Predicting CPU and GPU Features For Efficient and Sustainable ML Engineering

Kevin Li (kjl2185)

12/15/2022

Abstract

Microchips, such as CPUs and GPUs, are an essential component of modern computing systems. Their performance and efficiency can have a significant impact on the overall performance and energy consumption of a system. Accurate predictions of microchip features can help researchers, developers, and engineers design more efficient and sustainable technology. In this paper, we aim to predict the performance and power consumption features of microchips using Machine Learning techniques and evaluate the performance of different algorithms for this task. Our results show that there are models that can make accurate predictions of microchip type, power consumption, and performance. We use a dataset (1) from Kaggle that consists of 4853 samples and 14 features that relate to CPU and GPU power consumption, performance, and other details. We focus on predicting the chip type, chip power consumption, GPU chip performance, and CPU chip performance. For each of these case studies, we used 6-8 different classification and regression algorithms, including Decision Trees, Random Forests, Support Vector Machines, Neural Networks, and many more. Our results show that for each case study, there are models that can make predictions with high accuracy and precision on the microchips' type, power consumption, and performance. We evaluated the performance of the models using metrics such as accuracy, precision, recall, and F1 score for classification, and mean absolute error and R^2 score for regression. We compared the performance of the different algorithms in each case study and discussed the strengths and weaknesses of each algorithm.

1 Introduction

Microprocessors like CPUs and GPUs are pivotal to our modern computing systems. They enable fast and efficient processing of large amounts of data, which is crucial for many applications, such as machine learning, computer vision, and natural language processing. The performance and power consumption of microprocessors can have a significant impact on that of a system, and accurately predicting their features can help make improvements in efficiency.

There are several opportunities and challenges associated with predicting the features of microchips. For opportunities, the results of this research can have a significant impact on the ML engineering community and other users of microprocessors. By using our models, they can make informed and data-driven decisions on the selection of microprocessors for their tasks and applications, and they can optimize their performance and energy consumption based on the features and characteristics of the microprocessors. This can help improve the efficiency and sustainability of the devices and systems, and it can reduce their cost and complexity.

One of the challenges is that the performance and efficiency of microchips can be influenced by a wide range of outside factors, such as dependencies with the motherboard and RAM, the manufacturing process (the quality of materials and the precision and accuracy of manufacturing equipment), and the operating conditions (supply voltage and temperature). To better focus on the features of our microchips themselves and not on outside influences, we will assume that there are no limitations and dependencies from outside factors.

Another challenge is the availability and quality of data. In order to train and evaluate prediction models, it is necessary to have access to a large and diverse dataset of microchip features, which can be difficult to obtain. Furthermore, the data may be noisy, incomplete, or inconsistent, which can make it difficult to build accurate and reliable models. If we look at the dataset (1) that we used for this research, it has 4853 samples and 14 features. Though it's already considered a small dataset with low dimensionality, this was the largest available dataset found that portrayed microchips with features that could indicate performance and power consumption. Nevertheless, some features were also solely existent in microchips with the 'GPU' type. This resulted in us dividing the dataset into two to perform further analysis and predictions on CPU and GPU performance separately.

In this paper, we present a study of predicting the performance and power consumption features of microchips using Machine Learning techniques and evaluate the performance of different algorithms for this task. We focus on four case studies: predicting the chip type (CPU or GPU), the power consumption (TDP), the performance of GPUs (FP32 GFLOPS), and the performance of CPUs (Freq and Transistors). For each case study, we use a different set of features as input to the prediction models.

To predict the features of microchips, we use several different classification and regression algorithms. For the first case study, where the target variable is the chip type (CPU or GPU), we use binary classification algorithms, such as KNN, decision trees, random forests, logistic regression, and SVM. For the second and third case studies, where the target variables are the power consumption (TDP) and the performance of GPUs (FP32 GFLOPS), we use regression algorithms, such as linear regression, SVR, and neural networks. For the fourth case study, where the target variables are the performance of CPUs (Freq and Transistors), we use multivariate regression algorithms, such as ridge regression, lasso regression, and elastic net.

To evaluate the performance of the prediction models, we use several metrics, such as accuracy, precision, recall, and F1 score for classification algorithms, and mean absolute error and R^2 score for regression algorithms. We compare the performance of the different algorithms in each case study and discuss the strengths and weaknesses of each algorithm.

The rest of this paper is organized as follows. In the next section, we describe the dataset and the features we used for our predictions. In the following sections, we present the classification/regression algorithms we used for each case study. Afterward, we discuss the results of our experiments and compare the performance of different algorithms. Finally, we conclude the paper with a summary of our findings and suggestions for future work.

2 Dataset

As we've mentioned before, the dataset (1) contains 4853 samples, each representing a different microchip, such as a CPU or GPU. The samples have 14 features, which are characteristics or attributes of the microchips. The features are:

- Unnamed: A feature that seemed to index the microchips.
- Product: The name of the microchip.
- Type: The type of microchip, either CPU or GPU.
- Release Date: The date when the microchip was released.
- Process Size (nm): The process size of the microchip, measured in nanometers.
- TDP (W): The thermal design power of the microchip, measured in watts.
- Die Size (mm$\hat{2}$): The size of the die on the microchip, measured in square millimeters.
- Transistors (million): The number of transistors on the microchip, measured in millions.

- Freq (MHz): The clock frequency of the microchip, measured in megahertz.
- Foundry: The company that manufactured the microchip.
- Vendor: The company that designed the microchip.
- FP16 GFLOPS: The number of floating-point operations per second that the microchip can perform using 16-bit floating-point numbers.
- FP32 GFLOPS: The number of floating-point operations per second that the microchip can perform using 32-bit floating-point numbers.
- FP64 GFLOPS: The number of floating-point operations per second that the microchip can perform using 64-bit floating-point numbers.

These features can provide insight into the performance, efficiency, and other characteristics of the microchips. For example, the Type feature can be used to classify microchips as CPU or GPU and the TDP (Thermal Design Power) feature can provide information about the power consumption of the microchip. The FP32 GFLOPS feature can convey the performance of GPU microchips since it is the most commonly used precision and is considered to be a good general-purpose metric for most applications. The frequency and transistors features of CPU microchips capture some main influences of CPU performance. The other features provide additional information about the microchip's performance, manufacturing, and design.

To prepare the dataset for our analysis, we performed several data preprocessing steps, including cleaning and filtering the data to remove missing or invalid values, transforming the data to a suitable format, and scaling the data to have a consistent range and distribution. Further in later case studies, we also split the dataset into two subsets, one for CPU microchips and one for GPU microchips, in order to make separate predictions of performance for each type of microchip.

We first dropped the 'Unnamed' and 'Product' features since these features provided no value for our analysis. In other words, the name of the product has no influence. Furthermore, we dropped samples with duplicated values. After doing so, we had a dataset with 4185 samples and 12 features. At this point, we had to deal with missing data. For Process Size, we had 9 samples missing this feature. TDP had 626 samples, Die size had 715 samples, and Transistors had 711 samples. To reduce the loss of data and improve the performance of our predictive models, we used the simple and convenient method of mean imputation to handle the missing values for these features. Since we do not use GFLOPS until later case studies, we chose to ignore the samples with missing values in those features for now. Next, we use label encoding to convert our Type feature's categorical data into numerical values for convenience. We also use ordinal encoding to 'Foundry' and 'Vendor' features to preserve the inherent order of the categories while providing a more compact representation of the data. We can now analyze our dataset to get some familiarity.

For more information about the dataset, please view the dataset figures shown in Appendix A.

3 Case Studies

3.1 Predicting the Microprocessor Type

In this case study of our research, we sought to predict the type of microchip (CPU or GPU) based on a set of features. Accurately predicting the type of microchip is important for machine learning engineers, as it can inform decisions about hardware selection and optimization for specific tasks. For example, CPUs are generally better suited for tasks that require high levels of sequential processing, such as running operating systems and handling general-purpose computing tasks. In contrast, GPUs are designed for parallel processing and are often used for

tasks that require high levels of computational power, such as training deep learning models. By accurately predicting the type of microchip, machine learning engineers can select the most appropriate hardware for their specific needs and optimize their systems for maximum performance and efficiency.

The objectives of this case study are to (1) develop machine learning models that can accurately predict the type of microchip based on a set of features, and (2) compare the performance of different machine learning approaches on this prediction task. To achieve these objectives, we used a variety of binary classification algorithms and evaluated their performance using a range of evaluation metrics.

3.1.1 Input and Target Variables

The input variables are the features 'Vendor', 'Foundry', 'Process Size (nm)', 'TDP (W)', 'Die Size (mm$\hat{2}$)', 'Transistors (million)', 'Freq (MHz)' in the dataset, which provides information about the microchip's manufacturer, manufacturing process, size, power consumption, size of the integrated circuit, number of transistors, and clock frequency. We did not include 'FP16 GFLOPS', 'FP32 GFLOPS', and 'FP64 GFLOPS' because they were only existent in GPU's. These variables were used as input to the machine learning models in order to predict the type of microchip (CPU or GPU), which was our target variable ['Type'] in this case study. Since we've already completed most of the data preprocessing beforehand, the step here was to just perform a test-train split, with a test size of 0.2 and a random state of 42, to split the data into training and testing sets.

3.1.2 Methodology and Hyperparameter Tuning

We used a variety of machine learning approaches: Dummy Classifiers, K-Nearest Neighbors (KNN), Decision Trees, Random Forests, Logistic Regression, Support Vector Machines (SVM), and Neural Networks.

We used a Dummy Classifier as a simple baseline approach that can be useful for evaluating the performance of more advanced machine learning methods. The main advantage of using a Dummy Classifier is that it is very easy to implement and requires little to no data preparation. However, its predictions were based on the "most frequent" strategy and doesn't take into account the characteristics of the data, so it is unlikely to perform well on real-world prediction tasks. We did not perform any hyperparameter tuning for the Dummy Classifier.

KNN is a non-parametric classification method that can be effective for datasets with small numbers of observations and features. One advantage of using KNN is that it is simple to implement and does not require extensive data preparation or hyperparameter tuning. However, KNN can be computationally expensive for large datasets and may not be as accurate as other methods. We did some light hyperparameter tuning by iterating through 1 to 11 "n_neighbors" to compare accuracies.

Decision Trees are a popular machine learning approach that are easy to interpret and visualize. They are also relatively simple to implement and do not require extensive data preparation. However, decision trees can be prone to overfitting and may not generalize well to new data. To tune the hyperparameters of the decision tree model, we used grid search cross-validation (GridSearchCV) to search over a defined range of values for hyperparameters such as "max_depth", "min_samples_split", and "min_samples_leaf".

Random Forests are an ensemble method that combine multiple decision trees to make more accurate predictions. One advantage of using Random Forests is that they are relatively robust to overfitting and tend to perform well on a wide range of prediction tasks. However, they can be more computationally expensive to train and may require more data to perform well. To tune the hyperparameters of the random forest model, we used grid search cross-

validation (GridSearchCV) to search over a defined range of values for hyperparameters such as "n_estimators", "max_features", and "bootstrap".

Logistic Regression is a linear classification method that is widely used in practice due to its simplicity and interpretability. Logistic regression is also relatively efficient to train and does not require extensive data preparation. However, it is limited to linear decision boundaries and may not be as accurate as other methods for non-linear datasets. To tune the hyperparameters of the logistic regression model, we used grid search cross-validation (GridSearchCV) to search over a defined range of values for hyperparameters such as "penalty" and "C".

SVM is a classification method that can be effective for datasets with complex decision boundaries. One advantage of using SVM is that it is relatively robust to overfitting and can handle high-dimensional datasets. However, SVM can be sensitive to the choice of hyperparameters and may require more data to perform well. To tune the hyperparameters of the SVM model, we used random search cross-validation (RandomizedSearchCV) to search over a defined range of values for hyperparameters such as "kernel", "C", and "gamma".

Neural Networks are a powerful machine learning approach that can learn complex relationships in data. They are particularly well-suited for tasks that require high levels of computational power, such as image classification and natural language processing. However, neural networks can be difficult to interpret and may require more data and computational resources to train. To tune the hyperparameters of the neural network model, we used random search cross-validation (RandomizedSearchCV) to search over a defined range of values for hyperparameters such as "activation", "alpha", "learning_rate" and "max_iter".

For each of the machine learning approaches that we used, we trained the model using the training data and evaluated its performance on the test data using a range of evaluation metrics. This allowed us to compare the performance of the different approaches and identify the most effective method for predicting the type of microchip.

3.1.3 Results

We evaluated the machine learning approaches using a range of evaluation metrics, including accuracy, cross validation mean score, precision, recall, and f1-score. We display the results in Appendix B.

Overall, the performance of the different approaches was quite strong. The K-Nearest Neighbors classifier, Decision Tree classifier, and Random Forest classifier achieved the highest scores, with accuracy scores above 99.6% and f1-scores above 98% (Decision Tree and Random Forest had f1-scores as high as 99.6%).

One notable trend that we observed was that the more complex models (e.g., neural networks) did not necessarily outperform the simpler models (e.g., K-Nearest Neighbors). In fact, the K-Nearest Neighbors classifier outperformed the Neural Network classifier in terms of accuracy, precision, recall, and f1-score. This suggests that, in this case, a simpler model may be more suitable for predicting the type of microchip.

The baseline accuracy for the Dummy Classifier was relatively low, at 50.2%. This suggests that there is a strong signal in the data that allows for accurate predictions to be made, and that the machine learning models were able to effectively capture this signal.

Overall, our results suggest that it is possible to accurately predict the type of microchip (CPU or GPU) using machine learning techniques. The K-Nearest Neighbors classifier, Decision Tree classifier, and Random Forest Classifier appear to be particularly suitable for this task, achieving high levels of accuracy and f1-score.

3.2 Predicting the Microprocessor Power Consumption

In this case study, we aimed to predict the power consumption of microchips, specifically the TDP (thermal design power), using a set of features. Accurately predicting the TDP of a microchip is important for machine learning engineers, as it can inform decisions about power management, hardware selection, and optimization for specific tasks. For example, microchips with lower power consumption may be more suitable for use in portable devices or in environments with limited power resources, while microchips with higher power consumption may be more suitable for tasks that require high levels of computational power. High TDP values can result in increased energy consumption and heat generation, which can lead to degraded performance or even hardware failure. By accurately predicting the TDP of a microchip, machine learning engineers can design efficient and sustainable systems that can operate at optimal performance.

The objectives of this case study are to (1) develop machine learning models that can accurately predict the TDP of microchips based on a set of features, and (2) compare the performance of different machine learning approaches on this prediction task. To achieve these objectives, we used a variety of regression algorithms and evaluated their performance using a range of evaluation metrics.

3.2.1 Input and Target Variables

The input variables for this case study were 'Type', 'Vendor', 'Foundry', 'Process Size (nm)', 'Die Size (mm$\hat{2}$)', 'Transistors (million)', 'Freq (MHz)' in the dataset. These variables were used as input to the machine learning models in order to predict the power consumption (TDP) of the microchips, which was our target variable in this case study.

Before applying the machine learning models, we performed a standard scaling of the input variables using StandardScaler to ensure that all features were on the same scale. This is important because some algorithms can be sensitive to the scale of the input variables and may not perform well if the variables are not standardized. Standardization can also help to improve the interpretability of the model, as it ensures that the coefficients of the model are not influenced by the scale of the variables.

We then split the data into training and testing sets using a test size of 0.2 and a random state of 42, using the same test-train split as in our first case study.

3.2.2 Methodology and Hyperparameter Tuning

For this case study, we used a variety of regression models to predict the TDP of microchips. These models included the Dummy Regressor, K-Nearest Neighbors Regressor, Decision Tree Regressor, Random Forest Regressor, Linear Regression, Multi-Layer Perceptron Regressor, and Gradient Boosting Regressor.

We used the Dummy Regressor as a simple baseline approach, which makes predictions based on the mean value of the target variable. No hyperparameter tuning was performed for this model.

For the K-Nearest Neighbors Regressor, we used grid search cross-validation to tune the hyperparameters. The hyperparameters included the number of neighbors, weighting method, and algorithm used for computation. We also included the leaf size as a hyperparameter, which determines the number of samples to be included in each leaf of the tree.

The Decision Tree Regressor was tuned using grid search cross-validation, with hyperparameters including the maximum depth of the tree, the minimum number of samples required to split an internal node, and the minimum number of samples required to be at a leaf node. We also included the maximum number of features to consider when looking for the best split as a hyperparameter.

For the Random Forest Regressor, we used grid search cross-validation to tune the hyperparameters, which included the number of trees in the forest, the maximum depth of each tree, the minimum number of samples required to split an internal node, the minimum number of samples required to be at a leaf node, and the maximum number of features to consider when looking for the best split.

We used grid search cross-validation to tune the Linear Regression model, with the hyperparameter of whether to include an intercept term in the model. We also included the hyperparameter of whether to make a copy of the input data before fitting the model.

The Multi-Layer Perceptron Regressor was tuned using random search, with hyperparameters including the sizes of the hidden layers, the solver algorithm used, the activation function, the regularization strength, and the maximum number of iterations.

For the Gradient Boosting Regressor, we used grid search cross-validation to tune the hyperparameters, including the number of trees in the model, the learning rate, and the maximum depth of each tree.

3.2.3 Results

To evaluate the performance of the models, we used several evaluation metrics, including mean absolute error (MAE), mean cross-validation score, and the R^2 score. The MAE measures the average absolute difference between the predicted values and the actual values, with lower values indicating better performance. The mean cross-validation score is the average of the model's performance on the training set, calculated using cross-validation. The R^2 score is a measure of the goodness of fit of the model, with higher values indicating better performance.

Overall, we observed a range of performance among the different models. You can view the results in Appendix C. In summary, the K-Nearest Neighbors Regressor, Decision Tree Regressor, and Random Forest Regressor performed the best on this prediction task, with relatively low MAEs and high R^2 scores. The Dummy Regressor and the Linear Regression model had poorer performance, while the Multi-Layer Perceptron Regressor and the Gradient Boosting Regressor had intermediate performance. These results suggest that more advanced machine learning approaches, such as decision tree-based methods and random forests, may be more effective for predicting the TDP of microchips.

3.3 Predicting the performance of a GPU

In this case study, our goal was to predict the performance of a GPU, specifically the FP32 GFLOPS (floating point operations per second) using a set of features. Accurately predicting the FP32 GFLOPS of a GPU is important for machine learning engineers, as it can inform decisions about hardware selection and optimization for specific tasks. For example, GPUs with higher FP32 GFLOPS may be more suitable for tasks that require high levels of computational power, while GPUs with lower FP32 GFLOPS may be more suitable for tasks that do not require as much computational power. By accurately predicting the FP32 GFLOPS of a GPU, machine learning engineers can design systems that can operate at optimal performance.

To achieve our objectives, we used a variety of regression algorithms and evaluated their performance using a range of evaluation metrics. Before applying the machine learning models, we preprocessed the data by removing 551 samples that had their 'fp32 gflops' feature missing. We also split the dataset into two based on the 'cpu' and 'gpu' type. We then performed a standard scaling of the input variables using StandardScaler and split the data into training and testing sets using a test size of 0.2 and a random state of 42.

We used a variety of machine learning algorithms to predict the FP32 GFLOPS of the GPUs, including the Dummy Regressor, K-Nearest Neighbors Regressor, Decision Tree Regressor,

Random Forest Regressor, Linear Regression, Support Vector Regressor, and Gradient Boosting Regressor. The results are in Appendix D.

The Dummy Regressor, Linear Regression, and Support Vector Regressor all had relatively high MAE scores and lower R2 scores compared to the other models. The K-Nearest Neighbors Regressor, Decision Tree Regressor, Random Forest Regressor, Multi-Layer Perceptron Regressor, and Gradient Boosting Regressor all had lower MAE scores and higher R2 scores, indicating better performance on this prediction task. Among these models, the Gradient Boosting Regressor had the lowest MAE score and the highest R2 score, making it the best performer on this task.

3.4 Predicting The Performance Of CPU's Using Frequency and Transistors

In this case study, we aimed to predict the performance of CPUs using the frequency and number of transistors as input features. Accurately predicting the performance of a CPU is important for machine learning engineers, as it can inform decisions about hardware selection and optimization for specific tasks. High-performing CPUs may be more suitable for tasks that require high levels of computational power, while CPUs with lower performance may be more suitable for tasks with lower computational demands.

As we did similarly before, we set the target variable to be "Freq (MHz)" and "Transistors (million)" and split the data into training and testing sets using a test size of 0.2 and a random state of 42. To ensure that all features were on the same scale, we also performed standard scaling on the input variables using StandardScaler.

One notable trend that can be observed from the table is that the decision tree and random forest models had the best performance, followed by the neural network. This suggests that these models may be more effective at predicting the performance of a CPU than the other models considered.

Another trend that can be observed is that the linear regression, ridge regression, lasso, and elastic net models all had similar performance, with MAE scores that were relatively high compared to the decision tree and random forest models. This suggests that these models may not be as effective at predicting the performance of a CPU.

4 Conclusion

In this paper, we aimed to predict the performance and power consumption features of microchips using machine learning techniques and evaluate the performance of different algorithms for this task. We used a dataset from Kaggle that consisted of 4853 samples and 14 features related to CPU and GPU power consumption, performance, and other details. We focused on predicting the chip type, chip power consumption, GPU chip performance, and CPU chip performance.

For each of these case studies, we used 6-8 different classification and regression algorithms, including Decision Trees, Random Forests, Support Vector Machines, Neural Networks, and others. Our results showed that there are models that can make accurate predictions of microchip type, power consumption, and performance. We evaluated the performance of the models using metrics such as accuracy, precision, recall, and F1 score for classification, and mean absolute error and R^2 score for regression.

Overall, our main findings suggest that decision tree and random forest models may be the most effective at predicting the performance and power consumption of microchips. In particular, we found that these models were among the top performers in all case studies.

These findings have implications for the field of machine learning engineering, as they suggest that decision tree and random forest models may be useful for predicting the performance

and power consumption of microchips, which can help inform and optimize the design and use of efficient and sustainable computing systems. The results also highlight the potential for neural network models to be effective at predicting GPU chip performance.

There are several limitations to our study that should be considered. One limitation is that we assumed that there were no limitations and dependencies from outside factors on the performance and efficiency of the microchips. In reality, the performance and efficiency of microchips can be influenced by a wide range of outside factors, such as dependencies with the motherboard and RAM, the manufacturing process, and the operating conditions. It would be interesting to explore the impact of these factors on the performance and efficiency of microchips in future research.

Another limitation is the availability and quality of data. In this study, we used a dataset from Kaggle that consisted of 4853 samples and 14 features. While this dataset was sufficient for our purposes, it is relatively small and may not be representative of the full range of microchip features and characteristics. In future research, it would be useful to have access to larger and more diverse datasets to further improve the accuracy and reliability of the prediction models.

Overall, our study suggests that decision tree and random forest models may be effective at predicting the performance and power consumption of microchips, which can have implications for the design and use of efficient and sustainable computing systems. We also found that the neural network model may be effective at predicting GPU chip performance. However, there are limitations to our study, and there is potential for further research to explore the impact of outside factors and to access larger and more diverse datasets.

References

[1] chip_dataset.csv. CPU and GPU Performances Dataset. Retrieved from https://www.kaggle.com/datasets/michaelbryantds/cpu-and-gpu-product-data

A Appendix A: Dataset Figures

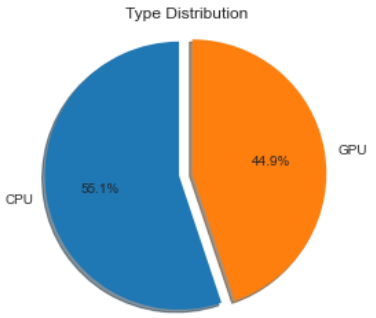

Figure 1: The Type Distribution of Microchips

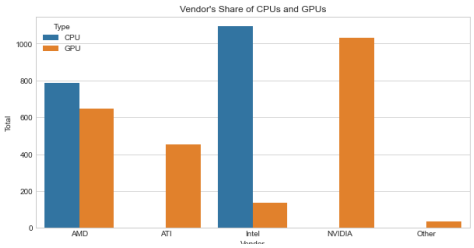

Figure 2: Vendor's Share of Microchips

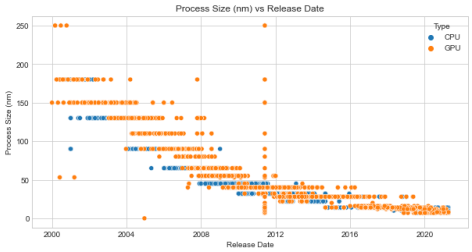

Figure 3: Process Size vs. Time

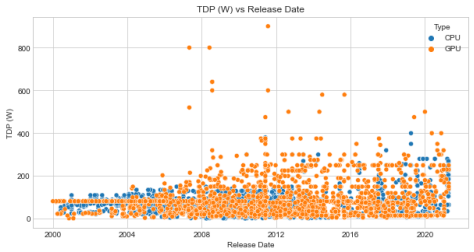

Figure 4: TDP vs. Time

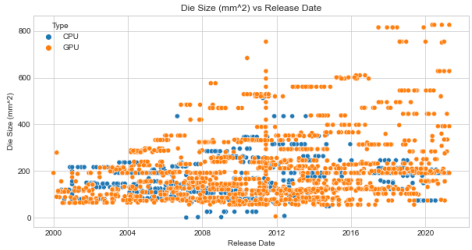

Figure 5: Die Size vs. Time

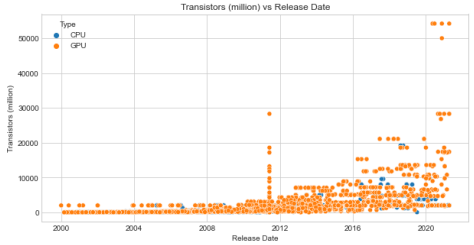

Figure 6: Transistors vs. Time

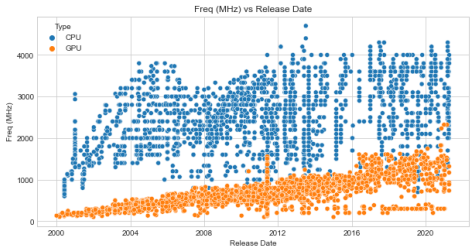

Figure 7: Frequency vs. Time

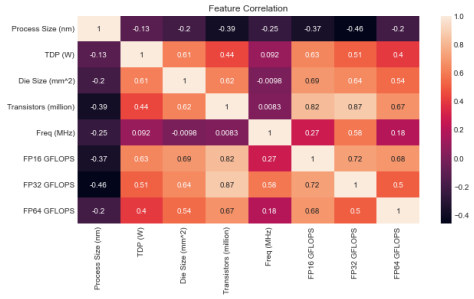

Figure 8: Feature Correlation

B Appendix B: Predicting the Microprocessor Type Results

Model	Accuracy	Cross-Validation Score	Precision	Recall	F1 Score
Dummy Classifier	0.501	-	-	-	-
kNN	0.998	0.991	0.989	0.989	0.989
Decision Tree	0.996	0.993	0.996	0.996	0.996
Random Forest	0.996	0.996	0.996	0.996	0.996
Logistic Regression	0.980	0.979	0.980	0.980	0.980
SVM	0.983	0.982	0.983	0.983	0.983
Neural Network	0.971	0.972	0.972	0.971	0.971

Table 1: Results of Predicting the Microprocessor Type

C Appendix C: Predicting the Microprocessor Power Consumption

Model	MAE	Cross-Validation Score	R^2 Score
Dummy Classifier	43.792	-	-0.001
KNN Regressor	20.964	23.641	0.601
Decision Tree Regressor	24.779	25.243	0.580
Random Forest Regressor	23.420	24.106	0.684
Linear Regression	34.349	35.166	0.403
Neural Network	23.489	47.390	0.679
Gradient Boosting Regressor	21.208	22.241	0.660

Table 2: Results of Predicting the Microprocessor Power Consumption

D Appendix D: Predicting the performance of a GPU

Model	MAE	Cross-Validation Score	R^2 Score
Dummy Regressor	2295.857	-	-1.394e-05
KNN Regressor	304.186	282.457	0.954
Decision Tree Regressor	323.574	335.416	0.951
Random Forest Regressor	338.609	304.142	0.943
Linear Regression	814.679	831.839	0.716
SVR	705.689	-	0.806
Neural Network	647.462	2464.304	0.825
Gradient Boosting Regressor	291.236	274.024	0.962

Table 3: Results of Predicting the Performance of a GPU

E Appendix E: Predicting the Performance of a CPU

Model	MAE	Cross-Validation Score	R^2 Score
Dummy Classifier	861.78	-	-0.002
Decision Tree Regressor	368.73	377.58	0.712
Random Forest Regressor	347.71	347.59	0.735
Linear Regression	637.95	628.40	0.370
Ridge Regression	636.67	627.72	0.371
Lasso	635.60	627.67	0.373
Elastic Net	636.03	627.26	0.371
Neural Network	445.46	880.97	0.615

Table 4: Results of Predicting the Performance of a CPU

F Appendix E: Additional Graphs

Due to the limitations of the length of this paper, additional graphs can be found at Kevin Li's Project Github Repository