Journal of High School Science

Using Machine Learning to analyze and predict the effect of different CPU features on CPU benchmarks for edge computing

Ling Y[1], Memik S[2]

Submitted: December 29, 2022, Revised: version 1, February 7, 2023, version 2, March 14, 2023
Accepted: March 15, 2023

Abstract

With the development of computer manufacturing technology, the Central Processing Unit (CPU) industry has seen exponential growth in all CPU features. Several essential CPU features that contribute to better performance are (i) L2 + L3 Cache, (ii) Thermal Design Power (TDP), (iii) Clock Speed, (iv) Turbo Speed, (v) Core Count, and (vi) Hyper-threading/Multi-threading. The performance or "power" of CPUs grew accordingly. With the explosive increase in computing power, more and more ideas that were, in the past, in the realm of fiction, become implementable. Amongst those ideas is Edge Computing. Edge Computing is a distributed framework of a computing system that processes the data at the data source to decrease the workload for the central server, thus greatly improving response time and broadening bandwidth availability when given a large dataset to process in a short period of time. To support Edge Computing, the CPU must have a superior CPU benchmark. This study focussed on several specific applications of the CPU benchmark for Edge Computing using: (i) 7-zip and (ii) Blender. The goal of this research was to use the dataset consisting of current CPU models in the market to train Machine Learning (ML) models to determine which CPU feature(s) is the most effective in improving each of the CPU benchmarks.

Keywords: CPU, Edge computing, CPU features, CPU benchmarks, Machine Learning, Clock speed, Core count, L2+L3 cache, Thermal design power, Hyper threading

---

[1]Corresponding author: Yihan Ling, Choate Rosemary Hall, 333 Christian Street, Wallingford, CT, United States. carson.ling1102@outlook.com
[2]Seda Memik, Instructor, Electrical and Computer Engineering Department, Northwestern University, 633 Clark Street; Evanston, IL 60208. seda@northwestern.edu

Introduction

The CPU, or Central Processing Unit, is key to almost all digital devices. One can compare it with the brain in the human body. Just as its name suggests, a CPU is the central processing component that executes instructions from the random access memory; RAM. The computations it performs can range from simple arithmetic calculations to running complicated scientific models. At its core, a CPU is composed of billions of microscopic transistors. The transistors do all the work of computing by turning on and off, regulating the zeros and ones that make up everything in the device. The more transistors a CPU possesses, the more powerful it is. With the development of technology, the number of transistors in a CPU is increasing exponentially. Gordon Moore proposed the idea of Moore's Law in 1965, which predicts that the number of transistors in a silicon chip will double every year (1). The trend of transistor numbers in CPU seems to be following Moore's prediction in recent years. The increase in the transistor numbers allows CPU designers to create new or improved features for CPUs.

A CPU has a perhaps more common name; viz. 'chip'. It is the chip we talk about when choosing laptops or gaming computers. When comparing different models of CPUs, we hear about their different features: Cores, Clock Speed, Cache size, threads, etc. As the number of transistors increases in the CPU, these different features improve considerably. For example, IBM first opened the door to the world of multi-core CPUs in 2001 by releasing the model POWER4 (2). Today, the best CPUs can have as many as sixty-four cores. The same trend applies to all the other CPU features discussed earlier.

With better CPU hardware features comes better performance. However, when discussing performance, it is best not to narrow the discussion to only the "overall performance" of a CPU. This is because a CPU is not restricted to only performing one specific task. When it comes to various types of computation, different elements of the CPU are more important than others. For instance, some programs might prefer a higher calculation speed over file encoding speed. To access CPU performance for individual needs, experts have developed CPU benchmarks. Each unique benchmark has its focus on a specific part of a CPU model's performance. Several common real-world CPU benchmarks include 7-zip, Blender, and Handbrake (3).

All CPU features are distinctive in that they all serve different purposes. Therefore, enhancement in one specific feature will add differently to the CPU performance and benchmarks. Some CPU features might influence the 7-zip benchmark more than Blender for instance. This research paper hopes to address the relationship between CPU features and benchmarks. In doing so, when engineers wish to increase a specific CPU benchmark, they can predict which CPU feature(s) would lead to the most enhancement in that specific CPU benchmark, saving time and resources in achieving their goal.

One of the subjects that engineers are paying great attention to is Edge Computing. In a world of big data, Edge Computing is rising to prominence to become one of the essential technologies for data processing since it achieves real-time computing and reduces delays in networks and data centers. Without Edge Computing, some new technologies that require processing of a large amount of information in a short amount of time - driverless vehicles for instance - would be overwhelmed by data (4).

Given the increasing importance of Edge Computing, this paper will also study the relationship between CPU features and benchmarks for Edge Computing specifically. This means that the study will, in addition to an unrestricted dataset, study ML models based on a restricted dataset that contains CPU models that are suitable for Edge Computing and focus the analysis on those CPU benchmarks that make Edge Computing more efficient. Specifically, recognizing the fact that an Edge Computing device must not overheat easily and consume too much power, the TDP measurement is one of the main restricting factors to a CPU's ability to take on the role of an Edge Computing device. Therefore, the restricted dataset studied by this research is obtained by taking a subset of all CPU models with TDP values less than or equal to 50 Watts from the unrestricted dataset.

Related work

In one paper, Baldini et. al. (5) used Machine Learning Algorithm to predict Graphics Processing Unit (GPU) performance based on CPU features and application runs. The training phase of the model input a set of data of existing applications and CPU features. The fine-tuned model could eventually take a new set of applications and CPU features and produce predictions on the corresponding GPU performance. With a set of 18 different benchmarks, Baldini et. al. were able to show an accurate prediction for two high-end GPU models, Tesla® and FirePro®, with an accuracy between 77% and 90%. They thus demonstrated that it was possible to use ML models to find the relationship between GPU performance and CPU features.

Zhang et. al. (6) studied the relationship between a high performance computing (HPC) system's temperature implication and different application runs using ML algorithms. By using a Gaussian process model, a neural network based model, and a linear regression based model, they were able to make predictions of a system's run-time temperature in order to improve thermal management.

Lou et. al. (7) proposed the idea that with the temperature model of an HPC system, it was possible to devise a thermal-aware task placement algorithm (TAPS). The algorithm could use the temperature prediction of a system over time to distribute tasks across different units of a computing system to reduce the peak temperature of each computing unit without reduction in performance. This is an example of the utilization of prediction in a specific computer benchmark (temperature in this case) to improve computer performance.

Background and Materials

Definitions

1. Core - A CPU core is a CPU's processor. Originally, one CPU only had one core. Eventually, engineers realized that one core was not enough to process a large amount of workload. Hence, multi-core CPUs were designed, which are better at multi-tasking (8, 9).
2. Cache - CPU cache is made up of static random-access memory. It is used to store data that needs to be accessed frequently. It is designed so that it can be accessed in a short amount of time to reduce the latency of access. CPU cache is divided into different levels: L1, L2, and L3. The storage size increases from L1 to L3. The access speed decreases from L1 to L3 (10).
3. Clock Speed & Turbo Speed - The Clock Speed of a CPU is the measure of the number of cycles the CPU executes per second. The larger it is, the faster the CPU is. Turbo Speed refers to the maximum Clock Speed a CPU can achieve. When dealing with large workloads, a CPU can adjust its Clock Speed dynamically up to its Turbo Speed (11).
4. Thermal Design Power (TDP) - TDP is the maximum amount of heat a CPU generates under extreme workload (12).
5. Thread - A thread is a virtual core a CPU can simulate. Intel CPUs use Hyper-threading to generate multiple threads, while AMD CPUs use Multi-threading to generate multiple threads. Despite the different names, Hyper-threading and Multi-threading are substantially similar ways to simulate virtual cores (13).
6. Process Size - Process Size describes the size of a single process node in a CPU. A process node is the smallest possible element of a chip (14).
7. 7-zip - 7-zip is a measurement of a CPU's data compression and decompression speed. The benchmark's numerical value is a representation of a million instructions per second (MIPS) executed in two different tests: compression and decompression with the LZMA algorithm (15). A high 7-zip performance is important to Edge Computing because Edge technology requires the ability to turn a large amount of information into a small amount, while retaining accuracy.
8. Blender - Blender is a measurement of a CPU's 3D rendering speeds. In the dataset studied in this paper, the smaller the Blender value, the better the CPU performance. Blender is important to Edge Computing since Edge Computing applications oftentimes are applied on cameras that take images or videos of a 3D scene as input and use 3D rendering to turn the 3D scene into 2D information for processing.

Dataset

In this research, we used the data from notebookcheck.net, an online database containing more than one thousand CPU models and their corresponding features and benchmarks. The dataset offers a built-in filter tool. Therefore, it was easy to extract only the data that contained a 7-zip score or a Blender score. In addition to 7-zip and Blender, the dataset contains many other benchmarks. However, since this research is not concerned with those benchmarks, they will not be further explored in this paper.

Machine Language

1. Linear Regression - Linear Regression is a linear model that examines the relationship between a set of independent variables and a set of dependent variables (assuming the relationship is linear). The form of the linear regression model can be represented in the form of $y = bx + c$, where y are the dependent variables, b is the slope, c is a constant, and x are the independent variables. A Linear regression model examines the independent variables' ability in predicting the set of dependent variables. In addition, the model also is a good indicator of which independent variables in particular are significant in predicting the dependent variables (16, 17).

2. Multilayer Perceptron (MLP) - MLP is a type of neural network that feeds forward. It consists of nodes and layers that are connected in a direct graphical sense. There are three types of layers in an MLP model—the input layer, hidden layers, and the output layer. The number of nodes in each layer and the number of hidden layers are predefined before building a model. The data flow from the input layer to the output layer in a forward direction through the hidden layer(s) as part of the learning process. During the learning process, each node that is not a part of the input layer uses a nonlinear activation function and uses back propagation to achieve supervised learning. MLP is a multiple feature regression model, meaning that its input can accept multiple inputs at the same time and consider one scenario as a combination of all those inputs (18,19).

3. Decision Tree - Decision Tree is a supervised machine learning algorithm that runs based on a tree-like structure that consists of a root node, branches, internal nodes (or decision nodes), and leaf nodes. The root node is the input, which feeds information into internal nodes that decide to which node the information will go next, which eventually leads to one of the possible outcomes—a leaf node. Decision Tree is a multiple feature regression model (20).

Source code

All of the following files of datasets and ML models' source code discussed in Section 4 and 5 can be found in the GitHub repository (21).

Dataset preprocessing

To make the information easier for the code to run on, we preprocessed the dataset into a .csv file and separated the information that was previously combined into one column. For example, the column, Cores/Threads, was separated into two columns as Cores and Threads. We also unified the units of all the data. Some data in L2 Cache + L3 Cache were measured in KB instead of MB, so we preprocessed all the L2 Cache + L3 Cache into MB.

Not every model in the dataset has a 7-zip benchmark as well as a Blender benchmark. Therefore, the 7-zip benchmark and Blender benchmark were separated as two different datasets and a .csv table was created for each dataset. Table 1 is a portion of the processed .csv dataset as an illustration.

Python library

We used python to build the ML models for this research. In addition to basic python code libraries, several other libraries were incorporated into the code. We used Scikit-Learn to build the ML models. Scikit-Learn is a free ML library available in Python in the form of a package named sklearn. For processing the dataset, we used the NumPy (22), Pandas (23), and Matplotlib (24) libraries .

Table 1: CPU features of existing models and their corresponding 7-zip benchmark

Model	L2 Cache (MB)	L3 Cache (MB)	TDP (Watt)	Clock Speed (MHz)	Turbo Speed (MHz)	Cores	Threads	Process (nm)	7-Zip
AMD Ryzen Threadripper PRO 3975WX	16	128	280	3500	4200	32	64	7	141386
AMD Ryzen Threadripper PRO 3995WX	32	256	280	2700	4200	64	128	7	140172
…	…	…	…	…	…	…	…	…	…
Intel Celeron N2820	1	0	7.5	2170	2390	2	2	22	2891
AMD Ryzen 5 PRO 2500U	2	4	15	2000	3600	4	8	14	2794

Methods

Environmental setup

We ran the following experiments using Python 3.9 on a 2018 MacBook Pro (15-inch) with a 2.6 GHz 6-Core Intel Core i7 Processor, a 16 GB 2400 MHz DDR4 Memory, a Raden Pro 560X 4 GB Intel UHD Graphics 630 1536 MB Graphics, and macOS Monterey 12.5 installed. We ran the python code using PyCharm IDE 2021.2 Community Edition. We used the Pandas and NumPy library to input the dataset and create NumPy data structures, we used the Scikit-Learn library to create Linear, MLP, and Decision Tree regression models, and we used the Matplotlib library to create a visualization of the results.

Datasets

We ran all the experiments on two different sets of datasets. We created a large dataset composed of all the available data and a smaller dataset more "personalized" for Edge Computing. The smaller dataset was created because there are certain limitations to the types of CPU used in real-world Edge Computing applications. For example, it is unrealistic to have a CPU that has a high TDP to act as an Edge Computing device since an Edge Computing device is distributed at the sources of data input so it should be small in size and not consume too much power.

Large Dataset results

This section presents the experimental results of the large dataset with 244 CPU models for

the 7-zip dataset and 248 models for the Blender dataset.

7-zip

Linear regression

Using the **linear_model** module from **sklearn** library, we built a linear regression model for each of the eight features. We assigned the CPU features as the x-axis and the 7-zip benchmark as the y-axis and trained the linear regression model. We used **Linear Regression ( ).score( )** function in **linear_model** to determine the R-Squared score for each linear model and **Linear Regression ( ).coef_** to determine the linear coefficient. We then used matplotlib to form a visualization of the model via a scatter plot representation of all the values in the dataset and a line representing the linear regression result.

Figures 1-8: graphs generated by matplotlib that show the linear regression result of L2 Cache, L3 Cache, TDP, Clock Speed, Turbo Speed, Core Count, Thread Count, and Process size, respectively (7-zip).

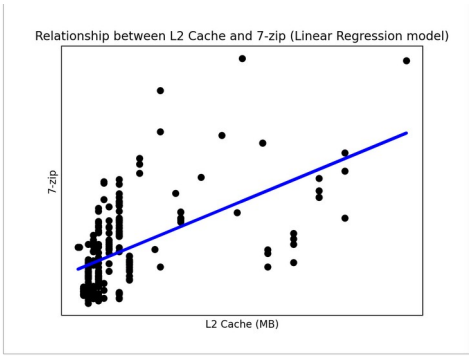


Figure 1

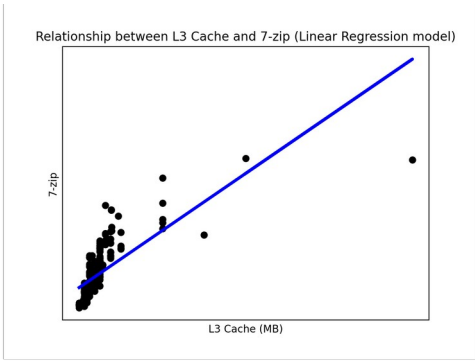


Figure 2

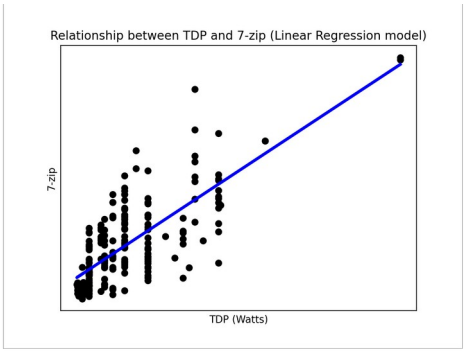


Figure 3

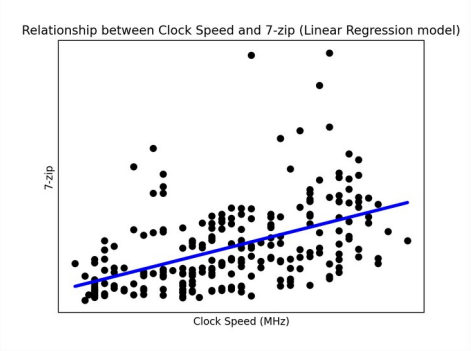


Figure 4

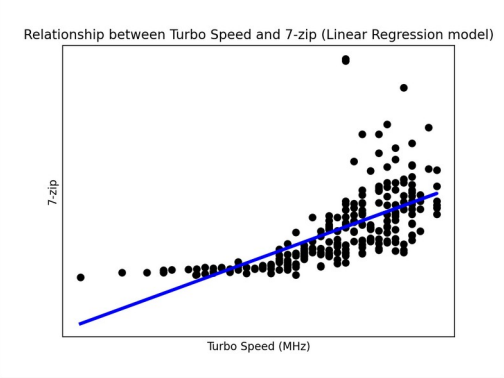


Figure 5

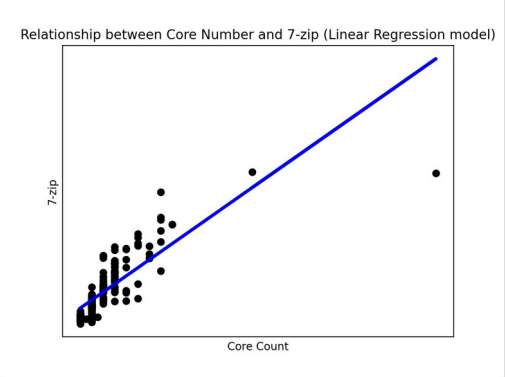


Figure 6

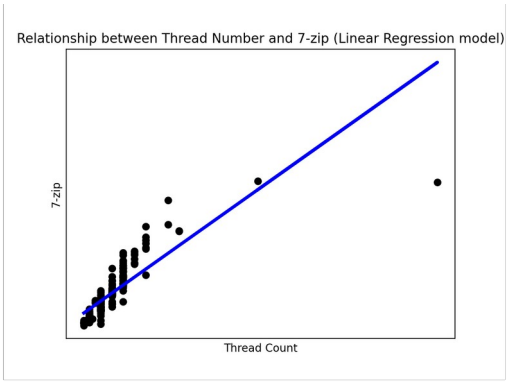


Figure 7

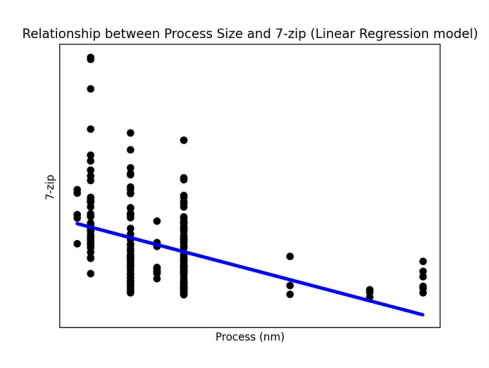


Figure 8

As is evident from the graphs, none of the data fit the linear model. Most of the data were scattered and did not follow a strictly linear behavior. However, four CPU features seemed to fit the regression line better than the others —L3 Cache, TDP, Core Count, and Thread Count. They seemed to be more consolidated around the linear line. L2 Cache had a low correlation with its linear line. TDP increased linearly; however, many of its data points were far from the line of best fit. More specifically, many of the data points with the same TDP value differed significantly in 7-zip value. This is because as technology develops, the general performances of CPU models improved significantly. However, their TDP values stayed similar to the existing values, much like the older models. Therefore, we can see this large variation of the 7-zip values for the same TDP value. For example, AMD 2017 model Ryzen 3 1200 had a TDP value of 65 watts and a 7-zip value of 11980. AMD 2022 model Ryzen 7 7700, on the other hand, presented with a 7-zip score of 85260 for the same TDP value. Clock Speed also increased linearly, but its datapoints were too scattered. Turbo Speed trended in an almost flat line with no increase in the 7-zip benchmark. However, after a certain point, the 7-zip score increased exponentially. Process Size not only showed a

low correlation with the linear model, it did not have a significant effect on 7-zip score as well. Even though Figure 8 showed that the highest 7-zip scores were associated with a low process size, this size also included a large number of data points with a low 7-zip score. The portion of the small process size—high 7-zip score points was significantly lesser than the portion of the small process size—low 7-zip score data points.

The data trend observed in Figures 2, 6, and 7 was that with an increase in the independent variable's value, the 7-zip score "leveled off". The data point that was the largest in the x value presented with a 7-zip number significantly lower than the value predicted by the linear model. This phenomenon might indicate that once the CPU features exceed a certain level, the 7-zip score plateaus out. However, there were only a limited number of CPU models with CPU features that high in our dataset. At the same time, it was observed that the other datasets that did not show a trend of leveling off (L2 Cache, TDP, Clock Speed, Process, etc.) contained sufficient data at the high end of the X-axis. This might be because that we cannot yet develop such high-end models for L3 cache, Core Count, and Thread Count. It could also be a data anomaly or artifact that is skewed in the L3 cache, Core Count, and Thread Count values. Therefore, the did not provide conclusive evidence of the hypothesis.

Table 2: R-Square and Coefficient for Linear Regression Models (7-zip)

	L2	**L3**	**TDP**	**Clock**	**Turbo**	**Cores**	**Thread**	**Process**
R-Squared	0.319	0.522	0.565	0.231	0.422	0.650	0.682	0.173
Coefficient	2432.20	837.03	449.08	13.96	19.42	3712.45	1930.8	-2049.04

As shown in Table 2, the R-Squared values for all of the models were low. The highest was only 0.682 (Thread Count), and the lowest was 0.173 (Process Size). This result once again confirmed that linear regression was not the best, or appropriate model to analyze this dataset.

Multilayer Perceptron (MLP)

Using the **MLPRegressor** module from **sklearn.neural_network** library, we built a multiple feature regression MLP model. A single data point consisted of eight CPU features (L2 Cache, L3 Cache, TDP, Clock Speed, Turbo Speed, Core Count, Thread Count, and Process Size) of the CPU model as input and the 7-zip score of the CPU model as the output. We randomized the order of CPU models in the dataset and separated 80% of the data (195 data points) into the training dataset and 20% of the data (49 data points) into the testing dataset. This separation was achieved through the **train_test_split** module in **sklearn.model_selection** library. For the MLP model, we chose a five-layer model with three hidden layers. The hidden layers had 64, 16, and 4 nodes going from the input layer to the output layer. An MLP model was then trained using the training dataset only.

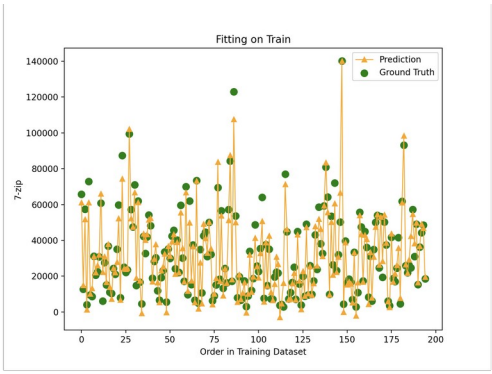


Figure 9: Matplotlib visualization of MLP model with training dataset as input (7-zip)

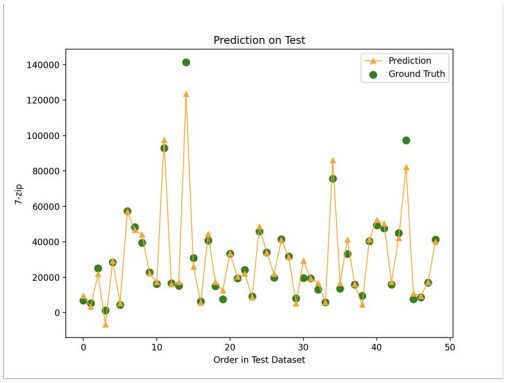


Figure 10: Matplotlib visualization of MLP model with testing dataset as input (7-zip)

In Figures 9 and 10, the green dots represent the ground truth value of each data point in the dataset. The yellow triangles represent the MLP model prediction after the training dataset was provided as the input to train the model. Note that for the training dataset prediction, this would be the second time the model sees the same data points as input. For the testing dataset prediction, this would be the first time the model has access to these data points.

As it can be seen from the graphs, the predictions are close to the ground truth values. This indicated that the MLP model was an optimal model for 7-zip score prediction. Therefore, this model may be useful when engineers are considering several different combinations of CPU features. They will be able to utilize this model to determine which combination of CPU features can yield the potential greatest 7-zip score without having to build a prototype for each combination.

MLP was a better fitting model than Linear Regression for this dataset because the determining element for the 7-zip benchmark might be (and perhaps is most likely to be) the combined effort from multiple features. However, the single feature Linear Regression model has the ability to only consider one factor at a time as input. However, MLP is more powerful in this perspective since it can accept multiple features as input at the same time and study all the features using multiple inputs. Therefore, the MLP model considered the situation that 7-zip might be influenced by multiple features at the same time.

To determine a quantitive analysis of the result obtained from the MLP model, we used the **mean_squared_error** module from **sklearn.metrics** to determine the mean squared error (MSE) of the model. This was calculated by taking the mean of the squares of each error value. We also used the **MLPRegressor.score( )** function to determine the R-Squared value of the MLP model. However, the MLP model described above is trained on a random 80% portion of the data in the dataset. To make sure every data point in the dataset was used in the training dataset and the testing dataset at least once, we used the cross-validation method to

run 10 different trials and calculated an average of the MSE and R-Squared in the 10 trials.

For the cross-validation process, we first randomized the order of the data in the dataset using the **shuffle** module from **sklearn.utils** library. Then, we sliced the data into 10 pieces with 24 data points in the first 9 pieces and 28 data points in the 10th piece. For each of the 10 trials, we used 1 of the 10 pieces that had not been used for the testing dataset before as the testing dataset and the rest, as training data. We recorded the MSE of both the training dataset and the testing dataset. We also recorded the R-Squared value of the models in the 10 trials.

Table 3: MSE and R-Squared values of 10 trials of MLP using cross validation (7-zip)

Trial	Train MSE	Test MSE	R-Squared
1	43492264.52	27887084.48	0.93949641
2	43132205.63	13552214.95	0.90497899
3	33352234.85	58645367.97	0.87804198
4	50934689.9	8922659.968	0.97724411
5	41154480.03	92976533.52	0.88909504
6	37909512.62	58870190.7	0.87883727
7	46770289.33	26382157.74	0.92432942
8	42662225.61	51507759.84	0.88005194
9	38518358.2	126581517.8	0.86401012
10	41983805.18	51868344.25	0.93951993
Avg:	**41991006.59**	**51719383.13**	**0.90756052**

The 0.91 R-Squared value from Table 3 provided an indication that the MLP model was a good predictor for the 7-zip benchmark using CPU features. It also showed that there was indeed a high correlation between CPU features and the 7-zip benchmark. While the average Training and Testing MSE might seem large to reach such a conclusion, we need to consider that the original 7-zip values were themselves high; the square of the mean of the 7-zip values in this dataset was about 1238798251. Compared to the mean square value of the dataset, the average MSEs are only 3.39% and 4.17% respectively for the Training and Testing results. Similarly, the difference between the average Training MSE and Testing MSE might seem large, however, it represents only 0.79% of the average MSE. This difference is normal since Testing data are ones that the model had only seen for the first time.

Decision Tree

We trained a multiple feature regression Decision Tree model using the **DecisionTreeRegressor** module from **sklearn.tree** library. The process of creating the Decision Tree model was similar to that of the MLP model. The input for the Decision Tree model consisted of the eight CPU features (L2 Cache, L3 Cache, TDP, Clock Speed, Turbo Speed, Core Count, Thread Count, and

Process Size) of a CPU model. The output was the 7-zip score of that CPU model. We randomly assigned 80% of the data to the training dataset and 20% to the testing dataset.

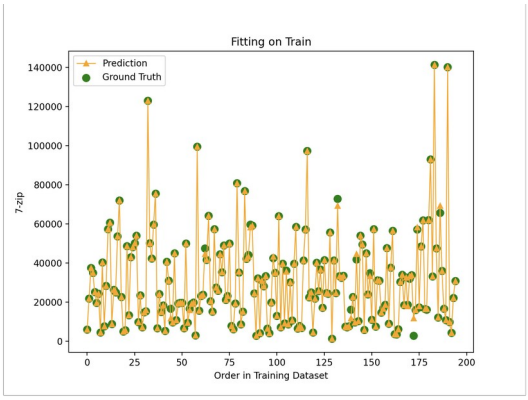


Figure 11: Matplotlib visualization of Decision Tree model with training dataset as input (7-zip)

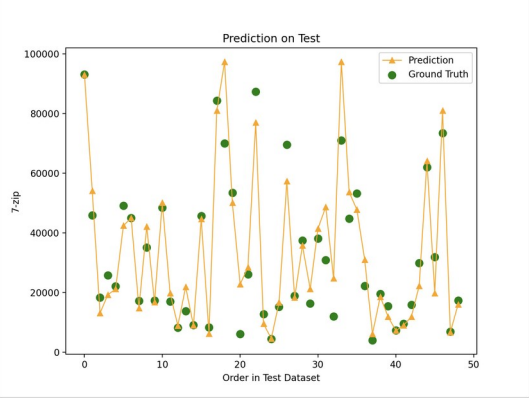


Figure 12: Matplotlib visualization of Decision Tree model with testing dataset as input (7-zip)

In Figures 11 and 12, the green dots represent the ground truth value of each data point in the dataset. The yellow triangles represent the Decision Tree model prediction yielded by the Decision model.

The Figures show that the result produced by the Decision Tree model was slightly better than the result from the MLP model studied previously. There were more overlapping prediction and ground truth value points in the Decision Tree model. This was especially evident in Figure 11. Therefore, this Decision Tree model is also a suitable model to use in determining which combination of CPU features may yield the potential greatest 7-zip score without having to build a different prototype to study a different combination of features.

The Decision Tree model is also a multiple feature regression model. Similar to the MLP model, it analyzed the combined effect of different CPU features on the 7-zip rating. Hence, it is a better model to study the effect of different inputs simultaneously, than the single feature linear regression model. To determine a quantitive analysis of the result obtained from the Decision Tree model, we studied the MSE and R-Squared values for 10 cross-validation trials. The process used for the cross-validation trials was similar to that described in the MLP model section.

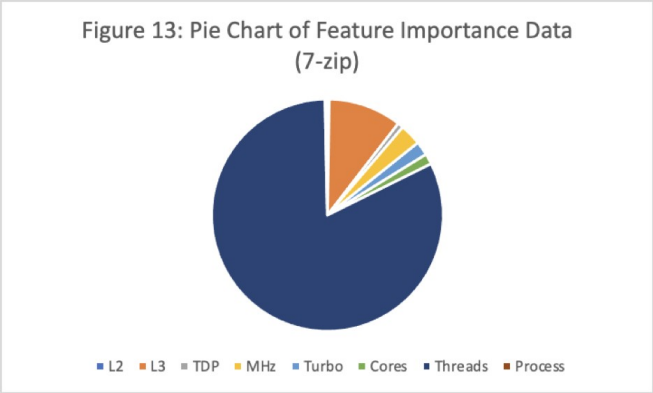
Table 4: Average MSE and R-Squared values of 10 trials of Decision Tree model using cross validation (7-zip)

	Train MSE	**Test MSE**	**R-Squared**
Avg:	1216197.383	43345071.43	0.92149906

For simplicity, Table 4 does not show the individual results from different trials. Instead, the average value is presented. There is no noticeable difference or outliers in individual data in the 10 trials, thus it is statistically sound to only present the average. Tables in the discussion below also do not contain any outliers or significant standard deviation around the average. A lower MSE in both the training dataset and the testing dataset suggested that the Decision Tree mode generally yielded a better prediction than the MLP model. A 0.92 R-Squared supported this conclusion. In addition, the Decision Tree model has the ability to determine the importance of each feature in arriving at the result. The importance values presented in Table 5 add up to a total of 1. The closer a particular value is to 1, the more influence it has on the output. We used the cross-validation method to perform 10 additional trials of the Decision Tree model. For each trial, we used **DecisionTreeRegressor.feature_importances_** to determine the feature importance for the eight features and recorded them.

Table 5: Average Feature Importance of Each CPU Feature in Decision Tree Model (7-zip)

	L2	L3	TDP	MHz	Turbo	Cores	Threads	Process
Avg:	0.003	0.102	0.008	0.030	0.020	0.015	0.820	0.003



As it can be seen in Table 5 and Figure 13, Thread Count was the most important feature and L3 Cache was the feature with the second highest importance. A 0.820 feature importance of Thread Count implied that Thread Count was the determining factor contributing to the 7-zip score. L3 Cache also exerted a high influence on the 7-zip score. The relationship between Figure 13 and the linear regression results in Figure 2 and Figure 7 now make sense. The two factors with the highest feature importance also presented with a more "consolidated" result in linear regression ("consolidated" in that the data points were not

scattered and fit the linear line better than others, but not in a high R-Squared value). However, the two other features that demonstrated a similarly good fit in the linear regression model, *viz.* Core Count and TDP, did not have as high a feature importance as L3 cache and Thread Count. This inconsistency may further support the conclusion that linear regression is not an appropriate model for predicting the effect of CPU features on 7-zip.

Blender

Linear regression

For building ML models to study the relationship between Blender score and different CPU features, we used the same process as that used for the 7-zip benchmark. The results of the Blender models were surprisingly different from the results yielded by the 7-zip models.

Figures 14-21: graphs generated by matplotlib that show the linear regression result of L2 Cache, L3 Cache, TDP, Clock Speed, Turbo Speed, Core Count, Thread Count, and Process size, respectively (Blender).

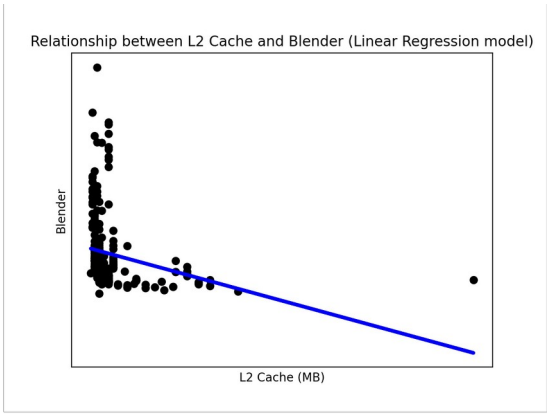


Figure 14

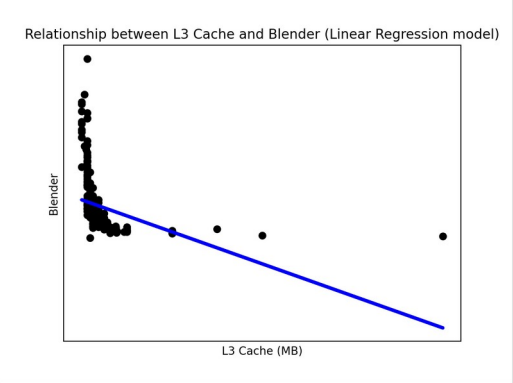


Figure 15

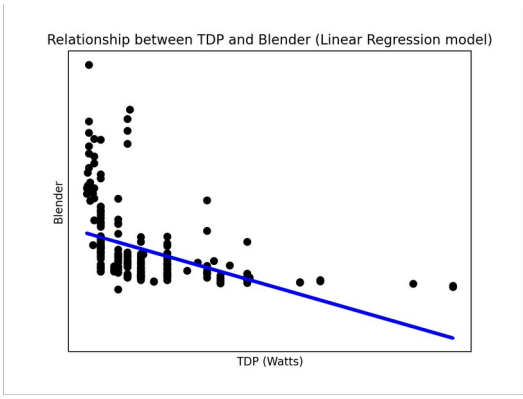


Figure 16

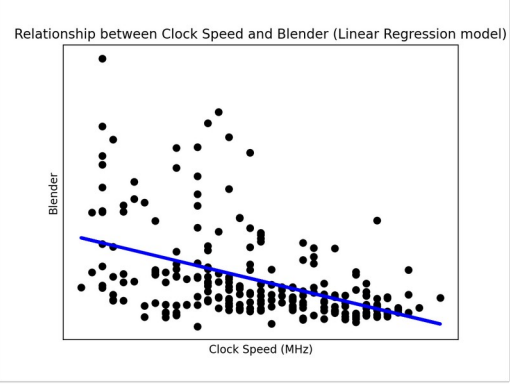


Figure 17

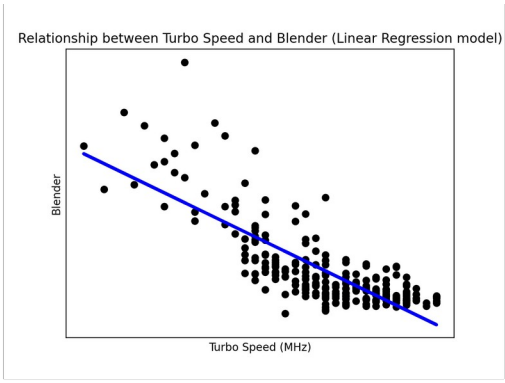


Figure 18

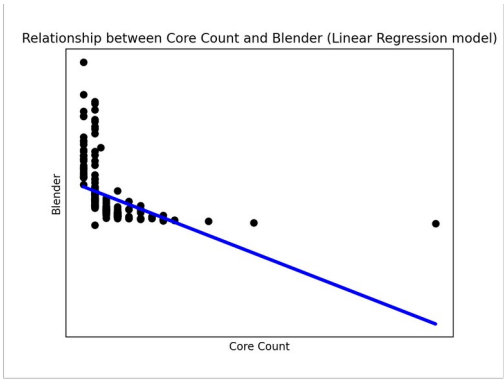


Figure 19

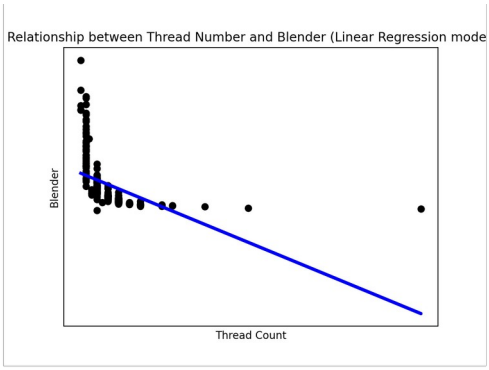


Figure 20

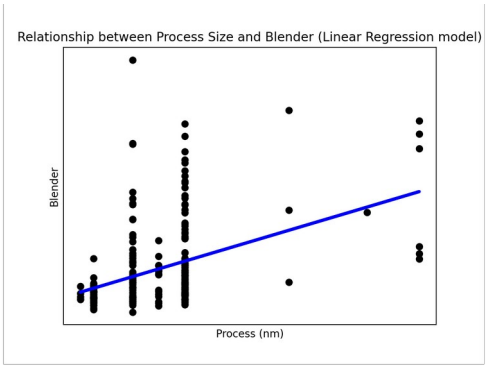


Figure 21

As it can be seen from the graphs, none of the CPU features was linearly correlated to the Blender benchmark. The linear model explains most of the variation between Turbo Speed and Blender, however, even the points in Figure 18 are quite scattered and are not clustered around the line of best fit. There did appear to be a pattern in some of the graphs. In Figures 14, 15, 16, 19, and 20, going from left to right on the x-axis, the Blender values of the data points begin high, but then dropped significantly. Next, they started to flatten out.

Note that a decrease in Blender value means better performance. Therefore, if some of the figures suggested that there might exist a horizontal asymptote in the graph, it may suggest that after reaching some point in development in some CPU features, CPU performance might be harder to improve going forward. This hypothesis may only apply to L2 Cache, L3 Cache. TDP, Core Count, and Thread Count since horizontal asymptotes were only apparent in Figures 14, 15, 16, 19, and 20. This might seem like a similar trend as the one we found in 7-zip linear regression model. However, the L2 Cache did not have an "asymptote" in 7-zip, but one was clearly apparent in Blender. This difference may be because Blender and 7-zip are two completely different benchmarks. Each CPU feature would respond differently under the two benchmarks.

It is therefore not surprising that L2 Cache behaved differently when evaluated against the 7-zip benchmark than when compared against the Blender benchmark.

Table 6: R-Square and Coefficient for Linear Regression Models (Blender)

	L2	L3	TDP	Clock	Turbo	Cores	Thread	Process
R-Squared	0.0621	0.1212	0.2018	0.2142	0.6287	0.1897	0.2192	0.1664
Coefficient	-23.6196	-11.8438	-7.1890	-0.4014	-0.8258	-57.6400	-31.5085	65.0826

The R-Squared from Table 6 results confirmed that linear regression was not an appropriate model to predict CPU feature correlation against this benchmark. However, Turbo Speed showed an R-Squared of 0.6287. Although not high, when compared with other R-Squared values, Turbo Speed was the most linearly correlated with Blender. Although several other features ostensibly show a linear relationship with Blender at first glance (namely clock speed, thread count and core count), the core and thread count present with a cliff-drop L-shape, meaning that although core and thread count can effectively improve Blender values for models with small x-values, once the models' x-values exceed a certain point, core and thread count ceased to improve Blender value. In contrast, we continued to see a downward trend for Blender even at the right end of our dataset. This means that CPU models with a higher Turbo Speed may have the greatest effect on the Blender benchmark. This conclusion is also supported by the R-Squared values. Turbo speed has R-Squared value of 0.6287, and clock speed, core count, and thread have R-Squared values only about 0.2 – far less than that of Turbo Speed.

Multilayer Perceptron (MLP)

We trained a multiple feature regression MLP model to predict the Blender value using all eight CPU features. The dataset was divided into two sections: training and testing. The training and testing datasets comprised 80% and 20% of the data respectively.

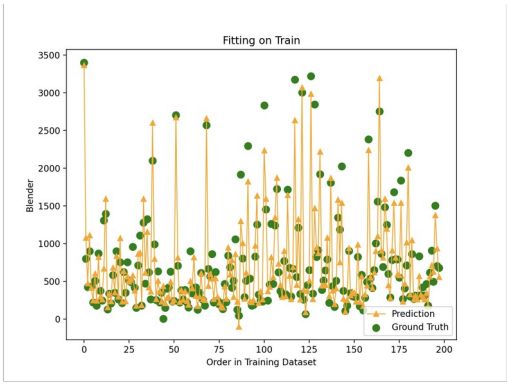


Figure 22: Matplotlib visualization of MLP model with training dataset as input (Blender)

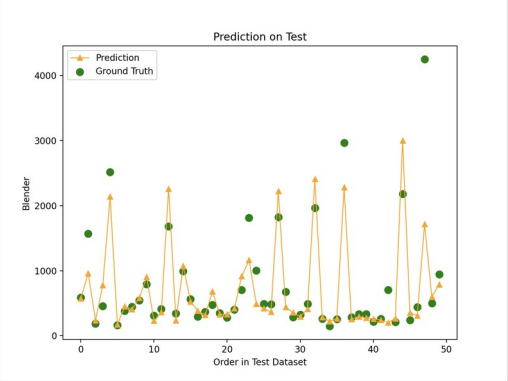


Figure 23: Matplotlib visualization of MLP model with testing dataset as input (Blender)

Figures 22 and 23 show that MLP seemed to perform reasonably well in predicting the Blender benchmark when given the CPU features as inputs. It was also evident that the model was equally good in predicting the benchmark when given the training or the testing data. We then ran a 10-trial cross-validation to determine the Training and Testing MSE and the R-Squared values.

Table 7: Average MSE and R-Squared values of 10 trials of MLP using cross validation (Blender)

	Train MSE	**Test MSE**	**R-Squared**
Avg:	122695.0639	143961.8008	0.717560062

The average R-Squared value was 0.72, which implied that MLP was a fair model for this prediction. The training MSE and the testing MSE were not significantly different. This suggested that the MLP model was equally good at predicting values from inputs it had seen before in training as well as predicting values from inputs that it had not seen before.

Decision Tree

We built a Decision Tree model using the same process that we used to derive the model for 7-zip. The result of this Decision Tree model (Figures 24 and 25) indicated that that this model yielded a greater number of correct predictions than the MLP model. However, there seemed to be a greater percentage of predictions correct in the Training set than those in the Testing set. Hence, we ran a 10-trial cross-validation process to determine the Training and Test MSEs as well as the R-Squared.

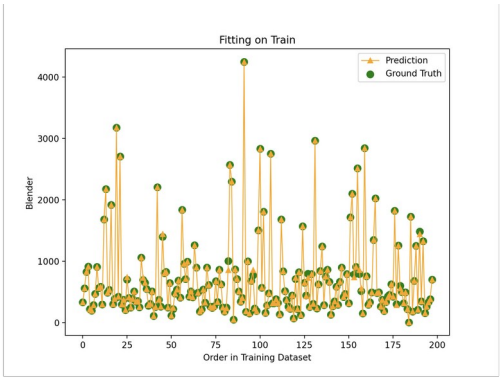


Figure 24: Matplotlib visualization of Decision Tree model with training dataset as input (Blender)

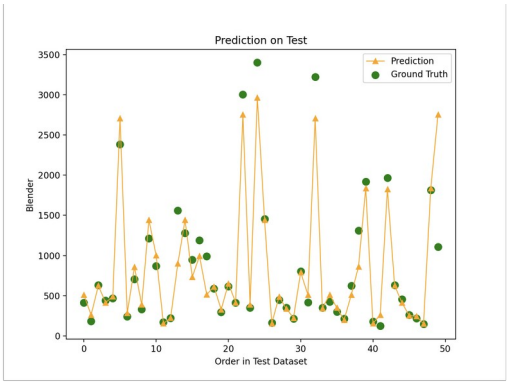


Figure 25: Matplotlib visualization of Decision Tree model with testing dataset as input (Blender)

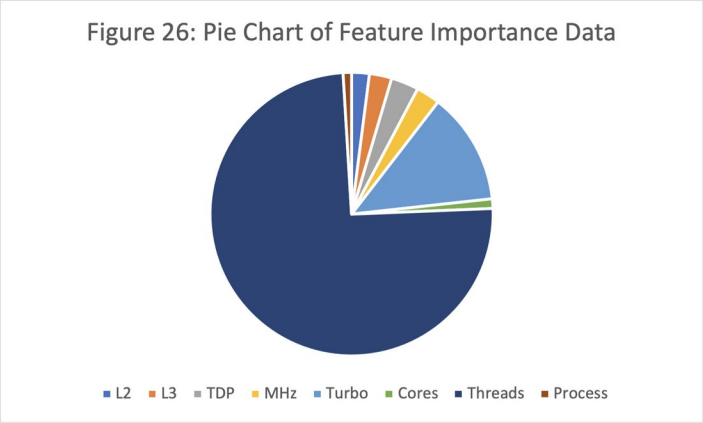
Table 8: Average MSE and R-Squared values of 10 trials of Decision Tree model using cross validation (Blender)

	Train MSE	Test MSE	R-Squared
Avg:	216.130478	64060.80074	0.863865698

Table 8 showed a significant gap between Train MSE and Test MSE. Hence, when the model encountered data that it had seen before as training input, it would yield a more accurate prediction. Therefore, it was concluded that this Decision Tree model was over-fitting. Nonetheless, both MSEs were lower than the MSEs yielded by the MLP model. The R-Squared value of the Decision Tree model was also higher than that of the MLP model. Therefore, we concluded that the Decision Tree model was generally more accurate than the MLP model for this experiment. We then ran another 10-trial cross-validation process to determine the feature importance assignments of this model.

Table 9: Average Feature Importance of Each CPU Feature in Decision Tree Model (Blender)

	L2	L3	TDP	MHz	Turbo	Cores	Threads	Process
Avg:	0.0203	0.0255	0.0316	0.0273	0.1280	0.0110	0.7468	0.0096



As it can be seen from the feature importance data in Table 9 and Figure 26, Thread Count continued to be the most important feature in determining CPU benchmark. The second-most important factor was Turbo Speed, which presented with the best R-Squared value in the linear regression model. Therefore, Thread Count and Turbo Speed were the two factors that most affected the the Blender benchmark. This might seem to contradict the conclusion of the linear regression model since Turbo Speed was the most-fitting factor in that model. However, based on the degree of fit, the linear regression model was not as good of a

prediction model as the Decision Tree model. Furthermore, the linear regression model only considered one CPU feature at a time; conversely, Decision Tree considered of all the CPU features simultaneously and assigned importance weightages. The Decision Tree model hence considers the relationship between CPU features and benchmarks as one that is not solely linear. The Decision Tree shows that Thread Count is the most fitting parameter while the linear does not, which indicates that the linear model may not capture all the variance in the data, which the Decision Tree model can.

Small Dataset results

We restricted the dataset to contain only CPU models with TDP values less than or equal to 50 Watts and ran the same test on the small dataset. We specifically chose to filter the data by restricting TDP value not because we wanted to optimize the degree of fit, but simply because of the fact that CPUs with relatively small TDP values were more accessible and applicable in many fields. One field that this paper focuses on is Edge Computing. Edge Computing cannot efficiently execute with high power consumption CPUs; therefore, we restricted the dataset by removing all the CPU models that required higher power input.

7-zip

The small dataset for 7-zip contains 179 data points.

Linear Regression

Figures 27-34: graphs generated by matplotlib that shows the linear regression result of L2 Cache, L3 Cache, TDP, Clock Speed, Turbo Speed, Core Count, Thread Count, and Process size, respectively (7-zip).

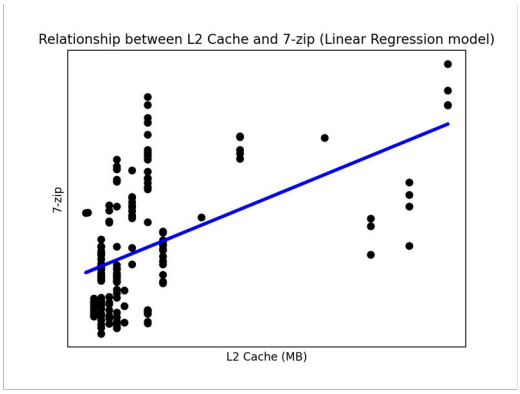


Figure 27

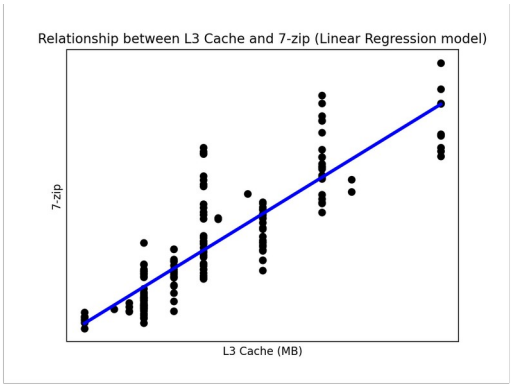


Figure 28

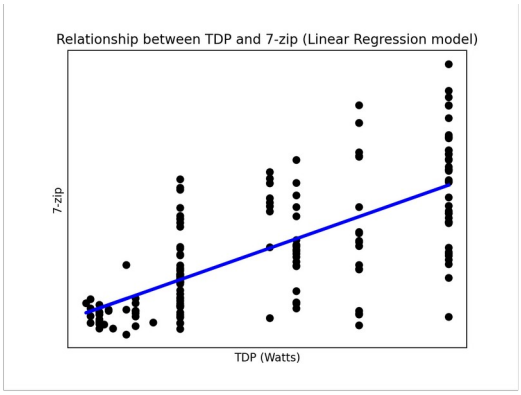


Figure 29

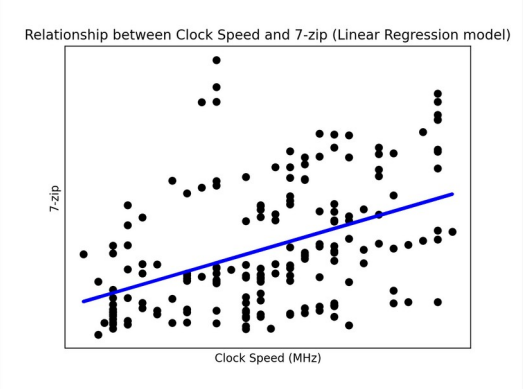


Figure 30

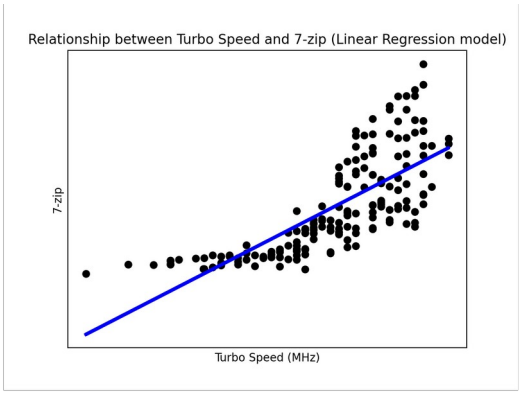


Figure 31

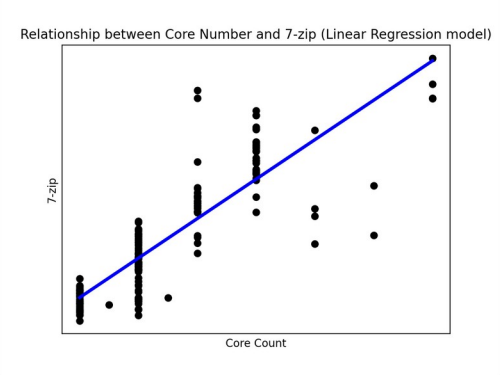


Figure 32

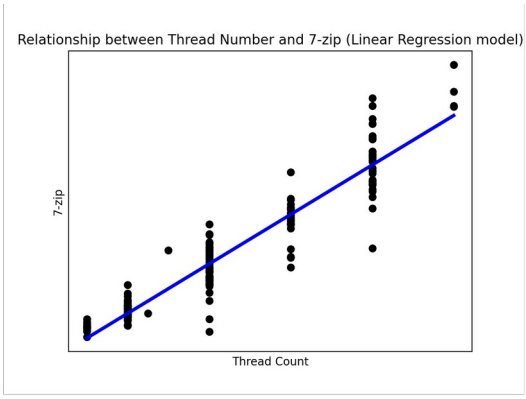


Figure 33

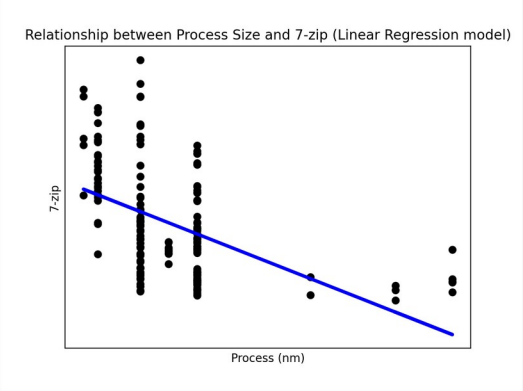


Figure 34

Table 10: R-Square and Coefficient for Linear Regression Models (7-zip)

	L2	L3	TDP	Clock	Turbo	Cores	Thread	Process
R-Squared	0.252	0.770	0.482	0.190	0.567	0.727	0.896	0.243
Coefficient	1673.85	2457.16	833.48	11.18	14.80	5371.80	3243.80	-1663.46

The data showed a significant increase in L3 Cache, Turbo Speed, Core Count, and Thread Count compared to the R-Squared values from the large dataset. In addition, those four CPU features also possessed a higher coefficient than the ones in the large dataset, which implied that with an increase in the same magnitude of x-value, there would be a higher increase in the 7-zip value for the CPUs in the smaller dataset than for the large dataset. This showed that for the small CPUs (represented by the small dataset), it was easier to increase the 7-zip values by strengthening the CPU features. In contrast, for larger CPUs (represented by the large dataset), it was harder to increase the 7-zip values. This further supported the conclusion that as the x-values increased (or when the CPU features increased over a certain point), it became more difficult for the CPU benchmarks to increase. In other words, a plateau might exist after a certain point.

Multilayer Perceptron (MLP)

The MLP model result yielded by the small dataset did not show any outstanding difference from that of the large dataset.

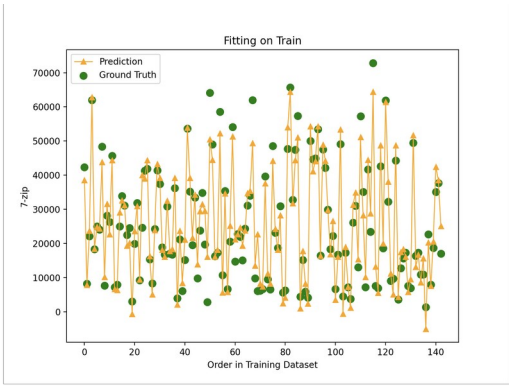


Figure 35: Matplotlib visualization of MLP model with training dataset as input (7-zip)

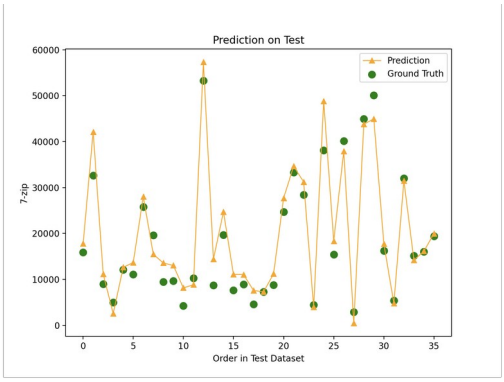


Figure 36: Matplotlib visualization of MLP model with training dataset as input (7-zip)

Table 11: Average MSE and R-Squared values of 10 trials of MLP using cross validation (7-zip)

	Train MSE	Test MSE	R-Squared
Avg:	23957602.38	25924858.22	0.882186315

Decision Tree

The graphs in Figures 37 and 38 as well as the data in Table 12 showed that the Decision Tree model was a better predictor for the 7-zip dataset than the MLP model. This conclusion was the same as the one studied in the large dataset.

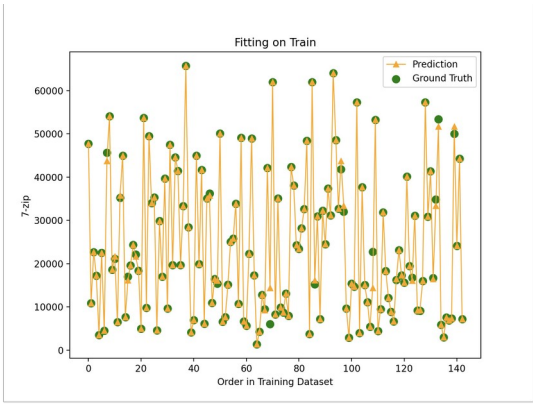


Figure 37: Matplotlib visualization of Decision Tree model with training dataset as input (7-zip)

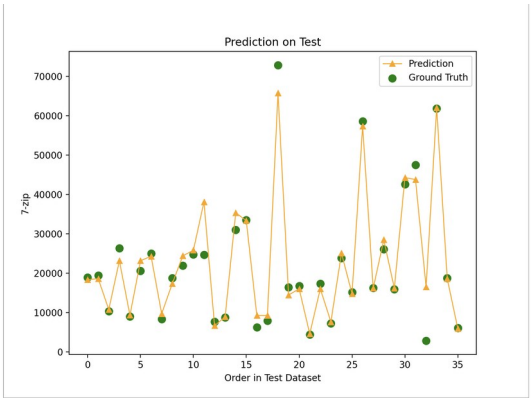


Figure 38: Matplotlib visualization of Decision Tree model with testing dataset as input (7-zip)

Table 12: Average MSE and R-Squared values of 10 trials of Decision Tree using cross validation (7-zip)

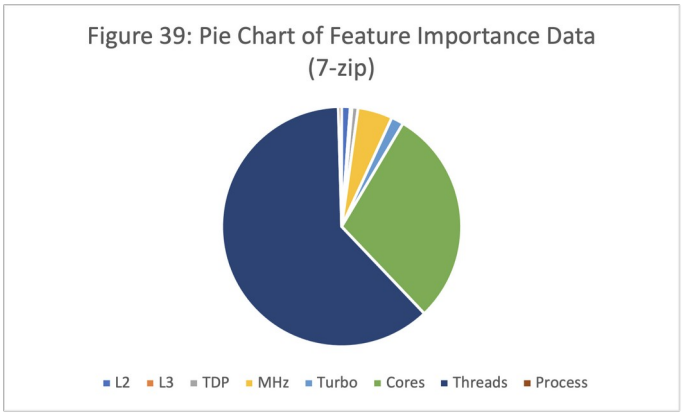
	Train MSE	**Test MSE**	**R-Squared**
Avg:	1645171.489	17959146.71	0.930300943

Table 13: Average Feature Importance of Each CPU Feature in Decision Tree Model (7-zip)

	L2	L3	TDP	MHz	Turbo	Cores	Threads	Process
Avg:	0.0115	0.0025	0.0081	0.0468	0.0167	0.2934	0.6166	0.0045

From the Feature Importance data above, it can be seen that Thread Count continued to be the most important factor in Decision Tree model prediction with the highest Feature Importance value of 0.6166. However, different from the result yielded by the large dataset, Core Count became the second-most important feature, while the Feature Importance value for L3 Cache was significantly lower. This is because, for CPUs small in TDP values, the number of cores is often (74.86% of the time in the small dataset) equal to half of the number of threads in the small dataset. Therefore, when a large percentage of the Core Count is in the same

proportion as the Thread Count, the two feature data become related. This means that using either one of the features, one can predict the value of the other. Therefore, the Decision Tree model sees the same relationship between the two input variables and thus concludes that using either feature to predict the variable is valid, resulting in a significant increase in the Feature Importance value of the Core Count.



Blender

The small dataset for Blender contains 171 datapoints. *Linear Regression*

Figures 40-47**:** graphs generated by matplotlib that show the linear regression result of L2 Cache, L3 Cache, TDP, Clock Speed, Turbo Speed, Core Count, Thread Count, and Process size, respectively (Blender).

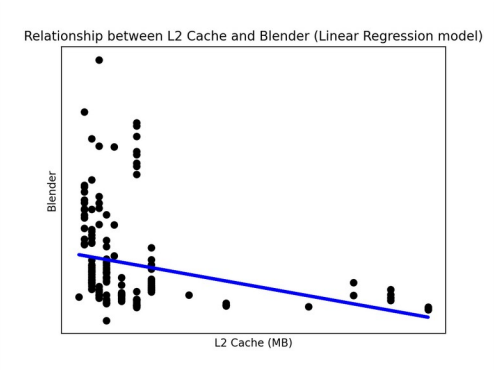


Figure 40

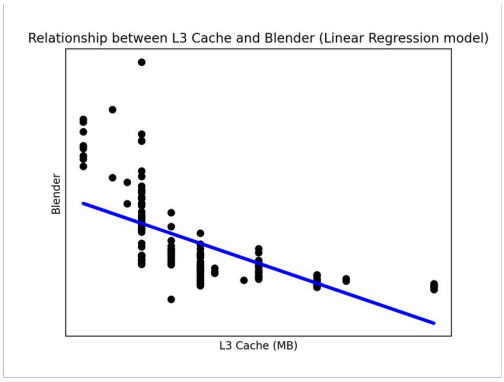


Figure 41

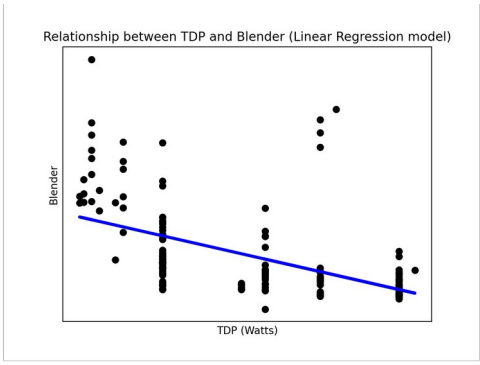


Figure 42

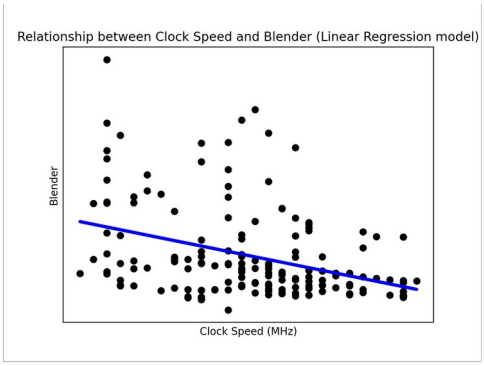


Figure 43

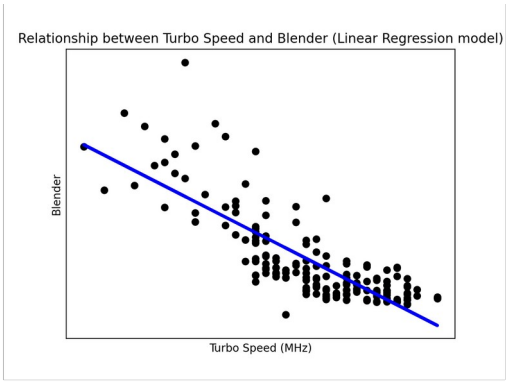


Figure 44

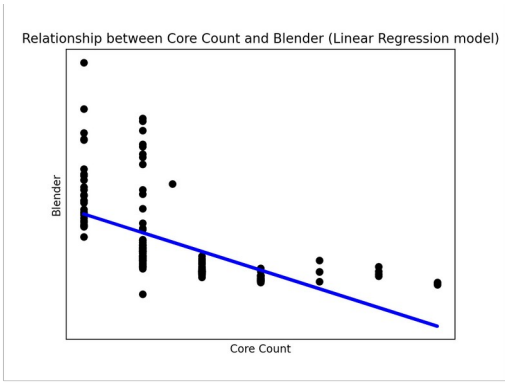


Figure 45

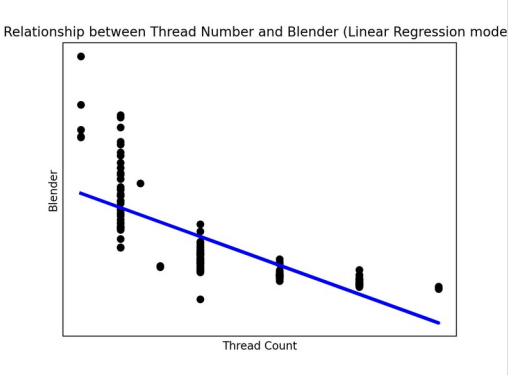


Figure 46

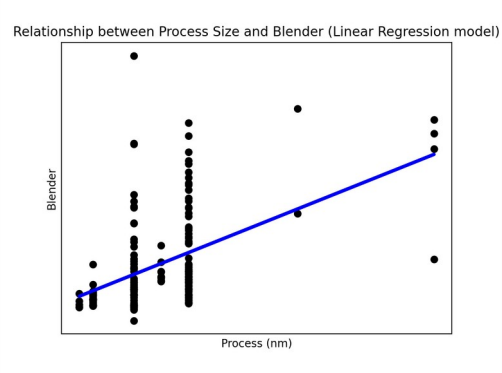


Figure 47

Table 14: R-Square and Coefficient for Linear Regression Models (Blender)

	L2	L3	TDP	Clock	Turbo	Cores	Thread	Process
R-Squared	0.081	0.451	0.291	0.139	0.648	0.360	0.579	0.233
Coefficient	-43.620	-89.285	-30.481	-0.460	-0.869	-171.287	-125.899	87.477

Figures 40-47 and Table 14 all show a similar data trend to the linear regression data in the large dataset. The L3 Cache, Core Count, and Thread Count's R-Squared values increased, but not significantly.

Multilayer Perceptron (MLP)

The MLP model was not as good a predictor for the small dataset as for the large dataset. The R-Squared value of the small dataset was significantly lower than the R-Squared value of the large dataset. From Figures 48 and 49 it is evident that the prediction points do not fit the ground truth points, especially for the ground truth values that are high in Blender values. MLP was not a good predictor for the small dataset.

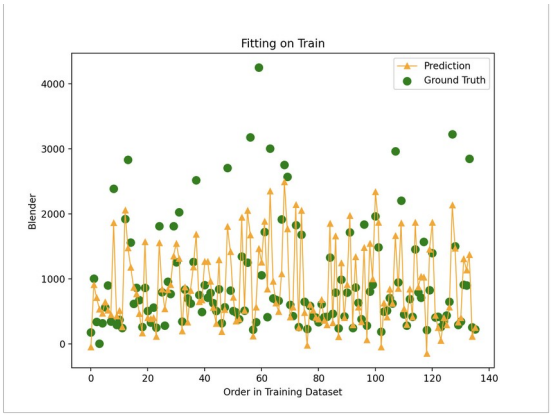


Figure 48: Matplotlib visualization of MLP model with training dataset as input (Blender)

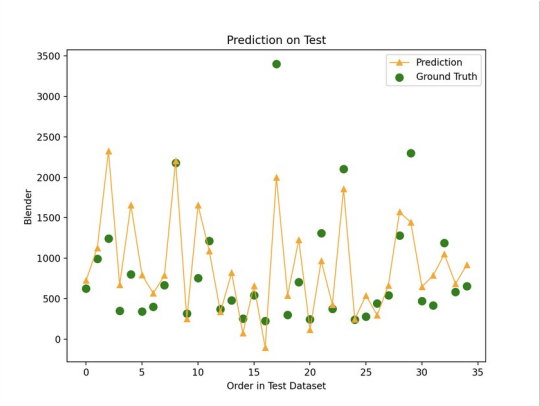


Figure 49: Matplotlib visualization of MLP model with testing dataset as input (Blender)

Table 15: Average MSE and R-Squared values of 10 trials of MLP using cross validation (Blender)

	Train MSE	Test MSE	R-Squared
Avg:	216618.9065	259642.4062	0.547426422

Decision Tree

Compared with the result of the large dataset, the average training MSE was not significantly different. The testing MSE was slightly higher and the R-Squared value was slightly lower. This difference is not significant however, it

could occur because the training dataset was smaller. The testing MSE of the small dataset was higher than the training MSE, which was the same as the large dataset.

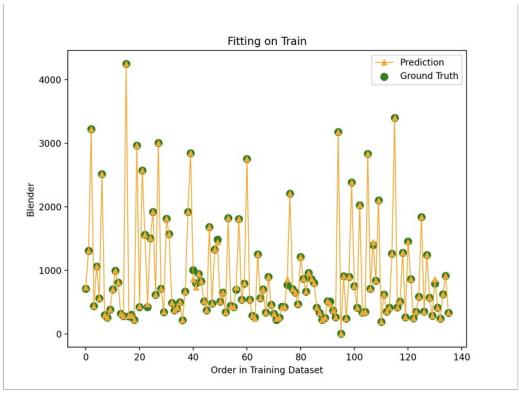


Figure 50: Matplotlib visualization of Decision Tree model with training dataset as input (Blender)

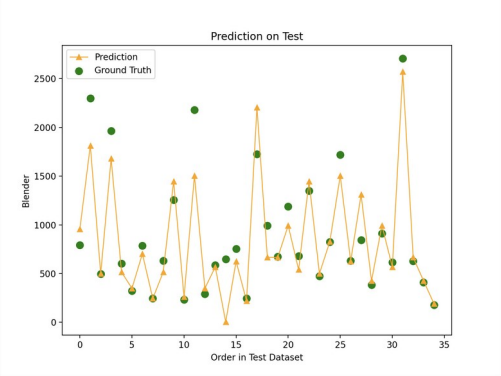


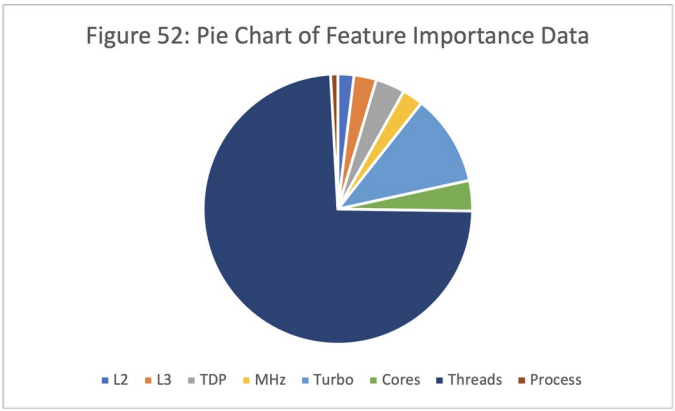
Figure 51: Matplotlib visualization of Decision Tree model with training dataset as input (Blender)

Table 16: Average MSE and R-Squared values of 10 trials of Decision Tree using cross validation (Blender)

	Train MSE	**Test MSE**	**R-Squared**
Avg:	311.0171383	142873.6389	0.747926144

Table 17: Average Feature Importance of Each CPU Feature in Decision Tree Model (Blender)

	L2	L3	TDP	MHz	Turbo	Cores	Threads	Process
Avg:	0.0193	0.0270	0.0353	0.0249	0.1095	0.0364	0.7390	0.0087



The Feature Importance data of the small dataset was significantly similar to that of the large dataset with Thread continuing to be the most important factor and Turbo as the second-most important feature.

Conclusion

In conclusion, this research successfully identified ML models that could identify the relationship between different CPU features and CPU benchmarks. In general, the Decision Tree model was found to be a more fitted model for this purpose than the linear regression and the MLP models. For all CPUs, Thread Count and L3 Cache size had the most effect on improving 7-zip value, while the Thread Count and Turbo Speed were the most influential factors in the Blender benchmark. For CPUs with a TDP value less than or equal to 50 Watts (models more suited for Edge Computing), Thread Count and Core Count were the most related features for 7-zip rating, and Thread Count and Turbo Speed were the most influential for the Blender benchmark. Generally speaking, Thread Count would be the most essential CPU feature to improve when engineers want to enhance the 7-zip or the Blender benchmarks. Researchers can also use the Decision Tree model to predict the benchmark values for a specific combination of CPU features. The prediction can help researchers prioritize future CPU designs to a specific set of CPU features that yields the best prediction result.

Acknowledgment

I would like to thank my instructor, Professor Memik, for guiding me through this paper and providing insightful resources and ideas. Our discussions were essential to the forming of this manuscript. She introduced me to Scikit-Learn and her insightful comments helped me greatly in overcoming any challenges encountered in this paper. I would also like to thank the development team at the Scikit-Learn Library. The Library made the process of building Machine Learning models easy and possible.

References

1. Moore, G.E. “Cramming More Components onto Integrated Circuits.” Proceedings of the IEEE, vol. 86, no. 1, 1998, pp. 82–85., https://doi.org/10.1109/jproc.1998.658762.

2. “Power 4.” IBM100 - Power 4 : The First Multi-Core, 1GHz Processor, IBM, https://www.ibm.com/ibm/history/ibm100/us/en/icons/power4/.

3. “How to read and understand CPU Benchmarks.” *Intel*, Intel, https://www.intel.com/content/www/us/en/gaming/resources/read-cpu-benchmarks.html.

4. “What Is Edge Computing and Why Is It Important?” *Microsoft Azure*, Microsoft Azure, https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-edge-computing/?cdn=disable.

5. Baldini, Ioana, et al. “Predicting GPU Performance from CPU Runs Using Machine Learning.” 2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing, 2014, https://doi.org/10.1109/sbac-pad.2014.30.

6. Zhang, Kaicheng, et al. “Machine Learning-Based Temperature Prediction for Runtime Thermal Management across System Components.” *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 2, 2018, pp. 405–419., https://doi.org/10.1109/tpds.2017.2732951.

7. Luo, Yingyi, et al. “Thermal Management for FPGA Nodes in HPC Systems.” *ACM Transactions on Design Automation of Electronic Systems*, vol. 26, no. 2, Oct. 2020, pp. 1–17., https://doi.org/10.1145/3423494.

8. “CPU Specs Explained - A Comprehensive Guide.” *ThunderboltLaptop*, Thunderbolt Laptop, 21 Jan. 2022, https://thunderboltlaptop.com/cpu-specs-explained/.

9. Harding, Scharon. “What Is a CPU Core? A Basic Definition.” *Tom's Hardware*, Tom's Hardware, 17 June 2022, https://www.tomshardware.com/news/cpu-core-definition,37658.html.

10. Trishanski, Stole. “What Is CPU Cache? - The Hero of Speed.” *XBitLabs*, 27 Aug. 2021, https://www.xbitlabs.com/what-is-cpu-cache/#:~:text=A%20CPU%20cache%20is%20made%20from%20static%20random-access,code.%20A%20good%20example%20is%20your%20favorite%20browser.

11. “CPU Speed: What Is CPU Clock Speed?” *Intel*, https://www.intel.com/content/www/us/en/gaming/resources/cpu-clock-speed.html.

12. Paul, Ian. “What Is TDP for CPUs and GPUs?” *How*, How-To Geek, 7 Sept. 2019, https://www.howtogeek.com/438898/what-is-tdp-for-cpus-and-gpus/.

13. Harding, Scharon. “What Is a CPU Thread? A Basic Definition.” *Tom's Hardware*, Tom's Hardware, 23 Aug. 2018, https://www.tomshardware.com/reviews/cpu-computing-thread-definition,5765.html.

14. Fox, Alexander. “What Is a Processor's Process Size and Why Does It Matter?” Make Tech Easier, 18 Feb. 2022, https://www.maketecheasier.com/processors-process-size/.

15. “B (Benchmark) Command.” *b (Benchmark) Command - 7-Zip Documentation*, https://documentation.help/7-Zip/bench.htm.

16. Moran, Melissa. “What Is Linear Regression?” *Statistics Solutions*, 10 Aug. 2021, https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/what-is-linear-regression/.

17. “Scikit-Learn | Machine Learning in Python.” *Scikit-Learn*, Scikit-Learn, https://scikit-learn.org/stable.

18. Abirami, S., and P. Chitra. “The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases.” *Advances in Computers*, vol. 117, no. 1, 2020, pp. 339–368., https://doi.org/10.1016/s0065-2458(20)x0003-9.

19. “What Is a Multilayer Perceptron (MLP)? - Definition from Techopedia.” *Techopedia.com*, Techopedia, 30 Mar. 2017, https://www.techopedia.com/definition/20879/multilayer-perceptron-mlp.

20. “What Is a Decision Tree.” *IBM Topics*, IBM, https://www.ibm.com/topics/decision-trees.

21. https://github.com/Yihan-Ling/CPU-Features-and-Benchmarks-using-Machine-Learning

22. *NumPy*, https://numpy.org/.

23. “Pandas.” *Pandas*, https://pandas.pydata.org/.

24. “Visualization with Python.” *Matplotlib*, https://matplotlib.org/.