



Article

Predicting CPU Multi-Core Benchmark Performance from Hardware Specifications: A Reproducible Hardware–Software Workflow Using Gradient Boosting

Mohammed Nasih Ismael

1. Applied Geography Department, College of Arts, University of Kirkuk, Iraq

* Correspondence: mohammedismael@uokirkuk.edu.iq

Abstract: Predicting CPU performance in the absence of physical benchmarking is a practical problem facing hardware engineers, system designers, and researchers with limited resources. This research proposes an open-source and reproducible hardware-software workflow that is capable of predicting Cinebench R23 multi-core scores based on hardware specifications alone, without any benchmark measurements as an input to the prediction models. This workflow is applied to a dataset of 215 state-of-the-art CPUs from various hardware vendors, including Intel, AMD, and Apple, with CPUs classified into Desktop and Laptop categories. Four different machine learning models, namely Linear Regression, Ridge Regression, Random Forest, and Gradient Boosting, are utilized in this workflow. All features, such as core count, thread count, base clock, and turbo clock, are actual hardware specifications available from datasheets prior to any physical benchmarking or testing. A log transformation was applied to the target variable to reduce skewness in data, resulting in a reduction from 4.035 to 1.127. It is found that Gradient Boosting performs better, with $R^2 = 0.845$, RMSE = 0.149, and MAE = 0.106 on the test data, validated by 5-fold cross-validation with a high R^2 of 0.853 ± 0.037 . Feature importance analysis of the models shows that thread count is the primary feature contributing to multi-core performance, followed by core count, while clock speed contributes to a much lesser extent, about 6% each, and manufacturer is of negligible importance, less than 1%. This workflow is reproducible, platform-independent, and executable on any standard laptop without any additional hardware, making it accessible to researchers in constrained environments.

Keywords: CPU performance prediction; machine learning; hardware specifications; Gradient Boosting; multi-core benchmark estimation; resource-constrained computing.

Citation: Ismael, M. N. Predicting CPU Multi-Core Benchmark Performance from Hardware Specifications: A Reproducible Hardware–Software Workflow Using Gradient Boosting. Vital Annex: International Journal of Novel Research in Advanced Sciences 2026, 5(3), 81-104.

Received: 10th Mar 2026Revised: 21st Mar 2026Accepted: 28th Mar 2026Published: 02nd Apr 2026

Copyright: © 2026 by the authors. Submitted for open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>)

1. Introduction

The rapid evolution of modern processor architectures has significantly increased the level of complexity in evaluating performance, particularly in multi-core processors where parallel processing dominates the rate of computation [1]. Estimating CPU performance is critical in a variety of application domains, such as hardware selection, system design, cloud computing, and performance improvement, among others [2]. Both in research and application domains, evaluating processor performance is critical in comparing processors and in decision-making in various application domains with varying workload demands [3].

Many standardized tests like Cinebench R23, SPEC CPU, and Geekbench measure the performance of processors by running controlled workloads to provide numerical values for the performance of each processor's capability. These standardized tests can provide reliable information; however, they have the limitation of requiring both physical hardware and a controlled environment to run the test [4]. Because of this limitation,

testing is not useful at the early stages of systems design, when making decisions regarding hardware procurement, or when comparing newly released processors. Further limitation is that to run the benchmarks across numerous systems takes a considerable amount of time and resources, which causes difficulty in making large comparisons of systems [5].

Mathematical models that utilize machine learning to estimate the speed of a microprocessor (or CPU) using its hardware and software will help solve these issues. Consequently, less traditional benchmarking is going to be performed to measure CPU performance [6]. Nonetheless, each of the existing methods require different low-level attributes such as cache size, memory speed, bandwidth, and microarchitecture, which may not be contained in the product specifications and require specialised programmes or previous testing to find. Additionally, many of the modelling techniques now are extremely complex and utilise bigger amounts of data than can be used in most cases of lack of resources [7, 8].

In conclusion, there is significant demand for an easy-to-use, repeatable method of estimating CPU performance using only well-known specifications; this method must be hardware-independent (i.e., able to be applied to multiple CPU designs). More specifically, developing accurate estimates of multi-core benchmark results based on basic datasheet characteristics (core count, thread count, etc.) Achieving this may be challenging, but it is ultimately rewarding and worthwhile. Solving this challenge will allow CPU performance to be estimated at early stages without needing to do physical benchmarks and will allow for more effective decision-making in both the research world and in industry contexts.

Research Problem

Even though benchmarking tools are commonly used to evaluate CPU performance, it is still difficult to anticipate ESO Benchmark outcomes before they can be empirically measured. Because many of the methodologies used to produce predictions suffer from one or more limitations, the difficulties encountered predicting benchmark outcomes will be discussed as follows: First, traditional benchmarking methodologies have been developed based on the assumption that test hardware will always be available and that test environments will always exist, making them less useful for early evaluation or for comparative analyses on a large scale. Second, a variety of machine learning-based prediction models typically depend upon architectural features and performance statistics that are not publicly available through the specifications of CPUs or any other form of hardware, leaving many of these models not usable in real-world applications. Finally, many prediction strategies utilize highly advanced models with high computer costs per trial, leaving these strategies unavailable to users who do not have the proper technology, resources, and expertise to conduct these types of computations.

Consequently, there is a gap in the practical techniques that are effective in predicting multi-core CPU performance based solely on simple and easily accessible hardware specifications. Therefore, there is a need to fill the gap and provide a simple and reproducible solution to bridge the gap between theory and practice.

Research Objectives

The objective of this study is to create a readily available and reproducible process for predicting the performance of a CPU's multi-core benchmark using only hardware specifications listed in a manufacturer's data sheet. The objectives of this study are as follows:

- a. Create a machine learning-based system that can predict Cinebench R23 multi-core benchmark scores without the need for physical testing.
- b. Investigate and compare the efficacy of several regression algorithms for predicting CPU performance.

- c. Investigate the importance of hardware features in relation to multi-core performance using feature importance.
- d. Validate the efficacy of the proposed process using cross-validation and external benchmark data.

Research Contributions

The main contributions of this study can be summarized as follows:

- a. A fully reproducible hardware–software workflow for CPU performance prediction that eliminates the need for physical benchmarking.
- b. A lightweight and platform-independent approach that can be executed on standard consumer hardware without specialized computational resources.
- c. A systematic comparison of multiple machine learning models for predicting multi-core performance using a minimal set of datasheet-based features.
- d. Empirical evidence demonstrates that accurate CPU performance prediction can be achieved using only publicly available hardware specifications.

Literature Review

Recent advancements in CPU performance prediction cover a wide range of analytical, statistical, and machine learning techniques that reflect the increased need for the accurate and effective prediction of CPU performance and capabilities [9]. Initially, CPU performance prediction was mainly based on analytical and simulation models that predicted CPU performance based on the hardware characteristics and the ability of the CPU to process the programs and applications running on the system [10]. Although the predictions were quite accurate, the need for low-level hardware specifications and complex modeling tools was a major limitation for the practical application of these models. Later, statistical models and regression analysis were introduced that predicted CPU performance based on statistical models and the features derived from the benchmarks and applications running on the system [11]. Recently, with the advent of machine learning models and techniques, CPU performance prediction has been based on neural networks and deep learning models that predict CPU performance based on the complex mapping and non-linear relationships between hardware characteristics and CPU performance [12]. Although the predictions are quite accurate and effective for CPU performance prediction, the need for complex hardware characteristics and features such as cache memory, memory bandwidth, instruction-level characteristics, and runtime profiling makes the practical application of these models quite difficult and complex. Moreover, some models and techniques focus on cloud computing and high-performance computing (HPC) systems that predict CPU performance based on cloud and hardware characteristics and features. A summary of the recent and effective CPU performance prediction models and techniques is presented in Table 1.

Table 1. Summary of CPU performance prediction approaches in the literature.

Study	Year	Dataset / Benchmark	Features Used	Model(s)	Performance Metric	Key Findings
Chennupati et al. [13]	2017	Scientific codes, PPT toolkit	Memory hierarchy, cache, latency, throughput	Analytical Memory Model (AMM) + regression	Not specified	Accurate runtime prediction using memory behavior and cache modeling
Mariani et al. [14]	2017	NAS benchmarks	~200 application + system features	Polynomial ML model	<15% error	Predicts execution time and cost in cloud environments

Rosales et al. [15]	2017	HPC applications (NAMD, GROMACS)	Memory bandwidth, CPU throughput	Analytical model	Not specified	Simple model predicts performance across architectures
Limaye & Adegbiya [16]	2018	SPEC CPU2017	Instruction mix, cache, branch behavior	Analysis (no ML model)	Not specified	Provides detailed workload characterization
Wurl et al. [17]	2019	Hardware asset datasets	System-level hardware data	Regression (PLS, robust regression)	Not specified	Predicts hardware requirements using statistical models
Oi [18]	2019	SPEC CPU2017	Benchmark scores only	No ML model	SPEC intrate, fpspeed	Shows variability in CPU performance across workloads
Foots et al. [19]	2020	CPU dataset (209 samples)	Memory, cache, channels, cycle time	Neural networks, regression	MSE, correlation	Neural networks capture non-linear performance relationships
Foots et al. (conf.) [20]	2020	CPU dataset (209 samples)	Same as above	DNN + regression	MSE, correlation	DNN improves prediction accuracy with non-linearity
Mankodi et al. [21]	2020	Simulated hardware (Gem5)	Computation & memory patterns	Multivariate ML model	>95% accuracy	Predicts performance & power across architectures
Saad & El-Mahdy [22]	2020	SPEC-MPI, NASA benchmarks	Cloud resource features	Analytical queueing model	MAPE, CV (~88%)	Predicts execution time of HPC cloud applications
Al Sefat et al. [23]	2021	CPU dataset	General CPU features	Regression models	Not specified	Compares regression algorithms for CPU prediction
Topçu & Öz [24]	2022	GPU benchmarks	Cache, memory, cores, bandwidth	ML (GB, SGD)	Accuracy up to 96%	Predicts GPU fault vulnerability
Wei-wei et al. [25]	2023	UCI CPU dataset	General CPU features	HBRB optimization +	Not specified	Interpretable model improves reliability
Sharif et al. [26]	2023	HPC applications	Computation & communication	Analytical model	~90% accuracy	Predicts performance in HPC systems
Cengiz et al. [27]	2023	SPEC CPU datasets	34 hardware features	CNN, deep learning	R ² up to 0.98	Deep models achieve very high accuracy
Wang et al. [28]	2024	SPEC CPU data	Core count, frequency	Statistical analysis	Not specified	Hardware features strongly influence benchmarks
Wang (AI failure) [29]	2024	Sensor + system logs	Real-time hardware data	CNN-LSTM, ML models	Not specified	Predicts hardware

Marquez & Mondragon [30]	2024	Scientific benchmarks	Storage + system features	EVT model	<7% error	failures using AI Models cloud storage impact on performance
Liu (MambaCPU) [31]	2025	PerfCastDB, SPEC	35 hardware features	State-space model	deep MSE, MAE, MAPE	Captures complex feature dependencies
Arslan & Demirbaga [32]	2025	Cloud monitoring data	Low-level infrastructure metrics	XGBoost, SVM, RF	>92% accuracy	XAI improves interpretability
BuQing et al. (CAPSim) [33]	2025	Instruction traces	Instruction-level features	Attention-based NN	Not specified	Faster and accurate CPU simulation

Although notable advancements have been achieved in the prediction of CPU performance, the literature indicates that there exist several limitations that need to be addressed in the prediction of CPU performance, as shown in Table 1. Firstly, most of the literature requires detailed architectural features such as cache configuration, memory bandwidth, instruction-level characteristics, and runtime profiling information that may not be readily available from the manufacturer's datasheet. This indicates that literature may not be readily applicable in real-world scenarios due to the unavailability of such information. Secondly, most of the literature requires the use of benchmarking and simulation approaches to obtain the required data, which may not be readily available from the physical hardware and may contradict the need for an early-stage prediction approach. Thirdly, the recent literature that utilizes machine learning and deep learning approaches to predict CPU performance may not be readily reproducible by all researchers due to the complexity and computational intensity of the approaches that may not be readily reproducible by all researchers with limited computational power and facilities. Moreover, most of the literature may be domain-specific and may not be readily applicable in the prediction of CPU performance in a standard environment.

The current research aims to introduce a lightweight and reproducible framework to predict CPU multi-core performance based on publicly available hardware specifications, in order to overcome the aforementioned limitations. Unlike other existing prediction models that heavily rely on various architectural features or benchmark-based inputs, this proposed model utilizes a limited set of features available in the datasheet, thus facilitating its applicability in early-stage prediction scenarios. A variety of regression models based on machine learning algorithms, including linear and hybrid models, are utilized to learn the mapping between hardware specifications and benchmark performance. Furthermore, this proposed framework is light-weight and computationally efficient, thus facilitating its execution on regular consumer devices. To prove the efficacy of this proposed prediction framework, experiments are performed using publicly available Cinebench R23 benchmark data, and cross-validation is performed to prove its robustness and generality in predicting CPU performance without any need to physically benchmark or employ feature extraction techniques.

2. Methodology

Workflow Architecture

The proposed hardware and software workflow is designed to be a reproducible and laptop-executable pipeline to predict CPU multi-core benchmark performance based on hardware specifications provided by the manufacturer, without any need for physical benchmarking. The overall architecture is described in Figure 1.

The process can be considered a five-stage workflow. First, a set of 215 contemporary CPUs is obtained from a publicly accessible Kaggle platform [34], where Cinebench R23 benchmark results are stored, including desktop and mobile CPUs from Intel, AMD, and Apple manufacturers. The features used are limited to the manufacturers' specifications, i.e., the number of cores, the number of threads, base and turbo clock speeds, processor brand, and CPU category, to keep them independent of experimental observations.

Secondly, structured preprocessing of the dataset occurs. Thirdly, the preprocessed data are fed to four machine learning models implemented as scikit-learn pipelines, which are trained in parallel. A dual validation technique involving cross-validation and external validation of benchmark data is used. Lastly, the Cinebench R23 multi-core score predictions are obtained. This whole process can be articulated as a five-phase pseudocode algorithm, as shown in Algorithm 1 in Appendix B.

The whole pipeline runs with Python version 3.12.4 and a Jupyter Notebook environment on a standard laptop with the following specifications: Intel Core i7 CPU with 6 cores and 12 threads, 31.8 GB RAM, and Windows 11 OS. The whole pipeline does not need any special hardware or GPU support. The main libraries used by the codebase are scikit-learn, pandas, NumPy, matplotlib, and SciPy, which indicates a straightforward implementation of the proposed method.

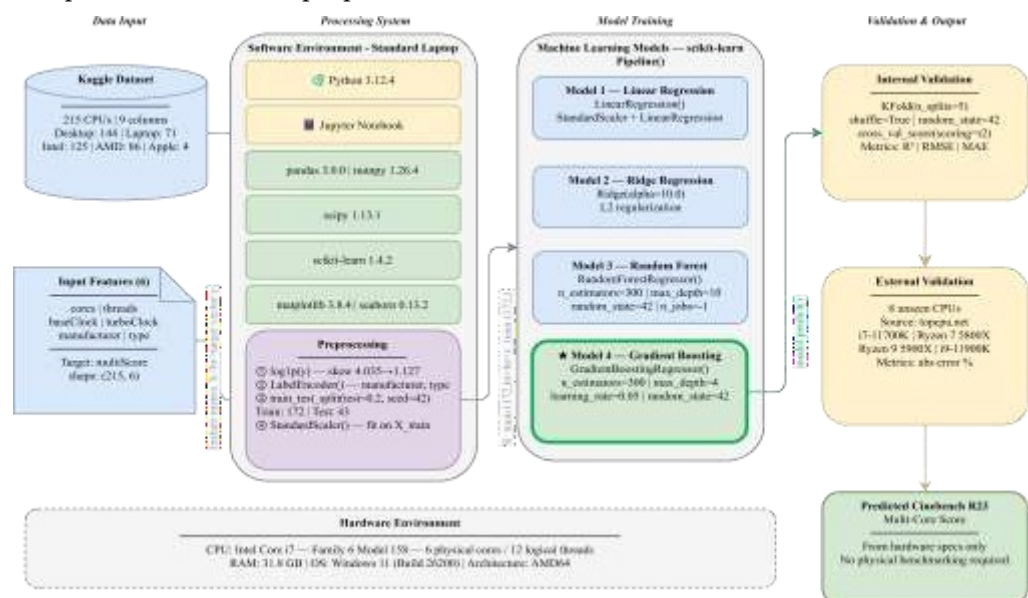


Figure 1. Architecture of the proposed workflow showing data input, preprocessing, parallel model training, validation, and prediction stages.

Dataset

The dataset used for this research was retrieved from Cinebench R23 CPU benchmark, which is publicly accessible at Kaggle (CPU_r23_v2.csv) [34]. Cinebench R23 is based on the software architecture of Cinema 4D. The Cinebench R23 CPU benchmark tests CPU performance using real-world applications, rendering complex 3D scenes to produce single-core and multi-core benchmarks for a wide range of CPU architectures. The dataset contains data about 215 different CPUs, represented by 9 different features, and does not contain any missing values. A sample of this dataset is provided in Table A1 of Appendix A.

The dependent variable, multiScore, represents the overall computational throughput for parallel workloads in a multi-core processor environment. The range for multiScore is between 5,423 and 75,671, while the average and median values are 12,980 and 10,890, respectively. The right skewness (skewness = 4.035) is mainly contributed by high-end processor brands like AMD Threadripper and EPYC. To address this skewness issue,

logarithmic scaling was implemented to decrease the skewness to 1.127, as shown in Figure 2.

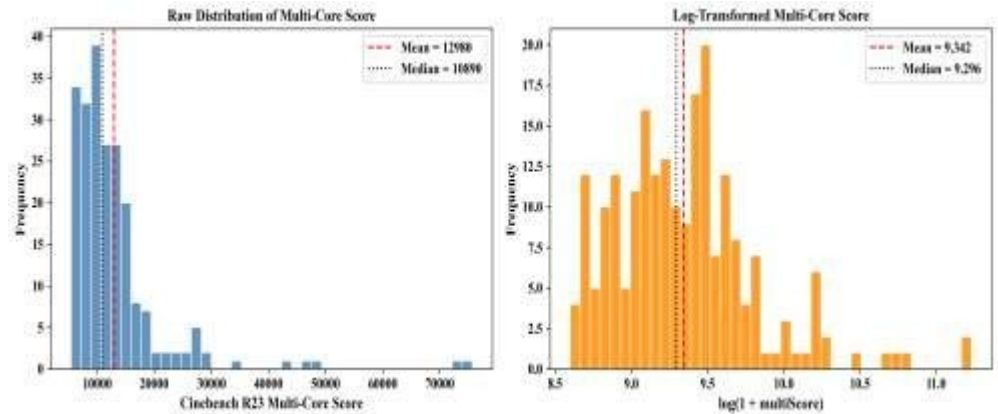


Figure 2. Distribution of Cinebench R23 multi-core scores before and after log transformation.

The dataset contains processors from different brands, namely Intel (58.1%), AMD (40.0%), and Apple (1.9%), categorized as Desktop (67.0%) and Laptop (33.0%). As shown in Figure 3, the Desktop processors have higher multi-core scores compared to Laptop processors. The difference in performance between Intel and AMD processors is not very large.

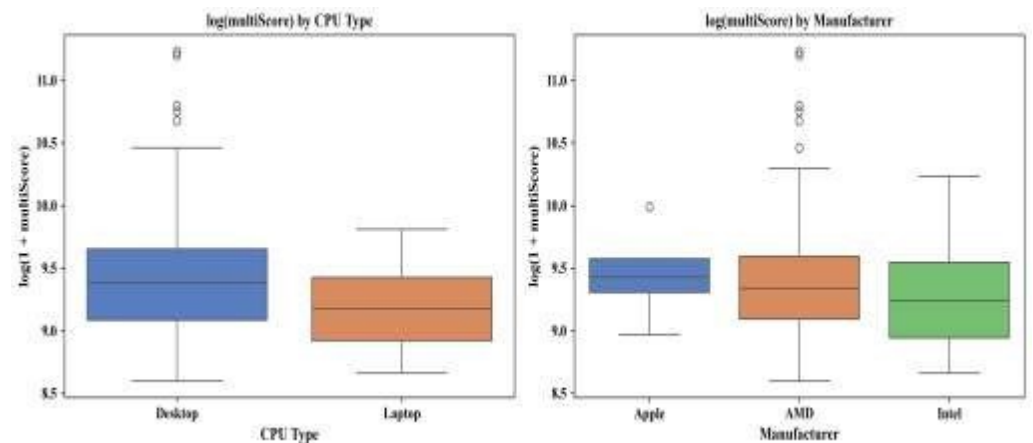


Figure 3. Multi-core score distribution by CPU type and manufacturer, highlighting higher performance for Desktop CPUs and similar trends between Intel and AMD..

The predictive model has six features as input variables. They are core count (cores), thread count (threads), base clock frequency (baseClock, GHz), turbo clock frequency (turboClock, GHz), processor manufacturer (manufacturer), and CPU type (type). All the features are available in publicly accessible hardware specifications. Hence, physical benchmarking is not considered in this model. Table 2 shows the statistical summary of numerical features.

Table 2. Descriptive statistics of numerical features in the dataset (n = 215).

Feature	Count	Mean	Std	Min	Q1	Median	Q3	Max
Cores	215	9.37	7.82	4	6	8	9	64
Threads	215	17.74	15.83	6	12	16	16	128
Base Clock (GHz)	215	3.05	0.65	1.1	2.5	3.2	3.6	4.2
Turbo Clock (GHz)	215	4.51	0.44	3.2	4.2	4.5	4.8	5.5

Multi-Core Score	215	12980	8905.7	5423	8187	10890	14394	75671
------------------	-----	-------	--------	------	------	-------	-------	-------

The range of hardware specifications varies from low-end processor configurations, which have four cores and a minimum frequency of 1.1 GHz, to high-end workstation processor configurations, which have up to 64 cores and a turbo frequency of up to 5.5 GHz. The significant range of multi-core processor performance (std dev = 8,905.74) reflects the difficulty of accurately predicting CPU performance based on hardware specifications alone.

Data Preprocessing

Prior to training the model, a well-structured four-step preprocessing procedure was adopted to ensure the quality and appropriateness of the data for machine learning, as shown in Figure 4. All the procedures were performed using the scikit-learn package (v1.4.2) and pandas' package (v3.0.0).

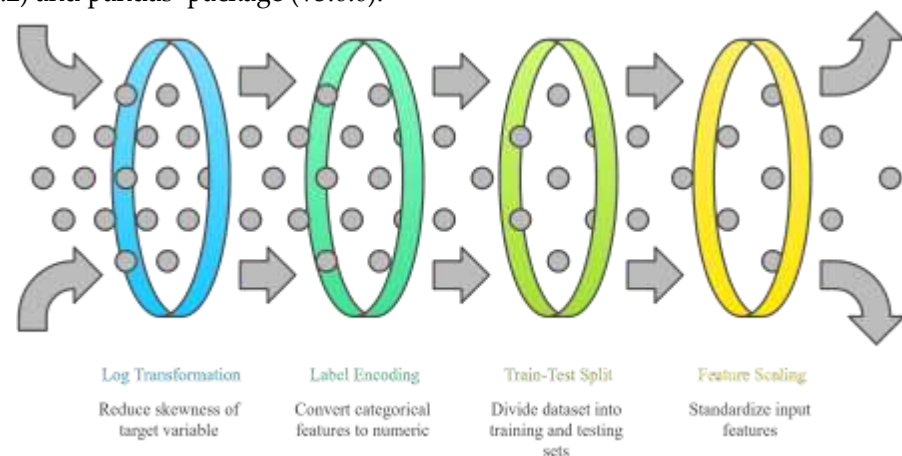


Figure 4. Four-step preprocessing pipeline applied to the Cinebench R23 dataset prior to model training.

Numerous steps were taken while preprocessing the dataset to ensure both consistency and reproducibility. The target variable (multiScore) has a strong right skew of 4.035 due to the high-end workstation processors so we used a logarithmic transformation to mitigate the skewness by transforming $y' = \log(1+y)$ which resulted in a skewness of 1.127 and brought it closer to normal distribution (shown in Figure 2). The model predictions made on the transformed target variable then had to be inverse-transformed $y = \exp(y') - 1$ in order to return them to their original scale. Next, both categorical variables (manufacturer and type) were encoded using the LabelEncoder which fit prior to splitting the dataset in order maintain consistency between the training and testing sets. After encoding the categorical variables, we then stratified the dataset and randomly split into two subsets where 80% was used for training (172 samples) and 20% was used for testing (43 samples), with the testing set being reserved for final evaluation. Feature scaling was applied to normalize all of our input features (all 15 of them) to a mean of 0 and standard deviation of 1 using the StandardScaler which was fit solely on the training data and applied the same standardization process to both training and test data to ensure that there is no leakage of data between both datasets, however tree-based models do not require feature scaling; scalability was added to provide an even footprint across all models. All preprocessing steps were performed using scikit-learn pipeline structures to ensure their reproducibility and eliminate data leakage through cross validation or model testing.

Feature Selection

The selection of features in the current investigation is based on two major factors. Firstly, the features should be based on actual hardware specifications that can be

obtained from the datasheet without the need to physically benchmark the hardware. Secondly, the features should be significantly correlated to the target variable, ensuring that the workflow is computationally feasible even when benchmark measurements are not obtained. The set of features was narrowed to eight features from the original set of attributes from the dataset: cores, threads, base clock, turbo clock, processor manufacturer, processor type, and two derived features obtained during preliminary analysis: clock range (clockRange = turboClock - baseClock) and threads per core (threadsPerCore = threads / cores). The correlation analysis was carried out using the Pearson correlation method to determine the linear correlation between all the features and the log-transformed target variable, $\log(1 + \text{multiScore})$. The correlation matrix obtained from the analysis is shown in Figure 5, and it is observed that the features cores and threads are highly linearly correlated to the target variable ($r = 0.780$ and $r = 0.770$, $p < 0.001$), emphasizing the importance of parallelism in multi-core benchmark performance. The base and turbo clocks are moderately linearly correlated to the target variable ($r = 0.195$ and $r = 0.206$, $p < 0.01$), indicating that clock speed is the secondary factor in determining the performance.

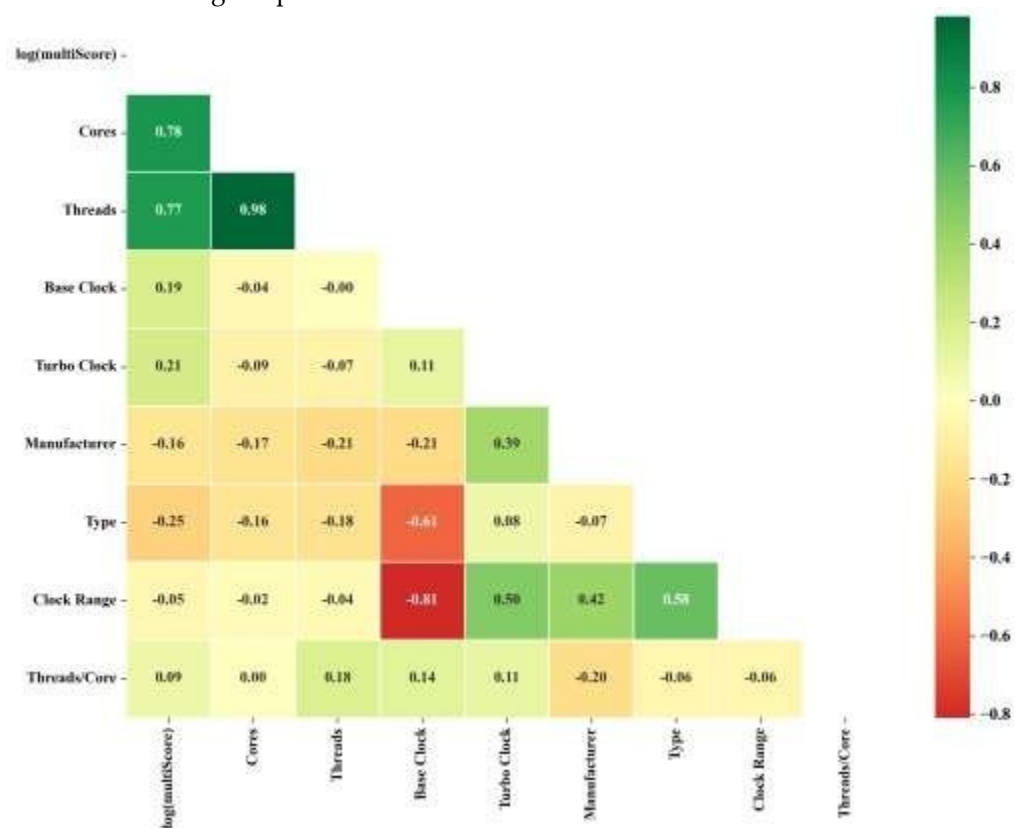


Figure 5. Pearson correlation matrix of all candidate hardware features and the log-transformed multi-core score, including correlation coefficients and significance levels.

Two new features, clockRange and threadsPerCore, showed little connection to the target variable ($r = -0.049$ and $r = 0.090$), meaning that there could be very limited predictive capability. Therefore, these two features were removed from the final feature set in order to simplify the model and limit introducing noise into the learning process. Since both manufacturer and type features have categories, they will be kept, because they document performance differences among Desktop and Laptop processors at the platform level and differences in the manufacturer, as shown in Figure 3.

Model training was completed using the following six feature types: cores, threads, baseClock, turboClock, manufacturer, and type. All features satisfy both selection criteria for model training because all of the features are available on manufacturer data sheets and all of the features show either a statistically significant correlation or a meaningful

distinction by category with respect to the target variable. Table 3 demonstrates the final features, including the Pearson correlation coefficients, p-values and data types for each final feature.

Table 3. Feature selection summary with Pearson correlation coefficients between each candidate feature and $\log(1 + \text{multiScore})$, including statistical significance and selection decision.

Feature	Pearson r	p-value	Selected
Cores	0.78	2.82×10^{-45}	Yes
Threads	0.77	2.11×10^{-43}	Yes
Base Clock (GHz)	0.195	4.12×10^{-3}	Yes
Turbo Clock (GHz)	0.206	2.39×10^{-3}	Yes
Manufacturer	-0.158	2.08×10^{-2}	Yes
Type	-0.249	2.21×10^{-4}	Yes
Clock Range	-0.049	4.79×10^{-1}	No
Threads Per Core	0.09	1.87×10^{-1}	No

The negative correlation values for the manufacturer and type features arise due to their ordinal encoding by LabelEncoder and not due to their inverse physical relationship. These two features were selected based on their statistically significant p-values ($p < 0.05$) and their practical relevance to platform-level performance.

Predictive Models

Four supervised machine learning regression models were implemented and evaluated in this study, ranging from simple linear models to advanced ensemble methods. All models were implemented using scikit-learn 1.4.2 and encapsulated within Pipeline objects combining feature scaling and model training into a single reproducible unit. The selection of these four models was deliberate, linear models serve as interpretable baselines, while ensemble methods capture non-linear relationships between hardware specifications and benchmark performance. A summary of all model configurations is presented in Table 4.

Table 4. Summary of machine learning model configurations and hyperparameters used in this study.

Model	Implementation	Key Hyperparameters
Linear Regression	LinearRegression()	Default
Ridge Regression	Ridge()	alpha = 10.0
Random Forest	RandomForestRegressor()	n_estimators = 300, max_depth = 10, random_state = 42, n_jobs = -1
Gradient Boosting	GradientBoostingRegressor()	n_estimators = 300, max_depth = 4, learning_rate = 0.05, random_state = 42

LINEAR REGRESSION

Linear Regression is the simplest and most interpretable model in this study, serving as the baseline against which all other models are evaluated. It models the relationship between the input feature vector x and the log-transformed target variable y' as a linear combination of weighted features, as expressed in Equation (1) [35]:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p = \beta_0 + \sum_{j=1}^p \beta_j x_j \quad (1)$$

where β_0 is the intercept term, β_j represents the learned coefficients for each of the input features, which are $p = 6$ in this case, and $\hat{y}$ is the predicted log transform of the multi-core scores. The coefficients of the model are learned by minimizing the Residual Sum of Squares (RSS), as defined in Equation (2) [35]:

$$RSS = \sum_{i=1}^n (y'_i - \hat{y}'_i)^2 \quad (2)$$

Here, n is the number of training samples, y'_i is the actual log score of the i -th sample, and $\hat{y}'_i$ is the predicted one. The model was implemented using the `LinearRegression()` function of the scikit-learn library. The default hyperparameters were used.

3.5.2 RIDGE REGRESSION

Ridge Regression builds on Linear Regression by adding an L2 regularization term to the objective function. This reduces the magnitude of the regression coefficients and hence reduces overfitting due to multicollinearity between the features. Due to the high correlation between cores and threads ($r = 0.98$, Figure 5), multicollinearity is a potential issue. However, Ridge Regression is able to handle this due to its ability to handle multicollinearity. The Ridge Regression cost function is given by Equation (3) [36]:

$$\text{minimize } \sum_{i=1}^n (y'_i - \hat{y}'_i)^2 + \alpha \sum_{j=1}^p \beta_j^2 \quad (3)$$

The trade-off between data fidelity and the reduction of magnitude for the coefficients is controlled through the regularization parameter α . In this research, the α value was determined to be 10.0 after some experimentation. The implementation of the model used the Ridge regression with α set to 10.0 using the scikit-learn library.

3.5.3 RANDOM FOREST

Random Forest is a bagging ensemble method where a set of B independent decision trees are constructed by using the bootstrap samples of the training data and their predictions are averaged to compute the final output, as shown in Equation (4) [37]:

$$\hat{y} = \frac{1}{B} \sum_{b=1}^B T_b(x) \quad (4)$$

Let $T_b(x)$ be the prediction made by the b -th decision tree for a given input vector x , with B representing the total number of trees. In a node split, a random subset of m features is considered, as expressed in Equation (5) [37]:

$$m = \lfloor \sqrt{p} \rfloor \quad (5)$$

The total number of input features is represented by p . This random selection of input features for the trees promotes diversity among the trees and also reduces variance. The configuration for the model included the number of trees set to $B = 300$, a depth of 10 for the trees, and `random_state = 42` for reproducibility. The computation for the algorithm was done in parallel using `n_jobs = -1` for the laptop's CPU cores. The algorithm used was `RandomForestRegressor()` from scikit-learn.

Gradient Boosting

The sequential ensemble method of Gradient Boosting is based on constructing decision trees in an additive fashion. Each successive tree is built to correct the residual errors of the previous ensemble. The prediction of the model for the m iteration is given by Equation (6) [38]:

$$F_m(x) = F_{m-1}(x) + \eta \cdot T_m(x) \quad (6)$$

where $F_{m-1}(x)$ represents the ensemble prediction from the previous iteration, η represents the learning rate, which controls the contribution of each newly fitted tree, and $T_m(x)$ represents the newly fitted tree on the negative gradient of the loss function. In the

case of regression with mean squared error loss, the negative gradient takes the form of residuals as given in Equation (7) [38]:

$$r_{im} = y'_i - F_{m-1}(x_i) \quad (7)$$

where r_{im} is the residual for the i -th sample at iteration m , y'_i is the actual log-transformed score, and $F_{m-1}(x_i)$ is the current ensemble prediction. This sequential correction mechanism allows Gradient Boosting to achieve higher accuracy than Random Forest by focusing iteratively on the hardest-to-predict samples. The parameters for the model were set with a value for $B = 300$ estimators, a maximum depth of 4, and a conservative learning rate $\eta = 0.05$ to prevent overfitting. The random state was set to 42. The implementation was done using the scikit-learn library's `GradientBoostingRegressor()` function. This was also determined to be the best model in the study.

Evaluation Strategy

The evaluation framework, as applied in this research, is intended to provide a thorough and rigorous evaluation of model performance, including both aspects of model internal consistency, as well as generalization. The evaluation framework is composed of three different parts, which are evaluation on a hold-out set, cross-validation, and external validation using CPU benchmarks.

Performance Metrics

To evaluate the performance of the models in all the experiments, three quantitative metrics were used. The first quantitative performance metric used in this research was the coefficient of determination, denoted as R^2 , which represents the proportion of variance for the log-transformed target variable that can be explained by the model, as shown in Equation (8):

$$R^2 = 1 - \frac{\sum_{i=1}^n (y'_i - \hat{y}'_i)^2}{\sum_{i=1}^n (y'_i - \bar{y}')^2} \quad (8)$$

where y'_i represents the actual log-transformed score for the i -th sample, $\hat{y}'_i$ is the predicted score, and $\bar{y}'$ is the mean of the actual values. A perfect predictive accuracy is indicated by an R^2 score of 1.0. A score of 0.0 means the model does no better than a mean predictor. R^2 values less than 0.0 indicate the model does worse than a mean predictor.

The second metric is Root Mean Squared Error (RMSE), which is a measure of the average size of prediction errors in the same units as the log-transformed target variable, as measured by Equation (9):

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y'_i - \hat{y}'_i)^2} \quad (9)$$

RMSE puts a greater penalty on larger errors than smaller errors due to the squaring involved. RMSE is more sensitive to outlier predictions. A smaller RMSE implies a higher predictive accuracy.

The third metric under consideration is the Mean Absolute Error (MAE), which measures the average absolute difference between the predicted and actual values; this gives a more robust evaluation by considering the presence of outlying values as described by Equation (10):

$$MAE = \frac{1}{n} \sum_{i=1}^n |y'_i - \hat{y}'_i| \quad (10)$$

All three metrics were computed on the log-transformed scale. To perform the external validation, another metric called Mean Absolute Percentage Error (MAPE), computed on the original score scale after the inverse transformation, resulted in an intuitive measure for the accuracy of the predictions in percentage form, as shown in Equation (11).

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\% \quad (11)$$

where y_i and $\hat{y}_i$ are the actual and predicted multi-core scores on the original scale after inverse log transformation.

Hold-Out Test Set Evaluation

To evaluate the four models of machine learning algorithms the primary evaluation method used was the use of a holdout data set composed of 43 data samples or 20% of the complete data set where this data set was completely excluded from the modelling process including the training, parameter selection, and pre-processing of the data. After training the 4 models on the training data set of 172 samples, the four models were evaluated on the hold out data set using the values of R^2 , RMSE and MAE.

Five-Fold Cross-Validation

Using five-fold cross-validation estimates model stability as well as investigates the likelihood of overfitting by utilizing scikit-learn's KFold method with the following parameters: $n_splits = 5$, $shuffle = True$, $random_state = 42$ on 172 samples in the training data set. The models were trained on four of five data sets (each having approximately 138 samples) in the training phase of five-fold cross-validation and subsequently tested on one of five data sets (approximately 34 samples). This is done to allow for five unique measurements of each model's performance. The average R^2 values of the five models, as well as their respective standard deviations, will provide insight into the average performance and stability of those models as applied to different data sets. A low standard deviation indicates that the model is stable and not dependent on the training sides; otherwise, if there is a significant difference between the R^2 values obtained in the cross-validation procedure and the R^2 values of the test data set, then the model has been overfitted. All data processing operations were included in the Pipeline, thereby preventing any occurrence of data leakage during five-fold cross-validation.

External Validation

An external validation was carried out using eight independent CPU processors, which are not included in the Cinebench R23 Kaggle dataset [39], in order to evaluate the generalization performance of the proposed approach on real-world scenarios. The verified benchmark scores for these independent processors were collected from a third-party website, 'topcpu.net,' which offers independent verified benchmark scores for different processors. These scores represent the Cinebench R23 multi core benchmark scores for different processors. The complete set of processors used for external validation is given in the Results section.

The Gradient Boosting model, which was found to be the best by evaluating the performance on the hold-out test set, was utilized to forecast the multi-core score for eight different processors using only their hardware specifications as input variables: cores, threads, baseClock, turboClock, manufacturer, and type. The predicted multi-core score was compared against the actual score from topcpu.net for the respective processors, and the MAPE was utilized as the primary performance evaluation metric, as described by Equation (11). This evaluation represents the most rigorous evaluation for the workflow's applicability to real-life scenarios, as it assesses the performance on completely independent information from a different database.

Hardware and Software Environment

All experiments conducted in this research were conducted on a standard consumer laptop computer without additional computing hardware. This serves to demonstrate the accessibility of the proposed workflow for researchers with limited resources. The

hardware and software configuration used for these experiments is provided in Table 5 to ensure reproducibility of all results.

The experimental platform used in this research is a laptop computer with an Intel Core i7 processor based on the sixth generation of microarchitecture (Intel Family 6 Model 158). It has six cores and twelve threads, which are supported by Intel Hyper-Threading Technology. It also has 31.8 GB of DDR4 RAM, which is sufficient for performing the dataset operations. The operating system used is Windows 11 (Build 26200) based on the AMD64 platform. It is mentioned that no graphics processing unit was used throughout the entire process, and all the computations done through scikit-learn are performed on the CPU only. This platform is a mid-range laptop computer that is readily available for use by researchers and practitioners.

The workflow was implemented using exclusively Python version 3.12.4 and was executed within a Jupyter Notebook for interactive reproducibility and provision of step-by-step documentation for all the processes. The core machine learning operations were performed using scikit-learn 1.4.2, which provided all model implementations, preprocessing utilities, pipeline structures, and cross-validation tools used in this study. Data loading, manipulation, and statistical operations were performed using pandas 3.0.0 and numpy 1.26.4. Statistical analysis including Pearson correlation computation and skewness calculation was conducted using scipy 1.13.1. All figures and visualizations were generated using matplotlib 3.8.4 and seaborn 0.13.2. The complete list of software dependencies is presented in Table 5.

Table 5. Hardware and software environment used in all experiments.

Component	Specification
Hardware	
Processor	Intel Core i7 (Family 6, Model 158)
Physical cores	6
Logical threads	12 (Intel Hyper-Threading)
RAM	31.8 GB
Operating system	Windows 11 (Build 26200)
Architecture	AMD64
GPU	Not used
Software	
Python	3.12.4
Development environment	Jupyter Notebook
scikit-learn	1.4.2
pandas	3.0.0
numpy	1.26.4
scipy	1.13.1
matplotlib	3.8.4
seaborn	0.13.2

3. Result and Discussion

Results

Exploratory Analysis Results

Before model development, exploratory analysis was performed to understand the nature of the data and explore correlations between hardware specifications and benchmark performance in multi-core. The distribution of the multi-core benchmark was strongly skewed to the right, with a skewness of 4.035, a mean of 12,979.50, and a median of 10,890. This indicates that the bulk of processors have average or mediocre performance, but a small number of powerful processors have much higher performance.

The range of performance was from 5,423 to 75,671. As can be seen in Figure 2, when a logarithmic transformation was applied, the skewness was reduced to 1.127.

As shown in Figure 3, desktop processors outperform their laptop counterparts. The median log-transformed score for desktop CPUs is 9.383 compared to 9.179 for laptop CPUs. This translates to 11,907 and 9,677 points, respectively. This is consistent with the thermal design power and core count for desktop architecture. In terms of manufacturers, the median log-transformed score peaks for Apple (9.428), followed by AMD (9.341) and Intel (9.244). Note, however, that the sample size for the Apple brand is only four, which limits the reliability of the inter-brand comparison.

Pearson correlation analysis, as depicted in Figure 5, showed that the most significant factors affecting performance were the core count and the thread count, with correlation coefficients of $r = 0.780$ and $r = 0.770$, respectively (both $p < 0.001$). The base and turbo clock frequencies were also seen to have some impact on the performance, although this was not as significant as the aforementioned factors, with correlation coefficients of $r = 0.195$ and $r = 0.206$, respectively (both $p < 0.01$). The clock range and the threads per core ratio were seen to have no significant impact on the performance, as indicated by the correlation coefficients of $r = -0.049$ and $r = 0.090$, respectively, thereby confirming the proposed elimination of these factors. The results, therefore, showed that the performance of multi-core processors was largely governed by the level of parallelism rather than the clock speeds of the individual cores.

Feature Importance Analysis

The feature importance scores obtained from the Gradient Boosting model were analyzed to understand the contribution of each hardware feature to the prediction of multi-core performance. The importance of each feature was calculated as the sum of the improvement in impurity over all nodes, multiplied by the proportion of samples seen by each tree. The features of importance scores are shown in Figure 6.

Thread count was identified as the most influential feature, contributing 54.45% of the total importance, followed by core count at 32.17%. Together, these features account for 86.62% of the model's predictive power, confirming that hardware parallelism is the dominant factor in multi-core benchmark performance. Base and turbo clock frequencies showed similar contributions, with importance scores of 6.01% and 5.98%, indicating a secondary but meaningful role of clock speed.

In contrast, CPU type and manufacturer had minimal contributions of 0.77% and 0.63%, suggesting that brand and form factors have limited predictive value once parallelism-related features are considered. These findings are consistent with the Pearson correlation analysis, where core and thread count also exhibited the strongest relationships with the log-transformed target variable, reinforcing the robustness of this conclusion.

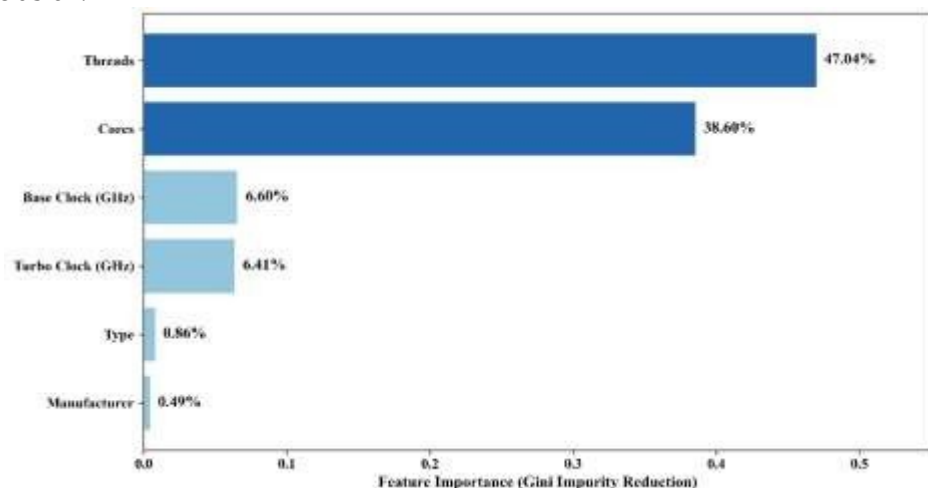


Figure 6. Feature importance scores derived from the Gradient Boosting model.

Model Performance Results

The four machine learning models were trained and tested based on the experimental framework described in the methodology. The metrics used to measure the performance of the models included R^2 , RMSE, and MAE. The results were based on both test set and cross-validation. The detailed results are shown in Table 6.

Table 6. Model performance comparison on the held-out test set and five-fold cross-validation (target: log-transformed Cinebench R23 multi-core score).

Model	CV R^2 (mean)	CV R^2 ($\pm$ std)	Test R^2	Test RMSE	Test MAE
Linear Regression	0.7036	0.0481	0.6141	0.2349	0.1895
Ridge Regression	0.7197	0.0552	0.5353	0.2577	0.1985
Random Forest	0.8823	0.0183	0.8096	0.165	0.12
Gradient Boosting	0.8532	0.0368	0.8454	0.1487	0.1062

Gradient Boosting produced the best results in terms of accuracy, as indicated by a high R^2 of 0.845 on the test set, RMSE of 0.149, and MAE of 0.106. This was further reinforced by the results from the cross-validation process, which indicated a mean R^2 of 0.853 with a standard deviation of 0.037. Random Forest came in second, with a high R^2 of 0.810 on the test set and an RMSE of 0.165, while its performance in the cross-validation process indicated a high R^2 of 0.882 with a standard deviation of 0.018.

Linear Regression and Ridge Regression, on the other hand, were observed to produce results with significantly lower accuracy, as indicated by a test set R^2 of 0.614 and 0.535, respectively, suggesting that even though a linear relationship exists, these models cannot capture other complex patterns in the dataset. A general overview of the performance of the models, as compared to this research, is illustrated in Figure 7.

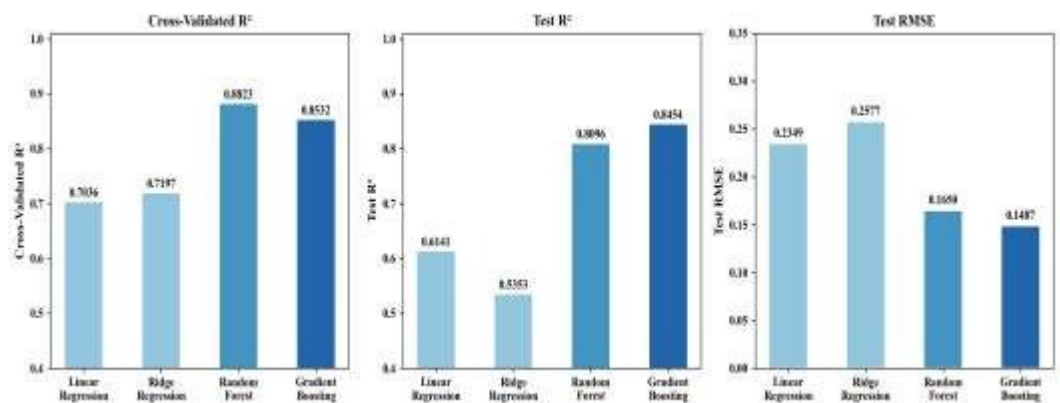


Figure 7. Comparison of cross-validated R^2 , test R^2 , and test RMSE across four machine learning models trained on hardware specifications.

Prediction Quality Analysis

The precision of the optimal model, Gradient Boosting, was further evaluated using actual vs. predicted plots and residual plots, as shown in Figure 8. The actual vs. predicted plot shows a high correlation between the predicted and actual values for multi-core processors, where the majority of the points are positioned close to the reference line. This shows that the model performs well across the entire performance range without apparent deviations to the lower or higher extremes. This shows consistency in the model's performance for both entry-level and high-end processors.

Further, from the residual analysis, the residuals are randomly and symmetrically distributed about zero with no apparent patterns or bias. The absence of

heteroscedasticity also supports the robustness of the model as the variance of the error seems constant across predictions. A small number of outliers are also observed for lower levels of predictions, which could be due to other architectural features not included in the selected features set. The test RMSE of 0.149 and MAE of 0.106 on the log scale imply an average prediction error of 10% on the original scale, which suggests that the predictions made by the Gradient Boosting model are unbiased and accurate for multi-core performance based on hardware specifications alone.

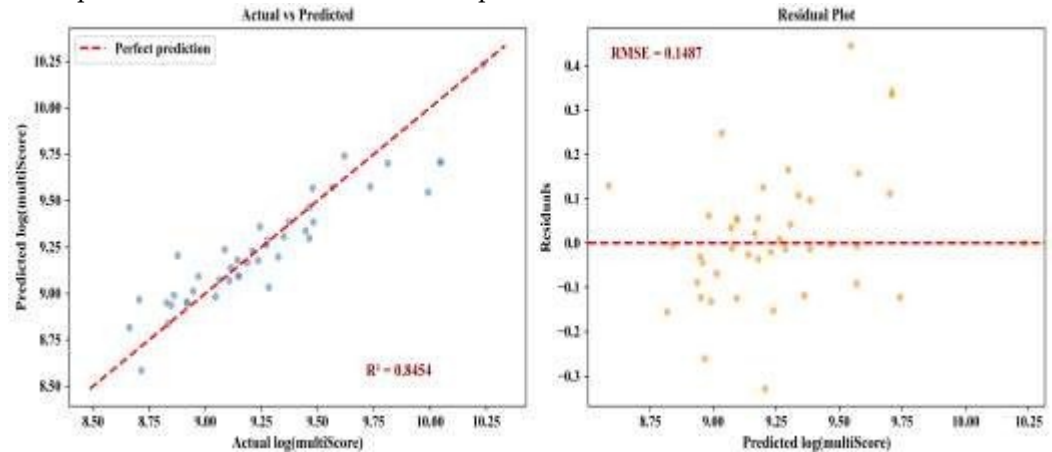


Figure 8. Actual versus predicted log-transformed multi-core scores and residual plot for the Gradient Boosting model on the held-out test set ($n = 43$).

External Validation Results

The purpose of this research project was to test the real-world applicability of the developed Gradient Boosted ML model using multi-core scores from eight processors that were not included in the training set. Manufacturer's data sheets were used to calculate the hardware specifications, and Cinebench R23 multi-core scores were then obtained from an external source for comparative analysis purposes. The results of the comparative analysis are presented in Table 7 results for each processor, and Figure 9 comparison of processed results.

Table 7. External validation results comparing predicted and actual Cinebench R23 multi-core scores for eight unseen processors.

CPU	Cores	Threads	Real Score	Predicted Score	Error (%)
Intel Core i7-11700K	8	16	15011	13883	-0.075
AMD Ryzen 7 5800X	8	16	15228	13997	-0.081
AMD Ryzen 9 5900X	12	24	21255	21565	0.015
Intel Core i9-11900K	8	16	15160	16156	0.066
AMD Ryzen 5 7600X	6	12	15204	10992	-0.277
AMD Ryzen 7 5800X3D	8	16	14259	14334	0.005
Intel Core i7-11700KF	8	16	15011	13883	-0.075
AMD Ryzen 7 3700X	8	16	12458	11899	-0.045
Mean Absolute Error					8.00%

The model resulted in a mean absolute percentage error (MAPE) of 8.0%, with errors ranging between 0.5% and 27.7%. The proposed model was able to accurately predict the performance of seven out of eight processors, as indicated by the error margin of 10%. This demonstrates the generalization performance of the model, particularly for CPUs that have similar characteristics to the training data set. The highest error was observed for the AMD Ryzen 5 7600X processor, which showed the highest deviation of 27.7%. This may be due to the Zen 4 architecture, which has resulted in significant performance

enhancements in terms of instructions per clock (IPC). The absence of this parameter in the input data may have resulted in the model underestimating the performance of the processor, as the architecture of this processor is more efficient than that of the training data. The rest of the processors showed a mean absolute error of 5.5%, thereby indicating the reliable performance of the proposed model in terms of CPU performance prediction without the need to use benchmark measurements.

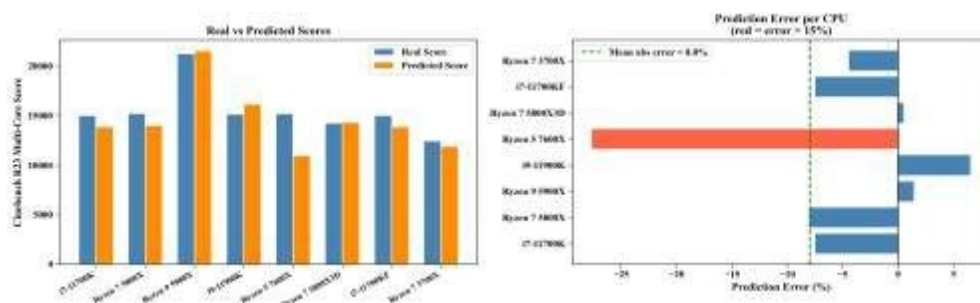


Figure 9. External validation results showing actual versus predicted Cinebench R23 multi-core scores and prediction error percentages for eight unseen processors.

Single-Core vs Multi-Core Performance Analysis

Another analysis was done to assess the relationship between single core and multi-core performance using the information provided in the dataset. The analysis resulted in a Pearson correlation coefficient of $r = 0.166$ ($p < .01$), showing a weak positive relationship between single core and multi core performance. Therefore, single core performance is not a good predictor of multi-core performance. For example, CPU processors with relatively poor single core performance can deliver extremely high multi-core performance if they have a considerable number of cores/threads, such as AMD Threadripper and EPYC processors.

This is in line with the feature importance, where thread count and core count combined contribute 86.62% of the overall prediction power, and the clock frequency-related features contribute a total of 12%. The low correlation between the single-core and multi-core scores also implies that they are different aspects of a processor's capability, where the multi-core is mostly influenced by hardware parallelism. This means that a method that only uses single-core metrics is not sufficient for predicting multi-core, and the emphasis given to core and thread count in the proposed workflow is highly supported.

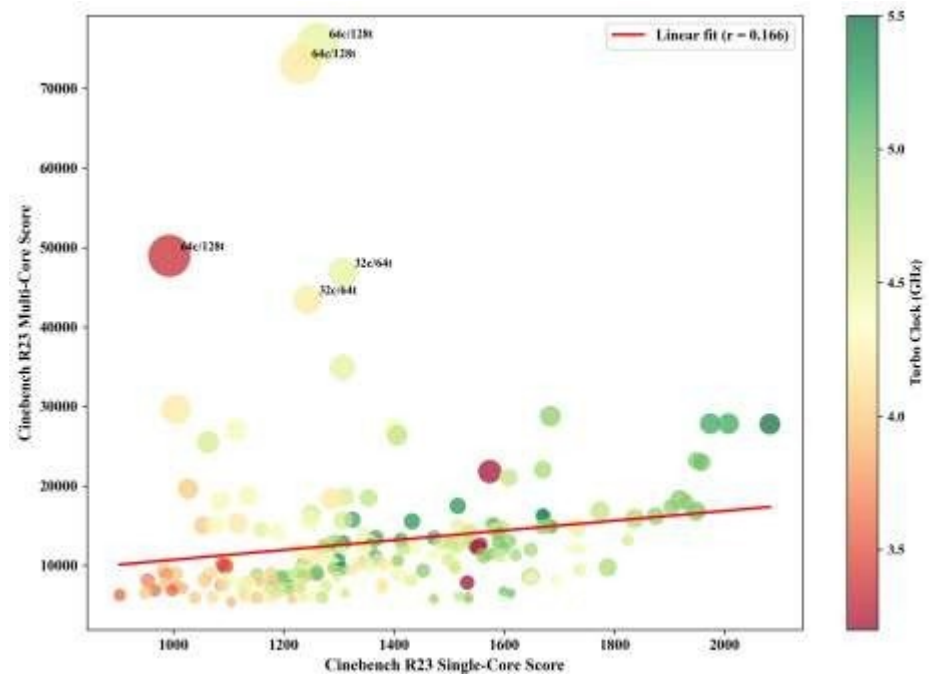


Figure 10. Scatter plot of single-core versus multi-core Cinebench R23 scores, with color representing turbo clock frequency and bubble size representing core count ($r = 0.166$, $p < 0.01$).

Discussion of Findings

Overview of Key Findings

The present study proposes a hardware-based and software-based workflow to predict multi-core benchmark performance, as measured by Cinebench R23 multi-core benchmark, based on CPU hardware specifications alone, eliminating the need to benchmark processors. This was implemented and tested in Python on a standard laptop. Among all tested models, Gradient Boosting was found to be the most efficient in terms of test set $R^2 = 0.845$, RMSE = 0.149, MAE = 0.106 on a logarithmic scale. Stability was checked by cross-validation, which revealed a mean $R^2 = 0.853$ and standard deviation = 0.037. When tested on an external dataset, a mean absolute percentage error of 8.0% was found, within a 10% error margin of all eight processors except one. Feature importance was found to reveal that thread count, and core count were the two most important features in multi-core benchmark performance, accounting for 86.62% of all importance. Other features such as clock frequency were found to have little to no effect. Features such as processor manufacturers and CPU types were found to have little importance. This finding supports the proposition that hardware parallelism is a key determinant of multi-core benchmark performance, which was further supported by correlation analysis. Moreover, a low correlation coefficient between single-core and multi-core benchmark performance was found ($r = 0.166$), which supports the proposition that efficiency per core is an insufficient determinant of multi-core benchmark performance, further supporting the role of hardware parallelism.

Hardware Parallelism as the Dominant Performance Driver

The accuracy of the prediction, where it was noted that 86.62% of the feature importance was accounted for by thread count and core count, is in line with established principles of parallel computing and is further supported by several other related research investigations into CPU performance prediction. Specifically, Tousimojarad and Lujan [40], noted that tree-based models could predict benchmark scores with an average error of less than 4% by using 29 hardware features, with feature importance indicating that specifications related to parallelism were the most important, thus further supporting the

notion that hardware parallelism is a universal driver of CPU benchmark performance, regardless of benchmark or CPU generation.

Wang et al. [41] utilized deep neural networks to forecast performance for new Intel CPU SKUs for SPEC CPU2006 and Geekbench benchmarks, and their method achieved a mean absolute error of 5.5% for SPEC CPU2006 benchmarks and 11.2% for Geekbench benchmarks. Their study confirmed that the relationships between hardware specifications and benchmark performance are not purely linear, which is consistent with the performance gap observed in our study between linear models (Test $R^2 = 0.614$) and ensemble methods (Test $R^2 = 0.845$). The superiority of non-linear models observed in both studies suggests that interaction effects between hardware features, particularly between core count and clock frequency, contribute meaningfully to performance outcomes.

Al Sefat et al. [23] evaluated multiple regression algorithms for CPU performance prediction using hardware specification features including clock speed, core count, and cache size, published in Springer. Their findings confirmed that clock speed, core count, and cache characteristics are among the most influential features in CPU performance prediction, consistent with the feature importance results of the present study. Liu et al. [42] further corroborated this finding in their study of Intel Xeon Scalable Processors, noting that hardware configuration features rather than vendor-specific characteristics were the primary drivers of benchmark performance variation across processor SKUs.

The weak correlation between single-core and multi-core scores observed in this study ($r = 0.166$) is consistent with the widely accepted understanding in computer architecture that multi-core performance scales primarily with the number of available execution units. This finding is further supported by the negligible contribution of manufacturer identity to performance prediction (0.63% feature importance), suggesting that competitive parity between Intel and AMD in recent processor generations is reflected in the data. The secondary contribution of clock frequency, which makes up 12% of the entire importance, is consistent with the assumption that an increase in clock speed provides a positive impact to per core performance yet does not compensate for the benefits of higher core counts in highly parallelized workloads.

In comparison with the relevant works cited above, the workflow proposed in this study has achieved a similar level of predictive performance in terms of predictive accuracy metrics (Test $R^2 = 0.845$; External MAPE = 8.0%), but with only six hardware specification features extracted from the manufacturer's datasheet. This indicates that the proposed workflow may be regarded as a viable alternative for researchers and engineers working under constrained conditions and may be viewed as a complementary approach to more complex architecture designed for specialized hardware configurations.

Model Performance and Generalization

The Gradient Boosting model performed best when compared to all models in the current study, as supported by the test $R^2 = 0.845$, RMSE = 0.149, MAE = 0.106 log scale, and MAPE = 8.0% during external validation. These results are interpreted in relation to previous literature findings in CPU and hardware performance prediction.

Kaya et al. [43] evaluated the performance of five different models for CPU performance prediction by using the regression models of Gradient Boosting and Neural Networks, with R^2 values of 0.958 and 0.976, respectively. The lower R^2 value of 0.845 in the present study is mainly due to the use of a more constrained set of features with only six specifications from the datasheet, whereas Kaya et al. also utilized additional features such as cache and lithography. The superiority of the Gradient Boosting models to the linear models is evident from both studies and mainly due to the difference in the features utilized for the models.

Al-Tarawneh et al.[44] evaluated the performance of Linear Regression, Artificial Neural Networks, and Random Forest models using the SPEC CPU2017 benchmark data set and found that the Random Forest model performed better than the other models on

all the evaluation metrics. This result is partially similar to the present study's result, in which the Random Forest model achieved a competitive test R^2 value of 0.810 and ranked second in the CPU performance prediction task by the Gradient Boosting model. The consistent high performance of tree-based ensemble models in both studies, which used different data and features, indicates that the best approach for CPU performance prediction tasks is the ensemble approach.

The study conducted by Upadhyay et al. [45] aimed to evaluate the performance prediction of multicore CPUs by applying parallel data mining to available data. The results show that hardware factors such as cache bandwidth, memory requirements, and CPU speeds are significant performance predictors for multicore CPUs. This finding affirms the main conclusion drawn in the current article, which asserts that hardware specifications related to parallel performance are the main performance predictors for multi-core CPU benchmarks, and data mining can be employed to create performance estimation tools for CPUs without the need for hardware support.

The five-fold cross-validation results of the present study ($CV R^2 = 0.853$, $SD = 0.037$) show that the results generalize well across different partitions of the data set. The similarity between the cross-validation R^2 (0.853) and the R^2 from the hold-out test (0.845) indicates that the model indeed performs well in terms of generalization without the problem of overfitting, even with the relatively small data set of 215 samples, which is a common concern for the stability of ensemble models.

Limitations

There are several limitations which need to be considered when interpreting the results obtained from the present study. Firstly, the dataset contains 215 CPUs from a single Kaggle repository. While such a dataset may be appropriate for statistical modeling, it may not accurately represent all processors available in the current market. This may be particularly relevant to smaller manufacturers such as Apple, which contributed only four processors to the dataset. Secondly, all predictions were based on a single metric: the Cinebench R23 score. This may suggest a rendering-centric model which may not accurately predict results based on alternative benchmarking schemes such as SPEC CPU or Geekbench. Finally, there are six features used from datasheet specifications. While such a set of features may be comprehensive, there may be several additional factors which significantly impact performance. This may include processor architecture, cache size, memory bandwidth, processor node, etc., which may all significantly impact the maximum achievable prediction accuracy. One such feature which was not included was instructions per clock (IPC), which has been significantly improved in several modern architectures. For example, the AMD Ryzen 5 7600X processor was found to have a significantly increased prediction error. This may be a result of IPC improvements. Finally, since all processors were released prior to 2022, such a model may become obsolete in the future. This may suggest a need to periodically retrain such a model to maintain accuracy.

4. Conclusion

This work proposes a hardware and software workflow for predicting the performance of a CPU in running the Cinebench R23 multi-core test, relying only on CPU specifications obtained from the manufacturer's datasheet. The proposed workflow, coded in Python and running on a typical laptop computer, eliminates the need for actual hardware benchmarking while maintaining a high level of predictive accuracy. The Gradient Boosting approach produced the highest performance results, with $R^2 = 0.845$, $RMSE = 0.149$, and $MAE = 0.106$. The cross-validation results were consistent, with mean $R^2 = 0.853$. The mean absolute error in the external validation set was found to be 8.0%, with most processors predicted within a range of 10%. The most important input feature was the number of cores and threads, which accounted for more than 86% of the model's

importance. The clock speeds were found to be less important, and the categorical variables were least important. The low correlation between the single and multi-core scores ($r = 0.166$) also points to the fact that the performance in the multi-core case is mainly influenced by the number of cores.

For future research it's recommended to:

- a. Expanding the feature set to include cache size, memory bandwidth, process node, and thermal design power.
- b. Incorporating larger and more diverse datasets across multiple benchmark suites to improve generalizability.
- c. Introducing IPC-aware proxy features to better capture microarchitectural efficiency.
- d. Extending the modeling framework to support multi-output prediction for single-core and multi-core performance scores.
- e. Building automated retraining workflows to maintain accuracy in the face of new CPU releases.
- f. Deploying the workflow as a user-friendly application to support practical, real-time estimation of CPU performance.

REFERENCES

- [1] R. H. AlShekh, S. A. Dawwd, and F. N. Qassabbashi, "Comparative Review of Multicore Architectures: Intel, AMD, and ARM in the Modern Computing Era," *Chips*, vol. 4, no. 4, p. 44, 2025. [Online]. Available: <https://doi.org/10.3390/chips4040044>.
- [2] C. Li, J. Zheng, H. Okamura, and T. Dohi, "Performance Evaluation of a Cloud Datacenter Using CPU Utilization Data," *Mathematics*, vol. 11, no. 3, p. 513, 2023. [Online]. Available: <https://www.mdpi.com/2227-7390/11/3/513>.
- [3] M. González-Rodríguez, L. Otero-Cerdeira, E. González-Rufino, and F. J. Rodríguez-Martínez, "Study and evaluation of CPU scheduling algorithms," (in eng), *Heliyon*, vol. 10, no. 9, p. e29959, May 15 2024. [Online]. Available: <https://doi.org/10.1016/j.heliyon.2024.e29959>.
- [4] T. M. Rezende, A. E. M. Medri, J. T. Santos, J. I. Silva Junior, M. A. Alves, and F. d. C. Zottino, "Benchmark review and case study of stress test for a gaming computer applied in CPU," *Anais do Simpósio Brasileiro de Testes de Software Sistemático e Automatizado (SAST)*, pp. 49-55% @ 0000-0000, 2024-09-30 2024. [Online]. Available: <https://doi.org/10.5753/sast.2024.3848>.
- [5] S. Ashiq *et al.*, "Challenges and barriers to software testing," *Bulletin of Business and Economics (BBE)*, vol. 13, no. 1, pp. 628-640, 2024. [Online]. Available: <https://doi.org/10.61506/01.00248>.
- [6] I. Zablah, L. Sosa-Díaz, and A. Garcia-Loureiro, "Relevance and Evolution of Benchmarking in Computer Systems: A Comprehensive Historical and Conceptual Review," *Computers*, vol. 14, no. 12, p. 516, 2025. [Online]. Available: <https://doi.org/10.3390/computers14120516>.
- [7] S. Mittal, J. S. Vetter, and D. Li, "A Survey Of Architectural Approaches for Managing Embedded DRAM and Non-Volatile On-Chip Caches," *IEEE Transactions on Parallel & Distributed Systems*, vol. 26, no. 06, pp. 1524-1537, 2015. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/TPDS.2014.2324563>.
- [8] C. Tirumalasetty and N. Reddy, "Exploring multi-level cache prefetching for fabric attached memory," presented at the Proceedings of the International Symposium on Memory Systems, 2026. [Online]. Available: <https://doi.org/10.1145/3767110.3767129>.
- [9] S. S. Bagui, C. Eller, R. Armour, S. Singh, S. C. Bagui, and D. Mink, "Analyzing Performance of Data Preprocessing Techniques on CPUs vs. GPUs with and Without the MapReduce Environment," *Electronics*, vol. 14, no. 18, p. 3597, 2025. [Online]. Available: <https://doi.org/10.3390/electronics14183597>.
- [10] V. Rodriguez-Bahena, L. Pizano-Escalante, O. Longoria-Gandara, and L. F. Gutierrez-Preciado, "Optimizing Pre-Silicon CPU Validation: Reducing Simulation Time with Unsupervised Machine Learning and Statistical Analysis," *Computers*, vol. 14, no. 9, p. 364, 2025. [Online]. Available: <https://doi.org/10.3390/computers14090364>.
- [11] M. Shafiabadi, H. Pedram, M. Reshadi, and A. Reza, "An accurate model to predict the performance of graphical processors using data mining and regression theory," *Computers & Electrical Engineering*, vol. 90, p. 106965, 2021/03/01/ 2021. [Online]. Available: <https://doi.org/10.1016/j.compeleceng.2020.106965>.

- [12] B. Gu, L. Zhao, C. Li, and X. Chi, "Review and analysis of performance prediction methods and tools for heterogeneous parallel programs," *CCF Transactions on High Performance Computing*, 2026/01/07 2026. [Online]. Available: <https://doi.org/10.1007/s42514-025-00236-z>.
- [13] G. Chennupati *et al.*, "A Scalable Analytical Memory Model for CPU Performance Prediction," Cham, 2018: Springer International Publishing, in *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation*, pp. 114-135. [Online]. Available: https://doi.org/10.1007/978-3-319-72971-8_6.
- [14] G. Mariani, A. Anghel, R. Jongerius, and G. Dittmann, "Predicting Cloud Performance for HPC Applications: a User-oriented Approach," presented at the Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Madrid, Spain, 2017. [Online]. Available: <https://doi.org/10.1109/CCGRID.2017.11>.
- [15] C. Rosales *et al.*, "Performance Prediction of HPC Applications on Intel Processors," in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 29 May-2 June 2017 2017, pp. 1325-1332. [Online]. Available: <https://doi.org/10.1109/IPDPSW.2017.45>.
- [16] A. Limaye and T. Adegbiya, "A Workload Characterization of the SPEC CPU2017 Benchmark Suite," in *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2-4 April 2018 2018, pp. 149-158. [Online]. Available: <https://doi.org/10.1109/ISPASS.2018.00028>.
- [17] A. Wurl, A. Falkner, P. Filzmoser, A. Haselböck, A. Mazak, and S. Sperl, "A Comprehensive Prediction Approach for Hardware Asset Management," Cham, 2019: Springer International Publishing, in *Data Management Technologies and Applications*, pp. 26-49. [Online]. Available: https://doi.org/10.1007/978-3-030-26636-3_2.
- [18] H. Oi, "Evaluation of Ryzen 5 and Core i7 Processors with SPEC CPU 2017," in *2019 IEEE International Systems Conference (SysCon)*, 8-11 April 2019 2019, pp. 1-6. [Online]. Available: <https://doi.org/10.1109/SYSCON.2019.8836790>.
- [19] C. Fooks, "Cpu hardware classification and performance prediction using neural networks and statistical learning," *International Journal of Artificial Intelligence and Applications (IJIAA)*, vol. 11, no. 4, 2020. [Online]. Available: <https://doi.org/10.5121/IJIAA.2020.11401>.
- [20] C. Fooks, P. Pal, R. Datta, and A. Segev, "Classification of Computer Hardware and Performance Prediction using Statistical Learning and Neural Networks," in *CS & IT Conference Proceedings*, 2020, vol. 10, no. 5: CS & IT Conference Proceedings. [Online]. Available: <https://doi.org/10.5121/CSIT.2020.100517>.
- [21] A. Mankodi, A. Bhatt, and B. Chaudhury, "Multivariate Performance and Power Prediction of Algorithms on Simulation-Based Hardware Models," in *2020 19th International Symposium on Parallel and Distributed Computing (ISPDC)*, 5-8 July 2020 2020, pp. 150-157. [Online]. Available: <https://doi.org/10.1109/ISPDC51135.2020.00029>.
- [22] A. Saad and A. El-Mahdy, "HPCCloud Seer: A Performance Model Based Predictor for Parallel Applications on the Cloud," *IEEE Access*, vol. 8, pp. 87978-87993, 2020. [Online]. Available: <https://doi.org/10.1109/ACCESS.2020.2992880>.
- [23] A. Al Sefat, G. M. Rasiqul Islam Rasiq, N. Nawjis, and S. K. T. Mehedi, "CPU Performance Prediction Using Various Regression Algorithms," Singapore, 2021: Springer Singapore, in *Proceedings of International Conference on Machine Intelligence and Data Science Applications*, pp. 163-171. [Online]. Available: https://doi.org/10.1007/978-981-33-4087-9_14.
- [24] B. Topçu and I. Öz, "Predicting the Soft Error Vulnerability of GPGPU Applications," in *2022 30th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 9-11 March 2022 2022, pp. 108-115. [Online]. Available: <https://doi.org/10.1109/PDP55904.2022.00025>.
- [25] C. Wei-wei, H. Wei, Z. Hai-long, Z. Guo-hui, M. Quan-qi, and H. Peng, "A Processor Performance Prediction Method Based on Interpretable Hierarchical Belief Rule Base and Sensitivity Analysis," *Computers, Materials & Continua*, vol. 74, no. 3, pp. 6119--6143, 2023. [Online]. Available: <https://doi.org/10.32604/cmc.2023.035743>.
- [26] M. B. Sharif, T. Hines, S. Ghafoor, M. Morales-Hernández, T. Dullo, and A. Kalyanapu, "A Performance Prediction Model for Structured Grid Based Applications in HPC Environments," in *2023 22nd International Symposium on Parallel and Distributed Computing (ISPDC)*, 10-12 July 2023 2023, pp. 77-84. [Online]. Available: <https://doi.org/10.1109/ISPDC59212.2023.00016>.
- [27] M. Cengiz, M. Forshaw, A. Atapour-Abarghouei, and A. S. McGough, "Predicting the Performance of a Computing System with Deep Networks," presented at the Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering, Coimbra, Portugal, 2023. [Online]. Available: <https://doi.org/10.1145/3578244.3583731>.

- [28] Y. Wang *et al.*, "A Detailed Historical and Statistical Analysis of the Influence of Hardware Artifacts on SPEC Integer Benchmark Performance," *IEEE Transactions on Computers*, vol. 73, no. 5, pp. 1262-1274, 2024. [Online]. Available: <https://doi.org/10.1109/TC.2024.3365941>.
- [29] X. Wang, "Hardware Failure Prediction Modes Powered by Artificial Intelligence," *Academic Journal of Science and Technology*, vol. 12, no. 3, pp. 199-201, %10/%10 2024. [Online]. Available: <https://doi.org/10.54097/5qrhwj73>.
- [30] J. Marquez and O. H. Mondragon, "Modelling the Impact of Cloud Storage Heterogeneity on HPC Application Performance," *Computation*, vol. 12, no. 7, p. 150, 2024. [Online]. Available: <https://doi.org/10.3390/computation12070150>.
- [31] X. Liu, "MambaCPU: Enhanced Correlation Mining with State Space Models for CPU Performance Prediction," in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6-11 April 2025 2025, pp. 1-5. [Online]. Available: <https://doi.org/10.1109/ICASSP49660.2025.10888500>.
- [32] S. Arslan and U. Demirbaga, "Explainable AI-Based Performance Prediction Using Optimised ML Models in Cloud Systems," in *2025 10th International Conference on Computer Science and Engineering (UBMK)*, 17-21 Sept. 2025 2025, pp. 1321-1326. [Online]. Available: <https://doi.org/10.1109/UBMK67458.2025.11206928>.
- [33] B. Xu *et al.*, "CAPSim: A Fast CPU Performance Simulator Using Attention-based Predictor," *arXiv preprint arXiv:2510.10484*, 2025. [Online]. Available: <https://doi.org/10.48550/arXiv.2510.10484>.
- [34] A. Jo. *CPU Cinebench R23 Scores*. [Online]. Available: <https://www.kaggle.com/datasets/alanjo/cinebench-r23-scores-may-2022>
- [35] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning* (no. 4). Springer, 2006.
- [36] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 12, no. 1, pp. 55-67, 1970/02/01 1970. [Online]. Available: <https://doi.org/10.1080/00401706.1970.10488634>.
- [37] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001/10/01 2001. [Online]. Available: <https://doi.org/10.1023/A:1010933404324>.
- [38] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189-1232, 2001. [Online]. Available: <https://doi.org/10.1214/aos/1013203451>.
- [39] T. CPU. *CPU Benchmark Comparison and Reviews*. [Online]. Available: <https://www.topcpu.net>
- [40] A. Tousi and M. Luján, "Comparative Analysis of Machine Learning Models for Performance Prediction of the SPEC Benchmarks," *IEEE Access*, vol. 10, pp. 11994-12011, 2022. [Online]. Available: <https://doi.org/10.1109/ACCESS.2022.3142240>.
- [41] Y. Wang, V. Lee, G.-Y. Wei, and D. Brooks, "Predicting New Workload or CPU Performance by Analyzing Public Datasets," *ACM Trans. Archit. Code Optim.*, vol. 15, no. 4, p. Article 53, 2019. [Online]. Available: <https://doi.org/10.1145/3284127>.
- [42] X. Liu, "Towards CPU performance prediction: New challenge benchmark dataset and novel approach," *Expert Systems with Applications*, vol. 298, p. 129520, 2026/03/01/ 2026. [Online]. Available: <https://doi.org/10.1016/j.eswa.2025.129520>.
- [43] G. Kaya, E. Şen, and O. Altay, "Predicting CPU Performance Score with Regression Analysis," (in en), *Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi*, vol. 16, no. 1, pp. 1-11, March 2025. [Online]. Available: <https://doi.org/10.24012/dumf.1493049>.
- [44] M. A. Al-Tarawneh, S. A. Al-Tarawneh, and K. S. Al-Maaitah, "Predicting Processor Performance Using Machine Learning Techniques: A Study on SPEC CPU2017 Benchmark Suite," *Int. J. Eng. Trends Technol*, vol. 69, pp. 108-117, 2021. [Online]. Available: <https://doi.org/10.14445/22315381/IJETT-V69I10P214>.
- [45] N. M. Upadhyay, R. S. Singh, and S. P. Dwivedi, "Prediction of multicore CPU performance through parallel data mining on public datasets," *Displays*, vol. 71, p. 102112, 2022/01/01/ 2022. [Online]. Available: <https://doi.org/10.1016/j.displa.2021.102112>.