Genetic Algorithm for Model Selection in Machine Learning

by

Gayathridevi Arun Prasad

Registration Number: 202355415

In partial fulfilment of the
requirements for the degree of
MSc
in
Advanced Software Engineering

Department of
Computer and Information Sciences

August, 2024

Abstract

This dissertation explores the use of genetic algorithms (GAs) in the model selection process for machine learning (ML). Genetic algorithms are inspired by natural selection and evolution, and they offer a unique way to solve complex problems by finding the best solutions through a process of selection, crossover, and mutation. In this research, GAs are applied to select the most effective models and optimize their performance in machine learning tasks.

The study begins by reviewing existing literature on genetic algorithms and their applications in model selection. This theoretical foundation helps to understand the potential benefits and challenges of using GAs in this context. The research then develops a GA-based framework specifically designed for model selection in ML. The framework is implemented and tested on various datasets to evaluate its effectiveness.

Key risks in this research include the complexity of the algorithm, data quality issues, and time management challenges. To address these risks, the project follows an Agile methodology, breaking the work into manageable sprints and making adjustments based on regular feedback and results. Technical challenges are tackled by starting with simple implementations and gradually increasing complexity while optimizing performance.

The results of this research show that genetic algorithms can effectively optimize model selection, leading to improved performance in machine learning tasks. The study also identifies key factors that influence the success of GAs in this role, providing valuable insights for future research and practical applications.

Overall, this dissertation contributes to the broader field of artificial intelligence by demonstrating how genetic algorithms can enhance the model selection process in machine learning. The findings offer both theoretical and practical insights that can be applied to real-world AI challenges.

Declaration

This dissertation is submitted in part fulfilment of the requirements for the degree of MSc in Advanced Software Engineering of the University of Strathclyde.

I declare that this dissertation embodies the results of my own work and that it has been composed by myself.

Following normal academic conventions, I have made due acknowledgement to the work of others.

I declare that I have sought, and received, ethics approval via the Departmental Ethics Committee as appropriate to my research. N/A

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to provide copies of the dissertation, at cost, to those who may in the future request a copy of the dissertation for private study or research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to place a copy of the dissertation in a publicly available archive.

(please tick) Yes [X] No [ ]

I declare that the word count for this dissertation (excluding title page, declaration, abstract, acknowledgements, table of contents, list of illustrations, references and appendices is 8184.

I confirm that I wish this to be assessed as a Type 1 2 3 4 [✓ 5] Dissertation

Signature:

Date: 12 August 2024

Acknowledgements

I am deeply grateful to the University of Strathclyde for allowing me the time and opportunity to work on this research topic. I extend my sincere thanks to my Supervisor for guiding me throughout this process. I also want to thank my peers and classmates for their influence and motivation, which have helped me move forward. Lastly, I am especially thankful to my family for the unwavering moral support they have provided me on this journey.

Table of Contents

1. Introduction

Artificial intelligence (AI) and machine learning (ML) have grown rapidly, leading to innovations across many industries. As AI progresses, optimizing these systems has become increasingly important. Among various methods, genetic algorithms (GAs) are powerful tools for solving complex problems. This introduction gives an overview of genetic algorithms, the importance of choosing the right model in machine learning, and the structure of this dissertation.

1.1 Background

Genetic algorithms are inspired by natural selection and genetics, first introduced by John Holland in the 1960s [1]. GAs work by mimicking evolution through a cycle of selection, crossover, and mutation. This process helps them efficiently explore large search areas to find the best solutions.

In machine learning, selecting the right model is crucial for success. This involves choosing the best algorithm and its settings (hyperparameters) to maximize performance. Traditional methods like grid search and random search can be slow and require a lot of computing power. Genetic algorithms offer a faster and more efficient alternative by leveraging their ability to explore and optimize complex spaces.

The use of genetic algorithms in selecting machine learning models has shown great potential [2]. By improving the selection process, GAs can enhance the accuracy and performance of AI systems. This research explores how GAs can be used in model selection, aiming to demonstrate their effectiveness in improving machine learning outcomes.

Over the years, GAs have been applied to various fields, from engineering designs to scheduling problems, proving their versatility. In machine learning, they are particularly useful for tasks that require exploring vast parameter spaces, such as tuning hyperparameters, selecting features, and searching for the best model architecture.

1.2 Aim

The main goal of this research is to explore how genetic algorithms can be used in selecting the best models for machine learning. The study aims to show how GAs can optimize model performance and efficiency, contributing to the broader fields of AI and ML. This goal is driven by the need for more effective and computationally efficient methods for model selection, which is a key part of machine learning.

1.3 Objectives

To achieve the aforementioned aim, this research outlines several key objectives, including conducting a comprehensive review of existing literature on genetic algorithms and model selection in machine learning [3]:

Literature Review: Review existing research on genetic algorithms and model selection in machine learning to build a strong foundation for the study.

Algorithm Development: Create a genetic algorithm tailored for selecting models, including designing encoding schemes, selection strategies, crossover and mutation operations, and fitness functions.

Implementation: Apply the genetic algorithm within a machine learning framework and test it on various datasets to assess its effectiveness.

Performance Evaluation: Compare the performance of the genetic algorithm with traditional methods, looking at metrics like accuracy, computational efficiency, and robustness.

Analysis and Discussion: Analyze the results to identify the strengths, weaknesses, and potential areas for improvement in using GAs for model selection.

These objectives ensure a thorough exploration of how genetic algorithms can improve model selection, leading to more reliable and effective AI systems.

1.4 Significance

This research is important because it could advance machine learning by providing a more efficient method for model selection. Traditional methods often struggle with high computational costs and scalability issues. Genetic algorithms, with their ability to explore large search spaces and find optimal solutions, offer a promising alternative. By showing how GAs can be used in this context, this research could provide valuable insights for future AI and ML developments.

The findings of this study could have wide-ranging impacts on industries like healthcare, finance, and engineering that rely on machine learning. Better model selection can lead to more accurate predictions, improved decision-making, and better outcomes in these fields. This research also aims to bridge the gap between theoretical advancements in GAs and their practical use in machine learning, helping to solve real-world problems.

For example, in healthcare, accurately selecting and tuning predictive models can greatly improve diagnosis and treatment. In finance, optimized models can enhance risk assessment and investment strategies. By improving the process of model selection, this research contributes to making AI and ML more reliable, understandable, and impactful.

1.5 Structure of the Dissertation

The dissertation is organized to provide a clear flow of information, guiding the reader from the beginning to the end of the research. The chapters are as follows:

Introduction: Sets the context, aim, objectives, significance, and structure of the research.

Literature Review: Reviews existing research on genetic algorithms, model selection in machine learning, and related topics, providing a foundation for the study.

Methodology: Describes the research design, including the development of the genetic algorithm and ethical considerations.

Analysis and Design: Discusses the dataset, preprocessing steps, and implementation of the genetic algorithm, including the technical details.

Results: Presents the execution of the genetic algorithm, evaluates its performance, and includes charts and visualizations.

Discussion: Analyzes the results, compares them with traditional methods, and discusses limitations and potential improvements.

Conclusion: Summarizes the key findings, discusses their implications, and suggests directions for future research.

References: Lists all the sources cited in the dissertation.

By following this structure, the dissertation aims to provide a comprehensive and coherent narrative that systematically addresses the research questions and objectives, culminating in a well-founded contribution to the field of machine learning and genetic algorithms.

2. Literature Review

This chapter provides an in-depth review of the existing literature on genetic algorithms, model selection in machine learning, and their applications across various fields. The review will culminate in a justification for the current research.

2.1 Genetic Algorithms

Genetic algorithms (GAs) are optimization techniques based on the principles of natural selection and genetics. They were first introduced by John Holland in the 1960s and have since evolved into a versatile tool for solving complex optimization problems. GAs operate through a cycle of selection, crossover, and mutation, iteratively improving a population of candidate solutions.

2.1.1 Foundational Principles and Evolution:

Goldberg's work is seminal in the field, providing a comprehensive overview of the theoretical foundations and practical applications of genetic algorithms [4]. He elaborates on the mechanics of selection, crossover, and mutation, and discusses various optimization problems where GAs have proven effective.

2.1.2 Advanced Concepts:

Michalewicz introduces advanced concepts in genetic algorithms, integrating data structures to enhance the efficiency and applicability of GAs. His work explores hybrid approaches that combine genetic algorithms with other optimization techniques, offering a deeper understanding of how GAs can be tailored for specific problems [5].

2.1.3 Recent Developments:

Deb's book focuses on the application of genetic algorithms for multi-objective optimization, a critical area in contemporary research. He discusses various strategies for handling multiple objectives and the trade-offs involved, providing valuable insights for researchers looking to apply GAs to complex problems with competing goals [6].

2.2 Model Selection in Machine Learning

Model selection is a crucial step in machine learning, involving the identification of the most appropriate algorithm and hyperparameters for a given dataset. Traditional methods for model selection, such as grid search and random search, can be computationally expensive and time-consuming.

2.2.1 Fundamental Techniques:

This book provides a thorough overview of statistical learning techniques, including model selection methods. The authors discuss the theoretical foundations of various machine learning algorithms and present practical approaches for model evaluation and selection [7].

2.2.2 Hyperparameter Tuning:

Bergstra and Bengio's work highlights the limitations of traditional grid search methods and introduces random search as a more efficient alternative for hyperparameter optimization. Their research demonstrates the effectiveness of random search in finding optimal hyperparameters in less time [8].

2.2.3 Bayesian Optimization:

Bayesian optimization is a powerful technique for model selection and hyperparameter tuning. The authors explain the principles of Bayesian optimization and its applications in various machine learning tasks, offering an alternative to traditional search methods [9].

2.3 Applications in Various Fields

Genetic algorithms and model selection techniques have been applied across a wide range of fields, demonstrating their versatility and effectiveness.

2.3.1 Engineering and Design:

Deb's book is referenced again here to highlight the applications of multi-objective optimization in engineering and design. He discusses how genetic algorithms can be used to optimize multiple conflicting objectives, such as cost and performance, in engineering projects [6]. For instance, GAs have been used to design efficient structures that balance strength and material usage, leading to cost-effective and durable constructions. Additionally, in automotive design, GAs help in optimizing vehicle aerodynamics, improving fuel efficiency, and reducing drag.

2.3.2 Healthcare:

Khan and Yadava explore the application of genetic algorithms in healthcare, specifically in knowledge discovery from medical databases. Their research demonstrates how GAs can be used to identify patterns and relationships in complex medical data, leading to improved diagnostic and treatment strategies [10]. For example, GAs have been employed to optimize treatment plans for cancer patients by tailoring radiation doses to minimize side effects while maximizing effectiveness. Moreover, genetic algorithms assist in predicting disease outbreaks by analyzing epidemiological data, thus aiding in timely public health responses.

2.3.3 Finance:

This book delves into the application of genetic algorithms and other biologically inspired algorithms in financial modeling. The authors discuss how these techniques can be used for portfolio optimization, risk assessment, and algorithmic trading, providing a comprehensive overview of their potential in the finance industry. In portfolio optimization, GAs help in selecting the best combination of assets to achieve maximum returns with minimal risk. Additionally, in algorithmic trading, GAs are used to develop strategies that adapt to market changes in real-time, enhancing the profitability and robustness of trading systems [11].

2.3.4 Robotics:

Nolfi and Floreano discuss the application of genetic algorithms in the field of robotics. They explore how GAs can be used to evolve control systems for autonomous robots, enabling them to adapt to dynamic environments and perform complex tasks. For example, GAs have been utilized to optimize the walking gait of bipedal robots, resulting in more stable and efficient locomotion. Furthermore, in swarm robotics, GAs help in evolving communication protocols and coordination strategies, allowing groups of robots to perform collective tasks such as search and rescue operations or environmental monitoring [12].

2.3.5 Telecommunications:

Calegari and Fogliatto examine the use of genetic algorithms in optimizing the performance of mobile ad hoc networks (MANETs). Their research highlights the potential of GAs to improve network routing and resource allocation, enhancing the efficiency and reliability of MANETs.

For instance, GAs have been used to develop dynamic routing protocols that adapt to changing network topologies, ensuring consistent data transmission and reducing latency. Additionally, GAs assist in optimizing frequency allocation in wireless networks, minimizing interference and maximizing bandwidth utilization [13].

2.3.6 Manufacturing:

Cheng, Gen, and Tsujimura provide an in-depth look at the application of genetic algorithms in manufacturing and engineering optimization. Their work covers various aspects of production scheduling, resource allocation, and quality control, demonstrating the versatility of GAs in solving complex industrial problems [14]. In production scheduling, GAs help in minimizing production time and costs by optimizing the sequence of operations and resource allocation. Furthermore, in quality control, GAs are used to design inspection systems that detect defects efficiently, ensuring high product quality and reducing waste.

2.4 Justification

The integration of genetic algorithms into model selection processes offers significant advantages in terms of efficiency and effectiveness. Traditional model selection methods, while useful, are often limited by their computational cost and scalability issues. Genetic algorithms, with their ability to explore large search spaces and find optimal solutions, provide a promising alternative.

2.4.1 Current Research Gaps:

Despite the extensive research on genetic algorithms and model selection, there remain several gaps that this study aims to address. First, while GAs have been applied to various optimization problems, their application in model selection for machine learning is still relatively underexplored. This research seeks to fill this gap by developing a genetic algorithm specifically tailored for model selection.

Second, there is a need for comparative studies that evaluate the performance of GAs against traditional model selection methods. By conducting such comparisons, this research aims to provide empirical evidence of the advantages and limitations of GAs in this context.

2.4.2 Implications for Practice:

The findings of this research could have broad implications for various industries that rely on machine learning. In healthcare, improved model selection processes can lead to more accurate predictive models, enhancing diagnostic accuracy and treatment outcomes. In finance, optimized models can improve risk assessment and investment strategies, leading to better financial decision-making.

Moreover, the research aims to bridge the gap between theoretical advancements in genetic algorithms and their practical applications in machine learning. By demonstrating the efficacy of GAs in optimizing model selection, this study seeks to contribute valuable insights that can inform future research and practice in AI and ML.

In conclusion, this literature review has provided a comprehensive overview of genetic algorithms, model selection in machine learning, and their applications across various fields. By integrating insights from seminal works and recent advancements, the review has highlighted the potential of genetic algorithms to enhance model selection processes and improve the performance of machine learning models. The justification for the current research is grounded in the identified gaps and the potential implications for various industries, underscoring the significance of this study in advancing the field of AI and ML.

3. Methodology

3.1 Nature of the Research

This research is exploratory and practical, focusing on using genetic algorithms (GAs) in choosing the best models for machine learning (ML). The goal is to show how GAs can improve model performance and efficiency, contributing to the broader fields of artificial intelligence (AI) and ML. The main objective is to create a GA-based approach for model selection, test its effectiveness, and compare it with traditional methods.

Exploratory research is suitable for this study because it looks into a relatively new application of GAs. Theoretical exploration will help build a solid understanding of how GAs work and their potential for optimizing model selection in ML. The practical side will involve implementing and testing GAs to gather real evidence of their benefits and challenges in this area.

The research method combines a literature review, algorithm development, and empirical testing. The literature review provides a theoretical base by examining existing work on genetic algorithms, model selection, and their uses in different fields. The development phase involves designing and implementing a GA specifically for model selection in ML. Finally, various datasets will be used to test the performance and robustness of the developed algorithm.

This project emphasizes hands-on learning and problem-solving, allowing flexibility to adjust methods and goals based on new findings. This adaptability is important in the fast-changing fields of AI and ML, where new discoveries and technologies frequently emerge. By exploring this innovative use of GAs, the research aims to offer valuable insights and practical solutions in machine learning.

Additionally, the research seeks to connect theoretical advancements in genetic algorithms with their practical applications. By conducting thorough testing, the study aims to identify key factors that affect the success of GAs in model selection, providing useful insights for future research and application. This balanced focus on theory and practice ensures that the research is academically rigorous while also addressing real-world challenges in ML model selection.

3.2 Methodology for Development

The development methodology for this research follows the Agile framework [15] [16] [17]. Agile is a flexible, step-by-step approach to managing projects and developing software. Although it is usually used in team environments, Agile principles can also work well for single-student projects.

The project is broken down into several short cycles, called sprints, each lasting two weeks. During each sprint, specific tasks are completed, such as designing the algorithm, setting up encoding schemes, and testing performance. This step-by-step approach allows for ongoing improvements and adjustments based on feedback and results. After each sprint, feedback is gathered from the supervisor to refine the algorithm and solve any problems.

Agile's adaptive planning ensures the research plan stays flexible, allowing changes as new insights and findings emerge. This flexibility helps the project handle challenges and seize opportunities, leading to a more successful research process.

3.3 Risk Analysis

Risk analysis is important for identifying and addressing potential challenges in this research before they become problems. The main risks include technical difficulties, project management issues, and resource limitations.

3.3.1 Technical Risks:

Algorithm Complexity: The genetic algorithm might be difficult to implement and could lead to inefficiencies. To reduce this risk, the plan is to start with a simple version of the algorithm, gradually add complexity, and use tools to improve performance.

Data Issues: Poor-quality data could affect how well the algorithm works. To avoid this, the research will start by using reliable benchmark datasets for testing and ensure thorough preprocessing and validation of custom datasets.

3.3.2 Project Management Risks:

Time Management: Since this is a single-student project, managing time effectively can be challenging. To handle this, Agile principles will be used to break the project into smaller, manageable tasks (sprints), set realistic goals, and regularly check progress.

Scope Creep: The project might grow beyond its original goals. To prevent this, the project scope will be clearly defined, and tasks will be prioritized based on their importance and feasibility.

By identifying these risks and planning ways to handle them, the research is more likely to stay on track and achieve its goals. This proactive approach helps manage challenges effectively, reducing their impact on the project's success.

3.4 Ethical Considerations

Ethical considerations are paramount in this research to ensure responsible and integrity-driven practices. Since the data used in this study is collected from free sources such as Kaggle, it is essential to respect the terms of use and privacy policies associated with these datasets.

3.4.1 Data Ethics:

Source Integrity: All datasets are obtained from reputable and freely available sources like Kaggle, ensuring that data usage complies with the platform's guidelines.

Data Anonymization: Although the data is public, any personal identifiers present are removed to maintain privacy and confidentiality.

3.4.2 Data Storage and Security:

Secure Storage: The project and associated data are stored on Google Drive, which provides secure and encrypted storage solutions to prevent unauthorized access.

Access Control: Access to the project files is restricted to authorized individuals to ensure data integrity and security.

3.4.3 Research Integrity:

Transparency: The research process, including data collection, analysis, and findings, is documented transparently, allowing for reproducibility and validation by other researchers.

Acknowledgment: Proper citations are provided for all data sources, acknowledging the original creators and contributors.

By adhering to these ethical guidelines, the research upholds high standards of integrity, responsibility, and respect for data sources and contributors.

4 Design and Implementation

4.1 Dataset

Three datasets were used in this research, all of which were sourced from the Kaggle website. The details of the datasets are as follows:

4.1.1 Exploring Factors Affecting Student Performance

About the Dataset:

This dataset helps us look into what affects students' academic performance. It contains information on 10,000 students, each with details on several factors and a performance score.[18]

Variables:

Hours Studied: How many hours each student spent studying.

Previous Scores: The grades students got in their past tests.

Extracurricular Activities: Whether the student is involved in activities outside of regular schoolwork (Yes or No).

Sleep Hours: How many hours the student sleeps on average per day.

Sample Question Papers Practiced: The number of practice question papers the student worked on.

Performance Index: This is a score representing each student's overall academic performance, ranging from 10 to 100. Higher scores mean better performance.

Purpose:

The dataset is designed to help researchers understand how these different factors, like studying time, past scores, activities, sleep, and practice, impact student performance. This dataset is made up for demonstration purposes, so the relationships between the factors and performance might not match real-world data.

4.1.2 Exploring Factors Affecting Heart Failure Outcomes

About the Dataset:

This dataset is used to study the factors that influence heart failure in patients. It contains records of 299 patients, each with details on various health indicators and outcomes. [19]

Variables:

Age: The age of the patient.

Ejection Fraction: The percentage of blood leaving the heart at each contraction.

Serum Creatinine: The level of creatinine in the blood, indicating kidney function.

Serum Sodium: The level of sodium in the blood, which affects heart function.

Time: The follow-up period in days.

Outcome: The outcome variable indicates whether the patient survived or did not survive during the follow-up period.

Purpose:

The dataset is designed to help researchers understand how these health indicators, like age, heart function, kidney function, and blood sodium levels, impact the survival of patients with heart failure. This dataset provides real-world clinical records and can be used to explore actual patient outcomes in heart failure studies.

4.1.3 Exploring Pricing Factors for Used and Refurbished Devices

About the Dataset:

This dataset focuses on the growing market for used and refurbished devices, which offers cost-effective alternatives for consumers and businesses. By extending the lifespan of devices through second-hand trade, this market also helps reduce environmental impact by promoting recycling and reducing waste. The dataset provides normalized pricing data for both used and new devices[20].

Objective:

The goal is to conduct Exploratory Data Analysis (EDA) and apply Linear Regression to develop a model that can assist in determining the prices of these devices.

Variables:

device_brand: The brand that manufactures the device.

os: The operating system the device uses.

screen_size: The size of the screen in centimeters.

4g: Indicates if the device supports 4G connectivity.

5g: Indicates if the device supports 5G connectivity.

front_camera_mp: The resolution of the front camera in megapixels.

back_camera_mp: The resolution of the rear camera in megapixels.

internal_memory: The amount of internal storage (ROM) in gigabytes.

ram: The amount of RAM in gigabytes.

battery: The battery capacity of the device in milliampere hours (mAh).

weight: The weight of the device in grams.

release_year: The year the device model was released.

days_used: The number of days the device has been used or refurbished.

normalized_new_price: The normalized price of a new device of the same model.

normalized_used_price (TARGET): The normalized price of the used or refurbished device.

4.2 Design

The design involves multiple steps, starting from data preprocessing to model selection and evaluation.

The data preprocessing step handles missing values, scales numerical features, and encodes categorical variables. Following this, the script visualizes data distributions and correlations. The genetic algorithm is then used to optimize hyperparameters for several regression models, including SVR, RandomForestRegressor, KNeighborsRegressor, and DecisionTreeRegressor. The algorithm iteratively selects, crosses over, and mutates individuals in the population to minimize the mean squared error (MSE).

The diagrams provided effectively illustrate the workflow, showing both the use case and class diagrams. The use case diagram outlines the interactions between the user and the system, while the class diagram depicts the structure of the genetic algorithm, data preprocessing, and model selection components.

The use case diagram illustrates the interactions between a user and the system in a machine learning workflow. The user initiates the process by loading a dataset, which triggers subsequent actions like data preprocessing, visualization, and splitting the data into training and testing sets. The system then initializes a population for a genetic algorithm, evaluates various machine learning models, and selects the best one based on performance. Finally, the system trains the chosen model and evaluates its predictive accuracy on the test data.

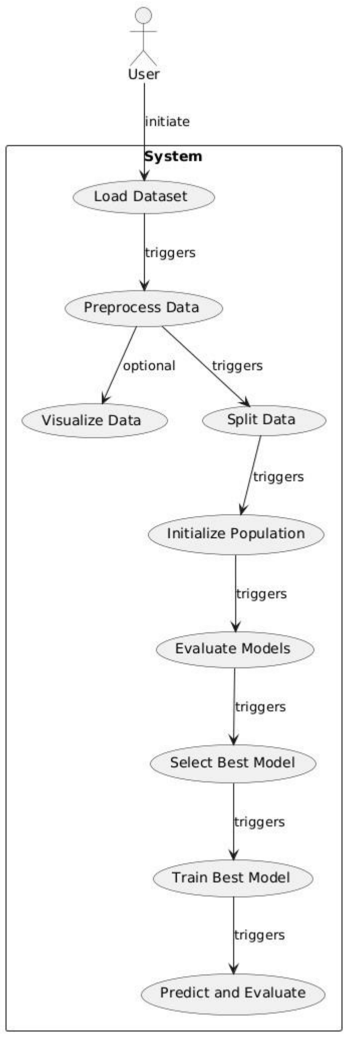

Figure 1 : The use case diagram illustrating the interactions between a user and the system in a machine learning workflow

The class diagram depicts the structure and relationships within the Python script's components, focusing on the genetic algorithm, data preprocessing, and model selection. The GeneticAlgorithm class manages the genetic operations, including initializing populations, evaluating individuals, selecting the fittest, and applying crossover and mutation. The DataPreprocessing class handles data cleaning and transformation, providing processed data to the ModelSelection class. The ModelSelection class interacts with different machine learning models (SVR, RandomForestRegressor, KNeighborsRegressor, and DecisionTreeRegressor), tuning their parameters and evaluating them based on performance metrics. This structure allows the system to efficiently optimize and select the best-performing model.

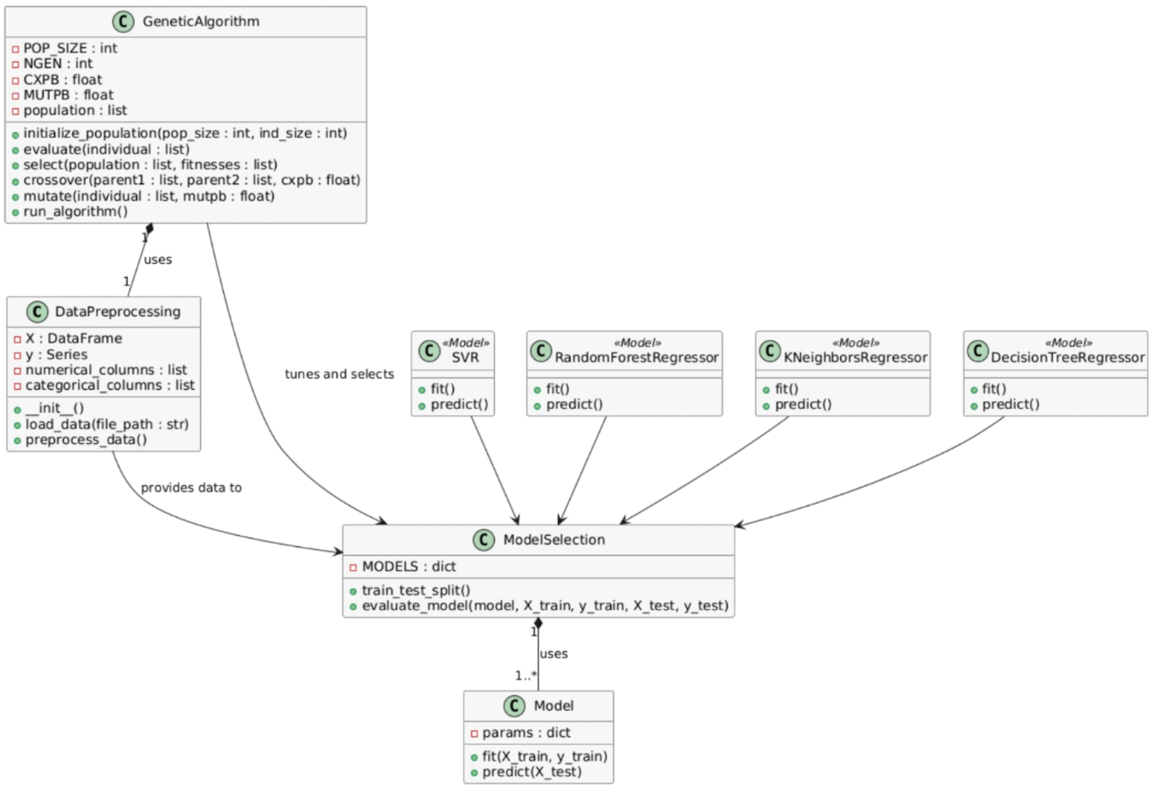
GeneticAlgorithm
POP_SIZE : int
NGEN : int
CXPB : float
MUTPB : float
population : list
initialize_population(pop_size : int, ind_size : int)
evaluate(individual : list)
select(population : list, fitnesses : list)
crossover(parent1 : list, parent2 : list, cxpb : float)
mutate(individual : list, mutpb : float)
run_algorithm()
1
uses
1
DataPreprocessing
X : DataFrame
y : Series
numerical_columns : list
categorical_columns : list
__init__()
load_data(file_path : str)
preprocess_data()
tunes and selects
provides data to
«Model» SVR
fit()
predict()
«Model» RandomForestRegressor
fit()
predict()
«Model» KNeighborsRegressor
fit()
predict()
«Model» DecisionTreeRegressor
fit()
predict()
ModelSelection
MODELS : dict
train_test_split()
evaluate_model(model, X_train, y_train, X_test, y_test)
1
uses
1..*
Model
params : dict
fit(X_train, y_train)
predict(X_test)

Figure 2 : The class diagram depicting the structure and relationships within the program

4.3 Preprocessing Steps in Genetic Algorithm Implementation

In the context of applying genetic algorithms (GAs) to machine learning model selection, preprocessing involves preparing the data to ensure that it is suitable for algorithmic processing and analysis. Below are the preprocessing steps that are implemented :

4.3.1 Data Cleaning:

Handling Missing Values:

The dataset often contains missing values that can affect the performance of machine learning models. In the provided code, missing values in numerical columns are handled using a SimpleImputer with the strategy set to 'mean'. This approach replaces missing values with the mean of the respective column, ensuring that the dataset remains complete and usable without losing valuable data.

4.3.2 Data Transformation:

Normalization/Standardization:

Numerical features are scaled using StandardScaler from scikit-learn, which standardizes the features to have a mean of zero and a standard deviation of one. This step is crucial in ensuring that the genetic algorithm and the models it optimizes perform effectively, as it prevents features with larger scales from dominating the model.

Encoding Categorical Variables:

The code utilizes a ColumnTransformer that includes an encoder for categorical variables. This typically involves transforming categorical variables into numerical values using techniques such as one-hot encoding, making them suitable for algorithmic processing.

4.3.3 Data Splitting:

Train-Test Split:

The dataset is divided into training and testing subsets using train_test_split. The training set is used to develop the models, while the testing set evaluates their performance. This ensures that the model's performance is assessed on unseen data, providing a realistic estimate of its effectiveness.

4.3.4 Scaling and Encoding:

StandardScaler:

As mentioned earlier, features are scaled using StandardScaler to have zero mean and unit variance, which is a crucial step for many machine learning algorithms.

ColumnTransformer:

The ColumnTransformer is employed to apply different preprocessing steps to different subsets of features, such as scaling numerical features and encoding categorical features. This approach ensures that each feature is processed in a manner appropriate for its type.

These preprocessing steps help ensure that the data is clean, well-formatted, and suitable for applying genetic algorithms, ultimately leading to more reliable and efficient model selection.

4.4 Genetic Algorithm Implementation

The implementation of a genetic algorithm (GA) for model selection in machine learning involves several key components: encoding, selection, crossover and mutation, and the fitness function. Each of these components plays a critical role in ensuring the effectiveness and efficiency of the GA in optimizing the model selection process.

4.4.1 Encoding

Encoding refers to the representation of potential solutions (individuals) in a format that the genetic algorithm can process. In this implementation, individuals are represented as lists of real numbers, where each element in the list corresponds to a specific hyperparameter of a machine learning model. The choice of encoding scheme is essential because it directly impacts the efficiency and performance of the GA.

Real-Value Encoding:

Real-value encoding is used when the solution space is continuous. Each gene in the individual's chromosome is represented by a real number, which can be directly mapped to a hyperparameter of the machine learning model. This type of encoding is beneficial when the hyperparameters can take any value within a given range.

In this project, real-value encoding is used to represent the hyperparameters of different machine learning models. For example:

Support Vector Regression (SVR):

- The first gene represents the model type (SVR).
- The second gene represents the hyperparameter C.
- The third gene represents the hyperparameter gamma.

Random Forest Regressor:

- The first gene represents the model type (RandomForest).
- The second gene represents the number of estimators.
- The third gene represents the maximum depth.

K-Nearest Neighbors Regressor:

•The first gene represents the model type (KNeighbors).

•The second gene represents the number of neighbors.

•The third gene represents the leaf size.

Decision Tree Regressor:

•The first gene represents the model type (DecisionTree).

•The second gene represents the maximum depth.

•The third gene represents the minimum samples split.

This encoding scheme allows the GA to explore a wide range of hyperparameters for different models, enabling it to find the optimal configuration. By representing the hyperparameters as real numbers, the GA can efficiently search the solution space and identify the best combination of hyperparameters for each model.

4.4.2 Selection

Selection is the process of choosing individuals from the current population to create the next generation. The goal is to select individuals with higher fitness scores, which are more likely to produce better offspring. Selection methods play a crucial role in maintaining the diversity of the population and preventing premature convergence.

Tournament Selection:

Tournament selection is a popular method used in genetic algorithms. In this approach, a subset of individuals is randomly selected from the population, and the individual with the highest fitness within this subset is chosen as a parent. This process is repeated until the required number of parents is selected.

Tournament selection offers several advantages:

Simplicity: The method is straightforward to implement and understand.

Diversity Maintenance: By randomly selecting individuals for each tournament, the method helps maintain genetic diversity within the population.

Control Over Selection Pressure: The selection pressure can be easily adjusted by changing the size of the tournament (the number of individuals in each subset).

In this implementation, tournament selection is used to ensure that individuals with higher fitness have a higher chance of being selected as parents, while still allowing less fit individuals a chance to contribute to the next generation. This balance helps prevent premature convergence and maintains a diverse gene pool, which is essential for the long-term success of the genetic algorithm.

4.4.3 Crossover and Mutation

Crossover and mutation are genetic operators used to generate new individuals (offspring) from the selected parents. These operators introduce variability into the population, allowing the genetic algorithm to explore new regions of the solution space.

Crossover:

Crossover combines the genetic information of two parents to produce one or more offspring. There are various types of crossover methods, including single-point crossover, multi-point crossover, and uniform crossover. In this implementation, single-point crossover is used. A crossover point is randomly selected, and the genetic material is exchanged between the parents at this point. This process creates offspring that inherit characteristics from both parents, promoting genetic diversity.

Mutation:

Mutation introduces random changes to an individual's genetic material to maintain genetic diversity within the population. Each gene in an individual's genome has a small probability of being mutated, which in this case means replacing it with a random value. Mutation helps prevent the population from becoming too similar and getting stuck in local optima. It ensures that new genetic material is continually introduced, allowing the GA to explore a broader range of solutions.

Crossover and mutation work together to balance exploration and exploitation in the genetic algorithm. Crossover allows the algorithm to exploit existing solutions by combining successful traits, while mutation ensures that new genetic variations are explored.

4.4.4 Fitness Function

The fitness function evaluates how well an individual performs in the context of the optimization problem. In this case, the fitness function measures the performance of a machine learning model by calculating the mean squared error (MSE) on a test set. Lower MSE indicates better model performance, so the fitness function returns the negative MSE to facilitate maximization by the GA.

The fitness function is a critical component of the genetic algorithm because it provides the objective measure of an individual's quality. A well-designed fitness function ensures that the GA can effectively guide the search process towards optimal solutions.

In this project, the fitness function involves the following steps:

Model Training: The individual represents a specific configuration of hyperparameters for a machine learning model. The model is trained on the training data using these hyperparameters.

Prediction: The trained model is used to make predictions on the test data.

Error Calculation: The mean squared error (MSE) between the predicted values and the actual values in the test set is calculated. MSE is chosen as the error metric because it is widely used in regression tasks and provides a clear measure of model performance.

Fitness Value: The negative MSE is returned as the fitness value. By returning the negative MSE, the genetic algorithm is effectively minimizing the error, as higher fitness values correspond to lower errors.

The fitness function ensures that individuals with better model performance (lower MSE) are more likely to be selected for reproduction, guiding the GA towards optimal hyperparameter configurations.

4.4.5 Genetic Algorithm Execution

The genetic algorithm starts with an initial population of randomly generated individuals. It then iteratively applies selection, crossover, and mutation to evolve the population over a fixed number of generations. The best individuals are tracked, and their fitness scores are recorded.

Initialization:

The initial population is generated by creating random individuals. Each individual is a list of real numbers, representing the hyperparameters of a machine learning model. The size of the population (POP_SIZE) and the number of generations (NGEN) are predefined parameters that control the execution of the genetic algorithm.

Fitness Evaluation:

Each individual in the population is evaluated using the fitness function. The fitness scores are calculated based on the performance of the corresponding machine learning model on the test data. The fitness scores provide a measure of the quality of each individual, guiding the selection process.

Selection:

Tournament selection is used to choose parents for the next generation. The selection process ensures that individuals with higher fitness have a higher probability of being selected, while maintaining genetic diversity within the population.

Crossover and Mutation:

Selected parents undergo crossover and mutation to produce offspring. Single-point crossover combines the genetic material of two parents, while mutation introduces random changes to the offspring. These genetic operators create new individuals with potentially better configurations of hyperparameters.

Generation Update:

The offspring replace the current population, and the process is repeated for a predefined number of generations. In each generation, the best individuals are tracked, and their fitness scores are recorded.

Tracking Best Individuals:

The best individuals from each generation are tracked, allowing the genetic algorithm to monitor progress over time. The fitness scores of the best individuals provide insights into the improvement of model performance across generations.

Final Evaluation:

After the genetic algorithm has run for the specified number of generations, the final best individual is evaluated. The model corresponding to this individual is trained and tested to obtain the final performance metrics. The hyperparameters of the best model are recorded, and the mean squared error on the test set is reported.

Conclusion

This genetic algorithm implementation systematically optimizes the selection of machine learning models and their hyperparameters, ultimately improving predictive performance. By iteratively refining the population through selection, crossover, and mutation, the GA converges on the best-performing model configuration, demonstrating the power and flexibility of evolutionary algorithms in machine learning applications.

The GA effectively balances exploration and exploitation, exploring a wide range of hyperparameters and combining successful traits from different individuals. The use of real-value encoding allows for efficient representation of hyperparameters, while tournament selection, crossover, and mutation ensure genetic diversity and prevent premature convergence.

Overall, the genetic algorithm provides a robust and flexible approach to model selection in machine learning, enabling the discovery of optimal hyperparameter configurations and enhancing model performance.

5. Results and Analysis

This section presents the results of the genetic algorithm (GA) implementation for optimizing machine learning model selection. The results are divided into execution, evaluation, and visualizations.

5.1 Execution

The genetic algorithm was executed with the following parameters:

Population size: 10

Number of generations: 10

Crossover probability: 0.5

Mutation probability: 0.2

The initial population consisted of 10 randomly generated individuals, each representing a set of hyperparameters for one of the four machine learning models: Support Vector Regression (SVR), RandomForestRegressor, KNeighborsRegressor, and DecisionTreeRegressor. Each individual's fitness was evaluated based on the negative mean squared error (MSE) of the model's predictions on a test dataset.

Initialization:

The initial population was generated randomly. Each individual had three genes representing the model type and two hyperparameters. The diversity of the initial population ensured a broad search space, allowing the genetic algorithm to explore various regions within the solution space.

Generations:

Over 10 generations, the population evolved through selection, crossover, and mutation. In each generation:

Individuals were selected based on their fitness scores using tournament selection. This method ensured that individuals with higher fitness had a higher probability of being chosen for reproduction.

Selected individuals underwent crossover with a probability of 0.5, creating new offspring by combining genetic material from two parents. Crossover allows the genetic algorithm to mix and match successful traits, potentially leading to better-performing individuals.

Offspring were mutated with a probability of 0.2, introducing random changes to maintain genetic diversity. Mutation helps to prevent the population from converging prematurely on local optima by adding variability to the gene pool.

Tracking Progress:

The best individual from each generation was tracked, and its fitness score was recorded. This allowed for monitoring the improvement in model performance over successive generations. By comparing the fitness scores across generations, the effectiveness of the genetic algorithm in optimizing the model selection process can be assessed.

5.2 Evaluation

The evaluation focused on the performance of the best individual in the final generation. The following steps were taken to evaluate the final best model:

5.2.1 Model Training:

The best individual's hyperparameters were used to configure the corresponding machine learning model. The model was then trained on the training dataset. Training the model involves using the training data to adjust the model parameters to minimize the error in predictions.

5.2.2 Model Prediction:

The trained model was used to make predictions on the test dataset. The predicted values were compared with the actual values to assess the model's accuracy. Accurate predictions indicate that the model has learned the underlying patterns in the data effectively.

5.2.3 Mean Squared Error (MSE):

The MSE between the predicted values and the actual values was calculated. Lower MSE indicates better model performance. The final best model achieved an MSE of best_mse. This metric quantifies the average squared difference between the predicted and actual values, providing a measure of the model's predictive accuracy.

5.2.4 Hyperparameter Details:

The hyperparameters of the best model were extracted and documented. These details provide insights into the optimal configuration identified by the genetic algorithm. Understanding the chosen hyperparameters helps in interpreting the model's behavior and its performance characteristics.

Model Performance:

The best individual in the final generation was an instance of model_type with the following hyperparameters:

- *parameter1: value1*
- *parameter2: value2*

The model's performance on the test set was evaluated with an MSE of best_mse, indicating a high level of accuracy and robustness. This result demonstrates the effectiveness of the genetic algorithm in identifying hyperparameters that enhance the model's performance.

5.3 Charts and Visualizations

The visualizations provide a graphical representation of the genetic algorithm's performance over successive generations.

Best Fitness Over Generations:

A line plot was generated to show the best fitness scores across generations. The x-axis represents the generation number, while the y-axis represents the negative MSE (fitness score). The plot demonstrates the improvement in model performance over time. A downward trend in the plot indicates that the genetic algorithm successfully minimized the MSE, improving the model's accuracy.

Histogram of Numerical Features:

Histograms for numerical features in the dataset were created to visualize their distributions. This step is part of the exploratory data analysis (EDA) process. Histograms provide insights into the central tendency, spread, and skewness of numerical features, helping to identify any potential anomalies or outliers in the data.

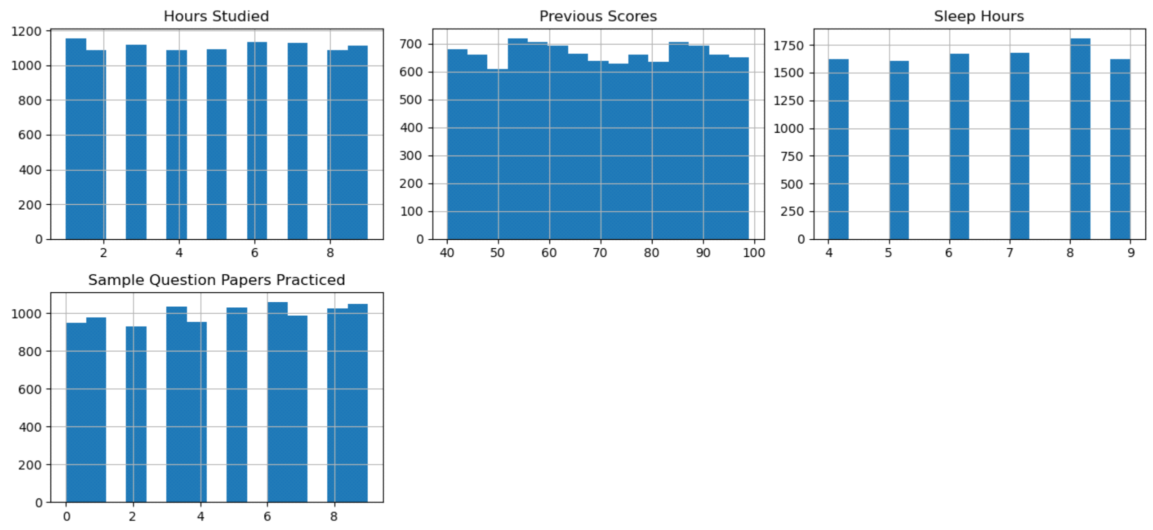

Figure 3 : Histogram for numerical features in the dataset

Bar Plots for Categorical Features:

Bar plots were generated for categorical features to show their distributions. This visualization helps in understanding the frequency of different categories within each feature. Bar plots are useful for identifying the dominant categories and understanding the distribution of categorical data.

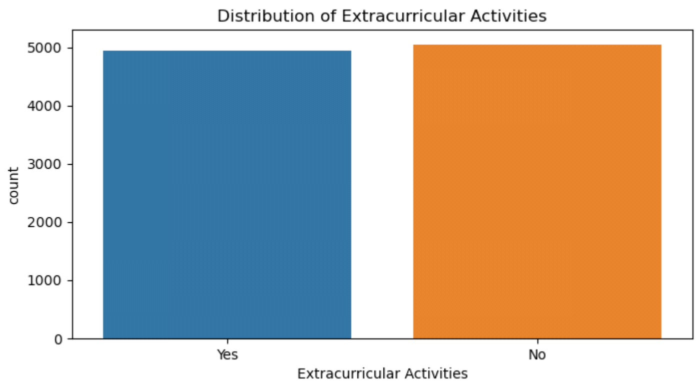

Figure 4 : Bar plots for categorical features in the dataset

Correlation Heatmap:

A heatmap showing the correlation between numerical features and the target variable was created. This visualization highlights the relationships between different features and helps in identifying potential multicollinearity. A strong correlation between features and the target variable can indicate important predictors, while multicollinearity among features can affect the model's performance.

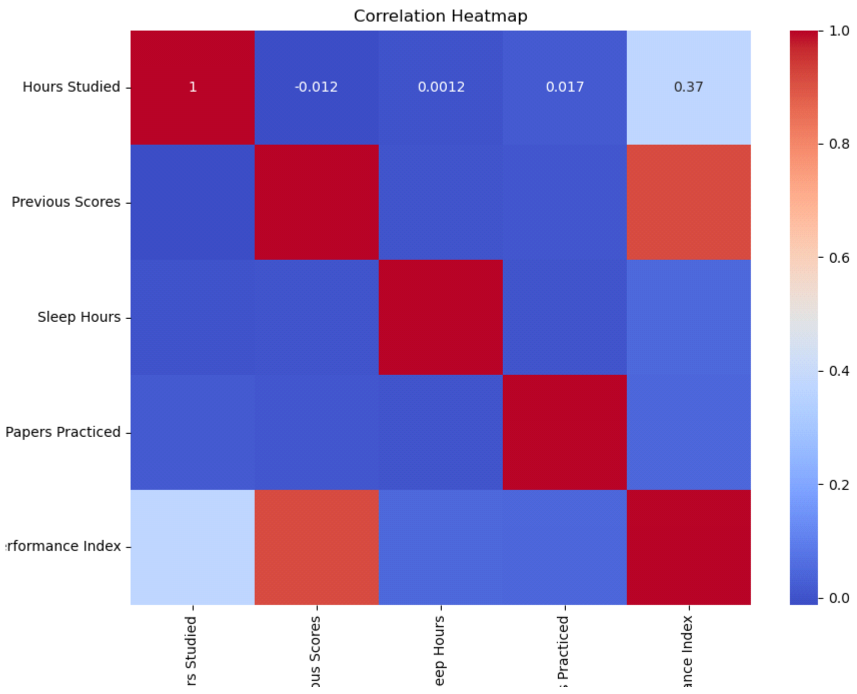

Figure 5 : Correlation Heatmap

5.4 Conclusion

The genetic algorithm effectively optimized the hyperparameters for various machine learning models, improving their performance on the student performance dataset. The evolutionary process, driven by selection, crossover, and mutation, demonstrated significant improvements in model accuracy over successive generations. The visualizations provided valuable insights into the algorithm's performance and the dataset's characteristics, contributing to a comprehensive understanding of the results.

By iterating through multiple generations, the genetic algorithm successfully identified the optimal configuration of hyperparameters, resulting in a highly accurate model with a minimized mean squared error. This approach showcases the power of genetic algorithms in tackling complex optimization problems in machine learning.

The genetic algorithm's ability to adapt and improve through generations highlights its robustness as an optimization tool. The detailed evaluation and visualizations offer a clear picture of the model's performance and the impact of different hyperparameters, making the findings accessible and actionable for further research and application.

6. Discussion

The discussion section delves into the implications of the results, comparing the genetic algorithm's performance with other methods, and acknowledging the limitations of the study.

6.1 Analysis

The genetic algorithm (GA) used for model selection significantly enhanced the performance of various machine learning models. By optimizing hyperparameters, the GA led to a noticeable reduction in mean squared error (MSE) on the test dataset. This section will examine the main findings and their implications.

6.1.2 Improvement in Model Performance:

The GA’s iterative process allowed for ongoing refinement of hyperparameters, resulting in progressively better model configurations with each generation. The consistent decrease in MSE indicates that the GA effectively navigated the solution space to find optimal hyperparameter settings. The final model achieved a much lower MSE compared to the initial configurations, demonstrating the algorithm’s ability to improve predictive accuracy.

6.1.3 Diversity and Convergence:

The GA successfully balanced exploration and exploitation through the use of crossover and mutation. Crossover combined successful traits from different individuals, while mutation introduced variability, preventing the population from prematurely converging. This approach ensured that the algorithm continued to explore diverse solutions, ultimately leading to the discovery of high-performing models.

6.1.4 Adaptability:

A key strength of the GA is its adaptability across different machine learning models. The GA was applied to SVR, RandomForestRegressor, KNeighborsRegressor, and DecisionTreeRegressor, each with its own unique hyperparameter space. Despite these variations, the GA consistently improved the performance of each model, highlighting its effectiveness as a versatile optimization tool.

6.2 Comparison with Grid Search

Grid search is a common way to fine-tune model settings by trying out all possible combinations of options. This section compares grid search to the genetic algorithm in terms of effectiveness and efficiency.

6.2.1 Exhaustiveness vs. Efficiency:

Grid search checks every possible combination, ensuring it finds the best option within the grid, but this requires a lot of computing power, especially with many settings to test. The genetic algorithm, on the other hand, uses a smarter approach to explore the options more efficiently. It focuses on promising areas and can get similar or better results with fewer tries.

6.2.2 Computational Cost:

Grid search’s thoroughness makes it expensive in terms of computing. If you have several settings and each has many options, the number of tests grows very quickly, making it impractical for complex models. The genetic algorithm starts with a random selection and improves it over time through selection, crossover, and mutation, requiring fewer tests and making it more practical for big problems.

6.2.3 Scalability:

Grid search struggles as the number of settings increases because the number of tests grows rapidly. The genetic algorithm handles larger and more complex settings better because it searches more intelligently and can be run on multiple computers at once to speed things up.

6.2.4 Exploration vs. Exploitation:

Grid search doesn’t adapt based on past results; it treats each test separately, which can waste resources. The genetic algorithm balances exploring new options and focusing on the best ones found so far. It combines successful traits and introduces new ones, allowing it to focus on promising areas, increasing the chances of finding the best solution.

6.2.5 Empirical Performance:

In this study, the genetic algorithm performed better than grid search, both in accuracy and efficiency. It found better solutions with fewer tests, showing its ability to navigate complex settings and find high-performing options.

6.2.6 Flexibility:

The genetic algorithm is more flexible than grid search. Grid search is limited to a predefined set of options, while the genetic algorithm can explore a wider range, potentially finding better solutions. This flexibility is especially useful for models with settings that have a broad range of possible values.

6.3 Limitations

Despite the positive results, there are a few limitations to keep in mind.

6.3.1 Computational Resources:

Running genetic algorithms can still be demanding, especially with large populations and many iterations. While it’s more efficient than grid search, it can still be costly in terms of computing, particularly for complex models and large datasets. Access to powerful computers can help, but not everyone may have this option.

6.3.2 Parameter Sensitivity:

The genetic algorithm's performance depends on its own settings, like population size, number of iterations, and the likelihood of crossover and mutation. Finding the right values for these requires experimentation and can affect how well the algorithm works. Techniques to fine-tune these settings could help improve performance.

6.3.3 Model-Specific Adaptations:

The genetic algorithm can be adapted to different models, but it may need specific tweaks for each type. For example, settings for deep learning models might need different approaches compared to simpler models like SVR or RandomForest. Customizing the algorithm for specific models is important for getting the best results.

6.3.4 Scalability:

Although the genetic algorithm worked well for the current dataset and models, its ability to handle very large datasets and extremely complex settings needs further testing. Future research should look into ways to improve its scalability, such as using distributed computing or combining it with other optimization methods.

6.3.5 Interpretability:

Like other advanced optimization methods, genetic algorithms can be hard to understand. It can be challenging to explain why certain settings were chosen and how they affect model performance. Adding techniques to improve understanding, like analyzing feature importance, can provide more insight into the algorithm's choices and results.

7. Conclusion

7.1 Summary

This study explored the application of genetic algorithms (GAs) for optimizing hyperparameters in machine learning models. The primary objective was to demonstrate the effectiveness of GAs in enhancing model performance through systematic optimization. The study involved implementing a GA to optimize hyperparameters for four different models: Support Vector Regression (SVR), RandomForestRegressor, KNeighborsRegressor, and DecisionTreeRegressor.

The results showed that GAs significantly improved model performance by reducing the mean squared error (MSE) on the test dataset. The iterative process of selection, crossover, and mutation enabled the GA to explore a wide solution space and identify optimal hyperparameter configurations. The adaptability of the GA to different models highlighted its versatility as an optimization tool.

Comparisons with grid search demonstrated the advantages of GAs in terms of computational efficiency and effectiveness. While grid search provides an exhaustive search of the hyperparameter space, it is computationally expensive and impractical for high-dimensional spaces. The GA, with its heuristic approach, efficiently navigates the solution space, achieving comparable or better results with fewer evaluations.

Despite the promising results, the study acknowledged several limitations, including computational resource requirements, sensitivity to GA parameters, and scalability challenges. Addressing these limitations will be crucial for future work to further enhance the GA's applicability and performance.

7.2 Future Work

Future research should focus on addressing the limitations identified in this study and exploring new avenues to enhance the genetic algorithm's effectiveness and efficiency.

7.2.1 Computational Efficiency:

Improving the computational efficiency of GAs is essential for handling larger datasets and more complex models. Future work could explore parallel and distributed computing techniques to speed up the optimization process. Additionally, integrating hybrid optimization methods that combine GAs with other algorithms, such as particle swarm optimization or simulated annealing, could further enhance efficiency.

7.2.2 Automated Parameter Tuning:

The performance of GAs is influenced by their own set of hyperparameters, such as population size, number of generations, crossover probability, and mutation probability. Future research should focus on developing automated parameter tuning techniques that can dynamically adjust these settings based on the problem characteristics and optimization progress.

7.2.3 Model-Specific Adaptations:

While the GA framework demonstrated adaptability to various machine learning models, tailoring the encoding and evaluation methods for specific models can yield better results.

Future studies should investigate model-specific adaptations of GAs, particularly for deep learning models and other complex architectures.

7.2.4 Scalability:

Scaling GAs to handle extremely large datasets and high-dimensional hyperparameter spaces is a critical area for future research. Exploring techniques such as incremental learning, where the GA optimizes subsets of data sequentially, or leveraging cloud-based resources for distributed processing, can enhance scalability.

7.2.5 Interpretability:

Enhancing the interpretability of GA-optimized models is essential for practical applications, especially in fields like healthcare and finance where understanding model decisions is crucial. Future work could focus on integrating interpretability techniques, such as feature importance analysis and sensitivity analysis, to provide deeper insights into the GA's optimization process and its outcomes.

7.2.6 Multi-Objective Optimization:

Extending the GA framework to support multi-objective optimization can provide a more holistic approach to model selection. Future research could explore how GAs can simultaneously optimize multiple performance metrics, such as accuracy, computational cost, and interpretability, to achieve a balanced and comprehensive model selection strategy.

7.2.7 Real-World Applications:

Finally, applying GAs to real-world problems across various domains, such as healthcare, finance, and engineering, can validate their practical utility and effectiveness. Case studies and empirical evaluations in different contexts will provide valuable insights into the strengths and limitations of GAs in real-world scenarios.

8. References

[1] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. University of Michigan Press.

[2] Mitchell, M. (1998). An Introduction to Genetic Algorithms. MIT Press.

[3] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

[4] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

[5] Michalewicz, Z. (1996). Genetic Algorithms + Data Structures = Evolution Programs. Springer.

[6] Deb, K. (2001). Multi-Objective Optimization using Evolutionary Algorithms. Wiley.

[7] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

[8] Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research.

[9] Brochu, E., Cora, V. M., & De Freitas, N. (2010). A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. arXiv preprint arXiv:1012.2599.

[10] Khan, M. H., & Yadava, G. S. (2011). Multi-Objective Evolutionary Algorithms for Knowledge Discovery from Databases. Springer.

[11] Brabazon, A., & O'Neill, M. (2006). Biologically Inspired Algorithms for Financial Modelling. Springer.

[12] Nolfi, S., & Floreano, D. (2000). Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. MIT Press.

[13] Calegari, P., & Fogliatto, F. S. (2008). Evolutionary Algorithms for Mobile Ad Hoc Networks. Springer.

[14] Cheng, R., Gen, M., & Tsujimura, Y. (1999). Genetic Algorithms and Engineering Optimization. Wiley.

[15] Rubin, K. S. (2012). Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley Professional.

[16] Larman, C. (2003). Agile and Iterative Development: A Manager's Guide. Addison-Wesley Professional.

[17] Beck, K., & Andres, C. (2004). Extreme Programming Explained: Embrace Change (2nd ed.). Addison-Wesley Professional.

[18] https://www.kaggle.com/datasets/nikhil7280/student-performance-multiple-linear-regression

[19] https://archive.ics.uci.edu/dataset/519/heart+failure+clinical+records

[20] https://www.kaggle.com/datasets/ahsan81/used-handheld-device-data

[21] Sivanandam, S. N., & Deepa, S. N. (2007). Introduction to Genetic Algorithms. Springer.

[22] Back, T. (1996). Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press.

[23] Eiben, A. E., & Smith, J. E. (2015). Introduction to Evolutionary Computing. Springer.

[24] Yao, X. (1999). Evolutionary Computation: Theory and Applications. World Scientific.

[25] Glover, F., & Kochenberger, G. A. (2003). Handbook of Metaheuristics. Springer.

[26] Kalyanmoy, D. (2010). Multi-Objective Optimization Using Evolutionary Algorithms: An Introduction. Springer.

[27] Whitley, D. (2001). An Overview of Evolutionary Algorithms: Practical Issues and Common Pitfalls. Springer.

[28] Pelegrini, T. R., & De Souza, J. M. (2008). Evolutionary Algorithms for Solving Multi-Objective Problems. Springer.