Optimization of CART Models Using Metaheuristics for Predicting Peach Firmness

Ivanovski, Tomislav; Gulić, Marko; Matetić, Maja

applied sciences

MDPI

Article

Optimization of CART Models Using Metaheuristics for Predicting Peach Firmness

Tomislav Ivanovski [1,2,*], Marko Gulić [2,3] and Maja Matetić [1,2]

[1] Faculty of Informatics and Digital Technologies, University of Rijeka, 51000 Rijeka, Croatia; majam@uniri.hr
[2] Center for Artificial Intelligence and Cybersecurity, University of Rijeka, 51000 Rijeka, Croatia; marko.gulic@pfri.uniri.hr
[3] Faculty of Maritime Studies, University of Rijeka, 51000 Rijeka, Croatia
* Correspondence: tomislav.ivanovski@student.uniri.hr; Tel.: +385-91-2800-220

Abstract: The current advancements in the field of machine learning can have an important application in agriculture and global food security. Machine learning has considerable potential in establishing knowledge-based farming systems. One of the main challenges of data-driven agriculture is to minimize food waste and establish more sustainable farming systems. The prediction of the right harvest time is one of the ways to obtain the mentioned goals. This paper describes multiple machine learning algorithms that are used to predict peach firmness. By accurately predicting peach firmness based on various peach measurement data, a more precise harvest time can be obtained. The evaluation of nature-inspired metaheuristic optimization algorithms in enhancing machine learning model accuracy is the primary objective of this paper. The possibility of improving the peach firmness prediction accuracy of regression tree models using various metaheuristic optimization techniques implemented in *GA* and *metaheuristicOpt* R packages is studied. The RMSE on test data of the default regression tree model is 1.722285, while the regression tree model optimized using the gray wolf optimization algorithm scored the lowest RMSE of 1.570924. The obtained results show that it is possible to improve the peach firmness prediction accuracy of the regression tree model by 8.8% using the described method.

Keywords: machine learning; nature-inspired metaheuristics; regression trees; peach firmness prediction; food quality

Citation: Ivanovski, T.; Gulić, M.; Matetić, M. Optimization of CART Models Using Metaheuristics for Predicting Peach Firmness. *Appl. Sci.* **2024**, *14*, 8539. https://doi.org/10.3390/app14188539

Academic Editors: Petru Alexandru Vlaicu and Basarab Matei

Received: 20 July 2024
Revised: 14 September 2024
Accepted: 17 September 2024
Published: 23 September 2024

1. Introduction

The food production process of the farming systems can be improved using the vast amounts of measured data in combination with machine learning algorithms. With the application of machine learning in the field of agriculture, more sustainable food production can be obtained, increasing global food security and reducing food waste. Determining the correct fruit harvest time is critical for reducing food waste along the food supply chain [1]. Fruit quality evaluation is important to accurately predict the harvest date and extend the fruit shelf life.

To accurately determine the quality of the fruit, physical and chemical quality detection methods exist, such as the detection of firmness, soluble solids concentration, and titratable acidity [2]. Manual fruit quality evaluation methods such as appearance screening by the operator are being replaced by automated evaluation systems based on machine vision, image processing technology, and other emerging technologies, which could improve the efficiency of the farming systems [2]. Some of the new technologies for predicting and identifying properties of various fruit are backpropagation neural networks [2], convolutional neural networks [3], discriminant analysis [4], and simple linear regression [5].

Previous work [1] investigated the possibilities of optimizing regression trees using nature-inspired metaheuristic algorithms. The work described in this paper extends [1]

by introducing new machine learning models and metaheuristics used for regression tree optimization. The desire to solve any problem in an optimal way is very common; hence, optimization techniques are extensively applied in the fields of science and engineering. Optimization techniques aim to find the best solution to a particular problem out of a set of possible options. Optimization problems usually consist of an objective function and constraints with either discrete or continuous variables [6]. With the larger search space, the complexity of the optimization problems increases, where real-world optimization problems are often NP (non-deterministic polynomial-time) problems that can be solved with non-deterministic algorithms, which are computationally very demanding [6]. In the fields of computer science and mathematical optimization, a metaheuristic is defined as a higher-level process that can satisfactorily solve an optimization problem, especially when given limited computing power or incomplete information [7]. Since metaheuristics make few assumptions about the optimization problem being solved, they can be applied to a variety of different optimization problems. Numerous metaheuristics employ stochastic optimization, whereby the solution obtained relies on a set of randomly generated variables [7]. The advantage of using metaheuristics is that they provide fast and cost-effective solutions to complex problems. However, they cannot ensure global optimality, unlike numerical optimization algorithms. By balancing between the exploration and exploitation phases, the metaheuristic algorithms aim to find the optimal solution to a given problem. Due to their stochastic character and the No Free Lunch (NFL) theorem, the optimal performance on one problem does not guarantee similar results on another problem by using the same metaheuristic algorithm [6]. This is the reason for the large number of currently existing algorithms, as well as the proposal of new ones. Considering the NFL theorem, additional metaheuristics are introduced in this research compared to [1] to try to improve the peach firmness prediction accuracy of the regression tree model.

The idea of this study is to build a precise machine learning model for estimating peach firmness. The starting point for this research is the work described in [1], in which multiple linear regression and classification and regression tree models were developed with the same goal in mind. This research supplements [1] by introducing new machine learning models. Both studies are unique in exploring the possibilities of applying metaheuristics for regression tree model optimization with the aim of accurately predicting peach firmness. In this research, bagging, boosting, and random forest ensembles are introduced, as well as artificial neural networks and adaptive-network-based fuzzy inference systems. To obtain the best model fit for the peach firmness prediction task, the RMSE for all the developed models is calculated.

In addition to introducing new machine learning algorithms, this research complements [1] by investigating in more detail the possibilities for optimizing machine learning models using metaheuristics. In [1], the genetic algorithm, bat algorithm, differential evolution, and particle swarm optimization were used to improve the accuracy of the regression tree model, where the genetic algorithm gave the most promising results. In this research, additional nature-inspired metaheuristics are introduced and their performance compared, such as the black hole optimization algorithm, gray wolf optimizer, dragonfly algorithm, and whale optimization algorithm. In addition, the calculated *t*-tests give the significance of prediction accuracy improvement between optimized regression trees and default trees and serve as empirical evidence that metaheuristics can be used in the optimization of machine learning models, more precisely regression tree models. The development of the improved regression tree model for predicting peach firmness is the main contribution of this research, as is the empirical evidence that metaheuristics can be successfully applied to optimize the regression tree model. Using the measured predictor values as inputs to the model, the peach maturation degree can be estimated. Since the developed model is trained on the dataset consisting of various peach measurements, it will accurately estimate peach firmness. It is encouraged that the same methodology is applied to developing firmness prediction models on other fruits or crops.

Due to the very high number of nature-inspired metaheuristic algorithms that currently exist, not all of them could be included in the research. Eight metaheuristics are now being studied as part of this research, ranging in age from the more traditional genetic algorithm to the more modern black hole optimization algorithm. Future plans call for adding more contemporary nature-inspired metaheuristics and evaluating how well they perform in comparison to the current algorithms. The research focuses on optimizing the regression tree models using metaheuristics with the goal of improving their peach firmness prediction accuracy. Other machine learning models currently included in the research are used as benchmarks for comparing the performance of optimized regression trees. It is feasible to include other machine learning models in future studies and compare their prediction performance to optimized regression trees.

The paper is organized as follows: The literature review is given in Section 2, while in Section 3, the dataset used in conducting the experiment is described. The machine learning and metaheuristic algorithms used in the experiment are described in Section 4. The results of the peach firmness experiment and the dedicated *t*-test are presented in Section 5. Finally, the results are discussed in Section 6, as well as the direction for future research.

2. Literature Review

The precise prediction of fruit harvest date and extension of its shelf life can be obtained by early assessment of fruit maturity. To accurately predict peach firmness based on the measured dielectric properties, a backpropagation neural network was used in [2]. In addition, the conducted correlation analysis returned high correlation coefficients between dielectric properties and peach firmness. The research described in [8] focuses on investigating the dielectric properties of banana fruit to develop a rapid and non-destructive assessment method and to control the fruit ripening treatment. Similarly, in [9], the dielectric properties of Fuji apples with red-dot disease are investigated. To identify the category of the fruit, Zhang et al. developed a 13-layer convolutional neural network (CNN) [3]. An apple bruise detection system based on hyperspectral imaging in the shortwave infrared range is described in [10]. The research uses partial least squares discriminant analysis to discriminate bruised pixel spectra from sound pixel spectra. The successful prediction of banana fruit maturity based on measured impedance using a simple linear regression model is conducted in [11]. The research also proves that the measured impedance and banana maturity are correlated. The research described in [4,12] uses image data to predict melon and apple maturity, respectively. In [4], the discriminant analysis model is used for prediction, while the backpropagation neural network model is used in [12]. For the purpose of predicting peach firmness, the backpropagation neural network, multiple linear regression, and simple linear regression model are developed in [5]. The experiment results in [5] show that the multiple linear regression model is the most accurate one. The research described in [1] extends [5] by including additional features in the dataset as well as including a new regression tree model that is optimized using various nature-inspired metaheuristic algorithms to additionally optimize the model. Multiple linear regression, default regression tree, and regression tree models optimized using the genetic algorithm, bat algorithm, differential evolution, and particle swarm optimization metaheuristic algorithms are used for predicting peach firmness in [1]. This research builds on [1] by introducing artificial neural networks, adaptive-network-based fuzzy inference systems, bagging, boosting, and random forest ensemble models. With respect to the NFL theorem, four new metaheuristic algorithms are applied for regression tree optimization.

Since metaheuristics are not problem-specific, they can be applied in a variety of applications. To optimize the energy consumption in different fields, the bat algorithm is used in [13], while in [14,15], the differential evolution and particle swarm optimization algorithms are used, respectively. The variation in the PSO algorithm is applied in [16] to improve the design of parabolic dish concentrators, while in [17] it is used to optimize the gateway placement in wireless mesh networks. To optimize the speed control of a brushless direct current drive, the bat algorithm is used in [18]. Similar to [19], the genetic

algorithm is used for tuning the control gains of the fuzzy controller to minimize the energy consumption of unmanned aerial vehicles.

The metaheuristic algorithms are also applied in the optimization of the machine learning models, including classification and decision trees. The paper in [20] describes a machine learning model based on an extreme learning machine (ELM) optimized by the bat algorithm. A convolutional neural network and bat algorithm are used in [21], where the bat algorithm is utilized in the feature selection process, which increases the accuracy of the overall model. A method described in [22] employs differential evolution algorithms to optimize the initial weights of the convolutional neural network, aiming to achieve near-global optimal solutions and expedite network convergence. The study in [23] introduces a differential evolution-based neural architecture search approach for brain vessel segmentation, while the work in [24] uses the improved particle swarm optimization algorithm to optimize the initial weights of the BP neural network and increase its efficiency. In [25], a black hole optimization algorithm is used to fine-tune the hyperparameters of the convolutional neural network model with the goal of increasing its accuracy. The modified African buffalo algorithm described in [26] is used to create efficient and optimal decision trees. The results show that the African buffalo algorithm successfully optimizes the decision tree model, increasing its accuracy and reducing its size. The obtained optimized trees are shown to be more stable and efficient than conventional decision trees. Further decision tree optimization research is described in [27], in which trees are optimized using the artificial bee colony optimization algorithm.

This paper presents machine learning algorithms that use five predictor variables as inputs to predict peach firmness. In contrast to the other peach firmness prediction research [2], which only employs impedance data as input to the created ANN model, there are more predictor variables in this study. Another study that employed impedance to predict fruit maturity is [11]. It uses the measured impedance value to predict the maturity of banana fruit using a simple linear regression. Regression tree models have an advantage over other peach firmness prediction methods discussed in that they produce easily interpretable rules. Since regression trees are nonlinear and nonparametric, they do not presuppose a particular connection between the predictors and the outcome. This is an advantage when modeling datasets with many features or many complex, nonlinear relationships among features and outcomes [28]. Other fruit maturity prediction methods utilize image data and machine learning models such as discriminant analysis [4] and convolutional neural networks [12]. Compared to the previous research [1], new metaheuristic algorithms are introduced to test which algorithm gives the largest prediction accuracy improvement when applied to the regression tree model optimization problem.

3. Dataset Description

The measurement data of 200 peaches constitutes the dataset used in the experiment. The measurements are performed in two consecutive days. On the first day, the first group of 100 peaches is taken out of the refrigerator and their features are measured, while on the second day, the measurements are performed on the second group of the same size as the first one. The peaches used in the experiment were bought on the market and kept in the refrigerator at 4 °C before being measured.

Multiple techniques are used for measuring peach features to obtain the dataset on which the experiment has been conducted. Using the alternate voltage of 500 mV amplitude and 10 kHz frequency, the peach impedance is measured. The measured electrical impedance of a peach gives insight into the structural characteristics of its tissue [29]. Changes in fruit ripening reflect changes in the membrane structure, and the latter are related to changes in dielectric properties [30–32]. The measured electrical impedance that is part of the experiment's dataset is represented using magnitude *Zs* and phase angle *Angle* values. By combining juice extracted from a peach and an alkaline solution, the titratable acidity *TA* of the peach is obtained. Using the juice extracted from a peach and a refractometer, the soluble solids content *SSC* is obtained. The peach firmness is measured

using a penetrometer. Peach firmness is a good maturity indicator [33] and is therefore used as an outcome feature in this research. The Delta E value represents the peach's color.

In order to improve the effectiveness of machine learning algorithms [34], the dimension of the initial dataset, which consisted of nine measured features, had to be reduced. The process of dimension reduction is conducted using the expert's domain knowledge, which resulted in the dataset of 200 observations and six features that are used in the experiment. The mass and volume are combined into *density*, while the ratio of SSC and TA is used as a single feature. The categorical feature *appearance evaluation* is not used in the experiment. The distribution of the measured features that are part of the final dataset is shown in Figure 1. The ratio of soluble solids content and titratable acidity follows a normal distribution skewed to the right. The same is true for the densify feature, yet the color feature follows a symmetric normal distribution. The impedance magnitude, impedance phase angle, and peach firmness follow a bimodal distribution. From the firmness distribution, which is skewed towards lower values, it is possible to conclude that most of the peaches that make up the dataset have a lower firmness value and therefore are riper [33].

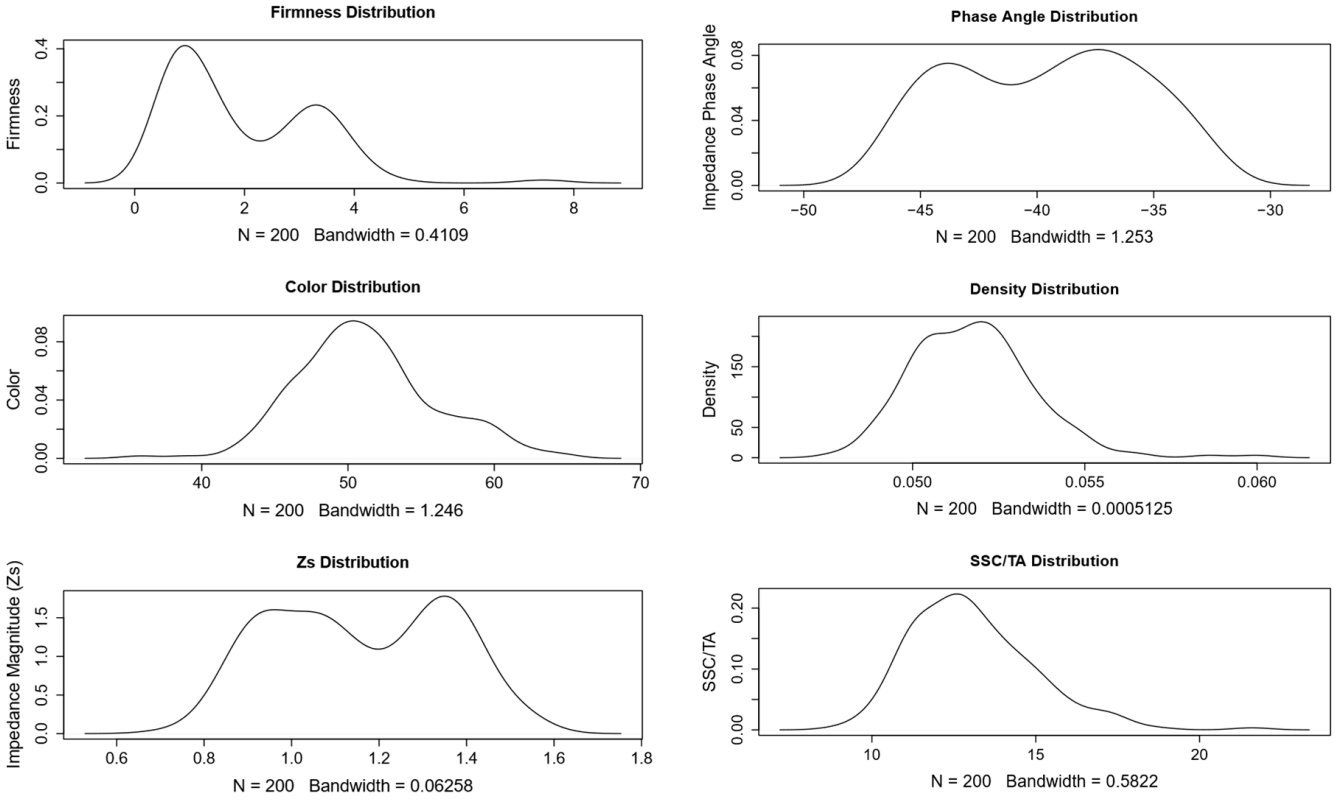

Figure 1. Distributions of features.

4. Methods

4.1. Multiple Linear Regression

The multiple linear regression (MLR) method assumes a linear relationship between multiple predictors and the outcome and is used for predicting a numerical outcome based on multiple numeric predictors [35]. In order to determine the best model fit and optimal coefficient values, the least squares method is used. The least squares method determines the coefficient values to minimize the sum of the squared errors, that is, the vertical distance between the predicted y value and the actual y value [28]. The multiple linear regression model that represents the linear relationship between predictors and the outcome can be expressed as follows [35]:

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \beta_3 \cdot X_3 + \ldots + \beta_n \cdot X_n + e, \quad (1)$$

where $\beta_{0...n}$ are multiple linear regression coefficients, $X_{0...n}$ are feature values, and e is a mean-zero random error term.

4.2. Classification and Regression Trees

The classification and regression trees (CART) algorithm can be used to build classification or numerical prediction models [34], with simplicity and transparency as their most prominent advantages.

The recursive partitioning and pruning processes are the main idea behind the CART algorithm, which is used to obtain the segmentation of the predictor space into multiple simple regions [35]. The resulting subgroups are more homogeneous in terms of the outcome feature, thereby creating useful prediction rules that are easily interpreted by humans. The CART algorithm produces a model with terminal and decision nodes that looks like a flowchart. While the terminal nodes contain the predicted value, the decision nodes provide the splitting value for a specific predictor. The arithmetic means of the outcome feature of training observations in the adequate region to which the observations belong are values of the terminal nodes. Figure 2 shows the default CART model created on the peach dataset.

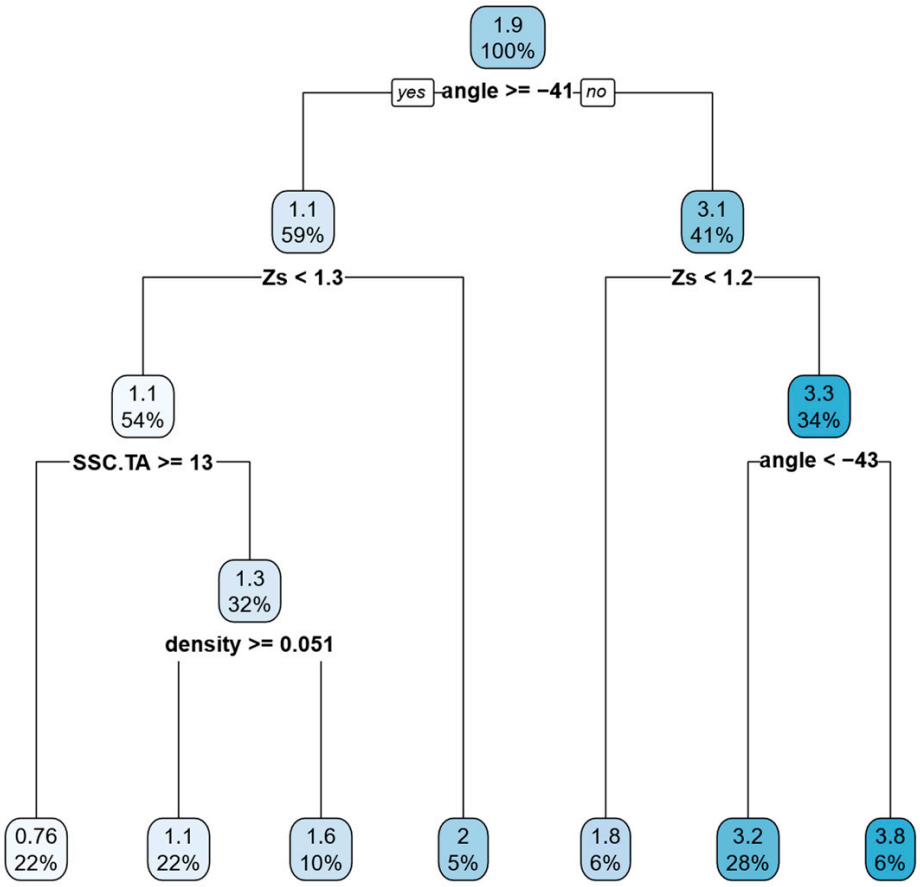

Figure 2. Regression tree model for peach firmness prediction.

CART models are unstable learners; that is, they tend to change substantially when the input data change only slightly [28]. The prediction performance of CART models can be considerably improved using ensemble methods [35].

4.3. Ensembles

The idea behind ensemble methods is the creation of a stronger learner by combining multiple weak learners [28]. Bootstrap aggregation (bagging) generates several training datasets by bootstrap sampling the original training dataset [28]. The created training

datasets are then used to generate a set of models using a single learning algorithm. The model's predictions are combined using averaging for the task of numeric prediction [28].

Bagging performs well if it is used with unstable learners, for example, CART models. Unstable models are essential to ensure the ensemble's diversity despite only minor variations between the bootstrap training datasets [28].

Boosting is another ensemble-based method that, like bagging, uses ensembles of models trained on resampled data and a vote to determine the final prediction [28]. Compared to bagging, the resampled datasets in boosting are constructed specifically to generate complementary learners [28]. In addition, rather than giving each learner an equal vote, boosting gives a weight to each learner based on its past performance [28]. Though boosting principles can be applied to nearly any type of machine learning algorithm, the principles are most used with decision trees [28].

Another ensemble-based method is random forest. It is a versatile method that combines the principles of bagging with random feature selection to add additional diversity to the CART models [28]. After the ensemble of trees is generated, the model uses a vote to combine the predictions. As the ensemble uses only a small, random portion of the full feature set, random forests can handle extremely large datasets, and relative to other ensemble-based methods, random forests tend to be easier to use and less prone to overfitting [28].

4.4. Artificial Neural Networks

Artificial neural networks (ANNs) are machine learning models used for classification and numerical prediction. The strength behind ANNs is their ability to capture complicated relationships among features in the dataset [34]. The multilayer feedforward network, which is a fully connected network with a one-way flow of information and no cycles, is the most widely applicable ANN architecture [34]. The input layer of the multilayer feedforward network consists of nodes that accept the predictor values. The successive layers of nodes receive weighted input from the previous layers, process the information, and output the information to the next layer. The output layer is the last layer of the ANN, and the layers between the input and output layers are called hidden layers [34].

To learn relationships among features in the dataset, the ANN uses the backpropagation algorithm [28]. Before the learning process is started, the weights of the ANN are initialized to random values. The basic working of the backpropagation algorithm is that it iterates through many epochs (cycles) of forward and backward phase processes until a stopping criterion is reached [28]. In the forward phase, the neurons are activated in sequence from the input layer to the output layer, applying each neuron's activation function along the way. When the final layer is reached, an output signal (prediction of the outcome feature) is produced [28]. In the backward phase, the ANN's output signal produced during the forward phase is compared to the true outcome value in the training data. The difference between these two signals makes the error, which is propagated backwards in the network to modify the connection weights between neurons and reduce future errors. The gradient descent technique is used to adjust the weight's values. The reason that it is important to have a differentiable neuron's activation function is that the gradient descent calculates its derivative to identify the gradient in the direction of each of the incoming weights [28]. The gradient identifies how steeply the error will be reduced or increased for a change in weight value. The backpropagation algorithm will attempt to find the weights that result in the greatest error reduction by an amount known as the learning rate [28]. Over time, the total error will be reduced, and the ANN will learn how to accurately predict the outcome. Figure 3 shows the ANN model used in this research for peach firmness prediction.

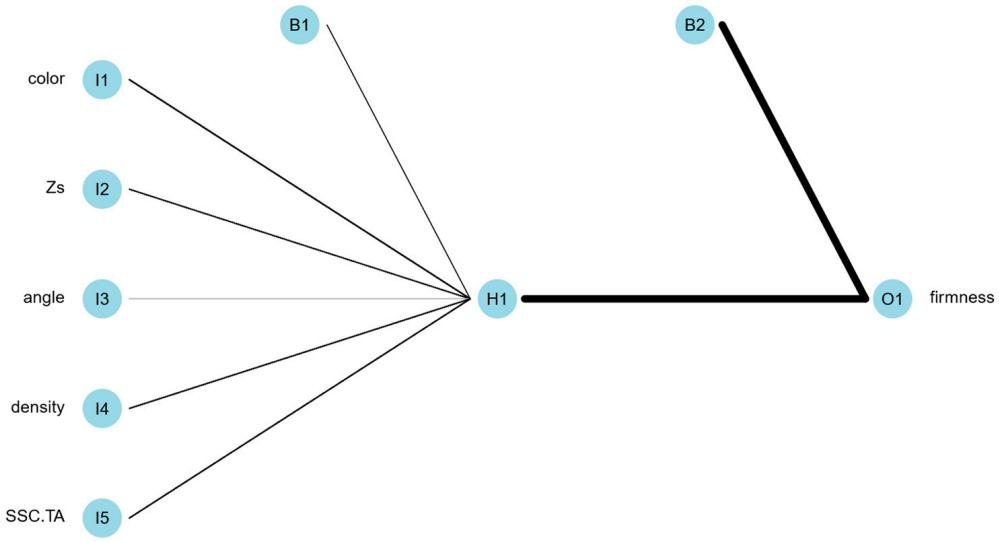

Figure 3. ANN used for peach firmness prediction.

4.5. ANFIS

The ANFIS (adaptive-network-based fuzzy inference system) is a fuzzy inference system implemented in the framework of adaptive networks [36]. Fuzzy if-then rules or fuzzy conditional statements are expressions of the form IF A THEN B, where A and B are labels of fuzzy sets characterized by an appropriate membership function [36]. Using linguistic labels and membership functions, a fuzzy if-then rule can easily capture the spirit of a "rule of thumb" used by humans [36].

Using the processes of fuzzification and defuzzification, the fuzzy inference systems perform inference operations upon fuzzy if-then rules [36]. The fuzzy inference system used in the experiment uses Takagi and Sugeno's fuzzy if-then rules. In it, the output of each rule is a linear combination of input variables plus a constant term, and the final output is the weighted average of each rule's output [36].

The ANFIS architecture consists of five layers. The first layer is responsible for the fuzzification process, which transforms crisp values into linguistic terms using the membership function. The second layer is denoted as the "rule layer" due to the task that each node represents the firing strength of a dedicated single rule. T-norm operators that perform generalized AND can be used as the node function in this layer [36]. The normalization of the computed firing strengths by dividing each rule's firing strength by the total firing strength is the responsibility of the nodes in the third layer [36]. The outputs of this layer are called normalized firing strengths. Every node in the fourth layer is defined by a set of parameters (p, q, r). Parameters in this layer will be referred to as consequent parameters. This layer outputs defuzzified outcome values. The single node in the fifth and final layer computes the overall output as the summation of all incoming signals [36].

The ANFIS adjusts its parameters during the learning stage using the least squares method, in which the coefficients of linear equations on the consequent part and mean and variance on the premise part are adjusted [36].

4.6. Genetic Algorithm

In the fields of computer science and mathematical optimization, a metaheuristic procedure may provide a sufficiently good solution to an optimization problem [7]. An inherent characteristic of all metaheuristic algorithms is that they make few or no assumptions about the problem being optimized and therefore can be easily applied to a wide range of optimization problems.

One of the metaheuristic algorithms is the genetic algorithm, which belongs to a family of evolutionary algorithms and can be applied to solving global optimization problems. The algorithm requires a definition of the quality function, which is applied to candidate

solutions as an abstract fitness measure—the higher the better [7]. Some of the better candidates are selected to be parents of the next generation based on a calculated level of fitness. Where the mutation operator is applied to a single candidate and produces one new candidate, the recombination operator is applied to two or more selected parents and generates one or more new offspring candidates [7]. The fitness values of all candidate solutions are evaluated, after which the candidates compete based on their fitness (and possibly age) for a place in the next generation [7]. This process can be iterated until a candidate with sufficient quality is found or a previously set computational limit is reached [7]. Figure 4 shows the rise in fitness value of the best candidate solution (green) and the population (blue) during the run time of the genetic algorithm used in decision tree model optimization, which is described in Section 5.

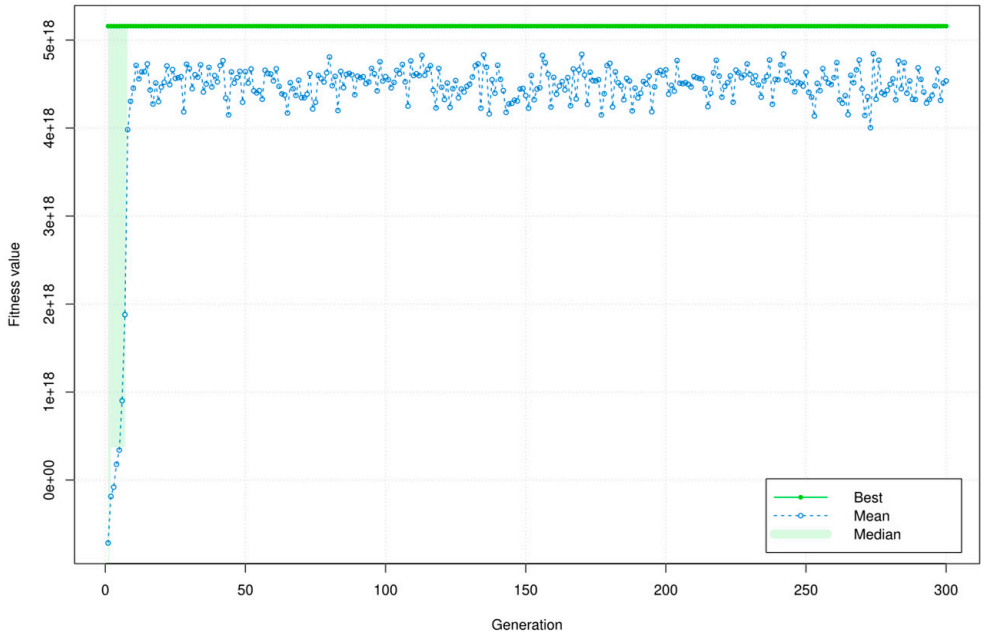

Figure 4. Change in fitness value.

4.7. Bat Algorithm

Another metaheuristic algorithm for global optimization is the bat algorithm (BA), which is inspired by the echolocation behavior of microbats [37]. The BA can be classified as a swarm intelligence (SI) algorithm since it is a global optimization technique that uses a swarm of multiple, interacting agents that perform search moves in the search space. A wide spectrum of SI-based algorithms has emerged in the last decades, but the lack of a mathematical framework and in-depth understanding of how such algorithms may converge are still some of the important issues [38]. The BA uses a frequency-tuning technique to increase the diversity of the solutions in the population [37], and as with other metaheuristic algorithms, the balance between exploration and exploitation can be controlled by tuning the BA's algorithm-dependent parameters.

At algorithm iteration t, each bat i is associated with a velocity v_i^t and a location x_i^t in a d-dimensional space [37]. Next are equations for x_i^t and velocities v_i^t, where x^* denotes the best solution in a population in any given iteration, while B [0, 1] is a random vector drawn from a uniform distribution [37].

$$f_i = f_{min} + (f_{max} - f_{min}) \cdot \beta, \quad (2)$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x^*) \cdot f_i, \quad (3)$$

$$x_i^t = x_i^{t-1} + v_i^t, \quad (4)$$

The loudness A_i and pulse emissions rate r_i must change during algorithm iterations to offer a way of managing the exploration and exploitation processes [37]. In the following equations, α and γ are constants.

$$A_i^{t+1} = \alpha \cdot A_i^t, \tag{5}$$

$$r_i^{t+1} = r_i^0 \, [1 - \exp(-\gamma \cdot t)], \tag{6}$$

4.8. Black Hole Optimization

The black hole optimization (BHO) algorithm is a metaheuristic algorithm inspired by the characteristics of black holes [39]. A black hole is a region of space-time whose gravitational field is so strong that not even light can escape it, and thus no information can be obtained from its region [39]. The algorithm starts by randomly initializing a population of candidate solutions (stars) in the search space [39]. At each iteration, the fitness value of each star is evaluated, and the best star is selected as the black hole X_{BH}. In following algorithm iterations, the black hole candidate solution does not move but attracts other candidate solutions [39]. If the star reaches a location with a lower cost than the black hole while moving towards it, the black hole is updated by selecting this star [39]. But if a star gets too close to the black hole, it will be swallowed by it and will disappear from the population [39]. In that case, a new star is randomly generated and placed in the search space [39]. The algorithm repeats the described steps until a termination criterion (a maximum number of iterations or a sufficiently good fitness) is reached [39].

4.9. Differential Evolution

The differential evolution (DE) algorithm solves real parameter global optimization problems and is like other EAs since it produces new offspring solutions through three mechanisms: mutation, crossover, and selection [40]. One of the algorithm's features is that each individual in the current generation is allowed to breed with multiple other randomly selected individuals from the population [39]. The algorithm utilizes directional information from the population and starts by initializing all agents in the search space with random positions, after which mutation is applied to create a vector v_i^t. Using the arithmetic recombination of the individuals in the current generation, the target vector u_i^t is created [39].

To create a mutated vector, the DE standard mutation operator needs three randomly selected different individuals from the current population [39]. The aim of the mutation operator is to recognize good variation directions and to increase the number of generations having fitness improvement [39].

$$v_i^t = x_j^t + F \, (x_k^t + x_l^t), \tag{7}$$

where F is a differential weight real constant with a value between 0 and 1. By mating the mutated individual v_i^t with x_i^t, the offspring u_i^t is created. The genes m of u_i^t are determined by the crossover probability $C_r \in [0, 1]$ and are inherited from x_i^t and v_i^t [39].

$$u_{i,m}^t = \begin{cases} v_{i,m}^t, \text{ if } \mathrm{rand}(m) \leq C_r \text{ or } m = \mathrm{rn}(i). \\ x_{i,m}^t, \text{ if } \mathrm{rand}(m) > C_r \text{ or } m \neq \mathrm{rn}(i). \end{cases} \tag{8}$$

The parent selection process for the next generation is performed by selecting the fittest individuals among x_i^t and its offspring u_i^t. Using the calculated fitness values, the winner is selected and promoted to the next generation. The old generation is replaced by the new one, and the search process continues until the stopping condition is fulfilled [39].

4.10. Gray Wolf Optimization

Many metaheuristic algorithms are inspired by the wolf's search manner [41]. The gray wolf optimization (GWO) algorithm is inspired by the leadership hierarchy and hunting mechanism of gray wolves [42]. Gray wolves are considered apex predators and live in

a pack that is characterized by a strict social dominant hierarchy [41]. The pack leaders are called alphas, and their decisions are dictated to the pack [41]. The second level in the hierarchy of gray wolves is a beta, and it is the best candidate to be the alpha in case one of the alpha wolves passes away [41]. Delta is the third level in the hierarchy. Delta wolves must submit to alphas and betas, but they dominate the omega, which is the lowest-ranking gray wolf [41]. In the mathematical model for the GWO, the best solution is called the alpha, while the second and third best solutions are named beta and delta, respectively [41]. The rest of the candidate solutions are assumed to be omega [41]. The algorithm mathematically implements tracking, surrounding, hunting, and attacking processes of gray wolves to solve global optimization problems [42]. At the beginning of the algorithm execution, it randomly initializes the gray wolf population. The fitness value of each candidate solution is calculated, and the best one is set as alpha, the second best as beta, and the third best as delta. All other candidate solutions (wolves) are assumed to be omega. Next, the position of candidate solutions is updated depending on the position of alpha, beta, and delta wolves [41]. These three are the best solutions obtained so far and oblige the other search agents to update their positions according to their position [41]. If the new positions of candidate solutions have better fitness, then the alpha, beta, and delta are replaced by the wolves with better fitness. The algorithm repeats the steps until the termination criteria are reached and alpha is returned as the optimal solution for the given problem.

4.11. Particle Swarm Optimization

Particle swarm optimization (PSO) is a metaheuristics algorithm that solves global optimization problems by having a population of candidate solutions (particles) that work under social behavior in swarms [43]. The socio-cognitive learning process that is based on a particle's own experience and the experience of the most successful particle in the swarm is the most prominent characteristic of the PSO algorithm [39]. Each particle in the swarm N_P is assigned an initial random position in the n-dimensional space for an optimization problem of n variables [39]. The position x_i and velocity v_i are variables that define each particle that is part of the swarm. In every iteration, each particle is updated by following the two best values: the best solution each particle has achieved so far x_i^* and the best value obtained so far by any particle in the population x_g [39]. At iteration t + 1, the swarm can be updated by the following [39]:

$$v_i(t+1) = v_i(t) + cr_1[x_i^*(t) - x_i(t)] + cr_2[x_g(t) - x_i(t)], \quad (9)$$

$$X_i(t+1) = x_i(t) + v_i(t+1), i = 1, \ldots, N_p, \quad (10)$$

where the acceleration constant $c > 0$, and r_1 and r_2 are uniform random numbers within [0, 1] [39]. The PSO algorithm is a popular choice for solving optimization problems due to its simple representation and low number of adjustable parameters [43].

4.12. Dragonfly Algorithm

The main inspirations for the dragonfly algorithm (DA) are the hunting and migration behaviors of dragonflies. The dragonflies create small groups for food search (static behavior) and larger groups for migrating (dynamic behavior) [44]. When searching for food, an exploration (diversification) strategy is applied where the dragonflies in small groups try to search the solution space and, thus, each group by itself searches for the optimum solution [44]. When migrating, the dragonflies will create larger groups and will migrate to the most promising locations, finding better solutions [44].

The DA solves global optimization problems by combining local and global searches using the behavior that was previously described. The algorithm first randomly initializes the dragonfly population. Each dragonfly's fitness value is determined, and the best dragonfly is chosen as a food source, while the dragonfly with the poorest fitness score is selected as the position of the enemy. Considering the neighboring radius, which rises linearly with each iteration, identify the nearby dragonflies for each dragonfly in

order to compute the behavior weight that affects the fly direction and distance. Next, using the velocity parameter and the calculated behavior weight, update each dragonfly's position. Finally, update the food and enemy positions by calculating the fitness score for all dragonflies in the population. If the termination criterion is satisfied, return the food position as the optimal solution for the given problem.

4.13. Whale Optimization Algorithm

The whale optimization algorithm (WOA) is a metaheuristic algorithm inspired by the social behavior and the bubble-net hunting strategy of humpback whales [45]. Whales form bubble clouds by exhaling underwater [45], and using the bubble-net hunting strategy, they gather their prey together inside the bubble clouds. This way the whales ensure that prey stays inside the bubbles, and as the whales move toward the surface, they catch the prey [45]. The WOA implements the following phases of the mentioned whale behavior: the exploration phase in which the search for the prey is conducted, the process of encircling the prey, and the exploitation phase in which the prey is attacked using the bubble-net method. The algorithm starts by initializing the population of search agents (whales) and evaluating their fitness values. After the initialization, the exploration phase and search for prey begin. When the prey has been found, the whales encircle it. In the WOA, surrounding the prey is considered finding the best solution or a point around it [45]. After finding the prey and encircling it, the prey is attacked using the bubble-net hunting strategy, after which the positions of other search agents are updated. Upon finishing the WOA search, the algorithm's parameters are altered, and in the event that a new whale with a higher fitness is found, the best whale's position is also updated.

As with other described metaheuristic algorithms, the WOA repeats the steps until the termination criteria are reached, after which the best solution is returned.

5. Results

Compared to the last experiment described in [1], multiple regression tree models are optimized and compared to gain a more accurate estimate of the performance of the optimization algorithms. In addition, a new approach in regression tree model optimization has been selected, in which the complexity parameter (CP) is being optimized and used in pruning the overgrown trees. A total of 1000 default pruned trees have been created and their performance compared with 1000 overgrown models optimized using each metaheuristic algorithm described in this paper. Eighty percent of the randomly selected dataset observations are used to train the models, and the remaining twenty percent are used to test them. The test data consist of 40 observations and are used to estimate the accuracy of the developed machine learning models, whereas 160 observations make up the training data used to train the models.

When using the *rpart* R library to build the default pruned tree, the pruning of the tree is performed automatically by selecting the CP value that produces the minimum error on validation data during the cross-validation procedure. Below each decision node is the yes–no conditional based on which the split is made. The mean value of firmness, which is calculated using observations that satisfy the conditions given by the decision nodes, is contained inside each decision and terminal node.

The approach in CART model optimization is to optimize the CP value using nature-inspired metaheuristics and use it to prune back the purposely overgrown CART model. The lower and upper bounds of the CP search space for all optimization algorithms are 0.00001 and 1, respectively. The optimization problem is set up to maximize the objective function; thus, the CP value with the highest fitness score will be selected.

The set parameters of the genetic algorithm implemented in the *GA* R library are population size, whose value is 200, and the number of best individuals to survive in each generation, which is set to 40 percent. The GA parameter values are determined experimentally with the goal of obtaining the best CART optimization performance. The parameter values for other metaheuristics that follow are obtained in the same way.

All other nature-inspired metaheuristics are implemented using the *metaheuristicOpt* R package. The population size parameter of the bat algorithm is set to 40, and the maximum and minimum frequencies are set to 0.1. The default value of 1 is assigned to the factor that increases pulse rate, while the value of 0.1 is assigned to the factor that decreases loudness. In the DE algorithm, each individual in the generation is permitted to mate with other randomly selected individuals. The crossover probability factor is set to 0.5, and the mutation operator's scaling factor is set to 0.8. The size of the population is 20. When implementing the PSO as a CART optimization algorithm, the inertia weight is set to 0.729. The value of 1.49445 is set for both the individual and group acceleration constants, while the population size is 20. For the dragonfly algorithm (DA), gray wolf optimizer (GWO), black hole optimization (BHO) algorithm, and whale optimization algorithm (WOA), the population size is set to 40 and the maximum number of iterations is 500.

To compare the performance of default and optimized CART models, multiple other machine learning models are introduced. These are MLR, ANFIS, ANN, bagged decision trees, boosted trees, and random forests. The mentioned models are implemented by the *frbs nnet*, *ipred*, *XGBoost*, and *randomForest* R packages, respectfully. The optimal parameters of the developed machine learning models were obtained using an automated parameter tuning technique implemented in the *caret* R package. Machine learning parameter tuning is the process of adjusting the model options with the goal of identifying the best fit. In the tuning process, a repeated 10-fold cross-validation is used as the resampling method. The *oneSE* selection function defined in *caret* is used to prioritize simpler models during the tuning process. Therefore, the parameter values are automatically obtained to gain the best model fit using the repeated 10-fold cross-validation and *oneSE* function, which chooses the simplest candidate model within one standard error of the best performance. By using 10-fold cross-validation repeated 10 times and selecting the simplest candidate model, the risk of overfitting is minimized. The tuned parameters for the ANN model are the number of units in the hidden layer and the weight decay. The result of the tuning process is the ANN model with 1 node in the hidden layer and 0.5 as the value of the weight decay parameter. The maximum number of boosting iterations in the tuned boosting model is 10, and the maximum depth of a tree is 1. For the random forest model, only the number of variables randomly sampled as candidates at each split is tuned, and its final value is 1. The tuned ANFIS model uses six linguistic terms, and the maximum number of iterations is limited to 17.

The developed models made predictions on the 1000 different test datasets, and the mean RMSE value shown in Table 1 is calculated.

Table 1. Test RMSE summary.

Model	RMSE
Default CART	1.722285
GA Optimized CART	1.706319
BA Optimized CART	1.707731
DE Optimized CART	1.58699
PSO Optimized CART	1.617216
DA Optimized CART	1.574756
GWO Optimized CART	1.570924
WOA Optimized CART	1.574094
BHO Optimized CART	1.575638
MLR	1.654896
ANFIS	6.127994
ANN	1.612046
Bagging	1.684522
Boosting	1.576703
Random Forest	1.726419

The *t*-test is conducted to measure the statistical difference between the mean RMSE values (Table 1) of the default CART and its optimized variations. Table 2 shows the results of the conducted *t*-tests.

Table 2. The *t*-test summary.

Models	*p*-Value
Default CART and GA Optimized CART	0.0541
Default CART and BA Optimized CART	0.07835
Default CART and DE Optimized CART	2.2×10^{-16}
Default CART and PSO Optimized CART	2.2×10^{-16}
Default CART and DA Optimized CART	2.2×10^{-16}
Default CART and GWO Optimized CART	2.2×10^{-16}
Default CART and WOA Optimized CART	2.2×10^{-16}
Default CART and BHO Optimized CART	2.2×10^{-16}

Next, the power of the *t*-tests is calculated, and the results are given in Table 3.

Table 3. The *t*-test power.

Models	Power
Default CART and GA Optimized CART	0.9999413
Default CART and BA Optimized CART	0.9999476
Default CART and DE Optimized CART	0.9897147
Default CART and PSO Optimized CART	0.9938731
Default CART and DA Optimized CART	0.9926904
Default CART and GWO Optimized CART	0.9925089
Default CART and WOA Optimized CART	0.9923514
Default CART and BHO Optimized CART	0.9926872

Since the conducted Welch *t*-test assumes the normal distributions of samples, Figure 5 shows the RMSE distributions for the default CART model and a few of its optimized variations.

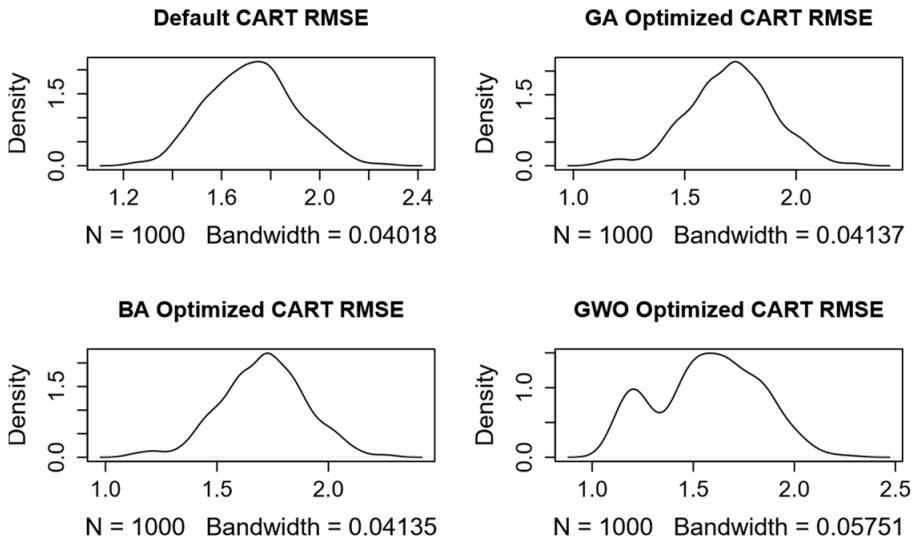

Figure 5. RMSE sample distribution of the default pruned tree and its optimized variations.

6. Discussion

From the summary of the RMSE results, it is possible to see that the GWO CART model gives the best predictions on test data. Other metaheuristics, such as the WOA, BHO, and DA, are also very successful in improving the accuracy of the default CART model.

The best machine learning model is boosting, which has very similar prediction accuracy to the best-optimized CART models.

In the previous research [1], the most accurate model was the CART model optimized using the GA, with an RMSE of 1.589032. Comparing it to the GWO model developed as part of this research, the GWO CART is slightly more accurate. It is important to note that this research uses 1000 samples of test data for predictions, while in [1], only one sample has been used. From the results of the *t*-test, it can be seen that the *p*-value associated with the Welch two-sample *t*-test is sufficiently low. Therefore, it is possible to reject the null hypothesis of no difference between the (true) prediction RMSE averages.

The goal of the experiment is to develop an accurate machine learning model for peach firmness prediction and to test if various metaheuristics can be utilized in the optimization of machine learning models, more specifically in the improvement of peach firmness prediction accuracy of CART models. MLR, regression trees, ANFIS, ANN, bagging, boosting, and random forest machine learning models have been developed and compared to obtain the most accurate model for predicting peach firmness. The GA, BA, DE, PSO, DA, GWO, WOA, and BHO metaheuristic algorithms were applied to test the possibility of improving the prediction accuracy of the regression tree model. Other related works that are investigating the optimization of machine learning models using metaheuristics also reported positive results. By improving the performance of regression trees used for predicting peach firmness, this research introduces a new area of application for the mentioned methodology. This empirical experiment shows that by using various metaheuristic optimization techniques implemented in *GA* and *metaheuristicOpt* R packages, it is possible to improve the accuracy of the default CART model. The significance of prediction accuracy improvement between optimized regression trees and default trees is given by the calculated *t*-tests.

By replacing manual peach evaluation methods with the automated system described in this paper, the efficiency of the farming systems could be improved and food waste reduced. Some practical challenges could arise when implementing the proposed optimized machine learning models in real-world agricultural settings. One of the challenges is the initial cost of the proposed system, as well as the complexity of integrating it into the existing farming system.

Future research could focus on introducing more metaheuristic algorithms and comparing their performance with the results shown in this paper. A generative adversarial network (GAN) can be utilized to generate a larger dataset that will be the basis for training and testing new machine learning models.

Author Contributions: Conceptualization, T.I.; methodology, T.I.; software, T.I.; validation, T.I., M.G. and M.M.; formal analysis, T.I., M.G. and M.M.; investigation, T.I.; resources, T.I.; data curation, T.I.; writing—original draft preparation, T.I., M.G. and M.M.; writing—review and editing, M.G. and M.M.; visualization, T.I.; supervision, M.G. and M.M. 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: The authors will make the raw data supporting this article's conclusions available upon request.

Acknowledgments: This research has been supported under Grant No. uniri-iskusni-drustv-23-143 of the University of Rijeka.

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

References

1. Ivanovski, T.; Zhang, X.; Jemrić, T.; Gulić, M.; Matetić, M. Firmness Prediction Using Optimized Regression Trees Models. In Proceedings of the 33rd DAAAM International Symposium on Intelligent Manufacturing and Automation, Vienna, Austria, 27–28 October 2022; Volume 33, pp. 480–489. [CrossRef]
2. Zhang, G.; Fu, Q.; Fu, Z.; Li, X.; Matetić, M.; Bakaric, M.B.; Jemrić, T. A Comprehensive Peach Fruit Quality Evaluation Method for Grading and Consumption. *Appl. Sci.* **2020**, *10*, 1348. [CrossRef]
3. Zhang, Y.D.; Dong, Z.C.; Chen, X.Q.; Jia, W.J.; Du, S.; Muhammad, K.; Wang, S.H. Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. *Multimed. Tools Appl.* **2019**, *78*, 3613–3632. [CrossRef]
4. Ahmad, U.; Bermani, D.P.; Mardison, M. Color Distribution Analysis for Ripeness Prediction of Golden Apollo Melon. *Telkomnika* **2018**, *16*, 1659–1666. [CrossRef]
5. Ivanovski, T.; Guoxiang, Z.; Jemrić, T.; Gulić, M.; Matetić, M. Fruit firmness prediction using multiple linear regression. In Proceedings of the 43rd International Convention MIPRO, Opatija, Croatia, 28 September–2 October 2020; pp. 1570–1575.
6. Gulić, M.; Žuškin, M.; Kvaternik, V. An Overview and Comparison of Selected State-of-the-Art Algorithms Inspired by Nature. *TEM J.* **2023**, *12*, 1281–1293. [CrossRef]
7. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2015.
8. Soltan, M.; Alimardani, R.; Omid, M. Evaluating banana ripening status from measuring dielectric properties. *J. Food Eng.* **2011**, *105*, 625–631. [CrossRef]
9. Ma, H.; Song, C.; Zhang, J.; Zhang, W.; Hu, F. Influence of Frequency of Electric Excitation Signal on Dielectric Property of Fuji Apples with Red-dot Disease. *Trans. Chin. Soc. Agric. Mach.* **2009**, *40*, 97.
10. Keresztes, J.C.; Goodarzi, M.; Saeys, W. Real-time pixel based early apple bruise detection using short wave infrared hyperspectral imaging in combination with calibration and glare correction techniques. *Food Control* **2016**, *66*, 215–226. [CrossRef]
11. Jamaludin, D.; Aziz, S.A.; Ibrahim, N.U.A. Dielectric Based Sensing System for Banana Ripeness Assessment. *Int. J. Environ. Sci. Dev.* **2014**, *5*, 286–289. [CrossRef]
12. Sabzi, S.; Abbaspour-Gilandeh, Y.; García-Mateos, G.; Ruiz-Canales, A.; Molina-Martínez, J.M.; Arribas, J.I. An Automatic Non-Destructive Method for the Classification of the Ripeness Stage of Red Delicious Apples in Orchards Using Aerial Video. *Agronomy* **2019**, *9*, 84. [CrossRef]
13. Xu, F.; Zi, S.; Wang, J.; Ma, J. A computing offloading strategy for UAV based on improved bat algorithm. *Cogn. Robot.* **2023**, *3*, 265–283. [CrossRef]
14. Alqarni, M.A.; Mousa b, M.H.; Hussein, M.K.; Mead, M.A. Improved wireless sensor network data collection using discrete differential evolution and ant colony optimization. *J. King Saud Univ.–Comput. Inf. Sci.* **2023**, *35*, 101725. [CrossRef]
15. Muloiwa, M.; Dinka, M.O.; Nyende-Byakika, S. Modelling and optimizing hydraulic retention time in the biological aeration unit: Application of artificial neural network and particle swarm optimization. *S. Afr. J. Chem. Eng.* **2024**, *48*, 292–305. [CrossRef]
16. Li, L.; Zhang, Y.; Li, H.; Liu, R.; Guo, P. An optimized approach for solar concentrating parabolic dish based on particle swarm optimization-genetic algorithm. *Heliyon* **2024**, *10*, e26165. [CrossRef] [PubMed]
17. Neema, M.; Gopi, E.S.; Reddy, P.S. Optimizing Broadband Access and Network Design in Wireless Mesh Networks using Multi-Objective Particle Swarm Optimization. *Procedia Comput. Sci.* **2023**, *230*, 275–286. [CrossRef]
18. Premkumar, K.; Manikandan, B.V. Bat algorithm optimized fuzzy PD based speed controller for brushless direct current motor. *Eng. Sci. Technol. Int. J.* **2016**, *19*, 818–840. [CrossRef]
19. Rodríguez-Abreo, O.; Rodríguez-Reséndiz, J.; García-Cerezo, A.; García-Martínez, J.R. Fuzzy logic controller for UAV with gains optimized via genetic algorithm. *Heliyon* **2024**, *10*, e26363. [CrossRef]
20. Mekaoussi, H.; Heddam, S.; Bouslimanni, N.; Kim, S.; Zounemat-Kermani, M. Predicting biochemical oxygen demand in wastewater treatment plant using advance extreme learning machine optimized by Bat algorithm. *Heliyon* **2023**, *9*, e21351. [CrossRef] [PubMed]
21. Chawla, R.; Beram, S.M.; Murthy, C.R.; Thiruvenkadam, T.; Bhavani, N.P.G.; Saravanakumar, R.; Sathishkumar, P.J. Brain tumor recognition using an integrated bat algorithm with a convolutional neural network approach. *Meas. Sens.* **2022**, *24*, 100426. [CrossRef]
22. Li, J.; Soradi-Zeid, S.; Yousefpour, A.; Pan, D. Improved differential evolution algorithm based convolutional neural network for emotional analysis of music data. *Appl. Soft Comput.* **2024**, *153*, 111262. [CrossRef]
23. Kuş, Z.; Kiraz, B.; Göksu, T.K.; Aydın, M.; Özkan, E.; Vural, A.; Kiraz, A.; Can, B. Differential evolution-based neural architecture search for brain vessel segmentation. *Eng. Sci. Technol. Int. J.* **2023**, *46*, 101502. [CrossRef]
24. Tao, T.; Hua, L. Decoupling control of bearingless brushless DC motor using particle swarm optimized neural network inverse system. *Meas. Sens.* **2024**, *31*, 100952. [CrossRef]
25. Li, T. Optimizing the configuration of deep learning models for music genre classification. *Heliyon* **2024**, *10*, e24892. [CrossRef] [PubMed]
26. Panhalkar, A.R.; Doye, D.D. Optimization of decision trees using modified African buffalo algorithm. *J. King Saud Univ.–Comput. Inf. Sci.* **2022**, *34*, 4763–4772. [CrossRef]
27. Cui, Y. Optimizing decision trees for English Teaching Quality Evaluation (ETQE) using Artificial Bee Colony (ABC) optimization. *Heliyon* **2023**, *9*, e19274. [CrossRef]
28. Lantz, B. *Machine Learning with R*, 2nd ed.; Packt Publishing Ltd.: Birmingham, UK, 2015.

29. Jocsak, I.; Vegvari, G.; Vozary, E. Electrical impedance measurement on plants: A review with some insights to other fields. *Theor. Exp. Plant Physiol.* **2019**, *31*, 359–375. [CrossRef]
30. Ibba, P.; Falco, A.; Abera, B.D.; Cantarella, G.; Petti, L.; Lugli, P. Bio-impedance and circuit parameters: An analysis for tracking fruit ripening. *Postharvest Biol. Technol.* **2020**, *159*, 110978. [CrossRef]
31. Grossi, M.; Riccò, B. Electrical impedance spectroscopy (EIS) for biological analysis and food characterization: A review. *J. Sens. Sens. Syst.* **2017**, *6*, 303–325. [CrossRef]
32. Weaver, G.M.; Jackson, H.O. Electric impedance, in objective index of maturity in peach. *Can. J. Plant Sci.* **1966**, *46*, 323–326. [CrossRef]
33. Harker, F.R.; Maindonald, J.H. Ripening of Nectarine Fruit Changes in the Cell Wall, Vacuole, and Membranes Detected Using Electrical Impedance Measurements. *Plant Physiol.* **1994**, *106*, 165–171. [CrossRef]
34. Shmueli, G.; Bruce, P.C.; Yahov, I.; Patel, N.R.; Lichtendahl, K.C., Jr. *Data Mining for Business Analytics*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2018.
35. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning with Applications in R*; Corrected at the printing 2017; Springer Science+Business Media: New York, NY, USA, 2013.
36. Jang, J.-S.R. ANFIS Adaptive-Network-based Fuzzy Inference System. *IEEE Trans. Syst. Man Cybern.* **1993**, *23*, 665–685. [CrossRef]
37. Yang, X.-S. Bat Algorithm: Literature Review and Applications. *Int. J. Bio-Inspired Comput.* **2013**, *5*, 141–149. [CrossRef]
38. Yang, X.-S. Nature-inspired optimization algorithms: Challenges and open problems. *J. Comput. Sci.* **2020**, *46*, 101104. [CrossRef]
39. Du, K.-L.; Swamy, M.N.S. *Search and Optimization by Metaheuristics-Techniques and Algorithms Inspired by Nature*; Birkhäuser: Basel, Switzerland, 2016.
40. Ahmad, M.F.; Isa, N.A.M.; Lim, W.H.; Ang, K.M. Differential evolution: A recent review based on state-of-the-art works. *Alex. Eng. J.* **2022**, *61*, 3831–3872. [CrossRef]
41. Hassanien, A.E.; Emary, E. *Swarm Intelligence: Principles, Advances, and Applications*; CRC Press: Boca Raton, FL, USA; Taylor and Francis Group: Abingdon, UK, 2016.
42. Li, Y.; Lin, X.; Liu, J. An Improved GrayWolf Optimization Algorithm to Solve Engineering Problems. *Sustainability* **2021**, *13*, 3208. [CrossRef]
43. Sengupta, S.; Basak, S.; Peters, R.A. Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 157–191. [CrossRef]
44. Marinaki, M.; Taxidou, A.; Marinakis, Y. A hybrid Dragonfly algorithm for the vehicle routing problem with stochastic demands. *Intell. Syst. Appl.* **2023**, *18*, 200225. [CrossRef]
45. Toren, M. Optimization of transformer parameters at distribution and power levels with hybrid Grey wolf-whale optimization algorithm. *Eng. Sci. Technol. Int. J.* **2023**, *43*, 101439. [CrossRef]