VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY

Ho Chi Minh City University of Technology

Faculty of Computer Science and Engineering

PROBABILITY AND STATISTICS (MT2013)

Assignment Report - Class CC03 - Group 07

"ANOVA and Regression Analysis of CPU's Thermal Design Power"

Supervisor: PhD. Phan Thi Huong

Students: Le Kieu Anh Vu – 2353340
Do Thanh Tan – 2353073
Le Pham Tien Long – 2352688
Pham Quang Minh – 2352761
Nguyen Huu Minh Khoi - 2352614
Hoang Thuy Tram – 2353202

Ho Chi Minh City, December 2024

Member list – Workload & Calibrated Score

No.	Fullname	Student ID	Workload	Calibrated Score
1	Le Kieu Anh Vu	2353340	Implementing R code for Descriptive Statistics section	+0
2	Do Thanh Tan	2353073	Implementing R code and writing report for Discussion and Extension section	+0
3	Le Pham Tien Long	2352688	Implementing R code and writing report for Inferential Statistics section, Arrange final report	+0
4	Pham Quang Minh	2352761	Writing report for Data Introduction, Knowledge Background and Data preprocessing sections	+0
5	Nguyen Huu Minh Khoi	2352614	Writing report for Descriptive Statistics section	+0
6	Hoang Thuy Tram	2353202	Implementing R code and writing report for Knowledge Background and Data Preporcessing sections	+0

Contents

List of Figures

List of Tables

Abstract

This project presents a comprehensive analysis of Intel CPU specifications, espcially Thermal Design Power (TDP). The data undergoes meticulous preprocessing, which includes handling missing values, transforming variables, and removing outliers. Key variables, such as launch date, lithography, recommended customer price, processor base frequency, and thermal design power are examined for their distribution and statistical properties. Descriptive statistics are followed by correlation analysis, revealing insights into how different specifications relate to each other. Inferential statistical techniques, including one-way ANOVA and multiple linear regression, are applied to assess the impact of lithography on TDP and to predict TDP based on other CPU features. In the end, the report discusses the advantages and disadvantages for two used methods, then suggests Non-parameter test and XGBoost as extensions to the analysis.

1 Data Introduction

The dataset is retrieved from Computer Parts (CPUs and GPUs) by author Ilissek. The dataset contains two CSV files: `all_gpus.csv` for Graphics Processing Units (GPUs), and `intel_cpus.csv` for Central Processing Units (CPUs).
In this project, we mainly focus on data from the `intel_cpus.csv` file. This dataset describes some important attributes of the CPU in the market shown in figure 1, note that *cate* and *cont* stand for categorical and continuous, respectively.

Name of variables	**Data Type**	**Unit**	**Description**
Product Collection	*cate* $-$ {<names of CPUs>}	None	A product collection of Intel CPUs refers to a range of processors organized by series, such as Core, Xeon, Celeron, Pentium, and others.
Vertical Segment	*cate* $-$ {Server, Embedded, Mobile, Desktop}	None	Describes some segments of the market where the CPU is mainly used for.
Status	*cate* $-$ {End of Interactive Support, Announced, End of Life, Launched }	None	Describes the status of the CPU in the market.
Launch Date	*cate* $-$ $\{1999, 2000, 2001, \ldots\}$	None	Describes the launch date of the product.
Lithography	*cate* $-$ $\{14, 22, 32, \ldots\}$	nm	The semiconductor technology used to manufacture an integrated circuit, reported in nanometers.
Recommended Customer Price	*cont* $-$ $\{x \mid 9.62 \leq x \leq 13011\}$	USD ($)	The price recommended to the customer by Intel for retailers selling the CPU.
nb of Cores	*cate* $-$ $\{1, 2, 4, \ldots\}$	None	The number of core is a hardware term that describes the number of independent central processing units (CPU).
nb of Threads	*cate* $-$ $\{1, 2, 4, \ldots\}$	None	The number of threads is the count of independent execution units that can run in parallel within a program or process.
Processor Base Frequency	*cont* $-$ $\{x \mid 300 \leq x \leq 4300\}$	MHz	Describes the rate at which the processor's transistors open and close.
Cache	*cont* $-$ $\{x \mid 0.015625 \leq x \leq 60\}$	MB	A small, high speed memory that stores the frequently accessed data and instructions.

Name of variables	Data Type	Unit	Description
TDP	$cont - \{x \mid 0.65 \leq x \leq 300\}$	W	Short for Thermal Design Power, representing the average power that the processor dissipates when operating at Base Frequency with all cores active.
Max Memory Size	$cont - \{x \mid 1 \leq x \leq 4198.4\}$	GB	The maximum amount of memory that the system can be allocated to use.
Max Memory Bandwidth	$cont - \{x \mid 1.6 \leq x \leq 25.6\}$	GB/s	The maximum rate at which data can be read from or stored into a semiconductor memory by the processor.
Graphics Base Frequency	$cont - \{x \mid 100 \leq x \leq 900\}$	MHz	The rate at which the GPU's cores process tasks under standard, non-boost conditions.
Graphics Max Dynamic Frequency	$cont - \{x \mid 0.5 \leq x \leq 135\}$	GHz	The highest clock speed that a GPU can achieve dynamically under optimal conditions.
Instruction Set	$cate - \{32, 64\}$	bit	A collection of machine-level commands that a CPU can execute.

Table 1: *Summary of variables used in this dataset.*

2 Background

2.1 ANOVA

2.1.1 One-way ANOVA

ANOVA, which stands for "Analysis of Variance", is a statistical method used to compare the differences between the means of more than 2 groups. This method begins by testing the hypothesis that these group means are equal. By analyzing the variation within and between groups, ANOVA helps us determine whether any observed differences are statistically significant or if they could have occurred by chance. This insight is crucial for understanding whether the groups differ meaningfully in terms of their average values.
One-way ANOVA is a technique to compare whether the means of two or more sample are significantly different through F distribution. This analysis of variance technique requires a numeric response variable y and a single explanatory variable X, hence "one-way".
In general, suppose we have k groups (samples) with $n_1, n_2, \ldots, n_k$ observations selected randomly and independently from k populations. Let $\mu_1, \ldots, \mu_k$ represent the population means, and let x_{ij} denote the j-th observation in the i-th group.

Assumption.

- The populations are normally distributed and has equal variances.
- The samples are random and independent.

ANOVA Testing.

From this, the one-way ANOVA hypothesis is:

- *The null hypothesis:* The means among groups are the same.

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_k$$

- *The alternative hypothesis:* At least two group means are significantly different from each other.

$$H_1 : \exists i, j, i \neq j : \mu_i \neq \mu_j$$

At first, we calculate the average value $\overline{x_i}$ for each group by:

$$\overline{x_i} = \frac{1}{n_i} \sum_{j=1}^{n_i} x_{ij}, \quad i = 1, 2, \ldots, k.$$

We calculate the general average value for all group:

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{k} \sum_{j=1}^{n_i} x_{ij} = \frac{1}{n} \sum_{i=1}^{k} n_i \overline{x_i}, \quad n = \sum_{i=1}^{k} n_i$$

Secondly, we calculate the variance inside each group (SS):

$$SS_i = \sum_{j=1}^{n_i} (x_{ij} - \overline{x_i})^2, \quad \text{with } i = 1, 2, \ldots, k.$$

Then we calculate the sum of squared difference between each observation and the mean of group (SSW). This represents the variance due to other factors, not the researched one.

$$SSW = SS_1 + SS_2 + \ldots SS_k$$

SSB represents the variation due to differences between groups, that is, the variation due to the research factor, the factor used to divide the groups:

$$SSB = \sum_{i=1}^{k} n_i (\overline{x_i} - \overline{x})^2$$

We define SST is the total sum of squared deviations (total variation).

$$SST = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \overline{x})^2$$

The variation of the observations from the mean is the sum of the variation explained by the research factor (SSB) and the variation due to other factors (SSW):

$$SST = SSW + SSB$$

Now we consider the approximate of variance within groups (MSW):

$$MSW = \frac{SSW}{n - k}$$

The approximate variance between group (SSB) is:

$$MSB = \frac{SSB}{k - 1}$$

From that the statistic variable F is:

$$F = \frac{MSB}{MSW}$$

With significance level α:

- If $F > F_\alpha(k - 1, n - k)$, we will reject H_0.
- If $F \leq F_\alpha(k - 1, n - k)$, we do not have enough evidence to reject H_0.

The value $F_\alpha(k - 1, n - k)$ is the percentile of Fisher distribution with degrees of freedom $k - 1$ and $n - k$ with significance level α.

Analysis of Variance for a Single-Factor Experiment, Fixed-Effects Mode

Source of Groups	Square Sum	Degrees of Freedom	Mean Square	Test Statistic
Between Groups	SSB	$k-1$	MSB	$\frac{MSB}{MSW}$
Within Groups	SSW	$n-k$	MSW	
Total	SST	$n-1$		

Table 2: *Analysis of Variance for a Single-Factor Experiment*

When we come to the conclusion of one-way ANOVA, there are two possibilities:

- We do not have enough evidence to reject H_0.
- We reject H_0 and accept H_1, indicating that there is a difference in mean values among the groups. However, we do not know which specific group(s) differ from the others. Therefore, we have to have a post-hoc test for the ANOVA.

2.1.2 Tukey HSD Test

For "after ANOVA" testing, we use the **Tukey Honestly Significant Difference** (Tukey HSD). The idea behind Tukey HSD test is to focus on the largest value of the difference between two group means. We state two hypothesis of this test as follows:

- *The null hypothesis:*
$$H_0 : \mu_i = \mu_j, i, j = \overline{1, n}, i \neq j$$
- *The alternative hypothesis:*
$$H_1 : \mu_i \neq \mu_j, i, j = \overline{1, n}, i \neq j$$

The test statistic is:

$$q = \frac{M_i - M_j}{\sqrt{\frac{MSW}{N}}}$$

where:

- $M_i - M_j$ is the difference between the pair of means. To calculate this, M_i should be larger than M_j.
- MSW is the Mean Square Within, and N is the number in the group or treatment.

In order to reject H_0, q should be greater than the critical values for Studentized Range Distribution:

$$q > q_{crit}$$

We find the value q_{crit} inside Studentized Range q Table at significance level α in $k-1$ column and $n-k$ row.

Tukey HSD confidence interval.

The Tukey confidence limits for all pairwise comparisons with confidence coefficient of at least $1-\alpha$ are:

$$\overline{x_i} - \overline{x_j} - \frac{1}{\sqrt{2}} q_{\alpha,k,n-k} MSE \sqrt{\frac{2}{n}} \leq \mu_i - \mu_j \leq \overline{x_i} - \overline{x_j} + \frac{1}{\sqrt{2}} q_{\alpha,k,n-k} MSE \sqrt{\frac{2}{n}}$$

2.2 Multiple Linear Regression (MLR)

Multiple linear regression is a statistical method used to quantify and model the relationship between one dependent variable and two or more independent variables. This approach extends simple linear regression by considering multiple predictors simultaneously, allowing analysts to assess how combined factors influence an outcome of interest. Widely applied in fields such as economics, healthcare, and agriculture, multiple linear regression is valuable for understanding both the strength of relationships and making predictions. For instance, it can be used to analyze how rainfall, temperature, and fertilizer affect crop growth, or to estimate expected crop yield based on specific levels of these variables. In this report, we explore the theoretical foundations, assumptions, and applications of multiple linear regression, providing insights into model fitting and interpretation for real-world data scenarios.

2.2.1 True Model Equation

If we have k predictor (or independent) variables, then a multiple linear regression model takes the form:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} + \varepsilon$$

where:

- y_i: The observed value of the response (or dependent) variable for the i-th observation in the dataset. This is the actual data point collected in the real world, which the model seeks to explain or predict based on the independent variables.
- β_0: The y-intercept, which represent the value of y_i when all the independent variables are equal to zero.
- $\beta_0,\ \beta_1, \ldots,\ \beta_k$: The regression coefficients corresponding to each predictor variable $x_{i1},\ x_{i2}, \ldots,\ x_{ik}$. These represent the change in the predicted value of y_i for a one-unit change in the corresponding independent value, while holding the other predictors constant.
- $x_{i1},\ x_{i2}, \ldots,\ x_{ik}$: The predictor variables for the i-th observation. These are the features or inputs used to predict the response variable.
- ε_i: The error term (or residual) for the i-th observation, which captures the unexplained variation in y_i that cannot be accounted for by the predictors:

  $$\varepsilon_i = y_i - \widehat{y_i}$$

  where: $\widehat{y_i}$ is the predicted value for the i-th observation.

The values for $\beta_0,\ \beta_1, \ldots,\ \beta_k$ are chosen using the Least Square Method, which minimizes the Residual Sum of Squares (RSS):

$$RSS = \sum_{i=1}^{n} (y_i - \widehat{y}_i)^2$$

where: n is the total number of observations (data points) in the dataset.

2.2.2 Fitted Model Equation (or Prediction Equation)

We use this model to make prediction, it based on estimated coefficients $\widehat{\beta_0},\ \widehat{\beta_1}, \ldots,\ \widehat{\beta_k}$ and excludes the error term.

$$\widehat{y_i} = \widehat{\beta_0} + \widehat{\beta_1} x_{i1} + \widehat{\beta_2} x_{i2} + \cdots + \widehat{\beta_k} x_{ik}$$

where: $\widehat{y}$: The predicted value for the i-th observation.

2.2.3 The Fit of a Multiple Linear Regression Model

Several key metrics are commonly used to assess how well a multiple linear regression model fits a dataset, each offering unique insights into the model's performance and accuracy:

2.2.3.a R-Squared (Coefficent of Determination)

This is the proportion of the variance in the response variable that can be explained by the predictor variables.

$$R^2 = 1 - \frac{RSS}{SST}$$

where:

- Residual Sum of Squares (RSS):

$$RSS = \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$$

- Sum of Squares Total (SST) is the sum of square difference between the observed values y_i and the mean of the observed values $\overline{y}$:

$$SST = \sum_{i=1}^{n}(y_i - \overline{y})^2$$

Interpretation of R^2:

- $R^2 = 1$: This means the model explains 100% of the variance in the dependent variable. All data points lie exactly on the regression line.
- $R^2 = 0$: This means the model does not explain any of the variance in the dependent variable. The model has no explanatory power, and the data points are scattered randomly.
- $0 < R^2 < 1$: This means the model explains some, but not all, of the variance. For example, an R^2 of 0.8 means that 80% of the variance in y is explained by the model, while the remaining 20% is unexplained.

In general, the larger the R-Squared value, the more precisely the predictor variables are able to predict the value of the response variable. How high an R-Squared value needs to be depends on how precise we need to be.

2.2.3.b Adjusted R-Squared

Adjusted R^2 adjusts the regular R^2 value by taking into account the number of predictions in the model. It provides a more accurate measure of model fit by penalizing the addiction of unnecessary predictors.

$$Adjusted\ R^2 = 1 - (1 - R^2) \cdot \frac{n-1}{n-k-1}$$

where:

- n is the number of observations (data points).
- k is the number of predictors (independent variables).

While R^2 will always increase or stay the same when more predictors are added, Adjusted R^2 may decrease if the added predictor does not significantly improve the model. This adjustment helps prevent overfitting when the model becomes overly complex and starts fitting the noise in the data. If a new variable genuinely improves the model's predictive power, the Adjusted R^2 will increase, reflecting a true improvement in model quality.

2.2.3.c Standard Error of the Estimate (Residual Standard Error)

This is the average distance that the observed values fall from the regression line. The smaller the standard error, the better a model is able to fit the data.

$$S = \sqrt{\frac{RSS}{n-k-1}}$$

where:

- RSS: Residual Sum of Squares.
- n: The number of observations (data points).

- k: The number of predictors (independent variables) in the model, not including the intercept.

If we are interested in making predictions using a regression model, the standard error of the regression can be more useful metric to know than R-Squared because it gives us an idea of how precise our predictions will be in terms of units.

2.2.4 Assumptions of Multiple Linear Regression

If any of these assumptions are violated, the results of our linear regression analysis may become unreliable or even misleading:

1. *Linear relationship:* There exists a linear relationship between the independent variable x and the dependent variable y.
   The Pearson correlation coefficient (or Product-moment correlation coefficient) is a measure of the linear association between two variables x and y:

   $$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

   It has a value between -1 and 1 where:
   - -1: Indicates a perfectly negative linear correlation between two variables.
   - 0: Indicates no linear correlation between two variables.
   - 1: Indicates a perfectly positive linear correlation between two variables

   In general, values close to 1 or -1 indicate a strong linear relationship, while values near 0 suggest a weak or no linear relationship.
2. *Independence:* The residuals should be independent of each other. This means there should be no correlation between consecutive residuals, particularly in time-series data.
3. *Homoscedasticity:* The residuals have constant variance at every lelvel of x. This means that, regardless of the value of x (or the predicted values of the dependent variable y), the residuals should have a consistent level of dispersion. Homoscedasticity ensures that the accuracy of the predictions does not change as the value of x changes.
4. *Normality:* The residuals of the model are normally distributed. This assumption is crucial for valid hypothesis testing (e.g., t-test, F-test) and ensures that the model's predictions are reliable. Non-normally distributed residuals can lead to misleading conclusions, particularly when the sample size is small.

2.2.5 Hypothesis Test for Predictors

In multiple linear regression, hypothesis tests play a critical role in evaluating the statistical significance of each predictor variable and the model as a whole. Two main hypothesis tests are typically conducted: the individual t-Test for each predictor and the overall F-Test for the model. Together, these tests provide insight into the predictive power and validity of the regression model.

2.2.5.a Overall Hypothesis Test for All Predictors (F-Test)

F-test evaluates whether the entire set of predictors collectively provides a statistically significant contribution to explaining the variance in the dependent variable y.

- Null hypothesis (H_0): All regression coefficients are equal to zero:

  $$H_0 = \beta_0 = \beta_1 = \beta_2 = \cdots = \beta_k = 0$$

  This implies that none of the predictors have any explanatory power for the dependent variable, and the model offers no significant improvement over one with no predictors.
- Alternative hypothesis (H_a): At least one regression coefficient is non-zero:

  $$H_a : \exists\, \beta_i \neq 0$$

  This suggests that at least one predictor variable contributes to explaining the variance in y.

- The F-test calculation:

$$F = \frac{\frac{SSR}{k}}{\frac{SSE}{n-k-1}}$$

where:

- SSR (Sum of Squares for Regression): Represents the portion of the total variance in y that is explained by the model, while the remaining variance is unexplained.

$$SSR = \sum_{i=1}^{n}(\widehat{y}_i - \overline{y})^2$$

- SSE (Sum of Squares for Error): Also known as Residual Sum of Squares (RSS), represents the portion of the total variance in y that is not explained by the model, reflecting the unexplained or residual variance.

$$SSE = \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$$

- k: The number of predictors in the model.
- n: The total number of observations.

This F-statistic is then compared to a critical value from the F-distribution based on a chosen significance level (α). A significant F-test result indicates that the model as a whole has explanatory power and as least one predictor is significant.
Interpreting the F-test:

- If the p-value of the F-test is less than the significance level (α), we reject H_0. This implies that the model with predictors is statistically significant and provides a better fit than a model with no predictors.
- If the p-value is greater than the significance level (α), we fail to reject H_0, suggesting that the predictors do not collectively explain a significant portion of the variance in y.

2.2.5.b Individual Hypothesis Tests for Each Predictor (t-Test)

Once the overall significance of the model is established, we can examine the statistical significance of each predictor individually. Individual t-Test assesses whether each predictor variable x_i has a significant impact on the dependent variable y_i, considering the effect of other predictors. For each predictor x_i:

- Null hypothesis (H_0): The coefficient of predictor x_i is zero:

$$H_0 : \beta_i = 0$$

This implies that predictor x_i does not have a statically significant effect on y_i.

- Alternative hypothesis (H_a): The coefficient of predictor x_i is not zero:

$$H_a : \beta_i \neq 0$$

This suggests that predictor x_i has a statically significant effect on y_i.

- The t-test calculation:

$$t = \frac{\widehat{\beta}_i}{SE(\widehat{\beta}_i)}$$

where:

- $\widehat{\beta}_i$: The estimated coefficient for predictor x_i.
- $SE(\widehat{\beta}_i)$: The standard error of the coefficient, reflecting the variability in the estimate.

$$SE(\widehat{\beta}_i) = \sqrt{\frac{\widehat{\sigma}^2}{\sum_{j=1}^{n}(x_{ij} - \overline{x}_j)^2}}$$

where:

* $\widehat{\sigma}^2$: The estimate of the variance of the residuals:

$$\widehat{\sigma}^2 = \frac{SSE}{n-k-1}$$

The smaller the value of $\widehat{\sigma}^2$, the more precise estimates of the coefficient, while larger values suggest more variability or uncertainty in the estimate.

* x_{ij}: The value of the j-th predictor variable for the i-th observation.
* $\overline{x_j}$: The mean of the j-th predictor variable over all observations.

The calculated t-value is compared to the critical t-value (or p-value) for a given significant level. Interpreting the t-Test:

- If the p-value for a predictor's t-test is less than the significance level (α), we reject H_0 for that predictor. This implies that x_i has a statistically significant effect on y_i.
- If the p-value is greater than the significance level (α), we fail to reject H_0, suggesting that x_i does not significantly contribute to the model y_i.

2.2.5.c Purposes of Hypothesis Testing in MLR

The purposes of hypothesis testing in MLR is twofold:

- To determine whether the entire set of predictors collectively improves the model's ability to predict the outcome (through the overall F-test).
- To identify which specific predictors significantly contribute to explaining variation in the dependent variable (through individual t-Test).

3 Data Preporcessing

3.1 Initial Data Loading and Handling NA Values

We read data from file `"Intel_CPUs.csv"` into a data frame named `"intelCPU"` for further analysis and proceed to print the first 15 rows in `"intelCPU"` to check if it has successfully loaded data. Figure 1 represents the result of this step.

	Product_Collection	Vertical_Segment	Processor_Number	Status	Launch_Date	Lithography	Recommended_Customer_Price
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
1	7th Generation Intel® Core™ i7 Processors	Mobile	i7-7Y75	Launched	Q3'16	14 nm	$393.00
2	8th Generation Intel® Core™ i5 Processors	Mobile	i5-8250U	Launched	Q3'17	14 nm	$297.00
3	8th Generation Intel® Core™ i7 Processors	Mobile	i7-8550U	Launched	Q3'17	14 nm	$409.00
4	Intel® Core™ X-series Processors	Desktop	i7-3820	End of Life	Q1'12	32 nm	$305.00
5	7th Generation Intel® Core™ i5 Processors	Mobile	i5-7Y57	Launched	Q1'17	14 nm	$281.00
6	Intel® Celeron® Processor 3000 Series	Mobile	3205U	Launched	Q1'15	14 nm	$107.00
7	Intel® Celeron® Processor N Series	Mobile	N2805	Launched	Q3'13	22 nm	N/A
8	Intel® Celeron® Processor J Series	Desktop	J1750	Launched	Q3'13	22 nm	N/A
9	Intel® Celeron® Processor G Series	Desktop	G1610	Launched	Q1'13	22 nm	$42.00
10	Legacy Intel® Pentium® Processor	Mobile	518	End of Interactive Support		90 nm	N/A

Figure 1: *Data frame `"intelCPU"` to check if it has loaded data successfully*

Looking at figure 1, we can see that at row 10 of column "Launch_Date" with a blank space and at row 10 of column "Recommended_Customer_Price" contains a value named "N/A". These 2 cells don't contain any specific values to represent the column's characteristic, so blank values and "N/A" values are considered as missing values in this case. Therefore, we read data from file "Intel_CPUs.csv" with additional step where we specify blank space and "N/A" as "NA", which is a default value to indicate missing values of R programming language, to the data frame "intelCPU" again.

	Product_Collection	Vertical_Segment	Processor_Number	Status	Launch_Date	Lithography	Recommended_Customer_Price
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
1	7th Generation Intel® Core™ i7 Processors	Mobile	i7-7Y75	Launched	Q3'16	14 nm	$393.00
2	8th Generation Intel® Core™ i5 Processors	Mobile	i5-8250U	Launched	Q3'17	14 nm	$297.00
3	8th Generation Intel® Core™ i7 Processors	Mobile	i7-8550U	Launched	Q3'17	14 nm	$409.00
4	Intel® Core™ X-series Processors	Desktop	i7-3820	End of Life	Q1'12	32 nm	$305.00
5	7th Generation Intel® Core™ i5 Processors	Mobile	i5-7Y57	Launched	Q1'17	14 nm	$281.00
6	Intel® Celeron® Processor 3000 Series	Mobile	3205U	Launched	Q1'15	14 nm	$107.00
7	Intel® Celeron® Processor N Series	Mobile	N2805	Launched	Q3'13	22 nm	NA
8	Intel® Celeron® Processor J Series	Desktop	J1750	Launched	Q3'13	22 nm	NA
9	Intel® Celeron® Processor G Series	Desktop	G1610	Launched	Q1'13	22 nm	$42.00
10	Legacy Intel® Pentium® Processor	Mobile	518	End of Interactive Support	NA	90 nm	NA

Figure 2: *Data frame "intelCPU" after processing blank space and "N/A" values*

To make sure that the process has done successfully, we print the first 15 rows of data frame "intelCPU" to see the result. Figure 2 represents the result of this step, we can see that both values in row 10 of columns "Launch_Date" and "Recommended_Customer_Price" have turned into "NA", indicating the process has done successfully. This data processing is done from 19 to line 24.

3.2 Column Selection

We begin to select only columns we care for our analysis, focusing on key attributes like product details, specifications, and performance metrics. These columns include "Product_Collection", "Vertical_Segment", "Status", "Launch_Date", "Lithography", "Recommended_Customer_Price", "nb_of_Cores", "nb_of_Threads", "Processor_Base_Frequency", "Cache", "TDP", "Max_Memory_Size", "Max_Memory_Bandwidth", "Graphics_Base_Frequency", "Graphics_Max_Dynamic_Frequency", and "Instruction_Set".

'Product_Collection' · 'Vertical_Segment' · 'Status' · 'Launch_Date' · 'Lithography' · 'Recommended_Customer_Price' · 'nb_of_Cores' · 'nb_of_Threads' · 'Processor_Base_Frequency' · 'Cache' · 'TDP' · 'Max_Memory_Size' · 'Max_Memory_Bandwidth' · 'Graphics_Base_Frequency' · 'Graphics_Max_Dynamic_Frequency' · 'Instruction_Set'

Figure 3: *Column names in data frame "cpuInfo"*

After choosing the columns and putting them inside a new data frame called "cpuInfo", we then use the "names()" command to print every column names exists in "cpuInfo" and "head()" command to print the first 15 rows of the data frame to verify if we have successfully selected desired columns, the results are illustrated in figure 3 and figure 4. Looking at those figures, we can see that every column we choose for the data frame "cpuInfo" has been successfully selected. The code is from 26 to line 35.

A data.frame: 15 × 16

	Product_Collection	Vertical_Segment	Status	Launch_Date	Lithography	Recommended_Customer_Price	nb_of_Cores	nb_of_Threads	Processor_Base_Frequency	Cache	TDP	Max_Memory_Size
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<int>	<int>	<chr>	<chr>	<chr>	<chr>
1	7th Generation Intel® Core™ i7 Processors	Mobile	Launched	Q3'16	14 nm	$393.00	2	4	1.30 GHz	4 MB SmartCache	4.5 W	16 GB
2	8th Generation Intel® Core™ i5 Processors	Mobile	Launched	Q3'17	14 nm	$297.00	4	8	1.60 GHz	6 MB SmartCache	15 W	32 GB
3	8th Generation Intel® Core™ i7 Processors	Mobile	Launched	Q3'17	14 nm	$409.00	4	8	1.80 GHz	8 MB SmartCache	15 W	32 GB
4	Intel® Core™ X-series Processors	Desktop	End of Life	Q1'12	32 nm	$305.00	4	8	3.60 GHz	10 MB SmartCache	130 W	64.23 GB
5	7th Generation Intel® Core™ i5 Processors	Mobile	Launched	Q1'17	14 nm	$281.00	2	4	1.20 GHz	4 MB SmartCache	4.5 W	16 GB
6	Intel® Celeron® Processor 3000 Series	Mobile	Launched	Q1'15	14 nm	$107.00	2	2	1.50 GHz	2 MB	15 W	16 GB
7	Intel® Celeron® Processor N Series	Mobile	Launched	Q3'13	22 nm	NA	2	2	1.46 GHz	1 MB	4.3 W	4 GB
8	Intel® Celeron® Processor J Series	Desktop	Launched	Q3'13	22 nm	NA	2	2	2.41 GHz	1 MB L2	10 W	8 GB

Figure 4: *The first 15 rows of data frame "cpuInfo"*

3.3 Missing Values Analysis

The code is from line 37 to line 91. We proceed to analyze the missing values of data frame "cpuInfo" with two tasks, computing the total number and the percentage of missing values in each column. In figure 5, we can see that "Graphics_Base_Frequency" and "Graphics_Max_Dynamic_Frequency" have the highest total number of missing values with 1444 and 1554 respectively. Because these are large numbers, we want to see what percentage they occupy in each column in question.

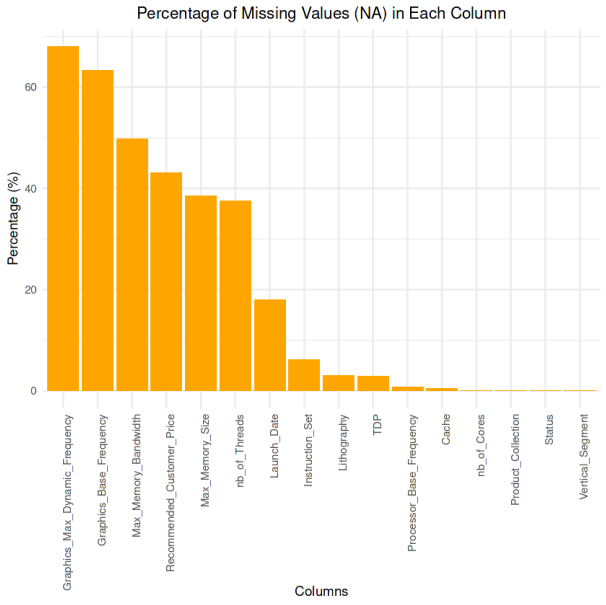
```
Product_Collection          Vertical_Segment
                 0                         0
            Status                Launch_Date
                 0                       412
      Lithography  Recommended_Customer_Price
                71                       982
      nb_of_Cores              nb_of_Threads
                 0                       856
Processor_Base_Frequency                 Cache
                18                        12
               TDP            Max_Memory_Size
                67                       880
Max_Memory_Bandwidth     Graphics_Base_Frequency
              1136                      1444
Graphics_Max_Dynamic_Frequency  Instruction_Set
                          1554              141
```

Figure 5: *The total number of NA values in each column in data frame "cpuInfo"*

Looking at figure 6, it is noticeable that both "Graphics_Base_Frequency" and "Graphics_Max_Dynamic_Frequency" have the highest proportion of missing values with approximately 63 percent and 68 percent, respectively. Both columns have more than half of their data is missing values, if we decide to use these two columns into our data analysis, their high percentage of missing data can cause biased or unreliable results for our model analysis and distort the overall dataset's representativeness, impacting insights and predictions. Therefore, we delete them from the data frame "cpuInfo" to improve the model's performance and facilitate better observations in the future.

Product_Collection	Vertical_Segment
0.0000000	0.0000000
Status	Launch_Date
0.0000000	18.0464301
Lithography	Recommended_Customer_Price
3.1099431	43.0135786
nb_of_Cores	nb_of_Threads
0.0000000	37.4945247
Processor_Base_Frequency	Cache
0.7884363	0.5256242
TDP	Max_Memory_Size
2.9347350	38.5457731
Max_Memory_Bandwidth	Graphics_Base_Frequency
49.7590889	63.2501095
Graphics_Max_Dynamic_Frequency	Instruction_Set
68.0683311	6.1760841

(a) *The percentage of NA values in each column in data frame* `"cpuInfo"`



(b) *Bar chart to illustrate the NA ratio in each column in data frame* `"cpuInfo"`

Figure 6: *The percentage of NA values in each column in data frame* `"cpuInfo"`

Meanwhile, observing from figure 6 again, we can see that the common point of `"Lithography"`, `"Processor_Base_Frequency"`, `"Cache"` and `"TDP"` is they all have percentage of missing values below 5 percent. This is an insignificant number of missing values so by deleting every row contains `"NA"` in each columns in question, we only discard a small portion of the dataset, which has a negligible impact on the size and representativeness of the dataset, but can simplify analysis since we can avoid the need for imputation or complex techniques to handle missing data, making the dataset cleaner and easier to analyze.
Figure 7 demonstrates the results after deleting columns `"Graphics_Base_Frequency"` and `"Graphics_Max_Dynamic_Frequency"` from `"cpuInfo"` data frame and deleting every row contains "NA" value of columns with proportion of missing values below 5 percent.

Product_Collection	Vertical_Segment
0	0
Status	Launch_Date
0	350
Lithography	Recommended_Customer_Price
0	885
nb_of_Cores	nb_of_Threads
0	794
Processor_Base_Frequency	Cache
0	0
TDP	Max_Memory_Size
0	813
Max_Memory_Bandwidth	Instruction_Set
1064	90

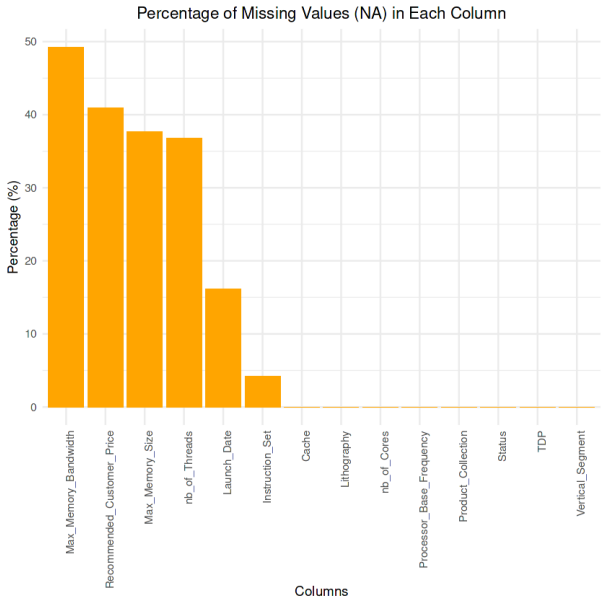
Figure 7: *The total number of NA values in each column in data frame* `"cpuInfo"` *after being processed*

Looking at figure 7, evidently, the total number of missing values in each column, including `"Lithography"`, `"Processor_Base_Frequency"`, `"Cache"` and `"TDP"`, drops to zero, while there are no columns `"Graphics_Base_Frequency"` and `"Graphics_Max_Dynamic_Frequency"` anymore, indicating that the processing has been completed successfully. Looking at figure 8, by deleting "NA" values in four columns in question, the percentage of missing values in each remaining column decreases slightly. For example, in figure 6a the percentage of missing values of `"Max_Memory_Size"` is approximately 38.5 percent but after processing, it drops slightly to about 37.6 in figure 8a. This indicating that this data

deletion has minimal impact on the overall dataset while still contributing to improved performance.

Product_Collection	Vertical_Segment
0.000000	0.000000
Status	Launch_Date
0.000000	16.203704
Lithography	Recommended_Customer_Price
0.000000	40.972222
nb_of_Cores	nb_of_Threads
0.000000	36.759259
Processor_Base_Frequency	Cache
0.000000	0.000000
TDP	Max_Memory_Size
0.000000	37.638889
Max_Memory_Bandwidth	Instruction_Set
49.259259	4.166667

(a) *The percentage of NA values in each column in data frame "cpuInfo" after being processed.*



(b) *Bar chart to illustrate the NA ratio in each column in data frame "cpuInfo" after being processed.*

Figure 8: *The percentage of NA values in each column in data frame "cpuInfo"*

3.4 Data Transformation

By looking at figure 4, we can see that not every value in each column is number since they include a number along with a word or a symbol such as "$" sign in "Recommended_Customer_Price", these current data type will prevent us from further calculation and analysis for model. So our first step is to transform the data type of columns that are not in numeric or integer values into these two value types so that we can make calculation later. Before processing them, we will print out data type of each columns by using the `str()` command to identify what are their current data type to determine suitable method of transformation.

```
$ Product_Collection        : chr  "7th Generation Intel Core i7 Processors" "8
    th Generation Intel Core i5 Processors" ...
$ Vertical_Segment          : chr  "Mobile" "Mobile" "Mobile" "Desktop" ...
$ Status                    : chr  "Launched" "Launched" "Launched" ...
$ Launch_Date               : chr  "Q3'16" "Q3'17" "Q3'17" "Q1'12" ...
$ Lithography               : chr  "14 nm" "14 nm" "14 nm" "32 nm" ...
$ Recommended_Customer_Price: chr  "$393.00 " "$297.00 " "$409.00 " ...
$ nb_of_Cores               : int  2 4 4 4 2 2 2 2 2 1 ...
$ nb_of_Threads             : int  4 8 8 8 4 2 2 2 2 NA ...
$ Processor_Base_Frequency  : chr  "1.30 GHz" "1.60 GHz" "1.80 GHz" ...
$ Cache                     : chr  "4 MB SmartCache" "6 MB SmartCache" ...
$ TDP                       : chr  "4.5 W" "15 W" "15 W" "130 W" ...
$ Max_Memory_Size           : chr  "16 GB" "32 GB" "32 GB" "64.23 GB" ...
$ Max_Memory_Bandwidth      : chr  "29.8 GB/s" "34.1 GB/s" "34.1 GB/s" ...
$ Instruction_Set           : chr  "64-bit" "64-bit" "64-bit" "64-bit" ...
```

Figure 9: *The data type of selected columns for data frame "cpuInfo"*

In figure 9, we can see that "Launch_Date", "Lithography", "Recommended_Customer_Price", "Processor_Base_Frequency", "Cache", "TDP", "Max_Memory_Size", "Max_Memory_Bandwidth" and "Instruction_Set" are all in character data type, so we will determine specific methods to convert each

of them into numeric value. This process helps us standardize and clean the data for further analysis. This step is done from code line 93 to 166. Here is a breakdown of each transformation:

1. **Launch_Date:**
   From figure 4, in column `"Launch_Date"`, the numbers in each value indicates the last two digit of a year. We want to convert them from character values into numeric format to demonstrate years. Therefore, we extract the last two digits in the string and transform them from character type into an integer. If the extracted number is greater than 90, it indicates a year in the 1990s, so we add 1900 to it. Otherwise, we add 2000 to represent a year in the 2000s.
2. **Lithography:**
   Based on figure 4, we see that each value in `"Lithography"` column contains a number with a unit nm, such as "14 nm". Therefore, we proceed to keep only the number by removing the "nm" unit and converting the remaining value into an integer. The column name is updated to `"Lithography (nm)"` to reflect the unit in nm.
3. **Recommended_Customer_Price:**
   From figure 4, the price in each row of `"Recommended_Customer_Price"` has a dollar sign next to it, sometimes a comma is used as a thousands separator and a hyphen to indicate a price range. Therefore, we proceed to remove dollar signs and commas to change the remaining values into numbers. If a price range is provided, the mean of the range is calculated. The column is then renamed to `"Recommended_Customer_Price (USD)"` to reflect the unit in USD.
4. **Processor_Base_Frequency:**
   Looking at figure 9, `"Processor_Base_Frequency"` currently is in character data type and has two units which are MHz and GHz. We standardize this column by converting GHz values to MHz values, where values in GHz unit is multiplied by 1000. After this, we proceed to clean the "MHz" and "GHz" suffix and turn values of the column into double data type. The column is renamed to `"Processor_Base_Frequency (MHz)"` to indicate the frequency is in MHz.
5. **Cache:**
   Looking at figure 9, the `"Cache"` column is in character data type and has it values of a number goes with a type of cache, so we decide to split the `"Cache"` column into two parts: `"Cache_Size (MB)"` and `"Cache_Type"`. We keep the data type of `"Cache_Type"` and fill in any blank cells with "Original", indicating that if a cache isn't assigned to any specific type of cache, it has the original type. Because the size of cache is in two units, MB and KB, so we standardize this column by converting KB values to MB values, where values in KB unit is divided by 1024. After this, we proceed to clean the "MB" and "KB" suffix and turn values of the column into double data type. The result is then represented by the column `"Cache_Size (MB)"`.
6. **TDP:**
   From figure 4, the value in the `"TDP"` column is original in character type where a number goes with a unit "W". Therefore, the values are converted to a numeric value by removing the "W" suffix, and the column name is updated to `"TDP (Watts)"` to indicate the unit of the column as Watts.
7. **Max_Memory_Size:**
   From figure 4, it is clear that `"Max_Memory_Size"` has two units which are GB and TB. We standardize this column by convert TB values into GB values, where we multiply values in TB unit by 1024, then removing the "GB" or "TB" suffix to convert the remaining values into double data type. The column is renamed to `"Max_Memory_Size (GB)` to reflect the unit as GB.
8. **Max_Memory_Bandwidth:**
   From figure 4, we can see that the `"Max_Memory_Bandwidth"` column has values as a number goes with a unit "GB/s". Therefore, we remove the "GB/s" suffix and convert the remaining values to double data type. The column name is updated to `"Max_Memory_Bandwidth (GB/s)"` to reflect the unit as GB/s.
9. **Instruction_Set:**
   Looking at figure 9, we can see that values in `"Instruction_Set"` are displayed with a syntax of *number-bit* or *Itanium number-bit*. Therefore, in order to get the numbers only, we extract the "Intanium" and "-bit" from the values and convert the numbers into integer values. We then change the name of the column to `"Instruction_Set (bit)"` to indicate the unit of the column as bit.

After transformation, we want to recheck again if we has done the process successfully. Therefore, we print all of the data type of each column again and the first 15 rows of the data set `"cpuInfo"` to verify. This step ensures that key numerical attributes are standardized, with appropriate units and data formats, for easier analysis and comparison.

```
$ Product_Collection              : chr  "7th Generation Intel Core i7
    Processors" "8th Generation Intel Core i5 Processors" ...
$ Vertical_Segment                : chr  "Mobile" "Mobile" "Mobile" ...
$ Status                          : chr  "Launched" "Launched" "Launched"  ...
$ Launch_Date                     : num  2016 2017 2017 2012 2017 ...
$ Lithography (nm)                : int  14 14 14 32 14 14 22 22 22 90 ...
$ Recommended_Customer_Price (USD): num  393 297 409 305 281 107 NA NA 42 ...
$ nb_of_Cores                     : int  2 4 4 4 2 2 2 2 2 1 ...
$ nb_of_Threads                   : int  4 8 8 8 4 2 2 2 2 NA ...
$ Processor_Base_Frequency (MHz)  : num  1300 1600 1800 3600 1200 1500 ...
$ Cache_Size (MB)                 : num  4 6 8 10 4 2 1 1 2 1 ...
$ Cache_Type                      : chr  "SmartCache" "SmartCache"  ...
$ TDP (Watts)                     : num  4.5 15 15 130 4.5 15 4.3 10 55 88 ...
$ Max_Memory_Size (GB)            : num  16 32 32 64.2 16 ...
$ Max_Memory_Bandwidth (GB/s)     : num  29.8 34.1 34.1 51.2 29.8 25.6 NA ...
$ Instruction_Set (bit)           : int  64 64 64 64 64 64 64 64 64 32 ...
```

Figure 10: *The data type of selected columns for data frame `"cpuInfo"` after transformation*

Lithography (nm)	Recommended_Customer_Price (USD)	nb_of_Cores	nb_of_Threads	Processor_Base_Frequency (MHz)	Cache_Size (MB)	Cache_Type	TDP (Watts)	Max_Memory_Size (GB)	Max_Memory_Bandwidth (GB/s)	Instruction_Set (bit)
<int>	<dbl>	<int>	<int>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<int>
14	393	2	4	1300	4	SmartCache	4.5	16.00	29.8	64
14	297	4	8	1600	6	SmartCache	15.0	32.00	34.1	64
14	409	4	8	1800	8	SmartCache	15.0	32.00	34.1	64
32	305	4	8	3600	10	SmartCache	130.0	64.23	51.2	64
14	281	2	4	1200	4	SmartCache	4.5	16.00	29.8	64
14	107	2	2	1500	2	Original	15.0	16.00	25.6	64
22	NA	2	2	1460	1	Original	4.3	4.00	NA	64
22	NA	2	2	2410	1	L2	10.0	8.00	NA	64

Figure 11: *The first 15 rows in data frame `"cpuInfo"` after transformation*

As we can see, from figure 10, every column we transformed has turned into numeric or integer values and the column names has an additional unit indicator alongside with them, satisfying our demand. Looking at each column in figure 11, we also can tell that the transformation has done successfully, such as with column `"Instruction_Set"`, its values only contain numbers and the name has changed into `"Instruction_Set (bit)"`.

3.5 Missing Values Filling Strategy

The filling strategy for missing values code is from line 168 to line 215. We address missing values (NA) in various columns of the "`cpuInfo`" data frame by filling in values based on specific strategies.

1. **Filling Missing Values in Recommended_Customer _Price (USD):**
   From figure 8a, we see that about 41 percent of values is absent in this column. In order to fill all the values, the missing values in the `"Recommended_Customer_Price (USD)"` column are filled by first

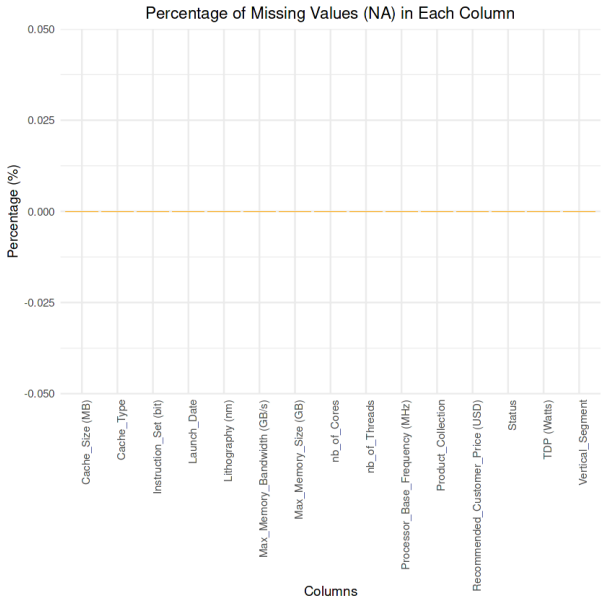
grouping the data by *"Product_Collection"*, and using the `fill()` function from the `tidyr` package to propagate the available values both upwards and downwards within each group. This ensures that prices are filled in a way that aligns with the corresponding price range of other machines in the same segment, preventing irrelevant values from being propagated. This is repeated for the `"Vertical_Segment"` to further ensure the missing values are appropriately filled. By using this method, it ensures consistency and standardization of prices within similar product collections and vertical segments.

2. **Filling Missing Values in Launch_Date:**
   From figure 8a, `"Lauch_Date"` has around 16 percent of missing values. We use the same method of `"Recommended_Customer_Price (USD)"` for `"Launch_Date"`, ensuring that the launch date is consistent within specific product collections and vertical segments. We fills missing values by propagating the nearest non-missing values both forward and backward within each group to ensure that missing launch dates are appropriately filled, maintaining logical consistency without introducing bias from unrelated rows.
3. **Filling Missing Values in Instruction_Set (bit):**
   From figure 10 and figure 11, it is clear that almost all values in the `"Instruction_Set (bit)"` column are either 64 or 32, so we decided to fill in the missing values by replacing "NA" with the most frequent value in the column. We use the `mode()` function to identify the most common value by counting the occurrences of the unique values. By applying this technique, the missing values are filled with the most frequent instruction set bit value, ensuring consistency in the dataset.
4. **Filling Missing Values in Other Columns:**
   For the `"nb_of_Threads"` column, in figure 11, we observe that many the number of threads is around twice the number of cores, some of them has the same value. Seeing this tight relationship between these two columns, we proceed to fill missing values in the `"nb_of_Threads"` by multiplying the corresponding value in `"nb_of_Cores"` by 2, assuming the number of threads is twice the number of cores. This approach is chosen because threads and cores are closely related, with threads typically being twice the number of cores, making it a logical assumption.

   Missing values in `"Max_Memory_Size (GB)"`, `"Max_Memory_Bandwidth (GB/s)"`, `"Graphics_Base_Frequency (MHz)"`, and `"Graphics_Max_Dynamic_Frequency (MHz)"` are filled with the median values of their respective columns, calculated while ignoring the NAs. The median is often used as a method to fill missing values (NAs) because it is less sensitive to outliners than the mean. By using median, we ensure that the NAs are replaced with a central value, maintaining the overall distribution of the data.

After filling missing values for all columns, we proceed to print the total number of missing values in each column and a bar chart demonstrating the percentage of missing values in each column alongside to check if we overlook any missing values after the filtering process for timely treatment. From figure 12, it is clear that there is no missing values left in the data frame since the total number of missing values in each columns noticeably drops to zero.

```
                  Product_Collection                    Vertical_Segment
                                   0                                   0
                              Status                         Launch_Date
                                   0                                   0
              Lithography (nm)       Recommended_Customer_Price (USD)
                                   0                                   0
                     nb_of_Cores                      nb_of_Threads
                                   0                                   0
Processor_Base_Frequency (MHz)                      Cache_Size (MB)
                                   0                                   0
                      Cache_Type                          TDP (Watts)
                                   0                                   0
           Max_Memory_Size (GB)                Max_Memory_Bandwidth (GB/s)
                                   0                                   0
         Instruction_Set (bit)
                                   0
```



(a) *The total number of missing values in each column after processing*

(b) *Bar chart to illustrate the percentage of missing values in each column after processing*

Figure 12: *The percentage of NA values in each column after processing*

We also decide to print the first 15 rows of the dataset `"cpuInfo"` to briefly check if the missing values has been replaced by a meaningful specific value. For example, in figure 2, the row number 10 of `"Launch_Date"` is filled with "NA" value, after this step, it has turned into 2004 illustrated in figure 13.

Product_Collection	Vertical_Segment	Status	Launch_Date	Lithography (nm)	Recommended_Customer_Price (USD)	nb_of_Cores	nb_of_Threads	Processor_Base_Frequency (MHz)	Cache_Size (MB)	Cache_Type	TDP (Watts)	Max_Memory_Size (GB)	Max_Mer
<chr>	<chr>	<chr>	<dbl>	<int>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>	
7th Generation Intel® Core™ i7 Processors	Mobile	Launched	2016	14	393	2	4	1300	4	SmartCache	4.5	16.00	
8th Generation Intel® Core™ i5 Processors	Mobile	Launched	2017	14	297	4	8	1600	6	SmartCache	15.0	32.00	
8th Generation Intel® Core™ i7 Processors	Mobile	Launched	2017	14	409	4	8	1800	8	SmartCache	15.0	32.00	
Intel® Core™ X-series Processors	Desktop	End of Life	2012	32	305	4	8	3600	10	SmartCache	130.0	64.23	
7th Generation Intel® Core™ i5 Processors	Mobile	Launched	2017	14	281	2	4	1200	4	SmartCache	4.5	16.00	
Intel® Celeron® Processor 3000 Series	Mobile	Launched	2015	14	107	2	2	1500	2	Original	15.0	16.00	
Intel® Celeron® Processor N Series	Mobile	Launched	2013	22	107	2	2	1460	1	Original	4.3	4.00	
Intel® Celeron® Processor J Series	Desktop	Launched	2013	22	72	2	2	2410	1	L2	10.0	8.00	
Intel® Celeron® Processor G Series	Desktop	Launched	2013	22	42	2	2	2600	2	SmartCache	55.0	32.00	
Legacy Intel® Pentium® Processor	Mobile	End of Interactive Support	2004	90	64	1	2	2800	1	L2	88.0	32.00	

Figure 13: *The first 15 rows of data frame `"cpuInfo"` after processing*

3.6 Summary

Here is the summarize table for this process:

No.	Operation	Description
1	Process blank space and "N/A" values	Replace blank spaces or "N/A" with "NA".

No.	Operation	Description
2	Remove columns and rows	Remove columns if their ratio $\geq 50\%$ and rows in columns with "NA" ratio $\leq 5\%$ such as `"Lithography"`, `"Cache"`, `"TDP"`.
3	Change `"Launch_Date"` data type	Extract the last two digits of a string. If the value is less than 90, add 2000 to interpret it as a year; otherwise, add 1900.
4	Change `"Lithography"` data type	Remove "nm" unit and convert the remaining number to int. Change column name.
5	Change `"Recommended_Customer_Price"` data type	Remove dollar signs, commas and handle hyphens to convert to double type. Change column name.
6	Change `"Cache"` data type	Split into 2 columns where the `"Cache_Type"` is in char and convert the `"Cache_Size (MB)"` to double type.
7	Change `"TDP"` data type	Remove the "W" suffix and convert to double type. Change column name.
8	Change `"Max_Memory_Size"` data type	The value is converted to double type and expressed in gigabytes (GB). Change column name.
9	Change `"Max_Memory_Bandwidth"` data type	The value is converted to double type and expressed in gigabytes per second (GB/s). Change column name.
10	Change `"Instruction_Set"` data type	The value is converted to int type and expressed in "bit" unit.
11	Filling Missing Values in `"Recommended_Customer_Price"`	We group the data by `"Product_Collection"`, `"Vertical_Segment"` and fill available values from both upward and downward directions.
12	Filling Missing Values in `"Launch_Date"`	We fill with nearest non-missing values from both upward and downward column.
13	Filling Missing Values in `"Instruction_Set"`	We fill with the most frequent value in the column.
14	Filling Missing Values in other columns	In `"nb_of_threads"` column, the NA values is twice as the corresponding value in `"nb_of_cores"`. In other columns, we fill with median values of the respective columns.

Table 3: *Data Preprocessing Operations and Descriptions Summary*

4 Descriptive Statistics

4.1 General view of all attributes

For descriptive statisfics, we first calculate the summary of our data frame for further analysis. We select columns that are in character data type of the data frame `"cpuInfo"`, including columns `"Product_Collection"`, `"Vertical_Segment"`, `"Status"` and `"Cache_Type"`. A new data frame called `"cpuFinal"` is made as a copy of the data frame `"cpuInfo"` but without these mentioned columns in order to compute and analyze. After this step, we would want to print out all of the data type of each columns to ensure again every column in the `"cpuFinal"` only contains numeric values. Looking at figure 14, the said condition is fulfilled, so we continue to do the last step, which is computing the summary of the data frame.

```
$ Launch_Date                    : num [1:2160] 2016 2017 2017 2012 2017 ...
$ Lithography (nm)               : int [1:2160] 14 14 14 32 14 14 22 22 22 ...
$ Recommended_Customer_Price (USD): num [1:2160] 393 297 409 305 281 107  ...
$ nb_of_Cores                    : int [1:2160] 2 4 4 4 2 2 2 2 2 1 ...
$ nb_of_Threads                  : num [1:2160] 4 8 8 8 4 2 2 2 2 2 ...
$ Processor_Base_Frequency (MHz) : num [1:2160] 1300 1600 1800 3600 1200 ...
$ Cache_Size (MB)                : num [1:2160] 4 6 8 10 4 2 1 1 2 1 ...
$ TDP (Watts)                    : num [1:2160] 4.5 15 15 130 4.5 15 4.3 ...
$ Max_Memory_Size (GB)           : num [1:2160] 16 32 32 64.2 16 ...
$ Max_Memory_Bandwidth (GB/s)    : num [1:2160] 29.8 34.1 34.1 51.2 29.8 ...
$ Instruction_Set (bit)          : int [1:2160] 64 64 64 64 64 64 64 64 64 ...
```

Figure 14: *The data type of selected columns for data frame "cpuFinal"*

By using the `summary()` command, we then obtain the minimum value, the 1st quartile, median, mean, the 3rd quartile and the maximum value of each column. This process provides a comprehensive statistical summary of the data, allowing for further analysis or insights into the dataset. We summarize the random variables by running the R code in line 233 and get the result:

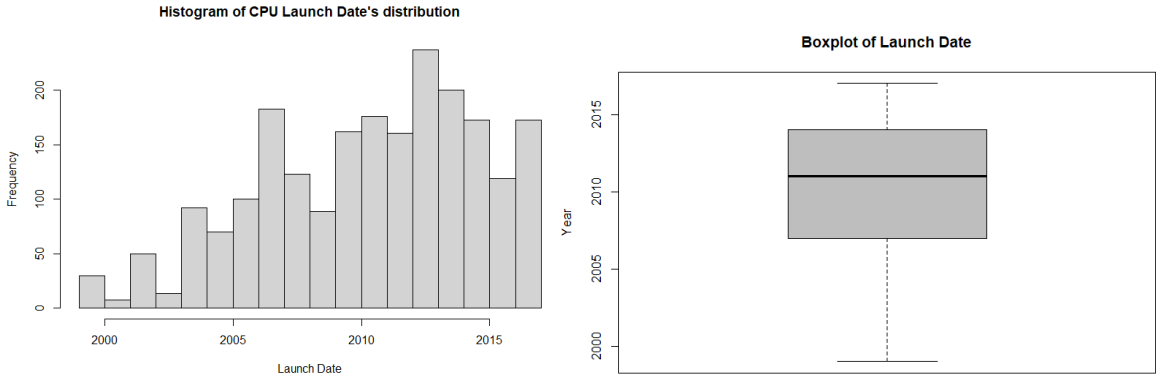
```
Launch_Date   Lithography (nm) Recommended_Customer_Price (USD)
 Min.   :1999   Min.   : 14.00   Min.   :    9.62
 1st Qu.:2007   1st Qu.: 22.00   1st Qu.:   93.75
 Median :2011   Median : 32.00   Median :  234.00
 Mean   :2011   Mean   : 49.29   Mean   :  666.73
 3rd Qu.:2014   3rd Qu.: 65.00   3rd Qu.:  539.50
 Max.   :2017   Max.   :250.00   Max.   :13011.00
  nb_of_Cores     nb_of_Threads     Processor_Base_Frequency (MHz)
 Min.   : 1.000   Min.   :  1.000   Min.   : 300
 1st Qu.: 2.000   1st Qu.:  2.000   1st Qu.:1730
 Median : 2.000   Median :  4.000   Median :2300
 Mean   : 4.161   Mean   :  7.726   Mean   :2286
 3rd Qu.: 4.000   3rd Qu.:  8.000   3rd Qu.:2830
 Max.   :72.000   Max.   :144.000   Max.   :4300
 Cache_Size (MB)     TDP (Watts)     Max_Memory_Size (GB)
 Min.   : 0.01562   Min.   :  0.65   Min.   :   1.0
 1st Qu.: 2.00000   1st Qu.: 27.00   1st Qu.:  32.0
 Median : 4.00000   Median : 51.00   Median :  32.0
 Mean   : 7.35041   Mean   : 61.01   Mean   : 188.6
 3rd Qu.: 8.00000   3rd Qu.: 85.00   3rd Qu.:  64.0
 Max.   :60.00000   Max.   :300.00   Max.   :4198.4
 Max_Memory_Bandwidth (GB/s) Instruction_Set (bit)
 Min.   :  1.60              Min.   :32.00
 1st Qu.: 25.60              1st Qu.:64.00
 Median : 25.60              Median :64.00
 Mean   : 30.95              Mean   :58.13
 3rd Qu.: 25.60              3rd Qu.:64.00
 Max.   :352.00              Max.   :64.00
```

Figure 15: *Statistical results of intel-CPUs*

Overall, the statistical results shown in figure 15 reflect clear advancements in computing technology, including improvements in lithography, increasing core and thread counts, and larger memory capacities, which align with real-world trends. However, extreme values for metrics like price, cache size, memory size, and core/thread counts indicate the presence of niche or specialized processors, distinct from typical consumer products. Additionally, the right-skewed distribution of several metrics, such as price and cores, highlights the dominance of a few high-end products within these categories.
These summary statistics provide a quick understanding of the distribution and spread of the data. For further analysis, we use Histogram and Boxplot graphs to examine the distribution. The code is from

line 235 to line 370.

4.1.1 Launch Date



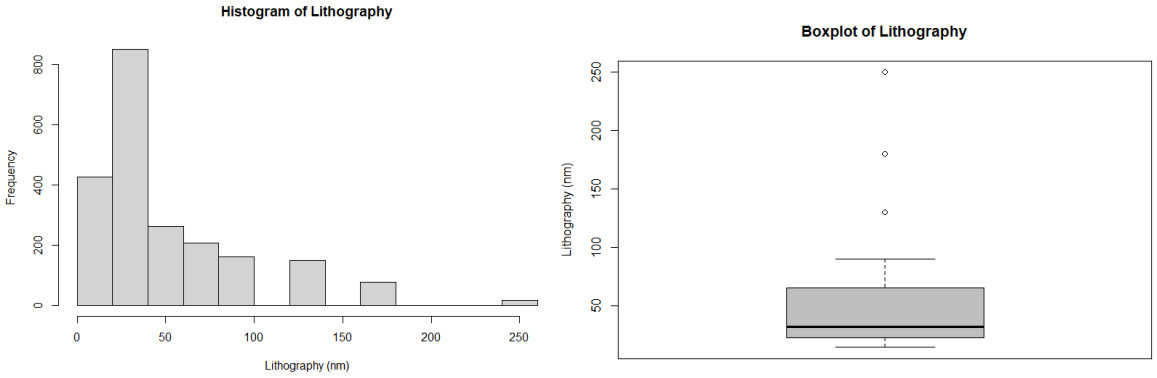
(a) *Histogram of Launch Date illustrates the distribution of CPU launches across years, highlighting peaks and trends in production over time*

(b) *Boxplot of Launch Date visualizes the spread and central tendency of launch dates, showing the range from 1999 to 2017 with a median around 2011*

Figure 16: *Launch Date plots*

- This stat shows how CPU releases are distributed over time which can help us identify trends like whether CPU production will increase in certain years.
- For the CPU launch dates, the launch dates range from 1999 to 2017. The median value is 2011, and the mean is the same. This very small difference suggests that the distribution is likely symmetrical.

4.1.2 Lithography



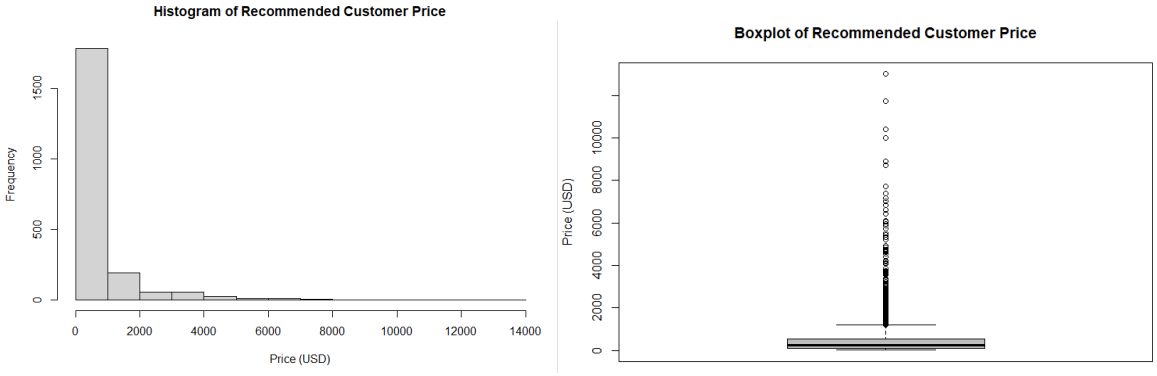
(a) *Histogram of Lithography reflects the transition to smaller lithography sizes over time*

(b) *Boxplot of Lithography highlights the range from older (larger) to modern (smaller) lithography sizes*

Figure 17: *Lithography plots*

- Lithography size is crucial in CPU development. Smaller sizes mean newer technology.
- Lithography sizes range from 14nm to 250nm, the median value is approximately 32nm, and the mean is 49.29nm. This suggests that the distribution is likely right-skewed.

4.1.3 Recommended Customer Price



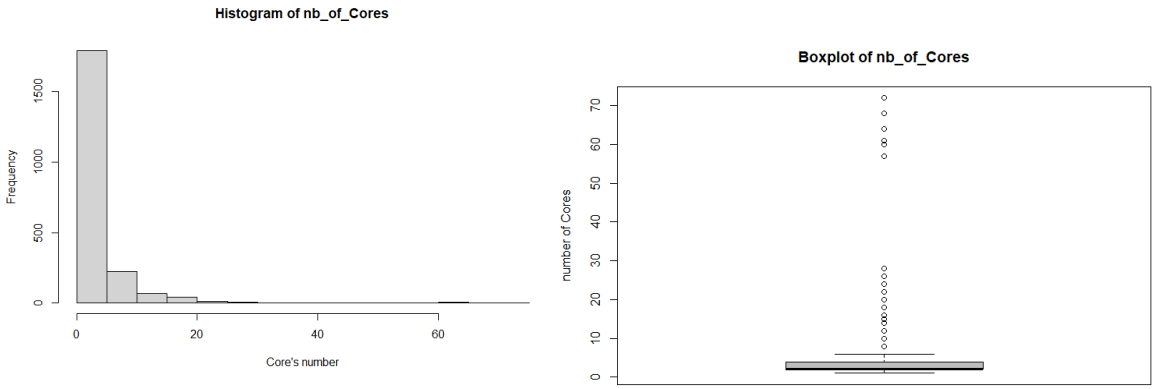
(a) *Histogram of Recommended Customer Price shows the price distribution, with most CPUs in the moderate range and a few high-end outliers*

(b) *Boxplot of Recommended Customer Price highlights the wide variability in prices with significant high-end outliers*

Figure 18: *Recommended Customer Price Graphs*

- CPU price is the main factor to see whether the customers or businesses buy and develop it.
- For the recommended customer price, prices range from 9.62 dollars to 13011 dollars. The median value is approximately 234 dollars, and the mean is 666.73 dollars. This alignment indicates that the distribution is likely left-skewed.

4.1.4 Number of Cores



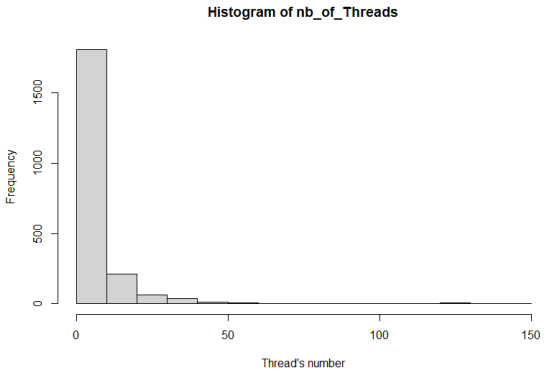
(a) *Histogram of nb-of-Cores displays the distribution of CPU core counts, with most CPUs having lower counts*

(b) *Boxplot of nb-of-Cores highlights the range, including high-core-count outliers*

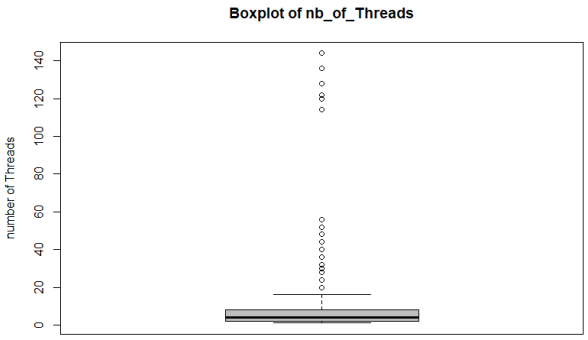
Figure 19: *Number of Cores plots*

- The number of physical CPU cores in a processor, each core can handle its own task simultaneously.
- The number of cores ranges from 1 to 72. The median value is 2, and the mean is higher at 4.161. This indicates that the distribution is likely left-skewed.

4.1.5 Number of Threads



(a) *Histogram of nb-of-Threads shows the distribution of threads, reflecting a trend toward multi-threaded processing*

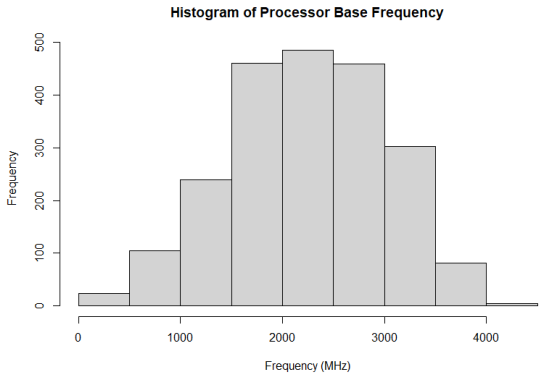


(b) *Boxplot of nb-of-Threads displays the variability in thread counts, with some high-thread outliers*

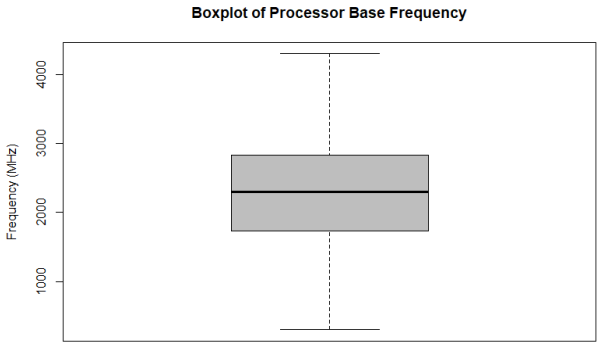
Figure 20: *Number of Threads plots*

- The number of threads a CPU can handle simultaneously, including physical and virtual cores.
- The number of cores ranges from 1 to 144. The median value is 4, and the mean is higher at 7.726. This indicates that the distribution is likely left-skewed.

4.1.6 Processor Base Frequency



(a) *Histogram of Processor Base Frequency illustrates the distribution of base frequencies, clustering around common values*

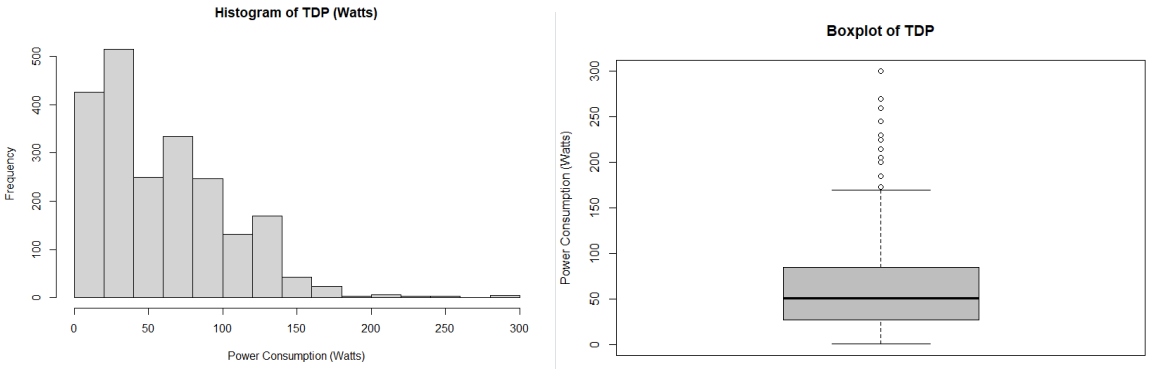


(b) *Boxplot of Processor Base Frequency shows a relatively stable range of base frequencies*

Figure 21: *Processor Base Frequency plots*

- The higher base frequency means the CPU can have a faster performance in tasks that rely heavily on CPU speed.
- Base frequencies range from 300 MHz to 4300 MHz. The median value is 2300 MHz, and the mean is approximately the same at 2286 MHz. This similarity indicates that the distribution is likely symmetrical.

4.1.7 Thermal Design Power (TDP)



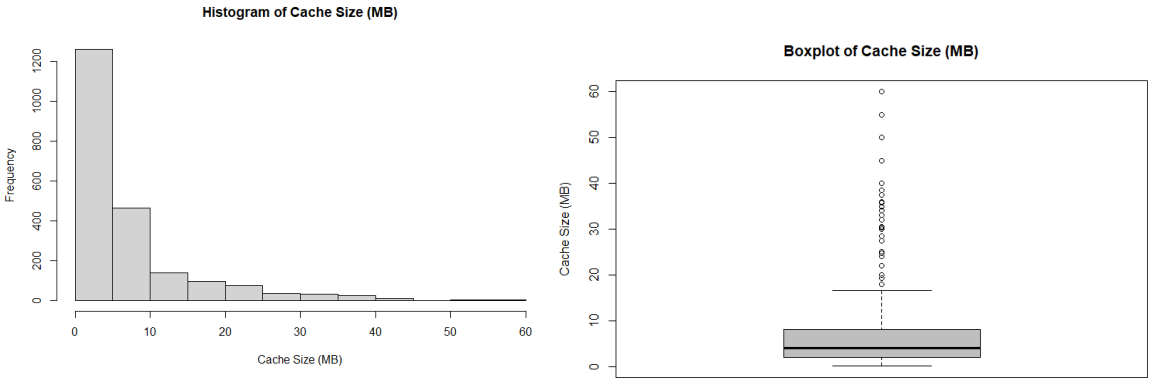
(a) *Histogram of TDP highlights the distribution of power efficiency across CPUs.*

(b) *Boxplot of TDP displays the TDP range, with some high-power outliers*

Figure 22: *TDP plots*

- Thermal Design Power (TDP) reflects power efficiency and energy consumption.
- For the thermal design power, TDP values range from 0.65W to 300W. The median value is approximately 51W, and the mean is slightly higher at 61.01. This difference suggests that the distribution is likely slightly left-skewed.

4.1.8 Cache Size



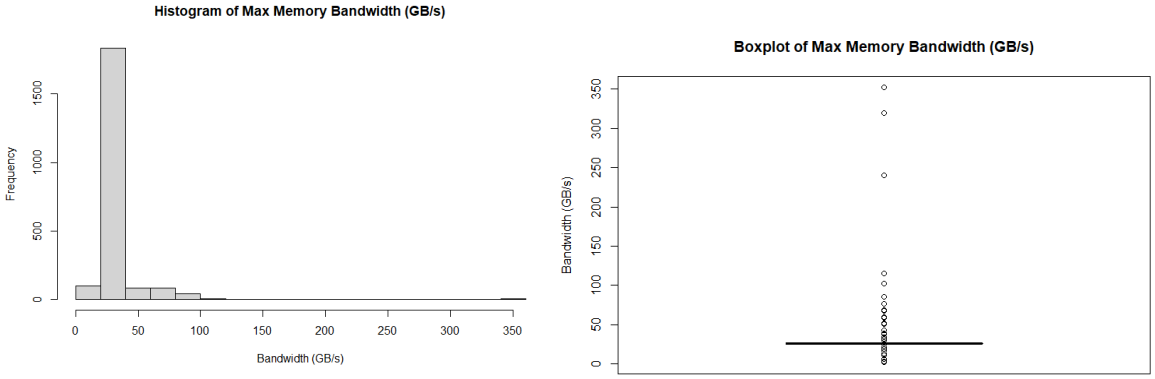
(a) *Histogram of Cache Size reflects the distribution of cache sizes, with most CPUs in the moderate range*

(b) *Boxplot of Cache Size highlights variability and large-cache outliers.*

Figure 23: *Cache Size plots*

- Cache size impacts CPU performance.
- Cache sizes range from 0.01562 MB to 60 MB. The median value is approximately 4 MB, and the mean is noticeably higher at 7.35041 MB. This difference suggests that the distribution is likely left-skewed.

4.1.9 Max Memory Bandwidth



(a) *Histogram of Max Memory Bandwidth shows typical memory bandwidths, with a concentration around common values*

(b) *Boxplot of Max Memory Bandwidth displays the range, with some high-bandwidth outliers*

Figure 24: *Max Memory Bandwidth plots*

- This shows the typical memory bandwidths supported by CPUs, critical for system performance.
- For the maximum memory bandwidth, memory bandwidths range from 1.6 GB/s to 352 GB/s. The median value is 25.6 GB/s, and the mean is around 30.95 GB/s. This suggests that the distribution is likely left-skewed.

4.1.10 Correlation

Next, we compare the relation between each pair of variables by using a correlation plot.

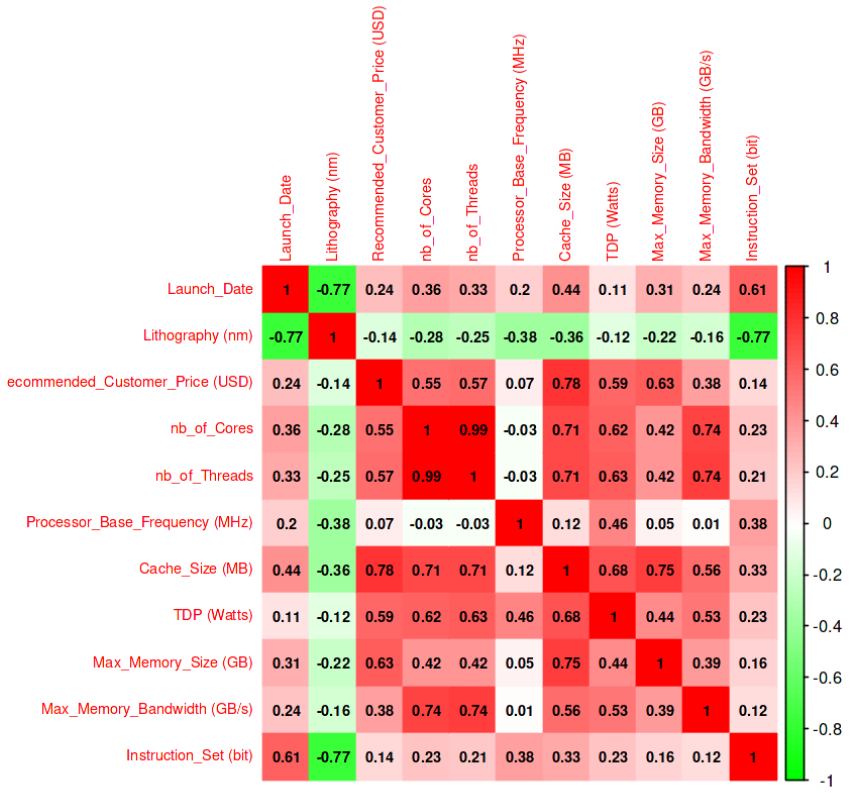


Figure 25: *Correlation between attributes for CPU*

- Launch Date has weak correlations with most other variables, indicating that the year of launch does not strongly affect attributes like Cache, Price, or nb-of-Cores. This suggests that CPU improvements are driven more by technological advancements than by the specific year of release.
- Lithography size negatively correlates with variables like Processor Base Frequency and Cache. Smaller lithography sizes often accompany higher frequencies and larger caches, reflecting technological advancements in manufacturing.
- Recommended Customer Price shows moderate positive correlations with Cache and nb-of-Threads. CPUs with larger caches and higher thread counts are priced higher, consistent with premium CPU segments.
- Nb-of-Cores correlates positively with Cache and TDP, indicating that CPUs with more cores typically require higher power and include larger caches to support enhanced performance.
- Nb-of-Threads shows a strong positive correlation with Cache and Recommended Customer Price, reflecting that CPUs with higher thread counts more expensive.
- Processor Base Frequency has almost no significant correlations with other variables. This suggests that base frequency is an independent design feature and does not directly influence attributes like cache size or price.
- TDP correlates weakly with nb-of-Cores and Cache. Higher power consumption generally accompanies CPUs with more cores and larger caches, reflecting increased performance and processing capabilities.
- Cache exhibits strong positive correlations with nb-of-Threads, Recommended Customer Price, and Max Memory Bandwidth. This indicates that CPUs with larger caches typically have higher thread counts, support higher memory bandwidth, and are priced higher.
- Max Memory Bandwidth correlates moderately with Cache and nb-of-Cores. Higher bandwidths are associated with CPUs that have larger caches and more cores, making them suitable for high-performance computing tasks.

4.2 Analysis of TDP Across Lithography Levels

Observing figure 25, we note that the correlation between TDP and Lithography is almost the weakest compared to the correlations between TDP and other attributes. Therefore, we aim to analyze TDP in relation to Lithography to gain deeper insights into their relationship. We first summarize the TDP grouped by Lithography using R from line 373 to line 382, then it yields the result:

```
  Lithography..nm. `5% Quantile` `95% Quantile`  Mean    SD
  <fct>                     <dbl>          <dbl> <dbl> <dbl>
1 14                         6             165    66.5 54.3
2 22                         8.1           140    60.6 48.0
3 32                        10.5           130    61.4 41.2
4 45                         2.41          130    58.5 40.3
5 65                        10             130    59.2 39.6
6 90                         7.5           129    82.3 36.3
7 130                        9.35          102.   49.9 28.9
8 180                       13.9            68.1  29.9 17.8
9 250                       19.5            39.9  29.9  6.95
```

Figure 26: *Summary statistics of TDP for various lithography sizes*

The 5% Quantile and 95% Quantile values indicate the range within which the majority of TDP values lie. Smaller lithography nodes (e.g., 14 nm, 22 nm) have wider TDP ranges, while larger lithography nodes (e.g., 180 nm, 250 nm) show narrower ranges. This trend suggests that smaller lithography, often associated with more advanced manufacturing processes, accommodate a broader range of TDP values, likely due to the inclusion of both high-performance and energy-efficient designs within the same technology. Conversely, larger lithography tend to focus on simpler designs (as they are created at the very first era of CPUs), resulting in narrower TDP ranges. For the mean and SD, we have a plot for easier interpreting (the code is from line 384 to line 401):

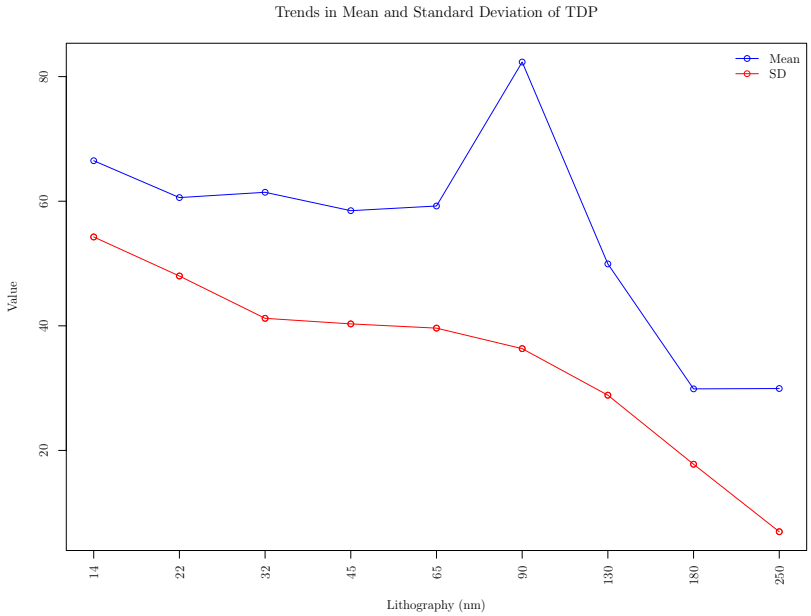


Figure 27: *Trends in Mean and Standard Deviation of TDP across different Lithography levels*

Looking at figure 27, we observe that the mean TDP values generally decrease as lithography size increases (from 14 nm to 250 nm). This suggests that older lithography technologies (with larger sizes) tend to have lower average TDP compared to newer, smaller lithography sizes.
The SD of TDP is higher for smaller lithography sizes (e.g., 14 nm, 22 nm) and decreases significantly for larger lithography sizes (e.g., 180 nm, 250 nm). This indicates that there is more variability in TDP values for modern, smaller-size technologies, potentially due to greater differences in chip design or thermal management challenges at smaller scales.
We will have a deeper insights for these attributes in Inferential Statistics section.

5 Inferential Statistics

5.1 ANOVA

5.1.1 Objective

As mentioned earlier at figure 25, we have observed a strong correlation between Thermal Design Power (TDP) and other attributes. This motivates us to utilize TDP for further analysis. In this section, we are into the difference in the TDP between the Lithography.
Performing ANOVA test requires us to set the null hypothesis and alternative hypothesis, which are:

- H_0 : The mean TDP is the same across all lithography types.
- H_1 : At least one lithography type has a mean TDP that differs from the others.

5.1.2 Assumptions Testing

Test for Normality assumption.

We choose Shapiro-Wilk test to ensure residual is normally distributed (415 to line 417).

```
    Shapiro-Wilk normality test

data:  residuals(litho_anova_model)
W = 0.93256, p-value < 2.2e-16
```

Figure 28: *Shapiro-Wilk test performed normality testing on the residuals*

Based on the results of the Shapiro-Wilk test (figure 28), the p-value is less than 2.2×10^{-16}. Since the p-value is significantly smaller than the significance level of 0.05, we reject the null hypothesis of the test, which states that the residuals are normally distributed. This indicates that the residuals of the lithography analysis model deviate from normality.

Test for Homoscedasticity assumption.

We choose Bartlett test to ensure the variances of residual are the same (line 419 to line 421).

```
    Bartlett test of homogeneity of variances

data:  TDP..Watts. by Lithography..nm.
Bartlett's K-squared = 237.59, df = 8, p-value < 2.2e-16
```

Figure 29: *Bartlett test performed homoscedasticity testing on the residuals*

Based on the results of Bartlett's test for homogeneity of variances (figure 29), the test statistic the p-value is less than 2.2×10^{-16}. Since the p-value is significantly smaller than the significance level of 0.05, we reject the null hypothesis of the test, which states that the variances across groups are equal. This indicates that the assumption of homoscedasticity is violated for the lithography analysis model.
As none of the assumptions were satisfied, the subsequent ANOVA results are presented for reference purposes only. Alternative methods or transformations will be proposed in Extension section.

5.1.3 One-way ANOVA Testing

Step 1. Perform ANOVA testing

We want to explore the relationship between the TDP and various lithography levels using ANOVA. To facilitate this, the independent variable is first converted into a factor using `as.factor()`, which enables its use in the ANOVA model as a categorical variable. The ANOVA model is then fit using the `aov()` function, Following the fitting of the model (line 411 to line 413), the summary of the results yields:

```
                    Df  Sum Sq Mean Sq F value Pr(>F)
Lithography..nm.     8  202852   25356   13.21 <2e-16 ***
Residuals         2151 4127666    1919
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 30: *ANOVA testing summary*

Based on figure 30, the F-value for lithography is 13.21, with a p-value of less than 2×10^{-16}, which is well below the significance threshold of 0.05. This indicates that the differences in TDP across the various lithography levels are highly significant. In other words, the lithography technology has a notable impact on the thermal performance.
The residuals, with a sum of squares of $4,127,666$ and mean square of $1,919$, suggest that there is still variability in TDP that cannot be explained by lithography alone, although the effect of lithography is clearly substantial.

Step 2. Perform Post-hoc analysis

Although the ANOVA test has indicated that there are significant differences in the means of TDP among the various lithography groups, it does not provide specific information about which pairs of groups differ from each other. To identify these pairwise differences, we perform a post-hoc analysis using Tukey's Honest Significant Difference (HSD) test. This test compares all possible pairs of lithography levels while controlling for the Type I error rate, ensuring that the comparisons are valid and that any significant differences in TDP are appropriately identified (line 423 to line 426).

```
    Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = TDP..Watts. ~ Lithography..nm., data = data_anova)

$Lithography..nm.
                diff         lwr         upr     p adj
22-14     -5.91567878 -14.781087   2.949729 0.4930541
32-14     -5.06513761 -15.054986   4.924711 0.8189280
45-14     -8.01768061 -18.674325   2.638964 0.3207986
65-14     -7.26923077 -18.765481   4.227019 0.5693934
90-14     15.84049080   3.321790  28.359191 0.0028394
130-14   -16.56506667 -29.470649  -3.659484 0.0022643
180-14   -36.61898734 -53.274183 -19.963791 0.0000000
250-14   -36.56315789 -68.451818  -4.674497 0.0113521
32-22      0.85054116  -8.738296  10.439378 0.9999990
45-22     -2.10200183 -12.383674   8.179670 0.9994102
65-22     -1.35355199 -12.503103   9.795999 0.9999888
90-22     21.75616957   9.555080  33.957259 0.0000012
130-22   -10.64938789 -23.247117   1.948341 0.1768296
180-22   -30.70330857 -47.121112 -14.285505 0.0000003
250-22   -30.64747912 -62.412796   1.117838 0.0685685
45-32     -2.95254299 -14.218172   8.313086 0.9964943
65-32     -2.20409315 -14.267016   9.858830 0.9997439
90-32     20.90562841   7.864608  33.946648 0.0000245
130-32   -11.49992905 -24.912773   1.912915 0.1621890
180-32   -31.55384973 -48.605122 -14.502577 0.0000004
250-32   -31.49802028 -63.595326   0.599285 0.0592398
65-45      0.74844984 -11.872211  13.369111 1.0000000
90-45     23.85817141  10.299588  37.416755 0.0000019
130-45    -8.54738606 -22.463973   5.369200 0.6089439
180-45   -28.60130673 -46.051604 -11.151009 0.0000138
250-45   -28.54547729 -60.856528   3.765573 0.1335513
90-65     23.10972157   8.881760  37.337683 0.0000175
130-65    -9.29583590 -23.865362   5.273690 0.5568660
180-65   -29.34975657 -47.325089 -11.374424 0.0000153
250-65   -29.29392713 -61.891530   3.303675 0.1187320
130-90   -32.40555746 -47.794683 -17.016432 0.0000000
180-90   -52.45947814 -71.105300 -33.813656 0.0000000
250-90   -52.40364869 -85.375725 -19.431573 0.0000304
180-130  -20.05392068 -38.961667  -1.146174 0.0279618
250-130  -19.99809123 -53.118991  13.122808 0.6314105
250-180    0.05582945 -34.698098  34.809757 1.0000000
```

Figure 31: *Tukey's HSD summary for pairwise comparisons after ANOVA*

The results of Tukey's HSD test are presented in figure 31, where the significant pairwise differences in TDP across lithography levels are shown. Notably, significant differences were found between several lithography groups, such as between $90nm$ and $14nm$, $130nm$ and $14nm$, and $180nm$ and $14nm$, among others, as indicated by the adjusted p-values (e.g., $p = 0.0000012$, which is significantly smaller than the threshold of 0.05, for the $90nm$ and $14nm$ comparison).
For easier interpretation, we visualize the results of Tukey's HSD test in a plot.

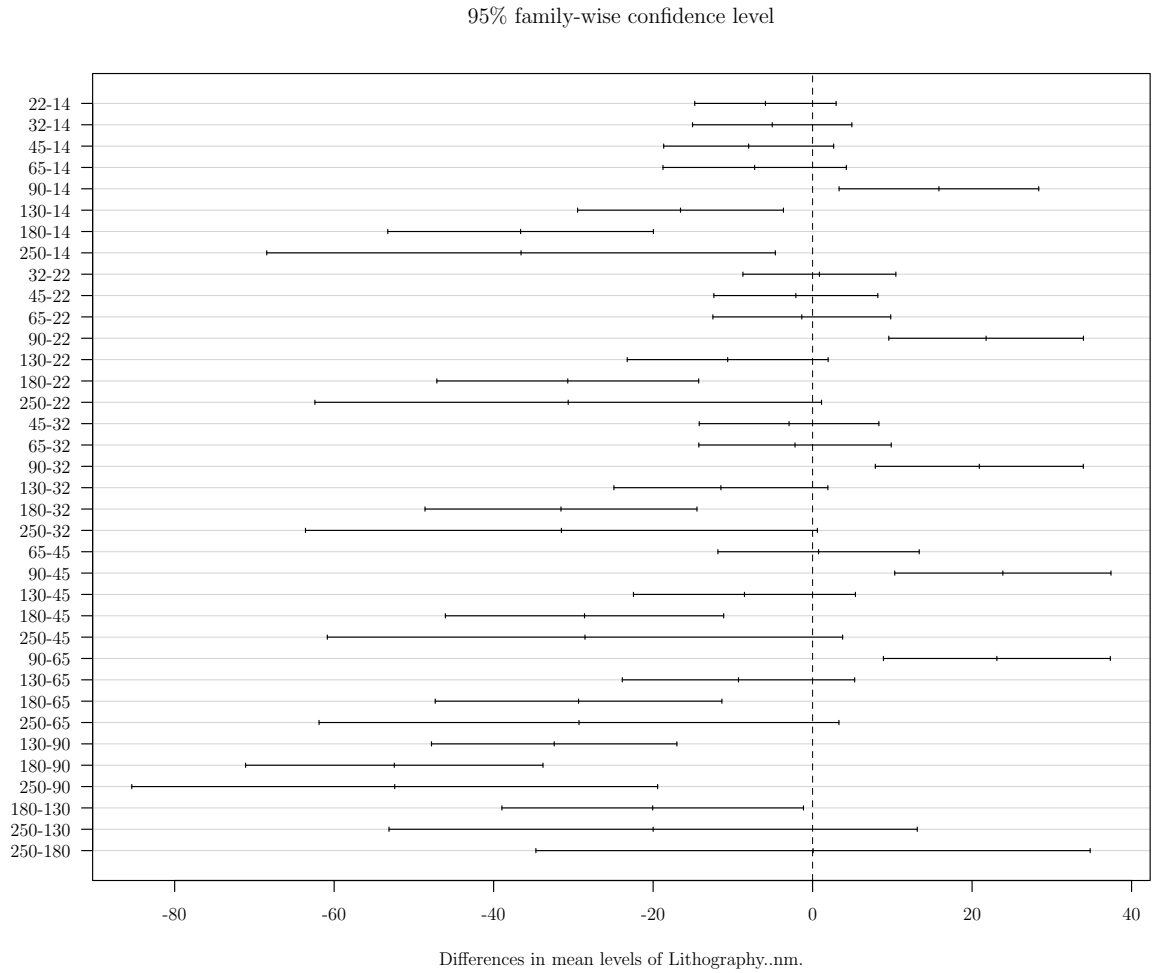


Figure 32: *Tukey's HSD confidence intervals at 95% for each pairwise comparison*

In figure 32, each pairwise comparison is represented by a horizontal line indicating the confidence interval for the difference in mean TDP. If the interval crosses zero, it suggests no significant difference between the two levels. For instance, comparisons like $45 - 32$, $65 - 45$, and $250 - 180$ show wide intervals that likely include zero, suggesting no significant differences in TDP Watts between these lithography levels. These intervals suggest that the TDP values for these pairs of lithography levels are not significantly different at the 95% confidence level.
On the other hand, some comparisons show intervals that do not overlap with zero, indicating statistically significant differences in TDP Watts between those lithography levels. For example, the comparisons between $250 - 14$, $180 - 22$, and $250 - 22$ show intervals that do not include zero, indicating that there are significant differences in TDP between these specific pairs of lithography levels.

5.1.4 Summary

A common trend among the pairs that show statistically significant differences is that the larger lithography levels (e.g., 250, 180, 130) tend to have significant differences when compared to smaller lithography levels (e.g., 14, 22). Specifically, comparisons such as $90 - 14$, $130 - 14$, $180 - 14$, and $250 - 14$ reveal significant differences in TDP. While pairs like $32 - 22$, $45 - 22$, and $65 - 22$ show no significant differences, or we can conclude that close lithography levels are relatively similar on TDP.

5.2 Linear Regression

5.2.1 Objective

In our model, we have

$$Y = \widehat{\beta_0} + \sum_{i=1}^{n} \widehat{\beta}_i X_i + \varepsilon$$

where:

- Y is the predicted TDP value.
- X_i, where $i = \{1, 2, \ldots, n\}$, are the respective values for independent variables.
- $\widehat{\beta}_i$ are the estimated coefficients for each independent variable X_i.
- ε is the error term accounting for the deviation of the observed values from the predicted values

From figure 25, we see that all other variables have certain linear relationship with TDP. In this analysis, we aim to develop a model that predicts the TDP (Watts) based on those various features of processors, let's say Launch_Date, Recommended_Customer_Price (USD), Lithography, nb_of_Cores, Processor_Base_Frequency (MHz), Cache_Size (MB), Max_Memory_Size (GB), Max_Memory_Bandwidth (GB/s) and Instruction_Set (bit).
The objective is to evaluate the relationships between these variables and the target variable, TDP. This approach allows us to understand and explain the relationship between Thermal Design Power and each attribute, developing a reliable predictive model to capture underlying data patterns.
Before perform testings, we will remove these outliers to ensure the accuracy of our model (line 429 to line 439). This ensures that the model is not skewed by extreme values, which could otherwise distort the results.

5.2.2 Assumptions Testing

In multiple linear regression, these assumptions include linearity, normality, homoscedasticity, and independence are assessed after the multiple linear regression model has been fitted to the data. All plots examined in this section are obtained from the model's residuals after fitting (line 447 to 461).

Test for Linearity assumption.

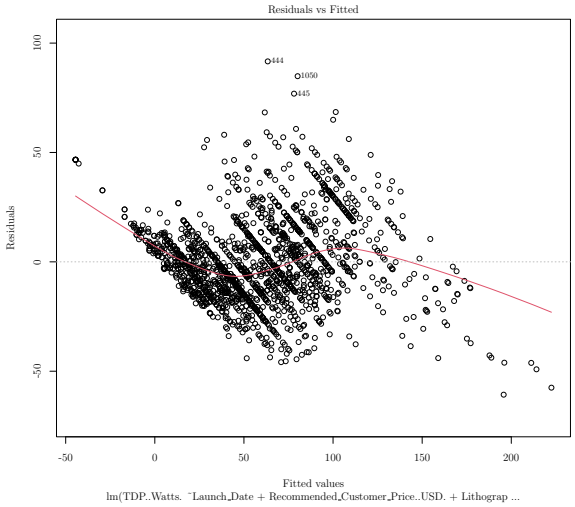


Figure 33: *Residuals vs Fitted plot for Linearity assumption*

The residuals vs. fitted plot is used to assess the linearity assumption of the model. In a well-fitted linear regression model, the residuals should be randomly scattered around the horizontal line at zero. This suggests that the assumption that the relationship is linear is reasonable.

In this plot (figure 33), the residuals do not appear to be randomly scattered around the zero line. Instead, there seems to be a distinct curvature in the pattern of the residuals. This nonlinear trend in the residuals indicates a potential violation of the linearity assumption. The curved red line, which represents a smooth fit of the residuals, further confirms that the relationship is likely not linear.

Test for Normality assumption.

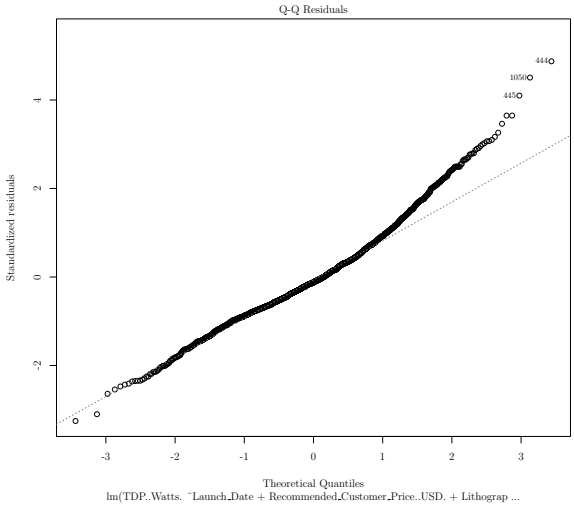


Figure 34: *Q-Q residuals plot for Normality assumption*

This is a Q-Q plot, which is used to assess whether the residuals from a model follow a normal distribution. figure 34 shows that the majority of the residuals align closely with the diagonal reference line, suggesting that the central portion of the residuals adheres reasonably well to the assumption of normality. For the upper tail, several data points exhibit upward deviations from the reference line, indicative of heavier-than-expected tails or potential high residual outliers. Meanwhile, lower tail has a minor deviation, with residuals falling below the line.

Test for Homoscedasticity assumption.

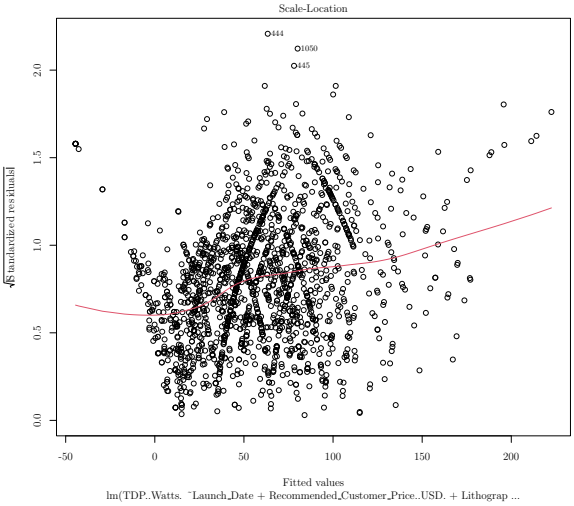


Figure 35: *Scale-Location plot for Homoscedasticity assumption*

The Scale-Location plot shows the standardized residuals plotted against the fitted values of the multiple linear regression model. This plot is used to assess the homoscedasticity assumption, which requires the variance of the residuals to be constant across all levels of the predictor variables.
Examining the plot (figure 35), we can see that the red line appears to curve upward slightly. This may indicate that the variance of residuals increases with larger fitted values, suggesting heteroscedasticity. The residuals appear to form some arcs or clusters, particularly visible in the middle range of fitted values.

Check for influential observations.

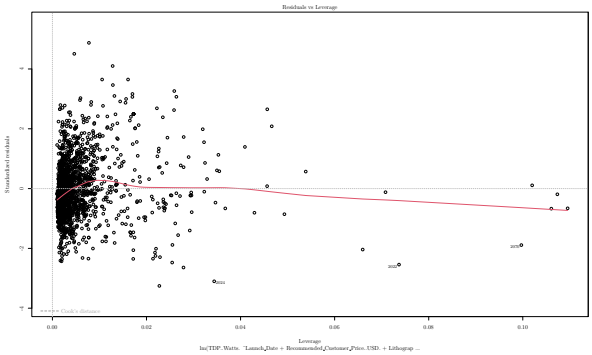


Figure 36: *Residuals vs Leverage plot for influential observations*

Influential observations in regression analysis are data points that have an outsized impact on the model's parameter estimates and overall fit. These points often arise due to their high leverage or extreme residual values. Detecting such observations is critical as they can distort the results, mislead interpretation, and compromise the generalizability of the mode.
To examine influential observations, a Residuals vs. Leverage Plot is used. This plot combines the residual information (fit discrepancies) and leverage (distance of predictors from their mean) to identify observations that are both highly influential and potentially problematic. The plot also overlays Cook's distance, a measure that combines leverage and residual information, to help detect influential data points. Points that exceed the Cook's distance threshold are considered candidates for further investigation.
In figure 36, most data points are clustered at low leverage values, indicating that the majority of observations have predictors close to the mean and do not significantly influence the model. There are a few points with high leverage that are potential candidates for further scrutiny (i.e. 2024, 2022, and 2070). Most residuals are near zero, suggesting that the majority of observations fit the model well. Besides that, no data points exceed the Cook's distance threshold, indicating that all observations are well within acceptable bounds for influence. Therefore, there are no significant outliers or influential points that could unduly affect the regression model's parameter estimates.
In the scope of this report, we stills perform the multiple linear regression model fitting to predict the TDP value based on the other attributes in the dataset despite of the assumptions violation. In further discussion, we will propose alternative method for our objective.

5.2.3 Multiple Linear Regression

As we discussed in Objective section, we aim to predict the TDP value based on the other attributes in the dataset. We will fit the model with all the attributes and then remove the insignificant ones to get the best model if necessary.

Step 1. Spliting dataset

For training and testing the model, we have settled on an allocation of 80% of our dataset for training purposes, reserving the remaining 20% for testing. This split ratio has demonstrated its efficacy in facilitating accurate predictions for our specific tasks (441 to line 45).

Step 2. Model Fitting

We fit the model using the `lm()` function in *R*, which allows us to specify the formula for the regression model. The formula is defined as `TDP..Watts. ~ .`, indicating that the TDP is the dependent variable, while all other variables in the dataset are independent variables. The model is then fitted to the training set, and the summary of the results is displayed (line 447 to 458).

```
    lm(formula = TDP..Watts. ~ Launch_Date + Recommended_Customer_Price..USD. +
    Lithography..nm. + nb_of_Cores + Processor_Base_Frequency..MHz. +
    Cache_Size..MB. + Max_Memory_Size..GB. + Max_Memory_Bandwidth..GB.s. +
    Instruction_Set..bit., data = training_set)

Residuals:
    Min      1Q  Median      3Q     Max
-60.726 -12.344  -2.274   9.972  91.644

Coefficients:
                                   Estimate Std. Error t value Pr(>|t|)
(Intercept)                       3.457e+03  3.709e+02   9.323  < 2e-16 ***
Launch_Date                      -1.760e+00  1.844e-01  -9.544  < 2e-16 ***
Recommended_Customer_Price..USD.  2.291e-03  6.829e-04   3.354 0.000813 ***
Lithography..nm.                  3.285e-01  2.026e-02  16.213  < 2e-16 ***
nb_of_Cores                       1.484e+00  2.979e-01   4.980 7.01e-07 ***
Processor_Base_Frequency..MHz.    3.099e-02  6.732e-04  46.036  < 2e-16 ***
Cache_Size..MB.                   2.426e+00  1.595e-01  15.211  < 2e-16 ***
Max_Memory_Size..GB.              3.496e-03  1.649e-03   2.120 0.034144 *
Max_Memory_Bandwidth..GB.s.      -1.503e-01  5.467e-02  -2.748 0.006054 **
Instruction_Set..bit.             5.539e-01  5.896e-02   9.394  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 18.88 on 1700 degrees of freedom
Multiple R-squared:  0.7876,    Adjusted R-squared:  0.7865
F-statistic: 700.4 on 9 and 1700 DF,  p-value: < 2.2e-16
```

Figure 37: *Fitting model summary using `lm()`*

Examined the fitted model summary (figure 37), we can obtain the following insights:
For the residuals, a relatively large range (from -60.726 to 91.644) suggests that there might be some large residuals that could indicate outliers or issues with the model. The median residual of -2.274 suggests that the model might slightly under-predict for some observations, but overall, the residuals seem reasonably spread around zero.
The p-values associated with each coefficient provide information about the significance of the relationship. In this case, all coefficients have p-values less than 0.05, indicating that they are statistically significant, so we will accept this model fitting with all attributes for further analysis.

- `Intercept`: 3457 (with a p-value $< 2 \times 10^{-16}$) - The intercept is highly significant, and the positive value suggests that when all independent variables are zero, the TDP is expected to be around 3457 watts, however, this situation cannot happen in real life.
- `Launch_Date`: -1.760 (p-value $< 2 \times 10^{-16}$) - A negative relationship is indicated between Launch_Date and TDP, meaning that as the launch date increases, the TDP tends to decrease. This effect is highly significant.
- `Recommended_Customer_Price..USD.`: 0.0023 (p-value = 0.000813) - A small but significant positive relationship between customer price and TDP, meaning that higher customer prices are associated with higher TDP values. This inference seems reasonable, as more expensive processors may have higher performance and power requirements, let's CPU for gaming, training AI purposes.
- `Lithography..nm.`: 0.3285 (p-value $< 2 \times 10^{-16}$) - Lithography size has a significant positive effect on TDP.

- `nb_of_Cores`: 1.484 (p-value $= 7.01 \times 10^{-7}$) - The number of cores has a strong positive effect on TDP, meaning that more cores generally lead to higher TDP values.
- `Processor_Base_Frequency..MHz.`: 0.03099 (p-value $< 2 \times 10^{-16}$) - A positive effect of base processor frequency on TDP, suggesting that higher base frequency results in higher TDP values.
- `Cache_Size..MB.`: 2.426 (p-value $< 2 \times 10^{-16}$) - Cache size has a strong positive relationship with TDP, meaning larger cache sizes tend to increase TDP.
- `Max_Memory_Size..GB.`: 0.0035 (p-value $= 0.034144$) - This variable has a small but significant positive effect on TDP.
- `Max_Memory_Bandwidth..GB.s.`: -0.1503 (p-value $= 0.006054$) - Max memory bandwidth has a small but significant negative effect on TDP, meaning that as bandwidth increases, TDP tends to decrease.
- `Instruction_Set..bit.`: 0.5539 (p-value $< 2 \times 10^{-16}$) - The instruction set bit significantly positively affects TDP, suggesting that larger instruction sets tend to increase TDP.

The model's R^2 value is 0.7876, indicating that the model explains 78.76% of the variance in the TDP values. The adjusted R^2 value is 0.7865, which is slightly lower but still indicates a good fit. The F-statistic is 700.4 with a very low p-value, suggesting that the model is statistically significant. The residual standard error is 18.88, which represents the average difference between the observed and predicted TDP values.
After applying fitting, we have our final multiple linear regression model as concluded. The regression coefficients for each independent variable in this model are as follows in figure 38.

```
TDP = 3457 - 1.760 * Launch_Date + 0.0023 * Recommended_Customer_Price..USD. +
    0.3285 * Lithography..nm. + 1.484 * nb_of_Cores + 0.03099 *
    Processor_Base_Frequency..MHz. + 2.426 * Cache_Size..MB. + 0.0035 *
    Max_Memory_Size..GB. - 0.1503 * Max_Memory_Bandwidth..GB.s. + 0.5539 *
    Instruction_Set..bit.
```

Figure 38: *Final multiple linear regression model*

Step 3. Model Testing

To evaluate the model's predictive performance, we use the testing set to make predictions and compare them with the actual TDP values. We then calculate the accuracy of the model by comparing the predicted values with the actual values. The mainly used accuracy is calculated as the mean square error (MSE) between the predicted and actual TDP values (line 463 to line 481).

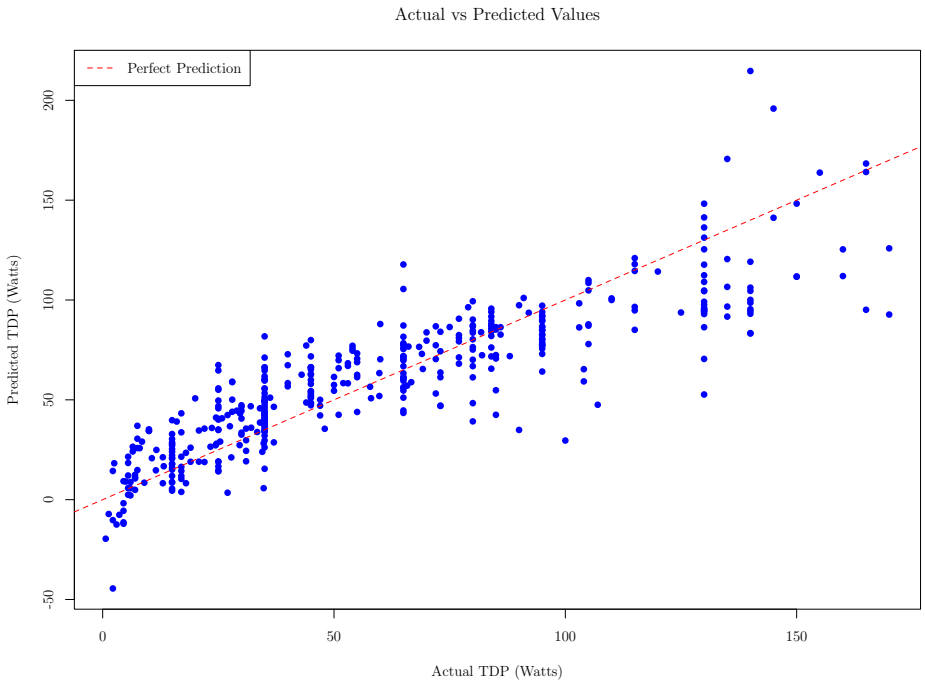


Figure 39: *Predicted vs. Actual TDP values for the testing set in multiple linear regression*

The plot in figure 39 shows the predicted TDP values against the actual TDP values for the testing set. The points are scattered throughout the plot, showing a general positive relationship between the actual and predicted TDP values. However, the scatter is quite wide, indicating that the model's predictions do not perfectly match the actual observations.

```
MAE: 15.03
MSE: 415.88
RMSE: 20.39
```

Figure 40: *Model evaluation metrics for the testing set*

In figure 40, the MAE represents the average magnitude of errors between the predicted and actual TDP values. A value of 15.03 watts suggests that, on average, the model's predictions deviate by approximately 15 watts from the actual TDP values. This gives us a sense of the model's average prediction accuracy, with a lower MAE indicating better performance, which is a quite high error compared to practical TDP values. MSE measures the average squared differences between the predicted and actual values. The relatively high MSE value of 415.88 reflects the presence of larger errors in some of the predictions. Since MSE penalizes larger errors more than smaller ones, the value indicates that some predictions may have significant deviations from the actual TDP values. RMSE is the square root of MSE, bringing the error back to the original units (watts). With an RMSE of 20.39 watts, the model's typical prediction error is about 20.39 watts.

5.2.4 Summary

In this section, we developed a multiple linear regression model to predict the Thermal Design Power based on various processor features, such as Launch Date, Recommended Customer Price, Lithography, Number of Cores, Base Frequency, Cache Size, Memory Size, Memory Bandwidth, and Instruction Set. We first explored the relationships between these variables and TDP, testing for linearity, normality, homoscedasticity, and influential observations. Several assumption violations were noted, including potential non-linearity and heteroscedasticity. Despite these violations, the model was still fitted, and significant variables were identified through their p-values.

The model fitting was performed on a training dataset (80% of the data) with the `lm()` function in R, resulting in an R^2 value of 0.7876, meaning the model explains 78.76% of the variance in TDP values. The final regression equation was derived, which included significant coefficients for each predictor variable.
Model testing on a separate testing set yielded evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The results indicated moderate prediction accuracy, with an MAE of 15.03 watts and an RMSE of 20.39 watts, suggesting the model's predictions typically deviate by around 15-20 watts from the actual TDP values. Despite these errors, the model provides useful insights for predicting TDP based on processor features, although further improvements might be necessary to refine its accuracy.

6 Discussion and Extension

6.1 Discussion

6.1.1 ANOVA

ANOVA is a powerful statistical technique that enables us to compare the means of multiple groups simultaneously. By examining the variability within and between groups, we can determine whether there are significant differences in the means. This allows us to identify the factors that influence the dependent variable and assess their impact on the overall variance.
However, ANOVA has several limitations. One of the main drawbacks is its assumptions. For ANOVA to be valid, the data must be normally distributed, and the variances of the groups should be equal. Violating these assumptions can lead to inaccurate results and misleading conclusions. Additionally, ANOVA is sensitive to outliers, which can skew the results and affect the validity of the analysis.

6.1.2 Multiple Linear Regression

One of the main benefits of linear regression models is their simplicity and the linear relationship they establish between variables, making the estimation process straightforward. Moreover, the mathematical equation behind linear regression is relatively easy to interpret.
However, underfitting can occur when a model fails to capture the underlying patterns in the data. This is often due to the hypothesis function being unable to adequately fit the data. In many real-world scenarios, the relationship between variables is not linear, meaning a straight line cannot effectively represent the data. In these cases, more complex functions are needed to better capture the relationships. As a result, linear regression models often suffer from low accuracy. Additionally, outliers can severely affect model performance, leading to decreased accuracy.

6.2 Extension

6.2.1 Non-parameter test

Due to the violation of assumptions when working with ANOVA, we use non-parametric tests as an alternative. Non-parametric tests do not require the data to be normally distributed or have equal variances, making them more robust to violations of these assumptions. One common non-parametric test is the Kruskal-Wallis test, which is used to compare the medians of two or more groups. The Kruskal-Wallis test is based on the ranks of the data values and is less sensitive to outliers compared to ANOVA. One common non-parametric post-hoc test is the Dunn test, which compares multiple groups to identify significant differences between them.
From line 491 to line 493 in the code, we perform the Kruskal-Wallis test to compare the means of the groups.

```
Kruskal-Wallis rank sum test

data:  TDP..Watts. by Lithography..nm.
Kruskal-Wallis chi-squared = 118.8, df = 8, p-value < 2.2e-16
```

Figure 41: *Kruskal-Wallis test*

From figure 41, we have a Kruskal-Wallis chi-squared value of 118.8 and a very small p-value, we reject the null hypothesis that the distributions of TDP are identical across the different lithography categories. This suggests that lithography size has a significant impact on the TDP values, and Dunn's test can be performed to understand the specific pairwise differences between the lithography groups (line 495 to line 497).

```
Dunn (1964) Kruskal-Wallis multiple comparison
p-values adjusted with the Bonferroni method.

Comparison            Z      P.unadj        P.adj
1    130 - 14 -2.35134843 1.870551e-02 6.733983e-01
2   130 - 180  4.20734971 2.583831e-05 9.301793e-04
3    14 - 180  6.59836253 4.157237e-11 1.496605e-09
4    130 - 22 -1.87958051 6.016527e-02 1.000000e+00
5     14 - 22  0.75203251 4.520315e-01 1.000000e+00
6    180 - 22 -6.28768313 3.222388e-10 1.160060e-08
7   130 - 250  2.20912354 2.716605e-02 9.779778e-01
8    14 - 250  3.24609683 1.169990e-03 4.211965e-02
9   180 - 250 -0.18367254 8.542704e-01 1.000000e+00
10   22 - 250  3.04881590 2.297452e-03 8.270828e-02
11   130 - 32 -2.43703158 1.480839e-02 5.331019e-01
12    14 - 32 -0.23443842 8.146446e-01 1.000000e+00
13   180 - 32 -6.58244281 4.627808e-11 1.666011e-09
14    22 - 32 -0.93953827 3.474545e-01 1.000000e+00
15   250 - 32 -3.29796172 9.738941e-04 3.506019e-02
16   130 - 45 -1.56422915 1.177638e-01 1.000000e+00
17    14 - 45  0.80483034 4.209176e-01 1.000000e+00
18   180 - 45 -5.80621801 6.389975e-09 2.300391e-07
19    22 - 45  0.18573978 8.526488e-01 1.000000e+00
20   250 - 45 -2.93821733 3.301055e-03 1.188380e-01
21    32 - 45  0.96921306 3.324389e-01 1.000000e+00
22   130 - 65 -1.65692379 9.753488e-02 1.000000e+00
23    14 - 65  0.53973485 5.893799e-01 1.000000e+00
24   180 - 65 -5.76857725 7.994356e-09 2.877968e-07
25    22 - 65 -0.04144994 9.669372e-01 1.000000e+00
26   250 - 65 -2.98515059 2.834387e-03 1.020379e-01
27    32 - 65  0.70852905 4.786168e-01 1.000000e+00
28    45 - 65 -0.18793498 8.509276e-01 1.000000e+00
29   130 - 90 -6.77505166 1.243618e-11 4.477025e-10
30    14 - 90 -5.90449446 3.537298e-09 1.273427e-07
31   180 - 90 -9.85816660 6.319269e-23 2.274937e-21
32    22 - 90 -6.60462936 3.985131e-11 1.434647e-09
33   250 - 90 -5.38122860 7.397917e-08 2.663250e-06
34    32 - 90 -5.48841993 4.055450e-08 1.459962e-06
35    45 - 90 -6.08421879 1.170607e-09 4.214184e-08
36    65 - 90 -5.63127224 1.788851e-08 6.439863e-07
```

Figure 42: *Dunn's test results for pairwise comparisons*

From figure 31 and figure 42, we observe that most of the pairs show the same conclusion, indicating that they either do or do not differ significantly in terms of the means of TDP. However, two pairs, namely $130 - 14$ and $250 - 30$, exhibit different results between the Tukey HSD test and the Dunn's test. This discrepancy could be caused by several factors:

- *Different Correction Methods:* The Tukey HSD test uses a different correction method for multiple comparisons compared to Dunn's test. While Tukey's test controls the family-wise error rate using a more conservative approach, Dunn's test applies the Bonferroni correction (in this project). The difference in how these methods handle multiple comparisons could lead to variations in significance levels.
- *Data Distribution and Outliers:* The presence of outliers or skewed data in specific pairs, such as

$130 - 14$ and $250 - 30$, could disproportionately affect the test results. If one of the groups in these pairs has extreme values, it may distort the conclusions drawn by the tests, especially with Dunn's test, which is more sensitive to data rankings.

- *Sample Size:* The power of both tests is influenced by the sample size in each group. If the sample sizes differ significantly between the groups in these pairs, one test might be more influenced by the larger sample sizes, leading to differing conclusions.

In conclusion, the discrepancies observed between the Tukey HSD and Dunn's test results are minimal. This suggests that, despite differences in the assumptions for two approacj and the methods used for multiple comparisons, both tests consistently identify the same significant differences and similarities between the groups, the results of the two tests are generally consistent.

6.2.2 XGBoost - Machine Learning Models for Regression

When testing assumptions for linear regression, the variances of residuals are not similar and not normally distributed. In this case, we can use machine learning models to predict the dependent variable. Machine learning models are more flexible and can capture complex relationships between variables. In this report, we choose XGBoost as an alternative to linear regression.

- *Handling Non-linearity:* Unlike linear regression, XGBoost does not assume a linear relationship between the features and the target variable. It can model intricate interactions and non-linear patterns in the data, making it more suitable for real-world datasets where such relationships are common.
- *Robustness to Outliers:* XGBoost is less sensitive to outliers compared to linear regression, which can be significantly influenced by extreme values. The decision tree-based structure of XGBoost allows it to handle such anomalies more effectively, reducing their impact on model performance.

From line 500 to 516, we import data and split it into training and testing sets, as same as linear regression model. Then, from line 518 to line 539, we convert numeric data to matrices for model training since XGBoost requires matrix inputs. After that, from line 541 to line 550, the XGBoost model was trained using the same set of independent variables and dataset split as the linear regression model, ensuring a consistent comparison of the two approaches. The model's performance was evaluated using standard metrics such as MAE, MSE, and RMSE. After training the XGBoost model, we analyzed the feature importance to identify the predictors that contributed most to the model's predictive accuracy.

```
                            Feature       Gain      Cover  Frequency
                            <char>      <num>      <num>      <num>
1:                  Cache_Size..MB. 0.29783850 0.14777704 0.12790425
2:   Processor_Base_Frequency..MHz. 0.25560539 0.19887352 0.19831026
3: Recommended_Customer_Price..USD. 0.23551681 0.21990567 0.24806383
4:                  Lithography..nm. 0.07210470 0.10293799 0.08378315
5:                       nb_of_Cores 0.05136875 0.08446524 0.07111007
6:                       Launch_Date 0.03382545 0.09199989 0.13893452
7:              Max_Memory_Size..GB. 0.03161580 0.09323008 0.07134475
8:      Max_Memory_Bandwidth..GB.s. 0.01140543 0.04224005 0.04881483
9:             Instruction_Set..bit. 0.01071916 0.01857052 0.01173433
```

Figure 43: *XGBoost model summary*

The feature importance analysis (the code is from line 573 to line 575) in figure 43 and figure 44 provides insights into which predictors contributed most to the model's predictive accuracy. The top features, ranked by their gain (the improvement in model performance brought by a feature), include:

- Cache Size (MB): With the highest gain (0.2978), this feature significantly impacts the model's predictions.
- Processor Base Frequency (MHz): This feature has a gain of 0.2556, indicating its strong influence on the target variable.
- Recommended Customer Price (USD): Contributing a gain of 0.2355, this feature also plays a critical role in the model.

Other features, such as lithography (nm), the number of cores, and launch date, also exhibit notable contributions but to a lesser extent.

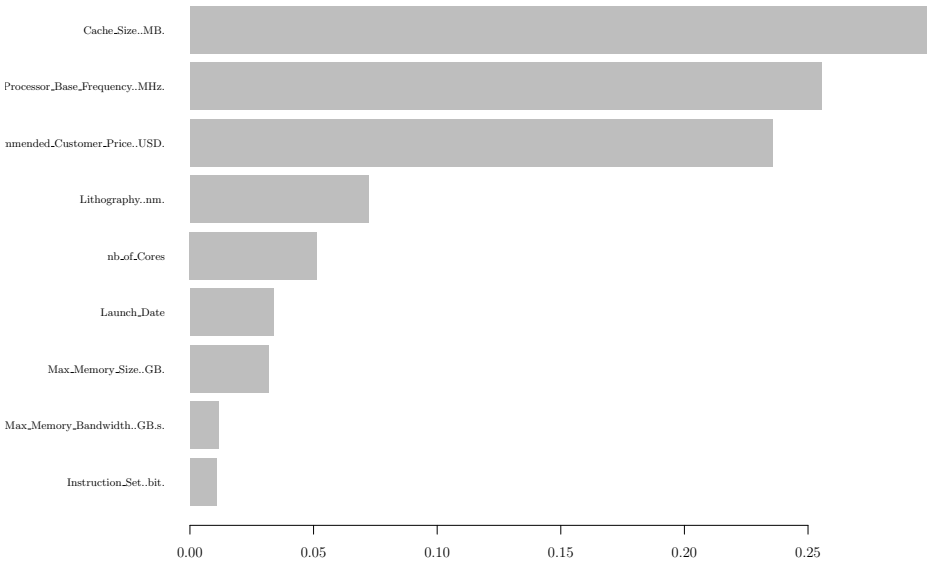


Figure 44: *Feature importance plot*

Now we can use the trained XGBoost model to make predictions on the testing set and evaluate its performance (from line 552 to line 570).

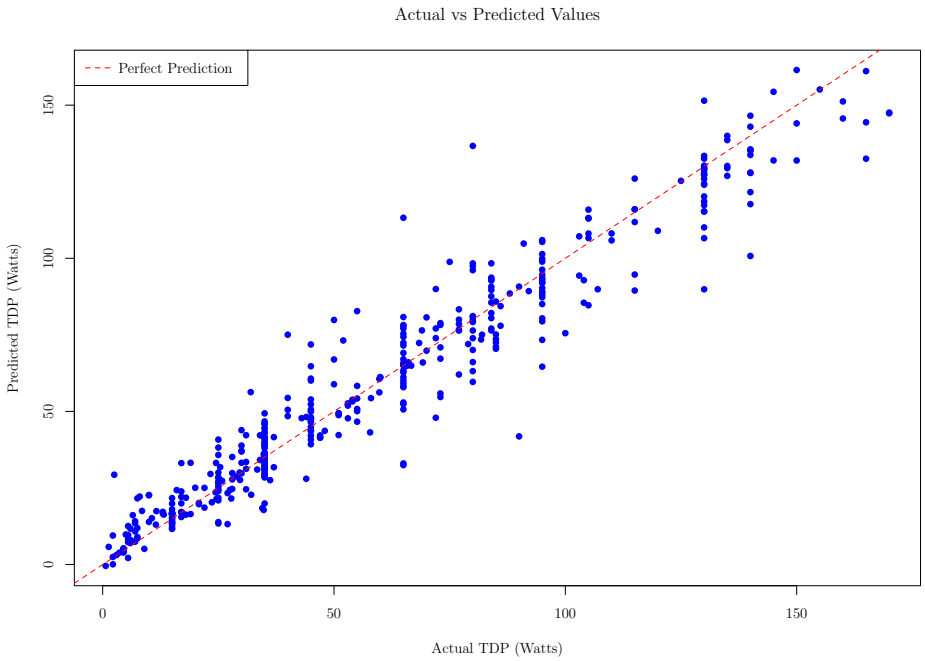


Figure 45: *Predicted vs. Actual TDP values for the testing set in the XGBoost model*

In figure 39 and figure 45, we observe that the predicted values of the XGBoost model are more closely aligned with the red line compared to those of the linear regression model. This indicates that the

XGBoost model produces predictions with smaller errors than the linear regression model. To substantiate this observation, we further examined the MAE, MSE, and RMSE, as follows:

```
MAE: 7.22
MSE: 115.67
RMSE: 10.76
```

Figure 46: *XGBoost model performance*

From figure 40 and figure 46, we can see that the XGBoost model outperforms the multiple linear regression model in terms of prediction. The XGBoost model achieved a lower MAE of 7.22, compared to the linear regression model's MAE of 15.03. Similarly, the XGBoost model exhibited a lower MSE of 115.67 and RMSE of 10.76.
This suggests that for datasets with non-linear relationships or complex feature interactions, extending beyond multilinear regression to advanced machine learning models like XGBoost can provide significant performance gains. Therefore, XGBoost is a better choice for this task due to its enhanced flexibility and predictive accuracy.

7 Data and Code Availibility

Data used in this report can be found on Kaggle.
The code implemented for this report can be found on GitHub.

References

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[2] Tianqi Chen, Tong He, Michael Benesty, and Vadim Khotilovich. Package 'xgboost'. *R version*, 90(1-66):40, 2019.

[3] Douglas C Montgomery and George C Runger. *Applied statistics and probability for engineers*. John wiley & sons, 2020.

[4] William N Venables, David M Smith, et al. An introduction to r: notes on r: a programming environment for data analysis and graphics, version 4.4.2. 2024.