VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Probability and Statistics (MT2013)

Assignment

The impact of CPU's characteristics on its Thermal Design Power

Class:	CC03	
Group:	10	
Advisor:	Phan Thị Hường	
Students:	Nguyễn Quang Phú	- 2252621
	Nguyễn Xuân Sơn	- 2252717
	Phạm Duy Tường Phước	- 2252280
	Trần Xuân Bách	- 2252058
	Phạm Đoàn Bảo Trung	- 2252858

HO CHI MINH CITY, May 2024

Contents

Member list & Workload

No.	Name	ID	Workload	Percentage
1	Nguyễn Quang Phú	2252621	Undertake data cleaning, compose a statement for descriptive statistics, and assume responsibility for conducting regression analysis.	100%
2	Nguyễn Xuân Sơn	2252717	Produce visualizations using the refined dataset and offer analytical insights. Make researching and implement the code on regression analysis.	100%
3	Phạm Duy Tường Phước	2252662	Conduct research, elucidate, and draft a report on fundamental theoretical concepts. Review content for specialized expertise.	100%
4	Phạm Đoàn Bảo Trung	2252858	Conduct joint research with Bách on XGBoost model theory and its practical implementation. Consolidate findings into a comprehensive report.	100%
5	Trần Xuân Bách	2252058	Responsible for finding extended statistical model and writing theoretical statement. Review content for specialized expertise.	100%

1 Data Introduction

In the realm of computer hardware, Thermal Design Power (TDP) stands as a pivotal metric, defining the maximum heat output of Central Processing Units (CPUs) under typical operating conditions. This research investigates the nuanced interactions between key CPU specifications – lithography, number of cores, launch date, processor frequency, graphic base frequency, and their impact on TDP.

Through empirical analysis and statistical modeling, this research aims to illuminate the intricate dynamics shaping CPU power specifications. Collecting information from documents, papers and the Internet, TDP is said to have close relationship with lithography, number of cores, launch date, processor base frequency and level of price. By providing insights into the relationship between lithography, number of cores, launch date, processor frequency, graphic base frequency, and TDP, this study seeks to empower system architects, hardware enthusiasts, and industry professionals in making informed decisions regarding system design and configuration. Understanding these factors is crucial for optimizing power management strategies and maximizing the efficiency of computing systems in diverse usage scenarios.

Regarding the variables used for analysis, we are given the dataset obtained from Computer Parts (CPUs and GPUs) by author Ilissek.

Some essentials attributes of the dataset are:

1. Vertical Segment (or Market Segment): [13]
- Data type: Categorical
- Unit: None
- Description: Describes the specific market for which a CPU is designed. The four market categories are: Mobile, Desktop, Embedded, and Server.

2. Status: [14]
- Data type: Categorical
- Unit: None
- Description: The status of CPU supply. It can be: Announce (coming soon), Launched (available and supported), End of Life (no longer made but supported), and End of Interactive Support (no longer supported).

3. Launch date: [10]
- Data type: Categorical
- Unit: None
- Description: The date that the CPU was available on the market. In format of: Quarter-Year (Example: $Q1'24$).

4. Lithography: [11]
- Data type: Categorical
- Unit: Nanometer (nm)
- Description: The measurement of the size of transistors is a useful metric for judging how powerful a CPU is.

5. Recommended Customer Price: [6]
- Data type: Categorical
- Unit: Dollar ($)
- Description: The price Intel suggests retailers sell their CPU products for.

6. Number of Cores: [5]
- Data type: Categorical
- Unit: None
- Description: The number of processing units on one CPU.

7. Processor Base Frequency: [1]
- Data type: Continuous
- Unit: GHz/MHz
- Description: The expected Clock Rate of a CPU. The larger the frequency means the faster the clock rate and the better performance.

8. Temperature: [1]
- Data type: Continuous
- Unit: Celsius degree
- Description: The maximum temperature allowed on the CPU.

9. Graphic Base Frequency: [12]
- Data type: Continuous
- Unit: MHz
- Description: The guaranteed graphics render clock frequency.

Some key factors of the dataset:
- Population: CPUs from Intel
- Observation: 2283 product collections produced by Intel
- Feature: 45 variables

2 Background

2.1 Regression Model

2.1.1 Multiple Linear Regression Model

Regression models are used to describe relationships between variables by fitting a line to the observed data. Regression allows researchers to estimate how a dependent variable changes as the independent variables change.

Multiple linear regression is used to estimate the relationship between two or more independent variables and one dependent variable. Multiple linear regression can help researchers when they want to know:

- How strong the relationship is between two or more independent variables and one independent variable.
- The value of the dependent variable at a certain value of the independent variables.

The formula of multiple linear regression:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon$$

Where:
- y is the dependent variable (respond variable)
- β_0 is the intercept
- β_i with i = 1, ... , n are regression coefficients
- x_i with i = 1, ... , n are independent variables (explanatory variables)
- ϵ is the model's error (known as residuals)

Some assumptions can be made before performing multiple linear regression model:

- ***Linearity:*** The independent and dependent variables have a **linear relationship** with one another. This implies that changes in dependent variables follow those in independent variables in a linear fashion. This means that there will be a "straight line" that can be drawn through the data points.

- ***Homoscedasticity:*** The **variance of the residuals** should be **consistent** across the linear model. Plotting standardized residuals against predicted values helps to check this assumption. A scatterplot or statistical software can be used for this test.

- ***Multivariate Normality:*** **Residuals** should follow a **normal distribution**. This can be visually checked using histograms with superimposed normal curves or with Normal Probability Plot methods.

- ***Independence of Observations:*** Observations should be independent, meaning the residuals should not have autocorrelation. The **Durbin Watson statistic** is used to test this, with values between 0 and 2 indicating positive autocorrelation, between 2 and 4 indicating negative autocorrelation, and a value of 2 indicating no autocorrelation.

- ***No Multicollinearity:*** Independent variables should not be highly correlated, which can lead to multicollinearity. The **Variance Inflation Factor** method is commonly used for this.

2.1.2 XGBOOST (eXtreme Gradient Boosting)

2.1.2.a Boosting

Boosting is an algorithm that helps in **reducing variance and bias** in a machine learning ensemble. The algorithm helps in the conversion of weak learners into strong learners by combining N number of learners. Boosting also can improve model predictions for learning algorithms. The weak learners are sequentially corrected by their predecessors, and, in the process, they are converted into strong learners.

2.1.2.b Gradient Boost

Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which **each new model is trained to minimize the loss function** such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.

Gradient Boost starts by making a single leaf, this leaf represents of an initial guess for the weights of all of the sample. When trying to predict a continuous value, the first guess is the average. Then gradient boost creates the tree which is based on the errors made by the previous tree, with the maximum number of leaves to be between 8 and 32.

Gradient boost scales all tree by the same amount, then it builds another tree based on the errors made by the previous one. After that, it continues to scale the tree and continues to build trees in this fashion until it has made the number of trees that meet our limit or it has made a tree that fail to improve the fit (fail to minimize the loss function).

2.1.2.c Equation

***Input*:** Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable Loss Function $L(y_i, F(x)) = \frac{1}{2}(observed{-}predicted)^2$

Step 1: Initialize model with a constant value: $F_0(x) = argmin\sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: For m = 1 to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} = (observed - F_{m-1}(x))$, for $i = 1, \ldots, n$

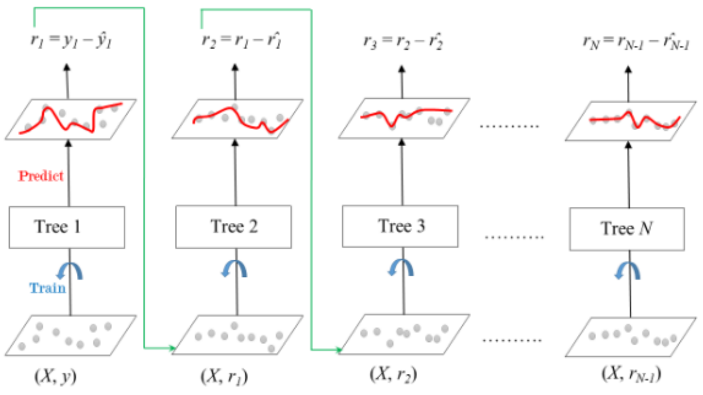
(B) Fit a regression tree to the rim values and create terminal regions R_{jm}, for $j = 1, \ldots, J_m$

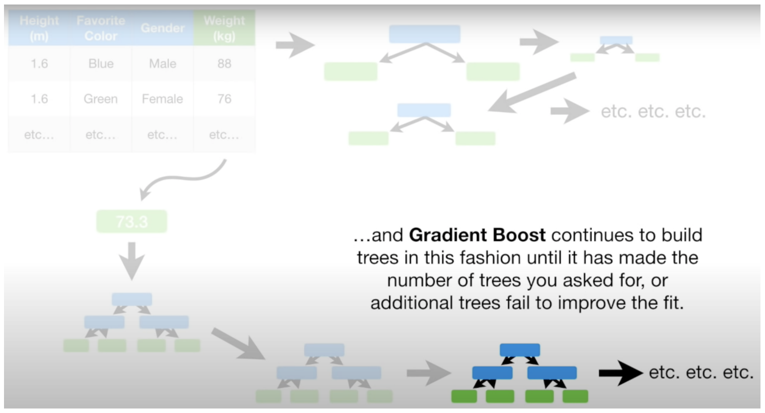
(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = argmin \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update the new Prediction $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

Step 3: Output the $F_M(x)$

Where:
- y_i refer to the data that we need to predict for each in the dataset.
- x_i refer to each row of measurements that we will use to predict
- $F(x)$ refer to the function to get predicted value.
- γ refer to the **predicted value**
- γ_{jm} refer to the **output value**
- m refer to the **current tree**
- M refer to the **last tree**
- r_{im}: r is short for **Residual** (Pseudo Residual), i is sample number, m is tree we build.
- R_{jm} refer to the **terminal region** or the **leaf** j^{th} of the tree m^{th}
- ν is greek value which is the **learning rate** and has the value between 0 and 1





Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
etc...	etc...	etc...	etc...

…and **Gradient Boost** continues to build trees in this fashion until it has made the number of trees you asked for, or additional trees fail to improve the fit.

Image 2.1: Images of Gradient Boosting Algorithm

2.1.2.d XGBOOST

XGBoost (extreme Gradient Boosting) is an improved version of Gradient Boosting. Its outstanding advantages are proven in the following aspects:

- ***Overfitting*:** XGBoost applies the Regularization mechanism so it significantly limits the Overfitting phenomenon (Gradient Boosting Method - GBM does not have regularization)

- ***Processing speed:*** XGBoost performs parallel calculations so processing speed can increase 10 times compared to GBM. In addition, XGboost also supports computing on Hadoop.

- ***Flexibility:*** XGboost allows users to use their own optimization functions and evaluation criteria, not limited to the provided functions.

- ***Handling missing values:*** XGBoost includes an automatic missing value handling mechanism within it. Therefore, this step can be skipped when preparing data for XGBoost.

- **Automatic trimming:** The tree pruning feature supports automatically skipping leaves and nodes that do not carry positive value during the tree expansion process.

2.2 Statistic measurement

2.2.1 The Q-Q plot

Quantile-Quantile (Q-Q) plots serve as a fundamental tool in statistical analysis, aiding researchers and practitioners in assessing the distributional similarity between two datasets or comparing a dataset to a theoretical distribution. Originating from the field of probability theory and statistics, Q-Q plots provide a visual means to evaluate **whether two datasets follow the same distribution or if one dataset conforms to a specific theoretical distribution**

From the reference of the University of Virginia [4], Q-Q plot is defined as follow:

"A QQ plot is a scatterplot created by plotting two sets of quantiles against one another. If both sets of quantiles came from the same distribution, we should see the points forming a line that's roughly straight."

To sum up, the interpretation of Q-Q plots relies on the principle that if two datasets come from the same distribution, their Q-Q plot should resemble a diagonal line. Any deviations from this line indicate differences in distributional characteristics between the datasets under comparison.

2.2.2 R-squared (R^2)

R-squared (R^2) is a statistical measure that serves as a fundamental tool in regression analysis, providing insights into the **goodness of fit of a regression model.**

Formula for R-Squared:

$R^2 = 1 - \frac{UnexplainedVariation}{TotalVariation}$

The concept behind R-squared revolves around comparing the variation of the dependent variable in the regression model to the total variation of the independent variable. Specifically, R-squared quantifies the proportion of the total variation in the dependent variable that is accounted for by the variation in the independent variables included in the regression model. R-squared values range from 0 to 1, with higher values indicating a better fit

2.2.3 P-value

In the realm of statistical hypothesis testing, the p-value stands as a pivotal concept, serving as a **measure of the strength of evidence against the null hypothesis.** Rooted deeply in statistical theory and inference, the p-value provides a quantitative means to determine the significance of observed data in relation to a specific hypothesis.

The concept behind the p-value revolves around assessing the probability of obtaining the observed results, or more extreme results, under the assumption that the null hypothesis is true. In other words, it quantifies the likelihood of observing the data if the null hypothesis were correct. A low p-value indicates that the observed data is unlikely to have occurred by random chance alone, thereby suggesting evidence against the null hypothesis.

Typically, in hypothesis testing, a significance level (often denoted as α) is predetermined, representing the threshold below which the null hypothesis is rejected. **If the p-value falls below this significance level**, it is considered statistically significant, **leading to the rejection of the null hypothesis** in favor of the alternative hypothesis. Conversely, **if the p-value exceeds the significance level, the null hypothesis is not rejected**, suggesting insufficient evidence to support the alternative hypothesis.

2.2.4 Variance Inflation Factor (VIF)

This is a **measure of the amount of multicollinearity in regression analysis**. Multicollinearity exists when there is a correlation between multiple independent variables in a multiple regression model. This can adversely affect the regression results. Therefore, the variance inflation factor can estimate how much the variance of a regression coefficient is inflated due to multicollinearity.

The formula and Calculation of VIF:
$VIF_i = \frac{1}{1-R_i^2}$
where:
R_i^2 = Unadjusted coefficient of determination for regressing the i^{th} independent variable on the remaining ones. In general terms,
VIF equal to 1 = variables are not correlated
VIF between 1 and 5 = variables are moderately correlated
VIF greater than 5 = variables are highly correlated

2.2.5 Durbin-Watson Test

One of the main assumptions in linear regression is that there is no correlation between consecutive residuals or **residuals are independent**. One way to determine this is to use the Durbin-Watson Test, which is used to detect the presence of autocorrelation in the residuals of a regression.

The Durbin-Watson test uses the following hypotheses:
$H0$ ***(null hypothesis)***: There is no correlation among the residuals.
$H1$ ***(alternative hypothesis)***: The residuals are autocorrelated.

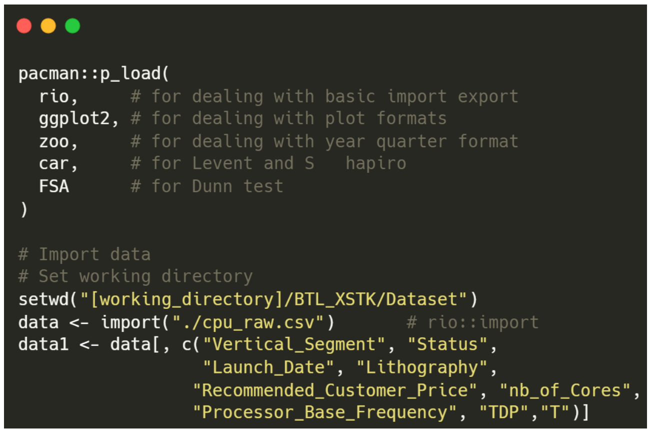
The test statistic for the Durbin-Watson test, typically denoted d, is calculated as follows:
$d = \frac{\sum_{t=2}^{T}(e_t - e_{t-1})^2}{\sum_{t=1}^{T} e_t^2}$
where:
- T: The total number of observations
- e_t: The t^{th} residual from the regression model

The test statistic always ranges from 0 to 4 where:
- $d = 2$ indicates no autocorrelation
- $d < 2$ indicates positive serial correlation
- $d > 2$ indicates negative serial correlation

In general, if d is less than 1.5 or greater than 2.5 then there is potentially a serious autocorrelation problem. Otherwise, if d is between 1.5 and 2.5 then autocorrelation is likely not a cause for concern.

3 Data Description

3.1 Importing and Reprocessing



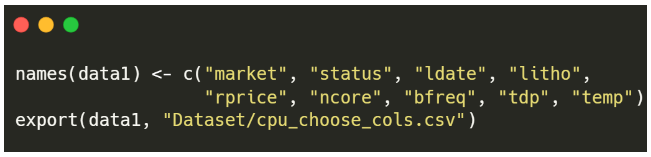
```
pacman::p_load(
  rio,      # for dealing with basic import export
  ggplot2, # for dealing with plot formats
  zoo,      # for dealing with year quarter format
  car,      # for Levent and S   hapiro
  FSA       # for Dunn test
)

# Import data
# Set working directory
setwd("[working_directory]/BTL_XSTK/Dataset")
data <- import("./cpu_raw.csv")       # rio::import
data1 <- data[, c("Vertical_Segment", "Status",
                  "Launch_Date", "Lithography",
                 "Recommended_Customer_Price", "nb_of_Cores",
                 "Processor_Base_Frequency", "TDP","T")]
```

Image 3.1: Code for importing, loading, choosing and preprocessing

- **rio**: Simplifies input and output tasks with an intuitive interface, making dataset importing and exporting safer and more convenient. It offers versatility by seamlessly handling various file formats, eliminating the need to alter commands for each format change.
- **zoo**: Facilitates the conversion of non-standard date formats, such as the Launch date in our dataset, into standard year-quarter format. Additionally, it provides useful functionalities like plotting and computing differences within these standardized formats.
- **ggplot2**: Renowned for its exceptional plotting capabilities within the R language.
- **car**: Utilized for conducting assumption tests like Levene's and Shapiro-Wilk's, ensuring the robustness of statistical analyses.
- **FSA**: Enables post hoc testing, enhancing the depth of statistical analysis.

We select the necessary columns that are useful for building models and prediction and removing all the remaining columns (we have explain this in Part 1. Introduction).

In our project, renaming variables is crucial for enhancing code convenience and ease of modification, as it avoids the hassle of dealing with lengthy variable names. Subsequently, we export this file named *cpu_ choose_ cols.csv* to the designated working directory.



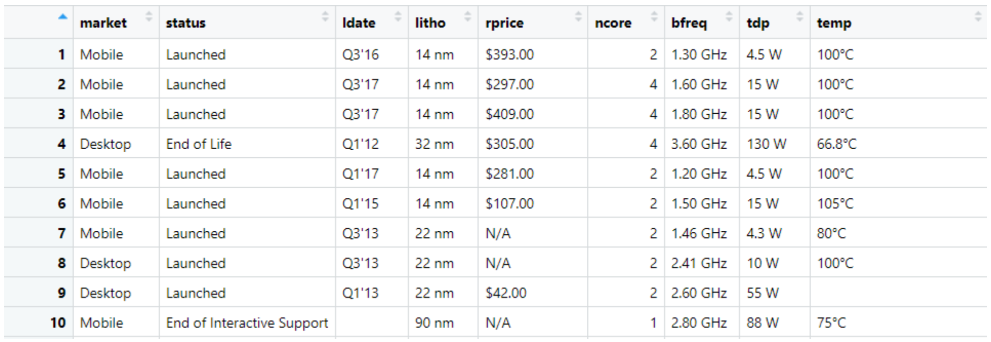
```
names(data1) <- c("market", "status", "ldate", "litho",
                  "rprice", "ncore", "bfreq", "tdp", "temp")
export(data1, "Dataset/cpu_choose_cols.csv")
```

Image 3.2: Code for Renaming the columns and Exporting

3.2 Data Cleaning

After selecting the appropriate attributes, we now possess a subset of the original raw dataset. However, as the values exhibit various types (such as strings, non-standard year-quarter formats, and numeric-strings), it is advisable to standardize them into reproducible types. This standardization process ensures that subsequent analysis is smoother, more uniform, and accurate.

It's worth noting that this cleaning process **does not automatically eliminate NA** (missing) values unless absolutely necessary. This approach is taken because there may be instances where these **NA** values hold significance and should not be discarded. Depending on the specific focus of the analysis, **NA** handling will be tailored accordingly. Thus, we avoid inadvertently losing any crucial instances during this preliminary cleaning phase. **market** and **status** will be left unchanged.



	market	status	ldate	litho	rprice	ncore	bfreq	tdp	temp
1	Mobile	Launched	Q3'16	14 nm	$393.00	2	1.30 GHz	4.5 W	100°C
2	Mobile	Launched	Q3'17	14 nm	$297.00	4	1.60 GHz	15 W	100°C
3	Mobile	Launched	Q3'17	14 nm	$409.00	4	1.80 GHz	15 W	100°C
4	Desktop	End of Life	Q1'12	32 nm	$305.00	4	3.60 GHz	130 W	66.8°C
5	Mobile	Launched	Q1'17	14 nm	$281.00	2	1.20 GHz	4.5 W	100°C
6	Mobile	Launched	Q1'15	14 nm	$107.00	2	1.50 GHz	15 W	105°C
7	Mobile	Launched	Q3'13	22 nm	N/A	2	1.46 GHz	4.3 W	80°C
8	Desktop	Launched	Q3'13	22 nm	N/A	2	2.41 GHz	10 W	100°C
9	Desktop	Launched	Q1'13	22 nm	$42.00	2	2.60 GHz	55 W	
10	Mobile	End of Interactive Support		90 nm	N/A	1	2.80 GHz	88 W	75°C

Image 3.3: The view of cpu_choose_cols.csv *before being cleaned and exported*

*Launched Dates (**ldate**)*

Our objective is to standardize non-standard year-quarter values into the format supported by the **zoo** package. However, upon examining the values within this column, we noticed variations, such as "Q1 '15", which deviates from the expected format. Consequently, our initial step involves ensuring uniformity by converting all strings into the format: "***Q<number>'<number>***".

Then, we utilize the function ***as.yearqtr***. The format string, represented as "Q%q'%y" aids the function in accurately interpreting the structure of our data.

*Lithography (**litho**) and Thermal Design Power (**tdp**)*

Our objective is to remove "**nm**" from the data of **Lithography** and "**W**" for data of **Thermal Design Power**, as every entry is consistently recorded in **nanometers** and **Watts**. In this code, we use the function **gsub()** to replace "**nm**" for litho and "**W**" for tdp with an empty string.

*Recommended Customer Price (**rprice**)*

From our view, we have seen values in three formats: "**$<number>", "$<number> - $<number>**" and "**N/A**". For the string values that in form of a range, we decide to take the Largest value and for the string values of "**N/A**", we will convert them to **NA** values.

*Base frequency (**bfreq**)*

Our objective is to remove "**GHz**" and "**MHz**" from the data, then convert all **MHz** value to **GHz** value by multiple with 0.001. Note that we also remove NA values in the column.

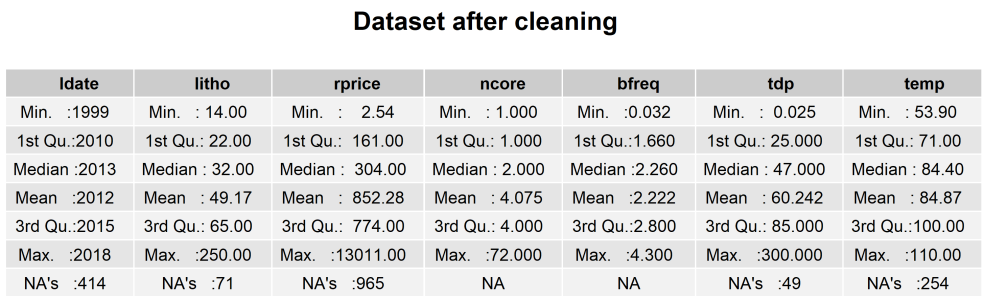
*Temperature (**temp**)*

Our objective is to identify and extract numeric values from the string and then determine the maximum among them. Here's the approach to process (Iterate through each row):

- Initially, we attempt to match every possible decimal number, including irrelevant ones. The rest are replaced with commas (","). This results in a string containing isolated numeric values.

- Next, we split these numbers to form a vector using the ***strsplit*** function. It's important to note that these numbers are still in string format. Then, we convert these strings into numeric values and aggregate them into a vector using ***unlist*** and ***lapply***. If after split the strings of numbers to form the vector but the size of the vector is 0, then we will assign **NA** to that value.

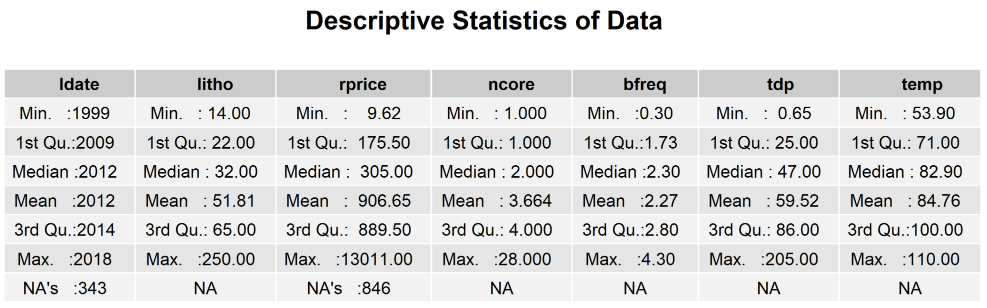
After doing so, we decide to take a look at the summary of the data after those previous cleaning. From our view and considerations, as the number of NA value of tdp, litho and temp is small (49, 71, and 254 compare with 2265 records in ***data1***), we decide to drop those NA values.



Dataset after cleaning

ldate	litho	rprice	ncore	bfreq	tdp	temp
Min. :1999	Min. : 14.00	Min. : 2.54	Min. : 1.000	Min. :0.032	Min. : 0.025	Min. : 53.90
1st Qu.:2010	1st Qu.: 22.00	1st Qu.: 161.00	1st Qu.: 1.000	1st Qu.:1.660	1st Qu.: 25.000	1st Qu.: 71.00
Median :2013	Median : 32.00	Median : 304.00	Median : 2.000	Median :2.260	Median : 47.000	Median : 84.40
Mean :2012	Mean : 49.17	Mean : 852.28	Mean : 4.075	Mean :2.222	Mean : 60.242	Mean : 84.87
3rd Qu.:2015	3rd Qu.: 65.00	3rd Qu.: 774.00	3rd Qu.: 4.000	3rd Qu.:2.800	3rd Qu.: 85.000	3rd Qu.:100.00
Max. :2018	Max. :250.00	Max. :13011.00	Max. :72.000	Max. :4.300	Max. :300.000	Max. :110.00
NA's :414	NA's :71	NA's :965	NA	NA	NA's :49	NA's :254

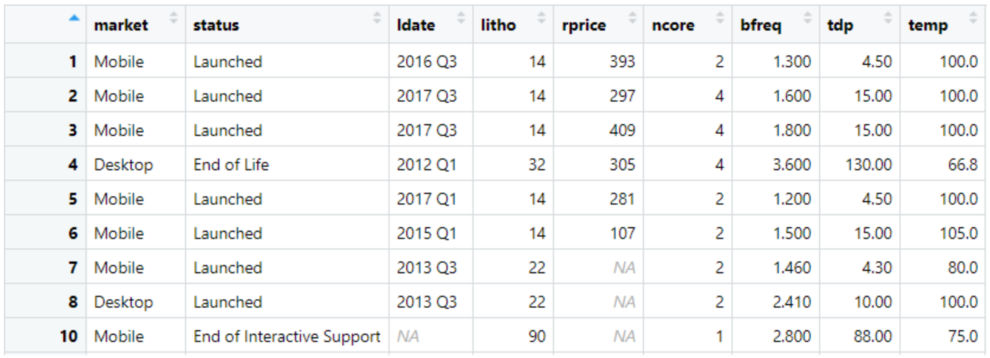
Image 3.4: Data summary after cleaning before dropping NA



Descriptive Statistics of Data

ldate	litho	rprice	ncore	bfreq	tdp	temp
Min. :1999	Min. : 14.00	Min. : 9.62	Min. : 1.000	Min. :0.30	Min. : 0.65	Min. : 53.90
1st Qu.:2009	1st Qu.: 22.00	1st Qu.: 175.50	1st Qu.: 1.000	1st Qu.:1.73	1st Qu.: 25.00	1st Qu.: 71.00
Median :2012	Median : 32.00	Median : 305.00	Median : 2.000	Median :2.30	Median : 47.00	Median : 82.90
Mean :2012	Mean : 51.81	Mean : 906.65	Mean : 3.664	Mean :2.27	Mean : 59.52	Mean : 84.76
3rd Qu.:2014	3rd Qu.: 65.00	3rd Qu.: 889.50	3rd Qu.: 4.000	3rd Qu.:2.80	3rd Qu.: 86.00	3rd Qu.:100.00
Max. :2018	Max. :250.00	Max. :13011.00	Max. :28.000	Max. :4.30	Max. :205.00	Max. :110.00
NA's :343	NA	NA's :846	NA	NA	NA	NA

Image 3.5: Data cleaning after dropping some NA values



	market	status	ldate	litho	rprice	ncore	bfreq	tdp	temp
1	Mobile	Launched	2016 Q3	14	393	2	1.300	4.50	100.0
2	Mobile	Launched	2017 Q3	14	297	4	1.600	15.00	100.0
3	Mobile	Launched	2017 Q3	14	409	4	1.800	15.00	100.0
4	Desktop	End of Life	2012 Q1	32	305	4	3.600	130.00	66.8
5	Mobile	Launched	2017 Q1	14	281	2	1.200	4.50	100.0
6	Mobile	Launched	2015 Q1	14	107	2	1.500	15.00	105.0
7	Mobile	Launched	2013 Q3	22	NA	2	1.460	4.30	80.0
8	Desktop	Launched	2013 Q3	22	NA	2	2.410	10.00	100.0
10	Mobile	End of Interactive Support	NA	90	NA	1	2.800	88.00	75.0

Image 3.6: Data after cleaning and dropping some NA values

4 Descriptive Statistics

4.1 Visualization

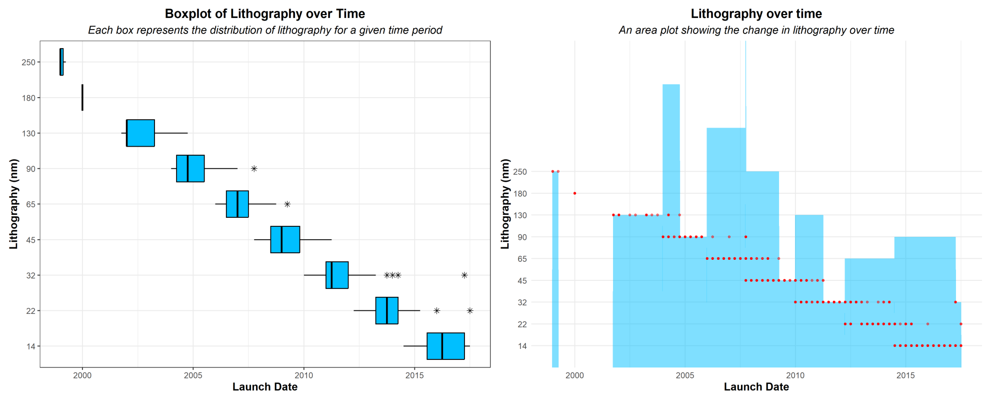


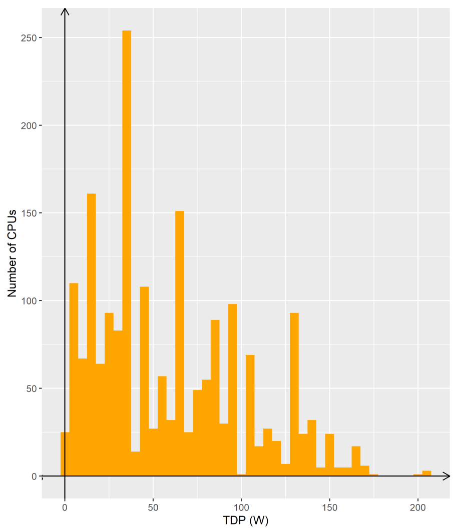
Image 4.1 and 4.2: Boxplot and Scatterplot of Lithography over the time (Launch Date)

The graphs illustrate the trend of lithography size (in nanometers) over time, specifically from the years 2000 to 2015. Here's a more detailed analysis:

- ***Boxplot of Lithography over Time***: This graph shows the distribution of lithography sizes for different time periods. Each box represents the range of lithography sizes within each period. The trend of decreasing size over the years is evident, indicating that lithography technology has been improving, with the ability to create smaller and more precise structures.

- ***Lithography over Time***: This area plot provides a visual representation of the change in lithography sizes over time. The shaded areas indicate the levels and distributions of lithography sizes. The red dotted lines mark significant changes or reductions in size, which could correspond to breakthroughs in technology or the introduction of new lithography techniques.

From things illustrated above, it can be concluded that lithography and launch date have strong relationship. Therefore, when analysing deeply into the relation of these elements with TDP, it is sufficient to use lithography as the representative element.



Min.	0.65
1st Qu.	25
Median	47
Mean	59.5170805541303
3rd Qu.	86
Max.	205
Mode	35

Image 4.3: Histogram and Summary of Thermal Design Power

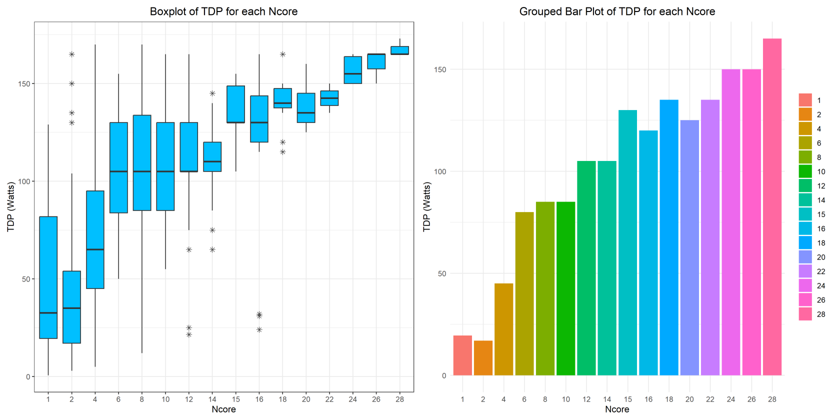


Image 4.4: An increasing trend of tdp when ncore increases

Boxplot Analysis:

- The box plot displays the distribution of TDP values across different Ncore categories.
- The median TDP value for each Ncore is indicated by the line within each box.
- The range of the boxes represents the interquartile range (IQR), showing the middle 50% of the data.
- Outliers, the data points that fall outside the typical range, are visible as individual points.
- It can be seen that **the general tdp will increase as the number of cores increases**. The low number of cores can lead to the variation of their tdp, which can be shown in 4-cpu cores and 8-cpu cores.

Grouped Bar Plot Analysis:

- The grouped bar plot shows the average TDP for each Ncore category. We decide to **limit the maximum tdp for each core to the first quantile (Q1)** of its distribution in order to clearly see the **upward trend of tdp as ncore increases**.
- Each bar's height represents the mean TDP value, making it easy to compare between different Ncores.
- Both plots provide a clear visualization of how TDP varies with the number of cores. The box plot is useful for understanding the spread and outliers in the data, while the grouped bar plot offers a straightforward comparison of average TDP values. Together, they offer a comprehensive view of the dataset's trends and patterns.

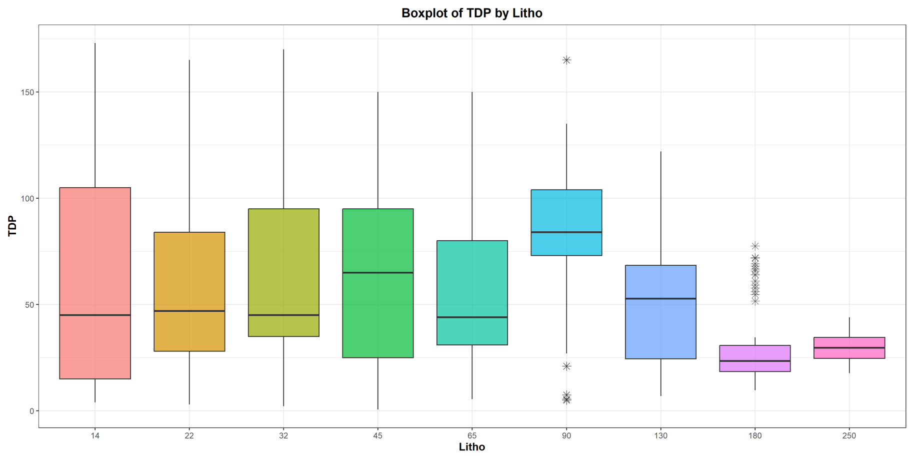


Image 4.5: TDP within different Lithography

Lithography as tdp is less convincing. However, we see that recent lithography techniques tend to have **stable TDP**.

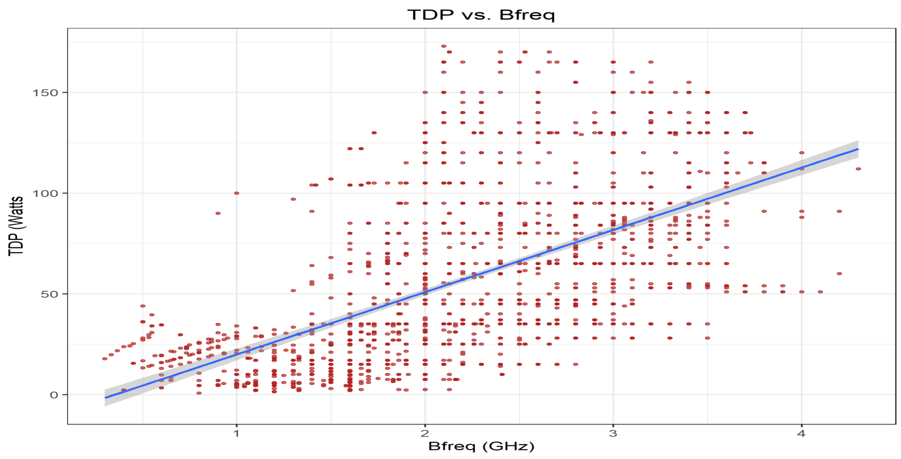


Image 4.6: Increasing trend of TDP when Base Frequency increases

As we can see from the plot of TDP together with Base Frequency, when the Base Frequency increases, it is likely that TDP increases too, which can conclude that the linearity relationship between Bfreq and TDP is true.

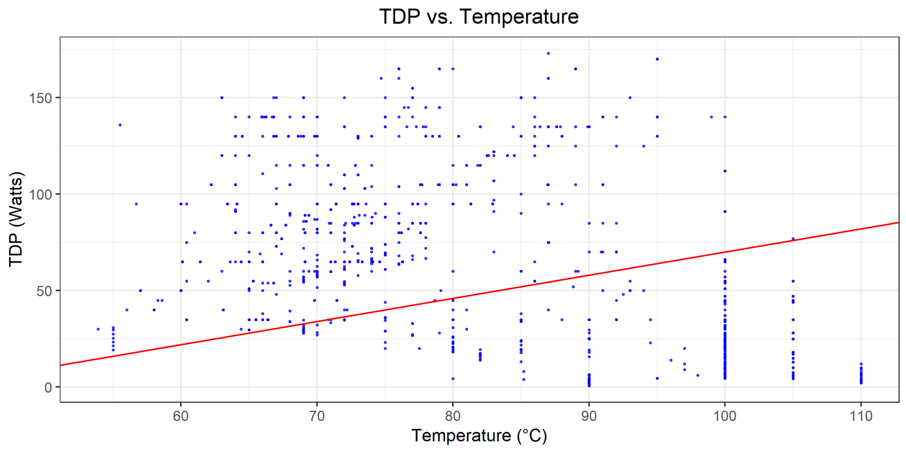


Image 4.7: Different trend of TDP in different ranges of Temperature

The scatter plot in Image 4.7 shows a positive correlation between TDP (Thermal Design Power) and temperature. Here's a brief analysis:

- ***Positive Correlation***: The red trend line indicates that as the temperature increases, the TDP also tends to increase.

- ***Data Concentration***: Most data points are clustered between 70°C and 90°C for temperature and 50 to 150 Watts for TDP, suggesting this is the common operating range.

- ***TDP Range***: The TDP values range from 0 to 200 Watts, which covers a wide spectrum of possible device power consumptions.

- ***Temperature Range***: The temperature on the x-axis ranges from 50°C to 110°C, indicating the thermal conditions under which the devices were tested.

This analysis suggests that devices with higher TDP ratings may require more robust cooling solutions to manage higher temperatures. It's important to consider both TDP and temperature when designing or selecting cooling systems for electronic devices.

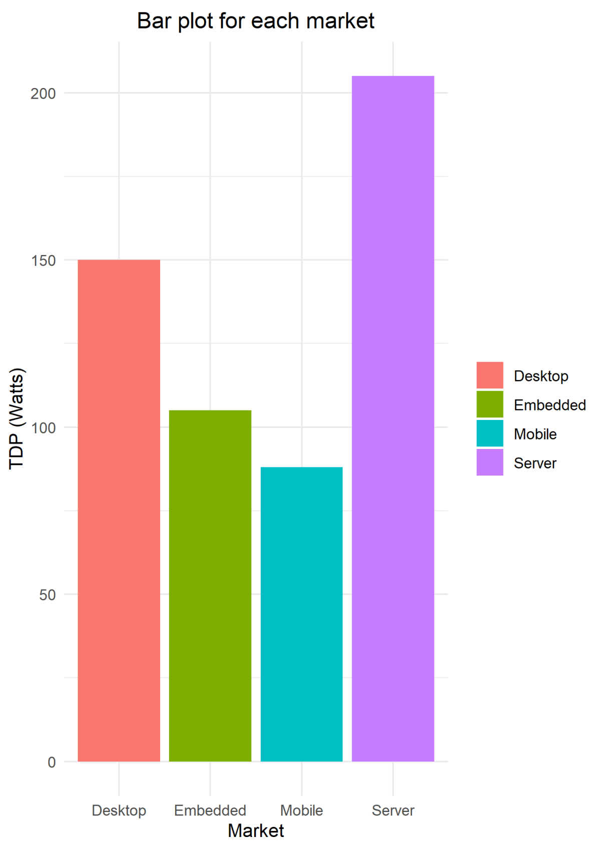
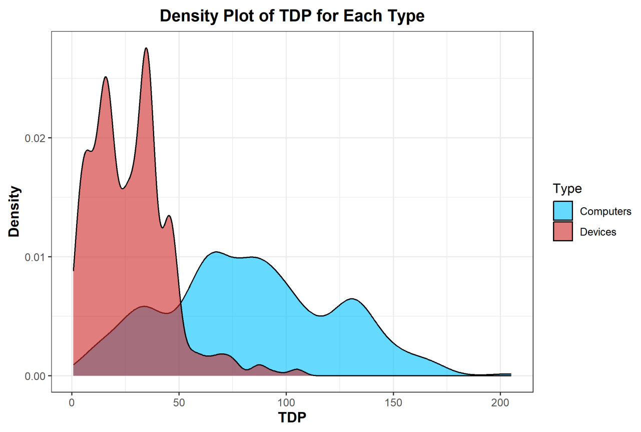


Image 4.8: Max Value of TDP of each Type of Market



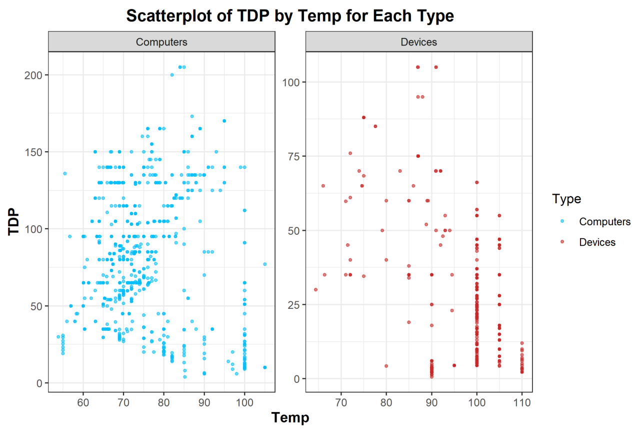


Image 4.9a and b: It is likely that when we put Desktop and Server together, they will likely have an increasing trend, whereas when we put Embedded and Mobile together, they will likely have an decreasing trend. And these two trends are seemly linearity.

Density Plot Analysis (Image 4.9a):
- Computers: Higher density at lower TDP values suggests that most computers are designed to operate efficiently with lower thermal output.

- Devices: Broader range in density, indicating a wider variety of TDP values, but also concentrated at lower TDPs.

Scatterplot Analysis (Image 4.9b):
- Temperature Impact: As temperature increases, the TDP for Computers shows a significant rise, whereas Devices maintain a stable TDP across various temperatures.

Small Conclusion:
- Computers are used for a **wide range of applications**, from simple tasks like browsing the internet to more **intensive tasks** like gaming or running complex simulations. This could result in a wider range of TDP values as different tasks require different amounts of power.
- In contrast, Devices might have a narrower TDP range because they are often designed for **specific, less power-intensive tasks**, and might use more standardized or optimized hardware components.

4.2 Conclusion

From the visualizations, we can draw several comments regarding these relationships, which prompt us to consider specific ***Regression models***:
- [Image 4.1, 4.2] ***litho*** $\sim$ ***ldate***: As we can see from the plot, lithography and launched date have a strong relationship. Therefore when considering the relationship between litho and launched date with TDP, it is sufficient to use Litho as the representative element.
- [Image 4.4] ***tdp*** $\sim$ ***ncore***: As the number of cores (ncore) increases, TDP tends to increase. This suggests the potential utility of a Linear regression model. However, we also notice distinct "clusters" across different ranges. Hence, a decision-based model such as Random Forest or XG-Boost might offer better performance.
- [Image 4.5] ***tdp*** $\sim$ ***litho***: No clear upward or downward trend is evident. Instead, TDP appears to converge and stabilize as Lithography increases.
- [Image 4.6] ***tdp*** $\sim$ ***bfreq***: Although the data for this relationship exhibits variability, there is a dominant linearly increasing trend. Thus, utilizing Linear regression could yield reliable results.
- [Image 4.7, 4.8, 4.9] ***tdp*** $\sim$ ***temp*** and ***tdp*** $\sim$ ***market***: Two distinct trends are apparent: an increasing trend above the reference line and a decreasing trend below it. Interestingly, we suspect that these trends correspond to different market segments, namely (Desktop + Server) and (Mobile + Embedded). To verify this, we plot them separately (Image 4.9b) and discover that indeed, distinct trends correspond to different market segments. This insight could be leveraged for classifying TDP into different markets using Logistic regression.

Some Considerations:
Taking into account the information illustrated above, it can be claimed that **ncore, bfreq** and **temp** play *important role* in analyze data relating to **tdp** while **litho** should not be used due to its minor contribution.

5 Inferential Statistics

The focus of our report is on Thermal Design Power (TDP) and how various attributes influence this key parameter. TDP represents the maximum heat a CPU's cooling system can handle without surpassing its limits. Essentially, it sets a heat threshold for the processor, and a well-designed cooling system can manage most of the heat it generates. TDP also gives an indication of a CPU's power consumption; higher TDP values suggest better cooling but also indicate higher energy consumption for heat dissipation.

We chose TDP over temperature because we're interested in understanding the energy required to maintain optimal CPU performance, rather than the temperature at which the CPU might be at risk of damage.

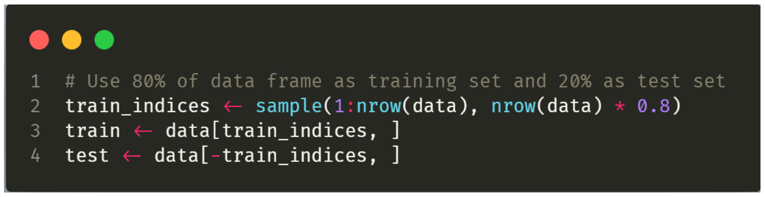
Our approach involves using regression models to predict TDP based on various factors like number of cores, base frequency, temperature, lithography, market segment, and status. Additionally, we'll explore whether the calculated TDP can help classify CPUs into categories like mass computing (servers) or personal devices like desktops and mobiles.

5.1 Data preparation

First, we load the clean dataset **cpu_clean.csv** from the above process.

As we have seen in Part 4, the **occurrences of tdp** with value ≥ 150 is rare, so we decided to remove them from our dataset.

To effectively use regression models and assess their accuracy, we need both a Training Set and a Test Set. This allows us to perform cross-validation to evaluate the model's performance. Additionally, we'll use visualization techniques, like plotting graphs and examining coefficients, to better understand the relationships in the data. The original dataset will be divided into two subsets: a Training Set and a Test Set. Specifically, 80% of the data will be allocated to the Training Set, while the remaining 20% will be used for the Test Set.



```r
# Use 80% of data frame as training set and 20% as test set
train_indices <- sample(1:nrow(data), nrow(data) * 0.8)
train <- data[train_indices, ]
test <- data[-train_indices, ]
```

Image 5.1: Code for Splitting the data into Training and Test Sets

Explanation:

To ensure consistent randomness across all our tests, we set a seed value to 123. This ensures that every time we execute the code, we'll get the same data split.

We use the ***sample()*** function to randomly select 80% of the data set's indices without replacement. The function returns these indices, which we then use for indexing to create the Training Set and Test Set.

Following this, we use straightforward indexing methods to allocate the selected indices to the Training Set and the remaining ones to the Test Set.

5.2 Regression Analysis

In this section, we will explore the relationship between Thermal Design Power and other features based on the checking of their linearity. There are many types of regression models such as simple linear regression, multiple linear regression, polynomial regression, logistic regression, and more. But in the limitation of this report, we will use a particular model, Multiple Linear Regression, to check if those features have any significant relationship with TDP.

5.2.1 Model Definition

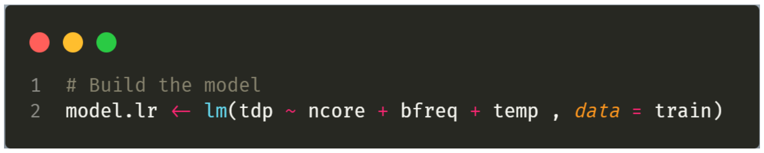
The formula of multiple linear regression that we use in this Assignment:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$$

Where:
- y is our dependent variable, specifically it is TDP (Thermal Design Power).
- β_0 is the intercept
- β_i with i = 1, 2, 3 are regression coefficients of the model with respect to x_i.
- x_i with i = 1, 2, 3 are independent variables such as ncore (Number of Cores), bfreq (Base Frequency) and temp (Temperature) respectively.
- ϵ is the model's error (known as residuals)

5.2.2 Multi-Linear Model

Since the scatter plot of the Thermal Design Power (TDP) with other variables suggests a linear relationship, our first step is to construct a linear regression model using the ***lm()*** function.



```r
# Build the model
model.lr <- lm(tdp ~ ncore + bfreq + temp , data = train)
```

Image 5.2: Code for building the Model

5.2.3 Summary of Multi-Linear Model

- Upon examining the summary table, the multiple R-squared value is 0.7186, and the Adjusted R-squared is 0.718. This suggests that approximately 71.8% of the variability in the Thermal Design Power variable is explained by the independent variables in the model. The remaining 28.2% is accounted for other factors. Thus, the model appears to perform reasonably well.

- The summary provides additional insights:

+ The ***residuals*** vary from approximately −70.268 to 91.102. The ***first, median, and third quantiles*** of the residuals are −11.068, −0.392, and 10.258, respectively, indicating a spread in prediction errors, though it's not substantial.

+ The ***Estimate*** column is the estimated effect, also called regression coefficient. For example in the Summary Plot below we can get that for every one percent increase in the Thermal Design Power, there is 5.1% increase in Number of Cores, 18.3% increase in Base Frequency and

−1.1% decrease in Temperature.

+ The Std.Error columns demonstrate the ***Standard Error*** of the Estimate. These show how much variation there is around the estimates of regression coefficient. For example we can read that the standard error of Estimate Temperature would be 0.03825.

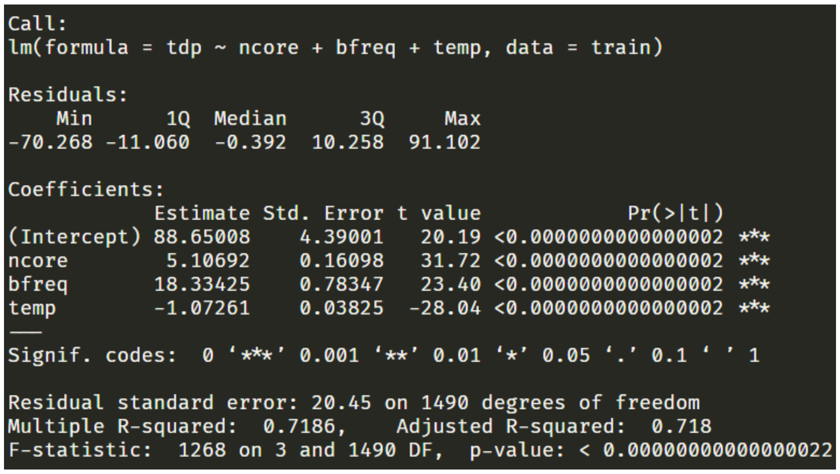
+ The ***t-value*** columns display the test statistic and the $Pr(>|t|)$ columns display the ***p-value***. This show how likely the t-value would have occurred by the chance if the null Hypothesis of no effect of the parameter is true. As we can see that the p-value for ***ncore, bfreq and temp*** are so low (much less than 0.0001), we can ***reject the null Hypothesis*** and conclude that ncore, bfreq and temp likely influence the Thermal Design Power.

+ The ***Residual Standard Error (RSE)*** of the model is approximately 20.45, representing the typical deviation of actual values from predicted values.

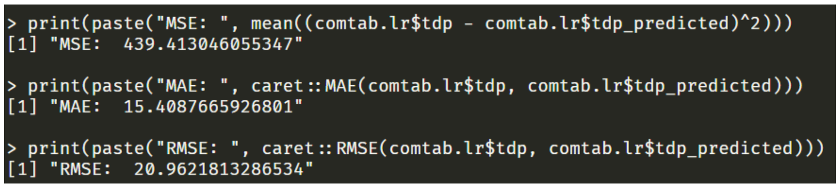
+ Both the ***Multiple R-Squared*** (0.7186) and ***Adjusted R-Squared*** (0.718) indicate that the model accounts for a significant proportion of the response's variability. The Adjusted R-Squared, which considers the model's complexity, is slightly lower than the Multiple R-Squared, indicating a good fit.

+ The overall ***F-statistic*** is 1268 with a corresponding p-value much smaller than 0.05. This tests the hypothesis that at least one predictor contributes significantly to the response. Given the small p-value, we reject the null hypothesis, confirming that at least one predictor significantly influences the response.

+ We also calculate ***Mean Absolute Error (MAE)***, ***Mean Square Error (MSE)*** and ***Root Mean Square Error (RMSE)*** of the model. After calculation, we get these values: ***MSE*** equals to 439.41, ***MAE*** equals to 15.41, and ***RMSE*** equals to 20.96. Our approach in the Extension Part of this report is to use another model to reduce these values and predict with smaller errors.



```
Call:
lm(formula = tdp ~ ncore + bfreq + temp, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-70.268 -11.060  -0.392  10.258  91.102 

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)    
(Intercept) 88.65008    4.39001   20.19 <0.0000000000000002 ***
ncore        5.10692    0.16098   31.72 <0.0000000000000002 ***
bfreq       18.33425    0.78347   23.40 <0.0000000000000002 ***
temp        -1.07261    0.03825  -28.04 <0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20.45 on 1490 degrees of freedom
Multiple R-squared:  0.7186,    Adjusted R-squared:  0.718 
F-statistic:  1268 on 3 and 1490 DF,  p-value: < 0.00000000000000022
```



```
> print(paste("MSE: ", mean((comtab.lr$tdp - comtab.lr$tdp_predicted)^2)))
[1] "MSE:  439.413046055347"

> print(paste("MAE: ", caret::MAE(comtab.lr$tdp, comtab.lr$tdp_predicted)))
[1] "MAE:  15.4087665926801"

> print(paste("RMSE: ", caret::RMSE(comtab.lr$tdp, comtab.lr$tdp_predicted)))
[1] "RMSE:  20.9621813286534"
```

Image 5.3: Summary of Multi-Linear Model.

From the Summary, we can form the following Estimated Regression Equation:
$tdp = 88.6501 + 5.1069 \cdot ncore + 18.3343 \cdot bfreq - 1.0726 \cdot temp$

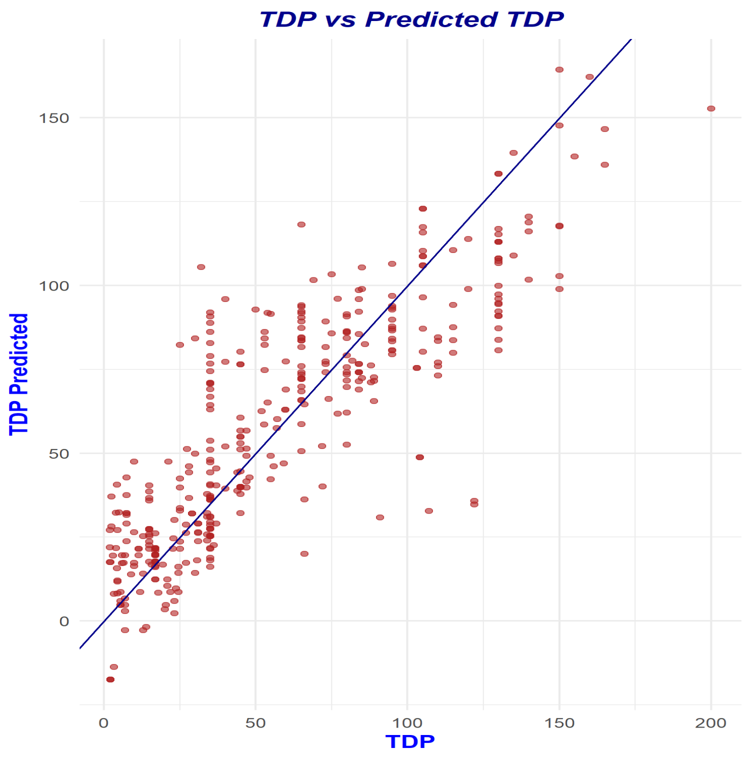


Image 5.4: Code for creating and Image of the Predict-Actual Comparing plot

To gain a clearer understanding of this trend, we will create a scatter plot for comparing the predicted values from the test set to the actual values in the dataset using the ***plot()*** function. We also plot the line (d): $y = x$ in the graph. The more concentration of the scattered points to this line, the more accurate the model is.

To conclude, this model provides a reasonable fit to the dataset, as indicated by the results and an image above. However, as from the image above, there are some predicted values not concentrated tightly on the line, that is the reason why we continue to investigate the more suitable model in the Extension Part of the Report.

5.2.4 Checking the Assumptions of the Model

There are 5 assumptions we must check for this model:

1. ***Linearity***: The independent and dependent variables have a linear relationship.
2. ***Homoscedasticity***: The variance of the residuals should be consistent across the model.
3. ***Multivariate***: Normality Residuals should follow a normal distribution.
4. ***Independence of Observations:*** No autocorrelation.
5. ***No Multicollinearity***: Independent variables should not be highly correlated.

First Assumption: Linearity.
We have checked this in the Descriptive Statistics already and this assumption is met.

Second Assumption: Homoscedasticity - Residual Errors have Constant Variance
We can verify this assumption using the **Scale-Location plot**. This type of plot displays the fitted value of the regression model along the x-axis and the square root of the standardized residuals along the y-axis. Ideally, we **would like to see the residual points evenly distributed horizontally around the red line**, indicating consistent variance.

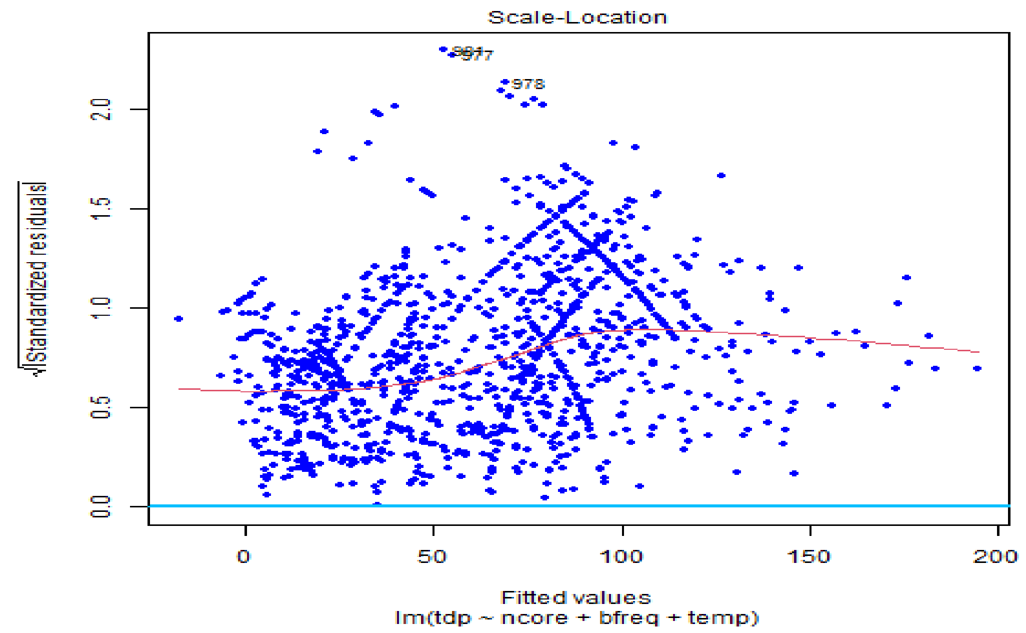


Image 5.5: Code for creating and Image of the Scale-Location plot.

From the plot we can see that the points are randomly scattered around the red line with roughly equal variability at all fitted values. Therefore, the second assumption is also met.

Third Assumption: Multivariate - The Errors are normally distributed

At first, we will check that Residual Errors have Mean Value of Zero by using the histogram of the residual.

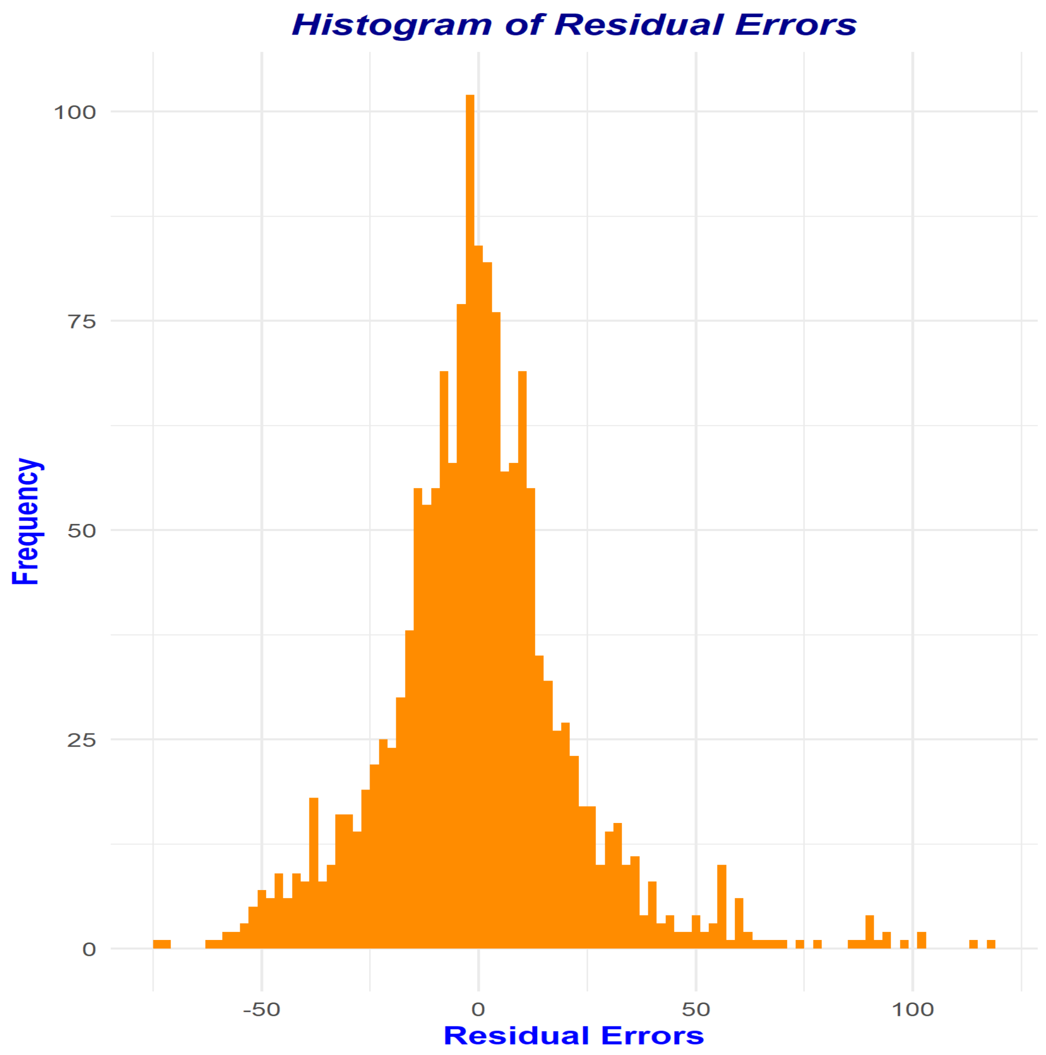


Image 5.6: Code for creating and Image of the Histogram of the residual.

We notice that the residuals are mainly centered around zero, indicating that they are mostly scattered around zero and equally distributed on both sides of the graph. This suggests that the mean of the residuals is close to zero. Furthermore, we can see that the graph has the bell shape, which looks like the shape of the Normality Distribution.

To verify that the Residual Errors are normally distributed, we'll use a Q-Q plot to assess normality. Ideally, we expect that the residuals will mostly scatter closely to the straight line, supporting the acceptance of the normality hypothesis.

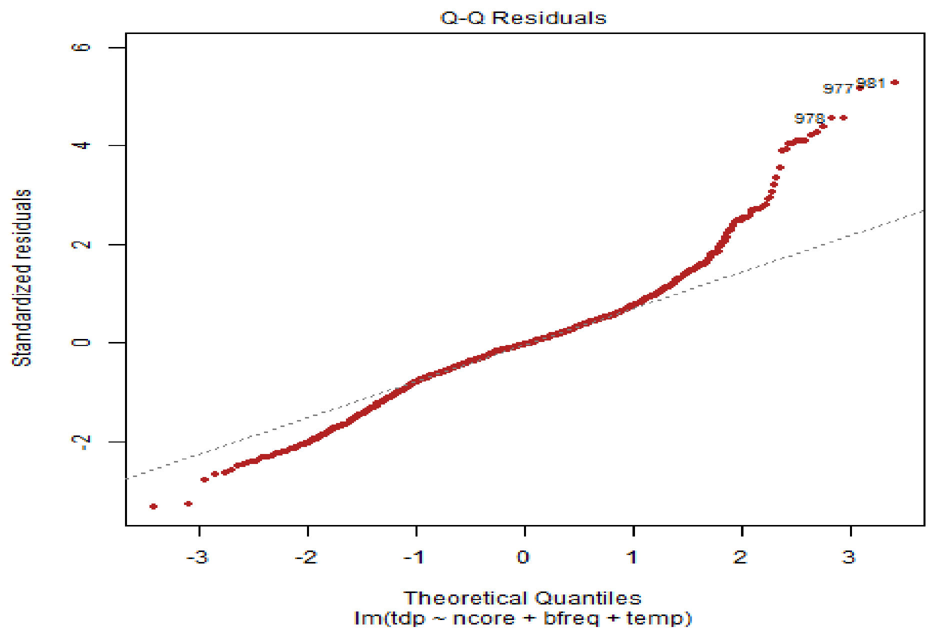
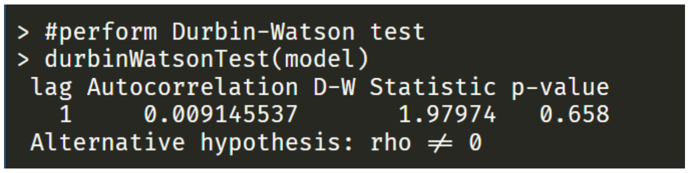


Image 5.7: Code for creating and Image of the Q-Q plot.

From the **Q-Q plot**, we can see that most points align closely with the line, with only a few outliers deviating from it. This suggests that the model is **reasonably accurate** in terms of normality, though not perfect. It's possible that **another model might better suit** this data.

Fourth Assumption: Independence of Observations.



```
> #perform Durbin-Watson test
> durbinWatsonTest(model)
 lag Autocorrelation D-W Statistic p-value
   1     0.009145537       1.97974   0.658
 Alternative hypothesis: rho ≠ 0
```

Image 5.8: Durbin-Watson Test.

Since the p-value is large (greater than 0.05) and the Test Statistic is 1.98 (nearly 2), we do not have enough evidence to reject the Null Hypothesis of the Durbin-Watson Test, which means that there is no correlation among the residuals (no autocorrelation) or the observations we get is independent.

Fifth Assumption: Multicollinearity.

Multicollinearity has been checked for the variables in the model using the Variance Inflation Factor (VIF). The VIF is a measure of how much the variance of the estimated regression coefficients are increased due to multicollinearity.

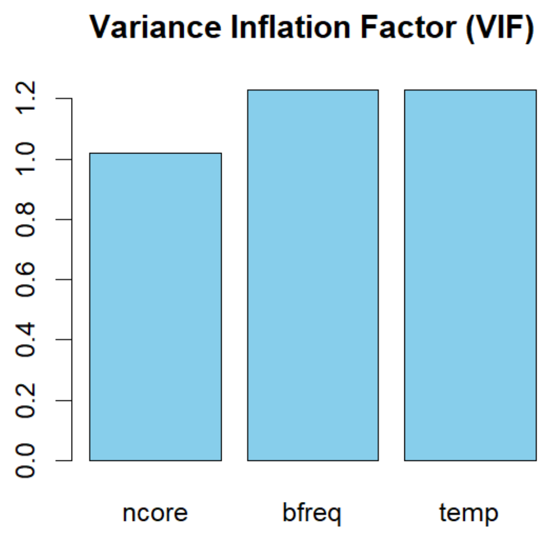


Image 5.9: VIF Summary.

Generally based on the theory that we provide in Background Section, a VIF greater than 5 or 10 indicates a problematic amount of multicollinearity. In this case, none of the variables have a VIF above 5, which means multicollinearity is not a problem.

5.3 Extension: XGBoost (eXtreme Gradient Boosting)

Before discusing about the use of XGBoost Model, we will discuss some advantages and disadvantages of using Multi-Linear Regression Model.

Advantages

The main benefit of multi-linear regression models is their simplicity due to linearity. This makes the estimation process straightforward, and the linear equations are easily interpretable at a basic level. Additionally, the mathematical equation for Linear Regression is quite straightforward and easy to understand.

Disadvantages

- Prone to be underfitting

Underfitting is a sitiuation that arises when a machine learning model fails to capture the data properly.This typically occurs when the hypothesis function cannot fit the data well.

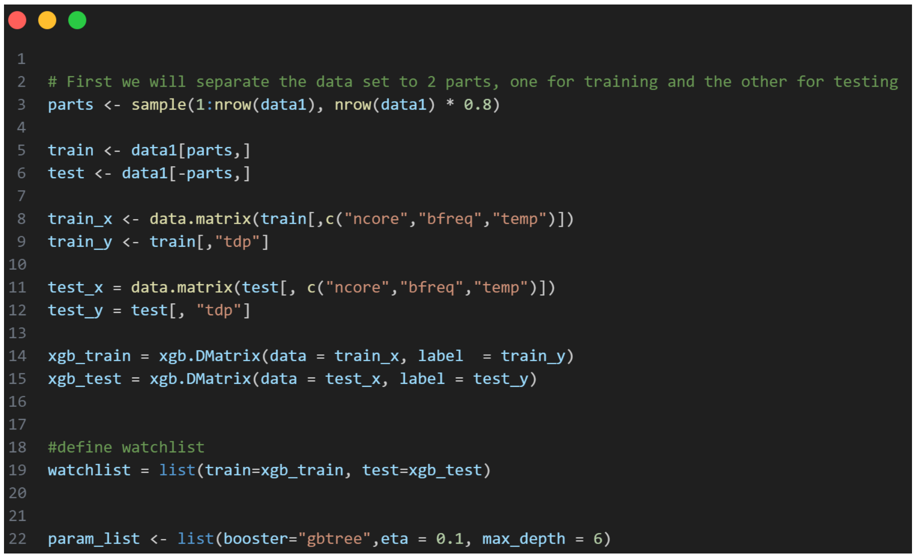
In most real life scenarios the relationship between the variables of the dataset isn't linear and hence a straight line doesn't fit the data properly. In such situations a more complex function can capture the data more effectively.Because of this most linear regression models have low accuracy.

- Sensitive to outliers

Outliers of a data set are anomalies or extreme values that deviate from the other data points of the distribution.Data outliers can damage the performance of a machine learning model drastically and can often lead to models with low accuracy.

Because of those disadvantages, in this section, we introduce the XGBoost model which is much more accurate in prediction than multiple linear regression model. XGBoost can efficiently handle NAs, categorical and continuous data. Moreover, this machine learning algorithm is also used in a large number of datasets on Kaggle and won numerous prizes on that platform through many years. Thus, this model is clearly suitable for analyzing regression of CPU attribute data that may contain a mix of categorical variables.

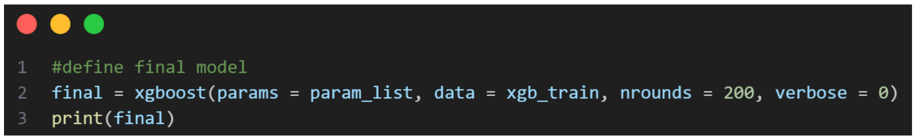
First, we will prepare the data for XGBoost model:



```

# First we will separate the data set to 2 parts, one for training and the other for testing
parts <- sample(1:nrow(data1), nrow(data1) * 0.8)

train <- data1[parts,]
test <- data1[-parts,]

train_x <- data.matrix(train[,c("ncore","bfreq","temp")])
train_y <- train[,"tdp"]

test_x = data.matrix(test[, c("ncore","bfreq","temp")])
test_y = test[, "tdp"]

xgb_train = xgb.DMatrix(data = train_x, label  = train_y)
xgb_test = xgb.DMatrix(data = test_x, label = test_y)


#define watchlist
watchlist = list(train=xgb_train, test=xgb_test)


param_list <- list(booster="gbtree",eta = 0.1, max_depth = 6)
```

Image 5.11: Code for preparing the data

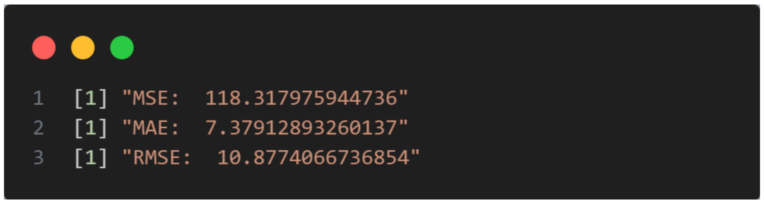
Then, we train our model with learning rate = 0.1 and the max depth of tree = 6 for maximize the accuracy beside the turn = 200:



```
#define final model
final = xgboost(params = param_list, data = xgb_train, nrounds = 200, verbose = 0)
print(final)
```

Image 5.12: Code for training data

For evaluating the performance of this model, we can calculate the ***Mean Absolute Error (MAE)*** and ***Root Mean Square Error (RMSE)***. The lower the values of 2 metrics, the better the model.



```
[1] "MSE:  118.317975944736"
[1] "MAE:  7.37912893260137"
[1] "RMSE:  10.8774066736854"
```

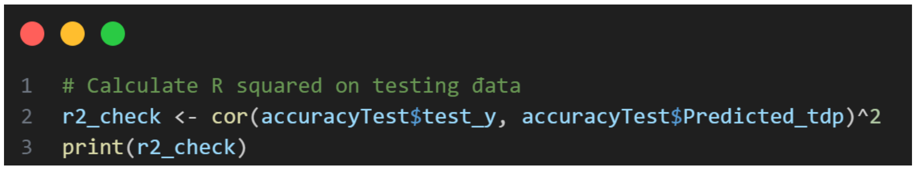
Image 5.13: Output of above code

MAE is commonly used metric for measuring the accuracy of forecast models. It provides a straightforward and intuitive way to quantify the average magnitude of errors in a set of predictions. ***MAE*** is calculated by taking the absolute difference between the actual values and the predicted values, and then divided by the number of observations in the test set.

The ***root mean square error (RMSE)*** measures the average difference between a statistical model's predicted values and the actual values. Mathematically, it is the standard deviation of the residuals. Residuals represent the distance between the regression line and the data points. Low ***RMSE*** values indicate that the model fits the data well and has more precise predictions.

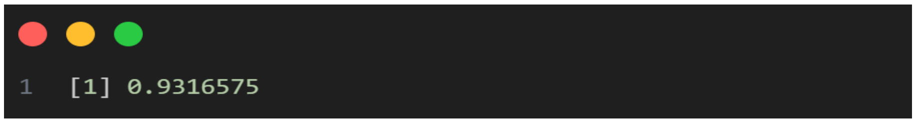
As we can see from the above result, ***MAE*** value is 7.3791 which is quite small compared to the range of the data. Similarly with ***RMSE***, the value is 10.8774 and it tells us that the model is relatively precise and works well with this dataset. Also note that these value are significantly smaller than those of ***Multi-Linear Model*** above (with 439.41 for ***MSE***, 15.41 for ***MAE*** and 20.96 for ***RMSE***). This means that ***XGBoost Model*** is better used for predicting the Thermal Design Power of a CPU based on ncore (Number of Cores of CPU), bfreq (Base Frequency of CPU) and temp (Max Temperature of the CPU).

To demonstrate evidently the efficiency of our model, the ***R-squared*** value will be taken into account:



```
# Calculate R squared on testing data
r2_check <- cor(accuracyTest$test_y, accuracyTest$Predicted_tdp)^2
print(r2_check)
```

Image 5.14: Code for calculating R-squared



```
[1] 0.9316575
```

Image 5.15: Output of R-squared

As aforementioned above, the R-squared value has a significant effect on deciding the performance of a model, the closer this metric to 1, the more fitted the model. In this output, 0.9316575 is so close to 1 which proves accurately the relationship between CPU attributes.
Finally, we plot the graph to see the accuracy between the predicted values of our model compared to the real values. The graph is plotted below:

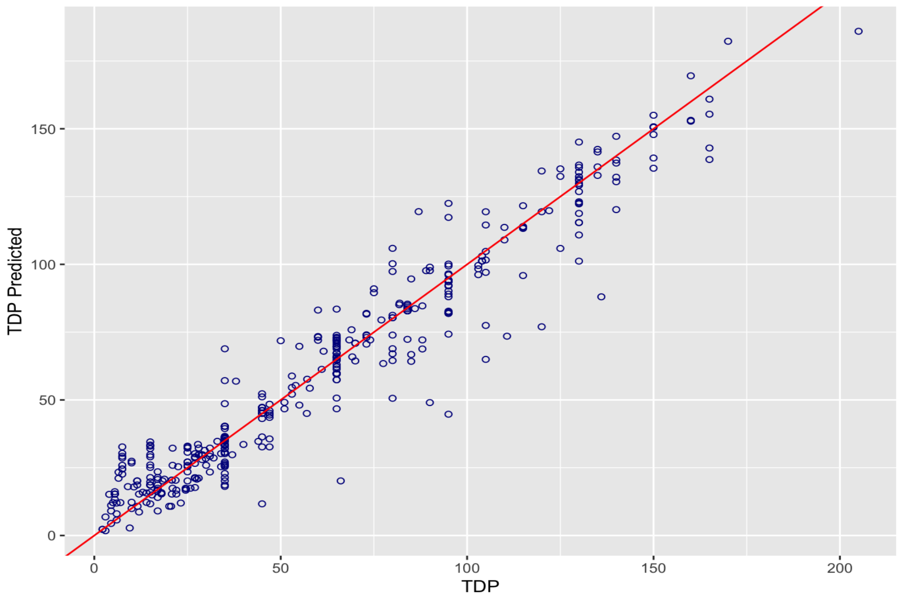


Image 5.16: Comparison between Actual value and Predicted Value

We can observe from the scatter plot that mainly, the predicted values lie around the red line. As more values stay near the line meaning the model is apparently more accurate.

In conclusion, compared to multiple linear regression model, XGBoost has outperformed in many aspects: errors, R-squared value and accurate prediction on the graph. However, we believe this model can still be improved by adjusting some of the parameters to achieve higher R-squared, smaller errors and better plots without getting overfit.

6 Conclusion

In our research, multiple test models have yielded overwhelmingly positive results, allowing us to confidently reject the null hypothesis. This supports our initial prediction that the thermal design of each CPU is significantly influenced by factors like the number of cores, base frequency, and the temperature experienced during intensive tasks in hard mode. This insight provides chip producers with valuable guidance to strike a balance between power consumption and performance, optimizing their competitiveness in the tech industry.

Using the R programming language to analyze statistical data and construct linear regression models, our team gained a clearer understanding of data extraction, processing, and analysis. This has enabled us to transform raw data into valuable, long-term resources and make informed predictions based on generalized situations.

While implementing this thesis, some minor errors are inevitable, and our predictions may not always align perfectly with reality. Nevertheless, we hope this report offers viewers a fresh perspective on addressing thermal design power challenges in practical scenarios.

References

[1] Lenovo. *How Thermal Design Power (TDP) impacts component efficiency.* https://www.lenovo.com/us/en/glossary/what-is-thermal-design-power/

[2] *Numeracy, maths and statistics - academic skills kit.* https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/regression-and-correlation/coefficient-of-determination-r-squared.html

[3] *P-values document.* https://sphweb.bumc.bu.edu/otlt/MPH-Modules/PH717-QuantCore/PH717-Module7-T-tests/Module7-ttests3.html

[4] *QQ plots document.* https://library.virginia.edu/data/articles/understanding-q-q-plots

[5] Mbanaso, Samuel. *CPU cores explained: Does the number of cores affect CPU performance?* https://www.auslogics.com/en/articles/does-more-cpu-cores-mean-better-performance/

[6] Roberts, Mathew. *What is the relationship between processing power and price for computers?* https://www.quora.com/What-is-the-relationship-between-processing-power-and-price-for-computers

[7] Foster, Nolan. *What is TDP for CPUs and GPUs?* https://www.acecloudhosting.com/blog/what-is-tdp-for-cpus-and-gpus/

[8] *Variance Inflation Factor (VIF).* (2010, August 14). Investopedia. https://www.investopedia.com/terms/v/variance-inflation-factor.asp

[9] Bobbitt, Z. (2020, April 2). *How to perform a Durbin-Watson test in R. Statology.* https://www.statology.org/durbin-watson-test-r/

[10] Cumins, N, Business News Daily (2018, May 24). *The history of Intel processors.* https://www.businessnewsdaily.com/10817-slideshow-intel-processors-over-the-years.html

[11] Research Gates (2020, November). *Optimizing Server Refresh Cycles: The Case for Circular Economy With an Aging Moore's Law.* https://www.researchgate.net/publication/346593629_Optimizing_server_refresh_cycles_The_case_for_circular_economy_with_an_aging_Moore%27s_Law

[12] CNX Software (2022, January 8). *TDP (Thermal Design Power) vs PBP (Processor Base Power) - Are there differences?.* https://www.cnx-software.com/2022/01/08/tdp-vs-pbp-thermal-design-power-vs-pbp-processor-base-power-differences/

[13] Curtis, R., Peterson, C., Moss, D., Hardy, D., Eiland, R., Shedd, T., Tunks, E., Shabbir, H., & Mottershead, T. *The Future of Server Cooling - part 2: New IT hardware features and power trends.* (2023, March 4). https://infohub.delltechnologies.com/en-us/p/the-future-of-server-cooling-part-2-new-it-hardware-features-and-power-trends-1/

[14] Intel. *Intel CPU Processor Family Product Lifecycle* (2022, December). https://www.intel.com/content/dam/support/us/en/documents/processors/Intel-CPU-Processor-Lifecycle-Technical-Paper.pdf