Journal of Quantum Technologies and Informatics Research
JQTAIR

KUANTUM TEKNOLOJİLERİ VE ENFORMATİK ARAŞTIRMALARI
Volume / Cilt: 2, Issue / Sayı: 3, 2024, pp. 115-128
E - ISSN: 3023-4735
URL: https://journals.gen.tr/index.php/jqtair/
DOI: https://doi.org/10.70447/ktve.2573

Received / Geliş: 12.10.2024
Acccepted / Kabul: 25.10.2024

RESEARCH ARTICLE / ARAŞTIRMA MAKALESİ

Meta-Sezgisel Yöntemler ile Müzik Popülarite Sınıflandırması için Özellik Seçimi

Feature Selection with Meta-Heuristics for Music Popularity Classification

Abdurrahim Hüseyin Ezirmik[1] **İdiris Dağ**[2]

[1] Balıkesir Üniversitesi, Bilgisayar Mühendisliği Bölümü, Balıkesir, Türkiye, e-mail: huseyin.ezirmik@balikesir.edu.tr
[2] Eskişehir Osmangazi Üniversitesi, Bilgisayar Mühendisliği Bölümü, Eskişehir, Türkiye, e-mail: idag@ogu.edu.tr

Öz

Günümüzde multimedya içerik üretimi büyük bir hızla artmış, bu da değerli bilgilere erişimi zorlaştırmıştır. Anlamlı verilere ulaşımı kolaylaştırmak amacıyla veri madenciliği kritik bir hale gelmiştir ve bu süreçte önemli bir adım, veri boyutunun azaltılmasıdır. Özellik seçimi, veri kümesindeki ilgisiz, gürültülü veya eksik verilerin çıkarılmasıyla veri boyutunu küçülterek, veri analizinde kullanılan yöntemlerin daha hızlı ve verimli çalışmasını sağlar. Bu çalışmada, doğadan ilham alınan meta-sezgisel algoritmalar kullanılarak özellik seçimi gerçekleştirilmiştir. Belirlenen özellikler, makine öğrenimi algoritmaları ve yapay sinir ağları ile müzik verilerini şarkı popülerliğine göre sınıflandırmak için kullanılmıştır. Müzik veri seti üzerinde yapılan iyileştirmeler ile sınıflandırma başarımı %3.2 oranında artırılmış ve sonuç olarak %88 doğruluk elde edilmiştir. Kullanılan yöntemler karşılaştırmalı olarak sunulmuş ve elde edilen bulgular değerlendirilmiştir.

Anahtar kelimeler: Yapay sinir ağları, Metasezgisel algoritmalar, Özellik seçimi, Sınıflandırma

Abstract

In today's world, the rapid increase in multimedia content production has made accessing valuable information more challenging. Data mining has become critical to facilitate access to meaningful data, and an important step in this process is reducing the size of the data. Feature selection reduces the data size by eliminating irrelevant, noisy, or missing data from the dataset, allowing the methods used in data analysis to operate faster and more efficiently. In this study, feature selection was performed using nature-inspired metaheuristic algorithms. The selected features were used to classify music data by song popularity with machine learning algorithms and artificial neural networks. Improvements made on the dataset increased classification performance by 3.2%, achieving an accuracy of 88%. The methods used were presented comparatively, and the findings were evaluated.

Keywords: Artificial neural networks, Metaheuristic algorithms, Feature selection, Classification

Citation/Atıf: EZİRMİK, A. H. & DAĞ, İ. (2024). Meta-Sezgisel Yöntemler ile Müzik Popülarite Sınıflandırması için Özellik Seçimi. *Kuantum Teknolojileri ve Enformatik Araştırmaları*. 2(3): 115-128, DOI: 10.70447/ktve.2573

Corresponding Author/ Sorumlu Yazar:
Abdurrahim Hüseyin Ezirmik
E-mail: huseyin.ezirmik@balikesir.edu.tr

1. INTRODUCTION

In today's world, music is present in every aspect of human life. With the growing influence of the entertainment industry, the volume of music data produced is increasing over time, making it difficult for listeners to access all this data [1]. Without a good method for discovering music, a significant portion of the music produced may go unnoticed. As multimedia content expands and digital libraries continue to grow, information retrieval and access are becoming increasingly important. The aim for this work is to tackle the challenge of extracting valuable insights from audio datasets. By using feature selection with metaheuristic algorithms, the study attempts to improve computational efficiency and enhance the accuracy of music popularity prediction models.

Music data consists of several audio files. To analyze an audio file, it is first necessary to determine the type of information provided [2]. Much research has been conducted on music, speech, and sound. However, studies on songs are relatively fewer and still ongoing. Information about songs, such as lyrics, genre, and era, is shared online. Digital music serves as a source for information such as artist, track name, and year. Many operations can be performed using this information. Examples of this include track classification and song recommendation systems [3].

Recently, research on feature selection has increased for various reasons [4]. This is due to the development of new applications dealing with large amounts of data, such as data mining, medical data processing, and multimedia information retrieval. Feature selection is efficiently and widely used in classification systems [5]. Identifying distinctive features enhances recognition success. In classification using selected features, fewer operations are required, noisy and irrelevant features are removed from the original data, classification success is improved, and classification based on features becomes easier to interpret. Training time is reduced, fewer measurements are made, and less memory is used. These factors provide meaningful and easier classification.

This article is organized into several key sections that provide a comprehensive overview of the research. The Related Works section reviews existing literature on feature selection and categorization, identifying domains and advancements. The Metaheuristic Algorithms section outlines the specific algorithms used, explaining their principles and selection criteria. The Material and Methods section details the dataset and experimental procedures to ensure transparency and reproducibility. In the Results and Discussion section, findings are presented. Finally, the Conclusion summarizes the key contributions and suggests directions for future research in feature selection using metaheuristic methods.

2. RELATED WORKS

This section provides a thorough review of the contributions from numerous works on meta-heuristic algorithms and feature selection in many fields. The contributions in meta-heuristic algorithms cover a wide range of applications. For example, Tayarani et al. (2014) provided a state-of-the-art overview of meta-heuristic strategies for vehicle engine design [6], while methodologies outlining their advantages and disadvantages in dealing with reactive power planning difficulties [7] were discussed by Shaheen et al. (2018). The use of meta-heuristics in parallel computing scheduling, along with obstacles and future research prospects [8], was focused on by Memeti et al. (2018). Furthermore, the significance of meta-heuristic strategies in academic scheduling difficulties [9] was investigated by Teoh et al. (2015), and a comparative analysis of five techniques —ACO, PSO, GA, BA, and LCA— was conducted by Kalra and Singh (2015) to demonstrate their effectiveness in task scheduling for grid and cloud frameworks [10].

In the domain of feature selection, the application of feature selection in mobile malware detection was examined by Feizollah et al. (2015), providing a detailed overview of its significance [11]. Feature selection strategies used in sentiment analysis and their application in opinion mining [12] were briefly reviewed by Asghar et al. (2014). Finally, Saeys et al. (2007) concentrated on bioinformatics, offering a basic taxonomy of feature selection approaches and discussing their usefulness in both classic and developing bioinformatics applications [13].

Recent studies using metaheuristic methods in the field of music are generally based on music generation [14-16]. This work stands out by applying nature-inspired meta-heuristic algorithms specifically for feature selection in music data, focusing on song popularity prediction. Unlike previous studies that cover various fields, this research targets a unique challenge in the music domain.

3. METAHEURISTIC ALGORITHMS

The term metaheuristic refers to an algorithmic framework that provides guidance or strategies for the development of different heuristic optimization algorithms, independent of the problem at hand [17]. This term is also used to describe the specific application of a heuristic optimization algorithm to a given problem. Technically, the term metaheuristic was first introduced by Fred Glover in 1986, combining the Greek word "meta" with "heuristic," meaning higher-level heuristic [18].

A higher-level heuristic approach involves methods that perform a probabilistic yet conscious search in the solution space [19]. These methods generate new solutions by starting from the set of solutions created at each step. Thus, by searching near the points that are closest to the optimum in the search space, they aim to reach the optimal solution while avoiding local optima selection [20].

Metaheuristic methods are techniques that direct the search process. They aim to explore the search space efficiently to obtain the best or near-optimal results. These methods span from local search techniques to complex learning processes. Typically, they offer an approximate solution that is non-deterministic. They are not limited to solving a specific problem but provide solutions to different types of problems. They are designed in a way that prevents convergence to local solutions in the search space.

For metaheuristic algorithms to produce good results, it is essential that the fundamental concepts of the method are well adapted to the problem. There are many types of metaheuristic algorithms [21]. Algorithms inspired by nature imitate the behavior patterns of living beings in natural environments. Some are population-based, while others are individual-based. Their objective functions can be either static or dynamic. They can be classified based on whether or not they use neighborhood structures or memory. These methods are seen as the nature-inspired extensions of classical heuristic algorithms. The variety of algorithms has increased as they have been developed for optimization purposes based on various scientific fields.

3.1. Ant Colony Algorithm

Ant Colony Optimization (ACO) is a metaheuristic method developed to solve difficult optimization problems. It is inspired by the behavior of real ants, which use the pheromone hormone they secrete in their natural environment as a means of communication [22]. Similar to the biological example, this optimization method is based on the indirect communication established through pheromone trails in an artificial ant colony. Pheromones serve as distributed, numerical information that ants use to probabilistically generate solutions to a problem, reflecting their search experience and adapting during the execution of the algorithm.

Real ants perform complex tasks, such as finding the shortest path to food sources and transporting the obtained food back to the nest, through collective behavior. The ant colony algorithm mimics the principle of how an ant colony can find the shortest path between two points using a simple communication mechanism [23]. There is a path that ants with poor vision can travel between the nest and the food source. During their trips, ants leave a chemical trail (pheromone) on the ground. Pheromone is a volatile substance with a distinct smell. This trail plays a role in guiding other ants toward the target point [24]. As the amount of pheromone on a certain path increases, the probability of ants choosing that path also increases.

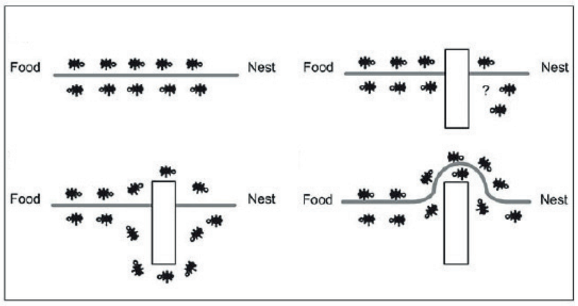

Figure 1. An ant colony searching for the best path between food and the nest [25]

Additionally, this chemical substance has a diminishing effect over time as it evaporates, and the amount of this substance secreted by an ant depends on the amount of food in the environment. As shown in Figure 1, when faced with an obstacle, each ant has an equal probability of choosing either the left or right path. Since the left trail is shorter than the right one and requires less travel time, the ant will leave a higher amount of pheromone. The more ants use a path, the more pheromone accumulates on that path. Thus, the shortest path is eventually determined.

3.2. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is another population-based, stochastic metaheuristic optimization method inspired by swarm intelligence [26]. It imitates the behaviors exhibited by natural organisms, such as birds and fish, when searching for a place with sufficient food. In these swarms, coordinated behaviors in Figure 2 using local movements emerge without any central control. PSO has been successfully designed to solve continuous optimization problems.

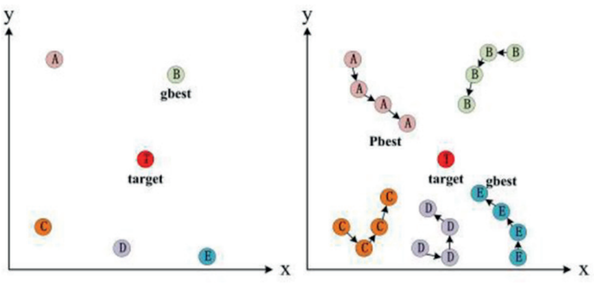

Figure-2. Geometrical illustration for PSO algorithm [27]

3.3. Simulated Annealing

In computer science, particularly in the field of optimization, one of the algorithms used is inspired by the annealing process applied during iron processing, which involves heating the iron and then allowing it to cool. The goal of the Simulated Annealing (SA) algorithm is to achieve overall improvement for any given problem [28]. In other words, it aims to find the global minimum or maximum value of any function or measure.

3.4. Genetic Algorithms

Genetic Algorithms (GA) are a very popular class of evolutionary algorithms. A GA typically applies a crossover operator to two solutions, which plays a significant role, and a mutation operator that randomly alters individual content to increase diversity [29]. GA use probabilistic selection, which is proportional selection. The replacement that determines selection is generational, meaning parents are systematically replaced by their offspring. The crossover operator is based on n-point or uniform crossover, while the mutation operator alters bits [30]. A fixed probability is applied to the mutation operator.

The main search components for designing an evolutionary algorithm are: gene representation, population initialization, objective (fitness) function, selection, mutation and crossover for reproduction, generational replacement, and stopping criteria in Figure 3.

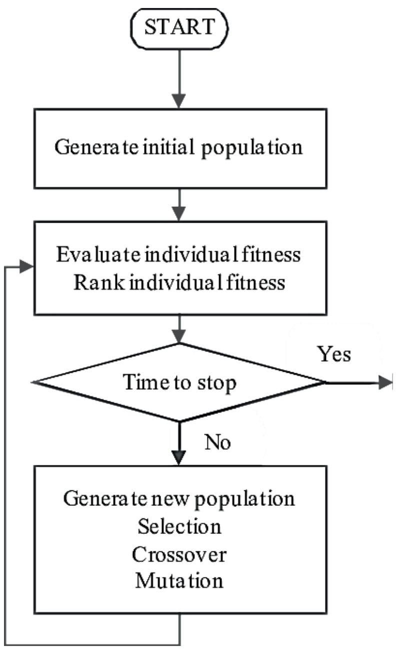

Figure 3. Genetic algorithm flowchart [31]

4. MATERIAL AND METHODS

4.1. Music Dataset

The open library used in this research, created by the Echo Nest company in collaboration with LabROSA, a laboratory for intelligent machine listening, aims to gather data on approximately one million contemporary and popular songs under the name Million Song Dataset [32]. The data includes standard information about songs, such as the artist's name, album, and year of release. Additionally, it contains more advanced information, such as the length of the song, the number of musical bars, and the fade-out duration.

The Million Song Dataset will be analyzed for classification purposes. The original dataset is 280GB in size and consists of one million tracks. In this study, a subset of 10.000 pieces has been used to reduce computational costs. The reduced dataset consists of 22 features, including artist name, title, duration, and tempo.

In the original dataset, there is a song popularity feature. The "song_hotttnesss" tag represents the song popularity. However, this feature does not include values for approximately 4.500 songs, which is almost half of the total number of data entries. Therefore, the BillBoard Top 100 list is used to determine popularity. If a song reaches the BillBoard Top 100 at least once, it is defined as a hit song. Of the 10.000 songs in the dataset, 1.192 songs are classified as popular songs. Popular tracks are represented by 1, while non-popular ones are represented by 0.

As shown in Figures 4 and 5, artist similarities and song loudness are related to the song's popularity. The similarity between artists shows a positive correlation, as expected. However, surprisingly, song loudness exhibits a negative correlation with popularity. It was anticipated that more popular songs would be louder, but this appears to be the opposite, as the overall average loudness of songs tends to be slightly higher. In Figure 5, loudness is plotted on the x-axis, while popularity is plotted on the y-axis. The reason more popular songs seem to be quieter could be due to the presence of exceptionally loud songs in the data, which lowers the overall popularity average of the songs.

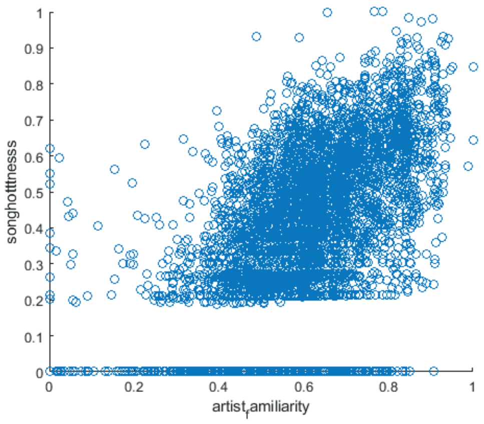

Figure 4. Artist similarity-popularity distribution graph

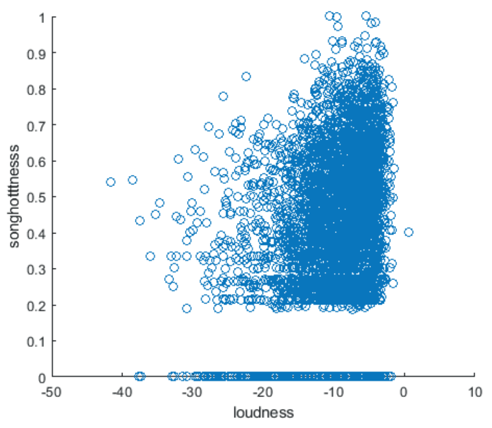

Figure 5. Loudness-popularity distribution graph

Data preprocessing has been performed on the music dataset to achieve more efficient results. Textual data, such as song titles and artist locations, has been removed from the feature set, allowing for the extraction of numerical data in Table 3 that can be computed using algorithms. The data numbers entered in the rows are given as count. Cells with no value in the dataset are entered as NaN. The means, standard deviations and minimum-maximum values of these data are presented in the table.

The data under the "year" tag, which stores the release year of pieces, shows an imbalance due to the high frequency of pieces with unknown release years, where a value of 0 was entered. To prevent this distribution from causing deviations, the year feature column was removed from the processed data. After data cleaning, a dataset with 16 numeric features and a target column (popularity class) was obtained. In its final state, the data consists of 16x10001 inputs and 1x100001 target. Based on this information, the comma-separated values (.csv) file on which feature selection will be performed was converted into formats (.mat, .arff) suitable for the software used.

4.2. Feature Selection

Feature selection is the process of determining a subset that represents a dataset and isolating the variables that best express this data [33]. This process selects the best k features from n features by scanning them according to the algorithm being used, thereby reducing the number of features and providing various benefits in problem-solving. Feature selection reduces the size of the attribute set, allowing the algorithm used for data analysis to run faster. It improves data quality by isolating noisy or incomplete data and prevents complexity by simplifying the dataset [34]. Additionally, it provides storage savings by reducing the data size.

Feature selection processes have been carried out using algorithms designed with MATLAB 2018a software, which enables effective and fast mathematical computations in areas such as statistics, optimization, and numerical analysis [35]. To measure classification performance, desired features were extracted using ACO, PSO, SA, and GA metaheuristic methods.

The Ant Colony Optimization algorithm was used to reduce the dimensionality by performing feature extraction on data containing inputs and targets. In the application phase, the desired number of features was specified as 4. The problem was defined, and a fitness function was created. The parameter values to be used in ant colony optimization were entered in Table 1.

Table 1. ACO parameter values

Parameter	Value
Number of ants (population)	50
Initial pheromone value	1
Pheromone trail information (alpha)	1
Heuristic parameter (beta)	1
Evaporation rate	0.05

In the feature selection process, a matrix is first created to store the tour, cost, and output values of 50 ants. In the loop, which will run for the number of iterations determined at the start, the feature where the tour will begin is randomly selected, starting with the first ant. Then, the probability of this ant moving to other features is calculated. Positions are subjected to roulette wheel selection based on pheromone values, and the next feature the ant will visit is determined [36]. Once the ant completes its tour, the obtained values are sent to the fitness function, and the cost value is calculated. The order of features in the ant's tour, the cost value of the tour, and the structure containing the desired number of features are recorded as output. Afterward, the pheromone values left by the ant are updated. The loop moves to the next ant, and the same processes are repeated. For each iteration, pheromone is evaporated by 0.05, and the best cost value found is recorded.

When feature selection is performed using Particle Swarm Optimization, the population size consists of 50 particles in total. The values of the Fi (phi) constants are taken as 2.05, and their sum is passed through the chi-square method to equal the inertia weight. The damping ratio of this weight is 0.99. The individual and social learning coefficients (c1, c2) are found by multiplying the Fi constants with the value obtained from the chi-square formula. Velocity limits are set, and the minimum limit is adjusted to be the negative of the maximum limit.

In Simulated Annealing, initial positions are determined using a random permutation function, which returns a random vector composed of the entered features. The positions found are evaluated using the fitness function, and the best solution is assigned. A list containing as many elements as the number of iterations is created to store the best cost values. The initial temperature is set to 10. In the main loop, which runs for the total number of iterations of the Simulated Annealing process, there is an inner loop that runs for the number of sub-iterations. A new solution is generated using the neighbor generation function. In this function, the swap, return, and join rates are applied as 0.2, 0.5, and 0.3, respectively. These rates are subjected to roulette wheel selection, and based on the result, one of these operations is applied to the initially determined tour. After determining the new tour in this way, the cost value for this tour is calculated. If the found value is better, the solution is updated. Once the sub-iteration is completed, the best cost value is retained in the loop for the main iteration, and the temperature is updated based on the cooling rate.

When using a Genetic Algorithm for feature selection, the initial phase was initiated after defining the parameters found in Table 2. An empty structure was defined to hold the positions and costs of the individuals. An array containing as many elements as the population size of individuals was created. In the loop, which runs for the number of individuals, genes consisting of bits were assigned to the elements using a discrete uniform distribution. The individuals were evaluated using the fitness function, and the obtained values were recorded. All individuals in

Table 3. Basic statistics on numerical data

Feature	Count	Mean	Std. dev.	Min.	Max.
artist_familiarity	9997	0.565	0.16	0	1
artist_hotttnesss	10001	0.386	0.144	0	1.083
artist_latitude	3742	37.157	15.599	-41.281	69.651
artist_longitude	3742	-63.934	50.508	-162.44	174.77
duration	10001	238.512	114.133	1.044	1819.8
end_of_fade_in	10001	0.759	1.868	0	43.119
key	10001	5.276	3.554	0	11
key_confidence	10001	0.45	0.275	0	1
loudness	10001	-10.485	5.4	-51.643	0.566
mode	10001	0.691	0.462	0	1
mode_confidence	10001	0.478	0.191	0	1
song_hotttnesss	5649	0.343	0.247	0	1
start_of_fade_out	10001	229.98	112.191	1.044	1813.4
tempo	10001	122.921	35.186	0	262.83
time_signature	10001	3.565	1.266	0	7
time_signature_confidence	10001	0.51	0.373	0	1
year	10001	935	996.651	0	2010

the population were ranked according to their fitness. The best solution was recorded, and an array was created to hold the cost values.

Table 2. Parameters used in genetic algorithm

nPop=50	Population size
pc=0.7	Crossover percentage
nc=2*round(pc*nPop/2)	Number of offspring
pm=0.3	Mutation percentage
nm=round(pm*nPop)	Number of mutants
mu=0.1	Mutation rate
beta=8	Selection pressure

4.3. Classification Algorithms

The data classification problem has countless applications across a wide range of data mining fields [37]. This is because the problem attempts to learn the relationship between a set of feature variables and a target variable of interest. In practice, since many issues can be expressed as relationships between features and target variables, this model provides broad applicability. The concept of classification simply involves distributing data among various classes defined on a dataset. Classification algorithms learn this distribution pattern from the given training set and then attempt to classify correctly when test data arrives, for which the class is not specified.

Classification algorithms typically consist of two stages: the training phase, in which a model is constructed from training examples, and the testing phase, which is used to assign labels to unlabeled test examples. The values that specify these classes on the dataset are referred to as label names and are used during both training and testing to determine the class of the data. In some cases, the training phase may be entirely skipped, and classification is performed directly based on the relationship between training examples and test examples. Instance-based methods, such as nearest neighbor classifiers, are an example of such a scenario [38]. Even in these cases, a preprocessing stage may be carried out to ensure efficiency during the testing phase.

The output of a classification algorithm can be presented in one of two ways. In one, a direct label is found for the test example. In the other, a numerical score is returned for each class label and the combination with the test example. This numerical score can be converted into a separate label by selecting the class with the highest score for a test example. The advantage of this scoring system is that it allows for the comparison of the tendency of different test examples to belong to a certain importance class and enables their ranking when necessary.

To test the classification performance of the obtained features, various classifiers were used. The open-source Weka 3 machine learning software, which contains many different clustering and classification algorithms and enables data mining applications, was utilized.

4.3.1. Naive Bayes Classifier

The Naive Bayes classifier (NB) is a classification technique based on Bayes' Theorem, named after the English mathematician Thomas Bayes, which assumes independence among predictions [39]. In simple terms, it assumes that the presence of a particular feature in a class is independent of the presence of any other feature. Even if these features are dependent on each other or on the presence of other features, all of these features contribute to the probabilities independently [40]. The Naive Bayes model is easy to construct and is particularly useful for very large datasets. Along with its simplicity, it is known to perform better than even highly complex classification methods.

$$P(c|x) = \frac{P(C|X)P(c)}{P(x)} \quad (1)$$

$$P(c|x) = P(x_1|c) \times \ldots \times P(x_n|c) \times P(c) \quad (2)$$

- $P(c|x)$ The asymptotic probability distribution of a given predictor for a class
- $P(c)$ The prior probability distribution for a parameter or parameter vector
- $P(x|c)$ The likelihood function of a given class
- $P(x)$ The prior probability of the predictor

4.3.2 K-Nearest Neighbors

The K-Nearest Neighbor (KNN) algorithms are a classification algorithm proposed by T. M. Cover and P. E. Hart in 1967 [41]. It takes multiple labeled points and uses them to learn how to label other points. To label a new point, it looks at the k nearest labeled points to that new point and uses the labels of these neighbors. Therefore, the label that appears most frequently among the neighbors becomes the label for the new point.

When determining the neighborhood condition, the distance of a point from other points is considered [42]. Typically, three different distance functions are used for distance calculations:

- Euclidean Distance

$$d(x,y)=\sqrt{\sum_{i=1}^{k}(x_i-y_i)^2} \tag{3}$$

- Manhattan Distance

$$d(x,y)=\sum_{i=1}^{k}|x_i-y_i| \tag{4}$$

- Minkowski Distance

$$d(x,y)=\left(\sum_{i=1}^{k}(|x_i-y_i|)^q\right)^{\frac{1}{q}} \tag{5}$$

IBk (Instance Based Learner), a derivative of KNN, is a pattern recognition method that classifies test data based on the nearest training examples in the feature space [43]. This algorithm performs classification based on the class of the k nearest neighbors. In the IBk algorithm, the classification of a vector is done using known class vectors. In this study, the value of k indicating the neighborhood was set to 3. The linear search algorithm was used in the neighbor finding process [44].

4.3.3. Decision Tree

A decision tree creates classification or regression models in the form of a tree structure [45]. As it divides a dataset into smaller subsets, a corresponding decision tree is developed step by step. The result is a tree with decision nodes and leaf nodes. A decision node has two or more branches, while a leaf node represents a classification or decision. The top decision node in the tree corresponds to the best prediction and is called the root node. Decision trees can handle both categorical and numerical data. The J48 decision tree, based on the C4.5 algorithm, was used in this study [46]. J48 utilizes information gain for attribute selection and includes pruning techniques to mitigate overfitting, ensuring robust model performance.

4.3.4. Support Vector Machines

It is possible to separate labeled groups located in a plane by drawing a boundary between them. The location where this decision boundary is drawn should be the point that is farthest from the members of the groups. Support Vector Machines (SVM) determine these boundaries. This method was developed in 1995 by Vladimir Vapnik, Bernhard Boser, and Isabelle Guyon [47]. Today, SVM is used in various classification problems, ranging from face recognition systems to text categorization. SMO (Sequential Minimal Optimization) is an algorithm that operates by using John Platt's sequential minimal optimization algorithm to train a support vector classifier [48].

4.3.5. Artificial Neural Networks

Artificial neural networks are developed by drawing inspiration from the way nerve system cells function in living organisms [49]. Their aim is to impart the learning ability of a living brain to computers. A neural network consists of units (neurons) organized in layers that transform an input vector into an output. Each unit receives an input, applies a typically nonlinear function to it, and then passes the output to the next layer. Networks are generally defined to feed forward [50]. A unit feeds its output to all units in the next layer but does not transmit feedback to the previous layer. Weights are applied to the signals that pass from one unit to another, and these weights are adjusted during the training phase to adapt the artificial neural network to the specific problem at hand [51].

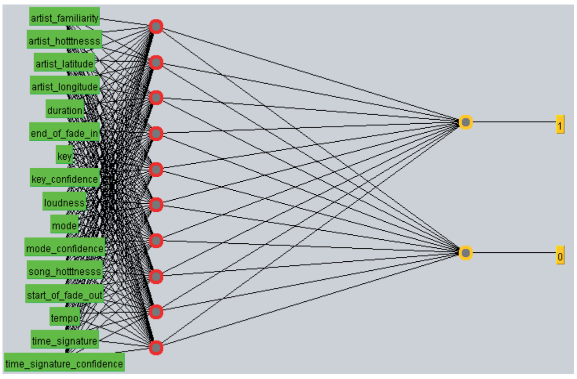

Figure 6. A Multi-layer Perceptron with 10 hidden layers

The most commonly used model of artificial neural networks is the Multi-layer Perceptron (MLP) [52]. Multilayer artificial neurons fundamentally consist of three parts in Figure 6. The input layer does not perform any information processing; it simply receives information and transmits it to the hidden layers. Each element in the input layer is connected to all processing units in the hidden layer. In this part, the information from the input layer is processed. A single hidden layer can solve many problems, but multiple hidden layers can also be utilized. The number of hidden layers varies depending on the type of problem. The output layer processes the information coming from the hidden layer and transmits it to the outside.

$$f(x) = b + \sum_{i=1}^{n}(x_i\, w_i) \tag{6}$$

Here:

b = bias, x = neuron input, w = weights, n = number of inputs from the previous layer, i = counter from 0 to n.

In artificial neural networks, the values of the inputs are multiplied by the weights of the connections, and the results are combined to find the net input of the network [53]. Once the net inputs are passed through an activation function, the net output of the network is obtained.

During the classification process, 10-fold cross-validation was used. Cross-validation divides the dataset into 10 random subsets, using 9 for testing and 1 for training. This process is repeated 10 times until all permutations are used for training and testing.

4.4 Performance Metrics

The most commonly used method for measuring classification performance is accuracy. It is calculated by dividing the number of correctly classified instances by the total number of instances.

$$A = \frac{(TP+TN)}{(TP+FP+FN+TN)} \tag{7}$$

TP (True Positive): This is used when the value in the test data matches the class predicted by the model. The classification is correct.

FN (False Negative): This occurs when the value in the test data is different from the class produced by the model, where a positive instance is incorrectly classified as negative. The classification is incorrect.

FP (False Positive): This occurs when the actual value is negative but is incorrectly classified as positive.

TN (True Negative): This is when the value is correctly classified as negative when it is actually negative.

Precision is the ratio of the number of true positives (TP) predicted as positive to the total number of instances predicted as class 1.

$$P = \frac{TP}{(TP+FP)} \tag{8}$$

The metric that indicates how many of all positive classes were correctly predicted is defined as sensitivity.

$$R = \frac{TP}{(TP+FN)} \tag{9}$$

In cases where sensitivity and precision metrics are not sufficient to produce meaningful results, it is necessary to evaluate these two metrics together. Therefore, the F-measure has been defined. This metric is the harmonic mean of precision and sensitivity.

$$F = \frac{2RP}{(R+P)} \tag{10}$$

5. RESULTS AND DISCUSSION

Through studies conducted using metaheuristic methods, feature selection was performed on the data in the music dataset to enhance classification performance. The most important features for the classification process were identified. A current dataset containing features that meaningfully represent the data was created, and the data was classified according to track popularity using various classifiers. It was observed that the artist_hotttnesss label, which represents artist popularity, was a significant feature in all the algorithms used. This indicates that an artist's recognition is an important factor in a song's popularity. Furthermore, it was concluded that the features tempo and loudness, which indicate the track's tempo and volume, are significant factors in determining song popularity. After 100 iterations, the lowest error value in Table 4 was achieved using the ant colony algorithm.

The results obtained from classification using different classifiers were compared after feature selection was performed using metaheuristic methods, as well as without feature selection. Initially, the best performance in the classification of raw data was achieved with the SMO algorithm among five different classifiers. According to Table 5, it was observed that the success rate increased after feature selection using metaheuristic methods compared to the raw dataset. In classifications using fewer features, the success rates of decision trees, Naive Bayes, kNN, and artificial neural networks increased compared to the previous state of the data, while there was no change in the success rate for classifications performed with support vector machines. Based on these results, the highest success obtained through feature selection was achieved with the J48 algorithm, which is a decision tree algorithm. The highest performance increase compared to the raw dataset was 3.23%, which was obtained using features selected by a genetic algorithm and the Naive Bayes classifier. The error rates obtained with this classifier are presented in Table 6.

Table 4. Algorithm comparison results according to iteration number

Method	Iteration	Feature Set	Min. Error
ACO	20	artist_hotttnesss, loudness, mode, mode_confidence	0.10172
	50	artist_hotttnesss, loudness, tempo, time_signature_confidence	0.10153
	100	artist_hotttnesss, loudness, tempo, start_of_fade_out	0.10109
PSO	20	artist_hotttnesss, loudness, tempo, key	0.10122
	50	artist_hotttnesss, artist_familiarity, tempo, start_of_fade_out	0.10113
	100	artist_hotttnesss, duration, tempo, end_of_fade_in	0.10119
SA	20	artist_hotttnesss, loudness, duration, time_signature	0.10227
	50	artist_hotttnesss, loudness, key, mode	0.10193
	100	artist_hotttnesss, loudness, key, mode_confidence	0.10149
GA	20	artist_hotttnesss	0.10541
	50	artist_hotttnesss	0.10517
	100	artist_hotttnesss	0.10507

Table 5. Classification results

Classifier	Raw (%)	ACO (%)	PSO (%)	SA (%)	GA (%)
IBk	84.53	84.96	85.04	84.83	85.48
NB	84.77	87.86	87.75	87.97	88.00
MLP	87.74	88.08	88.02	88.04	88.07
J48	88.02	88.08	88.08	88.08	88.08
SMO	88.08	88.08	88.08	88.08	88.08

Table 6. Naive Bayes classifier error rates

Metric	ACO	PSO	SA	GA
TP Rate	0.879	0.878	0.880	0.880
FP Rate	0.862	0.858	0.867	0.871
Precision	0.820	0.817	0.823	0.823
Recall	0.879	0.878	0.880	0.880
F-Measure	0.829	0.829	0.828	0.827

6. CONCLUSION

This study addresses the significant issue of improving machine learning classification performance through the use of efficient feature selection techniques. Finding the most relevant features in multimedia datasets is crucial, especially when it comes to music data analysis. In order to improve the accuracy and efficiency of predictive models, this research attempts to enhance the feature selection process by investigating different nature-inspired metaheuristic algorithms.

The results of this study show that using four different metaheuristic techniques greatly improves the ability to extract relevant features from the dataset. It has been demonstrated through comparative evaluations that these metaheuristic techniques not only improve classification accuracy but also make model training more effective. Furthermore, the use of artificial neural networks to assess the appropriateness of particular features highlights the potential for synergy between machine learning classification methods and reliable feature selection processes.

In future studies, the speed performance of the currently applied algorithms can be tested. It appears feasible to make improvements in terms of time and cost with different parameter values. Furthermore, it is believed that new studies could be conducted to measure the success of other metaheuristic algorithms in feature selection. The use of hybrid versions of heuristic optimization methods is also recommended for this purpose.

Acknowledgments

This article has been prepared based on master's thesis, *Meta-Heuristic Methods for Feature Selection and Categorization on Music Data* [54]. I would like to express my sincere gratitude to my thesis advisor Prof. İdiris Dağ for his exceptional guidance, insightful feedback, and unwavering support throughout this research journey.

REFERENCES

[1] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE,* vol. 96, no. 4, pp. 668-696, 2008.

[2] A. Lerch, *An introduction to audio content analysis: Music Information Retrieval tasks and applications*. John Wiley & Sons, 2022.

[3] P. Knees and M. Schedl, "A survey of music similarity and recommendation from music context data," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM),* vol. 10, no. 1, pp. 1-21, 2013.

[4] T. Dokeroglu, A. Deniz, and H. E. Kiziloz, "A comprehensive survey on recent metaheuristics for feature selection," *Neurocomputing,* vol. 494, pp. 269-296, 2022.

[5] R.-C. Chen, C. Dewi, S.-W. Huang, and R. E. Caraka, "Selecting critical features for data classification based on machine learning methods," *Journal of Big Data,* vol. 7, no. 1, p. 52, 2020.

[6] M.-H. Tayarani-N, X. Yao, and H. Xu, "Meta-heuristic algorithms in car engine design: A literature survey," *IEEE Transactions on Evolutionary Computation,* vol. 19, no. 5, pp. 609-629, 2014.

[7] A. M. Shaheen, S. R. Spea, S. M. Farrag, and M. A. Abido, "A review of meta-heuristic algo-

rithms for reactive power planning problem," *Ain Shams Engineering Journal,* vol. 9, no. 2, pp. 215-231, 2018.

[8] S. Memeti, S. Pllana, A. Binotto, J. Kołodziej, and I. Brandic, "A review of machine learning and meta-heuristic methods for scheduling parallel computing systems," in *Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications*, 2018, pp. 1-6.

[9] C. K. Teoh, A. Wibowo, and M. S. Ngadiman, "Review of state of the art for metaheuristic techniques in Academic Scheduling Problems," *Artificial Intelligence Review,* vol. 44, pp. 1-21, 2015.

[10] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian informatics journal,* vol. 16, no. 3, pp. 275-295, 2015.

[11] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, "A review on feature selection in mobile malware detection," *Digital investigation,* vol. 13, pp. 22-37, 2015.

[12] M. Z. Asghar, A. Khan, S. Ahmad, and F. M. Kundi, "A review of feature extraction in sentiment analysis," *Journal of Basic and Applied Scientific Research,* vol. 4, no. 3, pp. 181-186, 2014.

[13] Y. Saeys, I. Inza, and P. Larranaga, "A review of feature selection techniques in bioinformatics," *bioinformatics,* vol. 23, no. 19, pp. 2507-2517, 2007.

[14] L. Dong, "Using deep learning and genetic algorithms for melody generation and optimization in music," *Soft Computing,* vol. 27, no. 22, pp. 17419-17433, 2023.

[15] U. Boryczka, M. Boryczka, and P. Chmielarski, "ACO and generative art–artificial music," *Procedia Computer Science,* vol. 225, pp. 2624-2633, 2023.

[16] Q. Zhu, A. Shankar, and C. Maple, "Grey wolf optimizer based deep learning mechanism for music composition with data analysis," *Applied Soft Computing,* vol. 153, p. 111294, 2024.

[17] J. A. Parejo, A. Ruiz-Cortés, S. Lozano, and P. Fernandez, "Metaheuristic optimization frameworks: a survey and benchmarking," *Soft Computing,* vol. 16, pp. 527-561, 2012.

[18] F. Glover and K. Sörensen, "Metaheuristics," *Scholarpedia,* vol. 10, no. 4, p. 6532, 2015.

[19] G. Keren and K. H. Teigen, "Yet another look at the heuristics and biases approach," *Blackwell handbook of judgment and decision making,* pp. 89-109, 2004.

[20] J. D. Knowles, R. A. Watson, and D. W. Corne, "Reducing local optima in single-objective problems by multi-objectivization," in *International conference on evolutionary multi-criterion optimization*, 2001, pp. 269-283: Springer.

[21] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Metaheuristic algorithms: A comprehensive review," *Computational intelligence for multimedia big data on the cloud with engineering applications,* pp. 185-231, 2018.

[22] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine,* vol. 1, no. 4, pp. 28-39, 2006.

[23] M. Dorigo and T. Stützle, *Ant colony optimization: overview and recent advances*. Springer, 2019.

[24] E. Talbi, "Metaheuristics: From Design to Implementation," *John Wiley & Sons google schola,* vol. 2, pp. 268-308, 2009.

[25] E. Flórez, W. Gómez, and L. Bautista, "An ant colony optimization algorithm for job shop scheduling problem," *arXiv preprint arXiv:1309.5110,* 2013.

[26] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, 1995, vol. 4, pp. 1942-1948: ieee.

[27] H. Liu, X. Wang, and M. Li, "External force estimation for robotic manipulator base on particle swarm optimization," *International Journal of Advanced Robotic Systems,* vol. 18, no. 6, p. 17298814211063744, 2021.

[28] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical science,* vol. 8, no. 1, pp. 10-15, 1993.

[29] A. Hassanat, K. Almohammadi, E. a. Alkafaween, E. Abunawas, A. Hammouri, and V. S. Prasath, "Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach," *Information,* vol. 10, no. 12, p. 390, 2019.

[30] K. A. De Jong and W. M. Spears, "A formal analysis of the role of multi-point crossover in genetic algorithms," *Annals of mathematics and Artificial intelligence,* vol. 5, pp. 1-26, 1992.

[31] V. Kachitvichyanukul, "Comparison of three evolutionary algorithms: GA, PSO, and DE,"

Industrial Engineering and Management Systems, vol. 11, no. 3, pp. 215-223, 2012.

[32] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," 2011.

[33] H. Liu and H. Motoda, *Feature extraction, construction and selection: A data mining perspective*. Springer Science & Business Media, 1998.

[34] W. Fan, F. Geerts, and X. Jia, "Improving data quality: Consistency and accuracy," 2007: ACM.

[35] G. Ciaburro, *MATLAB for machine learning*. Packt Publishing Ltd, 2017.

[36] A. Lipowski and D. Lipowska, "Roulette-wheel selection via stochastic acceptance," *Physica A: Statistical Mechanics and its Applications,* vol. 391, no. 6, pp. 2193-2196, 2012.

[37] G. Kesavaraj and S. Sukumaran, "A study on classification techniques in data mining," in *2013 fourth international conference on computing, communications and networking technologies (ICCCNT)*, 2013, pp. 1-7: IEEE.

[38] B. Martin, "Instance-based learning: nearest neighbour with generalisation," 1995.

[39] J. Joyce, "Bayes' theorem," 2003.

[40] M.-L. Zhang, J. M. Peña, and V. Robles, "Feature selection for multi-label naive Bayes classification," *Information Sciences,* vol. 179, no. 19, pp. 3218-3229, 2009.

[41] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory,* vol. 13, no. 1, pp. 21-27, 1967.

[42] L. E. Peterson, "K-nearest neighbor," *Scholarpedia,* vol. 4, no. 2, p. 1883, 2009.

[43] R. Ade and P. Deshmukh, "Instance-based vs batch-based incremental learning approach for students classification," *International Journal of Computer Applications,* vol. 106, no. 3, 2014.

[44] M. R. Abbasifard, B. Ghahremani, and H. Naderi, "A survey on nearest neighbor search methods," *International Journal of Computer Applications,* vol. 95, no. 25, 2014.

[45] W. Y. Loh, "Classification and regression trees," *Wiley interdisciplinary reviews: data mining and knowledge discovery,* vol. 1, no. 1, pp. 14-23, 2011.

[46] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria, "Decision tree analysis on j48 algorithm for data mining," *Proceedings of international journal of advanced research in computer science and software engineering,* vol. 3, no. 6, 2013.

[47] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[48] J. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," 1998.

[49] A. Abraham, "Artificial neural networks," *Handbook of measuring system design,* 2005.

[50] T. L. Fine, *Feedforward neural network methodology*. Springer Science & Business Media, 2006.

[51] K. Jadav and M. Panchal, "Optimizing weights of artificial neural networks using genetic algorithms," *Int J Adv Res Comput Sci Electron Eng,* vol. 1, no. 10, pp. 47-51, 2012.

[52] M. Riedmiller and A. Lernen, "Multi layer perceptron," *Machine Learning Lab Special Lecture, University of Freiburg,* vol. 24, 2014.

[53] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems,* vol. 28, 2015.

[54] A. H. Ezirmik, "Meta-sezgisel yöntemler ile müzik verisi üzerinde özellik seçimi ve kategorizasyon," *Eskişehir Osmangazi Üniversitesi, Fen Bilimleri Enstitüsü,* 2020.