informatics

Article

Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis

Enas Elgeldawi [1,*], Awny Sayed [1,2], Ahmed R. Galal [1] and Alaa M. Zaki [1]

[1] Computer Science Department, Faculty of Science, Minia University, Minia 61519, Egypt; awny.sayed@mu.edu.eg (A.S.); arabie@mu.edu.eg (A.R.G.); alaa_zaki@mu.edu.eg (A.M.Z.)

[2] Faculty of Computing and Information Technology, Information Technology Department, King Abdulaziz University, Jeddah 21589, Saudi Arabia

* Correspondence: enas.elgeldawi@mu.edu.eg

Abstract: Machine learning models are used today to solve problems within a broad span of disciplines. If the proper hyperparameter tuning of a machine learning classifier is performed, significantly higher accuracy can be obtained. In this paper, a comprehensive comparative analysis of various hyperparameter tuning techniques is performed; these are Grid Search, Random Search, Bayesian Optimization, Particle Swarm Optimization (PSO), and Genetic Algorithm (GA). They are used to optimize the accuracy of six machine learning algorithms, namely, Logistic Regression (LR), Ridge Classifier (RC), Support Vector Machine Classifier (SVC), Decision Tree (DT), Random Forest (RF), and Naive Bayes (NB) classifiers. To test the performance of each hyperparameter tuning technique, the machine learning models are used to solve an Arabic sentiment classification problem. Sentiment analysis is the process of detecting whether a text carries a positive, negative, or neutral sentiment. However, extracting such sentiment from a complex derivational morphology language such as Arabic has been always very challenging. The performance of all classifiers is tested using our constructed dataset both before and after the hyperparameter tuning process. A detailed analysis is described, along with the strengths and limitations of each hyperparameter tuning technique. The results show that the highest accuracy was given by SVC both before and after the hyperparameter tuning process, with a score of 95.6208 obtained when using Bayesian Optimization.

Keywords: hyperparameter tuning; Arabic sentiment analysis; machine learning

check for updates

Citation: Elgeldawi, E.; Sayed, A.; Galal, A.R.; Zaki, A.M. Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis. *Informatics* **2021**, *8*, 79. https://doi.org/10.3390/informatics8040079

Academic Editor: Rigas Kotsakis

Received: 7 October 2021
Accepted: 13 November 2021
Published: 17 November 2021

1. Introduction

Social media has attracted billions of people to interact with each other. Today, many people use such media on a daily basis, not only as a platform to socialize with one another, but also to share opinions, commentary, and experiences. The significant impact of social media has been addressed in abundant research literature [1–3]. The importance of sentiment analysis comes from its ability to find conclusions and draw indirect inferences from a huge amount of given data. Analyzing a text written in a language with a complex morphology, such as the Arabic language, has always been a multi-level challenging process. Machine learning has become a prominent part of such a process. Today, machine learning is everywhere; we encounter many machine learning applications in our daily lives, from checking the weather for the next week or transferring our handwritten notes to a typed document, to adjusting the car seat to our preferred position. Machine learning algorithms have gained popularity in such a way that almost every scientific field now uses them. They have become integrated into numerous scientific disciplines, such as networking [4], transportation [5], text analysis [6,7], and bioinformatics [8]. This broad range of machine learning disciplines is due to their promising results and predictive performance in solving classification problems.

Machine learning algorithms automatically learn and hence adjust their internal parameters based on data. These types of parameters are called "model parameters", or simply "parameters" for short. However, there are other parameters that are not adjusted during the learning process but, rather, have to be pre-configured before the learning process starts. Such parameters are often referred to as "hyperparameters". The model parameters indicate how the input data are converted into the desired output, while the hyperparameters indicate how the model is structured. The performance of a machine learning model can drastically change depending on the choice and the values of its hyperparameters.

For example, the decision tree algorithm has a "tree_depth" hyperparameter; setting a moderate value for this hyperparameter can obtain good results, while a high value can lower the algorithm's performance. For this reason, hyperparameters should be set very carefully. Hyperparameters could be set using a number of methods for a specific dataset. One is to set them manually and calculate the accuracy accordingly. Then, other values of the hyperparameters can be tested and, with every adjustment, the corresponding accuracy can be calculated. Manually setting the values for the hyperparameters in such a trial-and-error fashion is a cumbersome and time-consuming process. Another method to find an appropriate hyperparameter configuration is to safely choose the default values of hyperparameters that are recommended by the software packages used in the implementation, which are in turn based on recommendations from the literature, as well as experience. Sometimes, the default values work well for a certain dataset, but this does not always mean that they give the best accuracy.

Alternatively, we can use hyperparameter optimization strategies. These strategies are data-dependent optimization algorithms, which try to minimize the expected generalization error of a machine learning model over the hyperparameter search space of considered candidate configurations. The ML algorithms have been analyzed first using the default hyperparameters values to be compared afterwards with the results obtained when using hyperparameter tuning algorithms. Most of the manuscripts in literature usually test the effect of using one hyperparameter tuning approach on the accuracy of one or more machine language techniques when solving a certain classification problem. The nature of the problem plays an important role in both the classification accuracy of the machine learning technique and the hyperparameters combination that provide the best classification accuracy.

The novelty of the manuscript is in the number of hyperparameter tuning algorithms being compared, the number of machine learning techniques being tested, and, the most important, the nature of the classification problem which is Arabic sentiment analysis. A number of manuscripts have used machine learning approaches to solve Arabic sentiment analysis, but none (as far as we know) have used hyperparameter tuning algorithms to find the best hyperparameters that will lead to the best classification accuracy for the machine learning algorithm used. It should be noted that this set of hyperparameters is not the same for every classification problem and differs according to the nature of the problem.

In this paper, five search strategies are used; these are Grid Search, Random Search, Bayesian Optimization, Particle Swarm Optimization (POS), and Genetic Algorithm (GA). They are used to hyperparameter-tune six machine learning algorithms, namely Logistic Regression (LR), Ridge Classifier (RC), Support Vector Machine Classifier (SVC), Decision Tree (DT), Random Forest (RF), and Naive Bayes (NB) classifiers. These algorithms are used in Arabic sentiment analysis in order to determine the tonality of Arabic reviews using our dataset, which contains 7000 reviews written in different forms of the Arabic language. The accuracy of the algorithms mentioned is calculated when the default values of the hyperparameters are used, and then calculated again after each of the hyperparameter tuning strategies are used. A before-and-after comparison is given. The contributions of our work can be summarized as follows:

- The paper reviews common hyperparameter tuning techniques, their benefits, and their drawbacks.

- A comprehensive comparative analysis among five hyperparameter tuning algorithms is given, as most of the previous and current literature typically focuses only on Grid Search and Random Search.
- Hyperparameter tuning for six machine learning models is performed to analyze sentiments over an Arabic text.
- The Arabic language is a challenging language; this paper is considered the first hyperparameter tuning study performed on an Arabic text.

The paper is organized as follows. Related work is given in Section 2. The five hyperparameter optimization models are presented in Section 3. Section 4 describes our Arabic sentiment analysis scheme. The experimental results and conclusions are given in Sections 5 and 6, respectively.

2. Related Work

In this paper, we present the use of hyperparameter tuning of machine learning algorithms to tackle the sentiment analysis problem of Arabic reviews. It is important to shed light on previous related work on both hyperparameter tuning and sentiment analysis.

2.1. Hyperparameter Tunning

Most of the previous work in hyperparameter tuning tends to focus only on Grid Search and Random Search, or a comparison between them [9–13]. A comprehensive study on tunability is given in [14]; the authors constructed the tuning problem in statistical terms as well as suggesting tunability quantifying measures of algorithms' hyperparameters. They presented a comprehensive study based on 38 datasets from the OpenML platform. In [15], the authors showed that the performance of SVM can be improved significantly using parameter optimization. They applied two methods which are Grid Search and Genetic Algorithm. Based on the average running time on different datasets, GA was almost 16 times faster than Grid Search. Another comparison between Grid search and Genetic Algorithms was introduced in [16]. The result of the implementation showed that Genetic Algorithm can find the hyperparameters with faster computational time than Grid search. The authors tested their comparison against Support Vector Machine, Random forest, Adaptive Boosting,and K Nearest Neighbour. In [17], the authors proposed the use of an evolutionary algorithm named SHADE to optimize the configuration of a deep learning model for the sentiment analysis of Spanish tweets. Their results showed that the hyperparameters found by the evolutionary algorithm enhanced the performance of the deep learning method. A Word2Vec model based on a convolutional neural network was constructed in [18]. The authors used a dataset collected from different newspapers over a number of Arabic countries. Our work differs from the work listed here in the sense that our main concern is increasing the accuracy of a machine learning algorithm through selection of its internal hyperparameters. An ensemble of surface and deep features for Arabic sentiment analysis was proposed in [19], and the model was evaluated on three datasets of Arabic tweets. The authors concluded that the Word2Vec method gives much better results than sentiment-specific embeddings.

2.2. Sentiment Analysis

Sentiment analysis has been a very rich topic for research. However, very few works have been conducted involving sentiment analysis of an Arabic text. In this section, we shed light on the relevant literature. In [20], the authors used three machine learning approaches, Naïve Bayes, Support Vector Machine, and K-Nearest Neighbor classifiers, to classify their in-house dataset of tweets. Their study found that SVM gives the highest precision, while K-Nearest Neighbor gives the best recall. Meanwhile, in [21], the authors studied the effect of preprocessing techniques on the classification behavior of machine learning algorithms. In [22], a sentiment analysis study on financial context news, gathered from Lithuanian language websites, has been introduced. Three machine learning algorithms have been applied (SVM, Naive Bayes, and long short term memory). The Naive Bayes algorithm has shown the best accuracy. A hybrid framework of machine learning and deep learning is proposed in [23], which combines Convolutional Neural Network and Random Forest classifier for sentiment analysis. The experimental result showed that the proposed model gives much better accuracy values than the existing base models.

The authors in [24] used two datasets, one with binary labels and the other with multiclass labels, to explore various natural language processing (NLP) methods to perform sentiment analysis. For the binary classification they applied the bag of words, and skip-gram Word2Vec models followed by Random Forest, SVM, or logistic regression classifiers. For the multi-class case, they implemented the recursive neural tensor networks (RNTN). A comprehensive comparative analysis of various machine learning classifiers is provided in [6], and a review of sentiment analysis in the Arabic Language is given in [25]. A rich study of Algerian newspaper comments is presented in [26], in which the authors created their own corpus and used Support Vector Machine and Naïve Bayes to classify Arabic comments into positive and negative sentiments.

As mentioned above, very limited research has been conducted in sentiment analysis involving the Arabic language, and there is almost no research concerning hyperparameter optimization for sentiment analysis on Arabic texts. To the best of our knowledge, this paper is considered the first study to tune machine learning hyperparameters used in the sentiment analysis of an Arabic text.

3. Hyperparameter Tuning

Speaking in statistical terms, hyperparameter tuning captures a snapshot of the current performance of a model, and compares this snapshot with others taken previously. In any machine learning algorithm, hyperparameters need to be initialized before a model starts the training. Fine-tuning the model hyperparameters maximizes the performance of the model on a validation set. In a machine learning context, a hyperparameter is a parameter whose value is set before initiating the learning process. On the other hand, the values of model parameters are derived via training the data. Model parameters refer to the weights and coefficients, which are derived from the data by the algorithm. Every algorithm has a defined set of hyperparameters; for example, for a Decision Tree, this is a depth parameter.

Before explaining our hyperparameter tuning approach, it is important to explain a process called "cross-validation", as it is considered an important step in the hyperparameter tuning process. Cross-validation (CV) is a statistical method used to estimate the accuracy of machine learning models. Once the model is trained, we cannot be certain of how well it will work on data that have not been encountered before. Assurance is needed regarding the accuracy of the prediction performance of the model. To evaluate the performance of a machine learning model, some unseen data are needed for the test. Based on the model's performance on unseen data, we can determine whether the model is underfitting, overfitting, or well-generalized. Cross-validation is considered a very helpful technique to test how effective a machine learning model is when the data in hand are limited. To perform cross-validation, a subset of the data should be set aside for testing and validating; this subset will not be used to train the model, but rather saved for later use. K-Fold is one of the most common techniques of cross-validation, and it is also the cross-validation technique that we used to validate our model (see Figure 1).

In K-Fold cross-validation, the parameter K indicates the number of folds or sections that a given dataset is split into. One of the folds is retained as a validating set and the machine learning model is trained using the remaining K-1 folds. Each fold of the K-Folds is used as a validating set at some point, with K scores (accuracy) given as a result. Finally, we average the model against each of the folds to obtain a final score for the model, as shown in Figure 1.

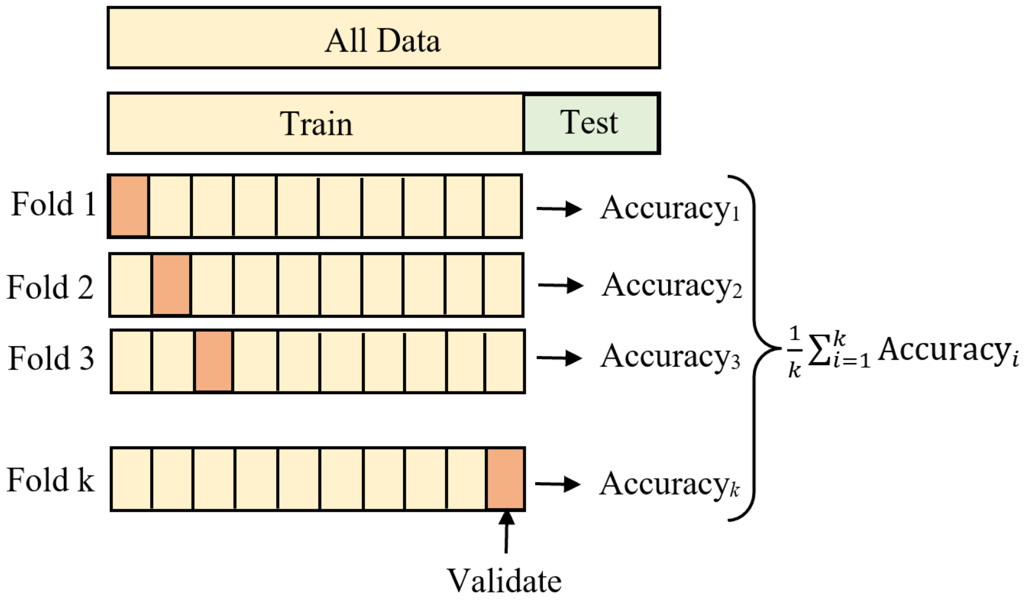


Figure 1. Cross-validation.

The importance of hyperparameters lies in their ability to directly control the behavior of the training algorithm. Choosing appropriate hyperparameters plays a a very important role in the performance of the model being trained. It is important to have three sets into which the data are divided, i.e., a training, testing, and validation set, whenever the default parameter is altered in order to obtain the necessary accuracy, so as to prevent data leaks. Thus, hyperparameter tuning can be simply defined as the process of finding the best hyperparameter values of a learning algorithm that produce the best model (see Figure 2).

The approach to finding the best hyperparameter values with respect to a specific dataset has traditionally been performed manually. To set these values, researchers often depend on their past experience of training these machine learning algorithms in solving similar problems. The problem is that the best setting for hyperparameters used to solve one problem may not be the best settings for another one, as these values will change with different datasets. Hence, it is difficult to define the hyperparameter values based on previous experience. A more automated, guided method is needed to explore alternative configurations of the machine learning model under study. Common hyperparameter optimization algorithms include Grid Search, Random Search, and Bayesian Optimization.

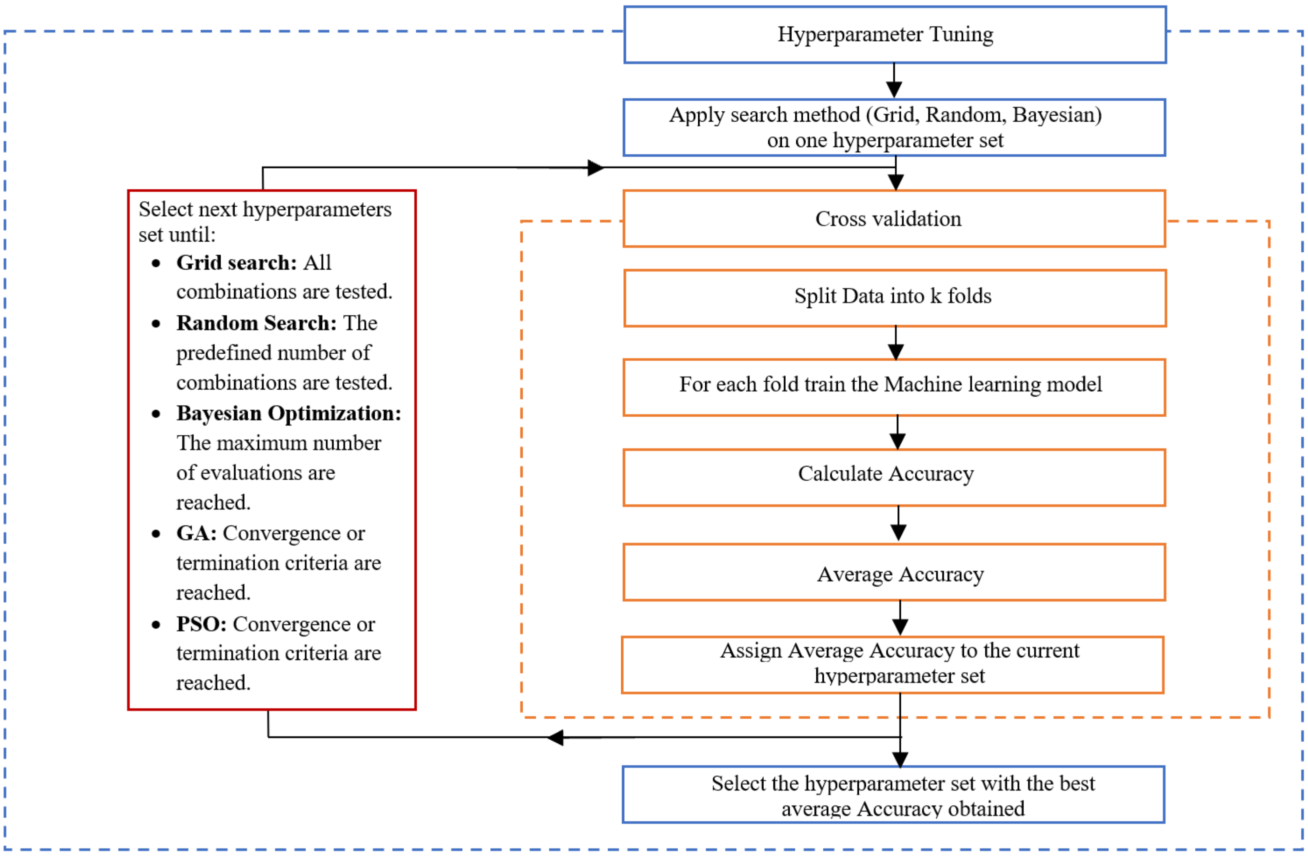

Hyperparameter Tuning
Apply search method (Grid, Random, Bayesian) on one hyperparameter set
Cross validation
Split Data into k folds
For each fold train the Machine learning model
Calculate Accuracy
Average Accuracy
Assign Average Accuracy to the current hyperparameter set
Select the hyperparameter set with the best average Accuracy obtained
Select next hyperparameters set until:
Grid search: All combinations are tested.
Random Search: The predefined number of combinations are tested.
Bayesian Optimization: The maximum number of evaluations are reached.
GA: Convergence or termination criteria are reached.
PSO: Convergence or termination criteria are reached.

Figure 2. Hyperparameter tuning process.

3.1. *Grid Search*

The most intuitive traditional approach for performing hyperparameter optimization is perhaps Grid Search [9]. It generates a Cartesian product of all possible combinations of hyperparameters. Grid Search trains the machine learning algorithm for all combinations of hyperparameters; this process should be guided by a performance metric, typically measured using the “cross-validation” technique on the training set. This validation technique ensures that the trained model obtains most of the patterns from the dataset. Grid Search is obviously the most straightforward hyperparameter tuning method. With this technique, we simply build a grid with each possible combination of all the hyperparameter values provided, calculating the score of each model, in order to evaluate it, and then selecting the model that gives the best results. To perform Grid Search, one selects a finite set of reasonable values for each hyperparameter; the Grid Search algorithm then trains the model with each combination of the hyperparameters in the Cartesian product. The performance of each combination is evaluated on a held-out validation set or through internal cross-validation on the training set. Finally, the Grid Search algorithm outputs the settings that achieve the highest performance in the validation procedure. The best set of hyperparameter values chosen in the Grid Search is then used in the actual model. Grid Search guarantees the detection of the best hyperparameters. However, one of its drawbacks is that it suffers severely when it comes to rapid convergence and dimensionality. If k parameters with n distinct values are tested, the complexity of Grid Search is expected to grow exponentially at a rate of $O(n^k)$ [27].

3.2. *Random Search*

As seen in the previous section, Grid Search is an exhaustive search for selecting a model. In Grid Search, we set up a grid of hyperparameter values, train a model, and calculate scores for the testing data for each combination, which can be very inefficient. Thus, searching 10 different hyperparameter values for each of five parameters, for example, will require 100,000 trials. If 10-fold cross-validation is used, this requires 1,000,000 model fits and 1,000,000 predictions, which would be costly in terms of both computing power and time. In contrast, Random Search [10] samples the search space and evaluates sets from a specified probability distribution. In brief, it is a technique in which random combinations of the hyperparameters are used to find the best solution for the model under consideration. For example, instead of rotating through all 100,000 samples, only 1000 random samples of hyperparameter sets are checked. The number of evaluations in Random Search must be set in the beginning, before the hyperparameter optimization process starts, and hence, the complexity of Random Search running n evaluations is $O(n)$ [28]. A significant drawback of the Random Search algorithm is that it does not use information from prior trials to select the next set and it also does not use a strategy to predict the next trial.

3.3. Bayesian Optimization

A more appealing approach to fine-tune hyperparameters through automated model tuning is the Bayesian Optimization algorithm. Bayesian Optimization is an informed search algorithm, which means that each iteration of this algorithm learns from the previous one, and the results of one iteration help in creating the next one. Bayesian Optimization resembles the Random Search method in the sense that it samples a subset of hyperparameter combinations; however, they differ in the way in which each combination is chosen. Bayesian Optimization [29] views the hyperparameter tuning process as the optimization of a black-box function. The function to be optimized is the model's final prediction score (i.e., accuracy) on a held-out test set. Any global optimization framework can then be applied to minimize this function. The model used to approximate the objective function is called a surrogate model. A number of surrogate models for Bayesian Optimization are in use, but perhaps the most common one is the Gaussian Process (GP). This is also the one used in our study. First, a few hyperparameter combinations are randomly chosen and tested. These randomly selected hyperparameters' values are then used to produce the first model of the objective function. Once a first draft of the model is obtained, there are two methods to choose the next combination: one can either choose a value close to the highest obtained value, where performance is likely to increase, or one can explore another subset of the hyperparameter search space, where another maximum could be found. Applying Bayesian Optimization with the Gaussian Process fitness function over a dataset of size n has a time complexity of $O(n^3)$ and space complexity of $O(n^2)$ [30]. The advantage of Bayesian Optimization is that it does not sample each combination in the search space, as Grid Search does, and, at the same time, it proceeds in a more systematic manner than Random Search.

3.4. Genetic Algorithm

Metaheuristic algorithms are mainly those approaches inspired by biological theories; they are known for their ability to solve non-continuous and non-convex optimization problems. One common example of a metaheuristic algorithm is Genetic Algorithm (GA). Metaheuristic algorithms generally begin by creating a population that represents a generation at each iteration. Each generation consists of a number of candidate solutions called individuals, which in turn have a number of properties represented by chromosomes. Each candidate solution (individual) in every generation is then evaluated until a global optimum is found [31]. Metaheuristic algorithms differ in the methods used to generate and select populations.

Genetic Algorithm (GA) [32] is based on the evolutionary theory that individuals with the best capability to adapt to environmental changes are more likely to survive and hence pass on their capabilities to future generations. As the next generation inherits the characteristics of the parents, they may produce individuals with better or worse versions of the parents. Better individuals will be more likely to survive and consequently have better offspring, while the worse individuals will gradually disappear. After several generations, the individual carrying the best chromosomes (properties) will be identified as the global optimum [33].

When applying GA to the hyperparameter optimization problem, each hyperparameter is represented by a chromosome, and the value of the hyperparameter is set to the decimal value of the representative chromosome. Every chromosome has several genes represented in binary digit format. Chromosome selection, crossover, and mutation operations are then performed on the genes of this chromosome to identify the optimal parameters. Chromosomes with good fitness function values will be selected and passed to the next generation with higher probability. In the next generation, these chromosomes generate new ones carrying the best characteristics of their parents. Crossover is used to generate new chromosomes by swapping proportions of the genes of different chromosomes, which represent the solution, in order to obtain the mixing of solutions in the search space [34]. Mutation is another kind of operation that can also be used to generate new chromosomes by randomly changing one or more genes of a chromosome [35]. Crossover and mutation operations ensure the diversity of the later generations, enable them to possess different characteristics, and reduce the chance of missing good ones [36].

Setting up the initialization step with an appropriate estimate is always considered an important step in any optimization problem, as it may speed up the convergence of the algorithm. Even if the initial values will iteratively be enhanced during the convergence process, it is always wise to choose initial values that bring the convergence closer to the optimum point without being trapped in unpromising local regions within the search space.

One of the advantages of Genetic Algorithm is that its initial values are randomly selected, making it very easy to implement without requiring excessive effort in choosing good initial values. This is because the mutation, crossover, and selection operations ensure that the the global optimum is not missed. However, being a sequential execution algorithm, the possibility to parallelize is lower. The time complexity of GA is $O(n^2)$ [37], which is considered a low convergence speed.

3.5. Particle Swarm Optimization

Another powerful metaheuristic example is Particle Swarm Optimization (PSO) [38,39], which is commonly used to solve optimization problems. PSO solves a problem by trying to optimize a solution in an iterative way with respect to some measure of quality by enabling a group of particles (swarm) to scan the search space in a semi-random manner [40]. PSO algorithms find the optimal solution through information sharing and cooperation among the particles in a group. In PSO, a swarm has a group of particles and each particle is represented by a vector containing the current position, the current velocity, and the best known position of the particle so far. After initializing the position and velocity of each particle, the current position and performance score of every particle are calculated. In the next iteration, the velocity of each particle is changed based on the calculated information from previous iterations, namely the position and the current global optimal position. The particles then move based on their new velocity vectors [41]. These steps are repeated until some convergence or termination criteria are reached.

PSO is easier to implement than GA, since PSO does not include crossover and mutation operations. In GA, all chromosomes share information with each other, so the entire population moves uniformly toward the optimal region, while, in PSO, only the individual best particle and the global best particle are transmitted to others. The computational complexity of the PSO algorithm is $O(nlogn)$ [42]. Generally, PSO has faster convergence than GA. Moreover, PSO is easy to parallelize since particles operate independently and only need to share information with each other after each iteration [36].

Unlike GA, the population in the PSO algorithm needs proper initialization; otherwise, it might only find a local optimum and lose the global optimum, especially for discrete hyperparameters. Many population initialization approaches have been developed to improve the performance and convergence of PSO [43]. However, using one of the population initialization approaches will increase the complexity and the execution time of PSO. Table 1 provides a complexity, parallelization, and initialization comparison among the different techniques.

Table 1. Comparison of hyperparameter tuning algorithms (where k is the number of hyperparameters and n is the number of values for hyperparameters).

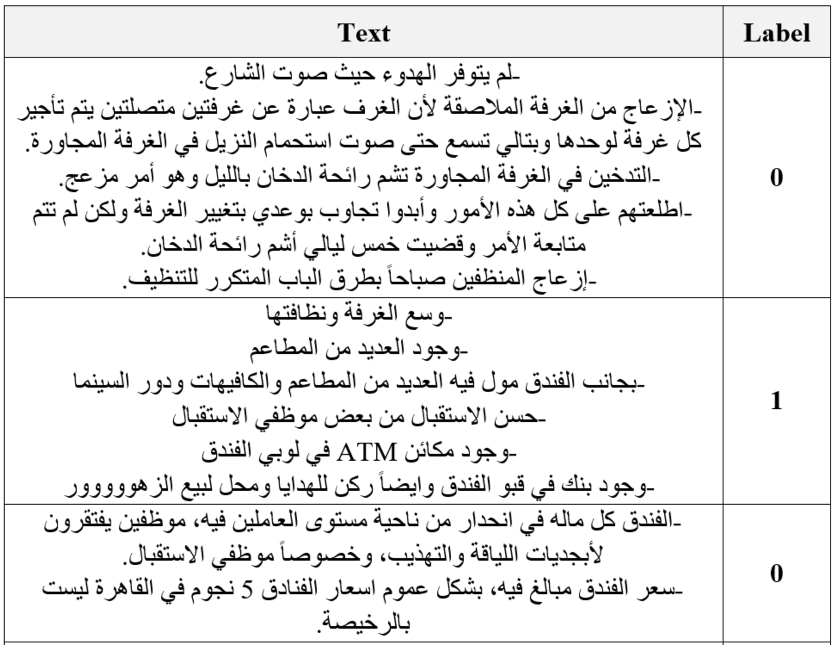
HP Approach	Complexity	Enable Parallelization	Easy Initialization
Grid	$O(n^k)$	-	✓
Random	$O(n)$	✓	✓
Bayesian	$O(n^k)$	-	✓
PSO	$O(nlogn)$	✓	-
GA	$O(n^2)$	-	✓

4. Proposed Architecture for Arabic Sentiment Analysis

We used our own constructed dataset for hotel reviews. To better analyze the polarity of these reviews as being positive or negative, useless content was removed. A brief discussion is given in the following subsections to explain the steps undertaken by our system in order to be fully constructed. Below is a description of each step.

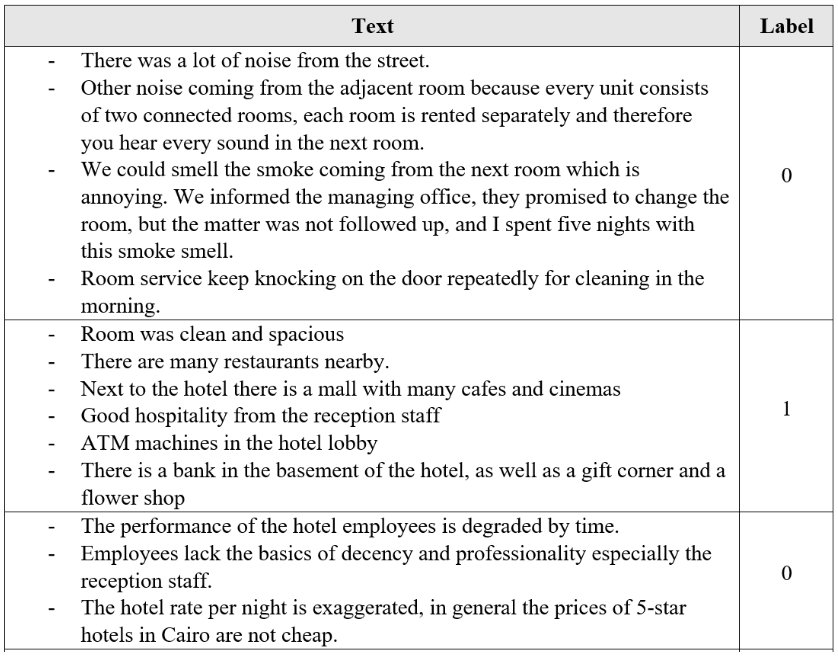
4.1. Data Collection

We used the Booking.com website to collect hotel reviews. A total of 3500 positive and 3500 negative hotel reviews were collected, generating a total of 7000 reviews. The number of words in the dataset is 95,906 words, with an avarage of 13 words per review. The corpus was named the Reviews Sentiment Analysis Corpus (RSAC), and it is available at the link provided in [44]. The reviews in RSAC are written in both standard Arabic and dialectical Arabic, which contains non-standard Arabic vocabulary, such as onomatopoeia and mimetic words. Figure 3 shows sample records from our constructed dataset (RSAC), and an English translation of the dataset is provided in Figure 4.



Text	Label
-لم يتوفر الهدوء حيث صوت الشارع. -الإزعاج من الغرفة الملاصقة لأن الغرف عبارة عن غرفتين متصلتين يتم تأجير كل غرفة لوحدها وبتالي تسمع حتى صوت استحمام النزيل في الغرفة المجاورة. -التدخين في الغرفة المجاورة تشم رائحة الدخان بالليل وهو أمر مزعج. -اطلعتهم على كل هذه الأمور وأبدوا تجاوب بوعدي بتغيير الغرفة ولكن لم تتم متابعة الأمر وقضيت خمس ليالي أشم رائحة الدخان. -إزعاج المنظفين صباحاً بطرق الباب المتكرر للتنظيف.	**0**
-وسع الغرفة ونظافتها -وجود العديد من المطاعم -بجانب الفندق مول فيه العديد من المطاعم والكافيهات ودور السينما -حسن الاستقبال من بعض موظفي الاستقبال -وجود مكائن ATM في لوبي الفندق -وجود بنك في قبو الفندق وايضاً ركن للهدايا ومحل لبيع الزهووووور	**1**
-الفندق كل ماله في انحدار من ناحية مستوى العاملين فيه، موظفين يفتقرون لأبجديات اللياقة والتهذيب، وخصوصاً موظفي الاستقبال. -سعر الفندق مبالغ فيه، بشكل عموم اسعار الفنادق 5 نجوم في القاهرة ليست بالرخيصة.	**0**

Figure 3. Sample records of our constructed dataset (RSAC).



Text	Label
- There was a lot of noise from the street. - Other noise coming from the adjacent room because every unit consists of two connected rooms, each room is rented separately and therefore you hear every sound in the next room. - We could smell the smoke coming from the next room which is annoying. We informed the managing office, they promised to change the room, but the matter was not followed up, and I spent five nights with this smoke smell. - Room service keep knocking on the door repeatedly for cleaning in the morning.	0
- Room was clean and spacious - There are many restaurants nearby. - Next to the hotel there is a mall with many cafes and cinemas - Good hospitality from the reception staff - ATM machines in the hotel lobby - There is a bank in the basement of the hotel, as well as a gift corner and a flower shop	1
- The performance of the hotel employees is degraded by time. - Employees lack the basics of decency and professionality especially the reception staff. - The hotel rate per night is exaggerated, in general the prices of 5-star hotels in Cairo are not cheap.	0

Figure 4. English translation of sample records of our constructed dataset (RSAC).

4.2. Cleaning and Annotation

In this step, RSAC was cleaned by removing whitespace and punctuation, correcting spelling mistakes, and deleting replicate instances. Each review instance in RSAC was manually annotated as being either positive or negative.

4.3. Preprocessing

In the preprocessing stage, data were prepared for a text mining process. Preprocessing subtasks were applied to the cleaned reviews, as discussed below.

- Normalization: Normalization converts all possible conjugations of a specific word to its standard form. Our normalizer has been fed with a predefined set of rules to provide a standardized form of the collected Arabic reviews. For example, the normalizer removes unnecessary characters such as punctuation, numbers, non-Arabic characters, and special characters. It also removes repeated letters, usually used to express certain impressions, such as exaggeration or affirmation, and replaces this repetition with a single occurrence of the character.
- Stop Word Removal: Stop words are frequently used words that appear in a text but are not semantically related to the context in which they exist. These words can be removed from the text without affecting the classification task's performance.
- Stemming: Stemming is the process of returning a word to its stem. Stemming trims words by reducing all forms of the word (i.e., adjectives, adverbs, etc.) to its origin base. Many Arabic words have the same stem. Hence, all such words are replaced with one word. This technique has a large impact on improving the efficiency of text categorization.

4.4. Feature Extraction

At this stage, the text is ready for processing, which starts with the feature extraction step. During this step, the annotated data are converted into a feature vector, which in turn is fed into the machine learning classifiers for training. A specific combination of features leads to a specific class. Each piece of text is converted into a feature with a certain weight. The weight of the word (feature) changes as the document changes because it is calculated with respect to the document containing the word under consideration. In our model, Term Frequency-Inverse Document Frequency, which is known as TF-IDF and considered to be one of the most popular term-weighting schemes, was used to construct the feature vectors. Other examples of term-weighting schemes include Boolean and Term Frequency (TF). TF-IDF measures the importance of a word according to how many times this word appears in a document. Hence, TF-IDF is defined as the frequency with which a word appears in a text fragment divided by the frequency of the word within the whole dataset.

4.5. Training and Testing Classifiers

This step in the sentiment analysis process depends on the approach used for classification. Six supervised ML algorithms are used; they are Logistic Regression (LR) [45], Ridge Classifier (RC) [46], Support Vector Machine (SVC) [47], Decision Tree (DT) [48], Random Forest (RF), and Naive Bayes (NB) classifiers. During this step, the machine learning classifier was trained to classify each Arabic review in the dataset as carrying either a positive tone or a negative tone. During the training stage, the classifier algorithm learns from the labelled data, which were hotel reviews in our case. Consequently, during the testing stage, the classifier needed to have the ability to classify new, non-labelled hotel reviews. Each classifier was then evaluated by measuring the accuracy.

5. Experimental Results

5.1. Using Default Hyperparameters

First, the accuracy of each machine learning model in classifying the Arabic reviews was calculated using the default values of the hyperparameters. We used the default values specified by the Python scikit-learn library package [49]. Figure 5 presents a box plot of the accuracy for all classifiers, while Figure 6 contains a tabular representation of the accuracy of each classifier. Figure 7 shows how each classifier was calibrated during the global optimum search. Figure 8 gives the confusion matrix of each classifier for better performance evaluation. The figures show that the Support Vector Classifier (SVC) outperformed the other classifiers.

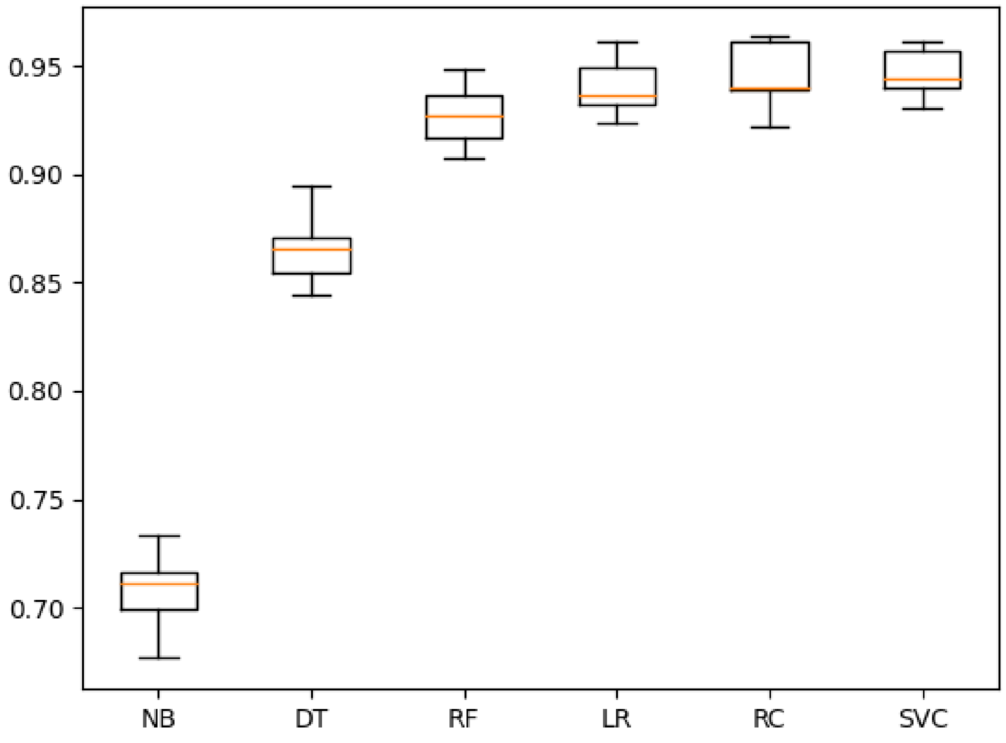


Figure 5. Box plot of all classifiers against the score (accuracy).

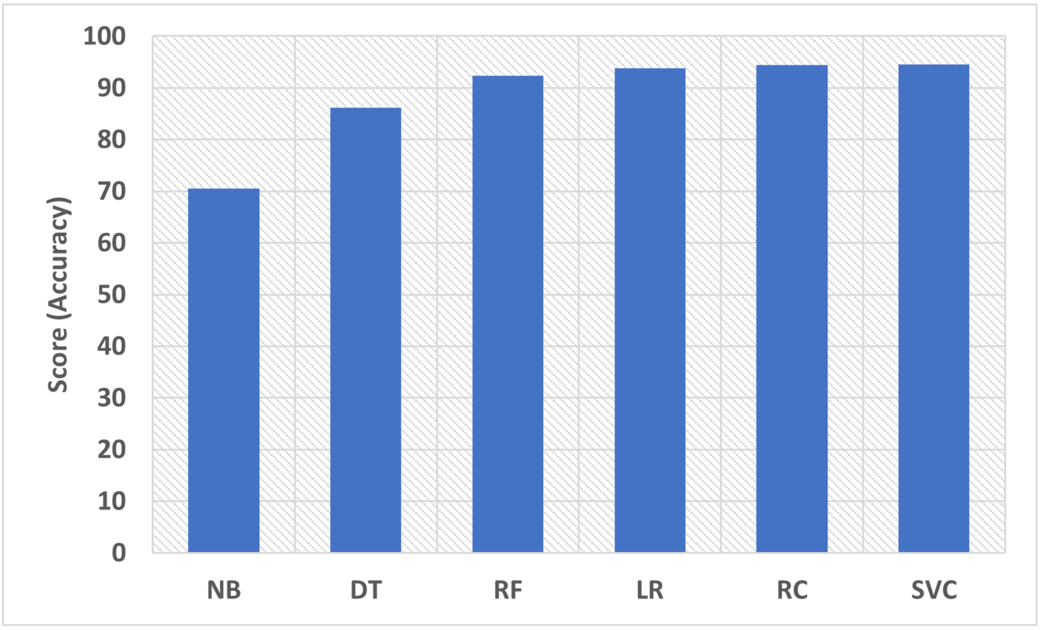


Figure 6. Default hyperparameters.

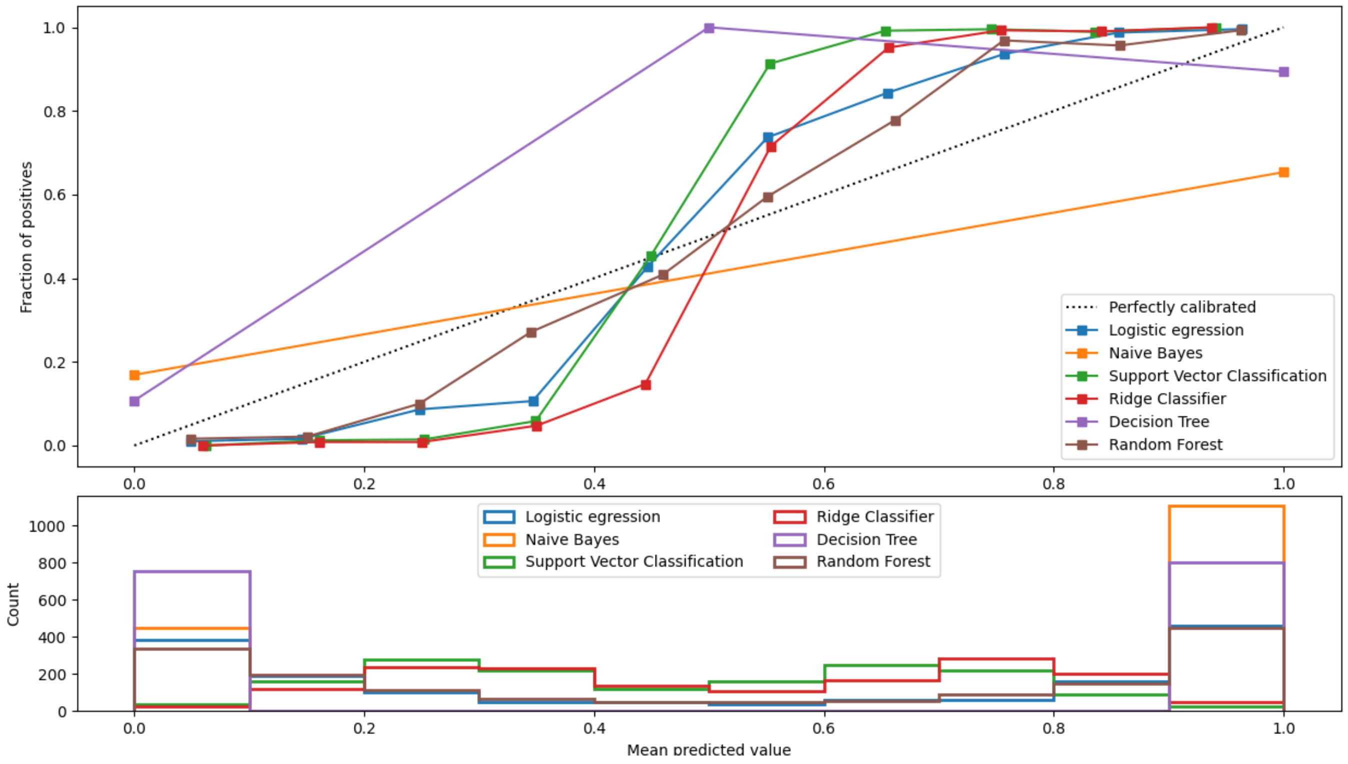


Figure 7. Calibration.

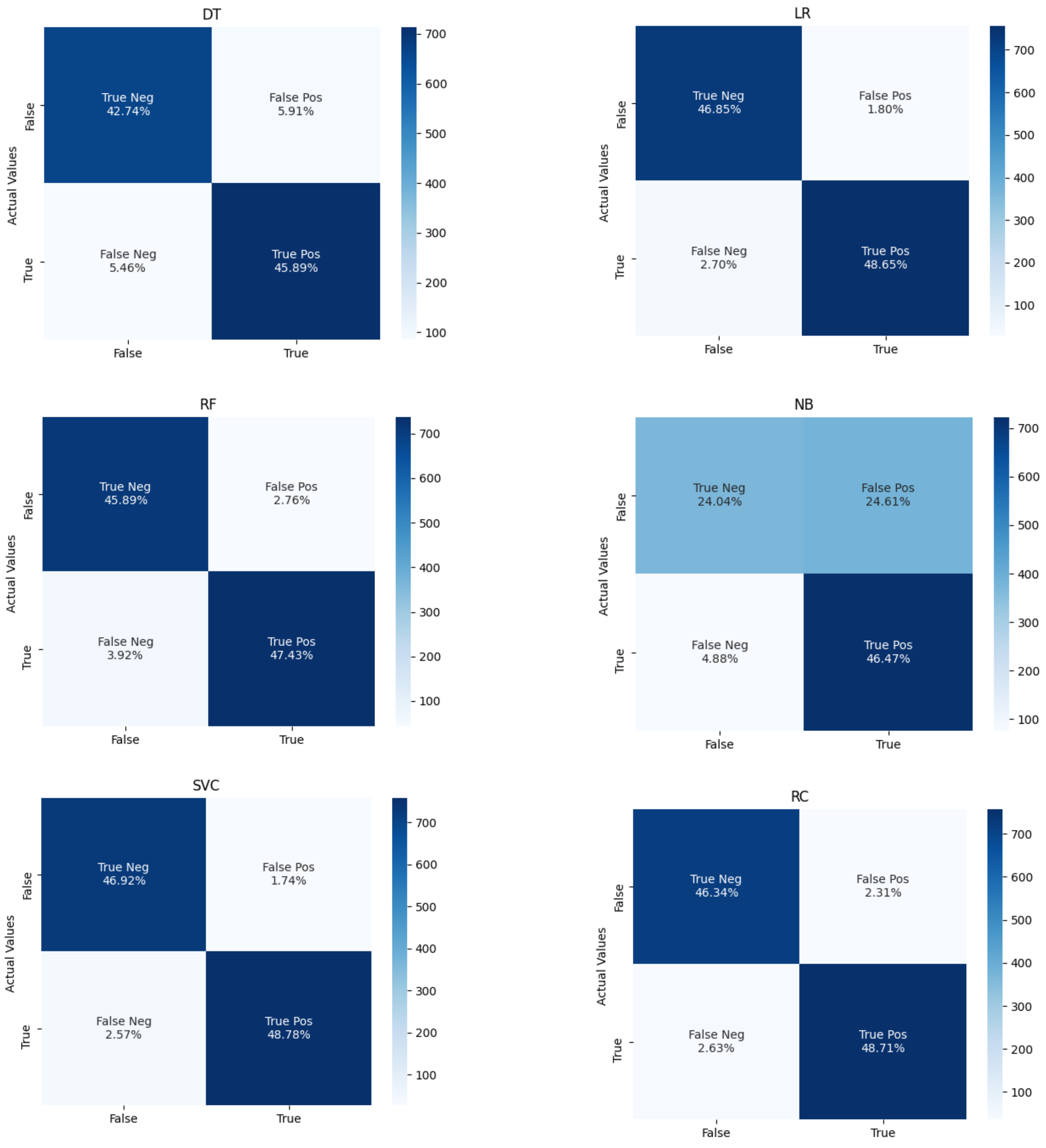


Figure 8. Confusion Matrix for Classifiers (DT, RF, SVC, LR, NB, and RC)

5.2. Using Hyperparameter Tuning Techniques

This section presents the results after the Grid Search, Random Search, Bayesian Optimization, Particle Swarm Optimization (PSO), and Genetic Algorithm (GA) strategies were used to tune the hyperparameters of each machine learning model, with the accuracy was calculated accordingly. Our system was implemented using Python running on a MacBook Air platform.

For Ridge Classifier (RC), Table 2 shows the five hyperparameter tuning techniques, the corresponding score (accuracy), and the set of the best hyperparameters that represent the best model configuration obtained by each technique and hence with the best accuracy.

Figure 9 depicts the score corresponding to each hyperparameter tuning strategy, as well as the accuracy obtained using the default values for the hyperparameters. Figure 10 shows the score for each parameter value in the search space for Ridge Classifier (RC).

It should be noted that, sometimes, default values work very well for some machine learning models and the specified dataset; however, this is not always the case when changing the model and/or the dataset used. For example, in terms of our dataset, setting the default value for the 'var_smoothing' hyperparameter in the Naïve Bayes model, which was 1×10^{-9}, gave an accuracy value of 70.5013; however, a higher accuracy value of 71.4402 was obtained when setting this hyperparameter to 1×10^{-9}. Similarly, setting the 'max_feature' hyperparameter to 'log2' in the Random Forest model yielded higher accuracy than setting it to its default value, which was 'auto'.

Another issue that should be noted is that some values of the hyperparameters should not be used together. For example, in the Logistic Regression classifier, the solver hyperparameter value 'sag' supports only 'l2' or 'no' penalties. Therefore, one should pay careful attention when setting the values. Of course, one can anticipate that Grid Search requires more iterations than the other two search algorithms as it rotates over every hyperparameter set. In the Ridge Classifier, for example, Grid Search tested 561 hyperparameter combination sets to find the best set, while, in both Random Search and Bayesian Optimization, we set the number of iterations to 100.

Similarly, Table 3 and Figures 11 and 12 show the results for Decision Tree (DT). Table 4 and Figures 13 and 14 show the results for Random Forest (RF). Table 5 and Figures 15 and 16 show the results for the Support Vector Classifier (SVC). Table 6 and Figures 17 and 18 show the results for Logistic Regression (LR). Finally, Table 7 and and Figure 19 show the results for Naive Bayes (NB).

Table 2. Hyperparameter tuning for Ridge Classifier.

HP Approach	Accuracy	Best Hyperparameters
Grid	95.2178	'alpha': 1.0, 'copy_X': True, 'fit_intercept': True, 'normalize': False, 'solver': 'auto', 'tol': 0.001
Random	95.1279	'tol': 0.001, 'solver': 'auto', 'normalize': False, 'fit_-intercept': True, 'copy_X': False, 'alpha': 0.9
Bayesian	95.1022	'alpha': 1.0, 'copy_X': True, 'fit_intercept': False, 'normalize': True, 'solver': 'lsqr', 'tol': 0.001
PSO	94.5494	'alpha': 0.807, 'tol': 0.0559
GA	94.5751	'alpha': 1, 'solver': 'auto', 'fit_intercept': 'False', 'normalize': 'True', 'copy_X': 'False', 'tol': 0.001

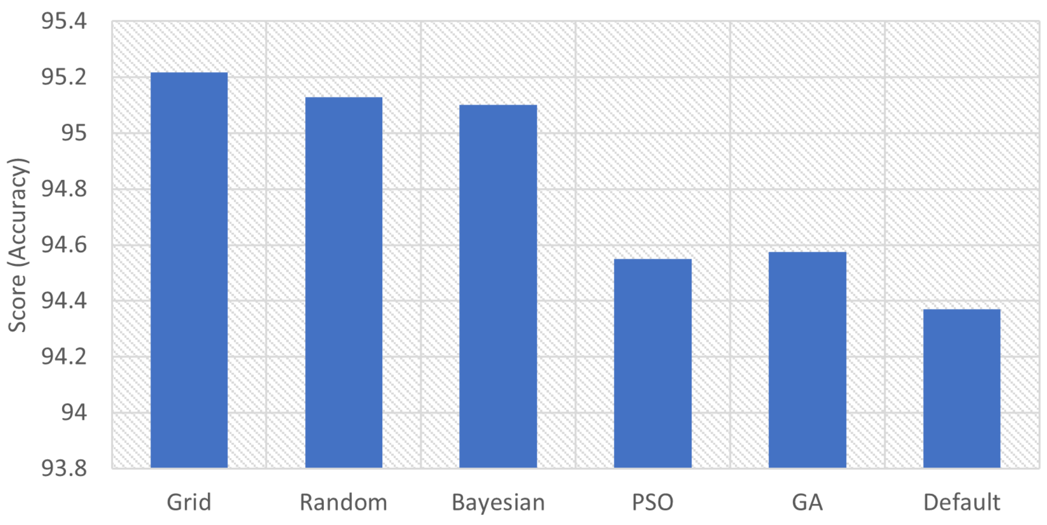


Figure 9. Hyperparameter tuning techniques versus score for Ridge Classifier (RC).

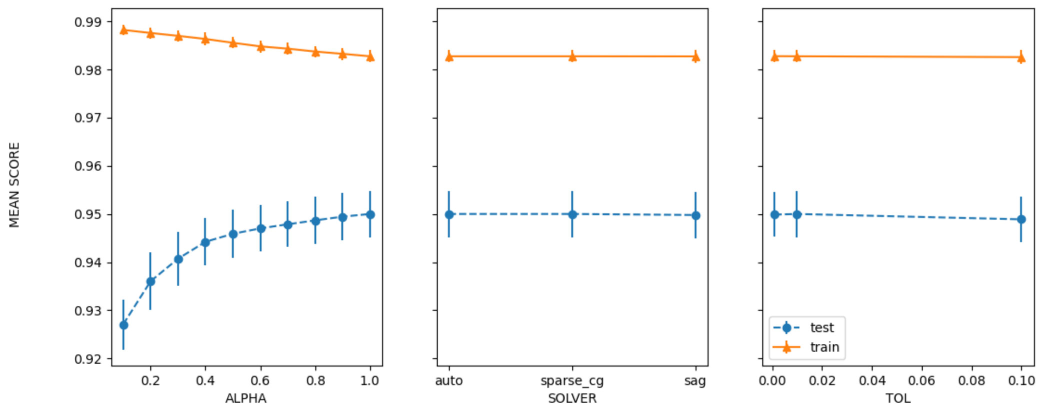


Figure 10. Score per parameter for Ridge Classifier (RC).

Table 3. Hyperparameter tuning for Decision Tree.

HP Approach	Accuracy	Best Hyperparameters
Grid	89.3773	'criterion': 'entropy', 'splitter': 'random'
Random	89.253	'splitter': 'random', 'criterion': 'entropy'
Bayesian	89.7459	'criterion': 'entropy', 'splitter': 'random'
PSO	86.4421	'splitter': 'random', 'criterion': 'entropy'
GA	87.5176	'criterion': 'entropy', 'splitter': 'random'

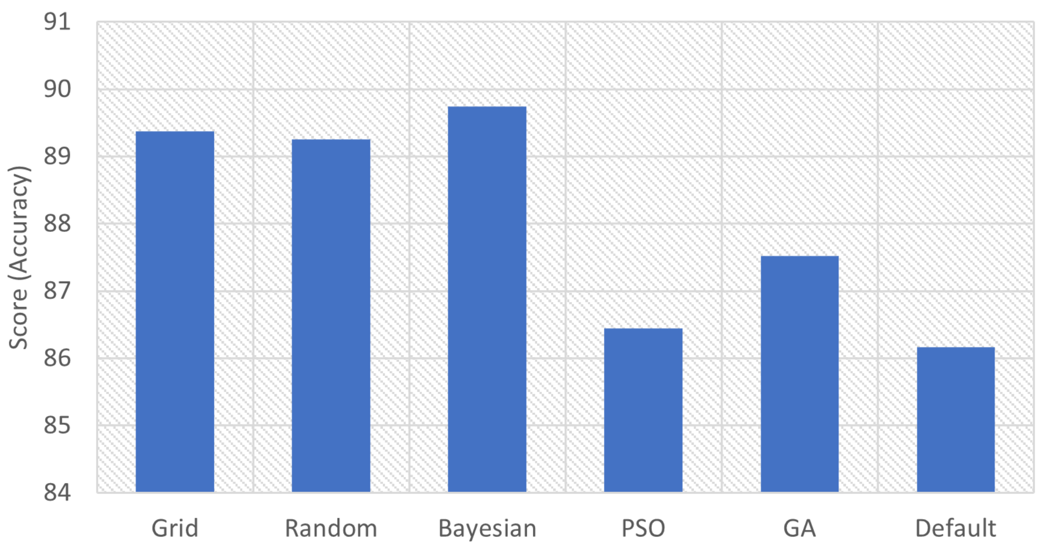


Figure 11. Hyperparameter tuning techniques versus score for Decision Tree (DT).

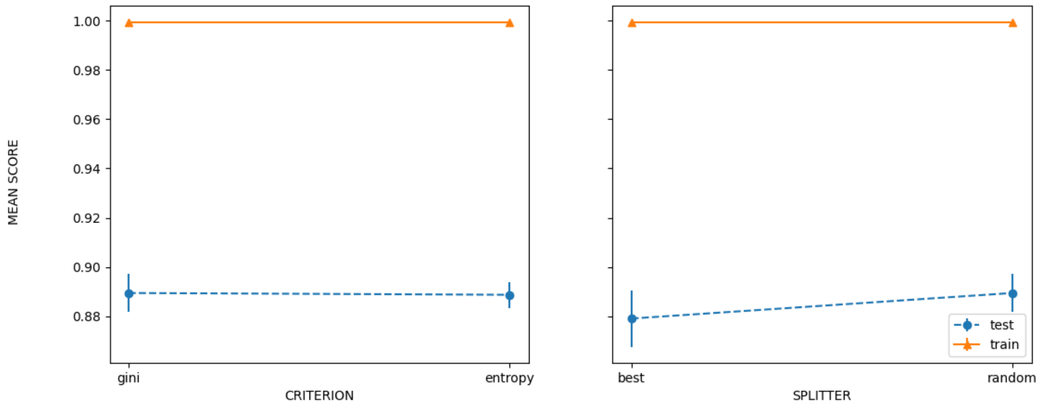


Figure 12. Score per parameter for Decision Tree (DT).

Table 4. Hyperparameter tuning for Random Forest.

HP Approach	Accuracy	Best Hyperparameters
Grid	94.335	'max_features': 'log2', 'n_estimators': 1000
Random	94.3907	'n_estimators': 1000, 'max_features': 'log2'
Bayesian	94.3907	'n_estimators': 1000, 'max_features': 'log2'
PSO	92.4869	'max_features': 'log2', 'n_estimators': 1000
GA	92.5183	'n_estimators': 'log2', 'max_features': 63, 'max_-depth': 48, 'min_samples_split': 6, 'min_samples_-leaf': 1, 'criterion': 'entropy'

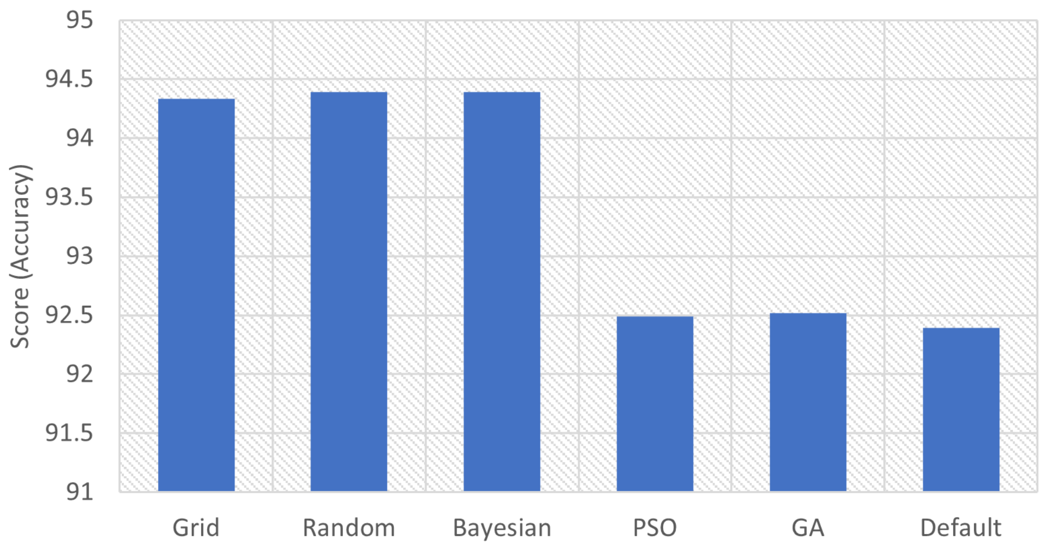


Figure 13. Hyperparameter tuning techniques versus score for Random Forest (RF).

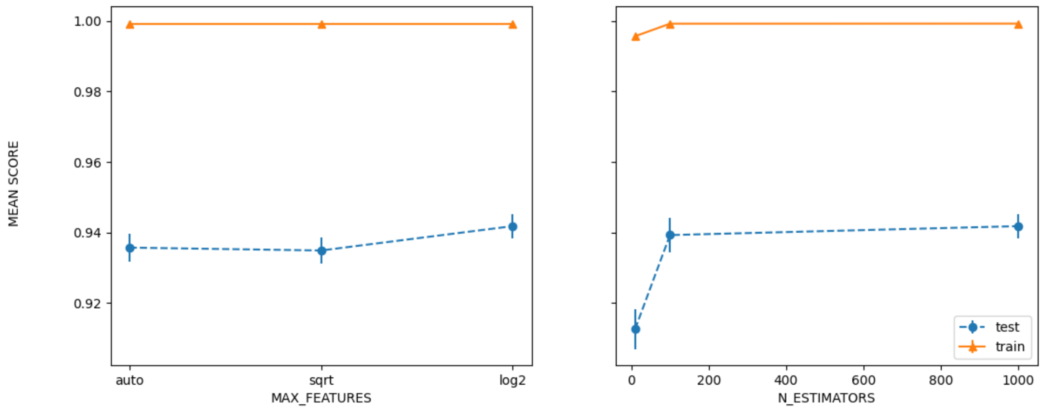


Figure 14. Score per parameter for Random Forest (RF).

Table 5. Hyperparameter tuning for Support Vector Classifier.

HP Approach	Accuracy	Best Hyperparameters
Grid	95.6206	'C': 1.0, 'gamma': 'scale', 'kernel': 'rbf'
Random	95.6206	'kernel': 'rbf', 'gamma': 'scale', 'C': 1.0
Bayesian	95.6208	'C': 1.0, 'gamma': 'scale', 'kernel': 'rbf'
PSO	94.6337	'C': 1.64, 'kernel': 'rbf', 'gamma': 'scale'
GA	94.7936	'C': 1.65, 'kernel': 'rbf', 'gamma': 'scale'

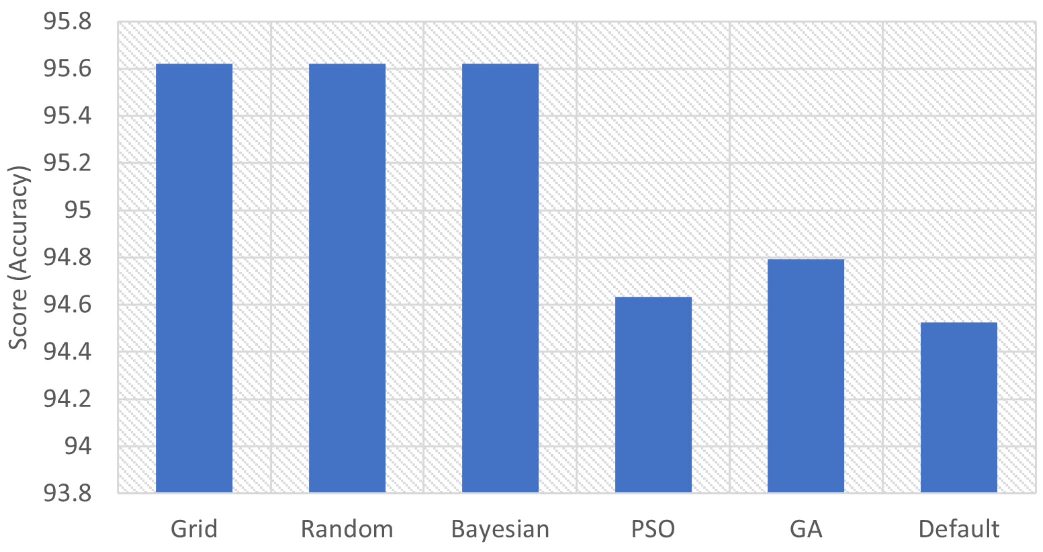


Figure 15. Hyperparameter tuning techniques versus score for Support Vector Classifier (SVC).

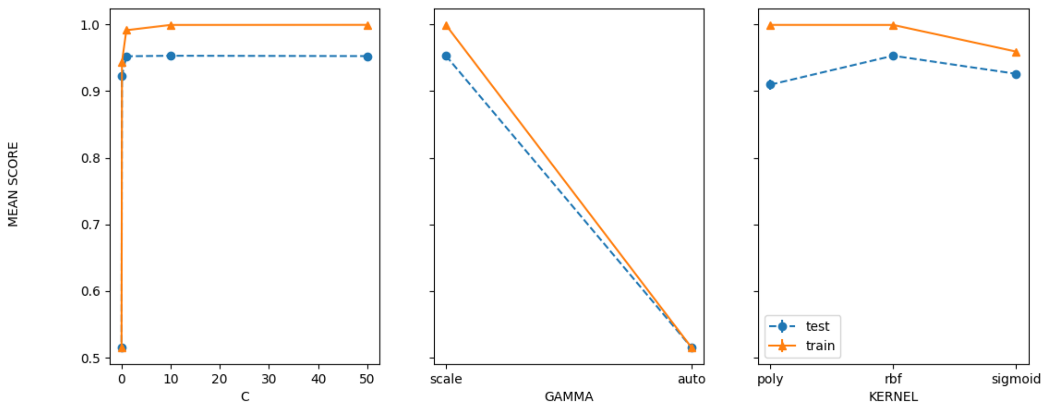


Figure 16. Score per parameter for Support Vector Classifier (SVC).

Table 6. Hyperparameter tuning for Logistic Regression.

HP Approach	Accuracy	Best Hyperparameters
Grid	95.0936	'C': 10, 'max_iter': 20, 'multi_class': 'ovr', 'penalty': 'l2', 'solver': 'saga'
Random	95.0936	'solver': 'saga', 'penalty': 'l2', 'multi_class': 'auto', 'max_iter': 100, 'C': 10
Bayesian	95.0938	'C': 10, 'max_iter': 20, 'multi_class': 'ovr', 'penalty': 'l2', 'solver': 'saga'
PSO	94.1123	'solver': 'liblinear', 'penalty': 'l2', 'multi_class': 'auto', 'max_iter': 30.0634765625
GA	94.2152	'solver': 'newton-cg', 'penalty': 'l2', 'multi_class': 'multinomial', 'max_iter': 20

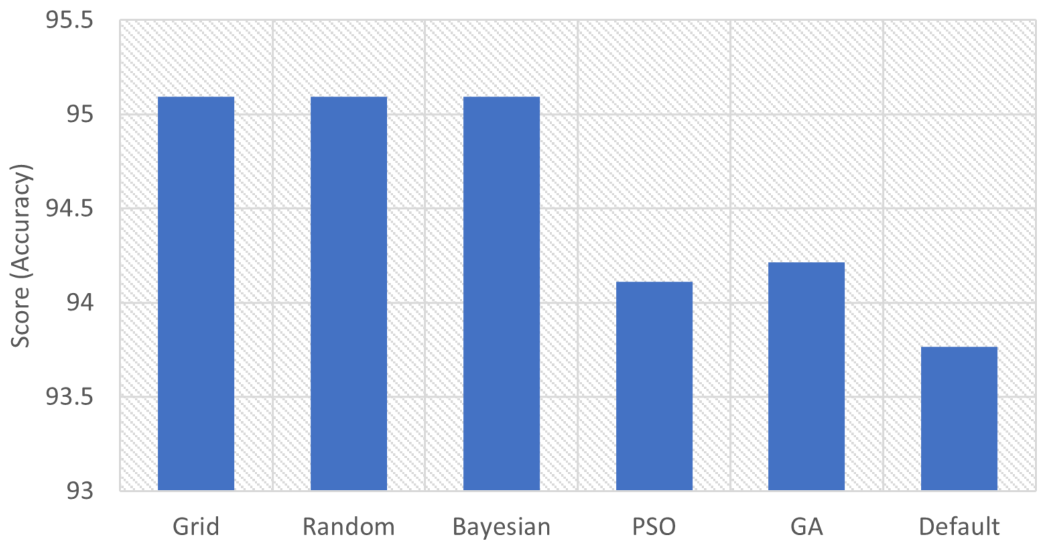


Figure 17. Hyperparameter tuning techniques versus score for Logistic Regression (LR).

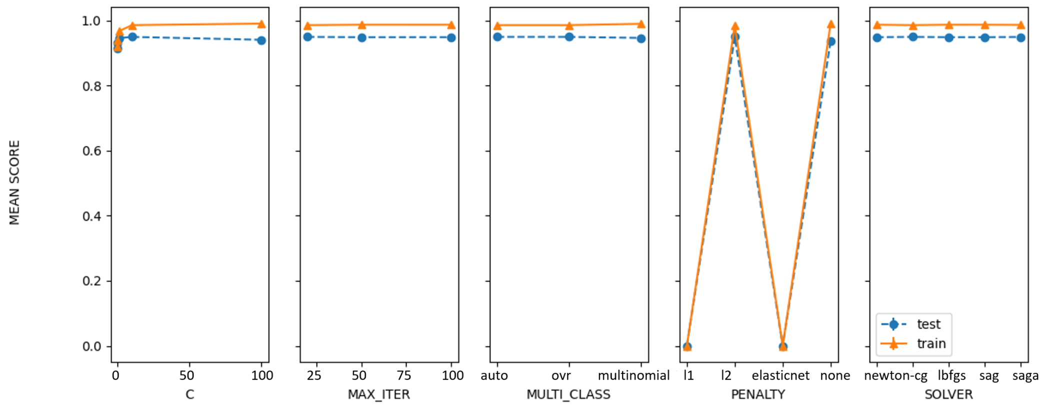


Figure 18. /hlScore per parameter for Logistic Regression (LG).

Table 7. Hyperparameter tuning for Naive Bayes.

HP Approach	Accuracy	Best Hyperparameters
Grid	71.4402	'var_smoothing': 1×10^{-7}
Random	71.4402	'var_smoothing': 1×10^{-7}
Bayesian	71.4402	'var_smoothing': 1×10^{-7}
PSO	78.3005	'var_smoothing': 9.965×10^{-5}
GA	78.3005	'var_smoothing': 0.0001

The RC algorithm clearly showed the highest accuracy obtained by Grid Search. The Grid Search, Random Search, and Bayesian Optimization hyperparameter tuning strategies obtained almost the same accuracy for LG, SVC, RF, and NB. Meanwhile, Bayesian Optimization showed noticeable superiority over the other hyperparameter tuning strategies for the DT algorithm. However, PSO and GA showed significant improvements for the NB accuracy. It is shown in the tables and figures that SVC gave the best accuracy both before and after hyperparameter tuning. Bayesian Optimization yielded slightly higher accuracy than Grid Search and Random Search when tuning SVC.

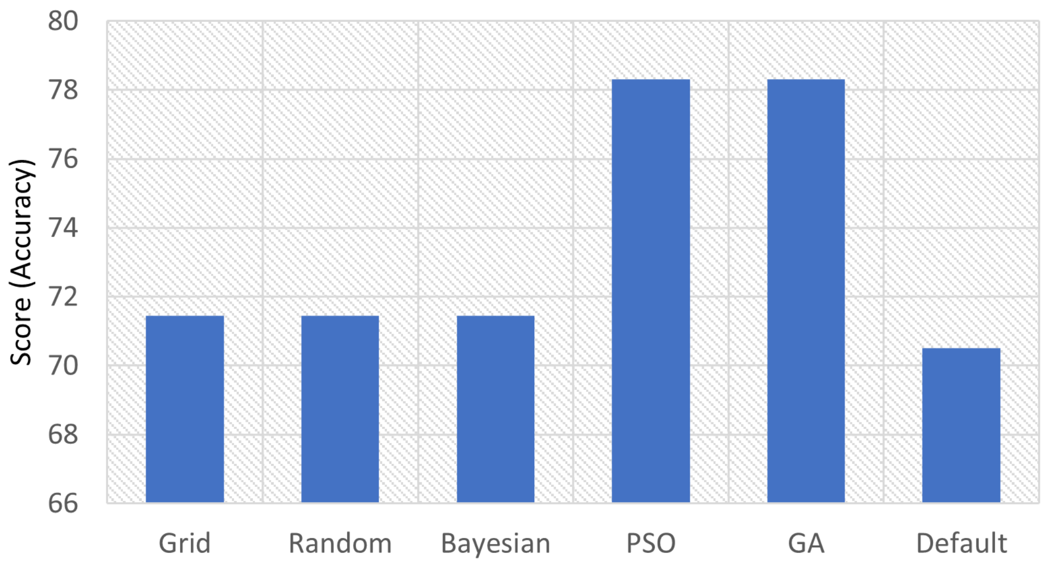


Figure 19. Hyperparameter tuning techniques versus score for (NB).

6. Conclusions

Sentiment analysis has become a very useful method to understand how a population feels about a given product, service, place, or even a public issue. In general, a machine learning approach is used when a certain text needs to be sentimentally analyzed. The classification accuracy of each machine learning model can be further optimized through hyperparameter tuning to achieve better accuracy than that obtained when using the model's default values of the hyperparameters. In this paper, five hyperparameter tuning approaches are presented: Grid Search, Random Search, Bayesian Optimization, Particle Swarm Optimization, and Genetic Algorithm. These approaches are used to perform the hyperparameter tuning of six machine learning algorithms in order to sentimentally analyze Arabic reviews. This paper is considered the first machine learning hyperparameter tuning study on an Arabic text. Our results show that the Support Vector Classifier offers the best accuracy both before and after hyperparameter tuning, with the highest score of 95.6208 obtained when using Bayesian Optimization. Meanwhile, Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) dramatically enhanced the score of the Naive Bayes classifier.

Author Contributions: Conceptualization, E.E. and A.M.Z.; Data curation, A.R.G.; Investigation, E.E. and A.M.Z.; Methodology, E.E. and A.M.Z.; Project administration, A.S.; Resources, E.E. and A.S.; Software, E.E. and A.R.G.; Supervision, E.E. and A.S.; Validation, A.S. and A.M.Z.; Visualization, E.E.; Writing—original draft, A.R.G. and E.E.; Writing—review & editing, E.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available at https://github.com/asooft/Sentiment-Analysis-Hotel-Reviews-Dataset (accessed on 7 October 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Buccafurri, F.; Lax, G.; Nicolazzo, S.; Nocera, A. Comparing Twitter and Facebook User Behavior. *Comput. Hum. Behav.* **2015**, *52*, 87–95. [CrossRef]
2. Madhyastha, H.V.; Radwan, A.A.; Omara, F.; Mahmoud, T.M.; Elgeldawi, E. Pinterest Attraction between Users and Spammers. *Int. J. Comput. Sci. Eng. Inf. Technol. Res.* **2014**, *4*, 63–72.

3. Elgeldawi, E.; Radwan, A.A.; Omara, F.; Mahmoud, T.M.; Madhyastha, H.V. Detection and Characterization of Fake Accounts on the Pinterest Social Networks. *Int. J. Comput. Netw. Wirel. Mob. Commun.* **2014**, *4*, 21–28.
4. Bacanli, S.; Cimen, F.; Elgeldawi, E.; Turgut, D. Placement of Package Delivery Center for UAVs with Machine Learning. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 7–11 December 2021.
5. de la Torre, R.; Corlu, C.G.; Faulin, J.; Onggo, B.S.; Juan, A.A. Simulation, Optimization, and Machine Learning in Sustainable Transportation Systems: Models and Applications. *Sustainability* **2021**, *13*, 1551. [CrossRef]
6. Sayed, A.A.; Elgeldawi, E.; Zaki, A.M.; Galal, A.R. Sentiment Analysis for Arabic Reviews using Machine Learning Classification Algorithms. In Proceedings of the 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE), Aswan, Egypt, 8–9 February 2020; pp. 56–63.
7. Sayed, A.; Abdallah, M.M.; Zaki, A.; Ahmed, A.A. Big Data analysis using a metaheuristic algorithm: Twitter as Case Study. In Proceedings of the 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE), Aswan, Egypt, 8–9 February 2020; pp. 20–26.
8. Girgis, M.R.; Elgeldawi, E.; Gamal, R.M. A Comparative Study of Various Deep Learning Architectures for 8-state Protein Secondary Structures Prediction. In *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2020*; Springer International Publishing: Cairo, Egypt, 2021; pp. 501–513.
9. Shekar, B.H.; Dagnew, G. Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data. In Proceedings of the 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), Gangtok, India, 25–28 February 2019; pp. 1–8.
10. Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
11. Liashchynskyi, P.; Liashchynskyi, P. Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS. *arXiv* **2019**, arXiv:abs/1912.06059.
12. Villalobos-Arias, L.; Quesada-López, C.; Guevara-Coto, J.; Martínez, A.; Jenkins, M. Evaluating Hyper-Parameter Tuning Using Random Search in Support Vector Machines for Software Effort Estimation. In Proceedings of the PROMISE'20: 16th International Conference on Predictive Models and Data Analytics in Software Engineering, Virtual Event, Association for Computing Machinery, New York, NY, USA, 8–9 November 2020; pp. 31–40.
13. Andonie, R.; Florea, A.C. Weighted Random Search for CNN Hyperparameter Optimization. *Int. J. Comput. Commun. Control* **2020**, *15*. [CrossRef]
14. Probst, P.; Boulesteix, A.; Bischl, B. Tunability: Importance of Hyperparameters of Machine Learning Algorithms. *J. Mach. Learn. Res.* **2019**, *20*, 53:1–53:32.
15. Syarif, I.; Prugel-Bennett, A.; Wills, G. SVM parameter optimization using grid search and genetic algorithm to improve classification performance. *Telecommun. Comput. Electron. Control* **2016**, *14*, 1502. [CrossRef]
16. Wicaksono, A.S.; Supianto, A.F. Hyper Parameter Optimization using Genetic Algorithm on Machine Learning Methods for Online News Popularity Prediction. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 263–267. [CrossRef]
17. Martínez-Cámara, E.; Barroso, N.R.; Moya, A.R.; Fernández, J.A.; Romero, E.; Herrera, F. Deep Learning Hyper-parameter Tuning for Sentiment Analysis in Twitter based on Evolutionary Algorithms. In Proceedings of the 2019 Federated Conference on Computer Science and Information Systems (FedCSIS), Leipzig, Germany, 1–4 September 2019; pp. 255–264.
18. Alayba, A.M.; Palade, V.; England, M.; Iqbal, R. Improving Sentiment Analysis in Arabic Using Word Representation. In Proceedings of the 2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR), London, UK, 12–14 March 2018; pp. 13–18.
19. Al-Twairesh, N.; Al-Negheimish, H. Surface and Deep Features Ensemble for Sentiment Analysis of Arabic Tweets. *IEEE Access* **2019**, *7*, 84122–84131. [CrossRef]
20. Duwairi, R.; Qarqaz, I. Arabic Sentiment Analysis Using Supervised Classification. In Proceedings of the 2014 International Conference on Future Internet of Things and Cloud, Barcelona, Spain, 27–29 August 2014; pp. 579–583.
21. Duwairi, R.; El-Orfali, M. A study of the effects of preprocessing strategies on sentiment analysis for Arabic text. *J. Inf. Sci.* **2014**, *40*, 501–513. [CrossRef]
22. Štrimaitis, R.; Stefanovič, P.; Ramanauskaitė, S.; Slotkienė, A. Financial Context News Sentiment Analysis for the Lithuanian Language. *Appl. Sci.* **2021**, *11*, 4443. [CrossRef]
23. Siji George, C.G.; Sumathi B. Genetic Algorithm Based Hybrid Model Of Convolutional Neural Network And Random Forest Classifier For Sentiment Classification. *Turk. J. Comput. Math. Educ.* **2021**, *12*, 3216–3223.
24. Pouransari, H.; Ghili, S. Deep learning for sentiment analysis of movie reviews. *CS224N Proj.* **2014**, 1–8.
25. Boudad, N.; Faizi, R.; Oulad Haj Thami, R.; Chiheb, R. Sentiment analysis in Arabic: A review of the literature. *Ain Shams Eng. J.* **2018**, *9*, 2479–2490. [CrossRef]
26. Rahab, H.; Zitouni, A.; Djoudi, M. SIAAC: Sentiment Polarity Identification on Arabic Algerian Newspaper Comments. In *Proceedings of the Computational Methods in Systems and Software Applied Computational (CoMeSySo 2017)*; Springer: Cham, Switzerland, 2017; pp. 139–149.
27. Lorenzo, P.R.; Nalepa, J.; Kawulok, M.; Ramos, L.; Ranilla, J. Particle swarm optimization for hyper-parameter selection in deep neural networks. In Proceedings of the Genetic and Evolutionary Computation Conference, Berlin, Germany, 15–19 July 2017.
28. Witt, C. *Worst-Case and Average-Case Approximations by Simple Randomized Search Heuristics*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 44–56.

29. Nguyen, V. Bayesian Optimization for Accelerating Hyper-Parameter Tuning. In Proceedings of the 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Sardinia, Italy, 3–5 June 2019; pp. 302–305.
30. Hensman, J.; Fusi, N.; Lawrence, N. Gaussian processes for big data. In Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence, Bellevue, WA, USA, 11–13 July 2013; pp. 282–290.
31. Man, K.; Tang, K.; Kwong, S. Genetic algorithms: Concepts and applications [in engineering design]. *IEEE Trans. Ind. Electron.* **1996**, *43*, 519–534. [CrossRef]
32. Friedrich, T.; Kötzing, T.; Krejca, M.S.; Sutton, A.M. The Compact Genetic Algorithm is Efficient Under Extreme Gaussian Noise. *IEEE Trans. Evol. Comput.* **2017**, *21*, 477–490. [CrossRef]
33. Itano, F.; de Abreu de Sousa, M.A.; Del-Moral-Hernandez, E. Extending MLP ANN hyper-parameters Optimization by using Genetic Algorithm. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
34. Srinivas, M.; Patnaik, L. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans. Syst. Man Cybern.* **1994**, *24*, 656–667. [CrossRef]
35. Smullen, D.; Gillett, J.; Heron, J.; Rahnamayan, S. Genetic algorithm with self-adaptive mutation controlled by chromosome similarity. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 504–511.
36. Yang, L.; Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* **2020**, *415*, 295–316. [CrossRef]
37. Lobo, F.; Goldberg, D.; Pelikan, M. Time Complexity of genetic algorithms on exponentially scaled problems. In Proceedings of the GECCO Genetic and Evolutionary Computation Conference, Las Vegas, NV, USA, 10–12 July 2000.
38. Shi, Y.; Eberhart, R.C. Parameter selection in particle swarm optimization. In *Evolutionary Programming VII*; Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E., Eds.; Springer: Berlin/Heidelberg, Germany, 1998; pp. 591–600.
39. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
40. Chuan, L.; Quanyuan, F. The Standard Particle Swarm Optimization Algorithm Convergence Analysis and Parameter Selection. In Proceedings of the Third International Conference on Natural Computation (ICNC 2007), Haikou, China, 24–27 August 2007; Volume 3, pp. 823–826.
41. Xiaojing, Y.; Qingju, J.; Xinke, L. Center Particle Swarm Optimization Algorithm. In Proceedings of the 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 15–17 March 2019; pp. 2084–2087.
42. Yan, X.H.; He, F.Z.; Chen, Y.L. A Novel Hardware/Software Partitioning Method Based on Position Disturbed Particle Swarm Optimization with Invasive Weed Optimization. *J. Comput. Sci. Technol.* **2017**, *32*, 340–355. [CrossRef]
43. Rauf, H.T.; Shoaib, U.; Lali, M.I.; Alhaisoni, M.; Irfan, M.N.; Khan, M.A. Particle Swarm Optimization With Probability Sequence for Global Optimization. *IEEE Access* **2020**, *8*, 110535–110549. [CrossRef]
44. RASC. Reviews Sentiment Analysis Corpus (RSAC). 2019. Available online: https://github.com/asooft/Sentiment-Analysis-Hotel-Reviews-Dataset (accessed on 7 October 2021).
45. Dreiseitl, S.; Ohno-Machado, L. Logistic regression and artificial neural network classification models: A methodology review. *J. Biomed. Inform.* **2002**, *35*, 352–359. [CrossRef]
46. Raschka, S. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *arXiv* **2018**, arXiv:abs/1811.12808.
47. Han, K.X.; Chien, W.; Chiu, C.C.; Cheng, Y.T. Application of Support Vector Machine (SVM) in the Sentiment Analysis of Twitter DataSet. *Appl. Sci.* **2020**, *10*, 1125. [CrossRef]
48. Gonçalves, P.; Araújo, M.; Benevenuto, F.; Cha, M. Comparing and combining sentiment analysis methods. *arXiv* **2013**, arXiv:abs/1406.0032.
49. Scikit Learn. Machine Learning in Python. 2020. Available online: https://scikit-learn.org/ (accessed on 7 October 2021).