.1.1.2 Breast Cancer Wisconsin

Breast Cancer Wisconsin dataset (WOLBERG WILLIAM, 1995) is made up of digitized images of a fine needle aspirate (FNA) of a breast mass. Characteristics of the cell nuclei present in the image are computed. Each cell nucleus is characterized by ten real-valued features: radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. Radius is the average distance from the center to points on the perimeter, while texture quantifies the standard deviation of grayscale values. Smoothness measures local variations in radius lengths, and compactness is calculated as $(perimeter^2/area) - 1.0$. Concavity reflects the severity of concave sections of the contour, with concave points indicating the number of such portions. Fractal dimension is determined by a "coastline approximation" minus one. To create a comprehensive **30-feature** dataset for each image, the mean, standard error, and "worst" (largest mean of the three largest values) of each of these ten core features are computed. For example, "Mean Radius" would be one of the features, "Radius SE" another, and "Worst Radius" a third. For each image a binary variable , classify the cell as malignant or benign. The figure 15 shows the class distribution, **424 instances**. For this dataset the instances have **real-valued features** and a **binary label class**.

4.1.1.3 Ionosphere

The Ionosphere dataset (SIGILLITO WING, 1989), originates from radar data collected by a high-frequency radar system situated in Goose Bay, Labrador. This system was operated by the Space Physics Group of The Johns Hopkins University Applied Physics Laboratory

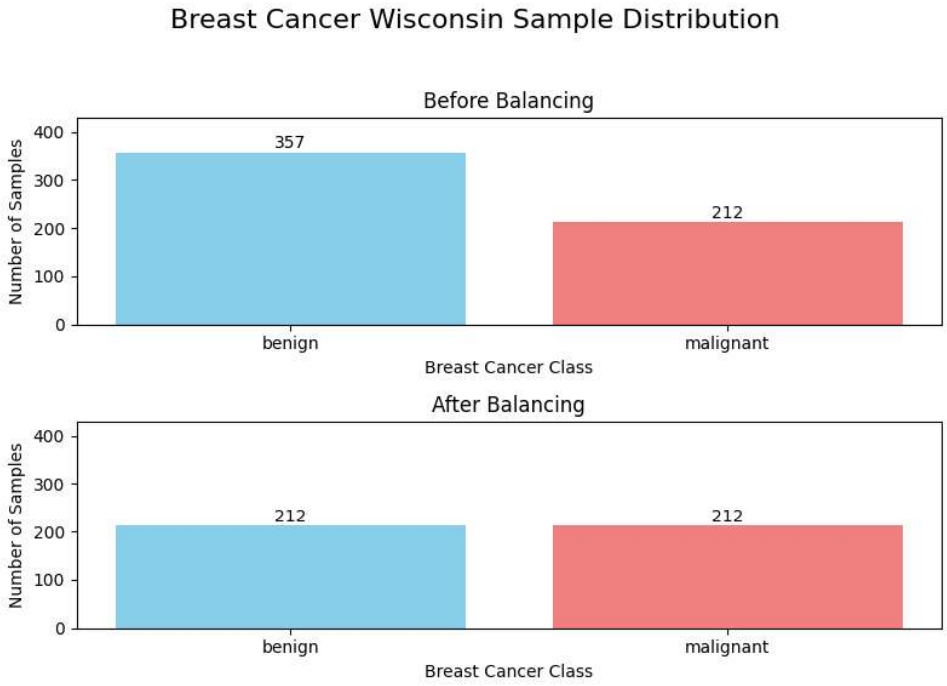


Figure 15 – Wisconsin class distribution

(JHU/APL). This system's targets were free electrons within the ionosphere. Radar returns are categorized as "good" if they indicate ionospheric structure, and "bad" if signals pass through without revealing such structure. The received signals were processed using an autocorrelation function, with inputs being the pulse time and pulse number. For the Goose Bay system, 17 pulse numbers were used, and each instance in the dataset is characterized by **34 attributes** (2 attributes per pulse number), representing the complex values derived from the electromagnetic signal's autocorrelation function.The feaure type is decimal and integer.The dataset has **351 instances** and after undersample for balancing, the distribution is presented in figure 16. The Ionosphere dataset, provides **decimal and integer** features and **categorical** binary class label -1 for "good" and 0 for "bad"- .

4.1.1.4 Diabetes

The Diabetes dataset (EARLY. . . , 2020) is a widely utilized resource in machine learning, designed to classify individuals at risk of early-stage diabetes. This dataset was meticulously compiled through direct patient questionnaires at the Sylhet Diabetes Hospital in Bangladesh, with medical approval ensuring its clinical relevance. It comprises **520 patient records**, each described by **17 attributes**, including the class label ("positive" or 'negative"). These attributes include demographic information like age and gender, alongside a comprehensive list of common diabetes symptoms such as polyuria, polydipsia, sudden weight loss, weakness, polyphagia, genital thrush, visual blurring, itching, irritability, delayed healing, partial paresis, muscle stiffness, alopecia, and obesity.

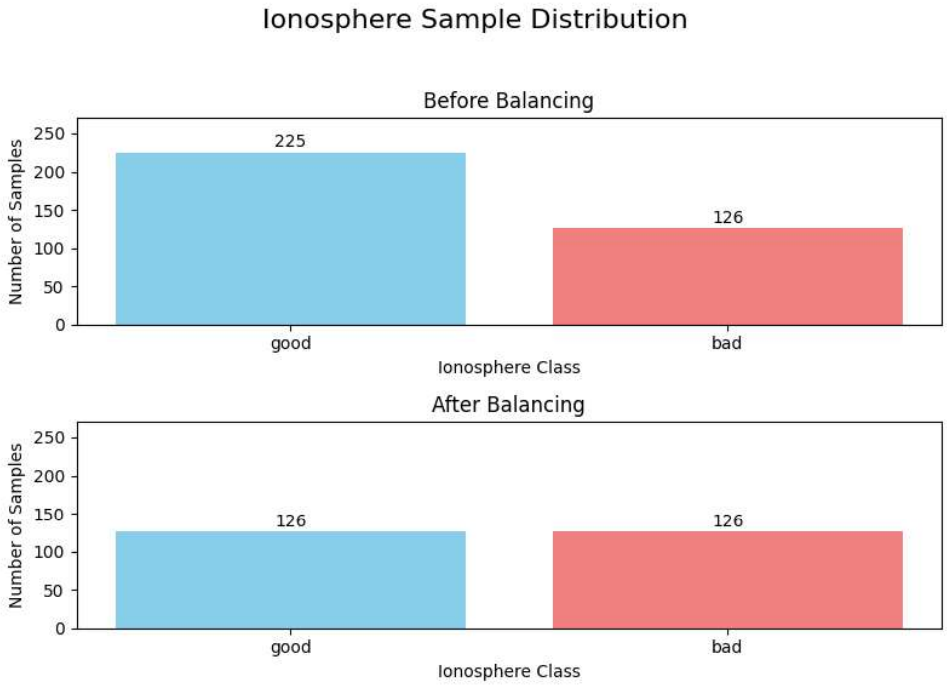


Figure 16 – Ionoephere class distribution

Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity	class
40	Male	No	Yes	No	Yes	No	No	No	Yes	No	Yes	No	Yes	Yes	Yes	Positive
58	Male	No	No	No	Yes	No	No	Yes	No	No	No	Yes	No	Yes	No	Positive
41	Male	Yes	No	No	Yes	Yes	No	No	Yes	No	Yes	No	Yes	Yes	No	Positive
45	Male	No	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No	Positive
60	Male	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Positive

Figure 17 – Diabetes Dataset

The figure 17 shows a tabular representation, the data type was transformed into **binary approach (yes= 1 and no = 0)**, figure 18 shows the class distribution.

4.1.1.5 Mine

The SONAR (Sound Navigation and Ranging) Mine Dataset (SEJNOWSKI; GORMAN, 1988) is a well-known benchmark dataset in machine learning, primarily used for the task of classifying underwater objects. It aims to distinguish between sonar signals bounced off a metal cylinder (representing a mine) and those bounced off a roughly cylindrical rock. This distinction is crucial for applications in maritime security, navigation, and underwater exploration.

The dataset was contributed to the UCI Machine Learning Repository by Terry Sejnowski (at the Salk Institute and University of California at San Diego) in collaboration with R. Paul Gorman of Allied-Signal Aerospace Technology Center. It consists of **208 instances**, each representing a sonar signal. Each instance has **60 continuous numerical features**, ranging from **0.0 to 1.0**. These features represent the energy within a particular frequency band, integrated

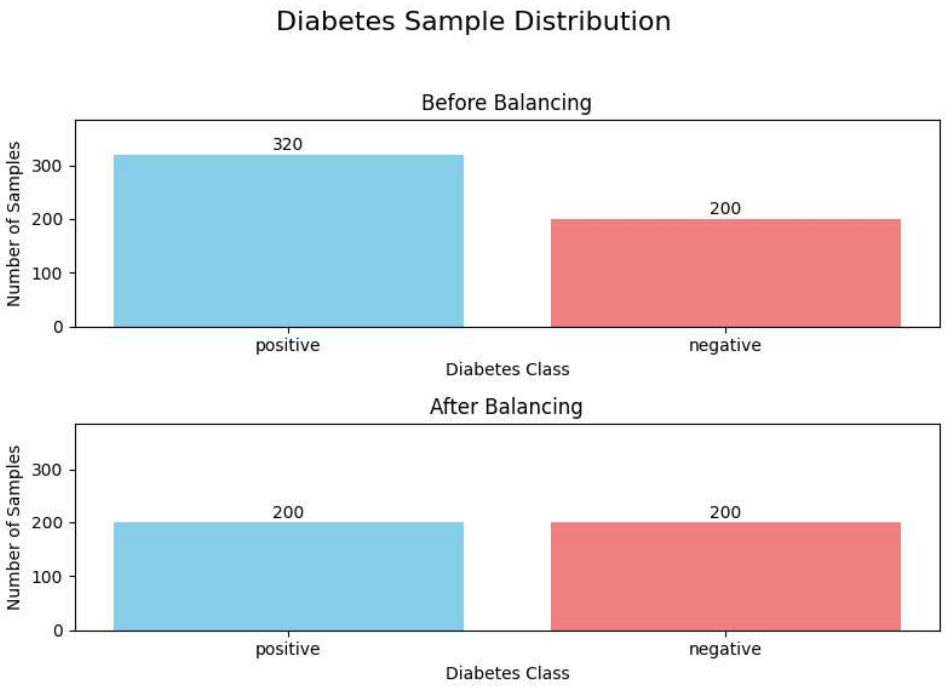


Figure 18 – Diabetes class distribution

over a certain period of time. The transmitted sonar signal is described as a frequency-modulated chirp, rising in frequency. The target variable is a categorical label: "R" for rock and "M" for mine (metal cylinder). The dataset contains 111 patterns from mines and 97 from rocks, making it a relatively balanced binary classification problem, the figure 19 shows the class distribution.

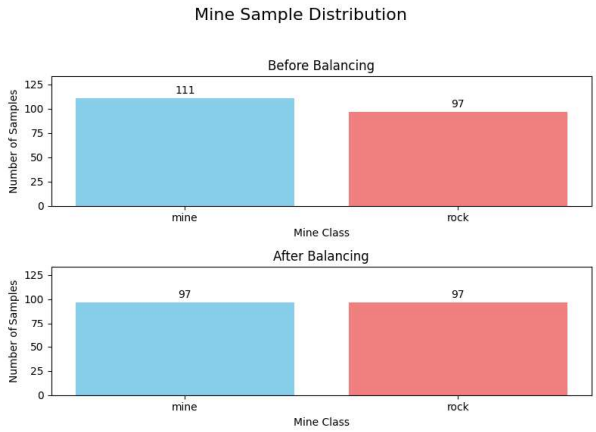


Figure 19 – Mine class distribution

From that, the Mine dataset, provides **continuous numerical** features and categorical class label transformed to binary ($R = 1$, $M = 0$).

4.1.1.6 BIM Detector

The Bim Detector dataset (FILHO, 2023) is a valuable resource to analyze and classify building information modeling (BIM) elements. The data, extracted from Revit software, consist of **1841** entries and **10** columns (9 features and the class label). It is a real dataset constructed during a research from the Computer Department in the Federal University of Sergipe (UFS). Revit, developed by Autodesk[1], is a specific software application that facilitates the BIM process. It enables civil engineers, architects, and other professionals to create and manage the digital models (the "BIMs") that contain comprehensive project data.

The dataset contains multiple class labels that represent a digital model. The features include several geometric attributes: **PosX**, **PosY**, and **PosZ** for the element's coordinates; **Area**, **Volume**, **Height**, **Width**, **Thickness**, and **Perimeter** to describe its physical dimensions. The class label identifies the type of BIM component: ***IfcDoor***, ***IfcWindow*** and ***IfcSlab***.

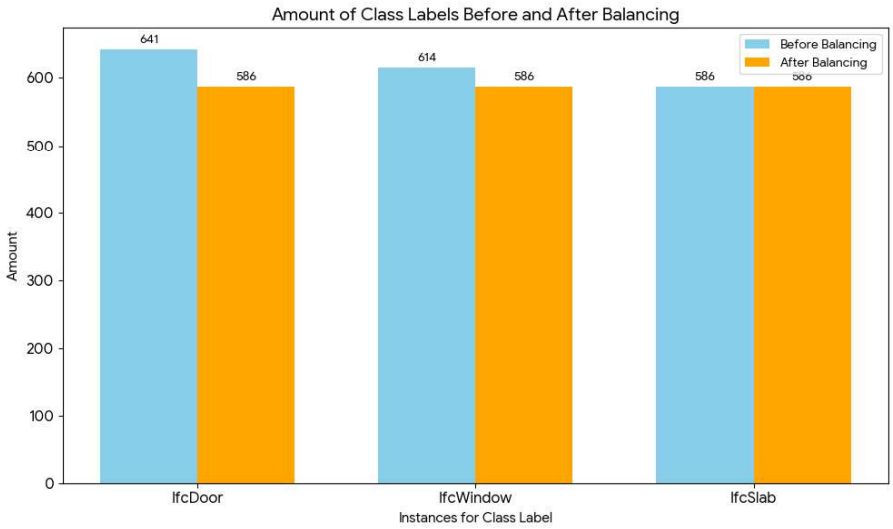


Figure 20 – Bim Detector class distribution

In the former, the dataset was not balanced and after preprocessing step the class distribution is presented in figure 20.

4.1.1.7 Musk

The Musk dataset (CHAPMAN; JAIN, 1994) describes a set of 92 molecules, categorized by human experts as either musks (47 molecules) or non-musks (45 molecules). The challenge arises because the "musky" property of a molecule isn't determined by a single fixed shape, but rather by one or more of its possible conformations (different 3D arrangements of its atoms). A molecule is labeled "musk" if at least one of its conformations exhibits muskiness, and "non-musk" if none of its conformations are musky. To capture this, the dataset provides **476**

[1] https://www.autodesk.com/

instances, where each instance represents a distinct, low-energy conformation of one of the 92 molecules. Each conformation is described by **166 continuous features**, which are "distance features" along rays, measuring aspects of the molecule's shape. Additionally, there are features related to the 3D displacement of the oxygen atom. The target variable is binary: **0 for non-musk and 1 for musk**. After preprocessing step the class distribution is presented in figure 21.

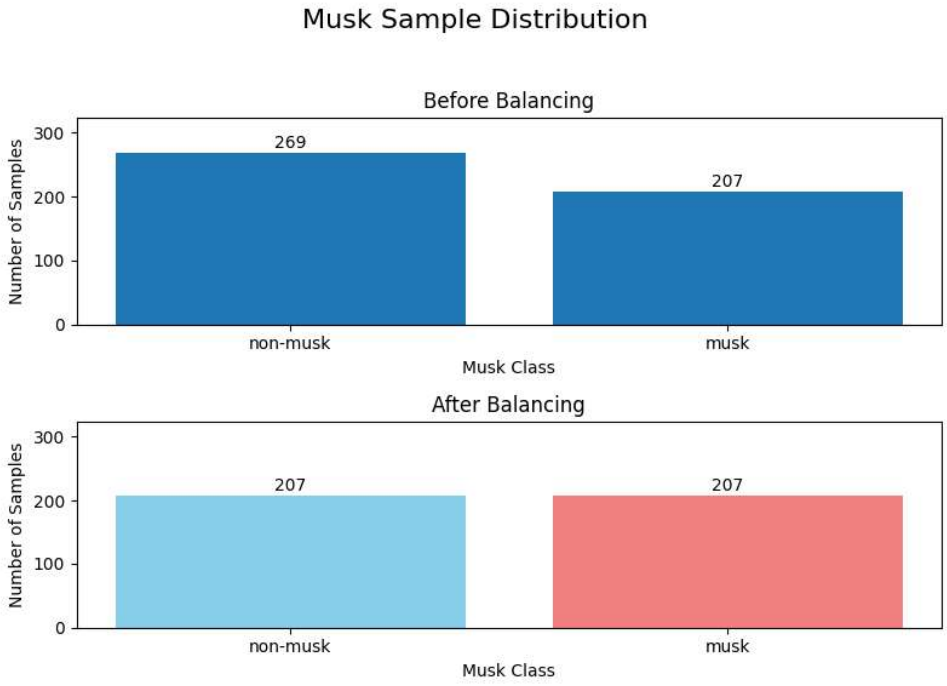


Figure 21 – Musk distribution

The table 10 presents a comparative overview of seven diverse datasets utilized in the experimental evaluation of feature selection methods.

Table 10 – Comparison of Datasets Characteristics

Dataset	**Feature Type**	**Area/Domain**	**Class Labels/Task**
Breast Cancer Coimbra	Quantitative	Medical (Breast Cancer Diagnosis)	Binary (presence or absence of breast cancer)
Breast Cancer Wisconsin	Real-valued (radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, fractal dimension, and their mean, standard error, and "worst")	Medical (Breast Cancer Diagnosis)	Binary (1 for presence or 0 for absence)
Ionosphere	Decimal and Integer (complex values from electromagnetic signal's autocorrelation function)	Scientific (Radar Data - Space Physics)	Binary (1 for "good" if indicating ionospheric structure, 0 for "bad" if signals pass through)
Diabetes	Binary (transformed from original data type, e.g., yes=1, no=0)	Medical (Early-Stage Diabetes Risk Prediction)	Binary (1 for "positive" or 0 for "negative" for diabetes risk)
Mine	Continuous numerical (ranging from 0.0 to 1.0, representing energy within a particular frequency band)	Signal Processing (Underwater Object Classification)	Binary (1 for rock and 0 for mine)
Musk	Continuous (distance features along rays, aspects of molecule's shape, 3D displacement of oxygen atom)	Chemistry/Molecular (Molecular Property Classification)	Binary (0 for non-musk, 1 for musk)
BIM Detector	Continuous numerical data, representing digital dimensions for civil engineering	Civil engineering digital data	Multi class categorical label: IfcDoor, IfcWindow and IfcSlab.

4.1.2 Algorithms Implementation

The implementation code was developed in Python[2], with the help of Anaconda distribution (ANACONDA. . . , 2020). The Anaconda is a free Python/R data science distribution that makes available AI and Machine Learning frameworks and libraries. The Extra-tree-based, filter-based ANOVA and SVM-RFE was developed using the *Scikitlearn* V1.0.2 library (PEDREGOSA et al., 2011). The Extra-trees implementation follow the same parameters in (ALFIAN et al., 2022). Scikitlearn also makes available a implementation of SVM-RFE (ADORADA et al., 2018). The GA method was developed using the DEAP Framework (FORTIN et al., 2012) following the (BACK; FOGEL; MICHALEWICZ, 1999).

[2] https://anaconda.com/app/

A good point to fix is that MAB-based and Extra-trees do not require as parameter a number of features to select(k). While SVM-RFE, GA and filter-based Anova require (this may require prior knowledge about the dataset area).

In our SVM model, used to evaluate the methods, we set the regularization parameters C = 1 and radial basis function (RBF) as kernel.

4.1.2.1 Extra-trees-based

The *Scikitlearn* library provides the *ExtraTreesClassifier* class, the term extra-trees comes from *Extremely Randomized Trees* a kind of ensemble method. Like random forests, ExtraTrees start by picking a random subset of features at each decision node to help decorrelate the individual trees in the ensemble. The key difference lies in how they find the best split point. Instead of carefully searching for the single optimal threshold that best separates the data for each chosen feature, ExtraTrees randomly draw multiple thresholds for each candidate feature. From these randomly generated options, the algorithm then simply picks the best one to use as the splitting rule for that node. Typically leads to a reduction in the model's variance, making it more robust and less prone to overfitting on the training data, in chapter 2 there is a theorical overview about machine learning models.

The relative rank (i.e. depth) of a feature used as a decision node in a tree can be used to assess the relative importance of that feature with respect to the predictability of the target variable. Features used at the top of the tree contribute to the final prediction decision of a larger fraction of the input samples.

The *ExtraTreesClassifier class* has the attribute *feature_importances_* with the ranking of the best features, this is the way how feature selection is solved using Extra-trees models.

The parameters used in the class was default (estimator number = 10), the figure 22 shows how Scikitlearn library can provides *Extremely Randomized Trees*.

```
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.datasets import make_classification

# X: instances from dataset, y: target class from dataset
X, y = make_classification(n_features=4, random_state=0)
clf = ExtraTreesClassifier(n_estimators=100, random_state=0)
# Build a forest of trees from the training set (X, y)
clf.fit(X, y)
# Get the more importants features
features_importances=extraTree.feature_importances_
indices_features = np.argsort(features_importances)[::-1]
```

Figure 22 – Extra-trees from Scikitlearn

4.1.2.2 SVM-RFE

The goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features, using a an external estimator that assigns weights to features (SVM). It works by combining the strengths of Support Vector Machines (SVMs) with a recursive elimination process. The core idea is to train an SVM model, which intrinsically assigns weights to features based on their importance in classifying the data. Features with smaller weights are considered less impactful on the model's decision-making.

The process is iterative: SVM-RFE starts by training an SVM on the entire set of features. After training, the model calculates the "importance" of each feature, typically by looking at the magnitude of its associated weight in the SVM. The least important features (those with the smallest weights) are then removed from the dataset. This reduced dataset is then used to train a new SVM, and the process repeats. This recursive elimination continues, discarding features at each step, until a desired number of features is reached or a stopping criterion is met. The features that remain at the end are considered the most important for accurate classification.

```python
from sklearn.datasets import make_friedman1
from sklearn.feature_selection import RFE
from sklearn.svm import SVR

# Generates X: instances from dataset, y: target class from dataset
X, y = make_friedman1(n_samples=50, n_features=10, random_state=0)
# Instantiate a classifier
estimator = SVC(kernel="linear")
selector = RFE(estimator, n_features_to_select=5, step=1)
# Fit the RFE model and then the underlying estimator on the selected features.
selector = selector.fit(X, y)
selector.support_
# array([ True,  True,  True,  True,  True, False, False, False, False, False])
selector.ranking_
# array([1, 1, 1, 1, 1, 6, 4, 3, 2, 5])
```

Figure 23 – SVM-RFE from Scikitlearn

The *Scikitlearn* provides the *RFE class*, the figure 23 demonstrates how, that performs recursive feature elimination using a SVM with a linear kernel . The *RFE class* receive as the parameter the number of features to select and a classifier (SVM).

4.1.2.3 Filter-based ANOVA f-test

Scikit-learn exposes feature selection routines using ANOVA F-test technique through the *SelectKBest* class. The class receive the scoring function that returns univariate scores and p-values and k features to select. ANOVA F-value is a powerful statistical tool utilized for feature selection, particularly when dealing with numerical input features and a categorical target variable (e.g., in classification problems). It quantifies the ratio of the variance between group means (the "signal" explained by the feature) to the variance within each group (the "noise" or unexplained variability).

```python
from sklearn.feature_selection import SelectKBest, f_classif
import numpy as np

def get_selected_features(X, y, k):
    # Apply SelectKBest with ANOVA F-value
    selector = SelectKBest(score_func=f_classif, k=k)
    # Run score function on (X, y) and get the appropriate features.
    selector.fit(X, y)
    # Return selected features as a boolean array
    return selector.get_support()
```

Figure 24 – Anova from Scikitlearn

The figure 24 shows a function that receives **X** (the instances from dataset), **y** (the target class from dataset) and the k features to select.

4.1.2.4 Genetic Algorithm

The DEAP Framework provides very common evolutionary algorithms. The experiments used the *eaSimple class* to implement a genetic algorithm to be used for feature selection. The core idea of genetic algorithm is to evolve a population of potential feature subsets (represented as binary strings where ‘1‘ indicates a selected feature and ‘0‘ indicates an unselected feature) over successive generations. Each individual in this population is evaluated by its "fitness," which is determined by the mean cross-validation accuracy of an SVM classifier trained exclusively on the features it represents. The algorithm then uses principles inspired by natural selection: individuals with higher fitness (better accuracy) are more likely to be selected to "reproduce" (via crossover) and "mutate" (randomly flip feature selections) to create the next generation. This iterative process drives the population towards individuals that represent better performing feature combinations.

The key parameters governing the genetic algorithm’s behavior are:

- **Population Size**: Set to 50, this parameter defines the number of individuals (feature subsets) in each generation of the genetic algorithm. A larger population might explore the search space more thoroughly but requires more computational resources.
- **Number of Generations**: Set to 30, this specifies how many iterations the genetic algorithm will run. More generations allow for greater evolution and refinement of the feature subsets, potentially leading to better solutions, but also increase computation time.
- **Crossover Probability**: Set to 0.8, this is the probability that two selected individuals will exchange genetic material (features) to create offspring. A high crossover probability encourages exploration of new feature combinations.
- **Mutation Probability**: Set to 0.2, this is the probability that an individual feature in a newly created offspring will be randomly flipped (e.g., a selected feature becomes unselected,

or vice-versa). Mutation introduces diversity into the population, preventing premature convergence to local optima.

- **Number of Selected Features(k)**: This parameter is external to the DEAP genetic algorithm setup but crucial to the experiment. The evaluate function penalizes individuals if the number of selected features is not exactly k.

Theses parameters and the fitness function (evaluate) algorithm 10 are passed to the *eaSimple class*.

Algorithm 10 Evaluation function

function EVALUATE($individual$, X_{norm}, y, k)
    $selected\_features \leftarrow$ boolean mask from $individual$
    if sum($selected\_features$) $\neq k$ **then**
        $fitness \leftarrow$ 0.0 (penalty for incorrect feature count)
    else
        Initialize SVM classifier clf (RBF kernel, C=1)
        $scores \leftarrow$ 5-fold cross-validation accuracy of clf on ($X_{selected\_features}$, y)
        $fitness \leftarrow$ mean of $scores$
    end if
    return $fitness$
end function

4.1.2.5 MAB-EgreedyFS and MAB-UCBFS

The proposed *MAB-EgreedyFS* and *MAB-UCBFS* algorithms were implemented in Python, leveraging standard data manipulation and machine learning libraries available within the **Anaconda distribution**. The core logic for both algorithms directly follows the detailed procedures outlined in chapter 3, specifically algorithms 5, 4, and 6 for MAB-EgreedyFS, and algorithms 8, 7, and 9 for MAB-UCBFS. The SVM classifier (with a radial basis function kernel and C=1) used within the reward functions of both MAB approaches was sourced from the Scikit-learn library, ensuring consistency with the evaluation classifier. No external or specialized MAB libraries were used; the algorithms were built from scratch to precisely implement the exploration-exploitation strategies as described. This allowed for fine-grained control over the "arm" selection and reward calculation mechanisms tailored for feature selection.

4.1.3 Evaluation Metrics and Statics

To rigorously evaluate and compare the performance of the proposed MAB-based feature selection algorithms against the others, a comprehensive evaluation strategy was employed. For each dataset in table 9 and for each feature selection method, the following metrics and statistical tests were applied:

- **Cross-Validation (CV) Score**: The primary metric for evaluating the classification performance was the accuracy score obtained from a 10-fold stratified cross-validation with an SVM classifier (performed 30 times). The *Scikitlearn* library provides the method cross_val_score to execute a cross-validation in several ways. Stratified cross-validation was chosen to ensure that each fold maintained the same proportion of class labels as the original dataset, which is crucial for balanced evaluation. For each dataset, the cross-validation scores were plotted against the number of selected features k (e.g., figure 33).

- **Conover Test**: To determine if the observed differences in accuracy between the feature selection methods were statistically significant, a Conover test was applied. This non-parametric post-hoc test is suitable for comparing multiple groups after a significant result from a Kruskal-Wallis H-test (which is an omnibus test for comparing more than two independent samples). The results of the Conover test are visualized as heatmaps (e.g., figure 26 , figure 36), where cell colors indicate the level of statistical significance (NS for not significant, and various $p - value$ thresholds: $< 0.05, < 0.01, < 0.001$). These heatmaps are symmetric with white diagonals, and they highlight which pairs of methods exhibit statistically significant differences in performance. The "ALL" category in the heatmaps represents the baseline accuracy when all features are used, providing a reference for the dimensionality reduction benefits.

4.1.4 Evaluation by Dataset

4.1.4.1 Feature Selection on Breast Cancer Coimbra

Table 11 displays the ranking based on cross-validation . Extra-Trees outperformed the other methods for the Coimbra dataset with a score of 0.8420 . This dataset has the smallest number of instances among those evaluated, with only **106 samples**. A smaller number of instances can lead to a lack of generalization in machine learning models. Since the reward function used in wrapper models (GA, MAB-based, SVM-RFE) requires a classifier, the Extra-Trees method, as an embedded model and being an extremely randomized algorithm, may have led it an advantage.

Rank	Method	Mean CV 30x Accuracy
1	Extra-Trees	0.842091
2	GA	0.807485
3	MAB-UCBFS	0.806242
4	SVM-RFE	0.798333
5	ANOVA Filter	0.769697
6	MAB-EgreedyFS	0.763758

Table 11 – Top FS method for Breast Cancer Coimbra dataset

The figure 25 shows the performance for a range of selected features. Let n be the total number of features and k be the number of selected features, the graph presents $2 \leq k < n$. The Extra-trees got the best accuracy with selected three features, $k = 3$. After $k = 3$ all the methods looks trace together.

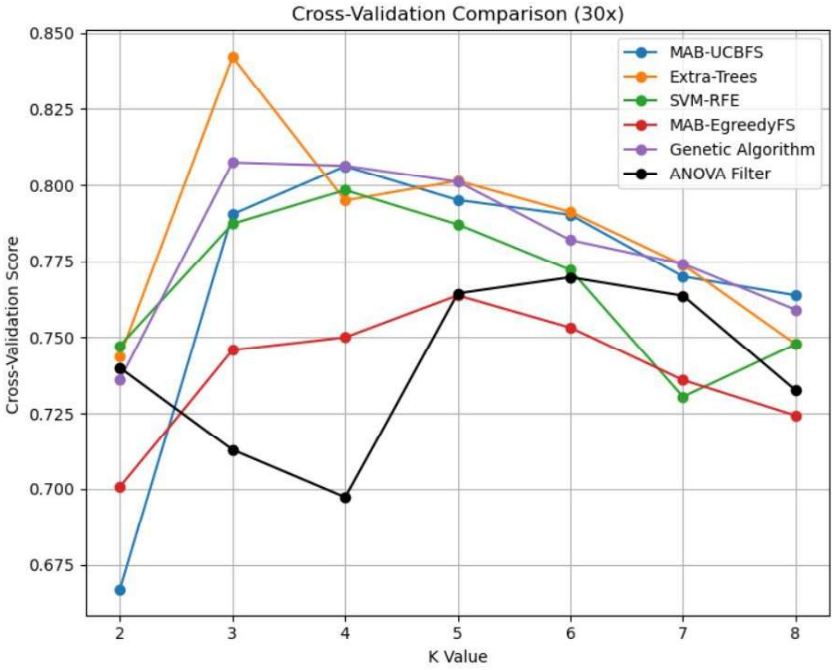


Figure 25 – Performance over k

By the heatmap 13 , Extra-trees (ET) is the most different method, stating its statistical significance.

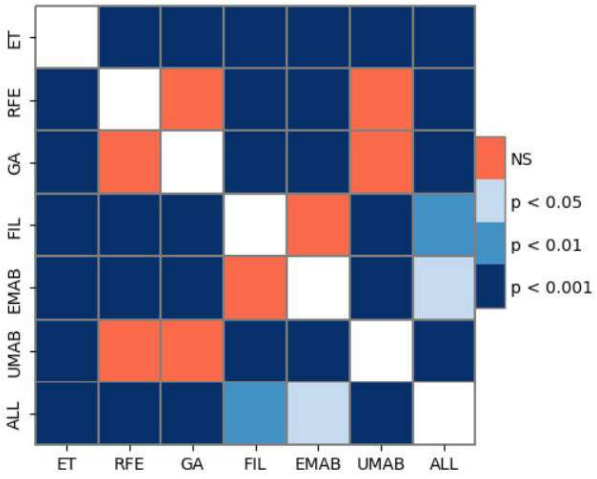


Figure 26 – HeatMap Conover test Coimbra dataset

4.1.4.2 Feature Selection on Diabetes

For the Diabetes dataset, table 12 shows that *MAB-UCBFS* achieved the highest maximum cross-validation score of 0.9675, ranking first among all methods. Extra-Trees and SVM-RFE

followed closely, demonstrating strong performances as well. *MAB-EgreedyFS* and GA performed similarly, both achieving a score of 0.9507. Recovering the Diabetes dataset has 400 instances more than Coimbra dataset all the methods performs better.

Rank	Method	Mean CV 30x Accuracy
1	MAB-UCBFS	0.967500
2	Extra-Trees	0.958917
3	SVM-RFE	0.957083
4	ANOVA Filter	0.954417
5	MAB-EgreedyFS	0.950667
5	GA	0.950667

Table 12 – Top FS method for Diabetes dataset

The MAB-UCBFS selected only 9 features (**Age, Gender, Polyuria, Polydipsia, Genital thrush, Itching, Irritability, delayed healing, Alopecia**) , achieved a slightly higher accuracy than the Extra-Trees model, which used 12 features (**Polyuria, Polydipsia, Gender, sudden weight loss, Alopecia, Age, partial paresis, Itching, Polyphagia, delayed healing, Irritability, visual blurring**) . This suggests the MAB-UCBFS method was more effective at identifying a minimal, highly predictive subset of features, demonstrating that fewer features do not necessarily mean lower performance. Specifically, the Extra-Trees model included features like **sudden weight loss, partial paresis, polyphagia, and visual blurring** that MAB-UCBFS omitted, while MAB-UCBFS included **genital thrush**, which Extra-Trees did not. The superior accuracy of the MAB-UCB model with fewer features highlights its potential advantage in creating a simpler, more efficient, and perhaps less costly model to deploy without sacrificing predictive power. Also leads to insights about what characteristics can be more valuable for diabetes.

Figure 27, illustrating the performance over k for the Diabetes dataset, reveals that MAB-UCBFS consistently maintained high accuracy across a broad range of selected features, particularly from $k = 6$ to $k = 14$. It reached its peak performance with a relatively small number of features, indicating its efficiency in identifying the most relevant subset. The other methods also showed good performance, with Extra-Trees, SVM-RFE, and ANOVA Filter reaching competitive accuracy levels, generally stabilizing after $k = 8$.

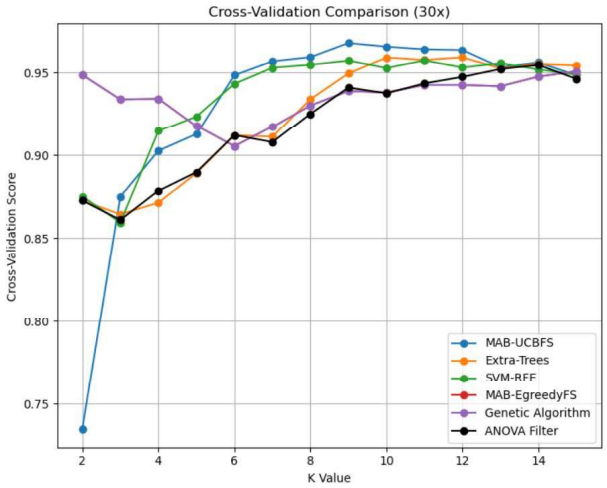


Figure 27 – Performance over k Diabetes dataset

The heatmap in figure 28 (Heatmap Diabetes) confirms the statistical significance of MAB-UCBFS's performance. It shows that MAB-UCBFS (UMAB) has a statistically significant difference ($p<0.001$) when compared to MAB-EgreedyFS (EMAB) and GA, and a significant difference ($p<0.05$) with ANOVA Filter (FIL). This indicates that MAB-UCBFS's superior performance on the Diabetes dataset is not merely by chance.

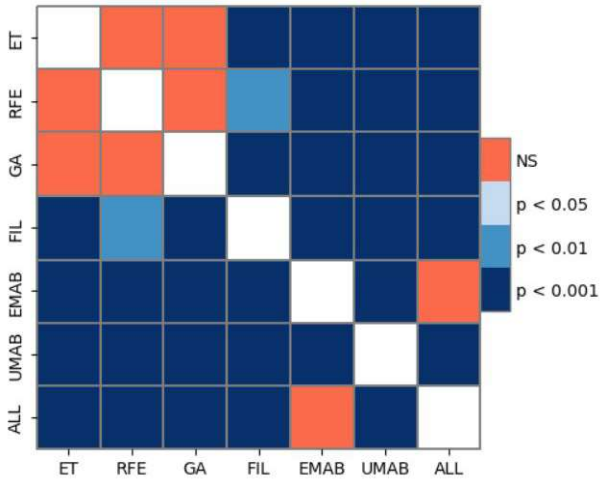


Figure 28 – Heatmap Diabetes

4.1.4.3 Feature Selection on Mine

On the Mine dataset, table 13 indicates that MAB-UCBFS again secured the top rank with a maximum cross-validation score of 0.850070. SVM-RFE was a close second at 0.848737, followed by GA, MAB-EgreedyFS, ANOVA Filter, and Extra-Trees showed slightly lower, but still competitive, performances.

The MAB-UCBFS method, using 28 features, achieved a slightly higher accuracy compared to the SVM-RFE method, which used 27 features. The primary difference lies in the specific

Rank	Method	Mean CV 30x Accuracy
1	MAB-UCBFS	0.850070
2	SVM-RFE	0.848737
3	GA	0.840553
4	MAB-EgreedyFS	0.834404
5	ANOVA Filter	0.833474
6	Extra-Trees	0.832246

Table 13 – Top FS method for Mine dataset

indices of the selected features, as the two set of selected feautures show very little overlap. For example, MAB-UCBFS included features like indices **0,9,12,15,18,22,25,28,31,42,44,45,46,50,51,53, and 54** from the original feature dataset, which SVM-RFE omitted, while SVM-RFE exclusively selected indices such as **3,4,8,11,13,21,29,30,33,34,37,38,41,43,58 and 59**. This suggests that, at this higher feature count, the MAB-UCBFS algorithm found a marginally more optimal feature combination for the given classification task, even with one more feature.

Figure 29 (Performance over k Mine dataset) demonstrates that MAB-UCBFS quickly reached a high level of accuracy and maintained it consistently as the number of features increased, showcasing its robustness. SVM-RFE also performed very well across different k values. Notably, for this dataset, the performance of most methods tended to stabilize or slightly fluctuate after around $k = 20 features$, suggesting that a significant portion of the predictive power is contained within a subset of the features.

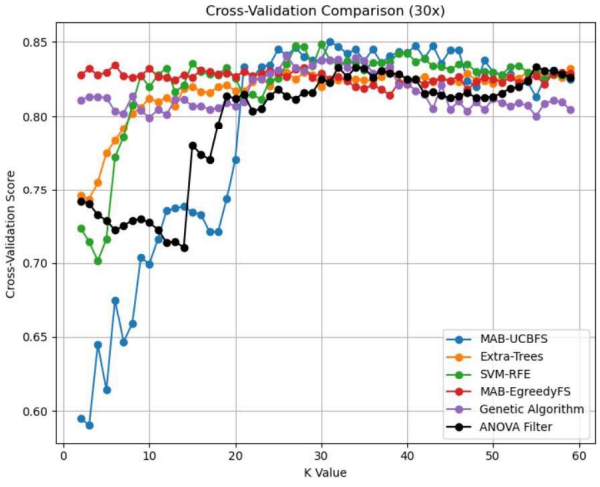


Figure 29 – Performance over k Mine dataset

The Conover test heatmap for the Mine dataset (figure 30) reveals significant differences. MAB-UCBFS (UMAB) shows a statistically significant difference ($p < 0.001$) against Extra-Trees (ET) and ANOVA Filter (FIL), and a significant difference ($p < 0.05$) against MAB-EgreedyFS (EMAB). This further supports the finding that MAB-UCBFS is a highly effective feature selection method for this type of data.

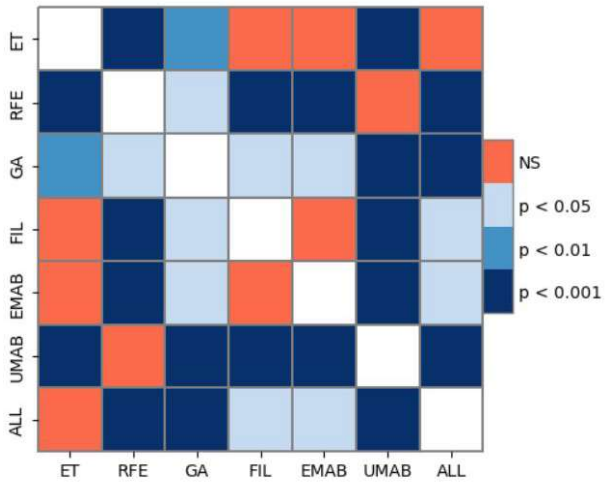


Figure 30 – Heatmap Mine dataset

4.1.4.4 Feature Selection on MUSK

For the Musk dataset, table 14 shows that SVM-RFE achieved the highest maximum cross-validation score of 0.879019, narrowly outperforming MAB-UCBFS, which came in second with **0.878223**. Extra-Trees, MAB-EgreedyFS, GA, and ANOVA Filter followed, indicating a generally high level of performance across all methods for this dataset.

Rank	Method	Mean CV 30x Accuracy
1	SVM-RFE	0.879019
2	MAB-UCBFS	0.878223
3	Extra-Trees	0.868403
4	MAB-EgreedyFS	0.867710
5	GA	0.865954
6	ANOVA Filter	0.864553

Table 14 – Top FS method for Musk dataset

Figure 14 (Performance over k Musk dataset) illustrates that most methods, including SVM-RFE and MAB-UCBFS, reached their peak performance with a moderate number of features (around $k = 60$ to $k = 80$) and maintained it as more features were added. MAB-EgreedyFS showed a slightly more erratic curve initially but stabilized at a competitive level.

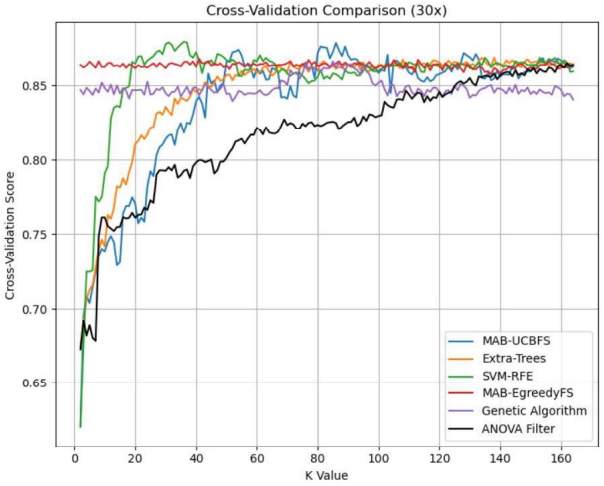


Figure 31 – Performance over k Musk dataset

The heatmap in figure 32 (Heatmap Musk dataset) indicates that while SVM-RFE and MAB-UCBFS were the top performers, the statistical differences between many of the methods were less pronounced compared to other datasets. MAB-UCBFS (UMAB) shows significant differences ($p < 0.001$) with ANOVA Filter (FIL), and ($p < 0.05$) with Extra-Trees (ET). This suggests that for the Musk dataset, several methods achieved statistically comparable high performance.

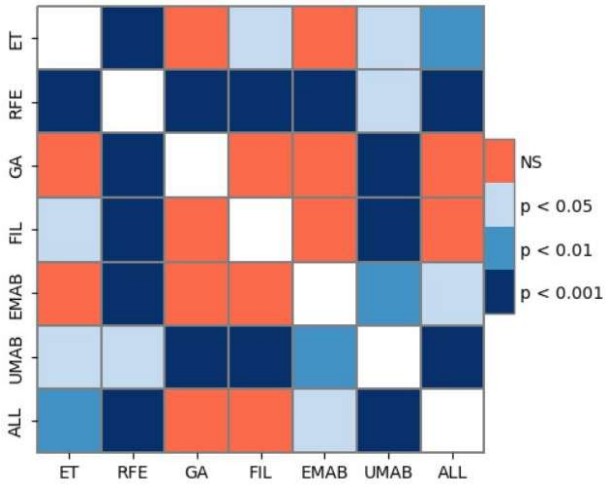


Figure 32 – Heatmap Musk dataset

4.1.4.5 Feature Selection on WINSCONSIN

Table 33 for the Breast Cancer Wisconsin dataset shows MAB-UCBFS again at the top, with a maximum cross-validation score of 0.973328. SVM-RFE was a very close second (0.970871), followed by GA, ANOVA Filter, Extra-Trees, and MAB-EgreedyFS, all demonstrating excellent performance above 0.96. This indicates that the Wisconsin dataset is highly separable, and most feature selection methods can identify effective subsets.

Rank	Method	Mean CV 30x Accuracy
1	MAB-UCBFS	0.973328
2	SVM-RFE	0.970871
3	GA	0.968987
4	ANOVA Filter	0.968774
5	Extra-Trees	0.968651
6	MAB-EgreedyFS	0.967940

Table 15 – Top FS method for Wisconsin dataset

The MAB-UCBFS method, using a much smaller set of 12 features, achieved a slightly higher accuracy than the SVM-RFE method, which required nearly twice the features (21 features) to reach an accuracy of approximately 0.9708. Specifically, MAB-UCBFS identified a highly effective core set (including indices **8,10,27** from the features dataset) that allows the classifier to generalize better with significantly less data input, while SVM-RFE's larger set included additional features (like **2,4,5,9,12,15,19,24,25,29,30,31**) that either added noise or were redundant. This outcome clearly favors the MAB-UCBFS approach for its ability to deliver higher performance with dramatically reduced complexity, making the resulting model more efficient and easier to interpret.

Figure 33 (Performance over k Wisconsin dataset) illustrates that most methods quickly achieved high accuracy with a relatively small number of features (around $k = 10 to k = 15$) and maintained this high performance as more features were included. MAB-UCBFS and SVM-RFE show very stable and high accuracy curves across the range of k values.

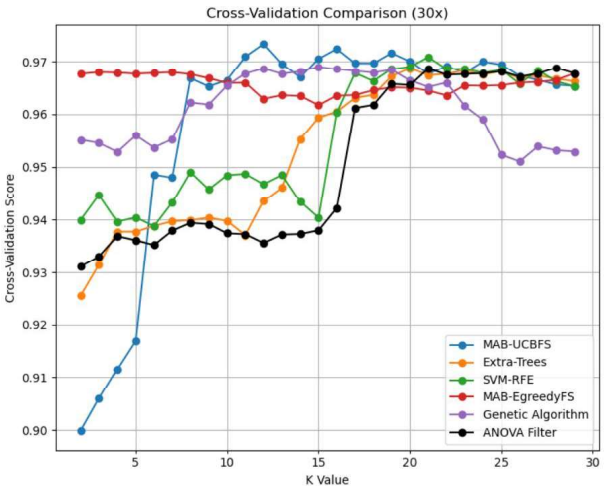


Figure 33 – Performance over k Wisconsin dataset

The Conover test heatmap for Wisconsin (figure 34) shows that MAB-UCBFS (UMAB) has statistically significant differences ($p < 0.001$) with Extra-Trees (ET), and ($p < 0.05$) with MAB-EgreedyFS (EMAB). While the overall accuracies are high for all methods, the statistical test highlights the subtle yet significant advantages of the top-performing algorithms.

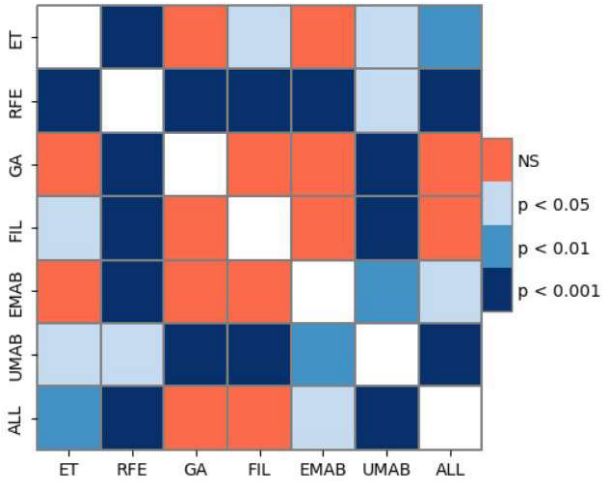


Figure 34 – Heatmap Wisconsin dataset

4.1.4.6 Feature Selection on Ionosphere

For the ionosphere dataset, Table 16 indicates that MAB-UCBFS once again achieved the highest maximum cross-validation score of 0.939785. GA came in second with **0.933667**, followed by Extra-Trees, SVM-RFE, ANOVA Filter, and MAB-EgreedyFS.

Rank	Method	Mean CV 30x Accuracy
1	MAB-UCBFS	0.939785
2	GA	0.933667
3	Extra-Trees	0.932000
4	SVM-RFE	0.929518
5	ANOVA Filter	0.928631
6	MAB-EgreedyFS	0.924426

Table 16 – Top FS method for Ionosphere dataset

Figure 35 (Performance over k Ionosphere dataset) shows that MAB-UCBFS consistently maintained a leading position in terms of accuracy across various numbers of selected features, particularly after $k = 15$. The performance curves for most methods tend to converge to a high accuracy level as more features are considered, but MAB-UCBFS often reaches this level more efficiently or maintains a slight edge.

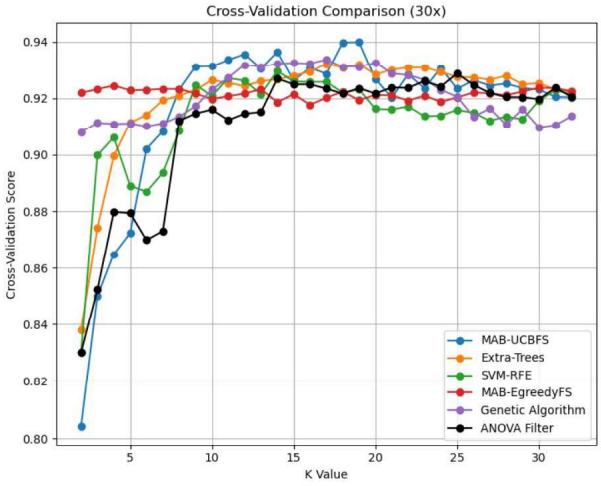


Figure 35 – Performance over k Ionosphere dataset

The Conover test heatmap for Ionosphere (figure 36) reinforces the findings, showing that MAB-UCBFS (UMAB) has a statistically significant difference ($p < 0.001$) when compared to MAB-EgreedyFS (EMAB), and significant differences ($p < 0.05$) with Extra-Trees (ET), SVM-RFE (RFE), and ANOVA Filter (FIL). This confirms that MAB-UCBFS provides a statistically superior performance on this dataset.

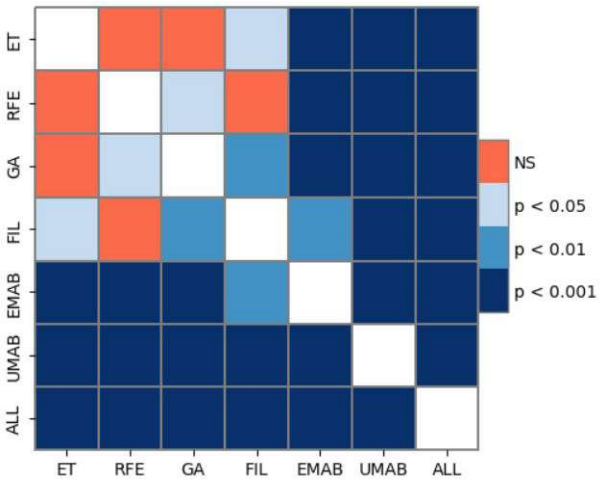


Figure 36 – Heatmap Ionosphere dataset

4.1.4.7 Feature Selection on BIM Detector

For the BIM Detector dataset table 17, Extra-trees achieved the best score, but followed very close by MAB-UCBFS. This dataset is the only one that contains a multiclass target. Identifying more than two patterns (like in binary target class) generally requires some abstractions for classification models like the **OnevsRest** strategy. Anova Filter gets third place for the first time, since its algorithm uses statistics not involving classification training,this may be an advantage.

Figure 37 shows that for k values from 4 to 8, all six methods perform almost identically, with their scores tightly clustered around 0.66. This suggests that for this range of k, the choice

Rank	Method	Mean CV 30x Accuracy
1	Extra-Trees	0.962457
2	MAB-UCBFS	0.961812
3	ANOVA Filter	0.952219
4	GA	0.800871
5	MAB-EgreedyFS	0.696572
6	SVM-RFE	0.669968

Table 17 – Top FS method for BIM Detector dataset

of method has little impact on the cross-validation score.

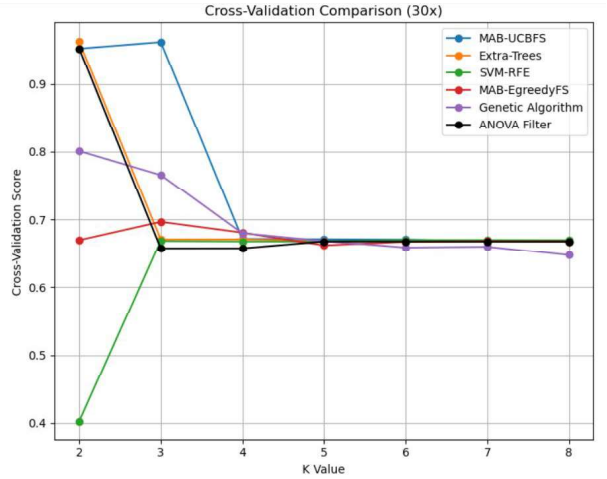


Figure 37 – Performance over k BIM Detector dataset

The figure 38 suggests that while the MAB-based approaches are statistically similar to each other, many of the other methods have unique performance characteristics.

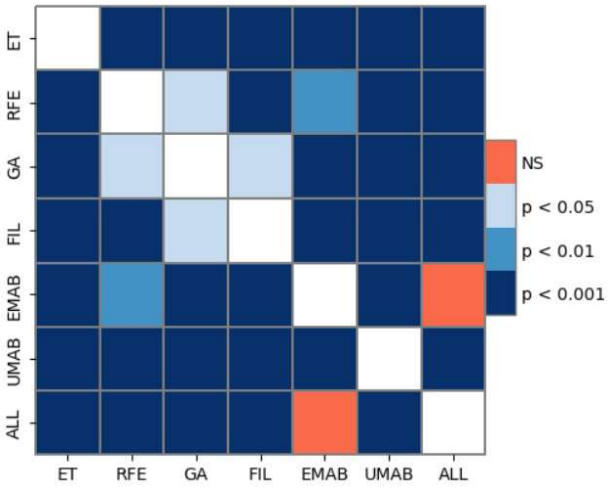
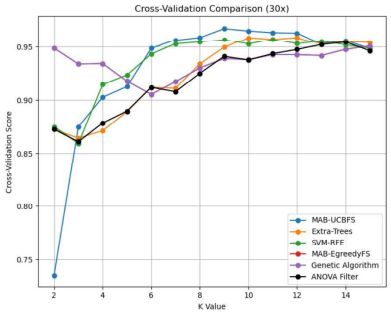


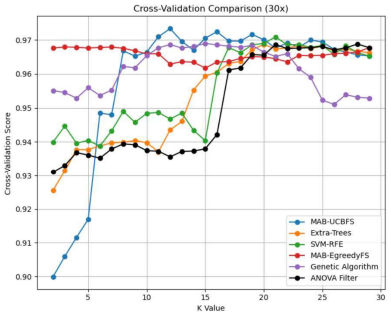
Figure 38 – Heatmap BIM Detector dataset

4.1.4.8 Put all togheter

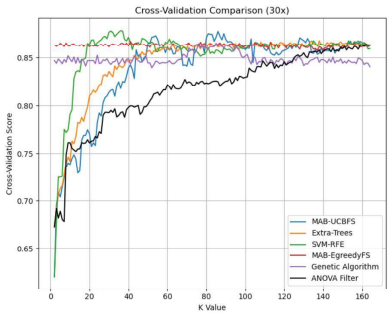
For each data set, it is possible to analyze how well a method performed for each k selected, $k < n$, where n is the total number of features. Graphs 39 show how MAB-UCBFS overcomes the other feature selection methods not only with the best accuracy for a given k, but in some cases a range of k .



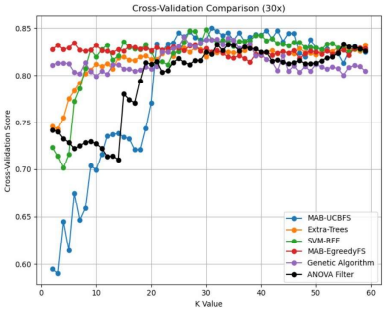
(a) Diabetes dataset evaluation



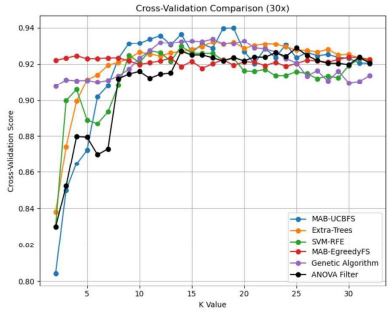
(b) Wisconsin dataset evaluation



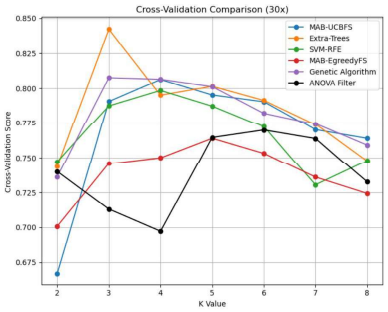
(c) Musk dataset evaluation



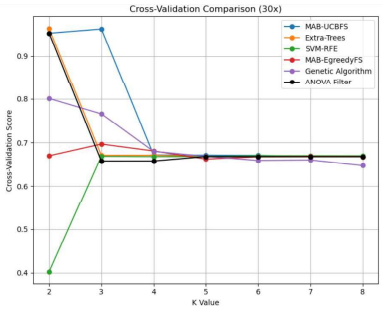
(d) Mine dataset evaluation



(e) Ionosphere dataset evaluation



(f) Coimbra dataset evaluation



(g) BIM Detector dataset evaluation

Figure 39 – Cross-Validation Score Comparison Across Multiple Datasets.

Each data set has a homogeneous class distribution.

Tables 18 and Table 19 show the overall performance of MAB-UCBFS, and in the table the highlighted values. The MAB-UCBFS method consistently demonstrates strong performance across multiple datasets, emerging as the "Best Method" for the Diabetes, MINE, IONOSPHERE, and Wisconsin datasets. This indicates its notable effectiveness in identifying relevant features that contribute to higher classification accuracy in these contexts.

Dataset	MAB-UCBFS	Extra-Tree-Based	SVM-RFE	MAB-EgreedyFS	GA	filter-based ANOVA	Best Method
Coimbra	0.8062	**0.8421**	0.7983	0.7638	0.8075	0.7697	Extra-Tree
Diabetes	**0.9675**	0.9589	0.9571	0.9507	0.9507	0.9544	MAB-UCBFS
MUSK	0.8782	0.8684	**0.8790**	0.8677	0.8660	0.8646	SVM-RFE
MINE	**0.8501**	0.8322	0.8487	0.8344	0.8406	0.8335	MAB-UCBFS
IONOSPHERE	**0.9398**	0.9320	0.9295	0.9244	0.9337	0.9268	MAB-UCBFS
Wisconsin	**0.9733**	0.9687	0.9709	0.9679	0.9690	0.9688	MAB-UCBFS
BIM	0.9618	**0.96248**	0.6699	0.6965	0.8008	0.9522	Extra-Tree

Table 18 – Comparison Table of Results

- **Coimbra Dataset**: For the Coimbra dataset, the Extra-Tree-Based method achieved the highest score of 0.8421, outperforming MAB-UCBFS which scored 0.8062. This suggests that for datasets with a very small number of instances and features, the inherent randomization and ensemble nature of Extra-Trees might provide a slight advantage in feature relevance estimation.

- **Diabetes Dataset**: MAB-UCBFS excelled on the Diabetes dataset, achieving the highest score of 0.9675. This highlights its capability to effectively select features in medical datasets, which are often critical for accurate diagnosis.

- **Musk Dataset**: On the MUSK dataset, SVM-RFE demonstrated superior performance with a score of 0.8790, narrowly exceeding MAB-UCBFS's score of 0.8782. It is possible to say that for high-dimensional datasets with complex relationships, SVM-RFE's iterative elimination based on SVM weights and MAB have the same performance.

- **Mine Dataset**: MAB-UCBFS again proved to be the top performer on the Mine dataset, with a score of 0.8501. This suggests its strong applicability in signal processing and classification tasks where identifying key features is crucial.

- **Ionosphere Dataset**: MAB-UCBFS also led on the Ionosphere dataset with a score of 0.9398, reinforcing its consistent strong performance across various scientific domains.

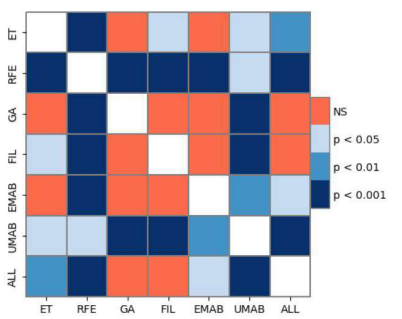
- **Wisconsin Dataset**: On the Breast Cancer Wisconsin dataset, MAB-UCBFS achieved the highest score of 0.9733, further solidifying its position as a highly effective feature selection method for medical diagnostic problems.

- **Bim Detector**:MAB-UCBFS achieved a very closed score with Exrta-trees, 0.9618, A excellent result for a multiclass dataset that may be more complex for a classifier learn. Recovering that MAB-based algorithm are wrapper model that applies a SVM in the reward function, SVM has a very important role.

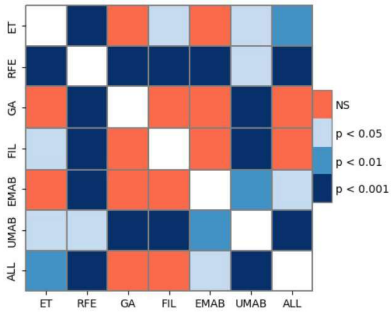
Table 19 – MAB-UCBFS Performance and Dataset Characteristics

Dataset	**MAB-UCBFS Max CV Score**	**Instances**	**Features**	**MAB-UCBFS Was Best Method**
Breast Cancer Wisconsin	0.9733	569	30	Yes
Diabetes	0.9675	400	16	Yes
Ionosphere	0.9398	351	34	Yes
Musk	0.8782	476	165	No (SVM-RFE was best)
Mine	0.8501	208	60	Yes
Breast Cancer Coimbra	0.8062	116	9	No (Extra-Tree-Based was best)
BIM Detector	0.9618	1841	10	No (Extra-Tree-Based was best)

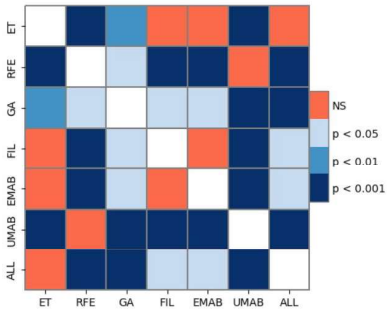
The statistical tests were used to provide a quantitative measure of the observed performance differences. The Conover test, a non-parametric post-hoc test, was applied to the maximum cross-validation scores of each method across the 30 runs. This test allowed for pairwise comparisons between all methods, determining if the differences in their best accuracies were statistically significant. The results are presented in the heatmaps (figure 40), where the color intensity indicates the p-value, with darker shades signifying higher statistical significance (e.g., $p < 0.001$).



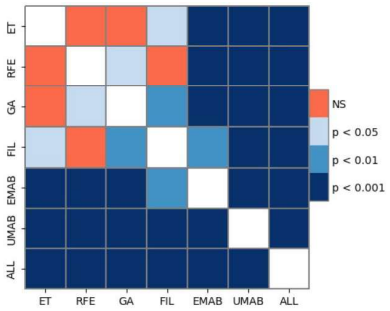
(a) Heatmap Conover Test of Wisconsin dataset



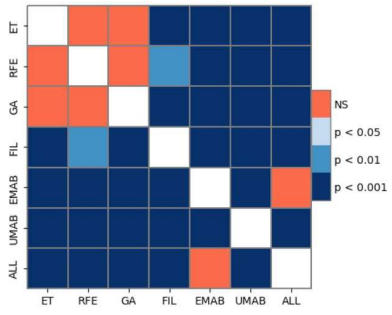
(b) Heatmap Conover Test of Musk dataset



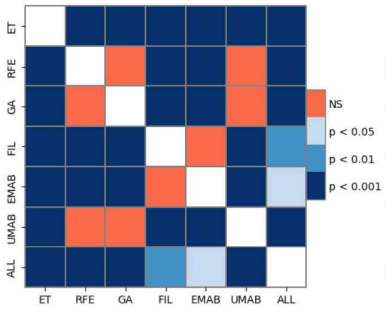
(c) Heatmap Conover Test of Mine dataset



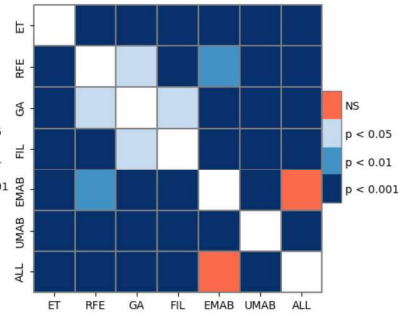
(d) Heatmap Conover Test of Ionosphere dataset



(e) Heatmap Conover Test of Diabetes dataset



(f) Heatmap Conover Test of Coimbra dataset



(g) Heatmap Conover Test of BIM dataset

Figure 40 – Heatmap Conover Test Results for Various Datasets

5
Conclusion

This research aimed to investigate the applicability and effectiveness of Multi-Armed Bandit (MAB) algorithms for feature selection in machine learning tasks. Driven by the challenge of high dimensionality, or the "curse of dimensionality" (COD), which can lead to computational complexities and overfitting in machine learning models, feature selection (FS) emerged as a critical dimensionality reduction technique. While traditional FS methods are prevalent, the integration of MAB in this context remained underexplored. Our primary objective was to propose and validate MAB-based algorithms for feature selection, specifically adapting the Epsilon-greedy and Upper Confidence Bound (UCB) algorithms.

Through a systematic review, we established the current associations of MAB with machine learning, noting its use in recommender systems, automatic selection, and active search. Although existing literature touched upon MAB in feature selection, a pure MAB-based approach directly addressing feature inclusion and exclusion for optimal model performance was a novel contribution. Our proposed MAB-EgreedyFS and MAB-UCBFS algorithms treated each feature's inclusion or exclusion as an "arm" in the bandit problem, dynamically updating the feature set based on the reward function, which was determined by the accuracy of an SVM classifier.

Experimental evaluations were conducted across six diverse datasets (Breast Cancer Coimbra, Breast Cancer Wisconsin, Ionosphere, Diabetes, Mine, and Musk) and compared against established feature selection methods: Extra-trees-based, SVM-RFE, filter-based ANOVA, and Genetic Algorithm (GA). The performance was measured using 10-fold stratified cross-validation with an SVM classifier, repeated 30 times, and statistical significance was assessed using the Conover test.

The results demonstrated that MAB-UCBFS consistently achieved strong performance, notably emerging as the "Best Method" for the Diabetes, Mine, Ionosphere, and Wisconsin datasets. For instance, on the Diabetes dataset, MAB-UCBFS achieved a maximum cross-validation score of 0.9675, outperforming Extra-Trees (0.9589), SVM-RFE (0.9571), MAB-EgreedyFS (0.9507),

GA (0.9507), and filter-based ANOVA (0.9544). Similarly, for the Mine dataset, MAB-UCBFS led with 0.8501, exceeding SVM-RFE (0.8487), GA (0.8406), MAB-EgreedyFS (0.8344), ANOVA Filter (0.8335), and Extra-Trees (0.8322). While not universally superior, as seen with Extra-Trees on the Coimbra dataset (0.8421 vs. MAB-UCBFS's 0.8062) and SVM-RFE on the Musk dataset (0.8790 vs. MAB-UCBFS's 0.8782), MAB-UCBFS demonstrated its robust and competitive nature across a majority of the tested scenarios.

The MAB-EgreedyFS and Filter-based ANOVA methods generally showed lower cross-validation scores in most datasets, suggesting limited effectiveness in this experimental setup compared to other methods. The Genetic Algorithm (GA) often yielded competitive results but was consistently outperformed by other methods when compared directly on specific datasets where an alternative method was identified as the "Best Method". The Conover test heatmaps further corroborated these findings, indicating significant statistical differences between the proposed MAB methods (especially UMAB, representing MAB-UCBFS) and other techniques, particularly on datasets like Diabetes, Ionosphere, and Coimbra.

In conclusion, this research successfully demonstrated the viability and strong performance of MAB-based algorithms, particularly MAB-UCBFS, as innovative and effective solutions for feature selection in machine learning. By balancing exploration and exploitation, these algorithms offer a dynamic and adaptive approach to identifying relevant feature subsets, ultimately contributing to more efficient, accurate, and interpretable machine learning models.

Promising future work can be done investigating the performance of MAB-based algorithms in multi-label datasets where the feature selection sounds more complicated, since a feature contributes for more than one class label in same time. Also MAB can be improved in its processing for avoid the same feature set to be evaluated during the iterations. This can occur since we have a limited number of feature so a sequence of arms can be chosen more than one time. There are others MAB strategies besides UCB and Epsilon-greedy like Thompson Sampling and Softmax (VERMOREL; MOHRI, 2005). Results obtained in this study instigate further research into the application of MAB in various feature selection challenges and their potential integration with more complex machine learning pipelines.

Bibliography

ADORADA, A. et al. Support vector machine-recursive feature elimination (svm-rfe) for selection of microrna expression features of breast cancer. In: IEEE. *2018 2nd international conference on informatics and computational sciences (ICICoS)*. [S.l.], 2018. p. 1–4. Citado 3 vezes nas páginas 15, 50, and 58.

ALFIAN, G. et al. Predicting breast cancer from risk factors using svm and extra-trees-based feature selection method. *Computers*, MDPI, v. 11, n. 9, p. 136, 2022. Citado 4 vezes nas páginas 15, 40, 50, and 58.

ANACONDA Software Distribution. Anaconda Inc., 2020. Disponível em: <https://docs.anaconda.com/>. Citado na página 58.

AUER, P.; CESA-BIANCHI, N.; FISCHER, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, Springer, v. 47, p. 235–256, 2002. Citado 3 vezes nas páginas 34, 36, and 39.

AUSTIN, A. *Building a Multi-Armed Bandit System from the Ground Up: A Recommendations and Ranking Case Study, Part I*. 2022. <https://medium.com/udemy-engineering/building-a-multi-armed-bandit-system-from-the-ground-up-a-recommendations-and-ranking-case-study-b5 Citado na página 34.

BACK, T.; FOGEL, D. B.; MICHALEWICZ, Z. *Basic Algorithms and Operators*. 1st. ed. GBR: IOP Publishing Ltd., 1999. ISBN 0750306645. Citado 2 vezes nas páginas 50 and 58.

BAHASSINE, S. et al. Feature selection using an improved chi-square for arabic text classification. *Journal of King Saud University-Computer and Information Sciences*, Elsevier, v. 32, n. 2, p. 225–231, 2020. Citado 2 vezes nas páginas 14 and 32.

CAÑAMARES, R.; REDONDO, M.; CASTELLS, P. Multi-armed recommender system bandit ensembles. In: *Proceedings of the 13th ACM Conference on Recommender Systems*. [S.l.: s.n.], 2019. p. 432–436. Citado na página 38.

CHAPMAN, D.; JAIN, A. *Musk (Version 1)*. 1994. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5ZK5B. Citado 2 vezes nas páginas 51 and 56.

COETZEE, F. M. Correcting the kullback–leibler distance for feature selection. *Pattern Recognition Letters*, Elsevier, v. 26, n. 11, p. 1675–1683, 2005. Citado na página 29.

DHAL, P.; AZAD, C. A comprehensive survey on feature selection in the various fields of machine learning. *Applied Intelligence*, Springer, p. 1–39, 2022. Citado 9 vezes nas páginas 14, 28, 29, 30, 31, 32, 33, 39, and 50.

EARLY Stage Diabetes Risk Prediction. 2020. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5VG8H. Citado 2 vezes nas páginas 51 and 53.

ELSSIED, N. O. F.; IBRAHIM, O.; OSMAN, A. H. A novel feature selection based on one-way anova f-test for e-mail spam classification. *Research Journal of Applied Sciences, Engineering and Technology*, Maxwell Scientific Publication Corp., v. 7, n. 3, p. 625–638, 2014. Citado na página 41.

FILHO, S. M. M. *PyBIM-Detector*. 2023. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5VG8H. Citado 2 vezes nas páginas 51 and 56.

FORTIN, F.-A. et al. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, v. 13, p. 2171–2175, jul 2012. Citado na página 58.

GEURTS, P.; ERNST, D.; WEHENKEL, L. Extremely randomized trees. *Machine learning*, Springer, v. 63, n. 1, p. 3–42, 2006. Citado na página 26.

GUYON, I. et al. Gene selection for cancer classification using support vector machines. *Machine learning*, Springer, v. 46, p. 389–422, 2002. Citado na página 41.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3. ed. Amsterdam: Morgan Kaufmann, 2012. (Morgan Kaufmann Series in Data Management Systems). ISBN 978-0-12-381479-1. Disponível em: <http://www.sciencedirect.com/science/book/9780123814791>. Citado na página 22.

JEON, H.; OH, S. Hybrid-recursive feature elimination for efficient feature selection. *Applied Sciences*, MDPI, v. 10, n. 9, p. 3211, 2020. Citado 2 vezes nas páginas 15 and 41.

KEELE, S. et al. *Guidelines for performing systematic literature reviews in software engineering*. [S.l.]: Technical report, ver. 2.3 ebse technical report. ebse, 2007. Citado na página 36.

KITCHENHAM, B. Procedures for performing systematic reviews. *Keele, UK, Keele University*, Citeseer, v. 33, n. 2004, p. 1–26, 2004. Citado na página 36.

LI, B.; MENG, M. Q.-H. Tumor recognition in wireless capsule endoscopy images using textural features and svm-based feature selection. *IEEE Transactions on Information Technology in Biomedicine*, IEEE, v. 16, n. 3, p. 323–329, 2012. Citado na página 29.

LI, J. et al. Feature selection: A data perspective. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 50, n. 6, p. 1–45, 2017. Citado na página 14.

LIU, H.; MOTODA, H. Feature selection for knowledge discovery and data mining. *(No Title)*, Springer US, 1998. Citado na página 30.

LIU, K. et al. Multi-armed bandit based feature selection. In: SIAM. *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. [S.l.], 2021. p. 316–323. Citado 4 vezes nas páginas 15, 38, 39, and 40.

LIU, Z.-T. et al. Speech emotion recognition based on feature selection and extreme learning machine decision tree. *Neurocomputing*, Elsevier, v. 273, p. 271–280, 2018. Citado 2 vezes nas páginas 15 and 32.

MITCHELL, M. *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998. ISBN 0262631857. Citado na página 26.

PATRCIO, M.; CARAMELO. *Breast Cancer Coimbra*. 2018. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C52P59. Citado 2 vezes nas páginas 40 and 51.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 58.

RASCHKA, S.; MIRJALILI, V. *Python Machine Learning, 3rd Ed.* 3. ed. Birmingham, UK: Packt Publishing, 2019. ISBN 978-1789955750. Citado 6 vezes nas páginas 18, 19, 20, 23, 25, and 26.

SCHAPIRE, R. E. The strength of weak learnability. *Machine Learning*, v. 5, n. 2, p. 197–227, Jun 1990. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/BF00116037>. Citado na página 25.

SEJNOWSKI, T.; GORMAN, R. *Connectionist Bench (Sonar, Mines vs. Rocks)*. 1988. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5T01Q. Citado 2 vezes nas páginas 51 and 54.

SHANG, Z. et al. Democratizing data science through interactive curation of ml pipelines. In: *Proceedings of the 2019 international conference on management of data*. [S.l.: s.n.], 2019. p. 1171–1188. Citado 2 vezes nas páginas 38 and 39.

SIGILLITO WING, B. *Ionosphere*. 1989. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5W01B. Citado 2 vezes nas páginas 51 and 52.

SLIVKINS, A. et al. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, Now Publishers, Inc., v. 12, n. 1-2, p. 1–286, 2019. Citado 3 vezes nas páginas 15, 33, and 34.

SO, A. et al. *The The Data Science Workshop: A New, Interactive Approach to Learning Data Science*. [S.l.]: Packt Publishing Ltd, 2020. Citado 2 vezes nas páginas 19 and 20.

VAPNIK, V. N. *Statistical Learning Theory*. [S.l.]: Wiley-Interscience, 1998. Citado na página 22.

VERMOREL, J.; MOHRI, M. Multi-armed bandit algorithms and empirical evaluation. In: SPRINGER. *European conference on machine learning*. [S.l.], 2005. p. 437–448. Citado 5 vezes nas páginas 16, 34, 35, 50, and 79.

WAHID, A. et al. Feature selection and classification for gene expression data using novel correlation based overlapping score method via chou's 5-steps rule. *Chemometrics and Intelligent Laboratory Systems*, Elsevier, v. 199, p. 103958, 2020. Citado 3 vezes nas páginas 14, 29, and 32.

WANG, H.; JING, X.; NIU, B. A discrete bacterial algorithm for feature selection in classification of microarray gene expression cancer data. *Knowledge-Based Systems*, Elsevier, v. 126, p. 8–19, 2017. Citado 3 vezes nas páginas 15, 32, and 40.

WOLBERG WILLIAM, M. *Breast Cancer Wisconsin (Diagnostic)*. 1995. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5DW2B. Citado 2 vezes nas páginas 51 and 52.

YAMADA, M. et al. High-dimensional feature selection by feature-wise kernelized lasso. *Neural computation*, MIT Press, v. 26, n. 1, p. 185–207, 2014. Citado na página 30.

ZHU, S.; COLES, J.; XIE, S. Active search using meta-bandits. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. [S.l.: s.n.], 2020. p. 3493–3496. Citado 3 vezes nas páginas 15, 38, and 39.