Qualified Investigation of ML Models for Forecasting SPEC Benchmark Performance

Vedavyas Gurla[1], Sarlan Sarlan[2], Ilham Ilham[2], Aaluri Seenu[3], Vasavi Oleti[4], Danampalli Sravanthi[5] and Hassnien S. Al Hashemi[6]

[1]*Department of CSE, Geethanjali College of Engineering and Technology, 501301 Hyderabad, India*
[2]*Department of Chemistry Study Program, Universitas Sulawesi Tenggara, 93121 Kendari, Indonesia*
[3]*Department of CSE, Shri Vishnu Engineering College for Women, 534204 Bhimavaram, India*
[4]*KKR and KSR Institute of Technology and Sciences, Vinjanampadu, 522017 Vinjanampadu, Andhra Pradesh, India*
[5]*Department of Computer Science Engineering- Data Science, MLR Institute of Technology, 500043 Hyderabad, Telangana, India*
[6]*Department of Medical Lab., Dijlah University College, 10021 Baghdad, Iraq*
gvedavyas.cse@gcet.edu.in, sarlan@un-sultra.ac.id, ilham@un-sultra.ac.id, aaluriseenu@svecw.edu.in, vasavioleti26@gmail.com, dsravanthi@mlrit.ac.in ,hassnhashmi@duc.edu.iq

Keywords: ML, SPEC CPU2025, Model Metrics, Optimization.

Abstract: Performance prediction via simulations is laborious and tedious. To avoid this problem, one way is to use supervised learning to forecast how well a system will do on SPEC benchmarks. This year's SPEC CPU includes a publicly available collection of results from 43 standardized performance tests divided into 4 suites and run on a variety of hardware setups. In this study, we will examine the dataset and try to find the answers to these questions: Can we reliably forecast the SPEC outcomes from the dataset's setups, without actually running the benchmarks? Secondly, which software and hardware aspects are most crucial? On the third point, in regard to forecast time and inaccuracy, which hyperparameters and models work best? thirdly, is it possible to use historical data to foretell how future systems will be performing? Preparing data, choosing features, this talk covers a wide range of topics, including hyperparameter tuning, employing decision trees, random forests, multi-layer perceptrons, and multi-task elastic-net neural networks to evaluate regression models, and more. There are three stages to feature selection: deleting features with zero variance, removing features with strong correlation, and finally, using Functional Recursion Using permutation importance, elastic-net coefficients, or importance metrics depending on trees to filter out candidates. Searching the hyperparameter space with a grid, we select the best models. Afterwards, we compare and evaluate their performance. We prove that using the initial set of 29 features in tree-based models yields 4% or better accuracy in predictions. With 10 characteristics, both the Random Forest and Quick Decision Tree models keep their average errors at 5% and 6%, respectively.

1 INTRODUCTION

Research on the use of Machine Learning (ML) to estimate computer systems' performance and enhance system design is ongoing [1]. System designers and engineers can learn more about how different configuration changes affect system performance and use that information to make more informed design decisions with the predicted results. Before developing their solutions, vendors research the market to achieve ideal positioning. Predicting how well future system configurations will perform is, nevertheless, no easy feat [2].

People also look for optimal system setups to maximize performance or make logical purchases to enhance the cost-performance ratio. Customers may need performance statistics for new systems or configurations they have never seen before. Therefore, it is insufficient to only possess access to the outcomes of various workloads executed on large clusters of computers [3]. Research into performance prediction and evaluation is driven by these difficulties. Nevertheless, it is difficult to accurately forecast the performance of unknown setups when doing many tasks simultaneously, in terms of either execution time or throughput. It could necessitate the

ever-more-complicated process of exact analytical modeling of future systems brought about by developments in computer architecture. In addition, the most accurate forecasts may not be produced by modeling techniques that depend on laborious and comprehensive simulations [2], [4]. Therefore, we depend on regression models to forecast performance instead of fine-grained system modeling [5]. These models figure out how different system configurations affect how well they handle different kinds of workloads. Included in SPEC CPU2017 are suites of widely used compute-intensive benchmarks that primarily evaluate compilers, memory subsystems, and processor characteristics. There are four different benchmark suites that SPEC offers, each with its own set of practical, portable programs that address problems of varying magnitude [6]:

First, the floating point rate, FP_rate; second, the integer rate, Int_rate; and lastly, the integer speed, Int_speed. We construct supervised learning models based on software and hardware characteristics extracted from the SPEC CPU2017 public dataset, which contains benchmarks of systems used for computer analysis. This document contains the results of the benchmarking procedure. The Multitask Elastic-Net (MT_EN), Decision Tree (DT), Random Forest (RF), and Multi-layer Perceptron (MLP) estimators have formed the basis of our models.

Prediction accuracy of the SPEC benchmarks has been the subject of prior research, with a focus on neural networks. Conversely, the features' significance in relation to the prediction models' contributions has been disregarded or ignored. The primary objective of our research is to establish a machine learning pipeline that can rapidly and accurately predict SPEC CPU2017 performance using regression models, and to give a simplified method for thorough evaluation. The significance of numerous software and hardware features is highlighted in this paper, which also evaluates several models' latency and prediction error using both the full and selected feature sets. Researchers are also looking at the possibility of using historical data from current systems to foretell how those systems will perform in the future. The scikit-learn [7] ML package is utilized in the Python-written open-source code. For example, engineers may be able to narrow the design space with the help of the presented ML pipeline and analysis, and consumers may be able to give more weight to crucial aspects when making purchases.

The benchmark performance ratios are the goal variables (outputs) for our regression models. Ratings and rankings on performance ratios for speed and rate standards as shown in Figure 1.

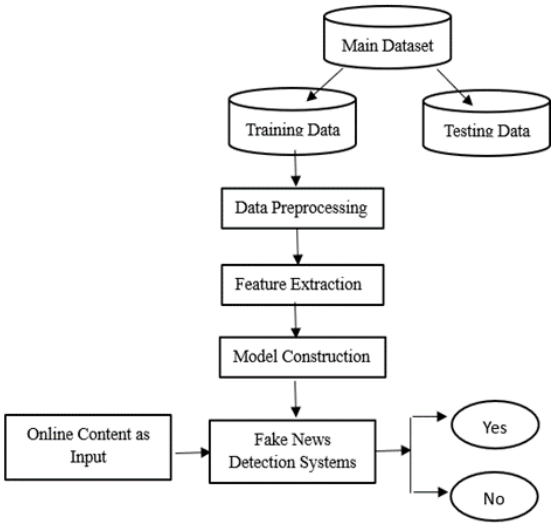

Figure 1: Fake news flowchart diagram.

$$Speed_{Ratio} = t_{Ref}/t_{SUT}$$

$$rate_{Ratio} = n \times (t_{Ref}/t_{SUT})$$

Where tRef is the time spent on a reference computer with one thread (for speed benchmarks) or one copy of the benchmark (for rate benchmarks), tSUT is the time spent on the machine being tested with multiple threads or copies, and n is the number of benchmark copies.

2 LITERATURE REVIEW

2.1 Computer Architecture and Systems Machine Learning Surveys

Computer systems and architecture have long been fine-tuned to run machine learning (ML) models efficiently. Rethink the relationship among ML and systems to transform computer architecture and system design. Both the completion of the virtuous cycle and an increase in designers' productivity are embraced by this. Our goal in this article is to provide a thorough overview of the literature on machine learning (ML) in relation to computer system design and architecture. Our first step is to create a taxonomy at a high level by thinking about the two most common uses of ML techniques in the field of architecture and system design: rapid predictive modeling and design process. Following that, we

provide a brief overview of the most popular ML strategies used to address the most prevalent issues in computer architecture and system design, as well as the problems themselves. We focus on computer architecture narrowly and assume data centers are gigantic computer warehouses. We also give some cursory attention to nearby computer systems like code generation as well as compiler, and they focus on how ML can transform design automation. We also see excellent future courses and prospects, and we think the community would benefit much from using ML to computer architecture and systems.

2.2 Specnet Predicts SPEC Scores Utilizing Deep Learning

The SPECnet is a deep neural network (DNN) that we demonstrate how to construct for the purpose of predicting SPEC® results. The SPEC CPU2006 suite has been around for over a decade (it was decommissioned in January 2018), and there are still thousands of submissions for its integer and floating point benchmarks. In order to train on the matching reported SPEC scores, we construct a DNN that takes hardware and software information from these submissions.

Then, for future machine configurations, we forecast scores using the learned DNN. With training and development/test errors ranging from 5% to 7%, we attain very high prediction accuracy rates of 93% to 95%. By meticulously modeling the performance of both the essential and non-essential parts of the system, it is possible to obtain an accuracy level of 97% to 98%, which is extremely close to what humans are anticipated to achieve. Applying SPECnet to SPEComp2012 and SPECjbb2015 is an additional step beyond the CPU2006 suite.

We demonstrate that such a DNN can also reasonably forecast (~85% accuracy) for these benchmarks, even though there are only hundreds of reported submissions for these suites. Our SPECnet solution is highly adaptable and expandable, built on top of the cutting-edge Tensor flow framework.

2.3 Using Public Datasets to Predict Workload or CPU Performance

Many different versions of general-purpose microprocessors are available on the market, all with identical functionality but different power consumption, frequency CPU cores, cache size, and memory bandwidth. Both the microarchitecture and the workloads being executed have an impact on their performance. The customer can't make an informed purchase decision without knowing the performance and pricing details, given a set of expected workloads. There are a plethora of benchmark suites available for use in evaluating CPU performance, and the outcomes for extensive sets of CPUs are frequently made public. Consumers seeking processor or workload efficiency statistics may locate repositories of benchmark results to be unhelpful at times. In addition, benchmark suites that aim to cover a wide variety of workloads could provide deceptive aggregate scores. then built a DNN model to understand the correlation between Intel CPU specs and Geekbench 3 and SPEC CPU 2006 performance, and then applied it to these issues. We demonstrate the ability to produce practical forecasts for novel processors and workloads. The two benchmark suites' performance scores are also compared and cross-predicted. This is the first literature to quantify such suites' self-similarity by results. This work aims to dissuade buyers from relying just on Geekbench 3, while simultaneously motivating researchers to assess their work using a wider range of workloads rather than just the SPEC CPU suites.

2.4 Machine Learning to Anticipate Computer System Design Performance Options

Computer companies invest a lot of time, energy, and capital into developing new systems and configurations; the success of these systems determines how much they can cut costs, how much they can charge, and how much market share they can win. Our primary focus in this effort is to streamline the architectural design and system design phases of parallel computers. Our approach uses neural network and linear regression models to predict the performance of any machine in the design space by extracting the performance levels of a tiny percentage of machines. We demonstrate that in the context of architectural design, a 3.4% error rate may be achieved by utilizing a small percentage of the design space (cycle-accurate simulations) to predict its performance completely.

We use Standard Performance Evaluation Corporation benchmark data to predict future system design performance. Focusing on systems with many processors, we demonstrate that our models can, on average, forecast future system performance to within 2.2% of the true value.

We are confident that these tools can greatly accelerate the exploration of the design space while also helping to reduce the associated research and development costs and time-to-market.

2.5 A Survey on Multi-Output Regression

A multitude of methods have been put forward in the past few years to tackle the ever-increasingly difficult undertaking of multi-output regression. Modern multi-output regression techniques, including those for issue transformation and algorithm adaptation, are reviewed in this article. Additionally, we showcase open-source software frameworks Public data sets for multi-output regression real-world challenges and the most prevalent performance evaluation methods.

3 METHODOLOGY

The reliability of neural network predictions on the SPEC benchmark has been the subject of earlier research. In addition, features' significance in contributing to the models has been disregarded or ignored. Our research aims to construct a machine learning pipeline that can rapidly and accurately predict SPEC CPU2017 performance using regression models, and then we will evaluate these models thoroughly. Finding which software and hardware features are most important and Evaluating prediction error and delay of various models on complete and decreased feature sets are two significant contributions of this work as shown in Figure 2.

Here, we forecast SPEC benchmark performance using supervised learning. SPEC is the Performance Evaluation Corporation. The public SPEC CPU2017 dataset contains 43 standardized performance tests from diverse system settings. These tests are organized into 4 suites. The following questions are intended to be answered by this paper's analysis of the dataset: I) Is it necessary to run the benchmarks in order to provide an accurate prediction of the SPEC results using the dataset's configurations? II) Which software and hardware features are most crucial?

In regards to prediction error and time, which hyperparameters and models provide the greatest results? and (IV) is it possible to use historical data to forecast how future systems will perform? We walk you through data preparation, feature selection, hyperparameter tuning, and regression model evaluation employing Multi-Task Elastic-Net, Decision Tree, Random Forest, and Multi-Layer Perceptron neural network estimators [8].

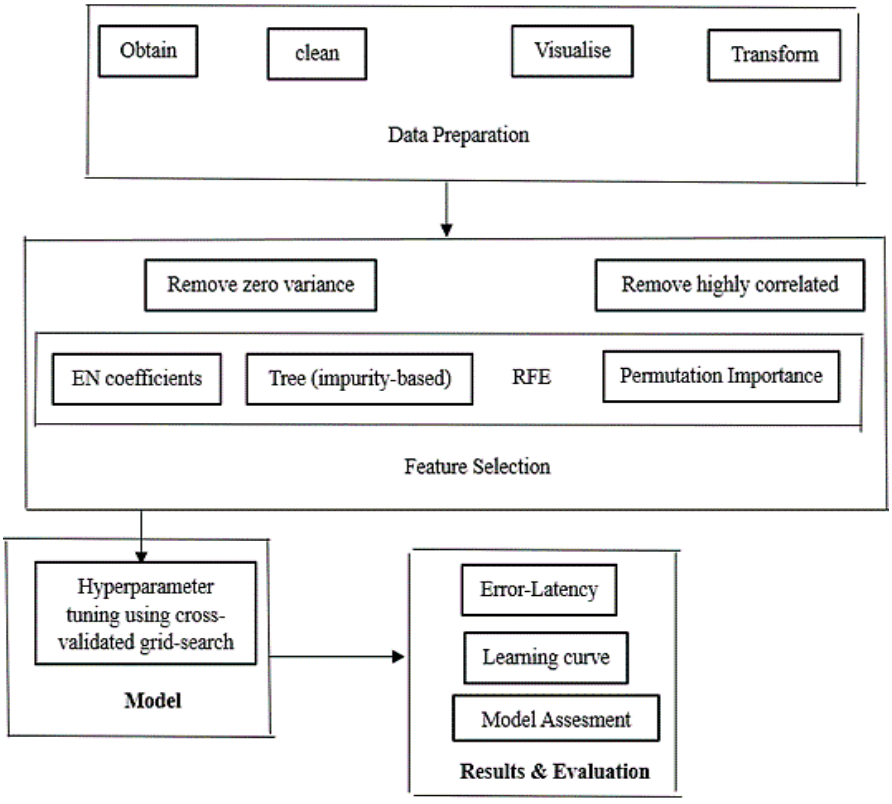

Figure 2: System architecture.

3.1 Modules

We designed the following modules for the project:

- Use the data exploration module to load data into the system.
- The module will be used to read and process data.
- Data would be separated into train and test using this module.
- Constructed models using ElasticNet, Decision Tree, Random Forest, NLP, and Voting Stacking. Algorithm accuracy calculated
- The user signup and login module allows for registration and login.
- User input: This modules provides prediction input.
- Prediction: final prediction showed.

Table 1: Feature set extracted from the publicly available SPEC CPU 2017 database.

Raw Feature	Description
arch	Architecture from lscpu
Nominal_mhz	Processing unit nominal clock frequency
max_mhz	Central processing unit maximum clock frequency
cpus	Number of CPU(s) from lscpu
threads_per_core	Thread(s) per core from lscpu
cores_per_socket	Core(s) per socket from lscpu
sockets	Scoket(s) from lscpu
numas	NUMA node(s) from lscpu
l1d_cache_kb	L1 data cache in KB
l1i_cache_kb	L1 instruction cache in KB
l2_cache_kb	L2 cache in KB
l3_cache_kb	L3 cache in KB
mem_kb	Main mem in KB (get it from the Memory field, if there is no /proc/meminf o)
mem_channels	Number of memory channels
channel_kb	Memory channel's capacity
mem_data_rate	Memory transfer rate in MT/s
os	Operating System

Table 1 lists all twenty-one raw features. Data related to the hardware includes the design, processor(s), and memory subsystem. Software specifics include the operating system, compiler, file system, parallel flag, and thread/copy count. To make them numeric, the operating system, compiler, and file system variables are categorical. Input variables might affect output depending on other input factors. For simplicity, this is the interaction effect. The equation cpus = sockets × cores_per_sockets × threads_per_core illustrates a three-way relationship [9].

4 IMPLEMENTATION

4.1 Algorithms

4.1.1 Multitask ElasticNet

Regularizing regression models with lasso and ridge penalties is elastic net linear regression. By learning from lasso and ridge regression's flaws, the method improves statistical model regularization [10].

4.1.2 Decision Tree

Decision trees are non-parametric supervised learning algorithms for classification and regression. Its tree structure contains a root node, branches, internal nodes, and leaf nodes.

4.1.3 Random Forest

The popular supervised machine learning technique Random Forest technique is used for classification and regression. The more trees a forest has, the stronger it is.

4.1.4 NLP

NLP algorithms are usually machine learning-based. NLP may use machine learning to automatically learn huge sets of rules by studying a corpus (like a book) and generating a statistical conclusion.

4.1.5 Voting Stacking

How votes are aggregated is the main distinction from stacking. Voting aggregates classifiers using user-specified weights, while stacking uses a blender/meta classifier as shown in Figure 3.

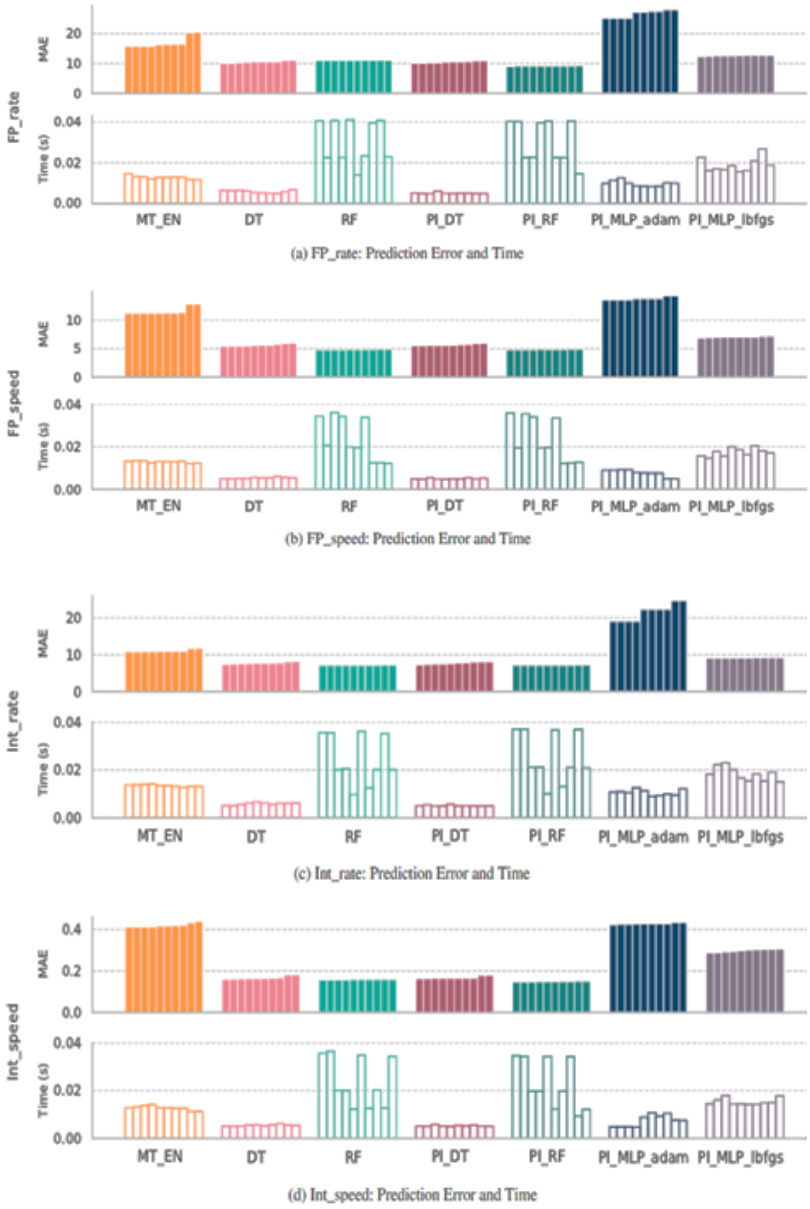

Figure 3. Comparison of forecasting time (in seconds) and mean cross-validation error (MAE) for the evaluated models. Lower values on both metrics indicate better predictive performance and computational efficiency.

5 EXPERIMENTAL RESULTS

To evaluate the developed system, a set of user interface screens and prediction workflows were tested. The system begins with a home screen that introduces the prediction tool and serves as the entry point for the user (Fig. 4). Following this, users are directed to a registration interface that supports the creation of user accounts and secure logins (Fig. 5). Once registered, users access the main interface where model selection, data upload, and configuration options are provided (Fig. 6). Here, users can upload their input feature values — these include system configuration details such as CPU parameters, memory details, and software-related metadata (Fig. 7). Upon submission, the system processes the input through the pre-trained models and generates performance predictions for the SPEC CPU2017 benchmark, which are then displayed clearly in the prediction output screen (Fig. 8).

Figure 4: Home screen.

Figure 5: User registration.

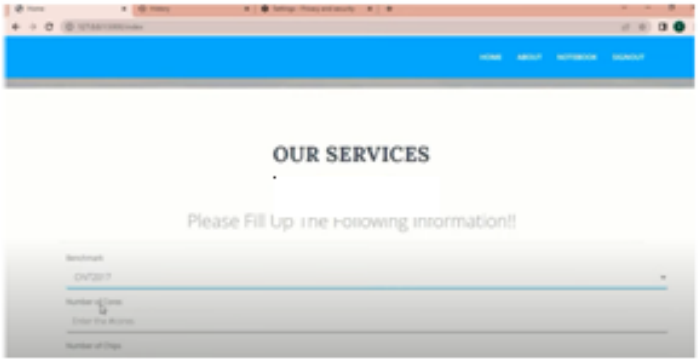

Figure 6: Main page.

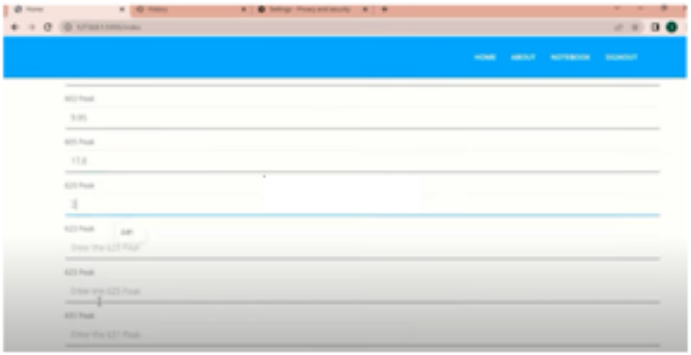

Figure 7: Upload input values.

Figure 8: Prediction result.

The prediction results demonstrate the feasibility and accuracy of our regression models in estimating benchmark performance without executing the full test suite. Tree-based models like Decision Trees and Random Forests consistently outperformed others in terms of both prediction accuracy and execution time. Importantly, the experimental interface enabled seamless interaction with the pipeline and visualization of results, validating both the backend model's performance and the usability of the developed platform.

6 CONCLUSIONS

This research looks at the possibility that supervised learning can forecast how well parallel systems will do on the SPEC benchmarks without actually running the benchmarks themselves. Extensive testing has proven that SPEC CPU2017 parallel and concurrent performance predictions are feasible. We have used grid search to discover the top ten most accurate models after investigating how changing the hyperparameters of the aforementioned four estimators affected their performance. Then, we compared the models' prediction delay and error rates. The most effective models have been those based on trees. They also find that smaller feature sets yield better results when using RFECV for feature selection. Additionally, in order to assess the accuracy of our data-driven performance predictions, we have examined learning curves of top tree-based models. With just 10% of the historical data used as the training set, we were able to demonstrate that this dataset is sufficiently predictive, but with 70% or more data, we were able to reduce 2-3 times the mean error. This dataset is four years old. It would be interesting to see how the dataset and fresh system generations affect the figures. As the final stage, we compared the final models' average goodness-of-fit (R2) and MAPE on the put-aside test set. Tree-based models (DT and RF) fared better overall and for specific benchmarks in R2 and MAPE. An explanation that stands out from the linear models is that they fail to account for some non-linear correlations (MT_EN). The fact that tree-based models typically perform better when many types of features are present is a probable reason, especially when contrasted with the neural networks MLPs. However, with thousands of samples or more, neural networks may do better. Once again, it will be fascinating to watch as the dataset grows and how these results evolve. With just 29 attributes, Decision trees and random forests may have MAPEs sub 4%. Models with fewer features benefit from random forests, which perform better with 10 features (1.5% < MAPE < 4.5% over four suites). On the other hand, decision trees are the way to go if interpretability is your top priority. Our regression models are more interpretable, have a better track record of properly predicting the SPEC CPU benchmarks, and provide more light on which software and hardware features are most important than in earlier research. By applying the RFE technique, we were able to determine that just a small subset of the available software and hardware attributes (less than five) are crucial to our models, and that a mere ten features are sufficient to produce very accurate predictions on this dataset. Our study provides an effective performance prediction, evaluation, and design space exploration pipeline.

REFERENCES

[1] N. Wu and Y. Xie, “A survey of machine learning for computer architecture and systems,” arXiv preprint arXiv:2102.07952, 2021. [Online]. Available: https://arxiv.org/abs/2102.07952.

[2] D. Das, P. Raghavendra, and A. Ramachandran, “SPECNet: Predicting SPEC scores using deep learning,” in Proc. Companion ACM/SPEC Int. Conf. Performance Engineering, Apr. 2018, pp. 29–32.

[3] Y. Wang, V. Lee, G.-Y. Wei, and D. Brooks, “Predicting new workload or CPU performance by analyzing public datasets,” ACM Transactions on Architecture and Code Optimization, vol. 15, no. 4, pp. 1–21, Jan. 2019.

[4] B. Ozisikyilmaz, G. Memik, and A. Choudhary, “Machine learning models to predict performance of computer system design alternatives,” in Proc. 37th Int. Conf. Parallel Processing, Sep. 2008, pp. 495–502.

[5] A. Tousi and C. Zhu, “Arm research starter kit: System modeling using gem5,” Arm Research, U.K., Jul. 2017.

[6] J. Bucek, K.-D. Lange, and J. V. Kistowski, “SPEC CPU2017: Next-generation compute benchmark,” in Proc. ACM/SPEC Int. Conf. Performance Engineering, 2018, pp. 41–42.

[7] F. Pedregosa, “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, Oct. 2011.

[8] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, “A survey on multi-output regression,” Data Mining and Knowledge Discovery, vol. 5, no. 5, pp. 216–233, 2015.

[9] D. Kocev, S. Džeroski, M. D. White, G. R. Newell, and P. Griffioen, “Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition,” Ecological Modelling, vol. 220, no. 8, pp. 1159–1168, Apr. 2009.

[10] D. Tuia, J. Verrelst, L. Alonso, F. Perez-Cruz, and G. Camps-Valls, “Multioutput support vector regression for remote sensing biophysical parameter estimation,” IEEE Geoscience and Remote Sensing Letters, vol. 8, no. 4, pp. 804–808, Jul. 2011.