6-29-2022

An Evolutionary Optimization Algorithm for Automated Classical Machine Learning

Leila Zahedi
Florida International University, lzahed001@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

AN EVOLUTIONARY OPTIMIZATION ALGORITHM FOR AUTOMATED CLASSICAL MACHINE LEARNING

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Leila Zahedi

2022

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Leila Zahedi, and entitled An Evolutionary Optimization Algorithm for Automated Classical Machine Learning, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Sitharama S. Iyengar

Nagarajan Prabakar

Leonardo Bobadilla

Shabnam Rezapour

M. Hadi Amini, Major Professor

Date of Defense: June 29, 2022

The dissertation of Leila Zahedi is approved.

Dean John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2022

DEDICATION

I dedicate this work to my parents, Eghbal Zahedi and Maryam Fouladi, who offered me unconditional love and support and have always been there for me. I appreciate you!

ACKNOWLEDGMENTS

This research is supported by the Dissertation Year Fellowship Award from Florida International University's (FIU) Graduate School. First of all, I would like to thank my advisor, Professor M. Hadi Amini, for encouraging my research under his supervision and allowing me to grow as a research scientist. I would also like to thank my committee members Professors Sitharama S. Iyengar, Nagarajan Prabakar, Leonardo Bobadilla, and Shabnam Rezapour at FIU. I also want to express my gratitude to Professor Matthew W. Ohland from Purdue University, who allowed me to collaborate with him on multiple projects, and my previous advisor, Professor Monique Ross, for her support during my Ph.D. studies. I especially want to thank Professor Lidia Kos of FIU's University Graduate School for all her support. No matter how busy she was, she always made time to meet me and help me during the hard times, and I cannot thank her enough.

Secondly, my thanks go to friends and group members. I appreciate their support. I would especially like to thank my dear friends, Professor Mitra Naseh, Zahra Abtahi, and Dr. Samira Pouyanfar for their comradeship.

Lastly, I am incredibly grateful for the sacrifices and deep love from my mother and father. Words can not express how grateful I am to them for all of the sacrifices they have made on my behalf. My four beloved sisters for always cheering me up, and my partner, Armin, for being such a significant influence in my life. I cannot thank him enough for encouraging me and supporting me throughout this experience. I would never have been able to finish my dissertation without their support and encouragement.

ABSTRACT OF THE DISSERTATION

AN EVOLUTIONARY OPTIMIZATION ALGORITHM FOR AUTOMATED CLASSICAL MACHINE LEARNING

by

Leila Zahedi

Florida International University, 2022

Miami, Florida

Professor M. Hadi Amini, Major Professor

Machine learning is an evolving branch of computational algorithms that allows computers to learn from experiences, make predictions, and solve different problems without being explicitly programmed [ENM15]. However, building a useful machine learning model is a challenging process, requiring human expertise to perform various proper tasks and ensure that the machine learning's primary objective –determining the best and most predictive model – is achieved. These tasks include pre-processing, feature selection, and model selection.

Many machine learning models developed by experts are designed manually and by trial and error. In other words, even experts need the time and resources to create good predictive machine learning models. The idea of automated machine learning (AutoML) is to automate a machine learning pipeline in order to release the burden of substantial development costs and manual processes. The algorithms leveraged in these systems have different hyper-parameters. On the other hand, different input datasets have various features. In both cases, the final performance of the model is closely related to the final selected configuration of features and hyper-parameters. That is why they are considered as crucial tasks in the AutoML. The challenges regarding the computationally expensive nature of tuning hyper-parameters and optimally selecting features create significant opportunities for filling the research gaps in the AutoML field. This dissertation explores how to efficiently and automatically select the features and tune the hyper-parameters of conventional machine learning algorithms.

To address the challenges in the AutoML area, novel algorithms for hyper-parameter tuning and feature selection are proposed. The hyper-parameter tuning algorithm aims to provide the optimal set of hyper-parameters in three conventional machine learning models (Random Forest, XGBoost and Support Vector Machine) to obtain best scores in regards to performance. On the other hand, the feature selection algorithm looks for the optimal subset of features to achieve the highest performance. Afterward, a hybrid framework is designed for both hyper-parameter tuning and feature selection. The proposed framework can discover sub-optimal configuration of features and hyper-parameters. The proposed framework includes the following components: (1) an automatic feature selection component based on artificial bee colony algorithms and machine learning training, and (2) an automatic hyper-parameter tuning component based on artificial bee colony algorithms and machine learning training for faster training and convergence of the learning models. The whole framework has been evaluated using four real-world datasets in different applications. This framework is an attempt to alleviate the challenges of hyper-parameter tuning and feature selection by using efficient algorithms. However, distributed processing, distributed learning, parallel computing and other big data solutions are not taken into consideration in this framework.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

ABBREVIATIONS

A2BCF	Automated Artificial Bee Colony for Feature Selection
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
CV	Cross Validation
EA	Evolutionary Algorithms
FSO	Feature Selection Optimization
GA	Genetic Algorithm
HPC	High-Performance Computational
HPO	Hyperparameter Optimization
ML	Machine Learning
RF	Random Forest
MIDFIELD	Multiple-Institution Database for Investigating Engineering Longitudinal Development
OBL	Opposition-Based Learning
PS	Population Size
PSO	Particle Swarm Optimization
SBS	Sequential Backward Selection
SFS	Sequential Forward Selection
SI	Swarm Intelligence
SVM	Support Vector Machine
XGBoost	Extreme Gradient Boosting

CHAPTER 1
INTRODUCTION

Abstract In this chapter, we first cover the background and introduction, as well as some of the challenges in hyper-parameter tuning and the feature selection approach. Then we present the proposed solution and the contributions of out work. We then explain the scope and limitations of this work. This chapter also provides the outline of this dissertation at the end.

1.1 Background and Introduction

Machine Learning (ML) is an evolving field of computational algorithms that allow computing devices to learn from past experiences and historical datasets [JM15]. ML can be leveraged to make predictions and decisions based on models created from large or complex datasets [SK17a]. It is one of the fast-growing research fields and has garnered much attention from academic and industrial researchers who apply it to discover the patterns and data representations from the raw data [WF02]. This prevalence of data representation is because ML algorithms can perform without being explicitly programmed and provide high performance, and are suitable for different types of problems [ZH19]. ML addresses the matter of manipulating[1], managing, mining, understanding, and adequately visualizing different kinds of data. These solutions include but are not limited to data processing and management for a variety of applications such as banking [CVSZ20, PD17], medicine and healthcare [HSW+19, KZW+17], marketing [PC09, CWL06], education [LGN+09, LAS+15], and engineering [NMS09, PMG+17], where it is challenging to develop classic algorithms to perform the expected tasks. However, building an efficient ML model

[1]Data manipulation is the process of organizing or altering data so that it is more readable and easier to interpret. Data manipulation improves the quality of data and analysis.

is a challenging process that requires consideration of efficiency and effectiveness in terms of both time and performance.

Overall, ML's main objective involves determining the most predictive model. In cases where datasets are so large, the model might not work well before eliminating the unrelated features. Further, various ML algorithms have different hyper-parameters, and tuning these hyper-parameters help achieve a model with better performance [WLU20]. Manually choosing the sub-optimal subset of features and hyper-parameters among all the combinations can be exhaustive, costly, and impractical. Several ML models developed by experts are all designed manually and by trial and error (such as manually testing the hyper-parameters) [HKV19]. In other words, even experts need the time and resources to create efficient predictive ML models [HZC21]. The idea of Automated Machine Learning (AutoML) emerged in an effort to automate the process of applying ML techniques and find ML solutions [HCB+14]. Hence, AutoML's focus is on users with little or no knowledge of ML as well as ML experts. AutoML releases the burden of immense development costs and manual processes that improve capability of decision-making. A complete AutoML includes data pre-processing, feature engineering, model selection, and model evaluation [HZC21, FEF+18, GBC+15].

Hyper-parameter tuning and feature selection are among the primary tasks in AutoML [BMTB20] and provide fairness to research and scientific studies, as it provides reproducibility and the same level of tuning for the problem at hand [HKV19]. The computationally expensive nature of tuning tasks, the complexity of ML algorithms, the increasing amount of large databases, and the need for building efficient ML models prompt new challenges in the AutoML field. These challenges create significant opportunities and fertile ground for rewarding future research avenues on AutoML, specifically on Hyper-Parameter Optimization (HPO) and Feature Selection Optimization (FSO) methods.

Building on the idea of using evolutionary computing for machine learning and data analytics proposed in [MAA20b, MAA20a], this dissertation proposes to leverage an evolutionary optimization algorithm, namely Artificial Bee Colony (ABC), for enabling automated classical machine learning algorithms, i.e., Random Forest, XGBoost, Support Vector Machine. It also develops a plug-and-play library, and its underlying algorithms, to enable automated hyper-parameter tuning and feature selection optimization in these classical ML algorithms. The final outcome of this dissertation is to provide a hybrid framework for feature selection and hyper-parameter tuning using classical ML algorithms for a few application domains in both classification and regression problems This study aims to address some of the main challenges in this area and provide solutions to build effective models in different applications.

Overall, some of the main challenges in HPO and FSO include:

- **Dimensionality:** High dimensionality introduced new challenges to the world of data science. One of these challenges is the Curse of Dimensionality (CoD). CoD refers to a set of problems that arise when working with high-dimensional data, and that have most often unfortunate consequences on the behavior and performances of learning algorithms [VF05]. CoD brings high time complexity to the problems and extends the time scientists must wait to obtain reasonable results [MAA20b]. Large datasets can make both HPO and FSO slow due to increasing the number of features and records. Therefore, the computing time and hence the FSO/HPO process slows down since the computation cost increases exponentially with the size of the dataset dimension.
- **Large configuration space:** HPO and FSO methods are time-consuming, and this not only increases with high dimensions of datasets but also with

larger search spaces. Many of the black-box (BB) optimization models do not fit HPO problems (with limited time and resource budgets) as they usually ignore the function evaluation time [YS20]. Hence, efficient algorithms should be used for these problems in order to find the optimal set of hyper-parameters [YS20], and features [MA14].

- **Expensive function evaluations:** ML algorithms have different objective functions. The complexity of objective functions depends on the complexity of the ML models themselves. For instance, some models have more training time as they are more dependent on the size of the dataset (e.g., support vector machines [YHB16]). The costly objective function of some ML algorithms makes them even more challenging when used for iterative optimization processes (such as nature-inspired algorithms), as they significantly increase the execution time (due to time-consuming training processes for each evaluation). Therefore, proper strategies are required for such ML algorithms to address this challenge and enhance the running time.

The challenges mentioned above motivated us to explore HPO and FSO techniques in an effort to analyze methods that can decrease the influence of dimensionality or large search spaces on execution time. In this dissertation, AutoML is considered for making predictions, in particular classification and regression tasks using classical machine learning algorithms using Random Forest (RF), Support Vector Machine (SVM), and Extreme Gradient Boosting (XGBoost). However, the issue explored in this dissertation can be conveniently extended to other ML algorithms. Further, the final outcome is a joint HPO+FSO tool that can be used to 1) find the optimal ML algorithm performance for a given dataset, 2) perform HPO, and 3) perform FSO.

1.2 Proposed Solution

Although tuning hyper-parameters and/or features lead to having models with better performance (e.g., accuracy, mean squared error, etc), the optimization process for finding the near optimal configuration is very time-consuming and sometimes impractical [YS20]. AutoML is a relatively new field and there is still space for improvement in regards to time and performance. Being time-consuming is one of the disadvantages of AutoML [ZWRM21]. This dissertation presents a novel framework that partly handles HPO and FSO problems in classical ML algorithms to address these challenges. The proposed framework includes:

- **Development of a novel FSO algorithm:** This algorithm is a novel Artificial Bee Colony (ABC) based algorithm fitted to FSO problems. Moreover, the algorithm is modified in order to improve the ML models' performance and simplify the structures' size, enhancing the execution time for finding the optimal set of features using learning algorithms (particularly clustering and opposition-based algorithms). This algorithm is further improved by filtering the poor configuration from the whole search space. The proposed feature selection method, *A2BCF*, inspired by Ghareh Mohammadi et al.[MA14], is later inserted into our proposed hybrid framework. *A2BCF* is able to improve the performance and/or run-time compared to the conventional hand-crafted features and efficiently extracts robust features from the raw data.
- **Development of a novel HPO algorithm:** In this dissertation, we first present the *HyP-ABC* algorithm in which ABC is fitted to the HPO problems to find close to the optimal hyper-parameters of ML algorithms. The *HyP-ABC* algorithm is further enhanced using learning algorithms. In particular, clustering and opposition-based algorithms are leveraged for improving

the performance and convergence rate of the algorithms. We call this algorithm *OPT-ABC*. This algorithm is specifically leveraged to two real-world applications such as education and energy demand. *OPT-ABC*, used in this framework, is later integrated into our proposed hybrid HPO-FSO framework.

1.3 Contributions

The contributions of this dissertation are multi-fold:

- A novel automated feature selection method is proposed as a preprocessing step for selecting the optimal set of datasets' features. In addition to the random-based strategy used in the original ABC algorithm, this ABC-based method includes clustering and opposition-based learning [Tiz05] methods as well as an early stopping strategy. The first method is used to provide diversity to the features' search space [CLWY18]. The second method is used to balance the exploration and exploitation of the algorithm and improve the algorithm's convergence rate. The early stopping strategy is also applied to eliminate the poor configurations early in the process and prevent training the ML algorithm on those solutions.
- A novel optimal and automated hyper-parameter tuning method is proposed. This is an iterative step for finding the optimal set of datasets' hyper-parameters. This method also includes clustering and opposition-based learning algorithms. These algorithms are designed to enhance the convergence rate of the original ABC-based HPO algorithms. As mentioned above, these methods are used to provide diversity (to hyper-parameter configurations), balance the exploration and exploitation of the algorithm, and improve the algorithm's convergence rate. The algorithms automatically determine how to modify the configura-

tions based on the performance during the training. In particular, it is applied to classical ML algorithms.

- A novel HPO-FSO framework is proposed, namely *FSHPO*, for automatically selecting the features and hyper-parameters in different regression and classification problems. This model integrates the *OPT-ABC* and *A2BCF* algorithms to create an end-to-end framework that contains two layers of FSO to capture the optimal set of features and HPO to capture the optimal set of hyper-parameters. *FSHPO* takes the cleansed dataset and returns the optimal architecture of the ML model.

- This dissertation does not only contribute to developing novel tools for HPO and FSO, but it also proposes to apply these tools to two major applications for the first time. Specifically, a real-world education dataset is used to assess the proposed HPO and FSO algorithms. Among other applications, this is the first time HPO-FSO has been used in the field of education to predict students' success.

1.4 Scope and Limitations

The proposed framework has several limitations and assumptions as follows: The proposed framework is evaluated on structured datasets in education and power demand datasets. However, proposed HPO and FSO models can also be extended to cover other data types such as images. When applying ML algorithms, only a few hyper-parameters have major effects on the performance of the model performance, and they are the main hyper-parameters that require tuning (such as the number of trees in the Random Forest Algorithm). The proposed hyper-parameter tuning method focuses on the *main* hyper-parameters in classical ML algorithms. However, certain other unimportant hyper-parameters may still affect the model performance slightly (such as shrinking in Support Vector Machine algorithm) and may be considered to optimize the ML model further. However, it increases the dimensionality of hyper-parameters search space.

However, there are other hyper-parameters with less impact on the performance that can be considered and automated in future work. The main focus of this dissertation is on supervised ML algorithms, and unsupervised learning algorithms are not being explored in this work. Although the proposed techniques can potentially be extended for unsupervised learning, it is out of the scope of this dissertation. The proposed framework alleviates the HPO and FSO challenges using efficient algorithms and techniques to improve the convergence rate. However, distributed processing, distributed learning, parallel computing and other big data solutions (such as GPU programming) are not taken into consideration in the proposed framework.

1.5 Outline

The rest of this dissertation is organized as follows. Chapter 2 covers the related work in optimization for ML, with a major focus on hyper-parameter optimization and feature selection optimization in ML algorithms. The proposed HPO-FSO framework for conventional ML algorithms, its main components, and the application and experiments of datasets are presented in chapter 3. In chapter 4, a novel evolutionary optimization-based feature selection technique using the Artificial Bee Colony algorithm is proposed. Chapter 5 discusses the proposed ABC-based hyper-parameter tuning technique. In chapter 6, a final hybrid model that represents ABC-based HPO+FSO for conventional ML algorithms is presented for the automated ML framework is presented. Finally, in chapter 8, the conclusions and future directions are given. Table 1.1 shows the chapters, and their content is taken from publications.

All codes generated for the experiments in this dissertation are publicly available for reproduction of the analysis and extension (to other ML models)[2].

Table 1.1: Related published research papers during my Ph.D.

Chapter/Section	**Re-Used Published Research**
Chapter 3	A2BCF: An Automated ABC-Based Feature Selection Algorithm for Classification Models, *Applied Sciences'22* [ZGMA22]
Chapter 4.1	Search Algorithms for Automated Hyper-Parameter Tuning, *ICDATA'21* [ZMR+21]
Chapter 4.2	OptABC: an Optimal Hyperparameter Tuning Approach for Machine Learning Algorithms, *ICMLA'22* [ZMA21c]
Chapter 6.3	ABC-based Optimal Hyper-parameter Tuning for Electric Load Forecasting, *ICAI'22* [SZM+22]

[2]https://github.com/LeilaZa?tab=repositories

CHAPTER 2

RELATED WORK ON HPO AND FSO IN ML ALGORITHMS

Abstract In chapter one we covered the introduction and background in the field of AutoML, more specifically HPO and FSO. In this chapter, we aim at covering a high-level summary of the mathematical optimization and optimization for ML. Then, we cover a review of related works for hyper-parameter tuning and feature selection in ML.

2.1 Mathematical Optimization

Mathematical optimization (also known as mathematical programming) is selecting the most optimal element from a set of available options, considering some criterion [Zio88]. These types of problems exist in various disciplines, and exploring methods to solve such problems has been of interest for centuries [DPW01]. The purpose of optimization problems is to minimize or maximize a function given a set of constraints. ML algorithms also involve solving optimization problems. When an ML model is being built, its weight parameters are initialized and optimized in a process until the performance reaches a maximum level [SCZZ19].

Similarly, the goal of HPO and FSO methods is to optimize the ML model's architecture by finding the optimal set of hyper-parameters and features, respectively. Optimization problems have two main components: a set of decision variables and a function that should be minimized or maximized. Depending on the class of the optimization problem, hyper-parameters/features may have certain acceptable ranges for the decision variables. Hence, optimization problems are divided into two categories [BHM77]:

1. **Unconstrained:** Decision variables in this method can take any value from real numbers, R, and can be expressed as 2.1

$$\min_{x \in \mathbf{R}} f(x) \tag{2.1}$$

   where $f(x)$ is the objective function.

2. **Constrained**: In constrained optimization problems, decision variables have constraints and can be subject to mathematical equalities or inequalities. Most of the real-world optimization problems are constrained optimization problems and can be denoted as shown in Equation 2.2.

$$\begin{aligned} \min_{x} f(x) \text{ subject to:} \\ g(x) <= 0, i = 1, 2, .., m, \\ h(x) <= 0, i = 1, 2, .., p, \\ x \in \mathbf{X} \end{aligned} \tag{2.2}$$

   where X is the domain for variable x. The domain X limits the possible values for decision variables and creates a feasible search space.

In ML models, most of the HPO and FSO problems are constrained optimization problems. They have constraints for decision variables, including but not limited to, specific accepted ranges, hyper-parameter values, and types and space/time constraints. These constraints limit the whole search area to feasible search spaces, and that is why the optimum solution might be the local optimum in the entire search space. However, the local optimum is guaranteed to be the global optimum if only the function is convex, as in these functions the global and local optimums are equal [Bub14].

On the other hand, non-convex optimization problems may require proper strategies to find global or close to global optimums instead of a local optimum.

There is a wide range of methods to solve optimization problems. One of the most common optimization methods is Gradient Descent (GD). The search strategy in the GD method uses the negative gradient direction in order to move toward the optimum solution. However, GD can be used to find the local minimum as it can not guarantee the discovery of the global optimum. GD can only guarantee to find the global optimum if the problem's objective function is convex. The Conjugate GD method is faster than the original GD, but its calculations are more complicated. There are also some other methods, such as Newton's method, which has a better convergence speed compared with GD but requires a larger space [YS20]. These traditional methods follow systematic steps to find the optimal solutions. On the other hand, we have heuristic strategies based on empirical rules to discover the optimal solutions. Although these methods do not guarantee the global optimum, they often find the approximated global optimum [SCZZ19].

2.2 Optimization for Machine Learning Algorithms

Usually, there are a few questions that we need to address in an ML problem:

1. **How do we select the candidate models, features, and hyper-parameters?**

   Choosing a model from a set of representative models in the fields of ML and statistics has always been a concern. This choice needs to be made based on selection criteria. Every ML model has a score or value (called performance), which demonstrates the quality of the model. Hence, the model with the better score should be chosen for decision-making, or further analysis [NC12].

In general, based on Occam's razor, "a model should be simple enough for efficient computation, and complex enough to be able to capture data specifics" [Bie03]. The problem with complex models is that there is always the possibility of overfitting, but a simple model can smooth out features' noise [Bie03]. Feature selection is a way of removing noise and random errors in the underlying data. In feature selection, we use techniques to select features from the more relevant data and remove the redundant or irrelevant ones without incurring much loss of information [BPWS+15].

Hyper-parameters of an ML model control the learning process [Kim19]. In each ML model, some hyper-parameters are essential and have a more considerable impact on the model's performance. In contrast, the rest may not have as much effect on the model performance. Therefore, a process of finding a set of hyper-parameters that enhances the performance of the model would be of great help [CDM15, KS96].

2. **What is the expected performance considering the trade-off between optimization performance and run-time?**

   For hyper-parameter tuning and feature selection problems, there is a trade-off between exploration (searching relatively unexplored regions of the space to avoid premature convergence) and exploitation (optimizing the hyper-parameters locally). In other words, there should be a balance between the number of configurations and the budget allocated to those configurations [YS20, KTH+19]. Therefore, the search space and selection strategy (i.e., optimization algorithm) for hyper-parameters and features should be based on the available computational budget and resources (such as time).

3. **What is the final model we use?**

   The goal of model selection is to determine the ML model with optimal generalization performance, considering a set of learning algorithms and the data [KTH+19]. As mentioned above, the best model is selected based on the key performance indicators (PI). There are different PIs available for model comparison, each fit for a specific set of problems (e.g., accuracy, precision, recall, F1) [GG05]. Based on the problem, the corresponding maximum performance is determined, and the final model is selected. However, the performance depends on the chosen features and hyper-parameters.

Determining the appropriate hyper-parameter and features' subset to ascertain the best-fitting model is a complicated optimization problem and a decisive issue. Manual testing is a common but ineffective way of tuning precisely because of model complexities, time-consuming model evaluation, and vast search space for models' hyper-parameters or features. On the other hand, the traditional automated exhaustive search goes over all possible hyper-parameters or features to find the optimal subset. However, such methods are impractical for large datasets. This issue is exacerbated when the number of features and/or when the number of hyper-parameters increases (with the time complexity of $O(2^m)$ and $O(n^k)$, respectively) [YGW16].

In summary, ML's main goal involves determining the best and most predictive model, which can be obtained by suitable feature engineering and tuning the hyper-parameters [EMS19]. Choosing the best subset of features and hyper-parameters directly affects the model's performance and users need a deep understanding of the ML models. The optimal parameter values and features depend on the application and, more specifically, the data itself. ML algorithms need automatic feature selection and hyper-parameter tuning approaches to be effective in the application.

2.2.1 Feature Selection in Classical Machine Learning Models

Feature engineering and feature selection are among the essential tasks in predictive analytic projects [LTS+17]. It is an indispensable approach that removes the noisy data which enhance the precision and enables the model to perform faster. Therefore the process of selecting features for large datasets is crucial to decreasing models' run-time and generating a logical input set for ML models. However, there is a trade-off between computational complexity and performance, and a smaller subset of features decreases the time complexity. Although there are many feature selection techniques that have been proposed for classification and regression tasks in the past few decades, they sometimes fail to optimally reduce the high dimensionality of the feature space. Emerging technologies such as heuristics optimization methods provide a new paradigm for feature selection due to their strength in enhancing the performance of computational demands, classification, and storage. They also help solve complex optimization problems in less time. Several heuristics approaches for feature selection compromise the performance to decrease the time complexity. Therefore, an appropriate, efficient FSO method that can reduce the number of features without decreasing the model performance is required. This section reviews the present state of feature selection with respect to meta-heuristics and hyper-heuristic methods.

There are many different definitions for feature selection. Nevertheless, the main idea is to select a subset of appropriate features from an initial wide range of features. FSO approaches try to find an optimal set of features by removing unimportant features with the goal of improving the prediction accuracy or structure's size without significantly decreasing the prediction accuracy of the model [KS96, KM14].

The removal of irrelevant features not only provides an efficient data representation, but also improves the learning rate, decreases the data dimensionality, and enhances the performance.

There are different approaches for feature selection. However, they can be divided into two main categories, namely, filter-based and wrapper-based. Each of these methods has its own advantages and disadvantages.

- **Filter methods:** In these methods, selecting the features is performed before leveraging the learning algorithm. Therefore, the features do not depend on the ML algorithm. In these methods, the importance of features is calculated (based on some predefined criterion), and then the best feature subset is selected. In these methods, no learning algorithm is used for choosing the features. The advantage of these methods is their faster speed. [ZGMA22]
- **Wrapper methods:** Unlike filter methods, wrapper methods generate different subsets of features by adding and removing features to achieve reasonable accuracy. The predictive accuracy of the classifier is used to evaluate the subset of features. The wrapper methods use a predefined classifier to explore a subset of features. Then it utilizes the classifier to measure the subset of features. The selection process continues till the desired performance is achieved. The advantage of these methods includes their high classification accuracy. However, their computational complexity is higher than filter methods. Therefore, researchers have been exploring different methods to enhance the convergence of wrapper methods. Wrapper methods have gained much attention due to their promising performance. Prior works have used wrapper methods for selecting the optimal features [ZGMA22, OSBS03, PX13, SIM11, YO11, MA14].

The conventional-based feature selection approaches such as filter-based methods are usually unable to handle problems that require an ample solution space due to inter-dependencies and nonlinear requirements amongst features [GT13]. Filter-based methods have the disadvantage of complexity and not being to provide satisfactory results. These issues and more have motivated researchers to explore other methods of obtaining better performing options [AAA+21].

Metaheuristic-based approaches have proven their performance in different fields as they deliver practical solutions in a reasonable time and their strength in overcoming the curse of dimensionality (CoD) by optimizing the performance of classification, and reducing the use of computational resources. Hence, This dissertation focuses on metaheuristic-based feature selection algorithms for classification and regression tasks due to their favorable performance than conventional feature selection methods.

When dealing with large datasets it is quite challenging to conclude which features are related to the problem and which are not. Also, if all the features are selected then the final outcome of the model might be affected due to the noisy features. Additionally, selecting all features makes the analysis process very time-consuming and should be avoided. Therefore, finding the best subset is essential to feed the model with the most related features to the problem at hand.

Figure 2.1 shows the ideology for FSO approaches. The feature selection step can be done manually or via some automated techniques.

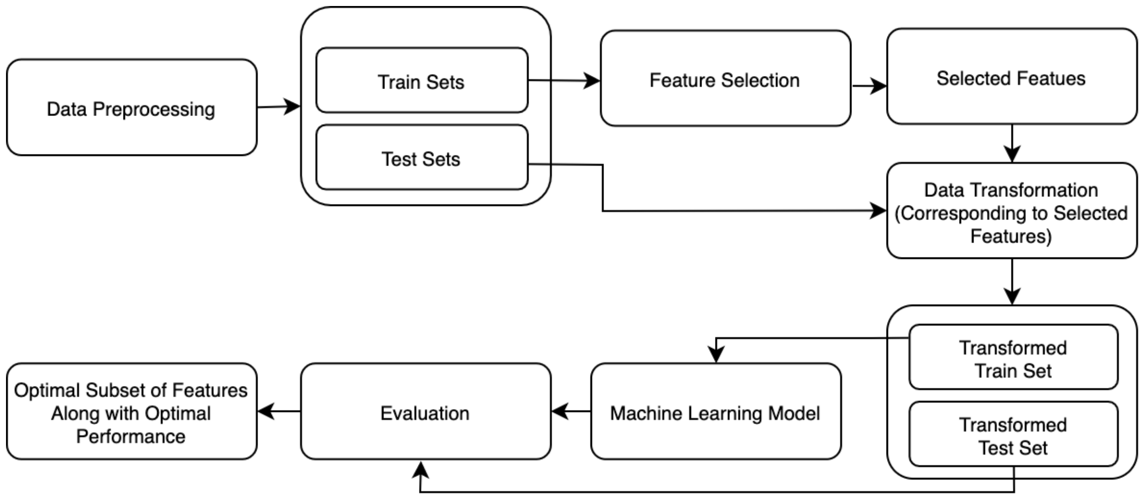


Figure 2.1: General feature selection schema

Depending on the models' number of features, finding the best subset of features among the 2^n existing combinations could be very costly. Therefore, researchers have provided different feature selection methods to reduce models' computational costs.

Metaheuristic-based algorithms

Selecting features from large datasets using traditional feature selection approaches is complicated and can not be easily solved. Therefore, Metaheuristic-based approaches came into the picture and became more established in the literature due to their performance when solving complex problems [BR03].

Metaheuristics are used in different feature selection tasks in various fields due to their excellent performance in global search and performance. The established literature divides the metaheuristic algorithms into population-based and local search algorithms [BR03]. The population-based algorithms examine a population of different solutions in the search space and enhance them iteratively to achieve the ideal solution. In [BR03, ABHAAD21], authors provide an extensive treatment of different metaheuristic references.

Dash and Liu state that feature selection methods typically have four basic steps as follows [DL97]:

1. A procedure to generate the next configuration candidate,
2. An evaluation function,
3. Stopping criteria,
4. Validation procedure.

The state of the art: metaheuristics methods

Feature selection approaches using metaheuristics have gained a lot of attention and are increasingly studied. This shows the importance of feature selection in different tasks. The two main categories of Metaheuristics are swarm intelligence, and evolutionary-based.
Swarm intelligence (SI) is a population-based stochastic optimization technique that is considered among nature-inspired algorithms. It works based on self-organized frameworks that can move iteratively in a planned way. This framework has a population of solutions that can directly or indirectly communicate locally [RBNF21]. Some examples of SIs are ant colonies, bee colonies, birds flocking, fish schooling, and microbial intelligence [Tal09].

There are SI-based metaheuristic methods existing in the literature for feature selection including PSO, ABC, amongst others.

The PSO algorithm is inspired by the social behavior of birds. In [BD15] presented a Hamming distance-based binary particle swarm optimization for high dimensional feature selection. This technique uses hamming distance to update the velocity of particles in a binary problem. In another study[YDwWq16], authors proposed an enhanced PSO method to improve the searchability of the PSO method based on the introduction of two new operators. In [JJJ18] proposed an algorithm with modifying PSO algorithm to classify cancer. There are also other works existing in the literature based on the PSO-based approach [ZDZY18, QA18, PBH18, GJM18, PPA19, XTPL20]

The ABC approach is inspired by the intelligent behavior of simulating the food search of bee populations. In [SK17b], the authors presented a hybrid of ABC and the ACO algorithms to provide a high-performing model. In another study, the authors integrated an ABC-based multi-objective optimization algorithm with a sample reduction technique. The results of this study proved an increase in the performance and a decrease in the computational complexity [WZS^{+}20]. Arslan et al. presented an ABC-based algorithm for high-dimensional symbolic regression with feature selection [AO19]. In another study, Grover and Chawla improved the ABC algorithm for feature selection by utilizing an intelligent strategy [GC20]. There are also other approaches using the ABC algorithm and can be found in the literature [HXZ^{+}18, BM19, HXKZ15]
There are also some methods that consider other aspects besides the model's predictivity in their feature selection method. For instance, Georges et al. looked into feature reproducibility, which means the same subset of features selected can be found and used in similar datasets in the field (e.g., structural and functional MRI data) [GMR^{+}20].

The domain of features in FSO problems is only binary. In other words, for each of the features in the dataset, the only options for that specific feature would be one or zero, for including or excluding the feature, respectively. As in FSO problems the feature can only take two values, and it is considered as a constrained optimization problem. Hence, in FSO problems the goal is to achieve [LNK+17]:

$$x^* = arg \min_{x \in 0,1} f(x) \tag{2.3}$$

where $f(x)$ is the objective function we aim to minimize (such as root mean square error or the error rate) or maximize (such as accuracy or F1-score). x^* is also the subset of features that generated the optimum value of $f(x)$.

2.2.2 Hyperparameter Tuning in Machine Learning Models

To improve ML models by HPO, we first need to know what are the main hyper-parameters that users need to tune to fit the model into a problem or a dataset. Whether the ML algorithm is built to model labeled or unlabeled datasets, they are divided into supervised and unsupervised algorithms, respectively [MISL18]. Supervised algorithms mainly include linear models, naíve Bayes (NB), decision-tree-based models, k-nearest neighbors (KNN), support vector machines (SVM) and deep learning algorithms [CNM06]. Unsupervised algorithms, on the other hand, aim to find hidden patterns from unlabeled data and based on the main goals they are classified into clustering and dimensionality reduction algorithms. They mainly include k-means, hierarchical clustering, expectation-maximization or principal component analysis (PCA), and linear discriminant analysis (LDA) [Kra16]. In this dissertation, the important hyper-parameters of common ML models are explored including ML algorithms scikit-learn [PVG+11], and XGBoost [CHB+15].

2.2.3 Hyper-parameter optimization techniques

Model-free algorithms

- Babysitting: This method is nothing other than manual or trial and error and is the most basic tuning method [Abr19]. In this method, use different possible configurations of hyper-parameters based on experience or guessing and repeat this process until they yield the desired results. Therefore, this method requires enough knowledge and time to be able to find the optimal hyper-parameters. In [ZLP+20], Zahedi et al., used manual tuning of the hyper-parameters to improve the performance of the ML models in a classification problem. However, in many problems, manual tuning the hyper-parameter is almost impossible, especially when dealing with a large number/ranges of hyper-parameters, and complex models [Ste18]. These issues motivated researchers to explore various approaches for the automated HPO.

- Grid Search: This method is the most common tuning approach and is considered an exhaustive search strategy that goes over all the configurations in the search space [IMNS20]. The most important disadvantage of GS is its inefficiency, especially when dealing with Curse of Dimensionality (CoD) [CSP+14]. In other words, this is due to the fact that the number of configurations grows exponentially when the number of hyper-parameters increases.

- Random Search: This method [BB12] is proposed to partly solve the challenges existing in the GS. Unlike GS, RS does not evaluate all the configurations in the search space. RS randomly selects and evaluates a pre-defined number of configurations. If the resources are limited RS is able to evaluate a larger search space than GS [BB12]. The advantage of RS over GS is that the probability of wasting time on a low-quality region in the search space in the

RS method is much less than the GS. The computational complexity of the RS method is O(n) where n is the pre-defined number of evaluations [Wit05]. Due to the random nature of the RS method, there are still a remarkable number of configurations that are unnecessarily evaluated. This is due to the fact that the evaluations are independent of each other and the algorithm does not exploit the regions that perform better.

Model-based algorithms

- Gradient Descent: This method [Ben00], is a first-order iterative optimization algorithm that discovers promising directions and moves toward the optimal solution. This approach selects a random point and moves toward the biggest gradient to discover the next point. Hence, in this method, the local optimal can be found after the convergence. Although gradient descent approaches are strong in finding the local optimum, they have some limitations as well. Firstly, they can only be used for continuous problems as some hyper-parameter types such as discrete parameters do not have gradients. Also, gradient descent methods are suitable approaches for convex problems as gradient descent may stuck in the local optimum instead of a global optimum.
- Bayesian Optimization: This method [SLA12] is a very popular method in HPO problems. Unlike model-free approaches, BO is more efficient. BO is a model-based technique that determines the future data points based on the evaluated results from previous data points. Hence it is cheaper than when all the configurations are evaluated. However, these characteristics make the model difficult to parallelize, as BO is considered a sequential method. There are different Bayesian HPO approaches such as Gaussian process (GP) [SLA12], Random Forest (RF) also know as sequential model-based algorithm

configuration (SMAC) [HHLB11], Tree-structured Parzen Estimator (TPE) [BBBK11], and Bayesian Optimization HyperBand (BOHB) [FKH18].

- Metaheuristic algorithms: Metaheuristic algorithms [GT13], are computational intelligence paradigms used for solving sophisticated optimization problems. The strength of metaheuristics is their capability to solve no-continuous and non-convex optimization problems. One major subcategory of metaheuristics is population-based algorithms, such as evolutionary algorithms, genetic algorithms, PSO, and so on. In all the population-based algorithms, the process starts with generating and updating a population and evaluating the members of the population until the optimum member is discovered. The main difference between population methods is the way they generate the population or choose the individual in the population [YWC+18].

As mentioned earlier, HPO is one of the most important components of AutoML. Hence, there are many studies including survey studies in the field of HPO. In [KTH+19], the authors did a thorough overview of HPO approaches existing in the literature, and covered challenges, and future work directions. Yang and Shami in [YS20] also offered a very high-level survey study around search spaces, different HPO approaches, and tools that are decent for first-time users. In another study [And19], Andonie offers an overview of HPO techniques by focusing on computational complexity aspects, which is a useful source for experts in ML as well. There are also several specialized overview studies in the field of HPO and AutoML. Some of these surveys focus on AutoML for deep learning models [HZC21, Tal20]. In some other studies, authors focused on HPO for smart grids forecasting models [KJ20],on AutoML on graph models [ZWZ21]. There are also some other overviews of AutoML [YWC+18, EMS19, YZ20].

Model selection constitutes choosing the best-fitting model with optimal performance for future prediction [Ric15]. This can be done by finding and tuning valid hyper-parameters. Hyper-parameter tuning is an essential process to make a model perform at its best [ADNDS+19]. Previous research shows that tuning hyper-parameters of ML models can significantly improve prediction performance.
Although hyper-parameters are critical in the resulting predictive models' quality, they have no obvious agreeable defaults in different utilizations and applications. However, the tremendous increase in the scale of data in real life makes it computationally expensive and practically impossible to manually tune the hyper-parameters. Hence, it has become vital to automate optimizing the hyper-parameters. In HPO methods, the goal is to find the value of hyper-parameters that significantly contribute to improving the accuracy of a model. Therefore, the search algorithm looks through different combinations of hyper-parameters (search space), which enable the model to generate a well-performed model according to the evaluation criteria and through an iterative process [YS20, DFNNS17]. Figure 2.2 shows the general schema for hyper-parameter tuning. The hyper-parameter tuning step can be done manually or via some automated techniques.

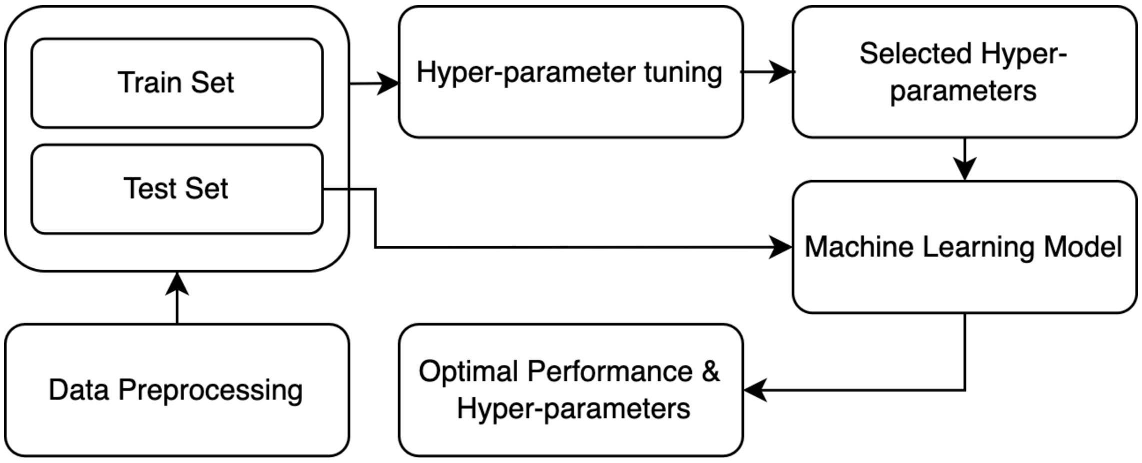


Figure 2.2: The general hyper-parameter tuning schema

HPO methods typically have four basic components:

1. An ML classifier or regressor
2. A search space
3. A search strategy to find optimal hyper-parameters
4. An evaluation function for performance comparison of different configurations.

In contrast to FSO problems where the domain of features is binary (only zeros and ones), in HPO problems, we deal with different types of hyper-parameters. Hyper-parameters could be continuous, discrete (including binary), or categorical. Depending on the value of a hyper-parameter used for a specific model, the value of another hyper-parameter may need tuning as well to have a well-performed model [Luo16b]. As mentioned in Section 2.1 optimization problems are divided into two categories: Unconstrained and Constrained. Although in some cases hyper-parameters can take real values, in most cases ML hyper-parameters take different ranges of values and have various constraints, so they are often complicated constrained optimization problems. Hence, in HPO problems the goal is to find x by solving the below problem [LNK+17]:

$$x^* = arg \min_{x \in X} f(x) \tag{2.4}$$

where $f(x)$ is the objective function we aim to minimize (such as root mean square error or the error rate) or maximize (such as accuracy or F1-score). x^* is also the set of hyper-parameters that generated the optimum value of $f(x)$.

2.2.4 Remarks

An important constraint that should not be overlooked in HPO and FSO problems is time resources. We usually need a lot of time to build an ML model with optimized function with the reasonable number of hyper-parameter settings or a subset of features. This is because each time one configuration of hyper-parameters/features is tested, the ML model requires to be trained to generate the model's performance. Finding the ideal set of hyper-parameters/features needs an exhaustive search over all the configurations and that is practically impossible. This is why most traditional optimization approaches are not a good fit for HPO problems; many of the optimization solutions are made for convex problems, while HPO problems are non-convex as they may find the local optimum instead of returning the global optimum [STH$^+$15]. The other reason is that many optimization problems are designed for continuous variables, while in HPO we are dealing with other types of hyper-parameters (categorical and discrete) as well. Last but not least, many of the optimization techniques ignore the function evaluation time, while they are very important in both HPO and FSO problems.

Therefore, there is a crucial need to develop effective techniques to find optimal solutions in a reduced search space and within a reasonable time.

In the next chapters, we cover how population-based algorithms, specifically Artificial Bee Colony (ABC) algorithms are used to improve the process of feature selection.

CHAPTER 3

OVERVIEW OF THE PROPOSED FRAMEWORK

Hyper-parameter tuning and feature selection are both key tasks in ML [BMTB20]. On one hand, as mentioned in previous chapters, the vast increase in the scale of data in real life makes it computationally expensive and practically impossible to manually adjust the hyper-parameters/features. Thus, automating the ML pipeline is crucial to ensure real-world deployment of these algorithms [KPZS20, Luo16a] On the other hand, although the availability of general Mathematical Optimization models provides tools to solve various optimization problems, the characteristics of an efficient optimization algorithm from the perspective of ML and mathematical optimization can be quite different [BPH06]. New challenges arise for researchers to explore new and more efficient models for ML problems. In fact, hyper-parameter or feature selection optimization is the process of finding optimal or near-optimal subset of hyper-parameters/features to maximize the objective function by using an optimization technique [ZGMA22, ZMA21c]. It is an emerging research area and has engaged many researchers in both ML and mathematical communities in recent years. Therefore, the goal of this work is to address some challenges in ML optimization, providing desirable properties of an optimization algorithm from the ML perspective, including [BPH06]:

- generalization,
- scalability to large data,
- reasonable performance regarding execution times,
- efficient implementation of an algorithm,
- exploitation of problem structure,

- reasonable convergence,
- robustness and numerical stability for class of ML models attempted,
- theoretically known convergence and complexity.

Currently, many research studies have been done in the field of AutoML to provide more efficient frameworks. The goal of this dissertation is to develop an algorithm to find the optimal features and hyper-parameters in conventional ML algorithms.

3.1 Theory

In this dissertation, an integrated framework for HPO and FSO is proposed. The sections below cover the proposed hybrid framework (which is the final outcome of this work) and each of its two components, namely, HPO and FSO.

3.1.1 A Hybrid Model for Automated Machine Learning

The final outcome of this dissertation is an end-to-end AutoML framework. The whole framework is shown in Figure 3.1, which consists of two major components: FSO and HPO. These components are integrated to partly address challenges in ML optimization and support functionalities in this area. In this framework, optimal features in the dataset are automatically generated using a novel artificial bee colony based algorithm to reduce the dimensionality of data and possibly improve the performance. Then, a novel hyper-parameter learning technique is utilized to further enhance the performance of model training.

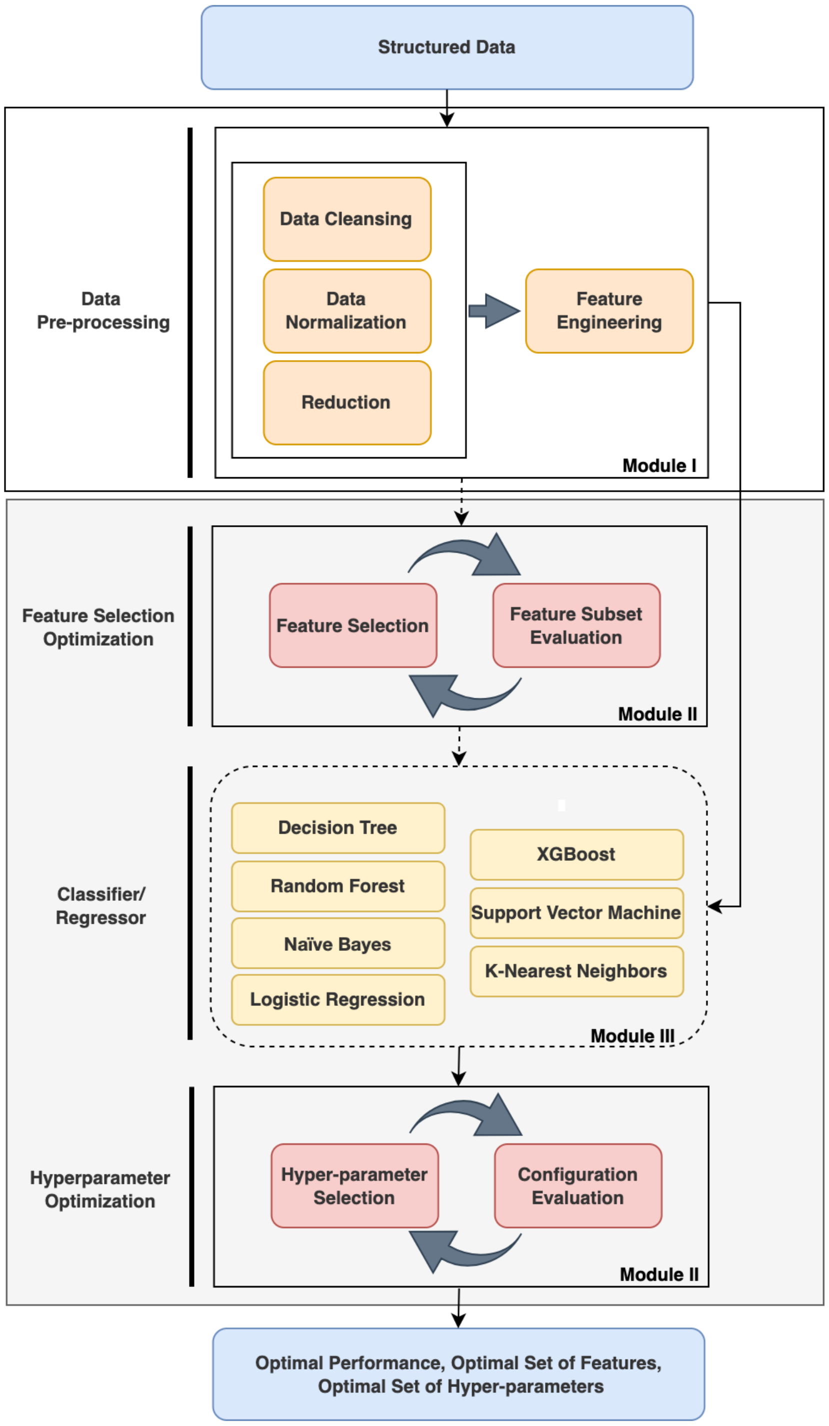


Figure 3.1: The proposed framework for HPO+FSO

3.1.2 Automated Feature Selection

In many fields such as ML and data mining, datasets have a large number of features. In these cases, feature selection is an essential step [GE03]. Feature selection entails selecting a subset of important and relevant features from the original dataset. This subset should be sufficient enough to predict the decision concepts as well as the original data. Feature selection is important in the sense that it reduces the dimensions of search space for search algorithms. Also, it can improve the performance and convergence rate of the algorithm [KS00].

Several search algorithms have been applied for feature selection [SB01, SHA19, RPA+13, OAE20, CS21] The most basic approach is finding the best subset of features by exhaustively going through the 2^n existing combinations, which could be very costly, especially for large datasets. It has been shown that this is an NP-hard combinatorial problem [SR92]. Hence, heuristic methods have to be considered. Although there are many search algorithms such as forward and backward feature selections, they have the disadvantage of high computational complexity and/or premature convergence [LY05]. To mitigate these problems, Evolutionary Algorithms (EAs) that are population-based meta-heuristic optimization algorithms have been applied. The advantage of EAs is their strength in global search. EAs are the techniques usually applied for feature selection approaches, and they usually produce reasonable results as they provide several trade-off solutions in a single run.

Artificial Bee Colony (ABC) is one of the most recently introduced EA techniques with a lot of successful applications to solve various problems. Moreover, the implementation of ABC is easy and has the ability to search for local and global solutions. However, ABC has some disadvantages such as low average accuracy for the exploration phase or a low convergence rate. The potential of improved versions of the ABC algorithm for wrapper feature selection, which addresses these challenges, has

The main goal of the FSO component in this dissertation is to propose a novel ABC-based selection that can find the optimal or near-optimal features without decreasing the performance. To fulfill this goal, this dissertation presents a novel feature selection method. A2BCF is proposed for a challenging real-world application and is based on clustering, opposition-based learning, and early stopping concepts:

- **Clustering:** One of the disadvantages of swarm intelligence algorithms including ABC is that the initial population has an impact on the final performance of the model. Hence, the better the quality of the population, the better the final performance of the algorithm. For very large search spaces, the diversity of the population is important, so that some search areas are not overlooked during the search process. In this dissertation, we use a clustering learning algorithm for decomposing the population for three main purposes:

    1. Preventing the algorithm from training each member of the population and saving time. Therefore, from each cluster, only the representative of the cluster is trained (assuming that the rest of the cluster members are similar to the representative)
    2. Provide diversity to the population by dividing the search area and searching all the divisions in order to prevent overlooking some quality division, and
    3. Improving the speed of the algorithm.

- **Early elimination of poor solutions:** As mentioned above, the quality of the population has an impact on the performance and speed of the algorithm. To further improve the speed of the proposed algorithm, it begins by filtering poor solutions from the population so that the algorithm starts the searching process on a quality population. In this dissertation, removing the worst solutions from the bee swarm helps the search agents spend more time around the potential solutions.

- **Opposition-Based Learning (OBL):** One of the disadvantages of ABC algorithms is its low average performance of the exploration phase in comparison to the exploitation phase. Opposition-based learning can enhance the convergence rate of the algorithm when the optimal point is close or in an opposite location. This method searches the opposite direction and other directions to improve the convergence ability [Tiz05]. It also improves the average performance of the exploration phase.

Figure 3.2 demonstrates the proposed pipeline, starting with data pre-processing and feature engineering, in which we clean up and normalize the data or transform some of the features. The next step is Wrapper FSO, which is an iterative optimization process of selecting features, where selected ML models operate as black-box. The output of this framework is the optimal subset of features along with the optimal performance.
The experimental results demonstrate the effectiveness of the proposed models compared to previous research. This technique increases the performance and convergence rate of the algorithm and reduces the structure's size of search space.

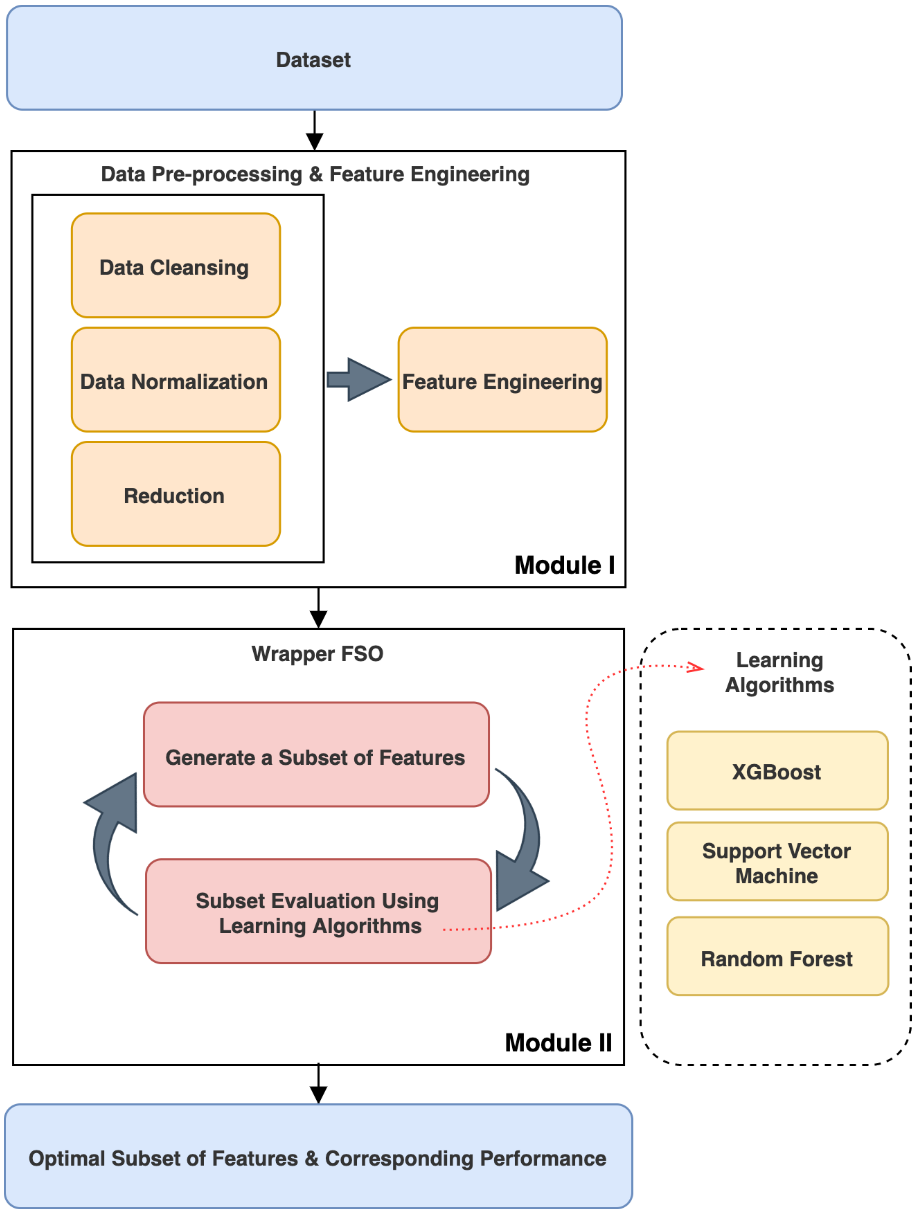

Dataset
Data Pre-processing & Feature Engineering
Data Cleansing
Data Normalization
Reduction
Feature Engineering
Module I
Wrapper FSO
Generate a Subset of Features
Subset Evaluation Using Learning Algorithms
Module II
Learning Algorithms
XGBoost
Support Vector Machine
Random Forest
Optimal Subset of Features & Corresponding Performance

Figure 3.2: Feature selection optimization component

3.1.3 Automated Hyperparameter Tuning

Hyper-parameters are the set of variables that are common between different ML algorithms, and users should set them in a way to ensure the model works at its best [CDM15]. Manually tuning the hyper-parameters [ZLP+20] or using the grid search over a reduced search space is a common practice [PVG+11]. However, these methods are impractical when the number of hyper-parameters increases [CSP+14]. Hence, automating the search process to tune the hyper-parameter has gained a lot of attention in the field of ML. Previous research shows the superiority of automated search strategies over manual search [BB12].

Despite the existing work on hyper-parameter optimization algorithms, there are still concerns regarding the trade-off between efficiency and effectiveness of current approaches [YS20]. One of the reasons is due to expensive function evaluations. The function evaluation depends on factors such as the data scale, model's complexity, and availability of computational resources. Each of these factors directly impacts the evaluation of each HP configuration, which can lead to high computational time [CDM15].
Although exhaustive methods, such as grid search and random search, have gained some success in finding near-optimal solutions, given the large-scale nature of the HPO problems, these approaches are impractical and expensive. Despite the success of other HPO methods existing in the literature, many of these methods entail a large number of function evaluations before finding the optimal results, which makes utilizing such methods for HPO problems challenging. Therefore, an efficient hyper-parameter optimization method should be able to address the challenge concerning the trade-off between efficiency and effectiveness, commonly seen in most HPO methods.

For the HPO component of this dissertation, a novel HPO algorithm, OptABC, is proposed to improve the performance and address the challenges mentioned in the previous section. OptABC is an ABC-based algorithm using learning algorithms, namely clustering, and OBL. These two learning algorithms are developed to address some challenges in HPO problems. As mentioned in the previous section, these two modifications enhance the performance of the algorithm and improve the convergence rate, as it provides diversity to the population and prevents the algorithm from training every single set of hyper-parameters. Also, OptABC strengthens the exploration phase of the algorithm by leveraging OBL. Then, the algorithm selects the better solution between a random solution and its opposite instead of only generating a random candidate.

Figure 3.3 shows the proposed pipeline for automated hyper-parameter tuning. The framework starts from data pre-processing and feature engineering. Subsequently, ML models operate as black-box, and an iterative process for selecting the best parameters on each ML algorithm starts. The output of this framework would be the optimal hyper-parameters of the ML algorithm and the optimal performance.

As can be seen from Figures 3.1, 3.2 and 3.2, a few steps were completed before the actual optimization in FSO/HPO experiments start. These steps include are considered as pre-processing steps which are required to transform the data into an understandable state for the ML algorithms. Tasks such as data cleaning, data normalization, data reduction, and feature engineering are included in this step. Each of these steps is summarized below:

- Data Cleaning: In the data cleaning step, which is a step for removing the corrupt and not applicable data, we remove the features with more than 60% missing values.
- Data Normalization: This step is for standardizing the range of features of data so that the data appear similar across all records and fields.
- Reduction: Data reduction, in this dissertation, refers to using a subset of the dataset rather than the entire dataset. Filtering and sampling are among the most common ways of data reduction.

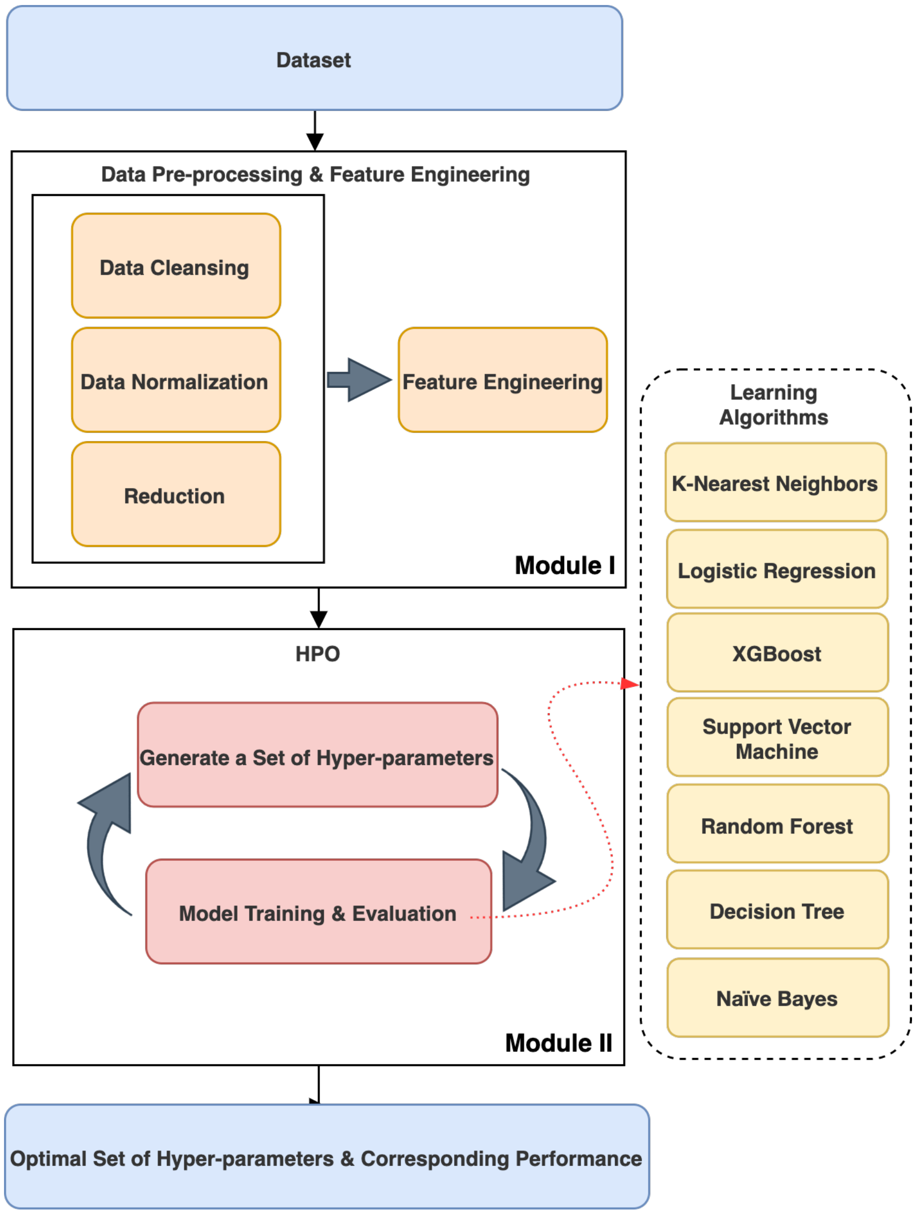


Figure 3.3: Hyper-parameter optimization component

- Feature Engineering: In this step, we create some new features from raw features that might be useful for the power of prediction. Moreover, one-hot encoding (setting up dummy variables for encoding categorical data) is used so that we can feed the data into ML algorithms.

The pre-processing steps mentioned above, are applied to a few datasets with different applications. A summary of these applications is covered in the next section.

3.2 Applications

In order to evaluate the effectiveness of the proposed approach, we selected two datasets in two different applications (including educational, engineering, housing, and image classification) as the benchmark datasets for HPO and/or FSO evaluations. MIDFIELD and MNIST datasets are to solve a classification problem and the PJM and Boston-Housing datasets are used to solve a regression problem. MIDFIELD dataset is used in all the chapters of this dissertation as the baseline dataset. In Chapter 7, all four datasets are used to evaluate the framework. However, for one of the datasets (PJM), the framework is only tested on HPO as the number of features in the dataset is small enough. In both cases, we first shuffled the datasets. The validation performance for the optimization process was then computed by 3-fold cross-validation on the training dataset. To use the same dataset for each classification/regression algorithm, we used one-hot encoding which creates binary columns for each of the categories. We also scaled numerical features linearly to the range [0, 1] by subtracting the minimum value and dividing by the maximum.
The list of the datasets used in each of these applications is given below:

- Classification

  MIDFIELD: An educational dataset to predict computing students' success in computing majors [LOO+22].

  MNIST: A Modified National Institute of Standards and Technology dataset to predict the handwritten digits [LeC98].

- Regression

PJM: An hourly electricity load data to predict the power demand [pjm].

Boston Housing: Is derived from information collected by the U.S. Census Service concerning housing in the area of Boston MA, to predict the value of housing prices [HJR78].

As it is shown in 3.1, the AutoML process in this study starts with the FSO and then based on the selected features, the hyper-parameter tuning starts. Hence, the next chapter covers the feature selection process. In chapter 5 we will discuss the hyper-parameter tuning approach and the details of the proposed approach.

CHAPTER 4

A2BCF: AN AUTOMATED ABC-BASED FEATURE SELECTION ALGORITHM FOR CLASSIFICATION MODELS IN AN EDUCATION APPLICATION

Abstract[1] Feature selection is an essential step of preprocessing in Machine Learning (ML) algorithms that can significantly impact the performance of ML models. It is considered one of the most crucial phases of automated ML (AutoML). Feature selection aims to find the optimal subset of features and remove the non-informative features from the dataset. Feature selection also reduces the computational time and makes the data more understandable to the learning model. There are various heuristic search strategies to address combinatorial optimization challenges. This chapter develops an Automated Artificial Bee Colony-based algorithm for Feature Selection (A2BCF) to solve a classification problem. The application domain evaluating our proposed algorithm is education science, which solves a binary classification problem, namely, undergraduate student success. The modifications made to the original Artificial Bee Colony algorithm make the algorithm a well-performed approach.

Keywords: AutoML; Artificial Bee Colony; classification; education; evolutionary computation; feature selection; optimization; swarm intelligence; wrapper method; student success

4.1 Introduction

Student retention is a major concern for STEM fields. It is particularly problematic in computing fields, where enrollment has not kept pace with industry demands.

[1]This chapter is an edited version of the author's previous work published in [ZGMA22] ©2022 MDPI.

Therefore, finding patterns in historical educational data can help the education community reveal the potential reasons for students' withdrawal from the computing field. This information provides guidelines to better understand the relative success of computing students and enable strategic solutions to achieve higher retention rates [ZLP+20].

The application of ML in education is currently very interesting to researchers and education communities. Building predictive models involve learning from data. Machine Learning (ML) is a technology that enables computers to learn without being explicitly programmed. One of the most important recent discussions in this field is related to the enhancement of ML algorithms' performance in different applications, such as education. In the context of ML, choosing the best model has always been a concern. Moreover, it is crucial to transfer the most important information to both save time and improve performance. In general, based on the principle of Occam's razor, "a model should be simple enough for efficient computation and complex enough to be able to capture data specifics" [Bie03].

The extraction of useful information and the presentation of scientific, educational decision-making is necessary, providing professionals with an additional source of knowledge. The educational datasets can be massive. The high-dimensional nature of many modeling tasks has given rise to a wealth of feature selection techniques.

Feature selection is a way of removing noise and random errors in the underlying data. In feature selection, we use techniques to select features that are more relevant to the problem and withdraw redundant or irrelevant data without incurring much loss of information [BPWS+15]. Feature selection facilitates the best performance for the ML model [KM14]. The main idea of feature selection is to select a subset of features for the model to either improve the model's performance or reduce the structure's size and computational cost [oH03]. In [KS96], Koller and

Sahami presented a filter feature selection method to reduce the overall computation cost. In Ahmed et al. [AYA09], authors concluded that information entropy can determine the importance of features and can be effective for the reduction of attributes. Choosing K out of N features helps the training time to be distributed by N/K [MA14]. In feature selection problems, the size of the search space for a dataset with N features would be 2^N [WBA09]. Therefore, researchers have provided different search methods to reduce models' computational costs. Dash and Liu believed these methods typically have four basic steps: (1) a generation procedure to generate the next candidate, (2) an evaluation function, (3) stopping criteria, and (4) a validation procedure [DL97]. Cleaning and featurization of the data are considered the most time-consuming steps of AutoML, and researchers have investigated different methods to improve the performance of the AutoML process [GYR+18].

There are various heuristic search strategies to address combinatorial optimization challenges. These algorithms include but are not limited to, nature-inspired algorithms such as Genetic Algorithms (GA) and Swarm Intelligence (SI), which include Particle Swarm Optimization (PSO) [SC14] and Artificial Bee Colony (ABC) [EA10].

This chapter proposes a new approach (A2BCF) for feature selection using an ABC algorithm to improve the performance of the classifier in an education application and predict students' success. The goal of A2BCF is to choose a subset of features from the whole feature space, reducing the total number of used features to improve the total computation time. It is specifically helpful for ML algorithms, such as support vector machines, as their training time is highly dependent on the dimensions of the data.

Our main contributions are listed as follows:

- A2BCF is a novel framework for efficiently selecting the features in a large structured dataset.
- The application of the developed optimization method is used for the first time in this chapter.
- We design a variant of the ABC algorithm that helps balance the exploration and exploitation phases of the ABC algorithm.

The remainder of the chapter is organized as follows: Section 4.2 covers a summary review of feature selection. Section 4.3 presents the original ABC. Section 4.4 introduces ABC-based feature selection, our modifications, and the proposed algorithm. Following that, Sections 4.6 and 4.7 discuss the experimental methodology and results. Finally, Sections 4.8 and 4.9 concludes this work.

4.2 Related Work

Recently, there has been an increase in the number of research studies that use dimensionality reduction as a preprocessing step to separate the noise and unimportant data from the important data. These methods include both feature selection and feature extraction methods. Feature extraction methods are the techniques used to extract new features from the data and are mostly used in image processing techniques. However, feature selection techniques are approaches that select an optimal subset of features from existing features with minimum error and information loss [MBN02]. Unlike feature extraction techniques, feature selection methods are leveraged to structured datasets that have identified features. Therefore, this chapter focuses on feature selection methods.

Feature Selection

Feature selection means finding a subset of features from the feature search space. Different techniques for feature selection include sequential forward selection (SFS) or sequential backward selection (SBS). The former begins from the minimum number of features and adds features during the selection process, while the latter starts from the whole feature set and discards the irrelevant feature during the selection process [KAK$^+$05, MCQDCJA10]. At the end of the process, the remaining subset of features in both methods is considered the optimal subset. Recently, there has been an increase in robust approaches based on evolutionary algorithms that decrease the number of features. In [FR07], Faraoun and Rabhi used GA for dimensionality reduction, improving the accuracy of the classification process. In another study, Aghdam et al. [AGAB09] proposed an Ant Colony Optimization (ACO) approach for the feature selection process. This study shows that ACO demonstrates lower computational complexity than stochastic algorithms and GA.

On the other hand, there are two major categories of feature selection, Filter and Wrapper methods [LY05]:

- ***Filter methods:*** In these methods, selecting the features is performed before leveraging the learning algorithm. Therefore, the features do not depend on the ML algorithm. These methods, the importance of features is calculated (based on some predefined criterion), and then the best feature subset is selected. In these methods, no learning algorithm is used for choosing the features. The advantage of these methods is their faster speed.

- ***Wrapper methods:*** Unlike filter methods, wrapper methods generate different subsets of features by adding and removing features to achieve reasonable accuracy. The predictive accuracy of the classifier is used to evaluate the subset of features. The advantage of these methods include their high classification accuracy. However, their computational complexity is higher than filter methods. Therefore, researchers have been exploring different methods to enhance the convergence of wrapper methods. Wrapper methods have gained much attention due to their promising performance. Prior works have used wrapper methods for selecting the optimal features [OSBS03, PX13, SIM11, YO11, MA14].

There has also been an increase in the use of evolutionary algorithms, specifically SIs, for the feature selection process in the past few years [RR15, ZEPS16, NGS+14, MA14]. Another wrapper feature selection method was proposed by Zawbaa et al. and Ng et al. using Bat Optimization algorithms [ZEPS16, NGS+14]. ABC algorithm is also an SI algorithm that has been used for feature selection problems [MA14].

4.3 Artificial Bee Colony (ABC) Algorithm

ABC, initially introduced by Karaboga [KB07], is one of the most recent SI methods that simulate the foraging behavior of honey bees. In ABC, the bee colony has three groups of bees: (1) employed, (2) onlookers, and (3) scouts. **Employed bees** have the responsibility of exploiting the food sources and sharing the information they gather with **onlooker bees**. Onlooker bees may or may not select a food source based on the received information. The higher the quality of a food source, the more it is likely to be chosen by the onlookers. Onlookers are also responsible for exploitation. On the other hand, **Scout bees** control the exploration process. If the scout bee finds a quality food source, it becomes an employed bee. In contrast, if a food source becomes thoroughly exhausted, the involved bee becomes a scout.

Although Scout bees have low investigation cost, the mean of the food sources they find is low [MA14]. Algorithm 7 shows the main steps of the ABC search strategy inspired by [MA14].

Algorithm 1 Search process by Artificial Bee Colony

```
1: Initialize the population (size=PS)
2: Evaluate the population
3: **while** Stopping criterion is not met **do**
4:     Assign each food source to an employed bee for exploitation in the neighbor-
       hood
5:     Onlooker bees choose food sources based on the information shared
6:     Scout bees search the area randomly to find quality food sources
7:     Memorize the best food source
8: **end while**
9: **return** Return best food source
```

As can be seen in the algorithm, the first steps include initialization and evaluation of the population. Next, employed bees start exploitation in the neighborhood using Equation (4.1).

$$V_{i,j} = X_{i,j} + rand(-1,1)(X_{i,j} - X_{k,j}) \tag{4.1}$$

where k is an index for one of the features and $rand(-1,1)$ is a random number uniformly distributed in the range $[-1,1]$. Then, the better food source between $V_{i,j}$ and $X_{i,j}$ is chosen using greedy selection. Afterward, employed bees share the gathered information with onlooker bees. Onlooker bees start calculating the probability values using roulette wheel selection, as shown in Equation (4.2).

$$P_i = \frac{f_i}{\sum_{j=1}^{PS}(f_j)} \tag{4.2}$$

where

$$fit_i = \begin{cases} \frac{1}{1+f_i}, & f_i \geq 0 \\ 1 + abs(f_i), & f_i < 0 \end{cases} \tag{4.3}$$

Equation (4.3) shows how the fitness (fit_i) is calculated using the objective function (f_i). Since the performance of the classifier is always positive, we always have the first function in the equation. Therefore, in this problem, we are maximizing the objective function or, in other words, we are minimizing the fitness. If onlooker bees select a food source, Equation (4.1) and greedy selection are used again to generate and choose new food sources.

In the ABC algorithm, the scout bee is chosen from employed bees. This selection is made based on a control parameter called **limit**. Therefore, if a specific food source is not enhanced until a fixed number of iterations (limits), that food source is abandoned (or exhausted) by its employed bee and the employed bee turns into a scout. The abandoned food source is then replaced with a randomly generated food source using Equation (4.4) that may lead to discovering rich, unknown food sources.

$$X_{i,j} = x_{min} + rand(0,1)(x_{max} - x_{min}) \tag{4.4}$$

In the end, the location of the best food source is memorized and the algorithm starts from the beginning. This process continues until the stopping criterion is met. The stopping criteria could be a threshold for different metrics such as desirable performance, maximum number of evaluations/iterations, or time.

4.4 Feature Selection Using ABC

As mentioned in the previous section, in a feature selection problem using ABC, each solution (candidate food sources) is a vector with size N, where N equals the number of features in the dataset. This vector is a bit vector that only consists of 0 and 1. If the value of a specific position (feature) is 1, the feature is to be included in the evaluation. On the other hand, if the corresponding value is 0, the feature would be excluded from the assessments.

Mohammadi et al. [MSAA19] motivated researchers to apply nature-inspired algorithms and evolutionary algorithms to solve large-scale optimization problems, specifically feature selection processes. In feature selection problems using ABC, the quality of each food source equals the accuracy of the classifier over the validation set with the corresponding subset of features.

Since the original ABC is only applicable to continuous problems, proper strategies should be adopted to transform it into a binary problem. The major steps to implement the ABC-based feature selection are as follows:

1. **Create initial population:** First, a population of food sources is generated. Each food source in the population is a bit vector consisting of 0s and 1s.
2. **Calculate fitness:** Each solution is submitted to the ML classifier and the accuracy is calculated. The accuracy is then saved as the fitness of the solution.
3. **Exploitation by employed bees:** An employed bee takes each bit vector to regulate the neighbor food sources (bit vector). SFS or SBS are the common approaches used by an employed bee in this step to find better solutions [NRS21].
4. **Exploitation by onlooker bees:** The information about the performance of solutions is shared with onlookers, and they select the subsets with a better probability of exploration. Then, the chosen bit vector goes under the same process as step 3 to become exploited.
5. **Memorizing the best food source:** After all the onlookers are done with their parts, the subset with the best quality is memorized.

6. **Exploration by scout bees:** In this step, a scout bee is assigned to deserted food sources, if any, and a bit vector of size N is generated randomly. This vector is then submitted to the ML classifier and its accuracy is stored.

The general structure and foremost steps of ABC-based feature selection are given in Figure 4.1.

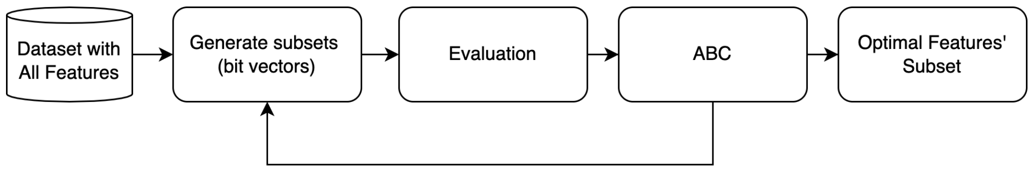


Figure 4.1: The general schematic approach to ABC-based feature selection.

To improve the efficiency of the original ABC approach, we made some further modifications to the algorithm. The main goal of this work is to provide optimized feature selection with a better accuracy rate. This work obtains the optimized feature subset by using Automated Artificial Bee Colony Feature Selection (A2BCF). The A2BCF algorithm selects the optimized features, and the efficiency is calculated using different ML classifiers.

One disadvantage of SI algorithms, including the ABC, is their premature convergence. These algorithms would be more efficient if their convergence rate were faster. The other challenge is the dependency of the algorithm on the initial population. In other words, if the proper initial population is selected, the algorithm reaches the optimal solution in a more reasonable time. There is an increased number of research studies exploring these challenges in order to enhance the convergence rate of these algorithms [KGOK14]. Balancing between the exploration and exploitation phases of such algorithms increases the efficiency of the algorithm. Therefore, ABC still has space for improvements. Karaboga and Akay [KA09] compared different variants of ABC algorithms and determined that ABC has relatively poor performance in the exploration phase. Hence, we consider improving the ABC algorithm's initial population and scout bee phase to design a more efficient ABC variant.

4.5 Proposed Feature Selection Method (A2BCF)

A2BCF algorithm is a novel ABC algorithm for feature selection, inspired by the OPT-ABC algorithm proposed in [ZMA21c] for hyper-parameter tuning purposes. The proposed algorithm returns the ideal set of features that increases the classifier's accuracy. Figure 4.2 shows the schematic flow chart of A2BCF. In this algorithm, the selected classifier is leveraged on every bit vector of features (features' subset). Since the accuracy depends on the learning algorithms, the method is wrapper-based.

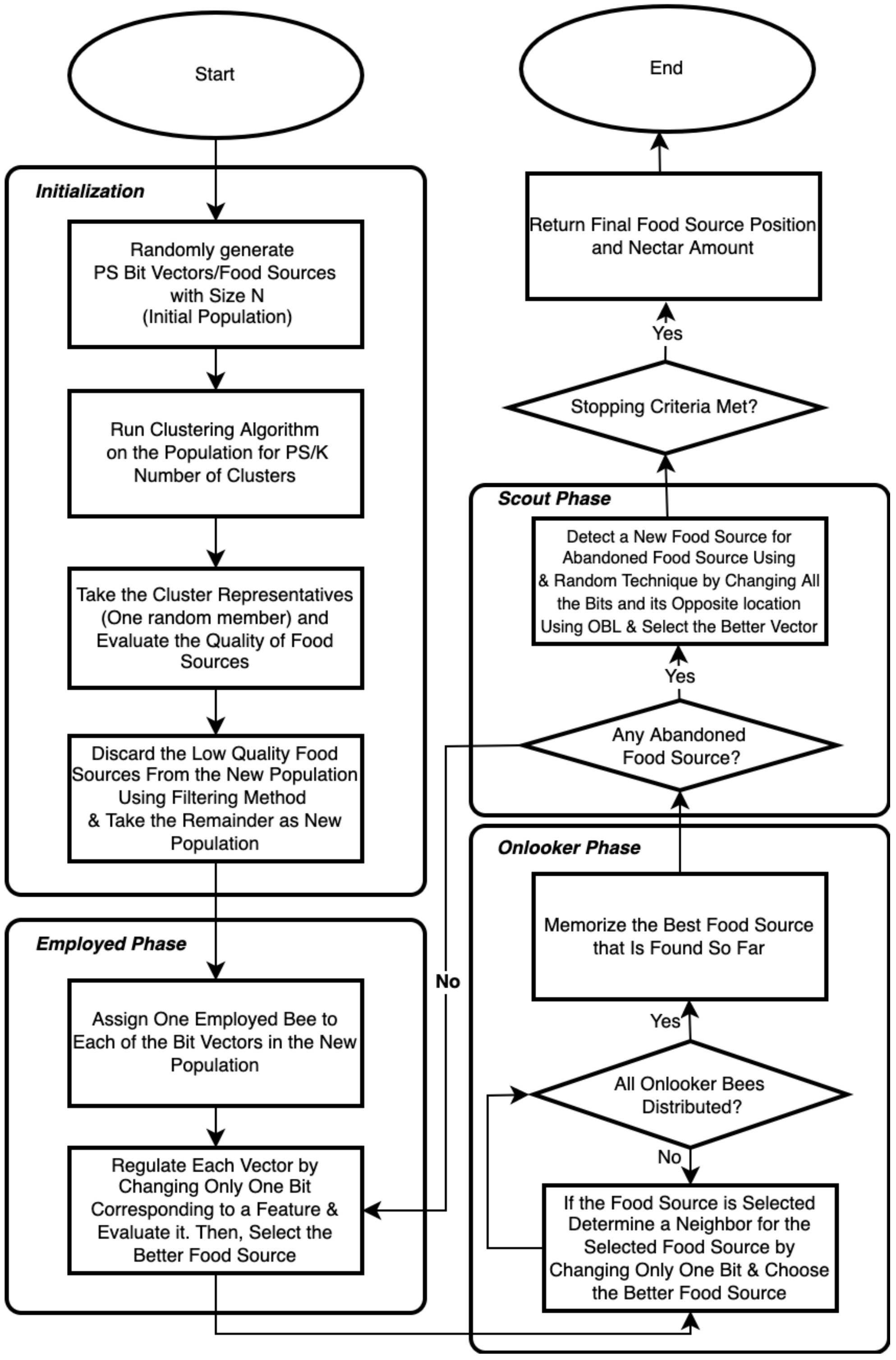

Start
End
Initialization
Randomly generate PS Bit Vectors/Food Sources with Size N (Initial Population)
Run Clustering Algorithm on the Population for PS/K Number of Clusters
Take the Cluster Representatives (One random member) and Evaluate the Quality of Food Sources
Discard the Low Quality Food Sources From the New Population Using Filtering Method & Take the Remainder as New Population
Employed Phase
Assign One Employed Bee to Each of the Bit Vectors in the New Population
Regulate Each Vector by Changing Only One Bit Corresponding to a Feature & Evaluate it. Then, Select the Better Food Source
Onlooker Phase
All Onlooker Bees Distributed?
No
If the Food Source is Selected Determine a Neighbor for the Selected Food Source by Changing Only One Bit & Choose the Better Food Source
Yes
Memorize the Best Food Source that Is Found So Far
Scout Phase
Any Abandoned Food Source?
No
Yes
Detect a New Food Source for Abandoned Food Source Using & Random Technique by Changing All the Bits and its Opposite location Using OBL & Select the Better Vector
Stopping Criteria Met?
Yes
Return Final Food Source Position and Nectar Amount

Figure 4.2: The flowchart of the proposed Automated ABC-based Feature Selection (A2BCF) algorithm (stopping criterion could be set to the maximum of number of evaluations or iterations)

.

The steps of the A2BCF are explained as follows:

- **Initialization:** Only some solutions are evaluated in this step instead of all the random initial population members. Hence, an agglomerative clustering technique using the cosine similarity function is first employed to group the similar or repetitive members of the initial population into the same cluster. As shown in Algorithm 2, we create a population of random food sources (bit vectors). Subsequently, only the representatives (one random member) of each cluster are taken as our new population to be evaluated by the objective function. In other words, the new population is a diversified sample of bit vectors from the original population. The clustering method (agglomerative clustering) used in this study is the most common type of hierarchical clustering and, in our case, the cosine similarity function is used to group the similar objects. Therefore, the most similar bit-vectors belong to the same cluster. This approach is inspired by Chavent [Cha98] for grouping the binary data samples.

  After taking all clusters' representatives, inferior food sources are discarded (based on a predefined threshold) from the new population. This is performed by filtering the low-quality representatives to avoid further evaluation of poor food sources. An A threshold of 50% is considered in this study to filter the low-quality food sources and focus on the food sources that have the potential to improve in a lower number of iterations. As explained in the feature selection steps using ABC, the algorithm changes only one bit of the vector to exploit the search space. Hence, if a bit vector has poor quality, it takes longer to exploit the food source and reach a high-quality food source. Therefore, early stopping of those food sources helps improve the convergence rate of the exploitation phase.

Classification accuracy is computed using 3-fold cross-validation to improve the classifier's reliability. In k-fold cross-validation, the dataset is divided into k equally sized folds and the ML algorithm is executed three times. Cross-validation is a preferred method over a single train–test split that trains on multiple trains–test splits and provides a better insight into how well a model performs on unseen data. Equation (4.5) below shows that we aim to minimize the classification error rate. The accuracy is used in the fitness function to evaluate the quality of food sources.

$$fit_i = \frac{1}{1 + f_i} \tag{4.5}$$

and

$$f_i = Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.6}$$

where TP, TN, FP, and FN stand for true positives, true negatives, false positives, and false negatives, respectively. The reason for selecting accuracy as the performance metric is that the Multiple-Institution Database for Investigating Engineering Longitudinal Development (MIDFIELD) dataset used in this study is balanced and has an almost equal number of observations for both classes (54:46 ratio). In binary classification problems, a balanced dataset is one where positive values are approximately the same as negative values (in our case, the number of students who graduated/dropped out from a computing field). However, in cases dealing with imbalanced data, other measures such as F1 score are more appropriate and can be used instead of accuracy.

- **Employed bee phase:** Each bit vector is taken by an employed bee, where it regulates the food sources (bit vector) by flipping only one of the bit values in the vector. If the neighboring (new) vector has a better fitness, it gets replaced with the previous vector, and its corresponding accuracy gets stored. In other words, the employed bee evaluates the model's accuracy by including (bit 1) or excluding (bit 0) only one of the features and passing the required information to onlookers.

- **Onlooker bee phase:** The information about the quality of the food sources is shared with the onlookers, and they then select the food sources with a better probability of exploration (based on Equation (4.7)). Then, selected food sources go under the same process as step 3 to become exploited. After all the onlookers are done with their parts, the food source with the best quality is memorized.

$$P_i = \frac{Fit_i - min(Fit)}{max(Fit) - min(Fit)} \tag{4.7}$$

- **Scout bee phase:** This phase has an additional step in A2BCF. In original ABC, a scout bee (assigned to the deserted food source, if any) generates a random novel food source of size N. While onlooker/employed bees change only one bit of the bit vector, scout bees change all the bits in the vector. As mentioned earlier, scout bees have a low mean in the food source quality that they find. Therefore, we added another step in which Opposition-Based Learning (OBL) is submitted to the abandoned food source to generate an opposite food source location based on Equation (4.8). Then, the algorithm moves forward with better food sources from random and OBL methods. The generated bit vector is then submitted to the ML classifier and its accuracy is stored.

$$\tilde{x}_{i,j} = 1 - x_{i,j} \tag{4.8}$$

- **Termination process:** The employed, onlooker, and scout bee phases will continue until the algorithm reaches the defined maximum number of evaluations/runs. In this study we set the maximum number of evaluations to 5000.

Algorithm 2 Agglomerative Clustering Algorithm for Initialization Phase

```
   Input: Dataset, $K = 100$, Population size (PS)
   Output: Population (Cluster centroid/representative)
 1: Randomly generate a population of size $PS$, with food sources of size
    length($Features\_set$)
 2: # Clustering the random population to k clusters
 3: Compute the proximity matrix (cosine similarity matrix)
 4: Let each vector be a cluster
 5: while The remaining clusters are more than PS/k do
 6:    Merge the two closest clusters
 7:    Update the matrix
 8: end while
 9: Take cluster representatives (one random member of each cluster), where k is
    defined as the new population
10: return population
```

Algorithm 3 presents the pseudocode of the A2BCF algorithm explained above.

Algorithm 3 Proposed Algorithm—A2BCF

Input: *threshold* = 50, *limit* = 3, max_{eval} = 5000, $trial_i$ = 0 **Output:** Optimal features subset

```
 1: Call Algorithm 2 (PS = 2000; 5000; 10,000)
 2: Discard the food sources with poor quality($<$ $threshold = 50$) from new population
 3: while Stopping criteria is not met(number of evaluations ¡$max_{eval}$) do
 4:     for $i$ to $population\_size$ do
 5:         Employed bee regulates the bit vector and find a new vector $N_i$ in the neighborhood. This is achieved by changing only one bit in the vector
 6:         Train the ML model with the novel vector (subset of features selected)
 7:         if the new vector has a better quality then
 8:             Replace the new vector with the original vector
 9:         else
10:             $trial_i$ +=1
11:         end if
12:     end for
13:     for $j$ to $population\_size$ do
14:         Onlooker bee calculates the exploration probability ($P_i$) according to Equation (4.2)
15:         if $rand(0, 1) > P_i$ then
16:             Onlooker bee regulates the current vector $N_i$ by changing only one of the bits in the vector, which gives a new vector in the neighborhood $M_i$
17:             Train the ML model with the updated features subset
18:             if the new vector has a better quality then
19:                 Replace the new vector with the original vector
20:             else
21:                 $trial_i$ +=1
22:             end if
23:         else
24:             Onlooker disregards the features subset and moves to the next bit vector
25:         end if
26:     end for
27:     Memorize and update the best subset so far
28:     if $trail_i > limit$ (food source is exhausted) then
29:         Scout bee generates a "Random" bit vector based on Equation (4.4)
30:         Scout bee generates the "OBL" bit vector based on Equation (4.8)
31:         Select the subset that gives better fitness between Random and OBL vectors
32:     end if
33: end while
34: return Optimal subset of features
```

4.6 Experimental Setup

This section presents the study's methodology, then the performance of our proposed algorithm, A2BCF, and comparison of the results with a previous study [ZLP+20] are covered in the next section. A2BCF is a wrapper feature selection approach to improve the classification accuracy and enhance the search to keep relevant features in a reasonable time. The database used for this experiment contains around 45,000 observations, and the number of classes is two. The overall structure of the whole experiment is presented in Figure 4.3. The A2BCF algorithm proposed in this study is inspired by a Hyper-parameter Optimization (HPO) method proposed in 2021 [ZMA21c].

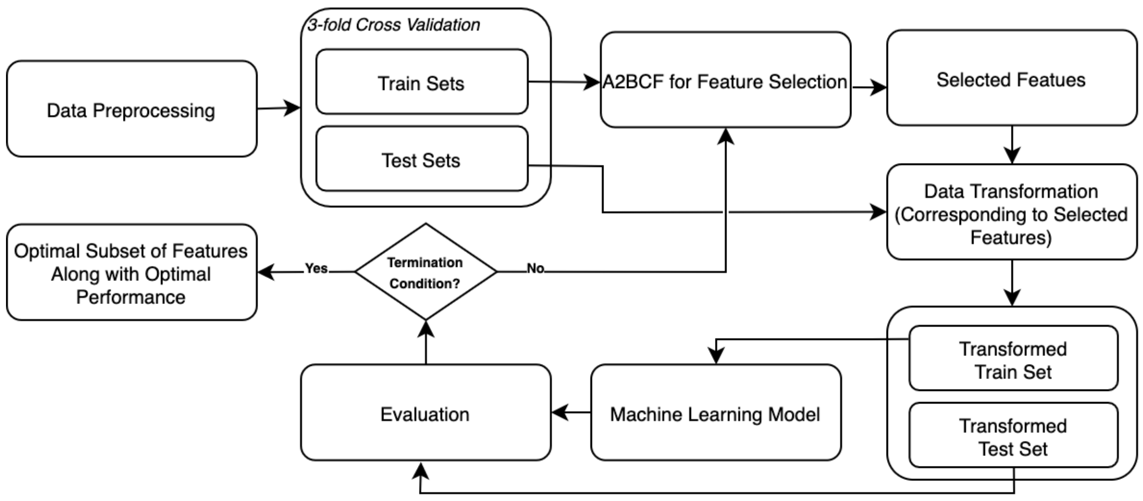


Figure 4.3: Structure of the proposed ABC-based Feature Selection (A2BCF) method

As mentioned above, we leveraged several ML classifiers to select the most important features of the dataset. The detail of the classifiers and the feature selection steps are described below.

4.6.1 Main Steps

The overall process of feature selection is done in several steps, including data preprocessing and Feature Selection Optimization (FSO), data transformation, training the transformed data. The details of these steps are explained below.

- **Data pre-processing :** This step is required to transform the data into an understandable state for the ML classifier. Tasks such as data cleaning, feature engineering, and feature scaling are included in this step. Filtering could also be considered as one of the data cleansing steps in which we included only students from computing majors to predict their success in their programs [ZLP+20]. In this study, one-hot encoding the variables is used for encoding the categorical types to their binary representations.
- **Feature selection optimization using A2BCF:** Feature selection is a primary task in automated ML that helps the model achieve two main goals: better performance and reduced computational time. This step is the study's main contribution, an iterative ABC-based process to find the optimal set of features in a dataset. The proposed algorithm (A2BCF) is an improved version of the Opt-ABC algorithm [ZMA21c] used for optimizing the hyper-parameter tuning of the ML algorithms [ZMR+21, ZMA21a]. The output in this step is a subset of features selected by the algorithm.
- **Data transformation:** This process converts the data to the required format of our destination system. In other words, it converts the original dataset to a new dataset with the selected feature from the feature selection process in the previous step. The new dataset is then divided into train and test sets as input for the next step.

- **Train the transformed data:** In this step, the desired ML algorithm is submitted, and the ML algorithm gets trained. Finally, the accuracy of the ML algorithm and final features are returned.

4.6.2 Machine Learning Classifiers

A summary of the ML classifier used in this experiment is as follows:

1. **Random Forest (RF)** [Bre01] is an ensemble method and a type of Decision Tree (DT) learner that operates by constructing many DTs in the training phase. That is the reason it is called 'forest.' The term 'random' is also because the trees are built differently with random samples and random features to add diversity to the models and decrease the chance of overfitting [Bre01, LW$^+$02, Lou14]. Random forest initially uses the bagging method to combine the predictions from each tree and calculate the overall predictions.

2. **Extreme Gradient Boosting (XGBoost)** [CHB$^+$15] is a developed version of Gradient Boosting that utilizes a gradient boosting framework as an ensemble. One of the XGBoost focuses is the efficiency and speed of the model and supports parallelization. Focusing on the computational speed and model efficiency. XGBoost also tries to prevent overfitting using Ridge and Lasso regularization. XGBoost trains the model iteratively, correcting or fixing the newer models in each iteration [CHB$^+$15]. XGBoost also has internal cross-validation. Hence, there is no need to identify the number of iterations in each run.

3. **Support Vector Machine (SVM)** is a supervised learning technique that generates input-output mapping functions. The mapping function for this study is a classification function in which nonlinear kernel functions are used

to transform input to a high-dimensional feature space. In this feature space, the data become more separable when compared to the raw input. SVM works by finding the maximum-margin hyper-planes between positive and negative observations. Using mapping function, SVM transforms the non-separable feature to linearly separable features [Wan05, SSBD14].

4.6.3 Setting

In order to evaluate the performance of the A2BCF algorithm, Multiple-Institution Database for Investigating Engineering Longitudinal Development (MIDFIELD) dataset was used [OZTA04] and the goal was to predict students' success in computing majors (CP=11). MIDFIELD is a unit-record longitudinal database for bachelor's students from 20 universities across the United States. This version of the MIDFIELD dataset is used for a binary classification problem and has 4532 samples with 91 features.

All the data are pre-processed using the techniques explained in the previous section. The features of the dataset are taken as the input for the A2BCF algorithm. To evaluate the fitness function, we use Random Forest (RF) classifier with ABC parameters set to Maximum-iterations=100, Initial Population Size (PS) =2000, 5000, and 10000, k=100, with an average run of 10 times. The experiment is performed using cross-validation (cv=3) to yield more robust results. Cross-validation splits the input data into training data and test data independent of each other. Although cross-validation may increase the computational time, it reduces the chances of overfitting and provides a more reliable model. In this study, we used parallel cross-validation, and the final accuracy would be an average of the accuracy for each of the folds. We used *Scikit-learn* libraries, along with other Python packages, to leverage the ML models. The study experiments were conducted using Python 3.8.5 on High-Performance Computational (HPC) resources.

4.7 Results

This section covers the final results using the proposed algorithm. The validating process of the suggested A2BCF algorithm was executed on MIDFIELD dataset [OZTA04, LOO+22]. We compare the performance of A2BCF with a previous study using the same number of folds as in CV (3-fold) [ZMR+21] while using the MIDFIELD dataset to answer the same classification question.

In summary, the proposed algorithm removes irrelevant features from the dataset in an effort to improve its accuracy. Reducing the features in this step can help the training process, training time, and/or other iterative processes such as HPO as it reduces the structure's size. In addition, diversifying the population improves the algorithm's convergence rate.

The overall numerical results with the experimental setting of Section 4.6.3 are given in Table 4.1. According to the Table, in all cases, the classifiers' accuracy after applying the A2BCF algorithm improved. As mentioned earlier, the accuracy is tested under different populations for three different classifiers (RF, XGBoost, and SVM). The advantage of A2BCF is notable when the PS is relatively small (PS = 50). Figure 4.4 shows the impact of the PS on the accuracy of the three ML algorithms. As can be seen, the accuracy of classifiers in all three cases is at its highest when the PS equals 50, and again drops when the PS increases. A potential reason is that the algorithm's number of iterations is lower when the stopping criteria (maximum number of evaluations) are met, and the algorithms do not have the chance to improve themselves so many times. As can be seen, setting the PS around 50 gives the algorithm the chance to improve its performance.

Table 4.1: Comparison of Automated ABC Feature Selection (A2BCF) under different population sizes with a previous study.

	Accuracy (%)			
Classifier	**Reduced Feature Set by A2BCF**			**Original Dataset** [ZMR+21]
	N = 20	**N = 50**	**N = 100**	**N/A**
RF	88.47	**88.51**	88.44	85.30
XGBoost	88.67	**88.76**	88.64	85.16
SVM	87.84	**88.00**	87.81	85.06

N is the secondary population size after clustering and 3-fold cross-validation.

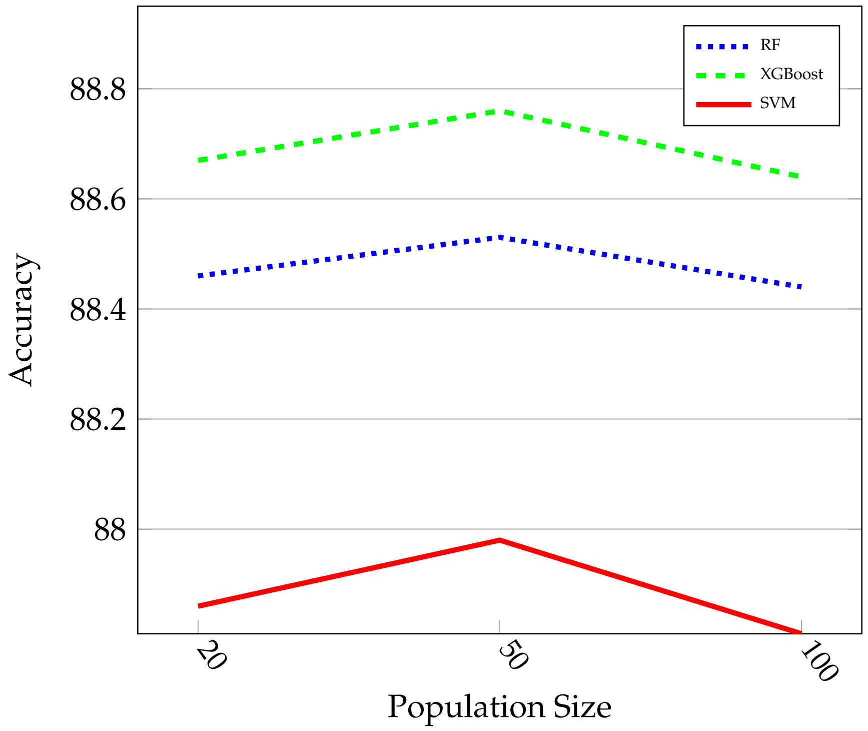


Figure 4.4: Impact of population size on classifier accuracy.

Among the three ML classifiers leveraged to predict student success, XGBoost achieves the best accuracy (88.76%) and is the most well-performed model. Hence, XGBoost is a good candidate for the final FSO framework in this application.

In summary, compared with a previous study [ZMR+21] that considered the features without the FSO method, A2BCF outperforms in predicting student success. A2BCF uses an ABC-based approach to optimize data features and improve classification accuracy.

We also compared the results with another FSO method. Table 4.2 provides the comparison of the result between the PSO [TAMS19] and the proposed A2BCF method. The results show that A2BCF performs better than PSO.

Table 4.2: Comparison of Automated ABC-based Feature Selection (A2BCF) and PSO feature selection approaches.

FSO Method	**Accuracy (%)**		
	N = 20	**N = 50**	**N = 100**
A2BCF	**88.51**	**88.76**	**88.64**
PSO [TAMS19]	87.88	87.70	88.00

N is the number of food sources/particles in ABC and PSO approaches, respectively.

Based on the obtained results, it can be concluded that by leveraging A2BCF, we achieve better accuracy.

4.8 Discussion

In this work, an improved version of the ABC algorithm, A2BCF, is proposed for an automated feature selection approach in different ML algorithms. The proposed method is explored through a binary classification problem and tested on an educational application to predict undergraduate student success in 20 universities across the United States. The proposed method improves the exploration capability of the basic ABC algorithm by incorporating OBL learning algorithms. Further, the clustering algorithm used in the initial phase improves the population diversity and decreases the chance of searching potential areas. The obtained results are compared with those of a recent study, and the results show the robustness of A2BCF. The behavior of A2BCF is explored in different conditions, and experimental results show that the algorithm improves the accuracy and can be employed to solve educational problems with relatively high dimensionality.

The main advantage of the proposed algorithm is its capability to find the optimal features in a reasonable time compared to exhaustive methods (such as grid search). A2BCF could be used as an ideal tool for preprocessing that optimizes the feature selection process, as it enhances the classification accuracy, and minimizes the computational resources. Additionally, in this study, we are targeting to solve a specific problem in an education application. To this end, we focus on developing a tailored FSO algorithm that improves classification accuracy to predict student success. Based on the findings, it can be concluded that A2BCF is a promising approach for FSO to improve classification accuracy.

This study is a part of an ongoing research study to develop a robust AutoML framework that works well in different applications. Some future works are planned in the following directions. First, since we are only building classification models, investigating how our method can be extended to deal with regression problems can be helpful. Another suggestion is to evaluate the algorithm's efficacy on other datasets, which opens the opportunity to explore how different applications affect the performance of the A2BCF. Moreover, additional effort is needed to further improve the tuning time of such methods, including the A2BCF, without decreasing the achieved performance. Last but not least, we plan to insert the implementation of these modifications into our hyperparameter optimization frameworks [ZMR+21, ZMA21a, ZMA21c] to further improve ML algorithms' performance in this study.

As mentioned earlier, the MIDFIELD dataset used is considered for predicting computing students' attrition rates in the computing field. Although the MIDFIELD dataset had been collected to denote the graduation rate of students and not for learning and teaching styles, our proposed approach is an automated and intelligent approach that can partly help education communities to provide guidelines for teaching and learning strategies.

4.9 Conclusions

Determining the optimum features is a crucial task in AutoML. In this work, the the high-dimensional vector of 91 features from the combination of different features is reduced to fewer configurations using A2BCF and achieving the 88% classification accuracy. The A2BCF algorithm with different conventional ML classifiers is applied to the feature set to obtain the optimal or close to the optimal set of features. The results show that A2BCF yields better performance, with 88.76% classification accuracy compared with the accuracy over the original dataset and the PSO method.

CHAPTER 5

AUTOMATED HYPER-PARAMETER TUNING

5.1 Search Algorithms for Automated Hyper-Parameter Tuning

Abstract[1] Machine learning is a powerful method for modeling in different fields such as education, and its capability to provide an accurate prediction of students' success makes it an ideal tool for decision-making tasks related to higher education. The accuracy of machine learning models depends on selecting the proper hyper-parameters. However, it is not an easy task because it requires time and expertise to tune the hyper-parameters to fit the machine learning model. In this study, we examine the effectiveness of automated hyper-parameter tuning techniques in the realm of students' success. Therefore, we add two automated Hyper-Parameter Optimization (HPO) methods, namely grid search and random search, to assess and improve a previous study's machine learning models' performance. The results show that, regardless of the ML model, automated HPOs improve performance. 3-fold cross-validation and accuracy are used to evaluate the experiment results. The experiment results show that applying random search and grid search on machine learning algorithms improves accuracy. We empirically show HPO methods' superiority on an educational real-world dataset (MIDFIELD) for tuning hyper-parameters of conventional machine learning classifiers. This work emphasizes the effectiveness of automated HPO while applying machine learning in the education field to aid faculties, directors' or non-expert users' decisions to improve students' success.

[1]This chapter is an edited version of the author's previous work published in [ZMR+21] ©2021 CSCE.

Keywords: Hyper-Parameter Tuning, Educational Data Mining, Machine learning Optimization, AutoML

5.1.1 Introduction

Overview

Machine learning (ML) is an evolving and fast-growing research field that learns from past experience [JM15]. It has garnered much attention from academic and industrial researchers to discover the patterns and data representations from the raw data [WF02]. Overall, ML models' main objective involves determining the best and most predictive model, which can be done by finding and tuning proper hyper-parameters. Hyper-parameter tuning is an essential process to make an ML model perform at its best [ADNDS$^+$19] and previous research shows that it can significantly improve the models' performance [SS04]. Therefore, it is required to train the ML models with different regularization parameters to build accurate models. Manual tuning the hyper-parameters is not only a common approach in graduate research studies but also many ML models are developed by experts that were all designed manually and by trial and error. This indicates that even ML developers and experts need the time and resources to create well-performed predictive ML models [HZC21]. Since different ML algorithms have different types of hyper-parameters, the tuning process for them is also different [DGMCEGC19]. Also, as the number of hyper-parameters and the range of their values increase, manual testing hyper-parameters becomes more and more impractical. This emphasizes the necessity for having automated processes of optimizing the hyper-parameters and has inspired recent research in approaches for automatic optimization of hyper-parameters.

In the context of ML, the advantages of automated hyper-parameter tuning are multi-fold, including lessening the human effort, enhancing performance according to the application and the problem at hand, and providing fairness to research and scientific studies [HKV19].

Motivation

The usage of information technologies in various fields has led to increasingly large-scale data derived from various settings. One of these settings is education, concerning better understanding of students' pathways and the environments they learn in. Educational Data Mining (EDM) is an emerging field focusing on mining datasets to answer educational research questions [FHV$^+$19][PA14]. One of the most important EDM applications is predicting students' success, or performance [PA14]. There are a variety of approaches to measure the success of students, such as graduation rates, completion on time, or GPA [DSdNDM$^+$13][BV19][ZLP$^+$20]. However, these metrics may overestimate or underestimate a sub-population persistence, such as non-traditional students, part-time students, or transferred students. Therefore, these populations are usually not considered in many studies, which leads to a lack of understanding of the educational pathways [OOL$^+$12].

Hyper-parameter tuning is an essential task to make a predictive model that performs at its best [ADNDS$^+$19]. Building such models is the main goal of ML models [EMS19]. However, despite hyper-parameters critical role in the resulting predictive models' quality, they have no clear agreeable defaults in different applications. Manually tuning the hyper-parameters not only needs a deep understanding of the ML models but is also impractical, time and cost-inefficient. Hence, it has become vital to automate the process of optimizing the hyper-parameters. In HPO, we usually aim to use the value of parameters that significantly contribute to improving a model's accuracy. Therefore the search algorithm looks through different combinations of hyper-parameter configurations (search space), which enable the model to generate the best model among the candidates through an iterative process [YS20, DFNNS17]. To be effective in the application, ML models' hyper-parameters need to be selected automatically and carefully. For a new dataset, the optimal parameter values depend on the application and, more specifically, the dataset itself. This work helps educational researchers and institutions better understand and develop ML models by identifying the appropriate set of HPs in an effective way.

Contribution

To address the issues mentioned above regarding manual tuning, this study applies two automated HPO methods to predict students' stickiness accurately. Grid Search (GS) and Random Search (RS) are among the automated parameter optimization methods. The advantage of using GS and RS is higher learning accuracy and its capability for parallelization, which is not an option for all the HPO methods. We apply GS and RS to find the most appropriate hyper-parameters for different conventional ML algorithms. The reason for selecting various ML algorithms is that the performance of a given ML model not only depends on the fundamental quality of the algorithm but also on the details of its tuning. Therefore, it is hard to decide if a given ML model is genuinely better or better tuned. In this study, leveraging different numbers of ML models gives us the option to choose the model with the best performance among different ML models rather than a single model.
The exploration conducted in this work builds upon established research by applying a larger dataset rather than datasets that are usually considered in studies of this nature. The main contributions are expanding upon existing research by incorporating automated HPO techniques leveraging conventional ML models into a single pipeline

to improve prediction performance and enhance educational decision-making. Also, this is the first time HPO is being applied to this dataset (MIDFIELD) [OZTA04]. Unlike other studies that study short time spans and single institutional datasets, this study dataset considers a 30-year longitudinal dataset for undergraduate students from 16 universities. In this work, we use a modified definition of graduation, stickiness (the fraction that ”stick” to the program or persist), for students who came into contact with their programs [ZER+21], to include the populations that are overestimated or underestimated in previous studies. This study aims to improve ML models’ classification for the MIDFIELD dataset using GS and RS and then compare it to the previous work.

Organization

In the following sections of this study, we first develop a clear and formal definition of HPO, and we provide a basic understanding of the concepts and methodologies used in this study. From there, we discuss the implementation and evaluation of these approaches. Next, we cover experimental results, conclusions, and future work ideas. Figure 5.1 shows the overall structure of this study.

5.1.2 Related works

5.1.3 Hyper-parameter Optimization

Every ML model has some hyper-parameters, and tuning these hyper-parameters is essential for making a model work at its best. The notion of hyper-parameters is different from parameters. Hyper-parameters are the parameters that need initialization before training the ML model since they represent the ML model architecture. In contrast, parameters can get initialized and updated during the model training

[KJ$^+$13]. Automatically tuning the hyper-parameters is one of the main tasks in automated machine learning (AutoML) to release the burden of manual and human tasks and improve the performance, reproducibility, and fairness of various studies [FH19]. There are several optimization techniques for HPO problems. The most common and conventional HPO methods are manual search or grad student descent (GSD), GS, and RS. These methods treat the hyper-parameters configurations independently and return the best model architecture. Each of these methods is defined as below:

Manual Search: Manual search or Grad Student Descent (GSD) is the most basic hyper-parameter tuning method. This method is also known as "trial and error," or "babysitting" [Abr19]. GSD is a prevalent approach among researchers and students. In this method, ML model users try different possible hyper-parameter values based on guessing or domain knowledge and repeat this process until they obtain an improved and satisfactory result or when they are out of time. Mohammadi et

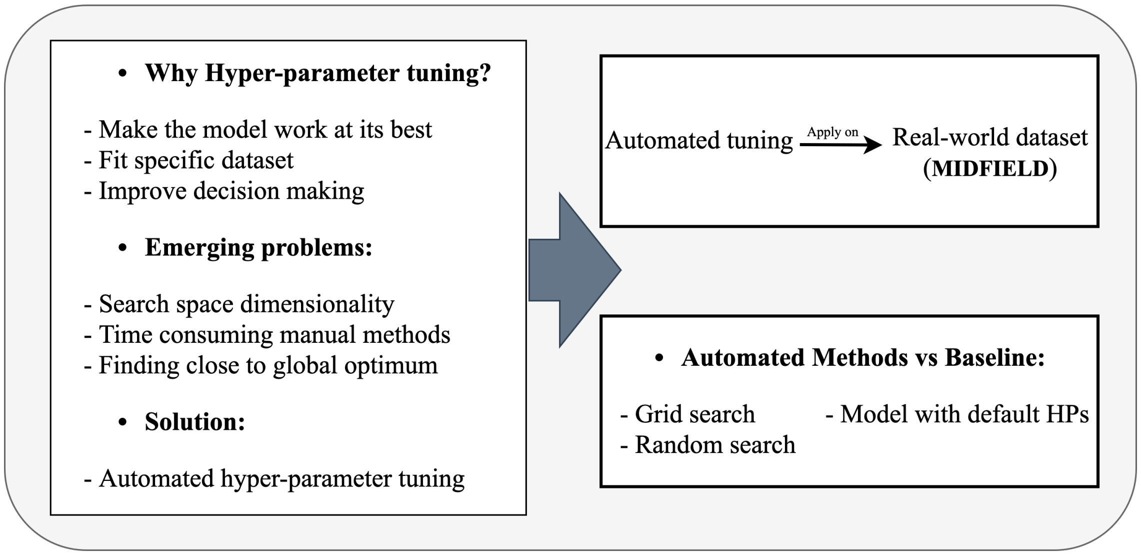

• Why Hyper-parameter tuning?
- Make the model work at its best
- Fit specific dataset
- Improve decision making
• Emerging problems:
- Search space dimensionality
- Time consuming manual methods
- Finding close to global optimum
• Solution:
- Automated hyper-parameter tuning
Automated tuning
Apply on
Real-world dataset (MIDFIELD)
• Automated Methods vs Baseline:
- Grid search
- Random search
- Model with default HPs

Figure 5.1: Overall structure - using grid search and random search

al. explored the effect of manual parameter tuning on performance by combining different embedded parameters and improved the accuracy of semantic auto-encoder in an image classification problem [MAA19]. However, this method needs a deep understanding of the ML models or sufficient time to get reasonable results. However, the complex nature of ML models and large search spaces make it an impractical approach [Olo18]. These factors behooved researchers into looking for automated HPO methods.

Grid Search: Grid Search (GS) is the brute-force way of searching hyper-parameters [CSP+14] with defined lower and higher bounds along with specific steps [SPBW16]. GS work based on the Cartesian product of the different set of values, evaluate every configuration, and return the combination with the best performance [HCB+14]. Algorithm 4 shows the pseudo-code for the GS. GS algorithms have a simple implementation; however, they can be very inefficient for large search spaces due to their exhaustive nature. This problem exacerbates as data dimensionality increases. The exponential growth in the search space or data dimensionality is called the curse of dimensionality (CoD) [CSP+14]. The time complexity of GS when we have k parameters and n values is $O(n^k)$ [YA98].

Algorithm 4 Implementation of the GS

```
   Input: list of N hyper-parameter along with their possible values and lower and
upper bounds with steps
   Output: best model architecture along with performance
1: for FOR every n in the list of N candidates do
2:     Train model with c on the training set
3:     Evaluate ML classification on validation set
4:     if ModelPerformanace > MaxPerformance then
5:         MaxPerformance = ModelPerformanace
6:         BestHP = n
7:     end if
8: end for
9: Return BestHP, MaxPerformance
```

Random Search: RS is another common and automated method of searching hyper-parameters [BB12]. RS chooses the hyper-parameters configurations on a random basis (instead of evaluating every configuration) and repeats this process until the defined resources are over. Algorithm 5 shows the pseudo-code for the RS. RS is much faster than GS but does not follow a path to find the optimal configuration. The time complexity of RS is $O(n)$ where n is the resource (ex. the number of algorithm iterations). The most important advantage of RS to GS is that it explores more of a search space and can yield better results when given enough resources.

Algorithm 5 Implementation of the RS

```
   Input: list of hyper-parameters along their possible values and ranges
   Output: best model architecture along with performance
 1: while budget condition n do
 2:    Generate an independent random list $L_n$ of possible hyper-parameter values
 3:    for FOR every l in the list of L candidates do
 4:       Train model with l on Training Set
 5:       Evaluate ML classification on Validation Set
 6:       if ModelPerformanace > MaxPerformance then
 7:          MaxPerformance = ModelPerformanace
 8:          BestHP = l
 9:       end if
10:    end for
11: end while
12: Return BestHP, MaxPerformance
```

5.1.4 Experimental Setup

The baseline ML algorithms (model with default parameters), GS, and RS are implemented in this work. The machines we use to perform these experiments come from Amazon Web Services (AWS). We use t2.xlargeLinux (64-bit)-based instances equipped with four v-CPUs up to 3.0 GHz scalable processor and 16 GB RAM. All

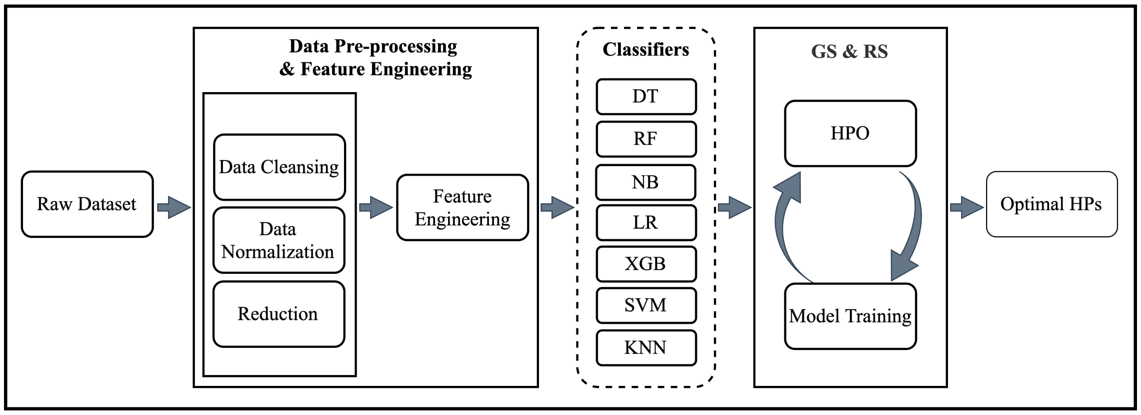


Figure 5.2: Flowchart of hyper-parameter optimization using GS and RS

implementations are done using Python 3.7. The involved ML and HPO algorithms are implemented and evaluated using different libraries, including Numpy, Pandas, Scipy, and Scikit-Learn.

Figure 5.2 shows the research methodology's flowchart in this work. The first phase to conduct this research is collecting the raw dataset. We then prepare the data for ML algorithms. Data preparation includes discrete tasks such as data reduction, data cleaning, data normalization, and feature engineering. Data reduction, here, refers to using a subset of the dataset rather than the entire dataset. Filtering and sampling are among the most common ways of data reduction. Since we are only interested in computing students, we filter the data and only include the students who enrolled in one of the following majors: computer science, computer engineering, software engineering, computer programming, information technology, and computing and information sciences. In the data cleaning step, which is a step for removing the corrupt and not useful data, we remove the features with more than 60% missing values. We also normalized or standardized the range of features of data. After the preprocessing step is the feature engineering step in which we create some new features from raw features that we think might be useful for the power of prediction. Additionally, in this step, we apply one-hot encoding to convert the categorical variables to numerical variables so that we can feed them into machine learning algorithms.

Next, we feed the data to different ML models, including, Decision Tree (DT), Random Forest (RF), Naive Bayes (NB), Logistic Regression (LR), XGBoost (XGB), Support Vector Machine (SVM), and K-Nearest Neighbor(KNN). These models operate as a black box, and therefore, no additional information about building them is required. The next step is the automated HPO method which is an iterative process of choosing hyper-parameters. The two most commonly-used performance metrics are used in our experiments. Accuracy is used as the classifier performance metric, which is the proportion of correctly classified data; also, we use the total time needed to complete an HPO process (or, in other words, tuning time).

For each experiment on our selected dataset, 3-fold cross-validation is applied in the training process (to prevent overfitting) as well as GS and RS for finding the optimal set of hyper-parameters. Then, we test the model using testing data. In each step, the accuracy of the model is calculated.

Algorithm 6 explains GRS pseudo-code that starts with initializing hyper-parameters candidates and calling both RS and GS applied on various ML models and ends with returning the final ML model, optimal hyper-parameter architecture, and its performance. In this study, we use the steps defined for GS were defined as 0.5. We also apply 3-fold cross-validation to find the optimal configuration.

Algorithm 6 Implementation of GRS-AutoHP

```
    Input: 1) $List_{ML}$ of models (DT, RF, NB, LR, XGB, SVM, KNN)
    2) Raw dataset
    Output: best model along performance and optimal architecture
 1: FinalPerformance = 0
 2: Call PreProcessing
 3: for every model in the $List_{ML}$ do
 4:    Call GS
 5:    Call RS
 6:    if (PerformanceGS > PerformanceRS
       & PerformanceGS > FinalPerformance) then
 7:       FinalModel = model
 8:       BestHP = OptimalGS
 9:       FinalPerformance = PerformaceGS
10:    else
11:       if (PerformanceRS > PerformaceGS
          & PerformanceRS > FinalPerformance) then
12:          FinalModel = model
13:          BestHP = OptimalRS
14:          FinalPerformance = PerformaceRS
15:       end if
16:    end if
17: end for
18: Return FinalModel, BestHP, FinalPerformance
```

Dataset

All of our experiments are conducted on MIDFIELD (Multiple-Institution Database for Investigating Engineering Longitudinal Development) [OZTA04] to provide practical examples. MIDFIELD is a longitudinal student record level dataset from 1988-2018 for all undergraduate, degree-seeking students at partner institutions. MIDFIELD basically includes everything that appears on students' transcripts, such as demographic data (ex. sex, age, and race/ethnicity) and information about major concentration, enrollment, graduation, and school and preschool students' performance. As previous research shows, computing majors have different patterns from other STEM majors[ZER+21], the data examined in this section is exclusive to computing fields, with about 45k observations. MIDFIELD is used as a binary-classification problem to predict computing students' success, more specifically stickiness. For our classification models, accuracy is used as the classifier performance metric. After completing each experiment, the model with the optimal ML architecture will be returned.

In this study, we compare the HPO methods (GS and RS) with the baseline ML models with default hyper-parameters. Since for each ML model, only a few hyper-parameters have significant impacts on the model's performance [YS20], in this study, we consider the main hyper-parameters of the ML classifiers for automated tuning. Table 5.1 shows the main hyper-parameters for each of the ML classifiers used in this study. To define the search space for HPO methods, hyper-parameters need initialization. In general, hyper-parameters have three different types: categorical, discrete, and continuous. For categorical and discrete hyper-parameters, initialization for both RS and GS are the same. For categorical values, we include all the valid options. For discrete values also, we specify a valid range so that the algorithm search through that range. For continuous types, however, we need to treat them differently. For RS, the algorithm chooses random float numbers in the range that we define. For GS, however, we define an incremental step as fixed points so that algorithm can check those values.

5.1.5 Experimental Results

Classification Results

This section discusses the results of our experiments. As mentioned earlier, our first scenario is applying different ML algorithms to the data with their default hyper-parameter values as our baseline method. The first column in Table 5.2

Table 5.1: Conventional machine learning classifiers' main hyper-parameters

ML Classifier	**Main Hyper-Parameters**
Decision Tree	criterion, max_depth, min_samples_split, min_samples_leaf, max_features
Naive Bayes	alpha, fit_prior
Support Vector Machine	C, kernel
XGBoost	n_estimators, max_depth, learning_rate, subsample, colsample_bytree
K_Nearest Neighbor	n_neighbors, weights, algorithm
Logistic Regression	penalty, c, solver
Random Forest	n_estimators, criterion, max_depth, min_samples_split, min_samples_leaf, max_features, bootstrap

shows the accuracy of our baseline. As can be seen, results indicate that Random Forest performs better than other ML classifiers and, as a result, the final model in the baseline model selection step. Next, we implement the GS and RS on the same dataset. Columns two and three in Table 5.2 are the results for these two HPO methods, respectively. The experiment results show that, regardless of the ML model, both HPO methods have increased the model performance. However, some ML models perform better than the rest. For example, NB is not a well-performed model in comparison to the other ML models. The reason is that in NB the probability of each class given different input values requires to be calculated and No coefficients need to be fitted by optimization procedures. However, this characteristic makes the NB algorithms faster than other ML models. Therefore, in cases the user wants to use NB models for their specific purpose, it can actually be a fast model for hyper-parameter tuning (NB has also a lower number of hyper-parameters). From Table 5.2 we can see that after applying GS, XGB is also one of the well-performed candidates among the classifiers, while this was not the case for our baseline approach. Also, regarding the RS method, XGB is a well-performed classifier and, as a result, is our final model. This indicates that relying on the default parameter to find the best model is not always the best decision. The accuracy comparison between the baseline approach and the RS and GS methods is summarized in Figure 5.3.

Comparison With a Previous Study

In this section, we compare the classification results from our experiment with previous work. In the previous study [ZLP+20], an experiment is conducted to predict students' graduation using manual tuning using MIDFIELD as their dataset (N=39k). In this study, we replicated the study and built models using the same configurations (N=45k).

Figure 5.3: Comparison of GS and RS with baseline

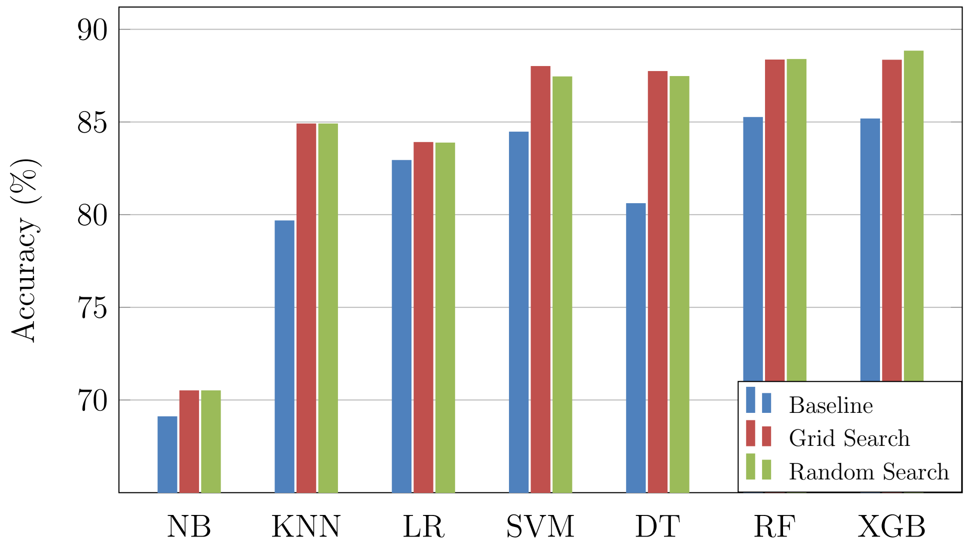


Table 5.2: Accuracy comparison of baseline, GS and RS

Classifier	**Baseline (%)**	**GS (%)**	**RS (%)**
Naive Bayes	69.09	70.49	70.49
Logistic Regression	82.92	83.89	83.86
K-Neasrest Neighbor	79.66	84.89	84.89
Support Vector Machine	84.45	87.99	87.43
Decision Tree	80.59	87.72	87.45
Random Forest	**85.24**	**88.34**	88.37
XGBoost	85.16	**88.33**	**88.80**

Previous work results and our replicated experiment results using manual tuning are described in Table 5.3. The last column of the table shows the optimal results achieved from the previous section using automated HPO. A summary of the results of this experiment is shown in Figure 5.4. As can be seen from the results, automated HPO beats manual tuning even when the domain knowledge is used to tune the hyper-parameters. Automated HPO could take longer than manual tuning, but the results are promising and can guarantee performances close to the global maximum performance.

Table 5.3: Accuracy comparison of manual and automated tuning

Classifier	**Previous Work (%)**	**Manual (%)**	**Automated (%)**
Naive Bayes	82.25	69.09	70.49
K-Neasrest Neighbor	75.38	81.43	84.89
Logistic Regression	83.18	82.92	83.89

Continued on next page

Table 5.3 – continued from previous page

Classifier	**Previous work(%)**	**Manual (%)**	**Automated (%)**
Support Vector Machine	85.27	85.06	87.43
Decision Tree	86.78	82.08	87.72
Random Forest	**88.27**	**85.30**	88.37
XGBoost	74.58	85.16	**88.80**

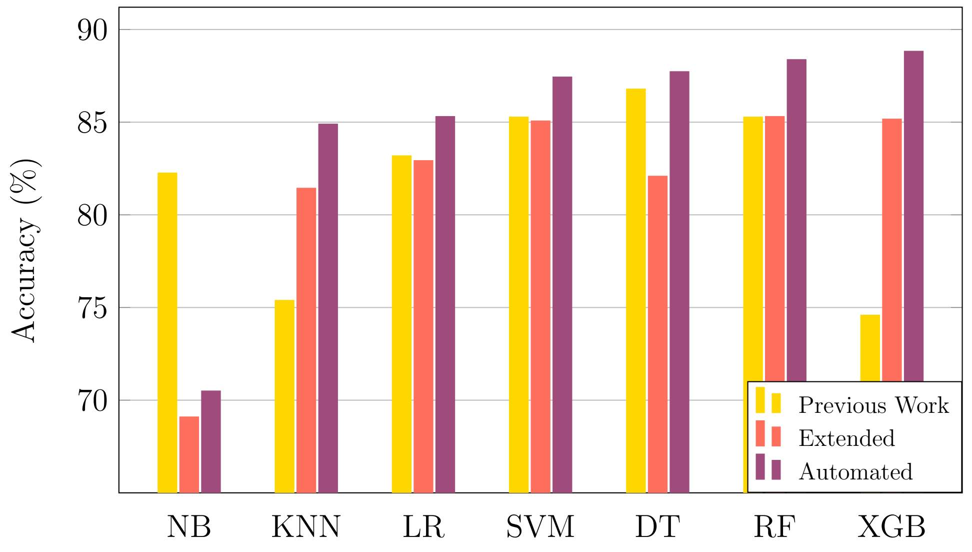


Figure 5.4: Comparison of automated HPO with previous work

5.1.6 Conclusions

GS and RS tuning methods are applied on seven ML models on the MIDFIELD dataset. We evaluate the DR, NB, SVM, XGB, KNN, LR, and RF models, the conventional ML models. We use a subset of computing students from 14 institutions across the U.S. The experiment results show that regardless of the ML model leveraged, GS and RS improve the classification accuracy for MIDFIELD dataset. Also, we see that such automated methods beat the manual tuning methods even when domain knowledge is used. Additionally, we observe that using HPO methods, the final selected model is XGB. However, the previous study and our experiment using manual tuning show that RF is a well-performed method. This indicates the importance of automated HP tuning. We conclude that applying HPO methods helps find the proper final model, optimal HPs and improves ML models' performance in the education field.

5.1.7 Discussion and Future Work

Machine learning has become an essential approach for tackling different kinds of problems and has been widely used in different applications. However, for a model to work in a specific application or dataset, its hyper-parameters require tuning. We demonstrated here that the GS and RS HPO methods exhibit improved prediction performance. As discussed, one of the advantages of the applied methods is that configurations in these approaches are independent, making the parallelization easy. However, these methods have their own disadvantages. Both grid search and random search are the techniques that search the areas that are not promising. In other words, they waste time looking for optimal solutions among configurations that are not optimal. This characteristic leads to high time complexity, which makes such methods computationally very expensive. This problem is exacerbated when the data scale (for various reasons), and search space scale, increase (also called the curse of dimensionality). Needless to mention that the ML models and the number of hyper-parameters also affect the time complexity. Hence, it is crucial for such problems to be solved in a computationally efficient way in order to have real-time and intelligent decision-making. In our case, using a large search space and one-hot encoding of the features resulted in rather high time tuning. More specifically, ML models, specially tree-based models and SVM, are computationally very expensive. This is due to various reasons including the number of hyper-parameters, their data types, their ranges, and the complexity of models. High time complexity, especially for these models is an issue that needs to be addressed in future research. Data reduction techniques using nature-inspired algorithms are being used in different applications to solve dimensionality problems to achieve promising running time and lower time complexities [MAA20b]. To achieve this goal, evolutionary algorithms have been used widely. One of the evolutionary algorithm applications is feature selection to reduce datasets' dimensions without decreasing the performance. Mohammadi *et al.* noted that such algorithms are robust enough to be used in different applications and research that try to do the dimension reduction [MAA20a]. Going forward, we plan to further examine the impacts of using nature-inspired algorithms in educational fields to optimize the ML models' hyper-parameters. Also, the role of specific nature-inspired algorithms to optimize ML models should be investigated.

5.2 HyP-ABC: Automated Hyper-Parameter Tuning Using Evolutionary Optimization for Classical Machine Learning Algorithms

Machine learning techniques lend themselves as promising decision-making and analytic tools in a wide range of applications. Different ML algorithms have various hyper-parameters. In order to tailor an ML model towards a specific application working at its best, its hyper-parameters should be tuned. Tuning the hyper-parameters directly affects the models' performance. However, for large-scale search spaces, efficiently exploring the ample number of combinations of hyper-parameters is computationally expensive. Some of the automated hyper-parameter tuning techniques suffer from low convergence rates and high experimental time complexities. In this chapter, we propose HyP-ABC, an automatic hybrid hyper-parameter optimization algorithm using the Artificial Bee Colony approach to be fitted for hyper-parameter tuning of machine learning algorithms in classification problems. This tool is evaluated using three interpretable machine learning classifiers suggested for exploration in previous research: random forest, extreme gradient boosting, and support vector machine. In order to ensure the robustness of the proposed method, the algorithm takes a wide range of feasible hyper-parameter values and is tested using a real-world educational dataset. Experimental results show that HyP-ABC is competitive with state-of-the-art techniques. Also, it offers fewer hyper-parameters to be tuned than other population-based algorithms, making it worthwhile for real-world hyper-parameter optimization problems.

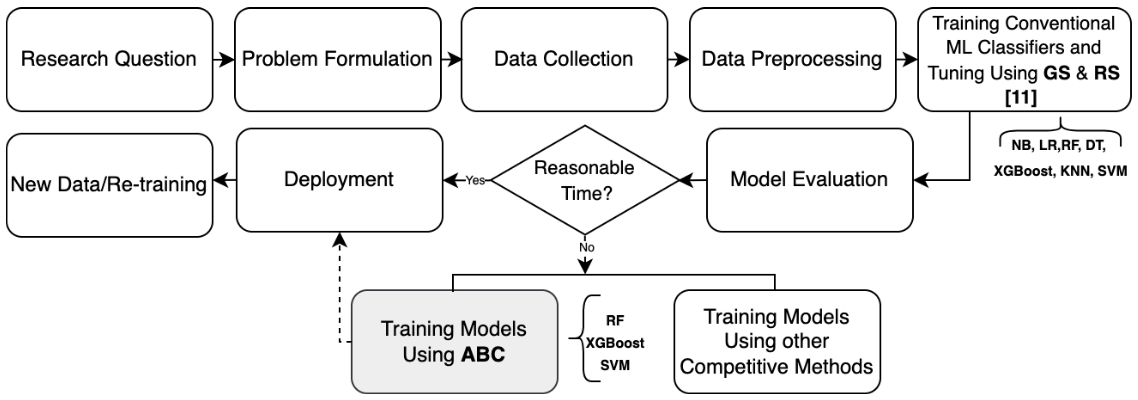
Keywords: AutoML, Artificial Bee Colony, Education, Hyper-parameter Tuning, Machine Learning Optimization

5.2.1 Introduction

Motivation

Deploying Machine Learning (ML) for real-world problems causes challenges, such as selecting the proper model among a set of candidate models, hyper-parameter tuning, or selecting the best features to feed to the ML models. An ML model's performance depends on such initial design decisions, which can be confusing to new users who desire to choose the most appropriate model [MAA19]. These decisions are usually made based on the model's obtained quality or in other words, Performance Indicator (PI) [NC12]. According to the principle of Occam's razor, a model should not be too simple nor too complex so that it can be efficient and can also capture data patterns without overfitting [Bie03, MAA19]. Hyper-parameter tuning chooses the values that have a more considerable impact on the ML model's performance. Hyper-parameters on an ML model control the convergence of the learning process. Hence, it is crucial to find the optimal values for their hyper-parameters [CDM15]. Independent of the expertise level of the users, obtaining desirable results using different datasets can be conducted manually and is a tedious task [SCK22, YS20]. This is where Automated ML (AutoML) comes into the picture to alleviate such burdens. Automated HPO is one of the decisive and primary tasks of AutoML [SY20].

Bhattacharyya et al., in [BMDC21] proposed an automated heartbeat classification framework. This framework takes advantage of an exhaustive HPO method, Grid Search, to tune the hyper-parameters of random forest and support vector machine classifiers. The results showed a significant enhancement in efficiency. In another study [VBF21], Vu et al., proposed a quantitative and constraint preserving score to ease the process of choosing hyper-parameters of visualization algorithms.



Research Question
Problem Formulation
Data Collection
Data Preprocessing
Training Conventional ML Classifiers and Tuning Using GS & RS [11]
NB, LR,RF, DT, XGBoost, KNN, SVM
Model Evaluation
Reasonable Time?
Yes
No
Deployment
New Data/Re-training
Training Models Using ABC
RF XGBoost SVM
Training Models Using other Competitive Methods

Figure 5.5: General framework of the study

This method uses Bayesian optimization to search the search space and obtain the most appropriate configuration. The experiments show promising results. In [ZMR+21], Zahedi et al. explored different classic ML algorithms using the most common HPO methods, namely, grid search and random search. The experimental results of their study indicate that ML algorithms such as SVM. RF and XGBoost require other hyper-parameter tuning methods due to the computationally expensive nature of these algorithms.

In HPO, the problem is more challenging when there are many configurations; as hyper-parameter value ranges increase, the search space grows exponentially. Therefore, common practices of tuning hyper-parameters for large search spaces are not ideal. In this work, we will show how population-based algorithms (PBAs) can tune the hyper-parameters of different ML algorithms.

5.2.2 Objective

Manual search or automated exhaustive search among n^k configurations of hyper-parameters are impractical and time inefficient [YGW16]. Manual tuning does not provide reproducibility. Exhaustive search, on the other hand, suffers from dimensionality issues in large search spaces. As such, there has been increased research in

HPO to optimize the performance of different ML models regarding both accuracy and time.

HPO methods may change the final model in comparison to when a model is being trained with default hyper-parameters [ZMR+21]. Hence, they provide fairness to research and scientific studies. Also, when making decisions, time is of the essence. Hence, many black-box optimization models do not fit such optimization problems because they do not consider the function evaluation time [YS20]. Therefore, proper algorithms should be applied to such problems to find the optimal set of hyper-parameters. Mohammadi et al. provided different examples of applying PBAs, such as feature extraction, in different domains and encouraged researchers to apply such methods to solve large-scale optimization problems [MAA20a]. Our goal is to design and develop an HPO framework to tackle the challenges discussed above; The framework utilizes ABC to minimize the run-time caused by large dimensions of search space, known as Curse of Dimensionality (CoD), and possibly improve the accuracy of the ML classifier. This framework will assist the predictions for intelligent and real-time decision-making and lessen the financial costs and labor burden.

In the literature, evolutionary algorithms have been used to tune the hyper-parameters of different ML algorithms [BBCY18, OB20, OB19]. To fit ABC to HPO problems in ML classifiers, this part uses HyP-ABC to build a binary classification model to improve the performance. Figure 5.9 shows why we ended up proposing the HyP-ABC algorithm. The figure demonstrates the challenges faced in some of the ML classifiers [ZMR+21] while applying hyper-parameter tuning on conventional ML classifiers including Naive Bayes(NB), Logistic Regression (LR), Random Forest (RF), Decision Tree (DT), Extreme Gradient Boosting (XGBoost), K-Nearest Neighbors (KNN) and Support Vector Machine (SVM). To address the challenges of those specific ML classifiers (RF, XGBoost, and SVM), we developed HyP-ABC for hyper-parameter tuning. HyP-ABC has some advantages over other automated nature-inspired algorithms or conventional tuning methods including Grid Search (GS) and Random Search (RS) or genetic and particle swarm algorithms.

5.2.3 Contribution

The main contributions of this study are as follows:

1. This chapter develops a novel tuning optimization method that fits HPO problems for addressing the challenges left in a recent work [ZMR+21] using a modified evolutionary optimization algorithm, specifically ABC.
2. The HyP-ABC algorithm outperforms the HPO methods utilized in an existing study in 2021 [ZMR+21] in terms of both tuning time and accuracy.
3. The HyP-ABC algorithm is competitive with other state-of-the-art existing HPO methods.
4. This chapter is the first to explore the optimization of ML classifiers tailored towards large-scale educational datasets like MIDFIELD [OZTA04]. The details regarding the MIDFIELD dataset is provided in out previous study [LOO+22].

5.2.4 Organization of chapter

The rest of the chapter is organized as follows: in section 5.2.6 we provide an introduction to PBA and a comprehensive explanation of ABC. Then, we present the related work regarding hyper-parameter tuning and different automated tuning methods in section 5.2.5. We also cover ABC applications in this section. Section 5.2.6 covers hyper-parameter tuning using the ABC approach and the advantages of

ABC over other techniques. In section 5.2.7, we propose the HyP-ABC algorithm; an ABC-based algorithm fitted to HPO problems. The experimental methodology is presented in Section 5.3.4, and Section 5.2.9 presents experimental results to demonstrate the performance achievable by the proposed approach. Last, section 5.2.10 and 5.2.11 present the results, conclude the chapter and discusses future directions.

5.2.5 Related Works

Hyper-parameter Tuning

GS is a brute-force method that searches all the hyper-parameters within a fixed search space. The advantage of this method is that it obtains the optimal solution in a discrete search space [ZMR+21]. However, it is computationally expensive in large-scale spaces. RS is another approach that randomly selects the values with limited resources (time or number of iterations). This method works well for search spaces containing continuous values. However, luck plays a part in this method, and giving more resources increases the chance of getting better results [ZMR+21]. There are some improved versions of RS, such as Hyperband (HB) which is more efficient, especially in cases where resources are limited.

Unlike GS and RS, Bayesian Optimization (BO) [EFH+13] prevents evaluating many of the unnecessary configurations based on the evaluations of the previous steps. BO is an approach that is commonly used for HPO problems and, unlike model-free approaches such as GS and RS, is a model-based technique. In other words, the future evaluations are based on the previously evaluated points. However, since BO methods work sequentially to balance exploitation and exploration, they are challenging to parallelize. The most popular BO techniques in HPO problems are Gaussian Processes (GP) and Tree-structured Parzen Estimator (TPE).

GA and PSO are the popular population-based algorithms used in HPO problems. GA is a search strategy inspired by Charles Darwin's theory of natural evolution. It is based on the process of natural selection, where the fittest individuals are the ones being selected for the next steps of reproduction to produce next-generation offspring. PSO is another evolutionary algorithm that traverses the search space in a semi-random manner. This algorithm is inspired by individual and social behaviors of biological populations [YWC+18].

For some models, such as tree-based models (RF and XGBoost), the number of main hyper-parameters or their ranges is higher than other ML models, which leads to larger search space scales, making them the most challenging for tuning [HKV19]. Furthermore, since different ML algorithms have different hyper-parameter types(continuous, integer, and categorical), they should be treated differently in tuning processes [DGMCEGC19]. For the ML classifiers selected to explore in this study, ABC is chosen since it enables parallel executions to improve tuning time, particularly for models that often require massive training time. Some other techniques, such as the GA method, usually take more time than ABC since it is challenging to parallelize. Also, although techniques such as PSO have a reasonable convergence rate, they have a higher chance of sticking into local optimums and proved to be less efficient in comparison to ABC regarding performance[KD16].

ABC Applications

This section covers some of the previous studies of ABC for different optimization problems. One of the applications of ABC can be seen in [MA14] where authors proposed an ABC-based approach called IFAB, distinguishing between clean and un-

Table 5.4: Comparison of GA, PSO, and ABC algorithms

Method	**Advantages**	**Disadvantages**	**Time Complexity**
GA	-No proper initialization is needed.	-Poor with parallelization -Introduces more hyper-parameters than PSO or ABC -Slow Convergence	$O(n^2)$ [SRS11]
PSO	-Enables parallelization. -Faster convergence than GA or ABC.	-Needs proper initialization. -May stuck in local optimum	$O(n \log n)$ [YS20]
ABC	-Enables Parallelization. -Higher efficiency Balances exploration & exploitation.	-Need proper initialization -Slower convergence than PSO.	$O(n \times D)$ [SB16]

clean images. The authors modified the original ABC to work for discrete problems. Moreover, Sarac Essiz and Oturakci developed a comparative analysis to explore the impacts of the ABC-based feature selection algorithm [SEO21] and eliminate non-informative features in the cyberbully detection problem. They also showed that their method has a better performance than some conventional methods like information gain, relief, and chi-square [SEO21].

Amar and Zeraibi [AZ20], combined ABC and Support Vector Regression (SVR) to predict minimum miscibility pressure and improve the oil recovery process. In their study, ABC was used to find the best hyper-parameters of the SVR model. In another study, Dokeroglu et al. developed a hybrid ABC optimization algorithm for the quadratic assignment problem using Tabu search to simulate exploration and exploitation phases [DSC19]. They showed that ABC performs well for most quadratic assignment problems and can compete with other state-of-the-art meta-heuristic algorithms existing in the literature.

Choong et al. utilized a modified choice function for the ABC method. This approach adjusts the neighborhood search of the employed and onlooker phases. The results showed that the modified choice function brought advantages to the search process [CKH+18]. Since training ML models is time-consuming, it is being argued that the use of EAs such as ABC is of great benefit to solve problems caused by the Curse of Dimensionality (CoD) [HXZ+18]. In [MDPS21], the authors proposed two algorithms to modify the original ABC. The former employed neural network initialization, and the latter utilized stochastic gradient descent in the employed phase of ABC to improve the convergence rate and observed promising results.

In another study, Zhao et al. performed a comparative analysis that proposed a modified version of ABC to improve the performance of SVM classification using parameter optimization. The proposed method utilized chaotic sequences for the initialization step, and they also defined an adaptive step size for the neighborhood search to boost the algorithm convergence. The modified ABC algorithm was examined to perform classification on two hyperspectral images. The authors then compared the results with three other PBAs and observed the superiority of the ABC method over the rest of the algorithms [ZZWC20].

Pandiri and Singh [PS18] developed a hyper-heuristic-based ABC algorithm for the k-Interconnected multi-depot multi-traveling salesman problem (k-IMDMTSP). The authors stated that due to the parameter values, it is impossible to have a unique algorithm that outperforms the rest of the algorithms emanating from k-IMDMTSP. Hence, they leveraged a hyper-heuristic-based ABC algorithm for this problem. The authors presented an encoding scheme to be used inside the algorithm. The experimental results exhibited smaller search space and improved performance compared to other approaches existing in the literature.

In another study, Mazini et al. took advantage of the hyper-parameters regulation method of ABC for feature selection to propose a reliable hybrid method to detect anomaly network-based intrusion. Experimental results on a network traffic dataset exhibited improved performance and detection rate compared to other intrusion detection systems in various scenarios [MSM19]. Gunel and Gor developed an ABC-based algorithm to solve initial value problems. To define a mutation operator, the authors used a dynamically constructed hyper-sphere to generate new food solutions to enhance the exploitation capability of the ABC. In this approach, the solutions for the differential equations were yielded through training neural networks [GG19].

In [SSH19], Sayed et al. used an ABC-based approach to tune the hyper-parameter of SVM to improve the learning performance. The proposed method was compared with popular swarm algorithms and showed promising results. Agrawal also proposed an extended version of ABC using some features of the Gaussian ABC scheme to tackle some of the disadvantages of the original ABC, such as a slow convergence. The proposed method was used to adjust the exploration and exploitation phases. They examined the proposed approach on some datasets and observed that the proposed method outperformed the original ABC in most experiments [A+20].

5.2.6 Population-Based Optimization Algorithms

Population-based optimization algorithms work based on generating and updating individuals in each generation. this continues until the optimal or close to the optimal solution is identified or until the stopping criterion is met [YCY+21]. These algorithms are based on the interaction between different individuals to find acceptable quality solutions. These algorithms garnered much attention because of their excellent performance and are particularly useful to solve optimization problems

[PC18]. PBAs include different heuristic algorithms such as Genetic Algorithms (GAs)[LSC05], Particle Swarm Optimization (PSO)[Shi04], Ant Colony Optimization (ACO)[MGDL04], Grey Wolf Optimization (GWO)[MML14], Fish Swarm Algorithms (FSA)[Li02], and Artificial Bee Colony (ABC)[Kar05]. The main difference between these algorithms is how a population is generated, and selected [YWC+18]. One of the main advantages of these algorithms is that, unlike many model-based optimization methods, they have the capability of parallelization [HKV19]. Among these algorithms, ABC is one of the most popular methods due to the excellent exploration and exploitation capabilities to find the satisfactory solution [KAO07]. Therefore, ABC was chosen for further exploration in this study.

Artificial Bee Colony (ABC)

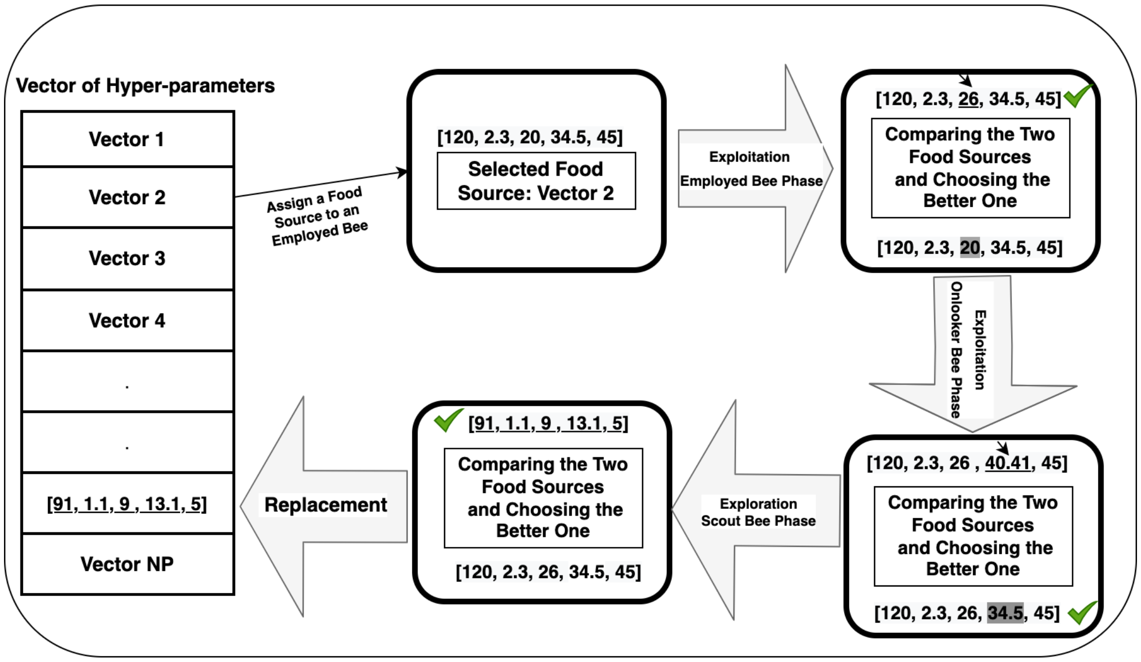
ABC, first defined by Karaboga [KB07], is one of the most recent and popular Swarm Intelligence (SI) algorithms that simulate honey bees' foraging behavior. In ABC, different groups of bees fly around in an area (search space). The bee colony is divided into 1) employed, 2) onlookers and 3) scouts. At the initialization stage, a random set of food sources get selected by **scout bee**, and their amount of nectar is determined. Then scout bees share the nectar information with employed bees and **employed bees** visit those food sources and choose a new food source in the vicinity and compare it with the other food source and select the better one. Next, the information is shared with **onlooker bees**, and they may or may not select a food source according to the information received (nectar amount). The higher the amount of nectar, the higher is the chance of getting selected by onlooker bees. If selected, the onlooker bee chooses a new food source in that neighborhood and compares the two food sources, and the preferred food source gets selected, and if not selected, the onlooker bee moves toward the next food source for evaluation. Employed and onlooker bees' responsibility is *exploitation* based on their own experience, where they leave poor food sources and move toward better ones through a greedy selection process. When a food source exploitation is exhausted or can not be further improved (based on trial limits), the employed bee discards the food source and becomes a scout bee and start *exploration* where it selects a new random food source and replaces it with the abandoned food source.The pseudo-code of the original ABC is described in Algorithm7 [KA09].

Algorithm 7 Implementation of original ABC

```
1: Initialize population
2: while Stopping criterion is not met do
3:     Assign food sources to employed bees
4:     Place the onlooker bees on the food sources based on the amount of nectar
5:     Send the scouts to different areas to find new food sources
6:     Memorize and update the best food source
7: end while
8: return  Best found food source
```

HPO using ABC

There are different optimization problems in multiple fields. To such problems, there exist classical approaches and heuristic approaches. Classical techniques are not efficient enough in solving optimization problems, and this is primarily due to dimensionality. Using brute-force search, looking for the best hyper-parameter in the search space is an NP-hard problem with a time-complexity of $O(n^k)$ (where k is the number of hyper-parameters, n is the number of distinct values for each hyper-parameter). Heuristic approaches such as genetic and evolutionary algorithms do not suffer from many drawbacks of classical methods when dealing with large search spaces caused by CoD. Among heuristics, ABC has shown more encouraging behavior. ABC is a population-based optimization technique and belongs to the family of stochastic SI methods. ABC is inspired by social interactions in honeybees and has recently garnered much success in different applications. It is as simple as some other swarm techniques such as PSO and is beneficial for NP-hard problems. Previous research shows that the ABC algorithm delivers relatively superior performance than PSO[KD16]. Hence, it is being explored in this study to find close to the optimal set of hyper-parameters in ML classifiers.



Vector of Hyper-parameters
Vector 1
Vector 2
Vector 3
Vector 4
.
.
[91, 1.1, 9 , 13.1, 5]
Vector NP
Assign a Food Source to an Employed Bee
[120, 2.3, 20, 34.5, 45]
Selected Food Source: Vector 2
Exploitation Employed Bee Phase
[120, 2.3, 26, 34.5, 45]
Comparing the Two Food Sources and Choosing the Better One
[120, 2.3, 20, 34.5, 45]
Exploitation Onlooker Bee Phase
[120, 2.3, 26 , 40.41, 45]
Comparing the Two Food Sources and Choosing the Better One
[120, 2.3, 26, 34.5, 45]
Exploration Scout Bee Phase
[91, 1.1, 9 , 13.1, 5]
Comparing the Two Food Sources and Choosing the Better One
[120, 2.3, 26, 34.5, 45]
Replacement

Figure 5.6: One iteration of the ABC-based Hyper-parameter Tuning (HyP-ABC) process as an example

Time Complexity

Table 5.4 summarizes GA, PSO, and ABC's advantages and disadvantages along with their time complexity. ABC requires a one-pass scan for each of the initialization, employed, and onlooker phases. Also, the ABC algorithm can repeat based on a maximum number of iterations/evaluations defined for the algorithm. The time complexity for each of the neighborhood searches is $O(N * D)$ where N is the population size (number of solutions, and D is the dimensionality. Therefore, the complexity of ABC can be presented as $O(3N * D * I_{max})$ and I_{max} is the fixed maximum number of iterations. Therefore the time complexity of the ABC is $(O(N \times D))$.

Advantages of ABC

BO [EFH+13], GA [LSC05] and PSO [GYW+08] are among the common approaches used for HPO problems. There are various characteristics that ABC has which make it efficient in solving optimization problems, especially for hyper-parameter tuning are as follows [XFY13]:

1. The underlying concept is easy to understand and implementation offer high accuracy.
2. Although almost all meta-heuristic make use of randomization to have global research as well as local search, ABC balances the exploration and exploitation steps (local and global search) [KA09]. This enables the algorithm to search for various parts of the search area. Randomization also helps the model not to get stuck in the local minimum and complete the global search. This is despite BO or PSO methods that may stick to a local optimum and fail to get to a global optimum [YS20].
3. Parallelization is one of the advantages of PBA because each population can be assessed on one machine [HKV19], while sequential methods such as BOs and GAs are challenging to parallelize since solutions are dependent on each other [DGMCEGC19].
4. Some PBA methods such as GA have some additional hyper-parameters (number of generations, crossover, mutation, and selection operators) to tune; therefore, GA has a lower convergence speed [LSC05]. Having fewer hyper-

parameters to be tuned by the user is one of the advantages of methods such as ABC.

Fit ABC to HPO Problems

In many optimization problems, including ML hyper-parameter tuning, some of the hyper-parameters are discrete. Therefore, the vectors of hyper-parameters could consist of discrete (integer and categorical) or continuous (float) values. Since the basic ABC is only applicable to continuous problems, proper strategies should be adopted to apply them to discrete or combinatory problems. Hence, the first step for modifying the original ABC to adapt to the HPO problems is configuring the initial population generation. Each sample should be described as a vector, including a set of hyper-parameters. Corresponding to each vector, there is a value for the objective function produced after training the ML model (model performance) and a fitness function calculated based on the objective function. After the population is generated, each vector is taken by one **employed bee** where it generates a new vector of hyper-parameters in the neighborhood, meaning that only one of the hyper-parameter in the current vector gets replaced with the same hyper-parameter of another vector. If the new vector has a better fitness, it gets replaced with the previous vector.
In the mutation step, **onlooker bees** start calculating the quality of the new population. Each of the vectors may or may not get selected depending on the computed quality. If a vector is selected, the onlooker bee generates a new vector of hyper-parameters in the vicinity and assesses it by comparing it with the previous one.

Similar to the employed bee phase, the better food source gets replaced. Finally, an exhausted vector of hyper-parameters (the vector that had the chance to improve the fitness but did not) is replaced with a random vector generated by the **scout**

bee. Unlike the vectors generated in employed and onlooker phases, the new vector changes all the hyper-parameters (hence, not in the vicinity). This phase helps the algorithm not to stick to the local optimum. The above phases repeat until a stopping criterion is met where the algorithm stops and returns the best performance achieved so far. The stopping criteria can be based on the number of iterations, evaluations, or time period. In this study, the stopping criteria is a constant number of evaluations for all the experiments.

Converting continuous ABC to combinatory version

The schematic objective of HyP-ABC is shown in Figure 5.6. As mentioned in the previous section, we face a combination of hyper-parameters types while the original ABC works only with continuous problems. Also, the range of variables in basic ABC is the same for all the dimensions. Therefore, proper strategies should be adopted to make them applicable to HPO problems.

5.2.7 HyP-ABC: Modified ABC Algorithm

There are different strategies used for tackling discrete optimization in swarm optimization [ZLGP12, WT08, DKA17, CKH$^+$18]. Rounding off is one of the most common approaches to tackle discrete variables. In this approach, the integer and continuous variables are treated similarly during the optimization process. However, when the optimization process is done in most studies, the variables are rounded off to the closest integer number.

Simplicity and low computational cost are among the main advantages of this method. However, entering impossible regions, and high variation of fitness between rounded and original values are among the disadvantages of this method. This study adjusts each vector's element based on its type and range in each iteration. Hence, the precise accuracy with the precise hyper-parameters is returned at the end of the optimization process. The remainder of this section explains how this process works.

As mentioned, the modified algorithm should treat various hyper-parameters differently in each iteration of generating and re-generating food sources. In this study, categorical variables in the initialization phase are encoded to integer numbers and treated as discrete variables afterward. Then, they are converted again to corresponding values after the optimization process is done and before training the model.

From this step on, if the selected hyper-parameter type by the employed/onlooker is discrete, the newly generated food source is modified to an accepted vector within the defined range for that specific dimension. For continuous variables, the algorithm performs similarly to the original ABC.

Algorithm 8 Hybrid artificial ABC algorithm

```
Input: $Population\_size$, $Search\_Space$
Output: Best food source identified
 1: Call a function to create an initial population $X$ based on the hyper-parameters
    type and range
 2: Set the trail=0 for all food sources
 3: while Stopping criteria is not met do
 4:    for $i$ to $Population\_size$ do
 5:       Employed bee generates a new food source in neighborhood ($N_i$) and mod-
          ifies the new food source to an accepted food source $M_i$ (type and range)

 6:       if $M_i == Xi$ then
 7:          Move to step 5
 8:       end if
 9:       Train the ML classifier with the modified food source
10:       if $f(M_i) < f(Xi)$ then
11:          $X_i = M_i$
12:          Reset $trial_i$
13:       else
14:          Increment $trial_i$ by 1
15:       end if
16:    end for
17:    for $i$ to $Population\_size$ do
18:       if $(rand(0, 1)) > (P_i)$ then
19:          Onlooker bee generates a new food source in neighborhood ($N_i$) and then
             modifies the new food source to an accepted food source $M_i$
20:          if $M_i == Xi$ then
21:             Move to step 19
22:          end if
23:          Train the model with the updated food source
24:          if $f(M_i) < f(X_i)$ then
25:             $X_i = M_i$
26:             Reset $trial_i$
27:          else
28:             Increment $trial_i$ by 1
29:          end if
30:       else
31:          Onlooker disregards the food source and moves to the next food source
32:       end if
33:    end for
34:    Memorize and update the best food source
35:    if trial >limit then
36:       Scout bee generates a new food source
37:    end if
38: end while
39: return Best achieved food source
```

Algorithm 8 shows the detailed steps for HyP-ABC. As shown, another step is also added to check if the updated food source equals the current food source to prevent training the same vector of hyper-parameter and save time.

This algorithm is specifically useful when the selected hyper-parameter for an update is discrete (due to having fewer options and a higher chance of getting the same value after the update). For binary categorical variables (such as criterion in RF), we value flip the hyper-parameter value ($flipped = 1 - binary\_value$). Values outside the ranges defined for the hyper-parameters in all phases get replaced with lower or higher bound values before training the ML model. The stopping condition in this algorithm is when the algorithm evaluates a specific number of evaluations.

HyP-ABC Steps

Regarding the initial population, each solution is a vector of hyper-parameters, X_i, with the length of D, where D is the dimension of the vector of the number of hyper-parameters. Unlike the original ABC that works only for continuous problems HyP-ABC also handles the other types of variables namely, categorical and integer (discrete) values. Moreover, unlike the original ABC, variables of the same type may have different ranges. Therefore algorithm needs to be fitted so that each variable has an acceptable range for the corresponding hyper-parameter.

$$X_{i,j} = x_{min,j} + rand(0,1)(x_{max,j} - x_{min,j}) \tag{5.1}$$

Where $X_{i,j}$ is the hyper-parameter value of food source i in dimension j, and $x_{max,j}$ and $x_{min,j}$ are upper-bound and lower-bound of j_{th} hyper-parameter.

Then, each vector is assigned to an employed bee. The employed bee generate a new food source $N_{i,j}$ in that neighborhood by changing one of the hyper-parameters

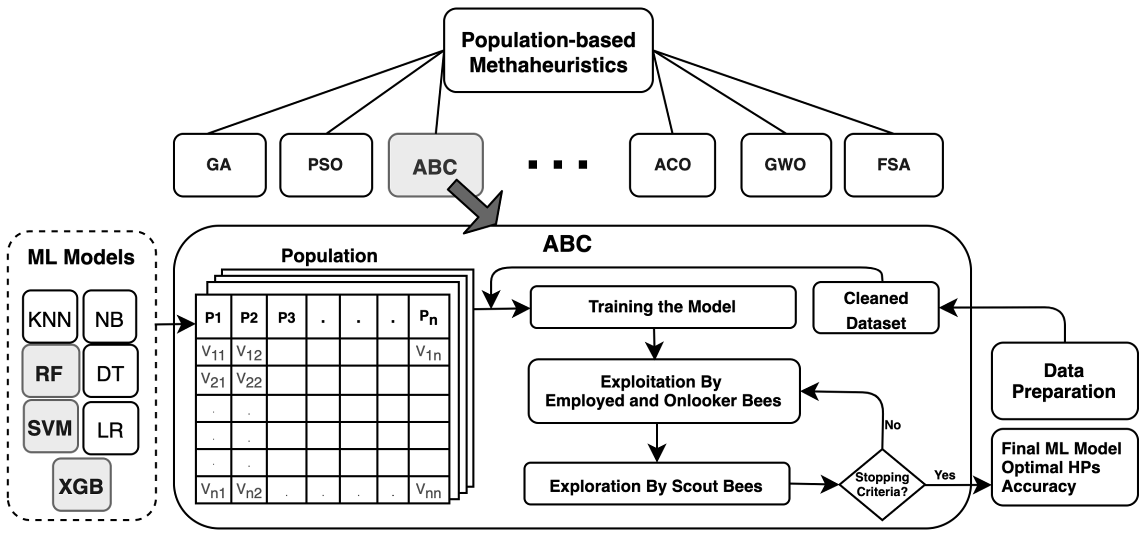


Figure 5.7: The schematic process of the study

using below equation:

$$N_{i,j} = x_{i,j} + rand(-1,1)(x_{i,j} - x_{k,j}) \tag{5.2}$$

Where $x_{k,j}$ represents the corresponding hyper-parameter value in the neighborhood. Then the fitness of the N_i and X_i are compared and the better food source becomes a new member of the population afterward. Fitness is calculated from the objective function:

$$fit_i = \begin{cases} \frac{1}{1+f_i}, & f_i \geq 0 \end{cases} \tag{5.3}$$

When employed bees are done with their part, they share the information with onlooker bees. Based on the received information they decide whether to further exploit a vector or not. In this phase, roulette wheel selection is utilized to calculate the probabilities, as shown below:

$$P_i = \frac{f_i}{\sum_{j=1}^{PN}(f_j)} \tag{5.4}$$

Where f_i is the fitness value for the $i$the food source and PN is the population size. Next, scout bees start exploring random configurations in search space without using experience or memorizing the locations. They do not use greedy selection when exploring new food sources. Exploration by scout bees is through the below equation:

$$X_{i,j} = x_{min} + rand(0,1)(x_{max} - x_{min}) \tag{5.5}$$

HyP-ABC algorithm also skips the repetitive food sources after modifications (as shown in lines 6 and 20 of the Algorithm 8) to prevent repetitive training of the ML classifier to reduce the tuning time.

5.2.8 Experimental Methodology

In this section, we describe the methodology used in this study aiming to improve the efficiency of a recent study [ZMR+21] in 2021, using conventional HPO methods. The process is performed in several consecutive phases, and it includes data pre-processing, feature engineering, leveraging ML classifiers, and ABC-based optimization steps.

Data pre-processing and feature Engineering

Data pre-processing is a required step in data analysis and data science to transform the raw data into an understandable format for ML algorithms [GLH15]. It includes data cleansing, data normalization, and data reduction. For data cleansing, we remove the features with more than 60% percent of the values missing. Data normalization is also a method to standardize the data when the different features' values vary widely. *StandardScaler* from Scikit-learn library was used for scaling the input data. We filter the data in the data reduction step and included only the target population, students from computing fields. One-hot encoding (categorical encoding variables to binary variables for each unique category) is also used in the feature engineering step. All the above steps are done using Python 3.7 programming language.

Hyper-parameter tuning

Tuning hyper-parameters is essential to yielding the best performance of an ML model. Due to the varied nature of the hyper-parameters in different ML models, we implement a HyP-ABC to tune the hyper-parameters of three ML classifiers. HyP-ABC is replaced with automated HPO methods performed in [ZMR+21]. Algorithm 9 shows the general steps of HyP-ABC and Figure 5.7 shows the schematic process of HyP-ABC framework.

Algorithm 9 HyP-ABC general steps

```
Input: $List_{ML}$ of models and raw dataset
Output: Best model, performance and optimal architecture
1: Call PreProcessing
2: for every model in the $List_{ML}$ do
3:     Call HyP-ABC
4:     if ABC.ACC larger than reported (GS &RS).Acc then
5:         Replace Model, ABC BestHPs
6:     end if
7: end for
8: return FinalModel, BestHP, Final.Acc, TuningTime
```

Since RF, XGBoost, and SVM are computationally expensive when using automated HPO methods [ZMR+21], they are selected to be explored in this study. The tuning time complexity in these models is mainly due to the model's complexity or the ranges of hyper-parameters.

5.2.9 Experimental Results

In this section, we discuss the dataset, evaluation metrics, and experimental results in depth.

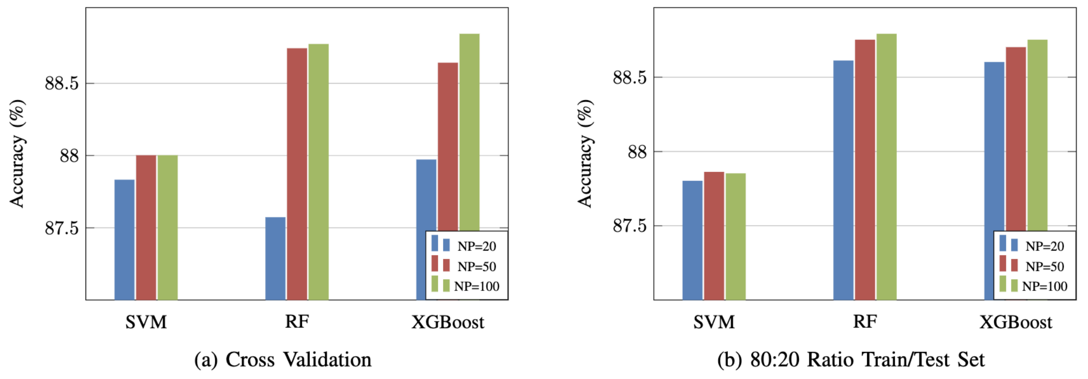


Figure 5.8: Effect of number of food sources on accuracy after specific number of iterations

Table 5.5: Performance evaluation of utilizing HPO methods to the ML classifiers on the MIDFIELD dataset (CV=3)

	Accuracy (%)		
HPO Method / ML Classifier	**RF**	**XGBoost**	**SVM**
GS (minimized search space)	86.24	86.33	85.38
RS	87.37	86.90	86.03
HyperBand	86.74	87.75	87.54
BO-GP	88.71	88.71	87.91
BO-TPE	88.33	88.73	87.90
PSO	88.76	88.63	87.98
GA	87.51	88.56	**88.08**
HyP-ABC	**88.77**	**88.84**	88.00

Dataset

In this study, we use real-world educational data, Multiple-Institution Database for Investigating Engineering Longitudinal Development (MIDFIELD)[OZTA04, LOO+22], to predict students' success in computing majors. This dataset is a unit-record longitudinal database for all undergraduate students from 19 universities across the United States. MIDFIELD contains all the information shown on students' academic records, such as demographic data and information about field, enrollment, graduation, and their academic performance before and during school. We used a reduced version of MIDFIELD, including only students majoring in computing fields (with CIP=11). This version of the MIDFIELD dataset is used for a binary classification problem and has 45322 samples with 91 features.

Metrics for Performance Evaluation

The metric used in this experiment to assess the performance of the classification and the proposed framework is accuracy. Accuracy is the fraction of the number of correct predictions to the total number of predictions [BGCL00]. This metric is used since the dataset is balanced [GFB$^+$11] and the number of samples belonging to each class is almost equal(46:54).

Cross-validation (CV) is used to reduce or prevent over-fitting. CV helps finding a stable optimum that is suitable for all the subsets of the dataset instead of only a particular validation set [HKV19]. It is important to note that CV also increases the tuning time by the number of folds. Therefore, we used 3-fold cross-validation to avoid high tuning times. However, we paralleled the CV process to reduce the impact of CV on execution time. We also separated the MIDFIELD dataset into train and test sets and repeated the experiment. The train set contains 80% of the dataset, and the test set includes 20% of the dataset.

In this experiment, the Scikit-learn and XGBoost libraries were used to leverage ML classifiers. All experiments were conducted using Python 3.7 on AWS servers with four v-CPUs up to a 3.0 GHz scalable processor and 16 GB RAM.

5.2.10 Results

HyP-ABC is developed to achieve the optimal or close to the optimal set of hyper-parameters of SVM, RF, and XGBoost algorithms. The higher number of hyper-

Table 5.6: ABC-based Hyper-parameter tuning (HyP-ABC) Execution time on the classifiers (When termination point is set to desirable accuracy= 88%)

80:20 ratio Test-Train set			
	Tuning Time (hrs)		
Population size (N)	**20**	**50**	**100**
SVM	23.95	53.05	62.57
RF	2.37	9.07	3.18
XGBoost	4.07	5.85	4.23
With 3-Fold Cross Validation ($N = 50$)			
SVM	34.15		
RF	10.55		
XGBoost	29.22		

parameters with possible wide ranges or complexity and the model's objective function is the motivation for selecting these ML classifiers.

Figure 5.8(a) and 5.8(b) show the effect of population size on accuracy among the three ML classifiers using CV as well as using train and test sets; As can be seen, the larger the population size, the better the accuracy. However, it shows that the impact of population size decreases when it gets very large (large populations also have the disadvantage of increasing the execution time). Our experiment considers the population size as 50 for comparison with other state-of-the-art HPO methods.

Table 5.6 shows the execution times when setting a desirable accuracy as the termination point (88%). As can be seen from the table, SVM has the highest execution time due to the SVM model's complexity and expensive objective functions. Table 5.5 summarizes the results and shows the accuracy of different HPO methods among different ML classifiers having the same resource (number of iterations). As shown, the accuracy of the HyP-ABC is or more than the rest of the approaches for the tree-based methods, showing the algorithm's capability for tuning ML models hyper-parameters in RF and XGBoost classifiers. As for SVM Classifier, GA has better marginal accuracy. Also, GA offers more hyper-parameters to tune compared to HyP-ABC, which still makes the proposed approach a promising alternative to GA.

It is important to note that for RF and XGBoost algorithms, the GS tuning time considering all the configurations leads to a tuning time of more than years. This estimation was calculated from the average tuning time for each configuration from the RS experiment. Hence, the experiment for the GS method is implemented using a reduced search space with larger steps to make the experiment feasible.

Unlike model-free methods such as GS and RS, HyP-ABC presented in this study is a model-based method. In other words, it has a methodical way of moving toward the optimal solution. The results of the study show that among all the ML algorithms developed in this experiment, XGBoost had the best accuracy score. Hence, XGBoost is the final model to be used on the MIDFIELD dataset.

5.2.11 Conclusion and Discussions

Hyper-parameter tuning is an essential step in automated ML, and it reduces the burden of manual tasks and is economical and time-efficient. However, searching through an ample space (wide ranges) of hyper-parameter is a time-consuming task. In this chapter, a modified evolutionary optimization algorithm for hyper-parameter tuning in ML classifiers referred to as HyP-ABC is proposed to enable hyper-parameter tuning of ML classifiers.

One of the limitations of this study is that it is developed for classification problems. Other types of problems, such as regression are the focus of our future work. Also, HyP-ABC is aimed to be tested on structured datasets, in this case, an educational dataset (MIDFIELD)[LOO+22] as such applications need explainable results. Hence, three interpretable ML algorithms are being explored in this study. The experimental results show the competitiveness of the proposed approach compared to other model-free methods and are ready to support users in various other applications. Another limitation of this work is that the proposed method is not compared with all the methods existing in the literature (such as i-race, paramILS, SMAC) as this algorithm provide an alternative approach for hyper-parameter tuning using evolutionary algorithms. However, they will be considered in future extensions of this work.

HyP-ABC improves the classification accuracy and can be deployed to solve HPO problems with large search spaces. To summarize, HyP-ABC is recommended for optimizing RF, XGBoost, and SVM. In future work, we will explore the semantic initialization of the population to improve the convergence rate further. Also, the time complexity of algorithms such as SVM is highly dependent on the number of features and samples of a dataset, and hence their function evaluation can be prolonged for large-scale datasets.

5.3 OptABC: An Optimal Hyperparameter Tuning Approach for Machine Learning Algorithms

Abstract [2] Hyperparameter tuning in machine learning algorithms is a computationally challenging task due to the large-scale nature of the problem. In order to develop an efficient strategy for hyper-parameter tuning, one promising solution is to use swarm intelligence algorithms. Artificial Bee Colony (ABC) optimization lends itself as a promising and efficient optimization algorithm for this purpose. However,

[2]This chapter is an edited version of the author's previous work published in [ZMA21c] ©2021 IEEE.

in some cases, ABC can suffer from a slow convergence rate or execution time due to the poor initial population of solutions and expensive objective functions. To address these concerns, a novel algorithm, OptABC, is proposed to help the ABC algorithm in faster convergence toward a near-optimum solution. OptABC integrates an artificial bee colony algorithm, K-Means clustering, greedy algorithm, and opposition-based learning strategy for tuning the hyper-parameters of different machine learning models. OptABC employs these techniques in an attempt to diversify the initial population, and hence enhance the convergence ability without significantly decreasing the accuracy. In order to validate the performance of the proposed method, we compare the results with previous state-of-the-art approaches. Experimental results demonstrate the effectiveness of the OptABC compared to existing approaches in the literature.

Keywords: Automated Machine Learning, Automated Hyper-Parameter Tuning, Artificial Bee Colony Algorithm, Evolutionary Optimization

5.3.1 Introduction

Overview

Nature-Inspired Optimization (NIO) and Machine Learning (ML) are two prominent and momentous sub-fields of Artificial Intelligence (AI). ML algorithms use data through experience and improve automatically, to solve different problems and generate knowledge. Recently, NIO has garnered much attention to solve such problems efficiently and effectively. These algorithms includes but not limited to Genetic Algorithms (GAs) [Sch01], Particle Swarm Optimization (PSO) [KE95], Ant Colony Optimization (ACO) [DBS06], and Artificial Bee Colony (ABC) [Kar05]. Among these approaches, ABC has been widely used in the literature due to its strong global search capabilities and a low number of parameters as compared with other nature-inspired algorithms. Therefore, it has been utilized in a wide range of applications to solve complex optimization problems [DSC19, MSAA21, MAA20a, GSPK18].

ABC is a swarm intelligence method in which different solutions are called food sources. The initial set of solutions is generated merely based on random distribution. In ABC, three different types of bees use their search strategy to achieve new food sources. **Employed** and **Onlooker** bees have similar search strategies and are responsible for exploitation, while **Scout** bees are responsible for exploring and inserting new solutions into the population.

Some of the previous studies report on the slow convergence of the primary ABC method or getting stuck to local optima [SZS17]. According to the literature, many of the ABC algorithms ignore the role of population initialization [BA17]. Therefore methods that can provide richer populations can be very beneficial regarding both convergence rate and finding better solutions.

In the context of hyper-parameter tuning of ML algorithms, assuming that there are M combinations and P hyper-parameters per combination, we can construct an $M \times P$ matrix. In case M and P are too large, it is too expensive to run the ML model on all the possible configurations. Additionally, ML algorithms have different objective functions depending on the complexity of the models themselves. For instance, long training time especially for large datasets is actually one of the SVM disadvantages [YHB16], because of its costly objective function. Support Vector Machines (SVMs) are supervised ML models that utilize associated learning algorithms to detect patterns existing in data. SVM applies to regression, classification, and outlier detection problems. Taking the training data marked with their belonging classes, SVM creates a classifier using a hyper-linear separating plane and builds a model that predicts the classes of new data points.

The high time complexity of SVM models (especially when working with large datasets) indicates expensive objective functions in the ABC algorithm as well, which leads to high execution time in comparison to other ML models [ZMA21b]. Random Forest and eXtreme Gradient Boosting (XGBoost) models are also among ensemble supervised machine learning models that use enhanced bagging and gradient boosting, respectively. These two models have proved a powerful predicting ability in different applications. However, they have more number of main hyper-parameters in comparison to other ML models which makes a large search space and hence increases the time complexity of the ABC algorithm.

In this study, we propose a modified version of the HyP-ABC algorithm in a previous work [ZMA21b] called OptABC, to tune the main hyper-parameters of the mentioned ML algorithms; The K-means clustering algorithm is used to offer heterogeneity to the population of solutions. This method can enhance the convergence rate by avoiding the evaluation of all solutions in the population by taking only the cluster centroids and evaluating them. Moreover, we add an Opposition-Based Learning (OBL) method to random food source search in the original scout phase to discover richer and unvisited food source positions to improve the balance in exploration and exploitation steps. The MIDFIELD dataset is used to verify the performance of OptABC in the experiments.

Motivation

ML model predictions are proven to be effective for accurate decision makings. However, to make a model work at its best, its hyper-parameters must be tuned. In Hyper-Parameter Optimization (HPO) problems, the goal is to build a model with the best set of hyper-parameters for an ML model to minimize the objective function or maximize the accuracy [SCZZ19]:

$$x = arg \min_{x \in S} f(x) \tag{5.6}$$

where $f(x)$ is the objective function or error rate that should be minimized, x is the optimal set of hyper-parameters, S is the search space, and $arg \min f(x)$ is the optimal set for which $f(x)$ reaches its minimum.

However, searching through a large number of configurations of hyper-parameters can be computationally highly expensive [ZMR+21]. Many of the HPO problems are non-convex optimization problems, meaning that they have multiple local optimums. Therefore, traditional optimization approaches are not a good fit for them [Luo16b]. Recently, evolutionary optimization techniques have gained much success in different applications to solve non-convex complex optimization problems [YWC+18]. These techniques do not guarantee to find the global optimum; however, they detect near to global optimum within a few iterations.

Existing automated hyper-parameter tuning techniques suffer from high time complexity for some of the ML models, especially SVM [ZMA21b] and tree-based models. This is due to the high time complexity of the SVM objective function, especially for large size datasets [MDG10], and or the number of hyper-parameters in the search space.

This chapter proposes OptABC, an automated novel hybrid hyper-parameter optimization algorithm using the modified ABC approach, to measure the ML models' classification accuracy. The main focus of this study is on the ABC algorithm. ABC operates based on the foraging behavior of honeybees and was first developed by Karaboga [Kar05]. Previous studies show that the performance of the ABC algorithm is competitively better than that of other population-based algorithms [S+17]. Thus, it has been used to solve different problems in different applications.

As mentioned in the previous section, slow convergence because of the stochastic nature [KLL13] and sticking in the local optima for complex problems [CZZ09] are among the disadvantages of the ABC algorithm. The challenge is exacerbated when dealing with expensive objective functions or large search spaces for evaluating the quality of the food sources.

In this chapter, an effort has been made to address the challenges mentioned above to make the HyP-ABC algorithm [ZMA21b] more efficient for ML algorithms to solve optimization problems.

Research Focus and Contribution

In this research study, we mainly focus on the slow convergence issue of the ABC algorithm. This issue can be derived from the shortcomings of the ABC algorithm and the complexity of ML models. These shortcomings are summarized as follows: Initial position of food sources, the position of the food sources by scout bees, and expensive evaluation function in the foraging process. To address these concerns, we propose an improved version of a previous ABC-based algorithm. The contribution of this section is threefold as follows:

1. Presents an automated hyper-parameter tuning method for different ML models using evolutionary optimization for large real-world datasets.
2. Presents a K-Means clustering approach to form a better initial population.
3. Determines the abandoned food source position by applying the OBL method in addition to the random search strategy to strengthen the exploration phase of ABC.

The proposed algorithm is evaluated on a real-world educational dataset, and the results are compared with an algorithm in a previous study in 2021[ZMA21b]. The proposed method provides better performance in terms of execution time without decreasing the accuracy in most of the cases.

Organization

The rest of this section is organized as follows. Section 5.3.2 highlights the background and related work regarding ABC algorithm and variants of ABC in different applications. Section 5.3.3 elaborately explains our novel approach. Section 5.3.4 and 5.3.5 presents our experimental methodology and results, followed by section 5.3.6, that concludes the section.

5.3.2 BACKGROUND REVIEW AND RELATED WORK

This section discusses the background review of this work, including the basic ABC algorithm, its different steps, and ABC-related works in different applications.

Overview of ABC Algorithm

ABC algorithm works based on the natural foraging process of honey bees swarm and was first introduced by Karaboga in 2005 [Kar05] to solve optimization problems. As mentioned in the previous section, this algorithm consists of three different phases. Initialization and exploration are performed by the scout bee; the Scout bee randomly picks the food sources, then each food source is assigned to the Employed bee for exploitation. Afterward, the Onlooker bees wait in the hive for the information that the Employed bees share with them. Once they received the information, they may exploit or abandon the food sources based on their quality. The abandoned food source is replaced with a discovered random food source by the Scout bee. This process continues until the termination criterion is achieved. A summarized mechanism of the original ABC algorithm is given below: Initialization The initial population containing different food sources (vectors) using a random strategy is created using the below formula: Then each of the vectors is assigned to one employed bee.

$$X_{i,j} = x_{min,j} + rand(0,1)(x_{max,j} - x_{min,j}) \tag{5.7}$$

Where $rand()$ returns a random number within the provided range. Employed Bee Phase Each of the Employed bees detect a neighbor food source, V_i, by equation 5.8. This is done by changing only one of the dimension's values kth in the vector. Then, the employed bee compares V_i with X_i and selects the better vector.

$$V_{i,j} = x_{i,j} + rand(-1,1)(x_{i,j} - x_{k,j}) \tag{5.8}$$

Where k and i are different.

Onlooker Bee Phase In the next phase, Employed bees share the information with onlooker bees. Then onlooker bees calculate the probabilities of vectors (P_i), based on roulette wheel selection in equation 5.9. Onlooker bees select or leave the vectors based on the calculated probability values. The vector with a higher probability is more likely to get selected by the Onlooker bees.

$$P_i = 0.9\frac{Fit_i}{max(Fit)} + 0.1 \tag{5.9}$$

Where Fit_i is the fitness value for the ith vector and is directly calculated from objective function, by equation 5.10.

$$fit_i = \begin{cases} \frac{1}{1+f_i}, & f_i \geq 0 \\ 1 + abs(f_i), & f_i < 0 \end{cases} \tag{5.10}$$

If an onlooker bee selects a vector, it generates a new vector solution similar to equation 5.8 and selects the better solution between V_i and X_i. Next, the best vector of solutions visited thus far is memorized.

Scout Bee Phase In the last phase, an exhausted food source (vector), if any, is determined by scout bee and exploration starts. In this phase, scout bee generate a random vector by equation 5.11 and replace it with the exhausted vector. This step is done without using any experience or greedy approach.

$$X_{i,j} = x_{min,j} + rand(0,1)(x_{max,j} - x_{min,j}) \tag{5.11}$$

The above phases repeat until the resource budgets, such as time or the maximum number of evaluations are exhausted, or when the desired results are achieved.

Different Variants and Applications of ABC

This section describes some of the related studies of ABC employed in various optimization problems. Researchers have used ABC to address feature selection problems and speed up the classification process [MA14] and eliminate non-informative features [MA14, SEO21, MSM19]. ABC has also been used to design automatically and evolve hyper-parameters of Convolutional Neural Networks (CNNs) [ZYC+19]. Choong et al. improved the neighborhood search algorithm of the original ABC in the exploitation phases of the original ABC [CKH+18]. The experimental results of this study showed that the proposed function has some advantages over other search processes [CKH+18]. In [MDPS21, ZZWC20], the authors proposed to modify the initialization and exploitation phases of ABC to improve the convergence. The authors tested the performance of their proposed algorithms and observed promising results.

Pandiri and Singh [PS18] developed a hyper-heuristic-based ABC algorithm for the k-Interconnected multi-depot multi-traveling salesman problem. The authors used an encoding scheme inside the algorithm. The study results showed that although the search space is smaller than previous schemes, it has better performance than previous methods applied in the literature. Gunel and Gor used a dynamically constructed hyper-sphere to modify the ABC algorithm to improve the exploitation ability of ABC. In this method, the best food source was used to define the mutation operator. The solutions for the differential equations were obtained by training neural networks employing the modified ABC [GG19].

In another study, Agrawal introduced an improved version of the ABC using some features of the Gaussian ABC to improve the slow convergence of the original ABC. This approach was used to modify the employed, onlooker, and scout phases. The experimental results of this study showed the superiority of the method over the original ABC in the majority of the experiments [A$^+$20].

Studies Employing Learning Techniques on ABC

Many previous studies use hybridization approaches combining ML algorithms and nature-inspired algorithms to improve their performance regarding accuracy and time. These studies aim to provide more intelligent forms of ABC algorithm in different applications. This section covers some of the previous studies that used learning algorithms to enhance the optimization process of the ABC algorithm. The most common methods used in the literature are clustering, reinforcement learning, and OBL.

Clustering has been used mostly in different ways in the initialization phase of the ABC algorithms. K-means clustering was either used to add diversity to the population or to detect the clusters in the optima [S$^+$17, IDS20, ZFLS19]. Reinforcement learning has also been used in several studies to improve the searching process of the ABC [ZZ20, FKPoS19]. This method is mostly used in the onlooker and employed phases of the ABC algorithm. OBL approach is another technique that has been used in previous studies to enhance the performance of the ABC method. This method has been employed in initialization [SA20, LLL$^+$19], employed and onlooker phases [SG18, ZLS15] .

5.3.3 Proposed Approach

One of the disadvantages of ABC, like other population-based algorithms, is requiring a suitable initial population. Also, ABC sometimes suffers from a slow convergence rate [ZMA21b]. In some applications, such as ML problems, inserting a solution into the problem and testing its impact on the results to compute the fitness value may require a long time [KAK20]. In such situations, learning-based models can be developed to guess the fitness values of the present solutions more quickly. Having a rich and diverse initial population can significantly improve the performance and convergence rate. Additionally, although ABC performs well at exploration, the Scout phase can still improve to guide the search process better and increase the likelihood of finding better food sources in the exploration process.

To address the above-mentioned inherent defects of ABC and accelerate the convergence rate of the algorithm, we propose an improved variant of an existing algorithm (HyP-ABC)[ZMA21b] using learning algorithms. Employing the Opt-ABC, the main hyper-parameters of the three models are optimized. Therefore the optimal solution would be a vector with a dimension that equals the number of main hyper-parameters. OptABC applies three modifications. We employ K-Means clustering and Opposition-Based Learning (OBL) strategies in the initial population generation. The third modification is constructed in the scout bee search using the OBL method. Figure 5.9 shows the overall flowchart process of our study.

The main structure of this novel algorithm, including two modified phases, can be summarized below.

The Generation of Initial Population

In HPO problems, each food source is a vector of hyper-parameters, X_i, j by equation 5.7, where $j = 1, 2, ..., D$, $i = 1, 2, .., PN$ and D and NP are the number of hyper-parameters and the number of population, respectively. The original ABC algorithm starts with generating randomly distributed food source locations. To improve the convergence rate of the ABC algorithm, we use a K-Means clustering algorithm

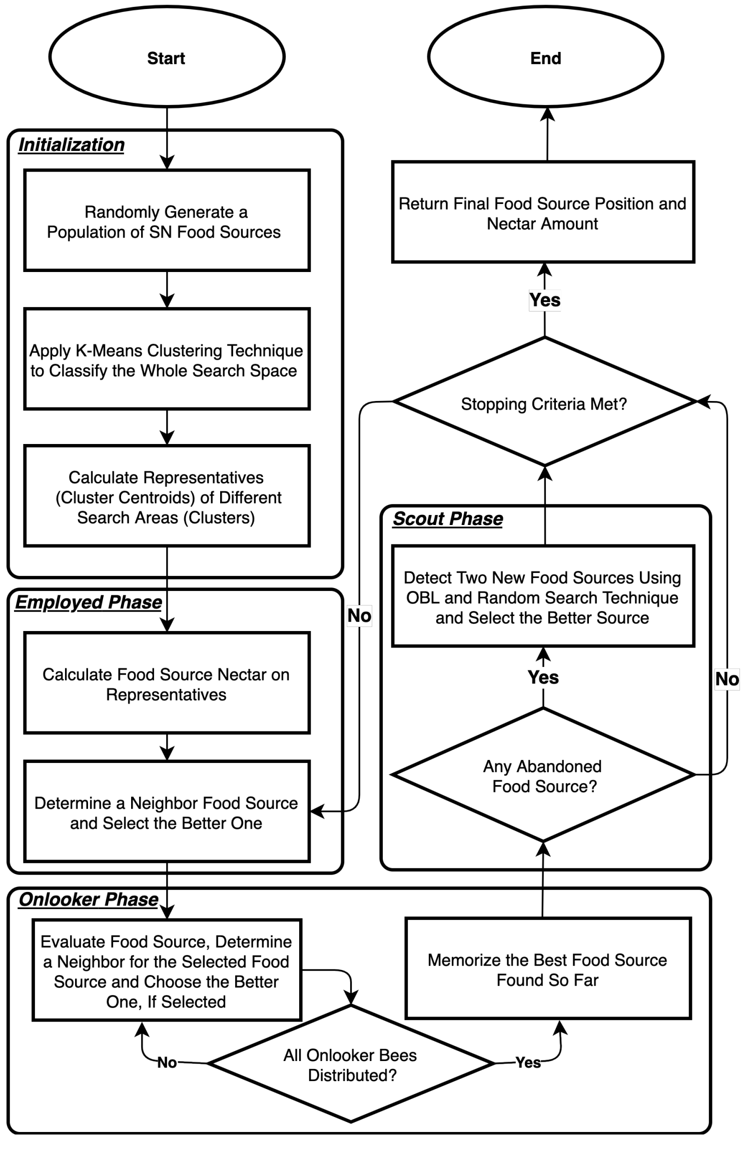

Start
End
Initialization
Randomly Generate a Population of SN Food Sources
Apply K-Means Clustering Technique to Classify the Whole Search Space
Calculate Representatives (Cluster Centroids) of Different Search Areas (Clusters)
Employed Phase
Calculate Food Source Nectar on Representatives
Determine a Neighbor Food Source and Select the Better One
Onlooker Phase
Evaluate Food Source, Determine a Neighbor for the Selected Food Source and Choose the Better One, If Selected
All Onlooker Bees Distributed?
No
Yes
Memorize the Best Food Source Found So Far
Scout Phase
Any Abandoned Food Source?
Yes
No
Detect Two New Food Sources Using OBL and Random Search Technique and Select the Better Source
Stopping Criteria Met?
No
Yes
Return Final Food Source Position and Nectar Amount

Figure 5.9: Overall flowchart process of Optimal ABC-based tuning (OptABC)

[M+67] to have a more diversified population. In this approach, we partition the randomly initiated population into k clusters. Then a loop starts assigning the food sources to the closest mean, and the cluster centroids (c^i) are determined. This process continues until no change is observed in the position of cluster centers. Instead of calculating the objective function for all the food resources, the final clusters' centroids (by equation 5.12) are taken as representatives of each cluster for a new population.

$$\mu_i = \frac{\sum_{j=1}^{PN} 1\{c^i = j\} x^i}{\sum_{j=1}^{PN} 1\{c^i = j\}} \qquad j = 1, 2, ..., k \tag{5.12}$$

This approach helps to avoid evaluating expensive objective functions for each single food source in the initial population. The pseudo-code of the modified initialization step is described in Algorithm 10.

Algorithm 10 HyP-ABC general steps

```
Input: Number of clusters ($k$)
Output: Cluster centroids/ New population
1: Randomly generate a population of PN food sources
2: Randomly select $k$ food sources from the generated population and assign them
   to each cluster
3: while centroid position changes do
4:     Assign each food source to its closest cluster
5:     Calculate new centroids (mean) of all clusters using equation 5.12
6: end while
7: Take final centroids as $population_{new}$
8: return $population_{new}$
```

Employed Bee Phase

In this phase, employed bee discovers a new food source, V_i, in the vicinity of the current food source. In other words, only one hyper-parameter of the current food source changes in this phase. Then, the employed bee compares V_i with X_i and selects the better food source.

$$V_{i,j} = x_{i,j} + rand(-1,1)(x_{i,j} - x_{k,j}) \tag{5.13}$$

Onlooker Bee Phase

Next, employed bees share the information with onlooker bees through waggle dance. Onlooker bees in the OptABC algorithm compute the probabilities of each food source. Then the probability of the food source is then compared to a random number between zero and one. Depending on the quality of the food source, onlookers may leave or exploit them. In this study, the probability of the food sources is calculated using min-max normalization in Equation 5.14 ($0 \leq P_i \leq 1$, $min(Fit) \leq Fit_i \leq max(Fit)$). In other words, the range of all fitness values is normalized so that each value contributes approximately proportionately when it is compared to the random number. Then a comparison is made between the probability and random numbers for further decisions.

$$P_i = \frac{Fit_i - min(Fit)}{max(Fit) - min(Fit)} \tag{5.14}$$

If the food source gets selected for further exploitation, a new solution is generated using equation 5.13, and the algorithm goes on with the best solution.

The Search Mechanism of Scout Phase

In the Scout phase, if a food source does not get updated a defined number of times (limit), it will be considered an abandoned food source, and its assigned employed bee turns to a scout bee for discovering a new food source location. In original ABC, this is done by using equation 5.11. As can be seen, the new food source is generated randomly and may not always be the best solution for an optimization problem. Hence, in this study, we consider operator adaptation to generate the new food source. In other words, in addition to detecting a new location for the abandoned food source using a random search technique, we also apply an OBL search technique. OBL technique is a new concept in ML which was first introduced by Tizhoosh (2005) [Tiz05] which is inspired by the opposition concept in the real world, Implying that Having a data point that is closer to an optimum point can result in faster convergence. However, if the optimum point is in the opposite direction (position), the search process needs more resources to find them. Therefore, searching in the opposites locations may help the algorithm converge faster. The opposition of each food source (a vector with D parameters) is calculated by equation 5.15 to generate an alternative food source in the opposite location. This technique strengthens the exploration phase and better guides the search process. In this phase, the food source with better quality gets replaced with the abandoned food source.

$$\tilde{x}_{i,j} = x_{max,j} + x_{min,j} - x_{i,j} \tag{5.15}$$

Fitness Function of OptABC

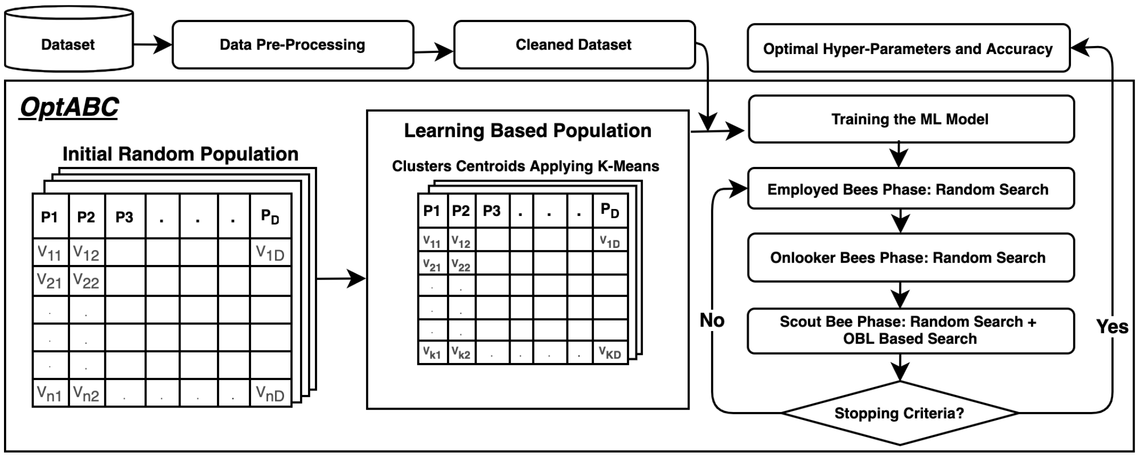
The fitness function for all the phases above in the proposed algorithm is calculated directly from the objective function (5.10). The main goal in ML-related problems is to maximize the objective function, which is computed from performance indicators such as accuracy for classification problems or mean absolute error for regression problems. In this study, our goal is to maximize the training objective function, which is accuracy-oriented and is calculated by the below formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.16}$$

Where TP, TN, FP, and FN are true positives, true negatives, false positives, and false negatives, respectively. Since we are dealing with a balanced dataset we use accuracy as the overall performance metric.

Framework and Time Complexity of OptABC

The proposed framework of OptABC is given in Figure 5.10, and Algorithm 11 shows the previous work process[ZMA21b], along with the changes we made to improve the ABC convergence rate.



Dataset
Data Pre-Processing
Cleaned Dataset
Optimal Hyper-Parameters and Accuracy
OptABC
Initial Random Population
Learning Based Population
Clusters Centroids Applying K-Means
Training the ML Model
Employed Bees Phase: Random Search
Onlooker Bees Phase: Random Search
Scout Bee Phase: Random Search + OBL Based Search
Stopping Criteria?
No
Yes

Figure 5.10: Proposed framework for Optimal ABC-based tuning (OptABC)

In this method, the number of initial solutions is PN, and the number of the secondary population is $\frac{PN}{K}$. The number of employed bees and onlooker bees is the same (k). In the employed bees phase, k new solutions are generated. Similarly, another k solutions are generated by onlooker bees. In the scout bee search phase, two solutions (One OBL-based solution and one random-based solution) are generated for each exhausted solution, and the best one is selected to get replaced with the exhausted food source. The random solution helps the algorithm escape from the local optimum, and the opposition solution can increase the probability of reaching better candidate solutions (compared to random search) [RTS08].

Algorithm 11 OptABC Algorithm - Modifications

```
Input: $population\_size$, $Search\_Space$
Output: Optimal or close to the optimal configuration
 1: Call Algorithm 10
 2: while Stopping criteria is not met do
 3:    for $i$ to $population_{new}\_size$ do
 4:       Employed bee generates a new food $N_i$ source in neighborhood and modify
          the new food source to an accepted food source $M_i$
 5:       Train the ML model with the updated food source
 6:    end for
 7:    for $j$ to $population_{new}\_size$ do
 8:       Onlooker bee calculate the probability $(Pi)$ of each solution based on Equa-
          tion 5.14
 9:       if $(rand(0,1)) > (P_i)$ then
10:          Onlooker bee generates a new food source in neighborhood $N_i$ and then
             modify the new food source to an accepted food source $M_i$
11:          Train the ML model with the updated food source
12:       else
             Onlooker disregard the food source and moves to the next food source
13:       end if
14:    end for
15:    Update the best food source and store it
16:    if trial >limit then
17:       Scout bee generates a food source based on Equation 5.11
18:       Scout bee generates a food source based on Equation 5.15
19:       Select the food source with better quality
20:    end if
21: end while
22: return Optimal food source
```

The time complexity of the proposed approach has not increased in comparison with the traditional ABC algorithm since not all solutions in the population are evaluated. ABC algorithm's time complexity depends on population size (PN), complexity of objective function (f) and maximum number of iterations (I_{max}) and number of clusters (k). Therefore, the time complexity of the proposed method is $O(I_{max}.(\frac{PN}{K}.f + \frac{PN}{K}.f + 2f)) = O(I_{max}.(\frac{PN.f}{K}))$, while the time complexity of original ABC is $O(I_{max}.(PN.f + PN.f + f)) = O(I_{max}.(PN.f))$.

5.3.4 Experimental Methodology

This section covers the methodology of our experiment. We leveraged our proposed methods to tune the main hyper-parameters of three ML models. As can be seen in figure 5.10 the overall process is done in several phases, including data pre-processing and feature engineering, generating a new population, training the SVM model, and ABC-based optimization steps. The OptABC algorithm proposed in this study is an improved version of recent work in 2021[ZMA21b].

Data Pre-Processing

Data pre-processing in the context of ML refers to the technique of transforming the original data into an appropriate and understandable state for ML algorithms. A summarized mechanism of the steps for data pre-processing is given below:

- **Data Cleansing:** In this step, the features with more than 60% percent missing values are eliminated from the dataset. The target population in this study are only the student majoring in computing fields. Therefore, the data is filtered to include only students from computing majors.
- **Feature Scaling:** Feature scaling is a technique to standardize the range of data, especially when there is a broad variation between values among different features to avoid biases from big outliers. We used *StandardScaler* from the Scikit-learn package was used for scaling the input values.
- **Feature Engineering:** In this step, one-hot encoding was utilized for encoding the categorical variables to binary representations.

Hyper-Parameter Tuning

Tuning hyper-parameters is a primary task in automated ML that helps the model achieve its best performance. Previous research reported on costly objective functions of ML models (specifically SVMs) regarding tuning time when using automated HPO methods [ZMA21b]. Therefore Tree-based SVM and (RF and XGBoost) algorithms are selected to be explored in this study. The proposed learning-based improvements in the OptABC, are an attempt to provide the algorithm with a richer population and strengthen the algorithm's exploration phase.

5.3.5 Experimental Results and Discussions

This section presents the performance metrics, and experimental results after applying the OptABC HPO algorithm.

Metric for Performance Evaluation

The experiment is performed in two different stages; training using 80:20 ratio train-test sets and cross-validation. Regarding the experiments using cross-validation, the dataset is partitioned into three folds to evaluate the ML algorithms. Different folds are assigned to the training and test sets. Specifically, for each run, $f = 1, 2, 3$, fold f is assigned to the test, and the rest of the folds are assigned to the training set. Cross-validation is employed to reduce the chances of overfitting. Although since it has an impact on tuning time, we leverage parallel cross-validation. In each run, accuracy and the average overall accuracy over three runs were reported. We also use execution time as another performance metric to measure the time regarding running time to return the optimal set of hyper-parameters.

In this experiment, the *Scikit-learn* libraries, along with other Python libraries, were used to leverage the ML models. All experiments were conducted using Python 3.8 on High-Performance Computational (HPC) resources.

Performance Comparison of Different Population Sizes

In this section, we compare the efficiency of OptABC with a previous method under different population sizes. The HyP-ABC [ZMA21b] uses the random search strategy, while OptABC employs two different learning algorithms in the initial and the Scout phase of the algorithm to enhance the convergence rate of the previous method. The results are shown in Tables 5.7, 5.8, and 5.9. The running times in hours stand for the time for all iterations until getting the desirable accuracy.

Table 5.7: Performance evaluation of Optimal ABC-based tuning (OptABC) and ABC-based Hyper-parameter tuning (HyP-ABC) to the RF classifier on the MID-FIELD dataset.

80:20 ratio Test-Train set						
	OptABC			HyP-ABC		
Population size	**20**	**50**	**100**	**20**	**50**	**100**
Accuracy(%)	88.71	88.75	88.76	88.71	88.74	88.78
Execution Time(hrs)	**1.6**	**3.17**	7.8	2.37	9.07	**3.18**
With 3-fold cross validation						
Accuracy(%)	88.79	88.77	88.77	87.57	88.74	88.77
Execution Time(hrs)	**0.46**	**1.09**	2.16	1.55*		

Table 5.8: Performance evaluation of Optimal ABC-based tuning (OptABC) and ABC-based Hyper-parameter tuning (HyP-ABC) to the XGBoost classifier on the MIDFIELD.

80:20 ratio Test-Train set						
	OptABC			HyP-ABC		
Population size	**20**	**50**	**100**	**20**	**50**	**100**
Accuracy(%)	88.65	88.73	88.86	88.60	88.70	88.75
Execution Time(hrs)	**1.98**	**3.85**	**3.42**	4.07	5.85	4.23
With 3-fold cross validation						
Accuracy(%)	88.29	88.64	88.72	87.97	88.64	88.84
Execution Time(hrs)	**4.83**	**19.54**	**17.6**	29.22*		

Table 5.9: Performance evaluation of Optimal ABC-based tuning (OptABC) and ABC-based Hyper-parameter tuning (HyP-ABC) to the SVM classifier on the MIDFIELD dataset.

80:20 ratio Test-Train set						
	OptABC			HyP-ABC		
Population size	**20**	**50**	**100**	**20**	**50**	**100**
Accuracy(%)	87.87	87.86	87.86	87.80	87.86	87.85
Execution Time(hrs)	**18.3**	**8.61**	**15.5**	23.95	53.05	62.57
With 3-fold cross validation						
Accuracy(%)	87.99	88.00	88.00	87.93	88.00	88.00
Execution Time(hrs)	**19.71**	39.01	40.10	34.15*		

As can be seen from the tables[3], in the majority of the cases, the execution time after applying the OptABC algorithm has improved without decreasing the classification accuracy. In summary, our proposed algorithm is quicker than HyP-ABC. This difference is more tangible when the population is small showing the capability of the algorithm in finding the optimal solution when the population is not too large.

[3]Asterik (*) indicates the best tuning time achieved from applying HyP-ABC [ZMA21b]

Diversifying the population helped to decrease the convergence rate with a chance of increasing accuracy. The advantage of our proposed method becomes notable when the population is small. We can defer that our approach is also applicable to other real-time applications. In summary, compared with the baseline, manual tuning methods, model-free and model-based HPO methods [ZLP+20, ZMR+21, ZMA21b] OptABC algorithm outperforms other methods in predicting student's success. OptABC uses an ABC-based algorithm to optimize the hyper-parameters of ML models, improving the classification accuracy effectually.

5.3.6 Conclusion

We presented OptABC, an artificial bee colony (ABC)-based optimization algorithm for hyper-parameter tuning for machine learning methods dealing with large datasets. Our proposed OptABC integrates the ABC algorithm, K-Means clustering, greedy algorithm, and opposition-based learning strategy for tuning the hyper-parameters of different machine learning models. We demonstrated OptABC in three experiments on SVM, RF, and XGBoost.

This hybrid method introduces a more intelligent approach to ABC and aimed to improve ABC's exploitation, exploration, and acceleration. The related experimental results on a real-world dataset compared with previous approaches on the same dataset demonstrate that the proposed OptABC exhibits faster convergence speed and running time without decreasing the accuracy in most cases and has an advantage over previous methods. In our future work, we plan to design strategies to further eliminate poor food sources in the initialization step to spend more of the evaluations on better solutions.

CHAPTER 6

A HYBRID MODEL FOR AUTOML FRAMEWORK

Abstract ABC-based hyper-parameter optimization and feature selection optimization algorithm, hereafter referred to as ABC-HFO2, is an open-source automated ML platform. This platform is developed to scale for large datasets through an automated feature selection process and tuning the hyper-parameters to guarantee sub-optimal performance for classical ML (we specifically focus on Random Forest, XGBoost, and SVM) algorithms. ABC-HFO2 is a fully automated platform for classical supervised algorithms. The result of the ABC-HFO2 AutoML is the set of sub-optimal features as well as a set of optimal hyper-parameters. The package can be easily exported for use in a production environment. The ABC-HFO2 AutoML algorithm relies on an evolutionary algorithm to train a larger number of configurations in regards to the number of features and number/range of hyper-parameters without training all the configurations. ABC-HFO2 AutoML uses a combination of Artificial Bee Colony (ABC) algorithm, agglomerative clustering algorithm and an opposition-based learning algorithm to yield results that are competitive with other frameworks relying on more complex techniques such as genetic algorithm or Bayesian optimization [HYLY19, VM21, VAQLJMM21]. ABC-HFO2 AutoML trains Random Forest, Support Vector Machine and Extreme Gradient Boosting algorithms.

Keywords: Hyperparameter Optimization, Feature Selection Optimization, Artificial Bee Colony Algorithm, Evolutionary Optimization

6.1 Foundations of ABC-HFO2

Python is an open-source and interpreted high-level general-purpose programming language. It is currently the fastest-growing programming language in the world and is being widely used. Scikit-learn is a widely used Python module integrating a large number of state-of-the-art ML algorithms for supervised and unsupervised problems ([PVG+11]). Although this package focuses on bringing ML to non-experts using a general-purpose high-level language, users do not usually know how to choose among the dozens of features or hyper-parameters to achieve good performance. There are even many ML models developed by experts that are all designed manually and by trial and error. In other words, even experts need the time and resources to create good predictive ML models ([HZC21]).

ABC-HFO2 addresses this problem by providing a unified ML framework and using the ABC algorithm to find a strong set of features and hyper-parameters for a given dataset. This framework is inspired by the OptABC and A2BCF algorithms proposed by ([ZMA21c, ZGMA22]). These algorithms use a modified version of the ABC approach to improve the running time for finding optimal configurations without significantly decreasing the performance. We show classification performance is often better than using standard selection optimization methods. Our approach will help non-expert users to be able to tune the hyper-parameters of their desired ML models appropriate to their applications, and therefore achieve improved performance.

The overall structure of the framework is presented in Figure 6.1. This framework is integrated to our previous work on a Hyperparameter Optimization (HPO) method proposed in 2021 [ZMA21c] and Feature Selection Optimization proposed in 2022 [ZGMA22].

The crux of ABC-HFO2 lies in its simplicity and requires no knowledge about the ML algorithms or their hyper-parameters unless users aim to change the ranges or add/remove hyper-parameters from the search space.

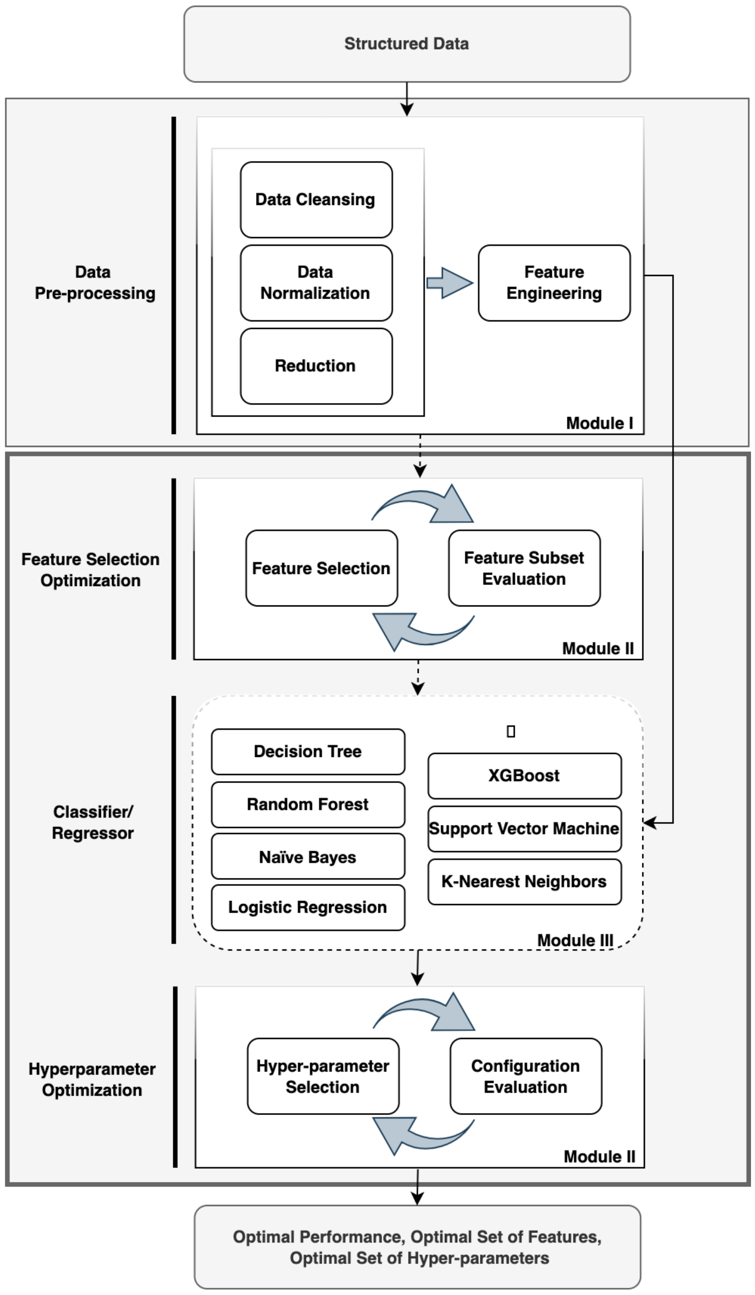


Figure 6.1: General schema of HFO2 framework

The framework also provides a stop accuracy parameter to accommodate impatient users to stop the algorithm when a predefined performance is achieved. The lower the stop accuracy, the less thoroughly the algorithm searches the space. However, we recommend a stop accuracy that takes at least several hours for production runs to guarantee sub-optimal results.

6.2 ABC-HFO2

The “ABCFSO” and “ABCHPO” algorithms can run as standalone algorithms for feature selection and hyper parameter optimization, respectively. However, the output of “ABCFSO” can also be used as the input of the “ABCHPO” algorithm so the HPO only runs with the sub-optimal selected features. Table 6.1 shows the integrated HFO2 ML algorithms along with their type of hyper-parameters and the number of each of them.

Table 6.1: Classifiers in HFO2 algorithm

Classifier	**Categorical**	**Numerical**	**Discrete Numerical**
Random Forest	1	0	5
XGBoost	0	3	2
Support Vector Machine	1	0	1

6.2.1 FSO

The ABCFSO algorithm in Python can be used as follows:

```
# Automated FSO
[1] from ABCHFS.ABCFSO import ABCFSO
[2] fso = ABCFSO(model="RF", cv=True, n_food=300, iterations=2)
[3] features = fso.fso(X,y)
```

To run the FSO algorithm, first the user needs to import the `ABCFSO` class from the `ABCFSO` module of the `ABCHFS` library (line `[1]` in the above code). Next, the user has to initiate the `ABCFSO` class with their desired parameters (line `[2]` in the above code). Lastly, to run the FSO algorithm, the user has to call the `fso` method by passing the features (`X` in the code) and the target value (`y` in the code) as input parameters (line `[3]` in the above code).

The ABCFSO class parameters description and the default values are described below:

- **model:** a string to determine the classifier, the algorithm accepts the following models: “XGB” for the XGBoost model, “SVM” for the Support Vector Machine, and “RF” for the Random Forest model, and the default value is set to ’RF’. The reason for focusing on two tree-based models in this framework is that these models have been shown to be more effective for large, structured, and partly discrete problems [EFH+13]. Also, previous research shows that SVM, RF, and XGBoost models are more complicated than other classical ML algorithms and require other strategies for HPO problems [ZMR+21]. However, other ML algorithms can be easily integrated into the framework.
- **cv:** a flag to determine whether the algorithm should use cross-validation or not, and the default value is set to True.
- **n_folds:** the number of folds for cross-validation where the default value is set to 3.
- **n_food:** the number of foods for the ABC algorithm where the default value is set to 500.
- **limit:** the threshold that determines the exhaustion of foods in the ABC algorithm where the default value is set to three.

- **iterations:** the number of iterations to run the ABC algorithm where the default value is set to five.
- **min_accuracy:** the threshold for removing poor quality food sources where the default value is set to 0.5.
- **stop_accuracy:** the threshold of the accuracy where the algorithm will stop, and where the default value is set to 0.9.
- **max_evaluation:** the maximum number of trainings in the algorithm, where the default value is set to 5000.

  The inputs of the FSO method are:
- **X:** pandas dataframe of features.
- **y:** pandas series of target variable.

The output of the FSO method is:

- **Features:** an array of binary values that determines the initially selected features.
- The final output of the FSO can be used as an input for the ABCHFO2 algorithm. As for the outputs, the algorithm also exhibits:
- **Global Optimum Accuracy:** the final accuracy of the algorithm.
- **Optimal Features:** the optimum selected features.
- **Duration:** the time it took the algorithm to find the optimum features.

Regarding the resource budgets, the user can use one of the below:

- **Number of evaluations:** the number of evaluations it took for the algorithm to find the optimum values.

- **Rounds:** the number of iterations it took for the algorithm to find the optimum values.

6.2.2 HPO

The ABCHPO algorithm can be used as follows:

```
# Automated HPO
[1] from ABCHFS.ABCHPO import ABCHPO
[2] hpo = ABCHPO(search=search ,model="RF", cv=True, n_food=500)
[3] res = hpo.hpo(X,y)
```

To run the HPO algorithm, first the user has to import the `ABCHPO` class from the `ABCHPO` module of the `ABCHFS` library (line `[1]` in the above code). Next, the user needs to initiate the `ABCHPO` class with their desired parameters (line `[2]` in the above code). Lastly, to run the HPO algorithm, the user has to call the `hpo` method by passing the independent variables (`X` in the code) and the target values (`y` in the code) as input parameters (line `[3]` in the above code).

The ABCHPO class parameters description and the default values are defined below. However, not all the parameters are mandatory and some of them are customizing the model.

- **search:** determines the hyper-parameters' search space of the ML model. For instance, for an RF model we could have:

```
search = {
  "n_estimators": {"type":"int", "range":[5, 500]},
  "max_features": {"type":"int", "range":[1,91]},
  "max_depth": {"type":"int", "range":[5,50]},
  "min_samples_split":{"type":"int", "range":[2,30]},
  "min_samples_leaf":{"type":"int", "range":[1,15]},
  "criterion": {"type":"ctg", "range":[0,1]}
      }
```

where the top-level keys determine the hyper-parameters to be optimized. The "type" defines the type of the hyper-parameter which can be "int" (i.e. integer), "ctg" (i.e. category) or "float" types. The "range" defines the range in which the algorithm should search for the optimized values. If the type is defined as categorical, the algorithm will generate dummy variables for each category using the one-hot encoding method.

- **model:** a string to determine the classifier, the algorithm accepts the following models: "XGB" for the XGBoost model, "SVM" for the support vector machine, and "RF" for the random forest model, where the default value is set to "RF"
- **cv:** a flag to determine whether the algorithm should use cross-validation or not where the default value is set to True.
- **n_folds:** the number of folds for cross-validation where the default value is set to three.
- **n_food:** the number of foods for the ABC algorithm where the default value is set to 500.
- **limit:** the threshold that determines the exhaustion of foods in the ABC algorithm where the default value is set to three.

- **iterations:** the number of iterations to run the ABC algorithm where the default value is set to two.
- **min_accuracy:** the threshold for removing poor quality food sources where the default value is set to 0.5.
- **stop_accuracy:** the threshold of the accuracy where the algorithm stops, and where the default value is set to 0.9.
- **max_evaluation:** the maximum number of trainings in the algorithm where the default value is set to 5000.
- **features:** an array of binary values that determines the sub-optimal features. This can be set to the output array of the "FSO" method from the ABCFSO algorithm. The default value is set to "None", meaning the algorithm will consider all the features while optimizing the hyper-parameters.

The inputs of the "HPO" method are:

- **X:** pandas dataframe of features,
- **y:** pandas series of target variable

The output of "HPO" method is the sub-optimal values of each hyper parameter. As outputs, the algorithm also exhibits:

- **Global Optimum Accuracy:** the final accuracy of the algorithm,
- **Optimal Parameters Value:** sub-optimal hyper-parameters values found by the algorithm.
- **Duration:** the time it took the algorithm to find the optimum values for the hyper-parameters.

- **Number of evaluations in HPO:** the number of evaluations it took for the algorithm to find the optimum hyper-parameters values
- **Rounds:** the number of iterations it took for the algorithm to find the optimum hyper-parameters values.

6.2.3 HPO-FSO Library (HFO2)

Below, a manual to describe how to use the HFO2 package is provided and gives a high-level overview for developers; Figure 6.2 also shows the output of an example run. The underlying algorithm for this framework is the Artificial Bee Colony algorithm, which is among evolutionary optimization algorithms and has proved to be beneficial when having different types of variables ([YS20])

```
# HFO2 AutoML
[1] from ABCHFS.ABCFSO import ABCFSO
[2] from ABCHFS.ABCHPO import ABCHPO
[3] fso = ABCFSO(model="RF", n_food=300, iterations=10)
[4] features = fso.fso(X,y)
[5] hpo = ABCHPO(search=search ,model="RF", n_food=500, features=features)
[6] res = hpo.hpo(X,y)
```

To run the HPO algorithm considering only selected features from the FSO, first user needs to import the `ABCSFO` and `ABCHPO` classes from the `ABCSFO` and `ABCHPO` modules of the `ABCHFS` library, respectively (lines `[1]` and `[2]` in the above code). Next, the user has to initiate the `ABCFSO` class with their desired parameters (line `[3]` in the above code). Then, the user will run the FSO algorithm by calling the `fso` method and passing the independent variables (`X` in the code) and the target values (`y` in the code) as input parameters (line `[4]` in the above code) and store the results in a variable (called `features` in the code) Next, the user has to initiate the `ABCHPO` class and pass the results of the `fso` method (`features` in the code) as its parameter (line `[5]` in the above code). Finally, to run the HPO algorithm, the user needs to call the `hpo` method by passing the independent variables (`X` in the code) and the target values (`y` in the code) as input parameters (line `[6]` in the above code).

A detailed algorithm for HFO2 can be found in Algorithm 12.

Algorithm 12 HFO2 Algorithm

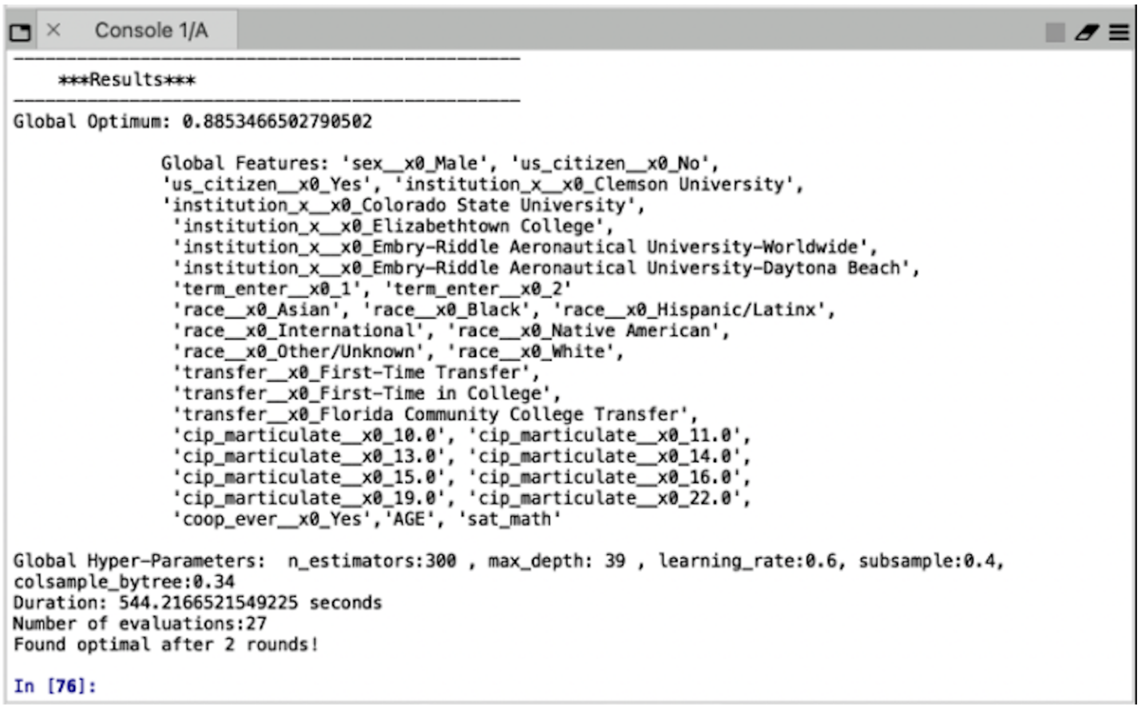
```
Output: Optimal features and hyper-parameters subset, performance
 1: Do clustering on the population
 2: Discard the food sources with poor quality from a new population
 3: while Stopping criteria is not met(number of evaluations ¡$max_{eval}$) do
 4:   for $i$ to $population\_size$ do
 5:     The employed bee regulates the bit vector and finds a new vector $N_i$ in the
        neighborhood. This is achieved by changing only one bit in the vector
 6:     Train the ML model with the novel vector (subset of features selected)
 7:     if the new vector has a better quality then
 8:       Replace the new vector with the original vector
 9:     else
          $trial_i$ +=1
10:     end if
11:   end for
12:   for $j$ to $population\_size$ do
13:     The onlooker bee calculates the exploration probability ($P_i$)
14:     if $rand(0,1) > P_i$ then
15:       The onlooker bee regulates the current vector $N_i$ by changing only one of
          the bits in the vector, which gives a new vector in the neighborhood $M_i$
16:       Train the ML model with the updated features subset
17:       if the new vector has a better quality then
18:         Replace the new vector with the original vector
19:       else
            $trial_i$ +=1
20:       end if
21:     else
          The onlooker disregards the features subset and moves to the next bit
          vector
22:     end if
23:   end for
24:   Memorize and update the best subset so far
25:   if $trail_i > limit$ (food source is exhausted) then
26:     Scout bee generates a "Random" bit vector
27:     Scout bee generates the "OBL" bit vector
28:     Select the subset that gives better fitness between Random and OBL vectors
29:   end if
30: end while
31: Return Optimal subset of features and pass to the HPO algorithm
32: Generate HPO configurations and run clustering on the new population
```

```
1: while Stopping criteria is not met do
2:     for $i$ to $population_{new\_}size$ do
3:         Employed bee generates a new food $N_i$ source in neighborhood and modifies
           the new food source to an accepted food source $M_i$
4:         Train the ML model with the updated food source
5:     end for
6:     for $j$ to $population_{new\_}size$ do
7:         The Onlooker bee calculates the probability $(P_i)$ of each solution
8:         if $(rand(0,1)) > (P_i)$ then
9:             The Onlooker bee generates a new food source in neighborhood $N_i$ and
               then modifies the new food source to an accepted food source $M_i$
10:            Train the ML model with the updated food source
11:        else
               The Onlooker bee disregards the food source and moves to the next food
               source
12:        end if
13:    end for
14:    Memorize and update the best food source
15:    if trial >limit then
16:        Scout bee generates a random food source based
17:        Scout bee generates an OBL food source
18:        Select the food source with better quality
19:    end if
20: end while
21: Return Optimal features, hyper-parameters, and performance[0]
```

6.3 Experimental Evaluation and Simulation Results

ABC-HFO2 is the first method to use ABC to automatically instantiate parametric classical ML framework at the push of a button. Previously, the FSO component of the library was implemented by the authors [ZGMA22], and the HPO component was also implemented by [ZMA21c]. The proposed library in this study is the product of these two components as an AutoML product.

[0]Performance refers to the performance metric selected by the user, such as accuracy in classification or mean squared error in regression problems.



```
------------------------------------------
    ***Results***
------------------------------------------
Global Optimum: 0.8853466502790502

               Global Features: 'sex__x0_Male', 'us_citizen__x0_No',
               'us_citizen__x0_Yes', 'institution_x__x0_Clemson University',
               'institution_x__x0_Colorado State University',
                'institution_x__x0_Elizabethtown College',
                'institution_x__x0_Embry-Riddle Aeronautical University-Worldwide',
                'institution_x__x0_Embry-Riddle Aeronautical University-Daytona Beach',
                'term_enter__x0_1', 'term_enter__x0_2'
                'race__x0_Asian', 'race__x0_Black', 'race__x0_Hispanic/Latinx',
                'race__x0_International', 'race__x0_Native American',
                'race__x0_Other/Unknown', 'race__x0_White',
                'transfer__x0_First-Time Transfer',
                'transfer__x0_First-Time in College',
                'transfer__x0_Florida Community College Transfer',
                'cip_marticulate__x0_10.0', 'cip_marticulate__x0_11.0',
                'cip_marticulate__x0_13.0', 'cip_marticulate__x0_14.0',
                'cip_marticulate__x0_15.0', 'cip_marticulate__x0_16.0',
                'cip_marticulate__x0_19.0', 'cip_marticulate__x0_22.0',
                'coop_ever__x0_Yes','AGE', 'sat_math'

Global Hyper-Parameters:  n_estimators:300 , max_depth: 39 , learning_rate:0.6, subsample:0.4,
colsample_bytree:0.34
Duration: 544.2166521549225 seconds
Number of evaluations:27
Found optimal after 2 rounds!

In [76]:
```

Figure 6.2: Example run of the framework

Using this package, it is possible to optimize hyper-parameters and features simultaneously or independently. Although it is possible that users go with the default parameters of the package, they can also modify the parameters with their desirable values.

The preliminary results of our framework are shown in Table 6.2 and 5.6[1]. The results show the effectiveness of the proposed algorithm on the presented datasets and are promising. However, in order to optimize the hyper-parameters of ML models, other HPO methods can be applied and each of these methods has its own advantages and disadvantages. Currently, there are no unique optimization methods that outperform all the existing methods due to the nature of the problem and the type of hyper-parameters. Hence, this topic can be extended more comprehensively.

[1]The PJM dataset is only evaluated on the HPO component of the HFO2 framework as it has few number of features.

Table 6.2: Performance on 3-fold cross-validation utilizing Hyper-parameter and Feature Optimization (HFO2) to the ML models

Dataset	RF	XGBoost	SVM/SVR
Classification (Accuracy)			
MIDFIELD	88.56	**88.89**	88.18
MNIST	97.38	97.38	**99.22**
Regression (RMSE)			
Boston Housing	**3.30**	3.33	4.24
PJM	**170.458**	278.394	334.04

The results just shown indicate that ABC-HFO2 is effective at optimizing the objective function. Although the use of cross-validation increases the tuning time compared with using a single train and test set, it also substantially increases the robustness of the algorithm against overfitting.

As can be seen from the results this is not sufficient to allow to conclude which classifier/regressor clearly dominated the others and depending on the application the selected model would be different.

Figure 6.3 shows a plot of accuracy over all the evaluations on the MNIST dataset. Also, Figure 6.4 shows how the population of food sources improves over 20 iterations on the MNIST dataset. Each box shows the overall quality of the population as well as the quartiles. In this example, the best accuracy at the end of each iteration is the upper whisker value of each box. As can be seen, the interquartile range (IQR), which assesses the variability where most of the values lie, is increasing in each iteration. In another word, the central portion of the population's quality is improving and is a fair measure of the spread as it is not affected by the outliers (found food sources found by scout bees).

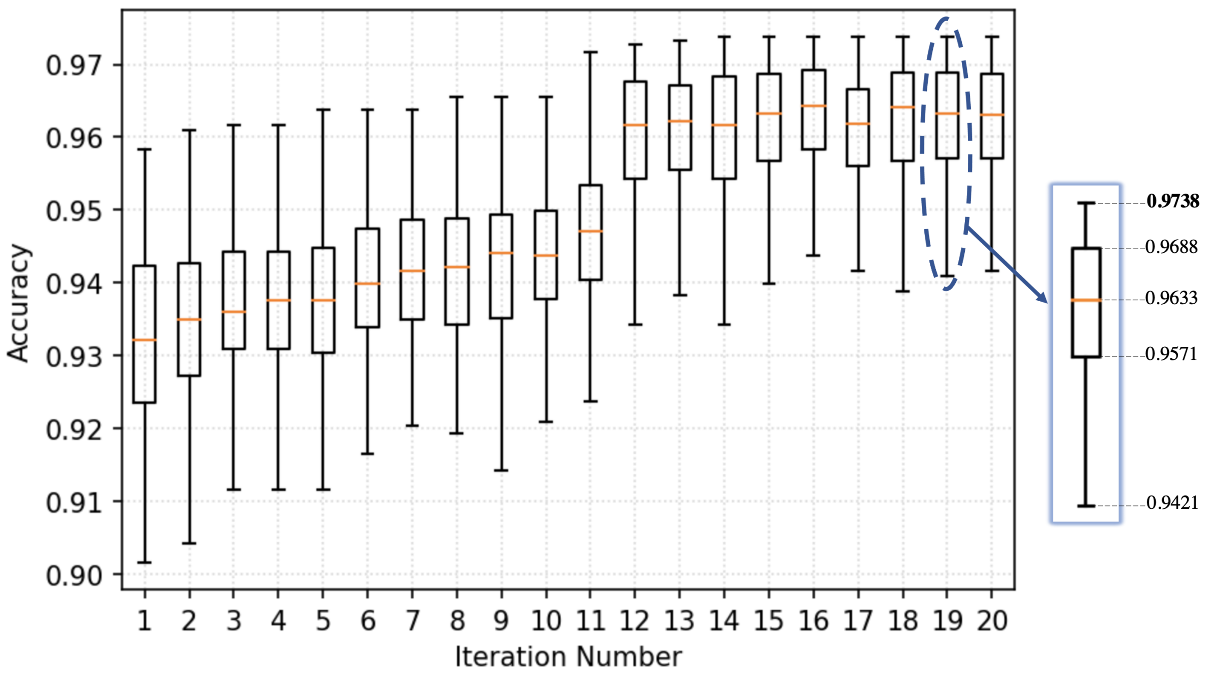


Figure 6.3: Accuracy over all the evaluations on the MNIST dataset.

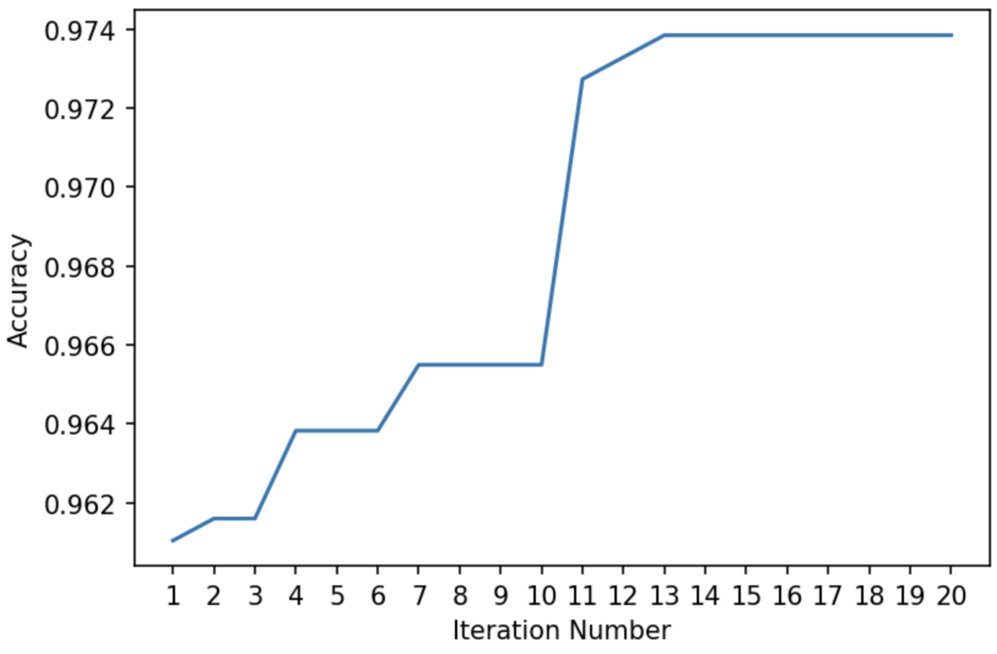


Figure 6.4: Classification accuracy on the MNIST dataset at the end of each iteration using HFO2 algorithm

6.4 Remarks

In this chapter, we covered the HFO2 python package developed for automated HPO and FSO. This framework takes advantage of two modified ABC algorithms, tuning the hyper-parameters of three classical ML algorithms and finding the datasets' sub-optimal subset of features. We have shown that the challenge of hyper-parameter optimization and feature selection optimization can be solved by an automated and practical tool. This is made possible by a modified version of the Artificial Bee Colony algorithm (ABC) that iteratively builds models of the hyper-parameter/features space and leverages these models to discover new points in the search space that are potential for investigation. The modified algorithm is an effort to partly address the disadvantage of the ABC algorithm (slow convergence) by limiting the number of training on the configurations and training the model on only representatives of configuration classes. Further, we leveraged an opposition-based step to the scout phase of the ABC algorithm to balance the exploration and exploitation phases of the ABC algorithm that the original algorithm lacks.

One of the main advantages of the proposed algorithm is that it offers fewer hyper-parameters in comparison to other evolutionary algorithms such as genetic algorithms, which avoids confusing the user by tuning the new algorithms' hyper-parameters.

In this work, we built a tool, ABC-HFO2, for three common ML algorithms, namely RF, XGBoost, and SVM in Python and makes it easy for non-experts to build well-performed and quality model application scenarios in hand. An empirical comparison on 4 datasets showed that ABC-HFO2 often outperformed standard methods. We have also written a freely downloadable Python library to make HFO2 is easy for end-users to access. We see several promising avenues for future work.

In the next chapter, we conclude the work in this dissertation and provide insights for future work.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusions

In this dissertation, an AutoML framework is proposed for classical ML algorithms. It includes two main components as follows: (1) automatic hyper-parameter tuning and (2) automatic feature selection. These components are integrated using evolutionary optimization algorithms and as a coherent entity to provide new solutions for existing challenges in the AutoML field. Each component is summarized as follows:

- A novel automated hyper-parameter tuning approach is proposed based on the ABC algorithm to overcome the difficulties of tuning large-scale search space and real-world data. Specifically, they are designed to tune the hyper-parameters of the ML algorithms with wider ranges or more hyper-parameters where the different configurations of hyper-parameters create a large search space. A modified ABC, clustering and OBL algorithms are leveraged to bridge the existing gaps and automatically tune the hyper-parameters. These techniques are combined to solve four important applications. While such algorithms improve the performance of classical ML algorithms for the specific applications that we evaluated in this dissertation, there is not a theoretical guarantee for convergence of the evolutionary optimization algorithm to the globally-optimal solution, i.e., the results of hyper-parameter tuning are sub-optimal.
- A novel feature selection algorithm is proposed to overcome the challenges in filter or wrapper selection methods. This algorithm is a combination of ABC, clustering, and OBL methods. It also takes advantage of filtering the

poor configurations in the early stages of the algorithm to improve the convergence rate of the algorithm. While such algorithms improve the performance of classical ML algorithms for the specific applications that we evaluated in this dissertation, there is not a theoretical guarantee for convergence of the evolutionary optimization algorithm to the globally-optimal solution, i.e., the results of feature selection are sub-optimal.

7.2 Future Work

Previous chapters showcased the effectiveness and efficiency of the proposed framework (depicted in Figure 6.1) for classical ML. However, there are still several challenges that need to be considered in future work, as explained below.

7.2.1 Automated pre-processing step

In this dissertation, the proposed framework mainly focused on HPO and FSO, which are two main components of the AutoML. However, a complete AutoML pipeline can also include pre-processing step as well. This is the limitation of the proposed HFO2 algorithm. Future work should focus on designing effective pre-processing steps. In other words, instead of cleaning the data and then feeding it to the framework, the user should be able to input the raw data and expect the software to automatically clean the dataset and subsequently do the FSO and HPO steps.

In addition to the main hyper-parameters of the ML algorithms, there exist other hyper-parameters with less impact, which can still have an effect on the performance of the model and can be automated. However, more advanced techniques are required to select the best hyper-parameters that can both enhance the performance and save training time.

7.2.2 Exploring deep learning algorithms

This dissertation mainly focuses on classical ML algorithms. Although integrating deep learning models to this framework can be implemented, using more advanced techniques for these algorithms that are time-consuming in nature can be explored as future work [EMH19]. Prior works on this topic have explored using evolutionary algorithms for convolutional neural networks [XY17].

7.2.3 Other Future Work

Other Modalities

The proposed AutoML framework utilizes different data modalities to improve classification/regression performance. The current framework is tested on structured datasets. In the future, this framework can be extended to leverage other data modalities such as non-structured data. It can also be utilized for not only classification or regression tasks in supervised algorithms but also for other tasks such as clustering or segmentation.

Unsupervised Learning

This dissertation utilizes various supervised learning techniques such as SVM, RF, and XGBoost for building models. However, unsupervised ML algorithms are also suggested to be explored and can be leveraged in our framework in the future.

Big Data and Distributed Systems

This dissertation's focus is mainly on improving algorithms for efficient HPO and FSO. In the future, advanced big data analytic techniques, such as distributed systems and cloud computing, will be used in this framework to further reduce computational costs and enhance the speed of the training process. For example, data parallelism and model parallelism techniques can be used to train ML models in a distributed system.

Statistical Analysis

Last but not least, in future work, researchers can explore the design of experimental techniques [EJKW$^+$00, Ant14] to find statistical performance guarantees for the HPO, FSO, and HPO+FSO algorithms that are developed in this dissertation for classical and non-classical ML models. This will not only provide another practical dimension to our proposed HPO+FSO solutions but also will provide future users with a certain level of assurance while using the recommended techniques.

[A$^+$20] SK Agrawal et al. Modified gbest-guided abc algorithm approach applied to various nonlinear problems. In *Optical and Wireless Technologies*, pages 615–623. Springer, 2020.

[AAA$^+$21] Esther Omolara Abiodun, Abdulatif Alabdulatif, Oludare Isaac Abiodun, Moatsum Alawida, Abdullah Alabdulatif, and Rami S Alkhawaldeh. A systematic review of emerging feature selection optimization methods for optimal text classification: the present state and prospective opportunities. *Neural Computing and Applications*, 33(22):15091–15118, 2021.

[ABHAAD21] Mohammed Azmi Al-Betar, Abdelaziz I Hammouri, Mohammed A Awadallah, and Iyad Abu Doush. Binary beta b-hill climbing optimizer with s-shape transfer function for feature selection. *Journal of Ambient Intelligence and Humanized Computing*, 12(7):7637–7665, 2021.

[Abr19] Steven Abreu. Automated architecture design for deep neural networks. *arXiv preprint arXiv:1908.10714*, 2019.

[ADNDS$^+$19] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Claudio Pomo, and Azzurra Ragone. On the discriminative power of hyper-parameters in cross-validation and how to choose them. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 447–451, 2019.

[AGAB09] Mehdi Hosseinzadeh Aghdam, Nasser Ghasem-Aghaee, and Mohammad Ehsan Basiri. Text feature selection using ant colony optimization. *Expert systems with applications*, 36(3):6843–6853, 2009.

[And19] Răzvan Andonie. Hyperparameter optimization in learning systems. *Journal of Membrane Computing*, 1(4):279–291, 2019.

[Ant14] Jiju Antony. *Design of experiments for engineers and scientists.* Elsevier, 2014.

[AO19] Sibel Arslan and Celal Ozturk. Multi hive artificial bee colony programming for high dimensional symbolic regression with feature selection. *Applied Soft Computing*, 78:515–527, 2019.

[AYA09] Eslam F Ahmed, Wang J Yang, and Maan Y Abdullah. Novel method of the combination of forecasts based on rough sets. *Journal of Computer Science*, 5(6):440, 2009.

[AZ20] Menad Nait Amar and Noureddine Zeraibi. Application of hybrid support vector regression artificial bee colony for prediction of mmp in co2-eor process. *Petroleum*, 6(4):415–422, 2020.

[BA17] Soudeh Babaeizadeh and Rohanin Ahmad. Enhanced constrained artificial bee colony algorithm for optimization problems. *Int. Arab J. Inf. Technol.*, 14(2):246–253, 2017.

[BB12] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[BBBK11] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *25th annual conference on neural information processing systems (NIPS 2011)*, volume 24. Neural Information Processing Systems Foundation, 2011.

[BBCY18] Hasan Badem, Alper Basturk, Abdullah Caliskan, and Mehmet Emin Yuksel. A new hybrid optimization method combining artificial bee colony and limited-memory bfgs algorithms for efficient numerical optimization. *Applied Soft Computing*, 70:826–844, 2018.

[BD15] Haider Banka and Suresh Dara. A hamming distance based binary particle swarm optimization (hdbpso) algorithm for high dimensional feature selection, classification and validation. *Pattern Recognition Letters*, 52:94–100, 2015.

[Ben00] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

[BGCL00] AF Bowers, C Giraud-Carrier, and JW Lloyd. A unifying view of knowledge representation for inductive learning. *Submitted for publication*, 2000.

[BHM77] Stephen P Bradley, Arnoldo C Hax, and Thomas L Magnanti. *Applied mathematical programming.* Addison-Wesley, 1977.

[Bie03] Alain Biem. A model selection criterion for classification: Application to hmm topology optimization. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 104–108. IEEE, 2003.

[BM19] Janani Balakumar and S Vijayarani Mohan. Artificial bee colony algorithm for feature selection and improved support vector machine for text classification. *Information Discovery and Delivery*, 2019.

[BMDC21] Shreya Bhattacharyya, Souvik Majumder, Papiya Debnath, and Manash Chanda. Arrhythmic heartbeat classification using ensemble of random forest and support vector machine algorithm. *IEEE Transactions on Artificial Intelligence*, 2(3):260–268, 2021.

[BMTB20] Martin Binder, Julia Moosbauer, Janek Thomas, and Bernd Bischl. Multi-objective hyperparameter tuning and feature selection using filter ensembles. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 471–479, 2020.

[BPH06] Kristin P Bennett and Emilio Parrado-Hernández. The interplay of optimization and machine learning research. *The Journal of Machine Learning Research*, 7:1265–1281, 2006.

[BPWS$^+$15] Mairead L Bermingham, Ricardo Pong-Wong, Athina Spiliopoulou, Caroline Hayward, Igor Rudan, Harry Campbell, Alan F Wright, James F Wilson, Felix Agakov, Pau Navarro, et al. Application of high-dimensional feature selection: evaluation for genomic prediction in man. *Scientific reports*, 5(1):1–12, 2015.

[BR03] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[Bub14] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *arXiv preprint arXiv:1405.4980*, 2014.

[BV19] Shahab Boumi and Adan Vela. Application of hidden markov models to quantify the impact of enrollment patterns on student performance. *International Educational Data Mining Society*, 2019.

[CDM15] Marc Claesen and Bart De Moor. Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*, 2015.

[Cha98] Marie Chavent. A monothetic clustering method. *Pattern Recognition Letters*, 19(11):989–996, 1998.

[CHB$^+$15] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4), 2015.

[CKH$^+$18] Tianqing Chang, Depeng Kong, Na Hao, Kehu Xu, and Guozhen Yang. Solving the dynamic weapon target assignment problem by an improved artificial bee colony algorithm with heuristic factor initialization. *Applied Soft Computing*, 70:845–863, 2018.

[CLWY18] Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79, 2018.

[CNM06] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168, 2006.

[CS21] Abhilasha Chaudhuri and Tirath Prasad Sahu. Feature selection using binary crow search algorithm with time varying flight length. *Expert Systems with Applications*, 168:114288, 2021.

[CSP$^+$14] Marc Claesen, Jaak Simm, Dusan Popovic, Yves Moreau, and Bart De Moor. Easy hyperparameter search using optunity. *arXiv preprint arXiv:1412.1114*, 2014.

[CVSZ20] Soohyun Cho, Miklos A Vasarhelyi, Ting Sun, and Chanyuan Zhang. Learning from machine learning in accounting and assurance, 2020.

[CWL06] Geng Cui, Man Leung Wong, and Hon-Kwong Lui. Machine learning for direct marketing response models: Bayesian networks with evolutionary programming. *Management Science*, 52(4):597–612, 2006.

[CZZ09] Dong-Xia Chang, Xian-Da Zhang, and Chang-Wen Zheng. A genetic algorithm with gene rearrangement for k-means clustering. *Pattern Recognition*, 42(7):1210–1222, 2009.

[DBS06] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[DFNNS17] Gonzalo I Diaz, Achille Fokoue-Nkoutche, Giacomo Nannicini, and Horst Samulowitz. An effective algorithm for hyperparameter optimization of neural networks. *IBM Journal of Research and Development*, 61(4/5):9–1, 2017.

[DGMCEGC19] Noemí DeCastro-García, Ángel Luis Muñoz Castañeda, David Escudero García, and Miguel V Carriegos. Effect of the sampling of a dataset in the hyperparameter optimization phase over the efficiency of a machine learning algorithm. *Complexity*, pages 1–16, 2019.

[DKA17] Rafet Durgut, Hakan Kutucu, and Sedat Akleylek. An artificial bee colony algorithm for solving the weapon target assignment problem. In *Proceedings of the 7th International Conference on Information Communication and Management*, pages 28–31, 2017.

[DL97] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(1-4):131–156, 1997.

[DPW01] Ding-Zhu Du, Panos M Pardalos, and Weili Wu. Optimization problems. In *Mathematical Theory of Optimization*, pages 1–21. Springer, 2001.

[DSC19] Tansel Dokeroglu, Ender Sevinc, and Ahmet Cosar. Artificial bee colony optimization for the quadratic assignment problem. *Applied soft computing*, 76:595–606, 2019.

[DSdNDM$^+$13] Adrián Domínguez, Joseba Saenz-de Navarrete, Luis De-Marcos, Luis Fernández-Sanz, Carmen Pagés, and José-Javier Martínez-Herráiz. Gamifying learning experiences: Practical implications and outcomes. *Computers & education*, 63:380–392, 2013.

[EA10] Mohammed El-Abd. A cooperative approach to the artificial bee colony algorithm. In *IEEE congress on evolutionary computation*, pages 1–5. IEEE, 2010.

[EFH$^+$13] Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, and Kevin Leyton-Brown. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, volume 10, page 3, 2013.

[EJKW$^+$00] L Eriksson, E Johansson, N Kettaneh-Wold, C Wikström, and S Wold. Design of experiments. *Principles and Applications, Learn ways AB, Stockholm*, 2000.

[EMH19] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[EMS19] Radwa Elshawi, Mohamed Maher, and Sherif Sakr. Automated machine learning: State-of-the-art and open challenges. *arXiv preprint arXiv:1906.02287*, 2019.

[ENM15] Issam El Naqa and Martin J Murphy. What is machine learning? In *machine learning in radiation oncology*, pages 3–11. Springer, 2015.

[FEF$^+$18] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Practical automated machine learning for the automl challenge 2018. In *International Workshop on Automatic Machine Learning at ICML*, pages 1189–1232, 2018.

[FH19] Matthias Feurer and Frank Hutter. Hyperparameter optimization. In *Automated Machine Learning*, pages 3–33. Springer, Cham, 2019.

[FHV$^+$19] Eduardo Fernandes, Maristela Holanda, Marcio Victorino, Vinicius Borges, Rommel Carvalho, and Gustavo Van Erven. Educational data mining: Predictive analysis of academic performance of public school students in the capital of brazil. *Journal of Business Research*, 94:335–343, 2019.

[FKH18] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1437–1446. PMLR, 2018.

[FKPoS19] Suthida Fairee, Charoenchai Khompatraporn, Santitham Prom-on, and Booncharoen Sirinaovakul. Combinatorial artificial bee colony optimization with reinforcement learning updating for travelling salesman problem. In *2019 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 93–96. IEEE, 2019.

[FR07] KM Faraoun and A Rabhi. Data dimensionality reduction based on genetic selection of feature subsets. *INFOCOMP Journal of Computer Science*, 6(3):36–46, 2007.

[GBC+15] Isabelle Guyon, Kristin Bennett, Gavin Cawley, Hugo Jair Escalante, Sergio Escalera, Tin Kam Ho, Núria Macia, Bisakha Ray, Mehreen Saeed, Alexander Statnikov, et al. Design of the 2015 chalearn automl challenge. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[GC20] Pallavi Grover and Sonal Chawla. Text feature space optimization using artificial bee colony. In *Soft computing for problem solving*, pages 691–703. Springer, 2020.

[GE03] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[GFB+11] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2011.

[GG05] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European conference on information retrieval*, pages 345–359. Springer, 2005.

[GG19] Korhan Günel and I Gör. A modification of artificial bee colony algorithm for solving initial value problems. *TWMS Journal of Applied and Engineering Mathematics*, 9(4):810–821, 2019.

[GJM18] Selvaraj Gunasundari, S Janakiraman, and S Meenambal. Multiswarm heterogeneous binary pso using win-win approach for improved feature selection in liver and kidney disease diagnosis. *Computerized Medical Imaging and Graphics*, 70:135–154, 2018.

[GLH15] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*, volume 72. Springer, 2015.

[GMR$^+$20] Nicolas Georges, Islem Mhiri, Islem Rekik, Alzheimer's Disease Neuroimaging Initiative, et al. Identifying the best data-driven feature selection method for boosting reproducibility in classification tasks. *Pattern Recognition*, 101:107183, 2020.

[GSPK18] Hao Gao, Yujiao Shi, Chi-Man Pun, and Sam Kwong. An improved artificial bee colony algorithm with its application. *IEEE Transactions on Industrial Informatics*, 15(4):1853–1865, 2018.

[GT13] Anupriya Gogna and Akash Tayal. Metaheuristics: review and application. *Journal of Experimental & Theoretical Artificial Intelligence*, 25(4):503–526, 2013.

[GYR$^+$18] Yolanda Gil, Ke-Thia Yao, Varun Ratnakar, Daniel Garijo, Greg Ver Steeg, Pedro Szekely, Rob Brekelmans, Mayank Kejriwal, Fanghao Luo, and I-Hui Huang. P4ml: A phased performance-based pipeline planner for automated machine learning. In *AutoML Workshop at ICML*, 2018.

[GYW$^+$08] XC Guo, JH Yang, CG Wu, CY Wang, and YC Liang. A novel ls-svms hyper-parameter selection based on particle swarm optimization. *Neurocomputing*, 71(16-18):3211–3215, 2008.

[HCB$^+$14] F Hutter, R Caruana, R Bardenet, M Bilenko, I Guyon, B Kegl, and H Larochelle. Automl 2014@ icml. In *AutoML 2014 Workshop@ ICML*, 2014.

[HHLB11] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In

International conference on learning and intelligent optimization, pages 507–523. Springer, 2011.

[HJR78] David Harrison Jr and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1):81–102, 1978.

[HKV19] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges.* Springer Nature, 2019.

[HSW$^+$19] Daniel Sik Wai Ho, William Schierding, Melissa Wake, Richard Saffery, and Justin O'Sullivan. Machine learning snp based prediction for precision medicine. *Frontiers in genetics*, 10:267, 2019.

[HXKZ15] Emrah Hancer, Bing Xue, Dervis Karaboga, and Mengjie Zhang. A binary abc algorithm based on advanced similarity scheme for feature selection. *Applied Soft Computing*, 36:334–348, 2015.

[HXZ$^+$18] Emrah Hancer, Bing Xue, Mengjie Zhang, Dervis Karaboga, and Bahriye Akay. Pareto front feature selection based on artificial bee colony optimization. *Information Sciences*, 422:462–479, 2018.

[HYLY19] Changwu Huang, Bo Yuan, Yuanxiang Li, and Xin Yao. Automatic parameter tuning using bayesian optimization method. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 2090–2097. IEEE, 2019.

[HZC21] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[IDS20] Yasmine Mahmoud Ibrahim, Saad Darwish, and Walaa Sheta. Brain tumor segmentation in 3d-mri based on artificial bee colony and level set. In *Joint European-US Workshop on Applications of Invariance in Computer Vision*, pages 193–202. Springer, 2020.

[IMNS20] MohammadNoor Injadat, Abdallah Moubayed, Ali Bou Nassif, and Abdallah Shami. Multi-split optimized bagging ensemble model selection for multi-class educational data mining. *Applied Intelligence*, 50(12):4506–4528, 2020.

[JJJ18] Indu Jain, Vinod Kumar Jain, and Renu Jain. Correlation feature selection based improved-binary particle swarm optimization for gene selection and cancer classification. *Applied Soft Computing*, 62:203–215, 2018.

[JM15] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[KA09] Dervis Karaboga and Bahriye Akay. A comparative study of artificial bee colony algorithm. *Applied mathematics and computation*, 214(1):108–132, 2009.

[KAK$^+$05] Mauricio Kugler, Kazuma Aoki, Susumu Kuroyanagi, Akira Iwata, and Anto Satriyo Nugroho. Feature subset selection for support vector machines using confident margin. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 907–912. IEEE, 2005.

[KAK20] Dervis Karaboga, Bahriye Akay, and Nurhan Karaboga. A survey on the studies employing machine learning (ml) for enhancing artificial bee colony (abc) optimization algorithm. *Cogent Engineering*, 7(1):1855741, 2020.

[KAO07] Dervis Karaboga, Bahriye Akay, and Celal Ozturk. Artificial bee colony (abc) optimization algorithm for training feed-forward neural networks. In *International conference on modeling decisions for artificial intelligence*, pages 318–329. Springer, 2007.

[Kar05] Dervis Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[KB07] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459–471, 2007.

[KD16] Vaishali R Kulkarni and Veena Desai. Abc and pso: A comparative analysis. In *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pages 1–7. IEEE, 2016.

[KE95] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[KGOK14] Dervis Karaboga, Beyza Gorkemli, Celal Ozturk, and Nurhan Karaboga. A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review*, 42(1):21–57, 2014.

[Kim19] Meejoung Kim. Supervised learning-based ddos attacks detection: Tuning hyperparameters. *ETRI Journal*, 41(5):560–573, 2019.

[KJ$^+$13] Max Kuhn, Kjell Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.

[KJ20] Rabiya Khalid and Nadeem Javaid. A survey on hyperparameters optimization algorithms of forecasting models in smart grid. *Sustainable Cities and Society*, 61:102275, 2020.

[KLL13] Fei Kang, Junjie Li, and Haojin Li. Artificial bee colony algorithm and pattern search hybridized for global optimization. *Applied Soft Computing*, 13(4):1781–1791, 2013.

[KM14] Vipin Kumar and Sonajharia Minz. Feature selection: a literature review. *SmartCR*, 4(3):211–229, 2014.

[KPZS20] Jonathan Krauß, Bruno Machado Pacheco, Hanno Maximilian Zang, and Robert Heinrich Schmitt. Automated machine learning for predictive quality in production. *Procedia CIRP*, 93:443–448, 2020.

[Kra16] Oliver Kramer. Scikit-learn. In *Machine Learning for Evolution Strategies*, pages 45–53. Springer, 2016.

[KS96] Daphne Koller and Mehran Sahami. Toward optimal feature selection. Technical report, Stanford InfoLab, 1996.

[KS00] Mineichi Kudo and Jack Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern recognition*, 33(1):25–41, 2000.

[KTH$^+$19] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka: Automatic model selection and hyperparameter optimization in weka. In *Automated Machine Learning*, pages 81–95. Springer, Cham, 2019.

[KZW$^+$17] Chayakrit Krittanawong, HongJu Zhang, Zhen Wang, Mehmet Aydar, and Takeshi Kitai. Artificial intelligence in precision cardiovascular medicine. *Journal of the American College of Cardiology*, 69(21):2657–2664, 2017.

[LAS$^+$15] Himabindu Lakkaraju, Everaldo Aguiar, Carl Shan, David Miller, Nasir Bhanpuri, Rayid Ghani, and Kecia L Addison. A machine learning framework to identify students at risk of adverse academic outcomes. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1909–1918, 2015.

[LeC98] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[LGN$^+$09] Ioanna Lykourentzou, Ioannis Giannoukos, Vassilis Nikolopoulos, George Mpardis, and Vassili Loumos. Dropout prediction in e-learning courses through the combination of machine learning techniques. *Computers & Education*, 53(3):950–965, 2009.

[Li02] Xiao-lei Li. An optimizing method based on autonomous animats: fish-swarm algorithm. *Systems Engineering-Theory & Practice*, 22(11):32–38, 2002.

[LLL$^+$19] Peizhong Liu, Xiaofang Liu, Yanming Luo, Yongzhao Du, Yulin Fan, and Hsuan-Ming Feng. An enhanced exploitation artificial bee colony algorithm in automatic functional approximations. *Intelligent Automation and Soft Computing*, 25(2):385–394, 2019.

[LNK$^+$17] Pablo Ribalta Lorenzo, Jakub Nalepa, Michal Kawulok, Luciano Sanchez Ramos, and José Ranilla Pastor. Particle swarm optimization for hyper-parameter selection in deep neural networks. In *Proceedings of the genetic and evolutionary computation conference*, pages 481–488, 2017.

[LOO$^+$22] Susan M Lord, Matthew W Ohland, Marisa K Orr, Richard A Layton, Russell A Long, Catherine E Brawner, Hossein Ebrahiminejad, Baker A Martin, George D Ricco, and Leila Zahedi. Midfield: A resource for longitudinal student record research. *IEEE Transactions on Education*, 2022.

[Lou14] Gilles Louppe. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*, 2014.

[LSC05] Stefan Lessmann, Robert Stahlbock, and Sven F Crone. Optimizing hyperparameters of support vector machines by genetic algorithms. In *IC-AI*, pages 74–82, 2005.

[LTS$^+$17] Hoang Thanh Lam, Johann-Michael Thiebaut, Mathieu Sinn, Bei Chen, Tiep Mai, and Oznur Alkan. One button machine for automating feature engineering in relational databases. *arXiv preprint arXiv:1706.00327*, 2017.

[Luo16a] Gang Luo. Predict-ml: a tool for automating machine learning model building with big clinical data. *Health Information Science and Systems*, 4(1):1–16, 2016.

[Luo16b] Gang Luo. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 5(1):1–16, 2016.

[LW$^+$02] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[LY05] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on knowledge and data engineering*, 17(4):491–502, 2005.

[M$^+$67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[MA14] F Ghareh Mohammadi and M Saniee Abadeh. Image steganalysis using a bee colony based feature selection algorithm. *Engineering Applications of Artificial Intelligence*, 31:35–43, 2014.

[MAA19] Farid Ghareh Mohammadi, Hamid R Arabnia, and M Hadi Amini. On parameter tuning in meta-learning for computer vision. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 300–305. IEEE, 2019.

[MAA20a] Farid Ghareh Mohammadi, M Hadi Amini, and Hamid R Arabnia. Applications of nature-inspired algorithms for dimension reduction: enabling efficient data analytics. In *Optimization, Learning, and Control for Interdependent Complex Networks*, pages 67–84. Springer, 2020.

[MAA20b] Farid Ghareh Mohammadi, M Hadi Amini, and Hamid R Arabnia. Evolutionary computation, optimization, and learning algorithms for data science. In *Optimization, Learning, and Control for Interdependent Complex Networks*, pages 37–65. Springer, 2020.

[MBN02] Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. Feature selection algorithms: A survey and experimental evaluation. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 306–313. IEEE, 2002.

[MCQDCJA10] Alexis Marcano-Cedeño, J Quintanilla-Domínguez, MG Cortina-Januchs, and Diego Andina. Feature selection using sequential forward selection and classification applying artificial metaplasticity neural network. In *IECON 2010-36th annual conference on IEEE industrial electronics society*, pages 2845–2850. IEEE, 2010.

[MDG10] Snehal A Mulay, PR Devale, and GV Garje. Intrusion detection system using support vector machine and decision tree. *International Journal of Computer Applications*, 3(3):40–43, 2010.

[MDPS21] C Mala, Vishnu Deepak, Sidharth Prakash, and Surya Lashmi Srinivasan. A hybrid artificial bee colony algorithmic approach for classification using neural networks. In *2nd EAI International Conference on Big Data Innovation for Sustainable Cognitive Computing*, pages 339–359. Springer, 2021.

[MGDL04] Vittorio Maniezzo, Luca Maria Gambardella, and Fabio De Luigi. Ant colony optimization. In *New Optimization Techniques in Engineering*, pages 101–121. Springer, 2004.

[MISL18] Abdallah Moubayed, MohammadNoor Injadat, Abdallah Shami, and Hanan Lutfiyya. Dns typo-squatting domain detection: A data analytics & machine learning based approach. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2018.

[MML14] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in Engineering Software*, 69:46–61, 2014.

[MSAA19] Farid Ghareh Mohammadi, Farzan Shenavarmasouleh, M Hadi Amini, and Hamid R Arabnia. Evolutionary algorithms and efficient data analytics for image processing. *arXiv preprint arXiv:1907.12914*, 2019.

[MSAA21] Farid Ghareh Mohammadi, Farzan Shenavarmasouleh, M Hadi Amini, and Hamid R Arabnia. Evolutionary algorithms and efficient data analytics for image processing. In *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*. IEEE, 2021.

[MSM19] Mehrnaz Mazini, Babak Shirazi, and Iraj Mahdavi. Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and adaboost algorithms. *Journal of King Saud University-Computer and Information Sciences*, 31(4):541–553, 2019.

[NC12] Andrew A Neath and Joseph E Cavanaugh. The bayesian information criterion: background, derivation, and applications. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(2):199–203, 2012.

[NGS$^+$14] Kenney Ng, Amol Ghoting, Steven R Steinhubl, Walter F Stewart, Bradley Malin, and Jimeng Sun. Paramo: a parallel predictive modeling platform for healthcare analytic research using electronic health records. *Journal of biomedical informatics*, 48:160–170, 2014.

[NMS09] Michael Negnevitsky, Paras Mandal, and Anurag K Srivastava. Machine learning applications for load, price and wind power prediction in power systems. In *2009 15th International Conference* *on Intelligent System Applications to Power Systems*, pages 1–6. IEEE, 2009.

[NRS21] Motahare Namakin, Modjtaba Rouhani, and Mostafa Sabzekar. An evolutionary correlation-aware feature selection method for classification problems. *arXiv preprint arXiv:2110.13082*, 2021.

[OAE20] Salima Ouadfel and Mohamed Abd Elaziz. Enhanced crow search algorithm for feature selection. *Expert Systems with Applications*, 159:113572, 2020.

[OB19] Tayyip Ozcan and Alper Basturk. Transfer learning-based convolutional neural networks with heuristic optimization for hand gesture recognition. *Neural Computing and Applications*, 31(12):8955–8970, 2019.

[OB20] Tayyip Ozcan and Alper Basturk. Human action recognition with deep learning and structural optimization using a hybrid heuristic algorithm. *Cluster Computing*, pages 1–14, 2020.

[oH03] Een Studie op Hyperspectrale. *Feature Selection and Classifier Ensembles: A Study on Hyperspectral Remote Sensing Data.* PhD thesis, University of Antwerp, 2003.

[Olo18] Skogby Steinholtz Olof. A comparative study of black-box optimization algorithms for tuning of hyper-parameters in deep neural networks, 2018.

[OOL$^+$12] Matthew W Ohland, Marisa K Orr, Richard A Layton, Susan M Lord, and Russell A Long. Introducing “stickiness” as a versatile metric of engineering persistence. In *2012 Frontiers in Education Conference Proceedings*, pages 1–5. IEEE, 2012.

[OSBS03] Luiz S Oliveira, Robert Sabourin, Flavio Bortolozzi, and Ching Y Suen. A methodology for feature selection using multiobjective genetic algorithms for handwritten digit string recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(06):903–929, 2003.

[OZTA04] Matthew Ohland, Guili Zhang, Brian Thorndyke, and Timothy J Anderson. The creation of the multiple institution database for investigating engineering longitudinal development (midfield). In *2004 Annual Conference*, pages 9–1244, 2004.

[PA14] Alejandro Peña-Ayala. Educational data mining: A survey and a data mining-based analysis of recent works. *Expert systems with applications*, 41(4):1432–1462, 2014.

[PBH18] Yamuna Prasad, KK Biswas, and Madasu Hanmandlu. A recursive pso scheme for gene selection in microarray data. *Applied Soft Computing*, 71:213–225, 2018.

[PC09] You-Jin Park and Kun-Nyeong Chang. Individual and group behavior-based customer profile model for personalized product recommendation. *Expert Systems with Applications*, 36(2):1932–1939, 2009.

[PC18] Juliano Pierezan and Leandro Dos Santos Coelho. Coyote optimization algorithm: a new metaheuristic for global optimization problems. In *2018 IEEE congress on evolutionary computation (CEC)*, pages 1–8. IEEE, 2018.

[PD17] Priyanka S Patil and Nagaraj V Dharwadkar. Analysis of banking data using machine learning. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pages 876–881. IEEE, 2017.

[pjm] Pjm - home.

[PMG$^+$17] Nikolaos G Paterakis, Elena Mocanu, Madeleine Gibescu, Bart Stappers, and Walter van Alst. Deep learning versus traditional machine learning methods for aggregated energy demand prediction. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE, 2017.

[PPA19] Elnaz Pashaei, Elham Pashaei, and Nizamettin Aydin. Gene selection using hybrid binary black hole algorithm and modified binary particle swarm optimization. *Genomics*, 111(4):669–686, 2019.

[PS18] Venkatesh Pandiri and Alok Singh. A hyper-heuristic based artificial bee colony algorithm for k-interconnected multi-depot multi-traveling salesman problem. *Information Sciences*, 463:261–281, 2018.

[PVG$^+$11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[PX13] Xinjun Peng and Dong Xu. A local information-based feature-selection algorithm for data regression. *Pattern Recognition*, 46(9):2519–2530, 2013.

[QA18] Omar Saber Qasim and Zakariya Yahya Algamal. Feature selection using particle swarm optimization-based logistic regression model. *Chemometrics and Intelligent Laboratory Systems*, 182:41–46, 2018.

[RBNF21] Mehrdad Rostami, Kamal Berahmand, Elahe Nasiri, and Saman Forouzandeh. Review of swarm intelligence-based feature selection methods. *Engineering Applications of Artificial Intelligence*, 100:104210, 2021.

[Ric15] Shane A Richards. Likelihood and model selection. 2015.

[RPA$^+$13] Douglas Rodrigues, Luis AM Pereira, TNS Almeida, João Paulo Papa, AN Souza, Caio CO Ramos, and Xin-She Yang. Bcs: A binary cuckoo search algorithm for feature selection. In *2013 IEEE International symposium on circuits and systems (ISCAS)*, pages 465–468. IEEE, 2013.

[RR15] A Sylvia Selva Rani and RR Rajalaxmi. Unsupervised feature selection using binary bat algorithm. In *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, pages 451–456. IEEE, 2015.

[RTS08] Shahryar Rahnamayan, Hamid R Tizhoosh, and Magdy MA Salama. Opposition-based differential evolution. *IEEE Transactions on Evolutionary computation*, 12(1):64–79, 2008.

[S$^+$17] G Sahoo et al. A two-step artificial bee colony algorithm for clustering. *Neural Computing and Applications*, 28(3):537–551, 2017.

[SA20] Tarun K Sharma and Ajith Abraham. Artificial bee colony with enhanced food locations for solving mechanical engineering design problems. *Journal of Ambient Intelligence and Humanized Computing*, 11(1):267–290, 2020.

[SB01] Sebastiano B Serpico and Lorenzo Bruzzone. A new search algorithm for feature selection in hyperspectral remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 39(7):1360–1367, 2001.

[SB16] Sangeeta Sharma and Pawan Bhambu. Artificial bee colony algorithm: A survey. *International Journal of Computer Applications*, 149(4):11–19, 2016.

[SC14] S Sivakumar and C Chandrasekar. Modified pso based feature selection for classification of lung ct images. *International Journal of Computer Science and Information Technologies*, 2014.

[Sch01] Lothar M Schmitt. Theory of genetic algorithms. *Theoretical Computer Science*, 259(1-2):1–61, 2001.

[SCK22] Priynka Sharma, Kaylash Chaudhary, and MGM Khan. The art-of-hyper-parameter optimization with desirable feature selection. *Proceedings of 2021 International Conference on Medical Imaging and Computer-Aided Diagnosis (MICAD 2021)*, pages 218–227, 2022.

[SCZZ19] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. A survey of optimization methods from a machine learning perspective. *IEEE transactions on cybernetics*, 50(8):3668–3681, 2019.

[SEO21] Esra Sarac Essiz and Murat Oturakci. Artificial bee colony–based feature selection algorithm for cyberbullying. *The Computer Journal*, 64(3):305–313, 2021.

[SG18] Tarun Kumar Sharma and Preeti Gupta. Opposition learning based phases in artificial bee colony. *International Journal of System Assurance Engineering and Management*, 9(1):262–273, 2018.

[SHA19] Gehad Ismail Sayed, Aboul Ella Hassanien, and Ahmad Taher Azar. Feature selection via a novel chaotic crow search algorithm. *Neural computing and applications*, 31(1):171–188, 2019.

[Shi04] Yuhui Shi. Particle swarm optimization. *IEEE connections*, 2(1):8–13, 2004.

[SIM11] Alok Sharma, Seiya Imoto, and Satoru Miyano. A top-r feature selection algorithm for microarray gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(3):754–764, 2011.

[SK17a] Diksha Sharma and Neeraj Kumar. A review on machine learning algorithms, tasks and applications. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 6(10):2278–1323, 2017.

[SK17b] P Shunmugapriya and S Kanmani. A hybrid algorithm using ant and bee colony optimization for feature selection and classification (ac-abc hybrid). *Swarm and Evolutionary Computation*, 36:27–36, 2017.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[SPBW16] Iwan Syarif, Adam Prugel-Bennett, and Gary Wills. Svm parameter optimization using grid search and genetic algorithm to improve classification performance. *Telkomnika*, 14(4):1502, 2016.

[SR92] Andrzej Skowron and Cecylia Rauszer. The discernibility matrices and functions in information systems. In *Intelligent decision support*, pages 331–362. Springer, 1992.

[SRS11] Sugandha Singh, Navin Rajpal, and Ashok Kale Sharma. K-fault tolerant in mobile adhoc network under cost constraint. In *2011 3rd International Conference on Electronics Computer Technology*, volume 6, pages 368–372. IEEE, 2011.

[SS04] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[SSH19] Gehad Ismail Sayed, Mona Soliman, and Aboul Ella Hassanien. Parameters optimization of support vector machine based on the optimal foraging theory. In *Machine Learning Paradigms: Theory and Application*, pages 309–326. Springer, 2019.

[Ste18] Olof Skogby Steinholtz. A comparative study of black-box optimization algorithms for tuning of hyper-parameters in deep neural networks. *Luleå University of Technology*, 2018.

[STH$^+$15] Evan R Sparks, Ameet Talwalkar, Daniel Haas, Michael J Franklin, Michael I Jordan, and Tim Kraska. Automating model search for large scale machine learning. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 368–380, 2015.

[SY20] Guoxin Sui and Yong Yu. Bayesian contextual bandits for hyper parameter optimization. *IEEE Access*, 8:42971–42979, 2020.

[SZM$^+$22] Yasaman Saadati, Leila Zahedi, Farid Ghareh Mohammadi, Shabnam Rezapour, Matthew W Ohland, and M Hadi Amini. Abc-based optimal hyper-parameter tuning for electric load forecasting. *The 24th International Conference on Artificial Intelligence*, 2022.

[SZS17] Hong-Yan Shi, Wei-Feng Zhao, and Yu Su. Balanced artificial bee colony algorithm based on multiple selections. In *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, pages 692–695. IEEE, 2017.

[Tal09] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[Tal20] El-Ghazali Talbi. Optimization of deep neural networks: a survey and unified taxonomy. 2020.

[TAMS19] Jingwei Too, Abdul Rahim Abdullah, and Norhashimah Mohd Saad. A new co-evolution binary particle swarm optimization with multiple inertia weight strategy for feature selection. In *Informatics*, volume 6, page 21. Multidisciplinary Digital Publishing Institute, 2019.

[Tiz05] Hamid R Tizhoosh. Opposition-based learning: a new scheme for machine intelligence. In *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and* *internet commerce (CIMCA-IAWTIC'06)*, volume 1, pages 695–701. IEEE, 2005.

[VAQLJMM21] Leonardo Villalobos-Arias, Christian Quesada-López, Marcelo Jenkins, and Juan Murillo-Morera. Hyper-parameter tuning using genetic algorithms for software effort estimation. In *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6. IEEE, 2021.

[VBF21] Viet Minh Vu, Adrien Bibal, and Benoît Frénay. Constraint preserving score for automatic hyperparameter tuning of dimensionality reduction methods for visualization. *IEEE Transactions on Artificial Intelligence*, 2(3):269–282, 2021.

[VF05] Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In *International work-conference on artificial neural networks*, pages 758–770. Springer, 2005.

[VM21] A Helen Victoria and G Maragatham. Automatic tuning of hyperparameters using bayesian optimization. *Evolving Systems*, 12(1):217–223, 2021.

[Wan05] Lipo Wang. *Support vector machines: theory and applications*, volume 177. Springer Science & Business Media, 2005.

[WBA09] Kashif Waqas, Rauf Baig, and Shahid Ali. Feature subset selection using multi-objective genetic algorithms. In *2009 IEEE 13th International Multitopic Conference*, pages 1–6. IEEE, 2009.

[WF02] Ian H Witten and Eibe Frank. Data mining: practical machine learning tools and techniques with java implementations. *Acm Sigmod Record*, 31(1):76–77, 2002.

[Wit05] Carsten Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 44–56. Springer, 2005.

[WLU20] Jonathan Waring, Charlotta Lindvall, and Renato Umeton. Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artificial Intelligence in Medicine*, 104:101822, 2020.

[WT08] Wu-Chang Wu and Men-Shen Tsai. Feeder reconfiguration using binary coding particle swarm optimization. *International Journal of Control, Automation, and Systems*, 6(4):488–494, 2008.

[WZS$^+$20] Xiao-han Wang, Yong Zhang, Xiao-yan Sun, Yong-li Wang, and Chang-he Du. Multi-objective feature selection based on artificial bee colony: an acceleration approach with variable sample size. *Applied Soft Computing*, 88:106041, 2020.

[XFY13] Yunfeng Xu, Ping Fan, and Ling Yuan. A simple and efficient artificial bee colony algorithm. *Mathematical Problems in Engineering*, 2013, 2013.

[XTPL20] Yu Xue, Tao Tang, Wei Pang, and Alex X Liu. Self-adaptive parameter and strategy based particle swarm optimization for large-scale feature selection problems with multiple classifiers. *Applied Soft Computing*, 88:106031, 2020.

[XY17] Lingxi Xie and Alan Yuille. Genetic cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1379–1388, 2017.

[YA98] Howard Hua Yang and Shun-ichi Amari. Complexity issues in natural gradient descent method for training multilayer perceptrons. *Neural Computation*, 10(8):2137–2157, 1998.

[YCY$^+$21] Danial Yazdani, Ran Cheng, Donya Yazdani, Jürgen Branke, Yaochu Jin, and Xin Yao. A survey of evolutionary continuous dynamic optimization over two decades—part b. *IEEE Transactions on Evolutionary Computation*, 25(4):630–650, 2021.

[YDwWq16] Zhang Yong, Gong Dun-wei, and Zhang Wan-qiu. Feature selection of unreliable data using an improved multi-objective pso algorithm. *Neurocomputing*, 171:1281–1290, 2016.

[YGW16] Hua Yin, Keke Gai, and Zhijian Wang. A classification algorithm based on ensemble feature selections for imbalanced-class dataset. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, pages 245–249. IEEE, 2016.

[YHB16] Li Ying and Xie Hai-Bo. Improved svm algorithm method of mine geological environment evaluation model. *Metal Mine*, 45(09):189, 2016.

[YO11] Jian-Bo Yang and Chong-Jin Ong. Feature selection using probabilistic prediction of support vector regression. *IEEE transactions on neural networks*, 22(6):954–962, 2011.

[YS20] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[YWC+18] Quanming Yao, Mengshuo Wang, Yuqiang Chen, Wenyuan Dai, Yu-Feng Li, Wei-Wei Tu, Qiang Yang, and Yang Yu. Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306*, 2018.

[YZ20] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.

[ZDZY18] Tao Zhang, Biyun Ding, Xin Zhao, and Qianyu Yue. A fast feature selection algorithm based on swarm intelligence in acoustic defect detection. *IEEE Access*, 6:28848–28858, 2018.

[ZEPS16] Hossam M Zawbaa, Eid Emary, Bazil Parv, and Marwa Sharawi. Feature selection approach based on moth-flame optimization algorithm. In *2016 IEEE congress on evolutionary computation (CEC)*, pages 4612–4617. IEEE, 2016.

[ZER+21] Leila Zahedi, Hossein Ebrahiminejad, Monique S. Ross, Matthew W. Ohland, and Stephanie J. Lunn. Multi-institution study of student demographics and stickiness of computing majors in the usa. In *2021 CoNECD*, Virtual - 1pm to 5pm Eastern Time Each Day, January 2021. ASEE Conferences. https://peer.asee.org/36110.

[ZFLS19] Yun Zhou, Shao Jun Fang, Hong Mei Liu, and Te Shao. The application of iabc_keans in array antenna pattern synthesis. In *2019 Photonics & Electromagnetics Research Symposium-Fall (PIERS-Fall)*, pages 1218–1223. IEEE, 2019.

[ZGMA22] Leila Zahedi, Farid Ghareh Mohammadi, and Mohammad Hadi Amini. A2bcf: An automated abc-based feature selection algorithm for classification models in an education application. *Applied Sciences*, 12(7):3553, 2022.

[ZH19] Marc-André Zöller and Marco F Huber. Benchmark and survey of automated machine learning frameworks. *arXiv preprint arXiv:1904.12054*, 2019.

[Zio88] Stanley Zionts. Multiple criteria mathematical programming: an updated overview and several approaches. *Mathematical models for decision support*, pages 135–167, 1988.

[ZLGP12] Iman Ziari, Gerard Ledwich, Arindam Ghosh, and Glenn Platt. Integrated distribution systems planning to improve reliability under load growth. *IEEE transactions on Power Delivery*, 27(2):757–765, 2012.

[ZLP$^+$20] Leila Zahedi, Stephanie J Lunn, S Pouyanfar, MS Ross, and MW Ohland. Leveraging machine learning techniques to analyze computing persistence in undergraduate programs. In *2020 ASEE Virtual Annual Conference Content Access*, pages 2–34921, 2020.

[ZLS15] Jia Zhao, Li Lv, and Hui Sun. Artificial bee colony using opposition-based learning. In *Genetic and evolutionary computing*, pages 3–10. Springer, 2015.

[ZMA21a] Leila Zahedi, Farid Ghareh Mohammadi, and M Hadi Amini. Hyp-abc: A novel automated hyper-parameter tuning algorithm using evolutionary optimization. ., 2021.

[ZMA21b] Leila Zahedi, Farid Ghareh Mohammadi, and M Hadi Amini. Hyp-abc: A novel automated hyper-parameter tuning algorithm using evolutionary optimization. *techrxiv preprint techrxiviv:1810.13306*, 2021.

[ZMA21c] Leila Zahedi, Farid Ghareh Mohammadi, and M Hadi Amini. Opt-abc: an optimal hyperparameter tuning approach for machine learning algorithms. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1138–1145. IEEE, 2021.

[ZMR$^+$21] Leila Zahedi, Farid Ghareh Mohammadi, Shabnam Rezapour, Matthew W Ohland, and M Hadi Amini. Search algorithms for automated hyper-parameter tuning. *The 17th International Conference on Data Science (Accepted)*, 2021.

[ZWRM21] Alejandro Gabriel Villanueva Zacarias, Christian Weber, Peter Reimann, and Bernhard Mitschang. Assistml: A concept to recommend ml solutions for predictive use cases. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–12. IEEE, 2021.

[ZWZ21] Ziwei Zhang, Xin Wang, and Wenwu Zhu. Automated machine learning on graphs: A survey. *arXiv preprint arXiv:2103.00742*, 2021.

[ZYC$^+$19] Wenbo Zhu, Weichang Yeh, Jianwen Chen, Dafeng Chen, Aiyuan Li, and Yangyang Lin. Evolutionary convolutional neural networks using abc. In *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, pages 156–162, 2019.

[ZZ20] Haitong Zhao and Changsheng Zhang. A decomposition-based many-objective artificial bee colony algorithm with reinforcement learning. *Applied Soft Computing*, 86:105879, 2020.

[ZZWC20] Cuijie Zhao, Hongdong Zhao, Guozhen Wang, and Hong Chen. Improvement svm classification performance of hyperspectral image using chaotic sequences in artificial bee colony. *IEEE Access*, 8:73947–73956, 2020.

VITA

LEILAZAHEDI

2010	B.S., Computer Engineering University of Isfahan Isfahan, Iran
2016	M.S., Information Technology Management Information Systems Yazd University of Science and Art Yazd, Iran
2021	M.S., Computer Science Florida International University Miami, Florida
2022	Ph.D. Candidate, Computer Science Florida International University Miami, Florida

PUBLICATIONS

Zahedi, Leila, Farid Ghareh Mohammadi, and M. Hadi Amini. *A2BCF: an ABC-based Feature Selection approach for Automated Machine Learning.* Special issue on Technologies and Environments of Intelligent Education. Applied Sciences, 2022, Under review.

Zahedi, Leila, Farid Ghareh Mohammadi, and M. Hadi Amini. *HyP-ABC: A Novel Automated Hyper-Parameter Tuning Algorithm Using Evolutionary Optimization.* IEEE Access, 2022, Under review.

Zahedi, Leila, Farid Ghareh Mohammadi, and M. Hadi Amini. *"OptABC: an Optimal Hyperparameter Tuning Approach for Machine Learning Algorithms."* In 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 1138-1145. IEEE, 2021.

Zahedi, Leila, Farid Ghareh Mohammadi, Shabnam Rezapour, Matthew W Ohland and M. Hadi Amini. *Search algorithms for automated hyper-parameter tuning.* The 17th International Conference on Data Science (ICDATA'21). Springer Nature, 2021.

Saadati, Yasaman*[1], Leila Zahedi*, Farid Ghareh Mohammadi and M. Hadi Amini.

[1]These authors (*) contributed equally to this work

ABC-based Optimal Hyper-parameter Tuning for Electric Load Forecasting. The 24th International Conference on Artificial Intelligence (ICAI'22), Springer Nature, 2022.

Zahedi, Leila, Stephanie Lunn, Samira Pouyanfar, Monique Ross, and Matthew W Ohland *Leveraging Machine Learning Techniques to Analyze Computing Persistence in Undergraduate Programs* 2020 ASEE Virtual Annual Conference Content Access(ASEE '20). ASEE, 2020.

Zahedi, Leila, and M. Hadi Amini. *Hyperparameter Optimization for Improving Educational Decision Making.* Presented at Grace Hopper Celebration of Women in Computer Science (GHC21) Conference. 2021.

Zahedi, Leila, and M. Hadi Amini. *Hyper-Parameter Optimization of Machine Learning Algorithms.* Presented at ACM Richard Tapia (vTAPIA21) Conference. 2021.

Zahedi, Leila, and Stephanie Lunn, and Monique Ross. *A Comparative Analysis of Machine Learning Techniques for Predicting Student Persistence in Computing Programs* Presented at ACM Richard Tapia (vTAPIA20) Conference. 2020.