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

PROBABILITY AND STATISTICS (MT2013)

Assignment

CPU Prices Analysis according to Technical Attributes using Multivariate Linear Regression

Class CC03 - Group 09

Instructor(s): PHAN THỊ HƯỜNG

Students: Lê Nguyên Khang - 2352470
Nguyễn Bảo Đạt - 2352232
Võ Hoàng Long - 2053192
Lê Đặng Khánh Quỳnh - 2353037
Trần Hải Đăng - 2352254
Võ Hoàng Ngân - 2353373

HO CHI MINH CITY, DECEMBER 2024

Contents

Member list & Workload

No.	Fullname	Student ID	Problems	% done
1	Võ Hoàng Ngân	2353373	- Data Introduction - Background	100%
2	Nguyễn Bảo Đạt	2352232	- Inferential Statistics	100%
3	Võ Hoàng Long	2053192	- Data analysis - Descriptive statistics	100%
4	Lê Đặng Khánh Quỳnh	2353037	- Data Introduction - Background	100%
5	Trần Hải Đăng	2352254	- Data analysis - Descriptive statistics	100%
6	Lê Nguyên Khang	2352470	- Inferential Statistics	100%

Table 1: Member list & workload

1 Data Introduction

1.1 Dataset Overview

The central processing unit (CPU) is the backbone of modern computing systems, playing a crucial role in executing instructions and performing calculations necessary for all software and hardware interactions. As the primary processing component, the CPU is responsible for handling a wide range of tasks across personal and business environments. While it is widely recognized for its role in general computing, the CPU is also essential in areas such as data processing, system management, and running complex algorithms. [1]

To collect data on the parameters of a CPU, several methods can be employed. One common approach is using performance monitoring tools such as CPU-Z or HWMonitor, which provide real-time information on key metrics like clock speed, temperature, and core usage. Additionally, Benchmarking software like Cinebench or Geekbench can also be used to evaluate the CPU's performance under load.

The goal of this project is to analyze specific characteristics of CPUs as presented in the provided dataset. Table 2 below summarizes the CPU dataset that we are working on.

Properties	**Information**
Data Source	Intel_CPUs.csv
Population	Collection of CPUs
Number of observation	2283
Number of features	45

Table 2: Overview of CPU dataset highlighting number of observations and features

1.2 Variables Overview

For the overview of each variable, Table 3, 4 and 5 summarize the variables in the dataset.

NO.	**Feature**	**Data Type**	**Unit**	**Description**
1	Product Collection	Categorical		A series of processors that share similar features, specifications, ...
2	Vertical Segment	Categorical		A specific market or industry category where products are targeted to meet the needs of consumers or businesses.
3	Processor Number	Categorical		A unique identifier or model number assigned to a specific processor.
4	Status	Categorical		The lifecycle stage.
5	Launch Date	Categorical		The official release date.
6	Lithography	Continuous	nm	The smallest possible feature size that can be created on the chip.
7	Recommended Customer Price	Continuous	$	Suggested retail price set by the manufacturer or vendor.
8	number of Cores	Discrete		Number of physical cores in the processor.

Table 3: Features Overview

NO.	Feature	Data Type	Unit	Description
9	number of Threads	Discrete		Number of execution threads that the processor can handle simultaneously.
10	Processor Base Frequency	Continuous	GHz, MHz	Default operating speed of a CPU.
11	Max Turbo Frequency	Continuous	GHz	The highest speed a CPU can achieve under Turbo Boost Mode.
12	Cache	Categorical		Small, high-speed storage area that holds frequently accessed data.
13	Bus_Speed	Continuous	GT/s	The speed at which data is transferred between different components of the computer.
14	TDP	Continuous	W	The maximum amount of heat CPU generates under full load that the cooling system is designed to dissipate.
15	Embedded Options Available	Categorical		Availability of specific configurations or variants of a processor designed for embedded systems.
16	Conflict Free	Categorical		A set of materials or processes that do not involve "conflict minerals".
17	Max Memory Size	Continuous	GB, TB	Maximum amount of RAM.
18	Memory Types	Categorical		Different kinds of RAM.
19	Max nb of Memory Channels	Discrete		The maximum number of independent data channels through which the CPU can communicate with system memory (RAM).
20	Max Memory Bandwidth	Continuous	Gb/s	Maximum amount of data that can be transferred between CPU and RAM.
21	ECC Memory Supported	Categorical		Whether a system or processor can work with Error-Correcting Code (ECC) memory.
22	Processor Graphics	Categorical		The integrated graphics processing unit (iGPU) that is built into the CPU.
23	Graphics Base Frequency	Continuous	MHz	The standard or default clock speed at which the GPU operates under normal or idle conditions.
24	Graphics Max Dynamic Frequency	Continuous	MHz, GHz	maximum clock speed that a iGPU can reach under load, when the system demands extra performance.
25	Graphics Video Max Memory	Continuous	GB, MB	The maximum amount of memory that a GPU can use for storing textures, buffers, etc.

Table 4: Features Overview

NO.	Feature	Data Type	Unit	Description
26	Graphics Output	Categorical		Type of video or display signal sending to a display device.
27	Support 4k	Categorical		Capability to handle 4K resolution.
28	Max Resolution HDMI	Categorical		The highest display resolution through the HDMI.
29	Max Resolution DP	Categorical		The maximum display resolution through the DisplayPort.
30	Max Resolution eDP Integrated Flat Panel	Categorical		The highest display resolution that can be supported by an Embedded DisplayPort connection on an integrated flat panel display.
31	DirectX Support	Categorical		Ability to work with DirectX, a collection of APIs.
32	OpenGL Support	Categorical		Ability to work with OpenGL (Open Graphics Library).
33	PCI Express Revision	Categorical		The specific version or generation of the PCI Express (PCIe).
34	PCI Express Configurations	Categorical		The different ways in which PCIe lanes can be arranged and allocated across devices on the motherboard.
35	Max nb of PCI Express Lanes	Discrete		The maximum number of PCIe lanes.
36	T	Continuous	°C	The operating temperature or thermal limit of CPU.
37	Intel Hyper Threading Technology	Categorical		A feature that allows a single physical CPU core to behave like two separate logical cores.
38	Intel Virtualization Technology_VTx	Categorical		A hardware-based technology that allows a single processor to act as if it were multiple separate processors.
39	Intel 64	Categorical		A technology used for virtualization that enables direct access to hardware devices for virtual machines.
40	Instruction Set	Categorical	bit	Size of the data it can handle in one operation.
41	Instruction Set Extensions	Categorical		The set of instructions that a processor understands and executes.
42	Idle States	Categorical		The power-saving modes CPU enters when it is not performing tasks.
43	Thermal Monitoring Technologies	Categorical		Features built into CPUs to manage and control temperature.
44	Secure Key	Categorical		A security feature provides a secure environment for sensitive data.
45	Execute Disable Bit	Categorical		A security feature that helps protect against types of malware and attacks.

Table 5: Features Overview

2 Background

2.1 Multiple Linear Regression model

Multiple linear regression (MLR) is a statistical technique used to model the relationship between a dependent variable (target) and multiple independent variables (predictors or features) to predict the outcome of a response variable.[2] Assumptions of MLR include:

- **Homoscedasticity**: The variance of the error terms should remain constant across the values of the independent variables.
- **Independence of observations**: The observations must be independent of each other. If two independent variables exhibit a high correlation ($r^2 > 0.6$), only one should be included in the regression model.
- **Normality**: The residuals, which are the differences between the observed and predicted values of the dependent variable, should follow a normal distribution.
- **Linearity**: A linear relationship must exist between the dependent variable and each independent variable. This can be verified using scatter plots or partial regression plots.
- **No multicollinearity**: Multicollinearity exists when there are strong dependencies among the independent variables $x_1, x_2, \ldots, x_n$. Although the estimates of the regression coefficients are very imprecise when multicollinearity is present, the fitted model equation may still be useful when it comes to interpolations.

Equation 1 formulates the multiple linear regression model as below:

$$Y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + \epsilon \tag{1}$$

Where, for $i = 1, 2, \ldots, n$ observations:

- Y: dependent variable
- $x_1, x_2, \ldots, x_n$: explanatory variables (independent variables)
- β_0: y-intercept (the value of Y when all $x_i = 0$)
- β_i: slope coefficients for the corresponding explanatory variable x_i
- ϵ: the model's error term (residuals)

Note, any regression model that is linear in parameters (the β's) is a linear regression model. For example, the model $Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$ is non-linear in the x variable, but it is still linear in the parameters β_0, β_1 and β_2.
The parameters of the Linear regression model are usually estimated by *Least squares method.* Suppose there are k observations and the number of independ variables is n. Denote x_{ij} the ith observation of variable x_j, the observations are

$$(x_{i1}, x_{i2}, \ldots, x_{ik}, y_i), i = 1, 2, \ldots, n$$

In least squares method, we want to minimize the sum of the squared residuals ϵ_i - the differences between the observation y_i and the fitted value $\hat{y}_i$, which means the minimization of the following Equation 2.

$$L = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{k} \beta_j x_{ij} \right)^2 \tag{2}$$

After finding a fitted model, the accuracy of MLR model can be evaluated by the following metrics:

- **R-squared**: indicate the amount of variability in the data explained or accounted for by the regression model. R-squared will always increase if we add a variable to the model, but this does not necessarily imply that the new model is superior. This problem can be addressed by the adjusted R-squared.
- **Mean squared error (MSE)**: is the mean squared differences between the observed value and the predicted value.
- **Root mean squared error (RMSE)**: is the square root of MSE

2.2 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial initial step in data science projects. It involves analyzing and visualizing data to understand its key characteristics, uncover patterns, and identify relationships between variables refers to the method of studying and exploring record sets to apprehend their predominant traits, discover patterns, locate outliers, and identify relationships between variables. EDA is normally carried out as a preliminary step before undertaking extra formal statistical analyses or modeling.

2.2.1 Correlogram

A correlogram (also called a correlation plot) is a type of chart that allows the analysis of the relationship between each pair of numeric variables of a dataset. The relationship between each pair of variables is visualized through a scatter plot, or a symbol that represents the correlation coefficient r_{XY} (e.g., bubble, line, number, etc.). [3]

The range of r_{XY}: $-1 \leq r_{XY} \leq 1$

- $-1 \leq r_{XY} < 0$: negative correlation. r_{XY} is closer to -1, indicating a stronger negative correlation between X and Y.
- $0 < r_{XY} \leq 1$: positive correlation. r_{XY} is closer to 1, indicating a stronger positive correlation between X and Y.
- r_{XY} is closer to 0, indicating a weak correlation between X and Y. $r_{XY} = 0$: indicating linear independence between X and Y.

A correlogram can help provide answers to the following questions:

1. Is the data random?
2. Is an observation related to an adjacent observation?
3. What is an appropriate model for the observed time series?

In this assignment, we use the default method of the function `cor()` in R for calculating Pearson's correlation coefficient. The formula for r_{XY} in this case is shown in Equation 3.

$$r_{XY} = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2 \sum_{i=1}^{n}(Y_i - \overline{Y})^2}} \tag{3}$$

Where:

- X_i, Y_i are the individual data points for variables X and Y,
- $\overline{X}, \overline{Y}$ are the sample means of X and Y,
- n is the total number of data points.

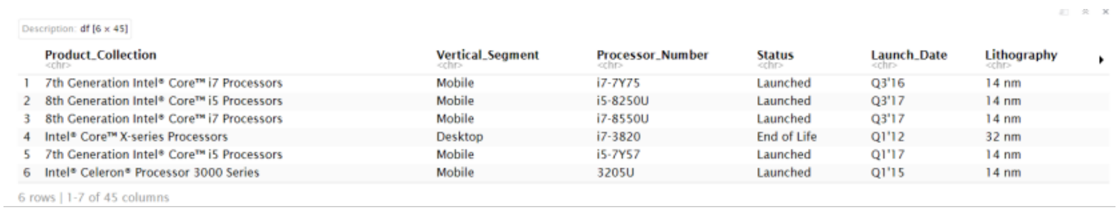
2.2.2 Other commonly used visualization tools

- **Quantile-quantile plot (Q-Q plot)** is used to determine if a dataset follows a certain probability distribution or whether two samples of data come from the same population or not. Q-Q plot are particularly useful for assessing the assumption of whether a dataset follow the normal distribution and some other known distribution.[4]
- **Histograms** are used to demonstrate how many of a certain type of variable occur within a specific range. They help us visualize the distribution. [5]
- **Scatter plots** show the relationship between one independent variable plotted on the x-axis and one dependent variable plotted on the y-axis. [6]
- **Boxplots** a graphical display that simultaneously describes several important features of a data set, such as center, spread, departure from symmetry, and identification of unusual observations or outliers.

3 Data Preprocessing

3.1 Data Reading

We started by going through all the information provided in the dataset. We then displayed a preview of the first few lines to get a quick look at what the data looks like. This helped us get familiar with the dataset and understand its structure before diving deeper into our analysis. The dataset is shown in Figure 1 and summarized in Figure 2. (We use the book *Applied Statistics and Probability for Engineers for references.*[7])

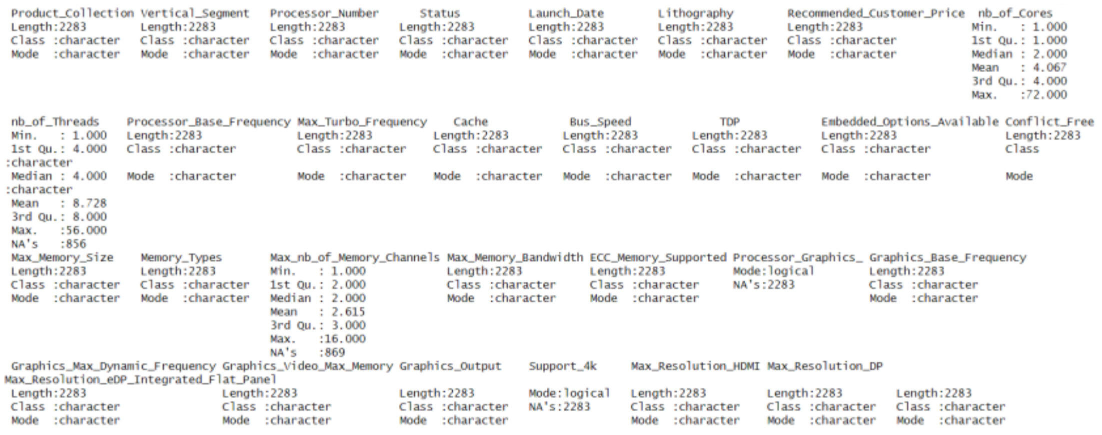


Description: df [6 × 45]

	Product_Collection <chr>	Vertical_Segment <chr>	Processor_Number <chr>	Status <chr>	Launch_Date <chr>	Lithography <chr>
1	7th Generation Intel® Core™ i7 Processors	Mobile	i7-7Y75	Launched	Q3'16	14 nm
2	8th Generation Intel® Core™ i5 Processors	Mobile	i5-8250U	Launched	Q3'17	14 nm
3	8th Generation Intel® Core™ i7 Processors	Mobile	i7-8550U	Launched	Q3'17	14 nm
4	Intel® Core™ X-series Processors	Desktop	i7-3820	End of Life	Q1'12	32 nm
5	7th Generation Intel® Core™ i5 Processors	Mobile	i5-7Y57	Launched	Q1'17	14 nm
6	Intel® Celeron® Processor 3000 Series	Mobile	3205U	Launched	Q1'15	14 nm

6 rows | 1-7 of 45 columns

Figure 1: Overview of Dataset



```
 Product_Collection Vertical_Segment   Processor_Number     Status          Launch_Date        Lithography        Recommended_Customer_Price  nb_of_Cores
 Length:2283        Length:2283        Length:2283        Length:2283        Length:2283        Length:2283        Length:2283                Min.   : 1.000
 Class :character   Class :character   Class :character   Class :character   Class :character   Class :character   Class :character           1st Qu.: 1.000
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character           Median : 2.000
                                                                                                                                             Mean   : 4.067
                                                                                                                                             3rd Qu.: 4.000
                                                                                                                                             Max.   :72.000

 nb_of_Threads    Processor_Base_Frequency Max_Turbo_Frequency    Cache             Bus_Speed            TDP            Embedded_Options_Available Conflict_Free
 Min.   : 1.000   Length:2283              Length:2283          Length:2283        Length:2283        Length:2283        Length:2283                Length:2283
 1st Qu.: 4.000   Class :character         Class :character     Class :character   Class :character   Class :character   Class :character           Class
:character
 Median : 4.000   Mode  :character         Mode  :character     Mode  :character   Mode  :character   Mode  :character   Mode  :character           Mode
:character
 Mean   : 8.728
 3rd Qu.: 8.000
 Max.   :56.000
 NA's   :856
 Max_Memory_Size    Memory_Types       Max_nb_of_Memory_Channels Max_Memory_Bandwidth ECC_Memory_Supported Processor_Graphics_ Graphics_Base_Frequency
 Length:2283        Length:2283        Min.   : 1.000            Length:2283          Length:2283          Mode:logical        Length:2283
 Class :character   Class :character   1st Qu.: 2.000            Class :character     Class :character     NA's:2283           Class :character
 Mode  :character   Mode  :character   Median : 2.000            Mode  :character     Mode  :character                         Mode  :character
                                       Mean   : 2.615
                                       3rd Qu.: 3.000
                                       Max.   :16.000
                                       NA's   :869
 Graphics_Max_Dynamic_Frequency Graphics_Video_Max_Memory Graphics_Output    Support_4k     Max_Resolution_HDMI Max_Resolution_DP
Max_Resolution_eDP_Integrated_Flat_Panel
 Length:2283                    Length:2283               Length:2283        Mode:logical   Length:2283         Length:2283        Length:2283
 Class :character               Class :character          Class :character   NA's:2283      Class :character    Class :character   Class :character
 Mode  :character               Mode  :character          Mode  :character                  Mode  :character    Mode  :character   Mode  :character
```

Figure 2: Summarization of Dataset

3.2 Examine the Percentage of Missing Values

To facilitate analysis, we will standardize all missing values by converting them to "NA". Currently, these missing values appear in various forms such as empty strings (""), newline characters ("\n"), and other formats. After this transformation, we intend to calculate and display the percentage of missing values for each variable in the dataset. This approach will provide a clearer picture of the data's completeness and help identify which variables may require further attention or imputation strategies. The percentage of missing values in those features is shown in Figure 3

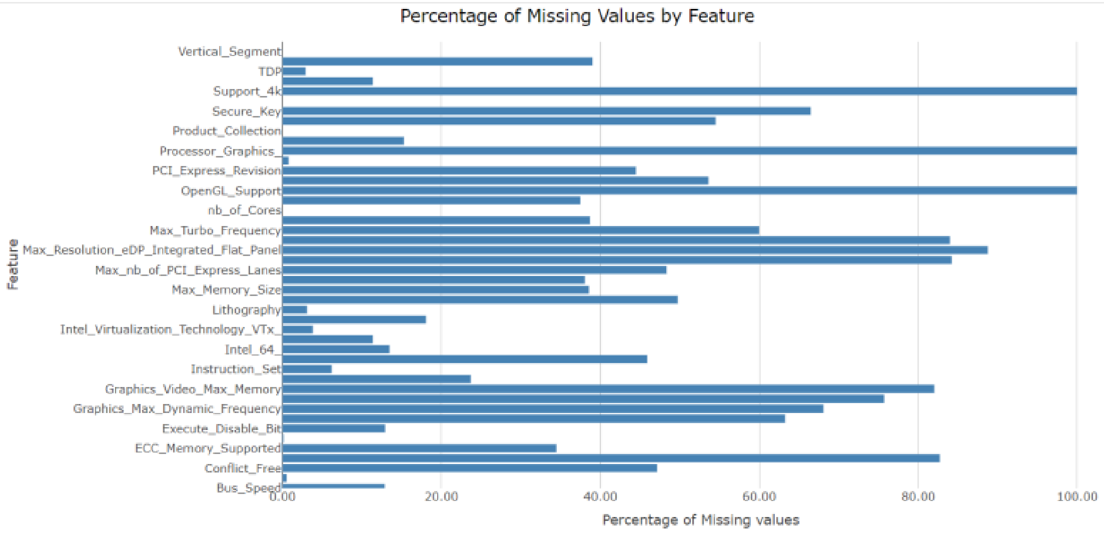


Figure 3: Percentage of Missing Values by Feature

As observed, certain features, such as *Processor_ Graphics*, *Support_ 4k*, *OpenGL_ Support*,

. . . , either lack values or have a high percentage of missing data. Therefore, we'll retain only those features with low percentages of missing data for the next part of this assignment, as they are likely the key factors influencing the final CPU product prices:

- *Product_ Collection*
- *Vertical_ Segment*
- *Status*
- *Launch_ Date*
- *Lithography*
- *Recommended_ Customer_ Price*
- *nb_ of_ Cores*
- *nb_ of_ Threads*
- *Processor_ Base_ Frequency*
- *Cache*
- *Instruction_ Set*
- *TDP*
- *Max_ Memory_ Size*
- *Max_ nb_ of_ Memory_ Channels*
- *Max_ Memory_ Bandwidth*

3.3 Data Transformation

When preparing data for descriptive analysis and statistical inference, it's crucial to refine and transform the selected features to enhance their usability. The primary objective is to segment the data within each feature into distinct, meaningful categories or ranges. This process of data transformation and categorization allows for more straightforward visualization and analysis. By organizing the data into well-defined groups or intervals, you can create more effective plots, charts, and graphs that clearly communicate patterns and trends. This approach not only simplifies the plotting process but also facilitates a more intuitive interpretation of the results, making it easier to draw insights and conduct further statistical analyses on the dataset. First, we have a helper function to help us transform the skewed distribution (i.e. the data is heavily leaned over one side, which may distort our model performance) into a more "normal", less skewed distribution. This useful method is **Log Transformation** defined in Equation 4. The purpose of adding 1 after each value is to ensure the result is larger than 0.

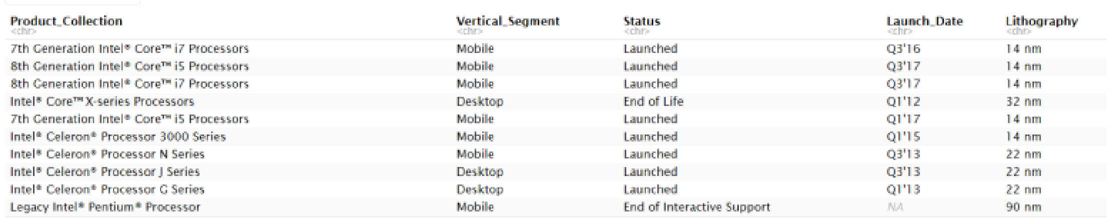
$$\mathcal{L} = \ln(X + 1) \tag{4}$$

a) In terms of **Product Collection**, we split and categorize the names into product mainlines, including *Celeron*, *Pentium*, *Quark*, *Atom*, *Itanium*, *Xeon*, *Core*. If the product has the name "Legacy", we add it before the product name as a prefix or subcategory.
While for **Launch Date**, we extract the year number of the product and then put it in a new column called *Year*; the rows that possess missing values or NAs will be dropped as they are unknown.

With **Lithography**, this feature's data can be easily formatted as the values are numeric in advance, therefore, what we only need to do is to remove the unit. Next, to deal with the missing values in this column, we will fill the lithography with the median values grouped accordingly to *Product_ Collection* and *Year*.
Finally, for **Number of Threads**, we can apply a similar data transformation process from *Lithography*, particularly utilizing the missing values filling technique of medians and dropping non-fillable rows.

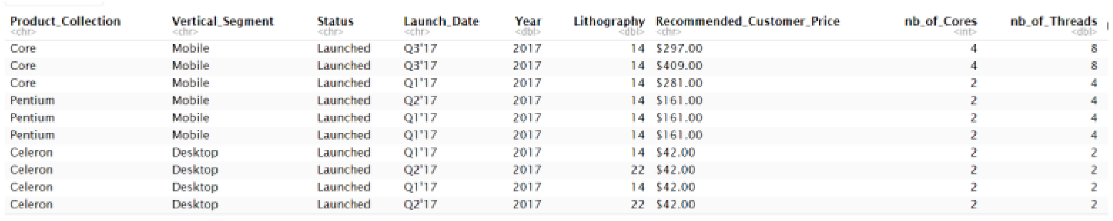
.

Figures 4 and 5 will show the dataset after the transformation of the above features.



Product_Collection <chr>	Vertical_Segment <chr>	Status <chr>	Launch_Date <chr>	Lithography <chr>
7th Generation Intel® Core™ i7 Processors	Mobile	Launched	Q3'16	14 nm
8th Generation Intel® Core™ i5 Processors	Mobile	Launched	Q3'17	14 nm
8th Generation Intel® Core™ i7 Processors	Mobile	Launched	Q3'17	14 nm
Intel® Core™ X-series Processors	Desktop	End of Life	Q1'12	32 nm
7th Generation Intel® Core™ i5 Processors	Mobile	Launched	Q1'17	14 nm
Intel® Celeron® Processor 3000 Series	Mobile	Launched	Q1'15	14 nm
Intel® Celeron® Processor N Series	Mobile	Launched	Q3'13	22 nm
Intel® Celeron® Processor J Series	Desktop	Launched	Q3'13	22 nm
Intel® Celeron® Processor G Series	Desktop	Launched	Q1'13	22 nm
Legacy Intel® Pentium® Processor	Mobile	End of Interactive Support	NA	90 nm

Figure 4: Dataset before Data Transformation a), Null Values Replacement and Fill



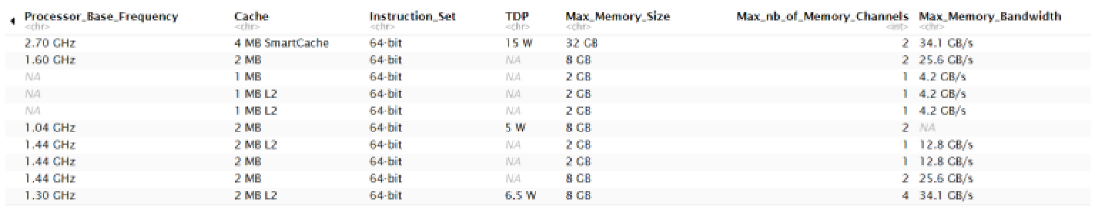
Product_Collection <chr>	Vertical_Segment <chr>	Status <chr>	Launch_Date <chr>	Year <dbl>	Lithography <dbl>	Recommended_Customer_Price <chr>	nb_of_Cores <int>	nb_of_Threads <dbl>
Core	Mobile	Launched	Q3'17	2017	14	$297.00	4	8
Core	Mobile	Launched	Q3'17	2017	14	$409.00	4	8
Core	Mobile	Launched	Q1'17	2017	14	$281.00	2	4
Pentium	Mobile	Launched	Q2'17	2017	14	$161.00	2	4
Pentium	Mobile	Launched	Q1'17	2017	14	$161.00	2	4
Pentium	Mobile	Launched	Q1'17	2017	14	$161.00	2	4
Celeron	Desktop	Launched	Q1'17	2017	14	$42.00	2	2
Celeron	Desktop	Launched	Q2'17	2017	22	$42.00	2	2
Celeron	Desktop	Launched	Q1'17	2017	14	$42.00	2	2
Celeron	Desktop	Launched	Q2'17	2017	22	$42.00	2	2

Figure 5: Dataset after Data Transformation a), Null Values Replacement and Fill

b) **Processor Base Frequency**: There are two frequency units in this feature, *MHz* and *GHz*. For easier computation, we convert both into one base standard unit, *MHz*. Missing values are then filled with the median frequency for each Vertical Segment, ensuring a more complete and consistent dataset for further analysis.

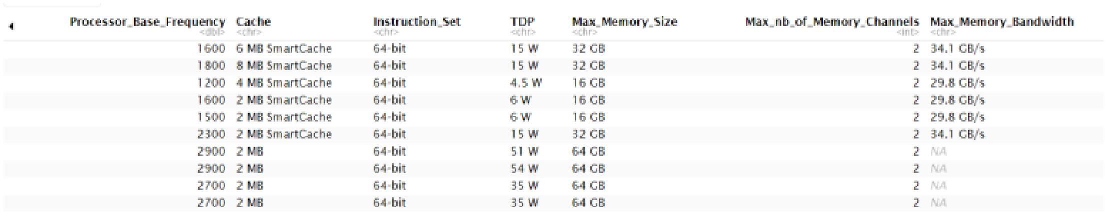
While with **Cache** we need to create a new column to store the type of cache. We drop all NA values and convert the remaining data into a standard unit (KB). There are 5 different types of cache: *Smart Cache*, *L3*, *L2*, *Last Level Cache*, *Blank*. To efficiently deal with these values and normalize them accordingly, we remove the trailing words after the standard units (*MB* and *KB*) and convert the values into one standard unit: *KB*. We then normalize them using a log transformation as shown in Equation 4, as there are both large and small numbers. Then, the types of cache will be split into another column and the *Blank* cache type will be replaced with *Normal*.

Figures 6 and 7 show the transformation of the above features on the dataset.



Processor_Base_Frequency	Cache	Instruction_Set	TDP	Max_Memory_Size	Max_nb_of_Memory_Channels	Max_Memory_Bandwidth
<chr>	<chr>	<chr>	<chr>	<chr>	<int>	<chr>
2.70 GHz	4 MB SmartCache	64-bit	15 W	32 GB	2	34.1 GB/s
1.60 GHz	2 MB	64-bit	NA	8 GB	2	25.6 GB/s
NA	1 MB	64-bit	NA	2 GB	1	4.2 GB/s
NA	1 MB L2	64-bit	NA	2 GB	1	4.2 GB/s
NA	1 MB L2	64-bit	NA	2 GB	1	4.2 GB/s
1.04 GHz	2 MB	64-bit	5 W	8 GB	2	NA
1.44 GHz	2 MB L2	64-bit	NA	2 GB	1	12.8 GB/s
1.44 GHz	2 MB	64-bit	NA	2 GB	1	12.8 GB/s
1.44 GHz	2 MB	64-bit	NA	8 GB	2	25.6 GB/s
1.30 GHz	2 MB L2	64-bit	6.5 W	8 GB	4	34.1 GB/s

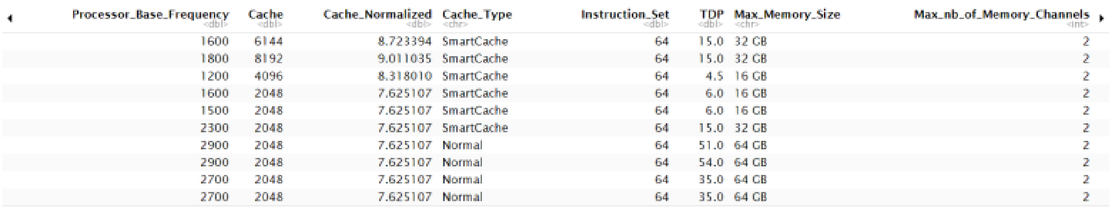
Figure 6: Dataset before Data Transformation b), Null Values Replacement and Fill



Processor_Base_Frequency	Cache	Instruction_Set	TDP	Max_Memory_Size	Max_nb_of_Memory_Channels	Max_Memory_Bandwidth
<dbl>	<chr>	<chr>	<chr>	<chr>	<int>	<chr>
1600	6 MB SmartCache	64-bit	15 W	32 GB	2	34.1 GB/s
1800	8 MB SmartCache	64-bit	15 W	32 GB	2	34.1 GB/s
1200	4 MB SmartCache	64-bit	4.5 W	16 GB	2	29.8 GB/s
1600	2 MB SmartCache	64-bit	6 W	16 GB	2	29.8 GB/s
1500	2 MB SmartCache	64-bit	6 W	16 GB	2	29.8 GB/s
2300	2 MB SmartCache	64-bit	15 W	32 GB	2	34.1 GB/s
2900	2 MB	64-bit	51 W	64 GB	2	NA
2900	2 MB	64-bit	54 W	64 GB	2	NA
2700	2 MB	64-bit	35 W	64 GB	2	NA
2700	2 MB	64-bit	35 W	64 GB	2	NA

Figure 7: Dataset after Data Transformation b), Null Values Replacement and Fill

Instruction Set and **TDP** we both trim all the characters to only extract the numeric values but in terms of dealing NA values we will fill the missing values based on the trend of the group for the former and drop all the rows with missing values for the latter (as the number of missing values is minimal and has trivial effects on the overall performance of the model). The result is shown in Figure 8.



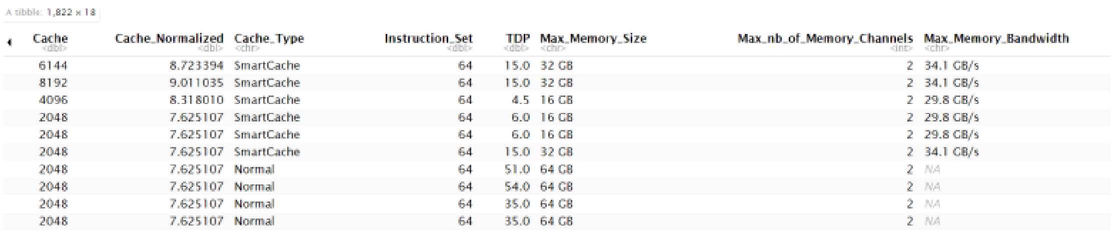
Processor_Base_Frequency	Cache	Cache_Normalized	Cache_Type	Instruction_Set	TDP	Max_Memory_Size	Max_nb_of_Memory_Channels
<dbl>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<chr>	<int>
1600	6144	8.723394	SmartCache	64	15.0	32 GB	2
1800	8192	9.011035	SmartCache	64	15.0	32 GB	2
1200	4096	8.318010	SmartCache	64	4.5	16 GB	2
1600	2048	7.625107	SmartCache	64	6.0	16 GB	2
1500	2048	7.625107	SmartCache	64	6.0	16 GB	2
2300	2048	7.625107	SmartCache	64	15.0	32 GB	2
2900	2048	7.625107	Normal	64	51.0	64 GB	2
2900	2048	7.625107	Normal	64	54.0	64 GB	2
2700	2048	7.625107	Normal	64	35.0	64 GB	2
2700	2048	7.625107	Normal	64	35.0	64 GB	2

Figure 8: Dataset after Data Transformation, Null Values Replacement and Fill

c) The **Memory** feature is divided into three distinct subcategories, each requiring its own data transformation approach. For the *Max Memory Size*, all values are standardized to gigabytes (GB). Due to discrepancies between converted and base values, a log normalization (Eq. 4) with an offset of 1 is applied, and the missing values are filled with the median grouped by *Product_ Collection* and *Vertical Segment*.

The *Max Number of Channels* and *Max Memory Bandwidth* undergo similar treatments for missing values, which are imputed using the median value grouped by *Product_ Collection* and *Vertical Segment*. However, the *Max Memory Bandwidth* column requires an additional step of removing units before imputation.

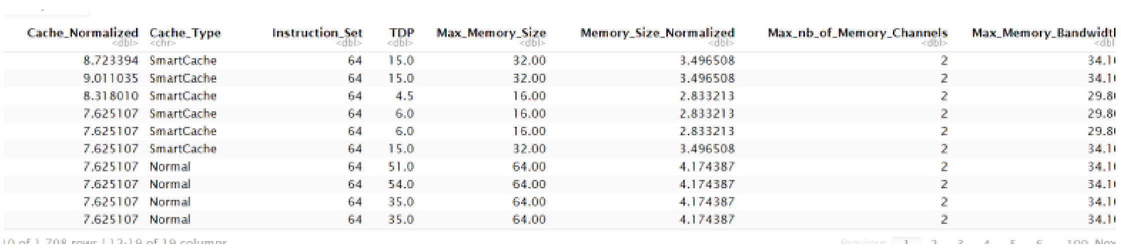
The modifications of the above features are shown in Figure 9 and Figure 11.



A tibble: 1,822 × 18

Cache <dbl>	Cache_Normalized <dbl>	Cache_Type <chr>	Instruction_Set <dbl>	TDP <dbl>	Max_Memory_Size <chr>	Max_nb_of_Memory_Channels <int>	Max_Memory_Bandwidth <chr>
6144	8.723394	SmartCache	64	15.0	32 GB	2	34.1 GB/s
8192	9.011035	SmartCache	64	15.0	32 GB	2	34.1 GB/s
4096	8.318010	SmartCache	64	4.5	16 GB	2	29.8 GB/s
2048	7.625107	SmartCache	64	6.0	16 GB	2	29.8 GB/s
2048	7.625107	SmartCache	64	6.0	16 GB	2	29.8 GB/s
2048	7.625107	SmartCache	64	15.0	32 GB	2	34.1 GB/s
2048	7.625107	Normal	64	51.0	64 GB	2	NA
2048	7.625107	Normal	64	54.0	64 GB	2	NA
2048	7.625107	Normal	64	35.0	64 GB	2	NA
2048	7.625107	Normal	64	35.0	64 GB	2	NA

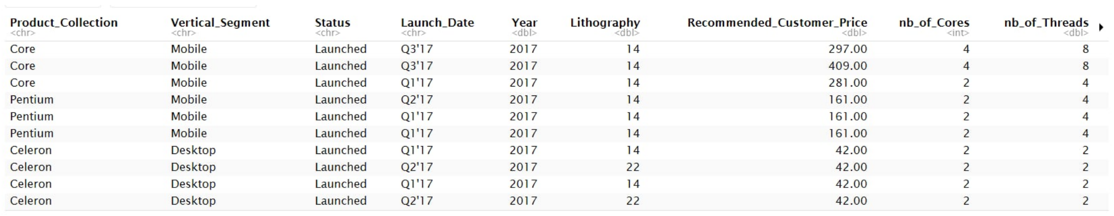
Figure 9: Dataset before Data Transformation c), Null Values Replacement and Fill



Cache_Normalized <dbl>	Cache_Type <chr>	Instruction_Set <dbl>	TDP <dbl>	Max_Memory_Size <dbl>	Memory_Size_Normalized <dbl>	Max_nb_of_Memory_Channels <dbl>	Max_Memory_Bandwidtl <dbl
8.723394	SmartCache	64	15.0	32.00	3.496508	2	34.1
9.011035	SmartCache	64	15.0	32.00	3.496508	2	34.1
8.318010	SmartCache	64	4.5	16.00	2.833213	2	29.8
7.625107	SmartCache	64	6.0	16.00	2.833213	2	29.8
7.625107	SmartCache	64	6.0	16.00	2.833213	2	29.8
7.625107	SmartCache	64	15.0	32.00	3.496508	2	34.1
7.625107	Normal	64	51.0	64.00	4.174387	2	34.1
7.625107	Normal	64	54.0	64.00	4.174387	2	34.1
7.625107	Normal	64	35.0	64.00	4.174387	2	34.1
7.625107	Normal	64	35.0	64.00	4.174387	2	34.1

Figure 10: Dataset after Data Transformation c), Null Values Replacement and Fill

d) Finally is for *Recommended_ Customer_ Price*, We remove dollar signs from the price values using gsub to prepare the data for numerical operations. For entries containing price ranges (indicated by a hyphen), the code calculates the mean of the range endpoints using a custom function applied through sapply, ensuring all prices are represented as single numeric values. The code then handles missing values by grouping the data by *Product_ Collection* and using the fill function to propagate existing values both upward and downward within each group. Finally, the code converts all price values to the double data type for consistency and prints the transformed dataset, completing the preprocessing steps necessary for further analysis.

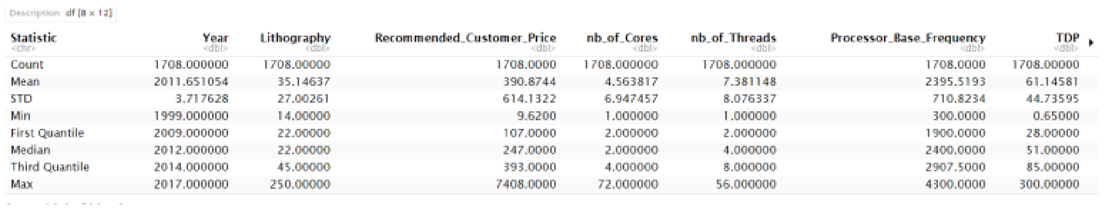


Product_Collection <chr>	Vertical_Segment <chr>	Status <chr>	Launch_Date <chr>	Year <dbl>	Lithography <dbl>	Recommended_Customer_Price <dbl>	nb_of_Cores <int>	nb_of_Threads <dbl>
Core	Mobile	Launched	Q3'17	2017	14	297.00	4	8
Core	Mobile	Launched	Q3'17	2017	14	409.00	4	8
Core	Mobile	Launched	Q1'17	2017	14	281.00	2	4
Pentium	Mobile	Launched	Q2'17	2017	14	161.00	2	4
Pentium	Mobile	Launched	Q1'17	2017	14	161.00	2	4
Pentium	Mobile	Launched	Q1'17	2017	14	161.00	2	4
Celeron	Desktop	Launched	Q1'17	2017	14	42.00	2	2
Celeron	Desktop	Launched	Q2'17	2017	22	42.00	2	2
Celeron	Desktop	Launched	Q1'17	2017	14	42.00	2	2
Celeron	Desktop	Launched	Q2'17	2017	22	42.00	2	2

Figure 11: Dataset after Data Transformation d), Null Values Replacement and Fill

4 Descriptive Statistics

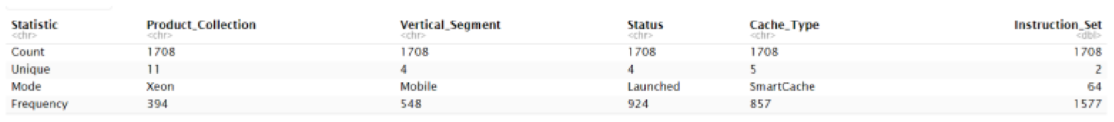
We will split the processed data frame into two tables of Categorical and Numerical data for future use. The table of categorical data includes: *Product_ Collection, Vertical_ Segment, Status, Cache_ Type, Instruction_ Set* and the rest are in the table of Numerical data. For the numerical data, we calculate various statistics including count, mean, standard deviation, minimum, first quartile, median, third quartile, and maximum for each numerical variable. For the categorical data, we compute the count of non-NA values, the number of unique values, the mode (most frequent value), and the frequency of the mode for each categorical variable. Figures 12 and 13 below show the statistics of both.



Description: df [8 x 12]

Statistic <chr>	Year <dbl>	Lithography <dbl>	Recommended_Customer_Price <dbl>	nb_of_Cores <dbl>	nb_of_Threads <dbl>	Processor_Base_Frequency <dbl>	TDP <dbl>
Count	1708.000000	1708.00000	1708.0000	1708.000000	1708.000000	1708.0000	1708.00000
Mean	2011.651054	35.14637	390.8744	4.563817	7.381148	2395.5193	61.14581
STD	3.717628	27.00261	614.1322	6.947457	8.076337	710.8234	44.73595
Min	1999.000000	14.00000	9.6200	1.000000	1.000000	300.0000	0.65000
First Quantile	2009.000000	22.00000	107.0000	2.000000	2.000000	1900.0000	28.00000
Median	2012.000000	22.00000	247.0000	2.000000	4.000000	2400.0000	51.00000
Third Quantile	2014.000000	45.00000	393.0000	4.000000	8.000000	2907.5000	85.00000
Max	2017.000000	250.00000	7408.0000	72.000000	56.000000	4300.0000	300.00000

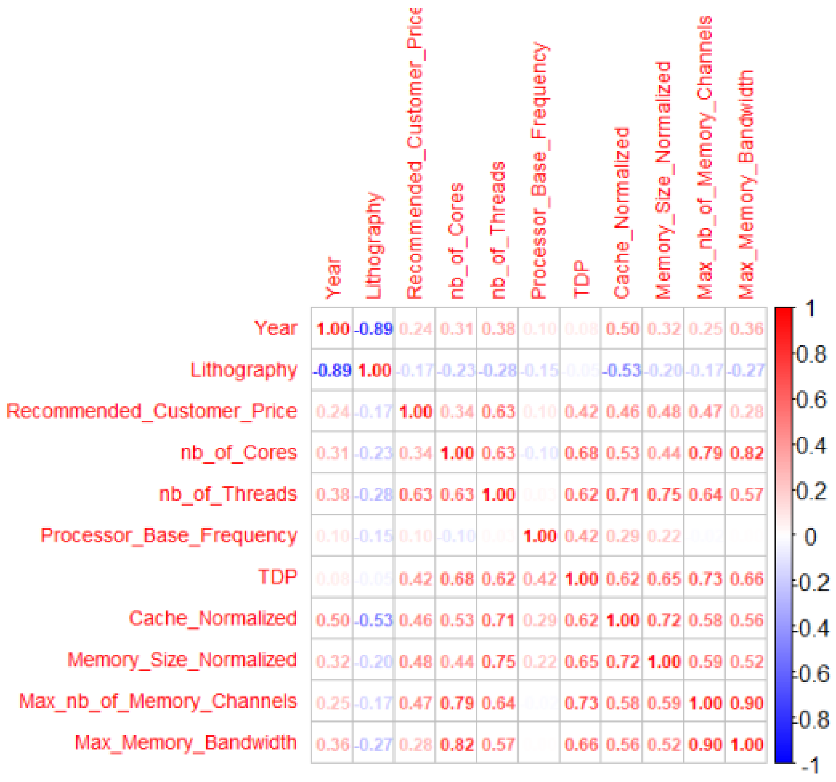
Figure 12: Statistics of Categorical Dataset



Statistic <chr>	Product_Collection <chr>	Vertical_Segment <chr>	Status <chr>	Cache_Type <chr>	Instruction_Set <dbl>
Count	1708	1708	1708	1708	1708
Unique	11	4	4	5	2
Mode	Xeon	Mobile	Launched	SmartCache	64
Frequency	394	548	924	857	1577

Figure 13: Statistics of Numerical Dataset

4.1 Correlation Matrix



	Year	Lithography	Recommended_Customer_Price	nb_of_Cores	nb_of_Threads	Processor_Base_Frequency	TDP	Cache_Normalized	Memory_Size_Normalized	Max_nb_of_Memory_Channels	Max_Memory_Bandwidth
Year	1.00	-0.89	0.24	0.31	0.38	0.10	0.08	0.50	0.32	0.25	0.36
Lithography	-0.89	1.00	-0.17	-0.23	-0.28	-0.15	-0.05	-0.53	-0.20	-0.17	-0.27
Recommended_Customer_Price	0.24	-0.17	1.00	0.34	0.63	0.10	0.42	0.46	0.48	0.47	0.28
nb_of_Cores	0.31	-0.23	0.34	1.00	0.63	-0.10	0.68	0.53	0.44	0.79	0.82
nb_of_Threads	0.38	-0.28	0.63	0.63	1.00	0.03	0.62	0.71	0.75	0.64	0.57
Processor_Base_Frequency	0.10	-0.15	0.10	-0.10	0.03	1.00	0.42	0.29	0.22	-0.02	[illegible]
TDP	0.08	-0.05	0.42	0.68	0.62	0.42	1.00	0.62	0.65	0.73	0.66
Cache_Normalized	0.50	-0.53	0.46	0.53	0.71	0.29	0.62	1.00	0.72	0.58	0.56
Memory_Size_Normalized	0.32	-0.20	0.48	0.44	0.75	0.22	0.65	0.72	1.00	0.59	0.52
Max_nb_of_Memory_Channels	0.25	-0.17	0.47	0.79	0.64	-0.02	0.73	0.58	0.59	1.00	0.90
Max_Memory_Bandwidth	0.36	-0.27	0.28	0.82	0.57	[illegible]	0.66	0.56	0.52	0.90	1.00

Figure 14: Correlation Matrix Visualization based on the coefficient scale

The correlation matrix shown in 14 gives us information about:

Year: This variable has a strong negative correlation with Lithography (-0.89) and positive correlation with Cache_Normalized (0.5), indicating that as technology evolves over the years, lithography tends to decrease, while showing improvement in cache technology over

Recommended Customer Price: The feature demonstrates a trend where the price tends to increase as other features increase, with the exception of Lithography. However, it's worth noting that the features positively influencing the price don't show very strong correlations. The highest correlation observed is 0.63, while the others range between 0.63 and 0.1. This suggests a moderate to weak relationship between most features and the price.

Number of Cores and Threads: Cores and Threads exhibit a robust positive correlation (0.63) with each other, which aligns with expectations given their intrinsic connection. Furthermore, both Cores and Threads demonstrate a moderate positive correlation with Cache Normalized, with coefficients of 0.68 and 0.63 respectively. This correlation suggests that as the number of cores or threads increases, there is a corresponding rise in cache utilization.

TDP: This feature is affected by all the features (>42 %) except Year and Lithography showing that higher performance generally results in higher TDP, reflecting the balance between processing power and energy consumption.

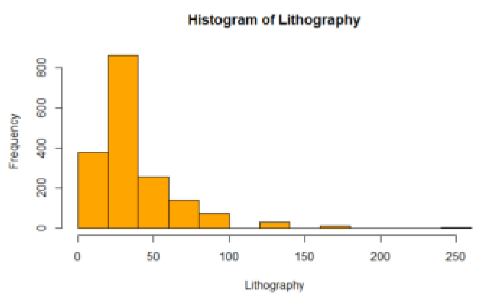
Processor Base Frequency: This variable has positive correlations that are slightly affected by other variables.

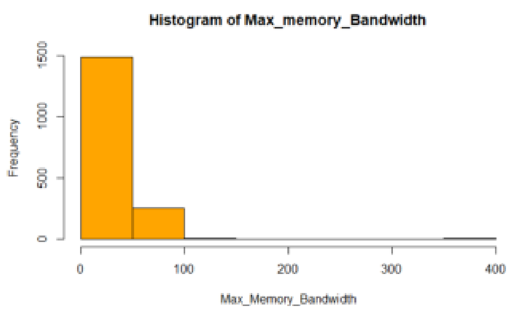
Lithography: The evolution of CPU lithography demonstrates technological advancement through miniaturization. As the lithography process becomes more refined, it allows for smaller transistor sizes, leading to more compact CPU designs. This reduction in size not only affects the processor's core components but also influences cache memory. The correlation between advanced lithography techniques and decreased cache size suggests that improvements in manufacturing processes enable more efficient cache designs, potentially offering better performance in a smaller footprint.

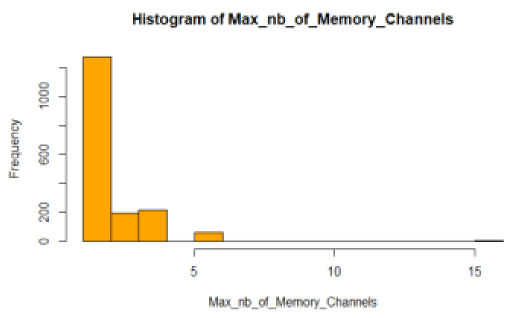
Cache: There is a moderately negative correlation with Lithography (-0.53), highlighting how improved caching can parallel with other features as well as ratio with the size of the CPU.

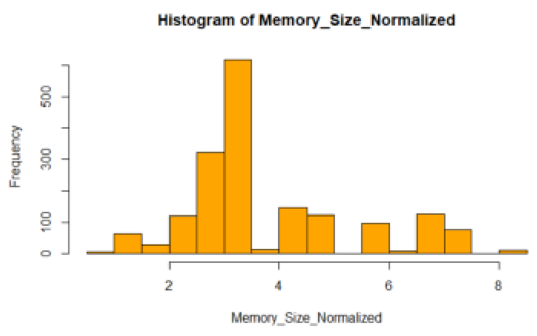
Memory: All of the three are not only highly positive with each other but also positively correlate with other features except Lithography

4.2 Histogram









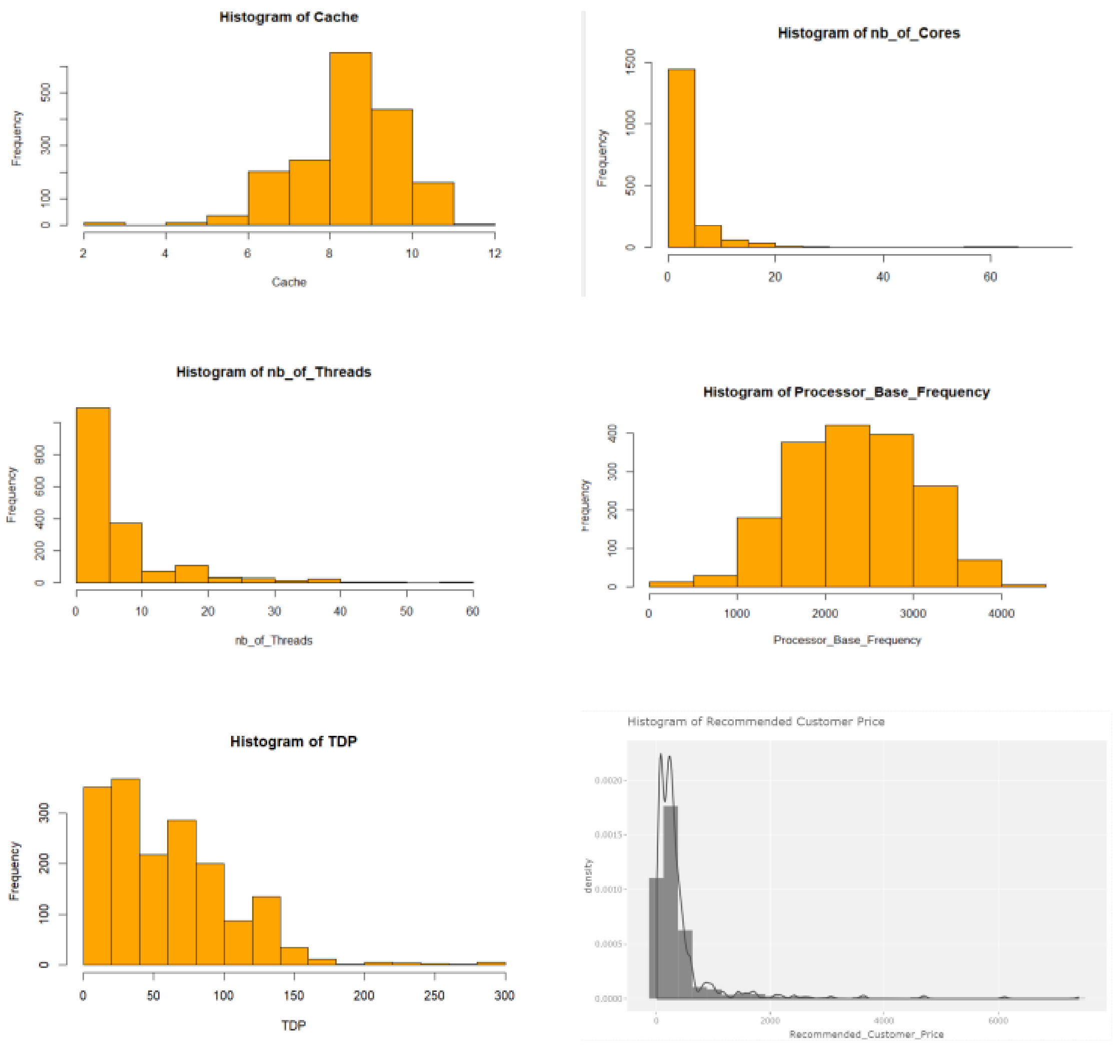


Figure 15: Histogram of all continuous variables

We plot all the continuous variables histogram in Figure 15 and the result of each graph is presented as:

Histogram of Lithography: The size of CPU ranges from 14 to 250 and the median is 22 which is more than its mean (35.14), so the distribution is right-skewed.

Histogram of nb_of_cores_and_threads: The distribution of number of cores and threads share the same variation in the graph. Their median is smaller than its mean which are 2 and 4 smaller than 4.56 and 7.38 for nb_of_cores and nb_of_threads respectively, so the distribution of them are right-skewed.

Histogram of Max_memory_Bandwidth and Memory_Channels: The graph reveals a similar pattern of variation for both the maximum memory channel and maximum memory bandwidth distributions. These distributions exhibit right-skewed characteristics, as evidenced by their median values being lower than their respective means. Specifically, for the number of cores, the median is 2 units smaller than the mean of 2.52, while for the number of threads, the median is 25.6 units less than the mean of 31.97.

Histogram of Processor Base Frequency: The histogram depicting Processor Base Frequency exhibits a unique range, spanning from 300 to 4300. A notable characteristic of this distribution is the striking similarity between its mean (2395.5) and median (2400) values. The data points appear to cluster around the median, forming a symmetrical pattern. This concentration of values near the center, coupled with the close proximity of the mean and median, strongly suggests that the distribution follows a normal distribution pattern.
Histogram of Recommend Customer price: This graph reveals a highly right-skewed distribution. Most data points are concentrated at the lower end of the price scale, indicating that the majority of customers are recommended relatively low prices. There is a long tail extending towards higher values, suggesting that fewer recommendations are made at these price points. Noticeable peaks around the lower prices highlight a high concentration in this range, while the spread towards the right indicates possible outliers, with some recommended prices being significantly higher.
Histogram of Cache: This is the only graph in all of the histograms that is left-skewed with the mean of 8.4 and the median is 8.31. You can easily see in the graph that it is slightly left-skewed. This shows that the majority of Cache values are concentrated around or slightly above the median.
Histogram of Memory_Size_Normalize: The most prominent peak is centered around 3, suggesting that a significant portion of the data is concentrated at this value. Smaller peaks appear at other intervals, varying from 4 to 8 but not as high as that of varying from 2 to 4.
Histogram of TDP: The plot shows most data points cluster at lower TDP values. The highest frequency is observed within the 0 to 50 range, suggesting that a significant portion of the dataset consists of these lower values. As the TDP values increase, there is a noticeable decline in frequency, with few occurrences beyond 150. The data spans from 0 to just under 300, but values beyond 100 are relatively sparse.

4.3 Variable trend

4.3.1 Box Plot

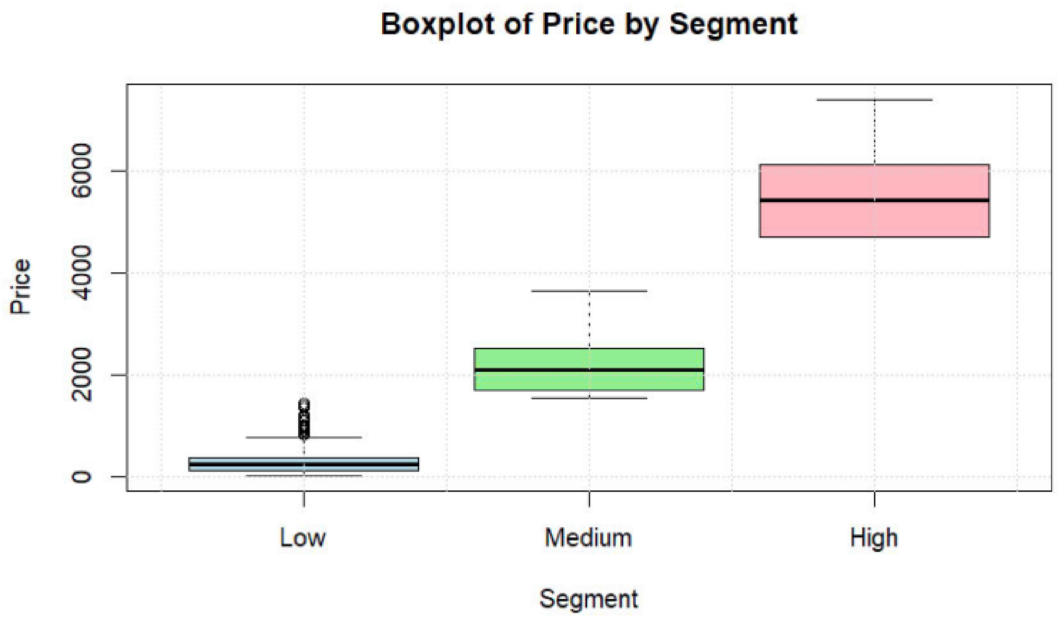
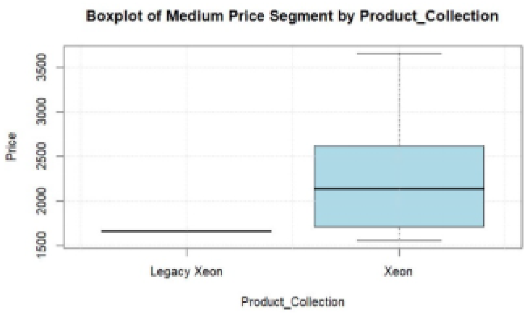


Figure 16: Boxplot of Price by Segment

Figure 16 highlights the distribution of prices across three segments: **Low**, **Medium**, and **High**. The **Low segment** (blue) shows the lowest price concentration, with minimal variability and some outliers. This indicates a majority of items in this category are clustered near the lower end, with a few higher-priced exceptions. The **Medium segment** (green) displays a moderate price range, with a broader spread compared to the Low segment, indicating more variability within this group. Prices are centered between 2,000 and 4,000 units. The **High segment** (pink) has the widest price range and consistently higher values, with no outliers. This shows that items in this category are generally premium and uniform in their pricing structure. Overall, the data suggests a clear stratification of prices across segments, with higher segments consistently associated with greater cost and range.



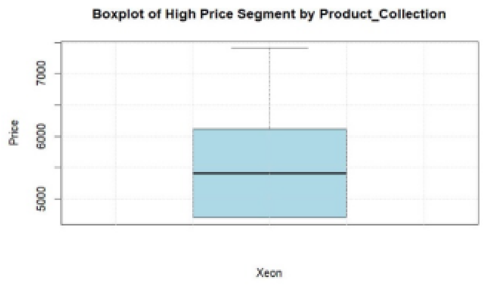


Figure 17: Boxplot of Medium and High Price Segment by Product Collection

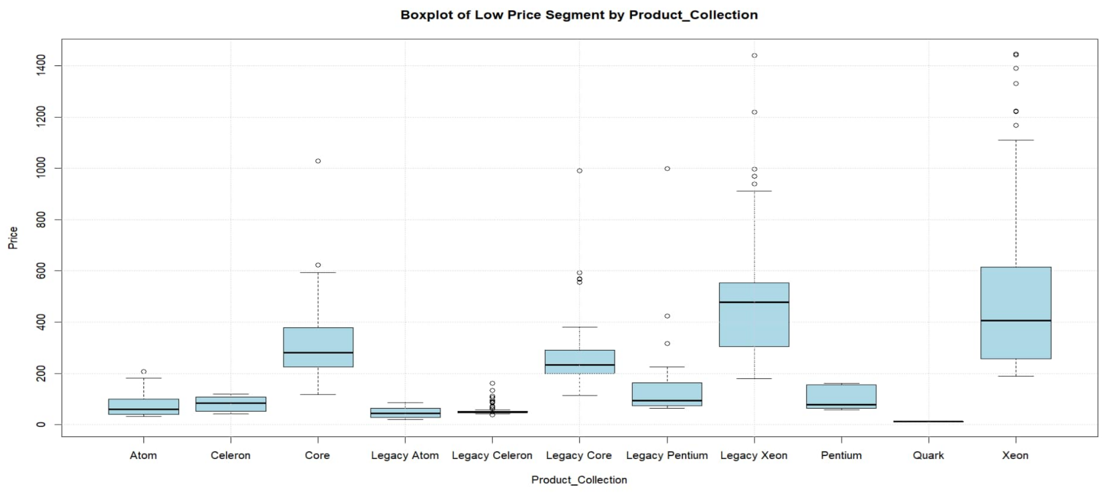


Figure 18: Boxplot of Low Price Segment by Product Collection

The three boxplots shown in Figure 17 and 18 present pricing trends across vertical segments categorized into medium, high, and low price ranges. In the medium segment, the "Embedded" vertical demonstrates greater price variability and a higher median compared to "Server," suggesting a wider range of products. The high price segment focuses exclusively on "Server," showing extensive price variation with a slight skew towards higher prices, indicated by consistent spread without outliers. In the low-price segment, the "Server" vertical has a higher median and wider variability, overlapping with higher segments, while "Desktop" maintains lower, consistent pricing. "Embedded" and "Mobile" exhibit moderate pricing with some variability, indicating a mixed approach to pricing strategies. Overall, the "Server" segment appears more prominent in higher price categories, while "Desktop" shows stability in the lower segment. These patterns illustrate distinct pricing strategies and variations across different product lines and verticals.

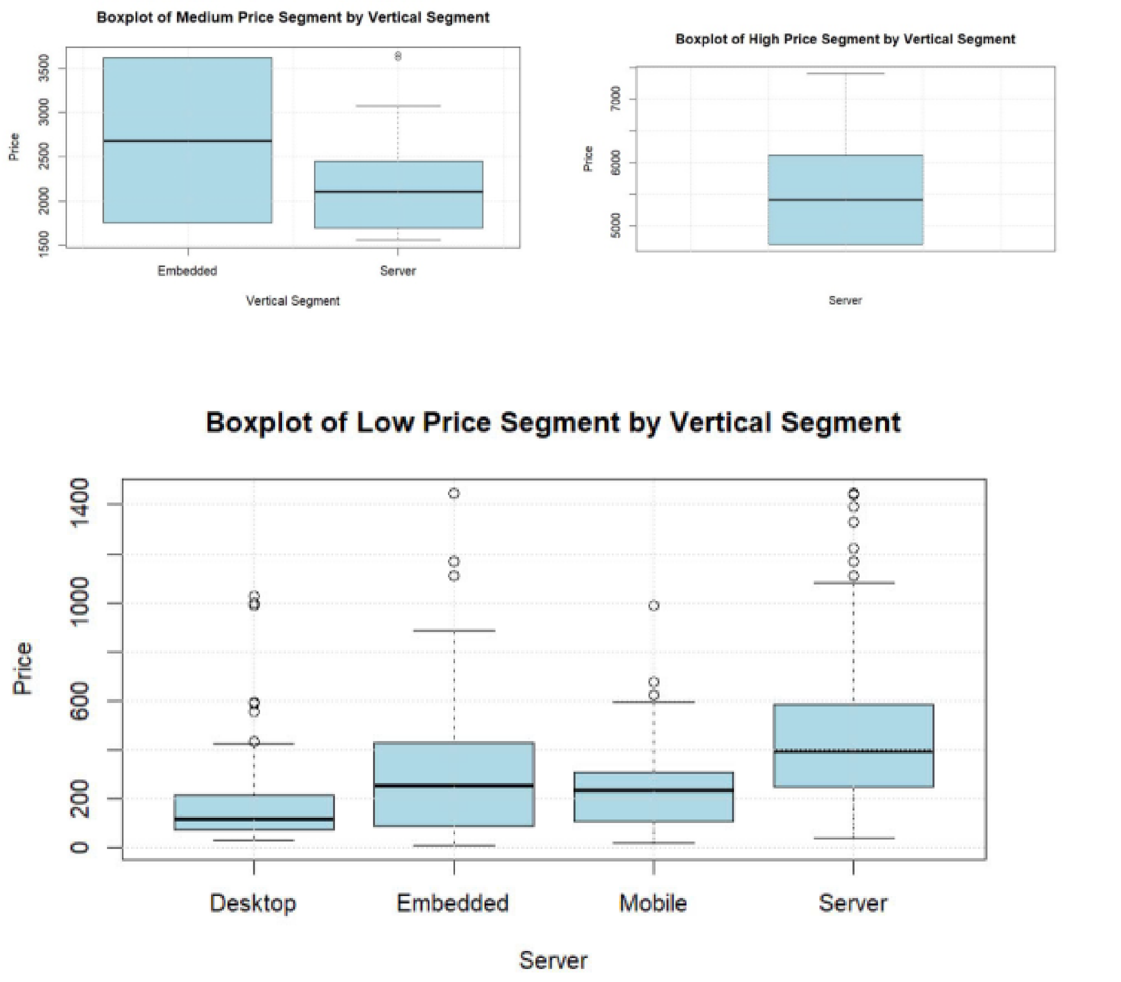


Figure 19: Boxplots of each Price Division according to Vertical Segment

The boxplots in Figure 19 compare price segments across various vertical segments, revealing distinct patterns in each category. The first plot shows that the "Embedded" vertical has a wider price range in the medium segment compared to "Server," which has a more compact distribution with a few outliers. The second plot highlights the "Server" vertical in the high price segment, showing a moderate spread with no evident outliers, indicating consistent pricing. The third plot, which examines the low price segment, depicts more variability among verticals. The "Desktop" vertical has the smallest interquartile range, while "Server" exhibits more spread and outliers, indicating greater price fluctuations. "Embedded" and "Mobile" fall in between, with "Embedded" showing slightly higher values than "Mobile." Overall, the data visualizations illustrate varied pricing strategies across segments, with each having its own distinct distribution characteristics and outliers.

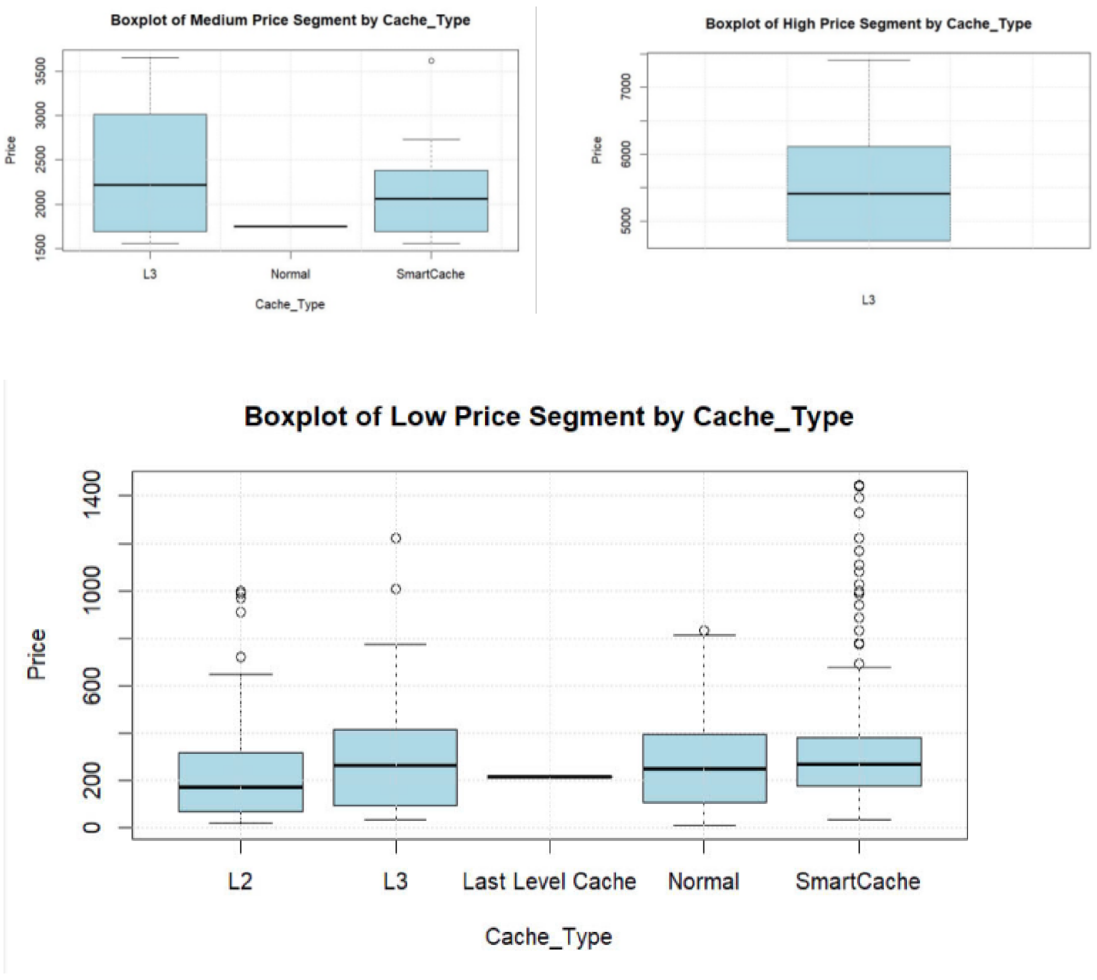


Figure 20: Boxplots of each Price Division according to Cache Type

What we interpret from Figure 20 is that, in the low price segment, SmartCache exhibits the most variability and several outliers, while L2 and L3 have relatively lower medians with moderate variation. Normal cache type prices also display moderate variability. The high price segment is dominated by L3 cache, showcasing a broader range but no outliers, indicating a consistent pricing structure. For the medium price segment, L3 again demonstrates higher price variability with a higher median compared to SmartCache, which shows less variation. Notably, Normal cache does not feature in the medium price range, suggesting a position either below or above this segment. Overall, L3 caches appear prominent in the high and medium price ranges, while SmartCache and Normal are more frequent in the lower price bands.

4.3.2 Scatter graph

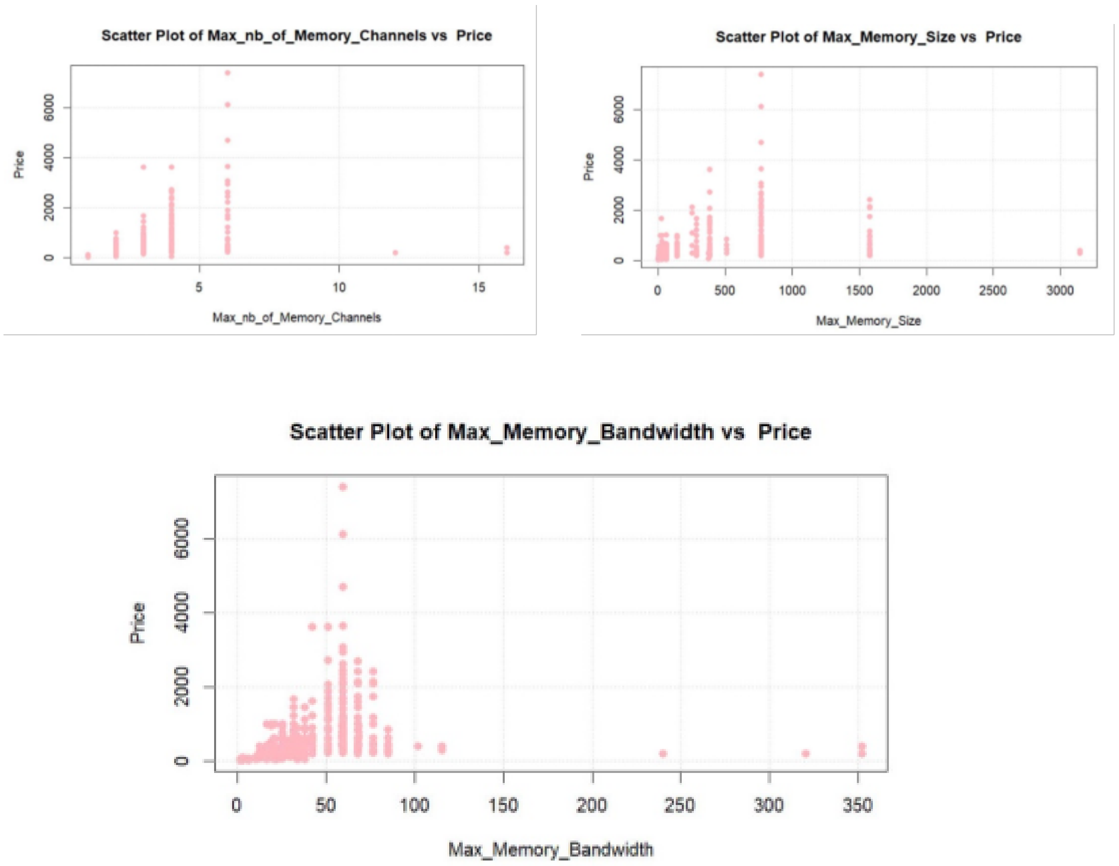


Figure 21: Scatterplot of each Price Division according to all Memory Aspects

From Figure 21, in the *Max_nb_of_Memory_Channels vs. Price* plot, prices increase notably with more memory channels, though most products remain between 1 to 5 channels. As we move to the *Max_Memory_Size vs. Price* plot, there's a broad distribution along memory size, yet price peaks prominently between 500 and 1500, indicating demand or value in this range. The *Max_Memory_Bandwidth vs. Price* plot shows a concentration of products priced up to 200 with max bandwidth between 0 to 150. Across all plots, higher price points typically correspond with increased memory features, yet some outliers exist that deviate from these trends. This analysis suggests that while higher memory specifications often result in higher prices, there are products that may offer good value or specialized features outside these general trends. We can conclude that manufacturers seem to target various market niches by balancing between advanced memory features and pricing strategies.

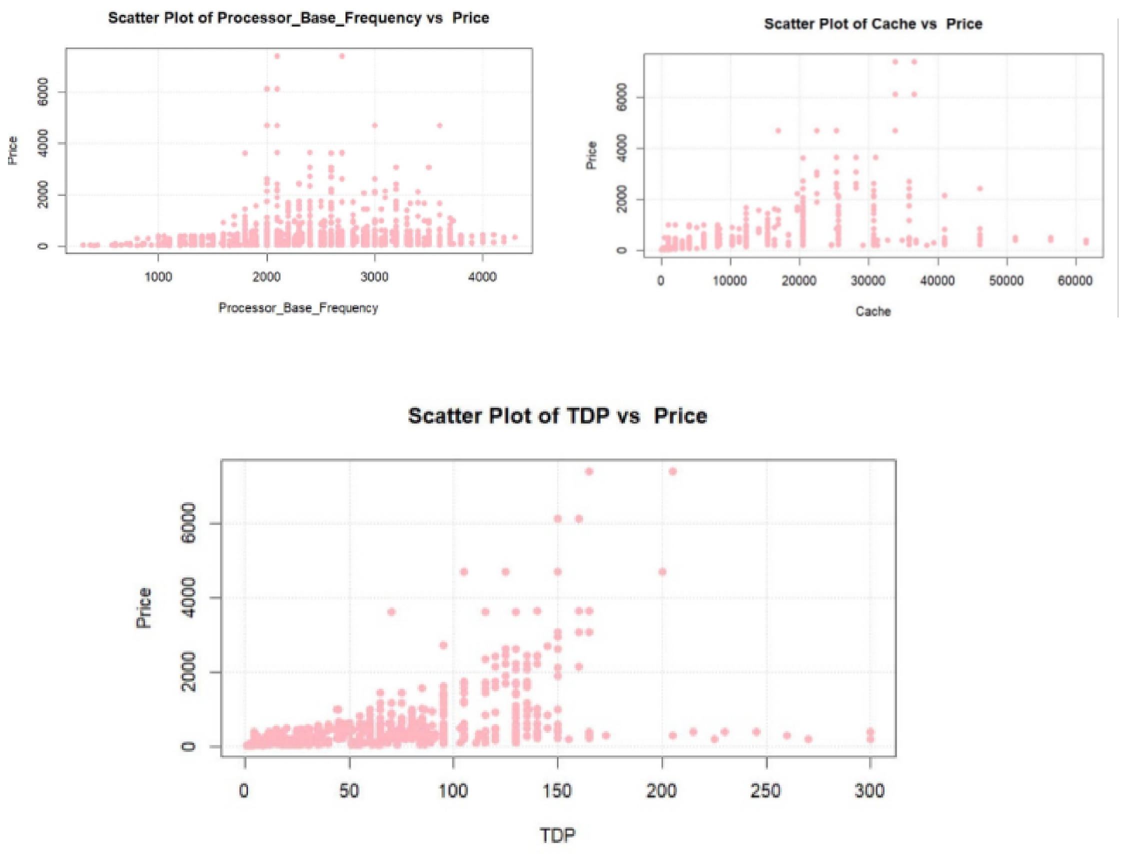


Figure 22: Scatterplot of each Price Division according to Frequency, Cache Size and TDP

In the Processor Base Frequency plot shown in Figure 22, there is a noticeable concentration of prices below 2000, showing limited correlation with frequency, though slight variability increases with higher frequencies. The Cache plot shows a wider spread, especially between 10,000 and 40,000, where prices again fluctuate below 2000 with sporadic outliers extending beyond 6000. The TDP plot clusters heavily below a price of 2000, with a wide range of TDP values, indicating minimal direct correlation yet a trend of increasing prices as TDP rises. Across all variables, the majority of processors are priced lower, with a few high-end outliers. This might suggest a market that predominantly caters to moderate-priced models, with high performance options available. The graphs collectively suggest higher price points are not solely dependent on one characteristic but a combination, likely appealing to a niche market seeking specific high-end features. Overall, the data indicates a more substantial emphasis on value for performance within specific ranges of frequency, cache, and TDP.

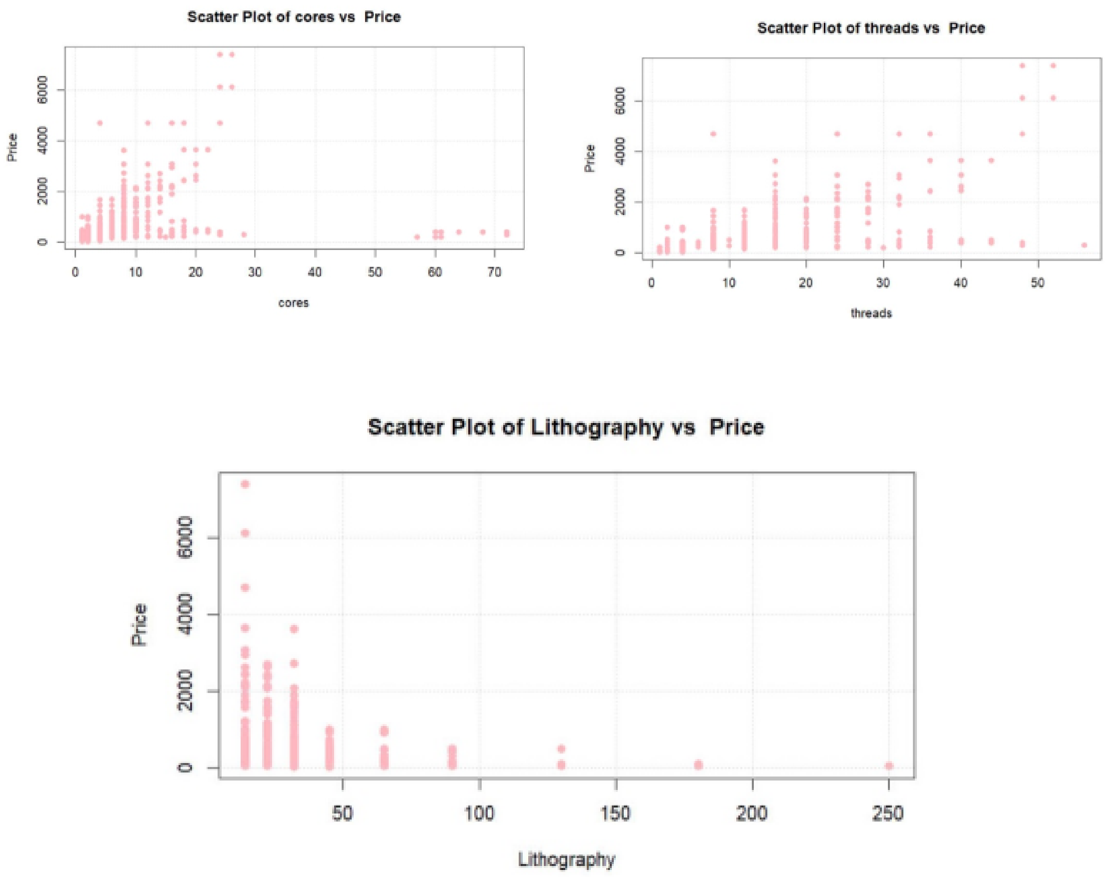


Figure 23: Scatterplot of each Price Division according to Lithography, Core and Thread numbers

The first two scatter plots in Figure 23 highlight the correlation between the number of cores and price, showing a notable increase in price as core count rises, with a cluster of higher price points beyond 20 cores. The plot of threads versus price reveals a similar trend, indicating that higher threading capabilities typically correspond to increased pricing, especially noticeable beyond 30 threads. The lithography versus price plot illustrates an inverse aspect; most processors cluster at lower lithography values around 50 nm, with higher prices found in these lower lithography categories. As lithography increases, prices tend to decrease, indicating advanced manufacturing technology leads to premium pricing. Overall, there is a clear trend that greater cores and threads contribute to higher processor prices. Conversely, advanced lithography seems to support premium pricing trends. These insights suggest that cutting-edge processing capabilities, in terms of core and thread counts, drive price increases, while advancements in lithography technology also play a significant role.

5 Inferential Statistics

After thoroughly exploring the descriptive aspects of the CPU dataset, we now turn our attention to the application of inferential statistics to gain deeper insights and make broader conclusions about the population from which this sample was drawn. The goal of this analysis is to explore any patterns or trends that may provide insights into the broader dynamics of the CPU market. In other words, to analyze and predict the future price based on the various aspects. Therefore, we will use a model to help us achieve our goal, which is Multivariate Linear Regression.

5.1 Using Multivariate Linear Regression to predict the CPU prices based on its attributes

In this section, we will build a multiple linear regression model with with *Recommend Customer Price* as the **dependent variable** while the **independent variables** are *Product Collection*, *Year*, *Vertical Segment*, *Status*, *Lithography*, *nb of Cores*, *nb of Threads*, *Processor Base Frequency*, *Cache*, *Cache Type*, *Instruction Set*, *TDP*, *Max Memory Size*, *Max nb of Memory Channels*, *Max Memory Bandwidth.* The Multivariate Linear Regression is described in Equation 5.

$$\text{Price} = \beta_0 + \beta_{\text{Product Collection}} x_1 + \ldots + \beta_{\text{Memory Bandwidth}} x_n + \epsilon \tag{5}$$

After determining the variables to work with, now we build the model step by step. The first step involves splitting the dataset into 2 subsets of train data and test data. The purpose of this step is to ensure we 'feed' the model with a portion of data and evaluate the performance on the other set. We opt for a proportion of 80 : 20 for train and test respectively to ensure that the model has enough training data to learn. The random seed is set at 42. Now we will build the model accordingly.

5.1.1 Modeling

The model is then built and give the results as shown as below:

Result

```
Call:
lm(formula = Recommended_Customer_Price ~ ., data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-2.42615 -0.23401  0.01936  0.22428  1.82452

Coefficients:
                                   Estimate Std. Error t value Pr(>|t|)
(Intercept)                      -1.083e+02  3.570e+01  -3.034 0.002471 **
Product_CollectionCeleron        -7.492e-02  1.189e-01  -0.630 0.528623
Product_CollectionCore            6.577e-01  1.208e-01   5.445 6.46e-08 ***
Product_CollectionLegacy Atom     3.556e-01  2.118e-01   1.679 0.093528 .
Product_CollectionLegacy Celeron -1.726e-01  1.399e-01  -1.234 0.217634
Product_CollectionLegacy Core     5.428e-01  1.222e-01   4.443 9.84e-06 ***
Product_CollectionLegacy Pentium -1.278e-01  1.369e-01  -0.934 0.350771
Product_CollectionLegacy Xeon     7.419e-01  1.476e-01   5.027 5.86e-07 ***
```

```
Product_CollectionPentium         3.409e-02  1.230e-01   0.277 0.781759
Product_CollectionQuark           9.869e-01  3.836e-01   2.572 0.010238 *
Product_CollectionXeon            6.553e-01  1.338e-01   4.898 1.12e-06 ***
Vertical_SegmentEmbedded          4.083e-01  7.230e-02   5.648 2.10e-08 ***
Vertical_SegmentMobile            6.006e-01  5.281e-02  11.373  < 2e-16 ***
Vertical_SegmentServer            3.150e-01  9.032e-02   3.487 0.000509 ***
Year                              5.292e-02  1.768e-02   2.992 0.002834 **
Lithography                       2.731e-03  3.994e-03   0.684 0.494193
StatusEnd of Interactive Support  8.669e-01  1.885e-01   4.598 4.78e-06 ***
StatusEnd of Life                 9.091e-01  1.825e-01   4.980 7.44e-07 ***
StatusLaunched                    9.495e-01  1.771e-01   5.362 1.02e-07 ***
nb_of_Cores                      -1.361e-02  4.216e-03  -3.229 0.001280 **
nb_of_Threads                     1.887e-02  3.469e-03   5.440 6.66e-08 ***
Processor_Base_Frequency          1.437e-04  3.766e-05   3.816 0.000144 ***
Cache                             5.318e-01  5.570e-02   9.547  < 2e-16 ***
Cache_TypeL3                     -1.620e-01  9.744e-02  -1.663 0.096686 .
Cache_TypeLast Level Cache       -1.096e+00  1.720e-01  -6.371 2.83e-10 ***
Cache_TypeNormal                 -2.614e-01  7.349e-02  -3.557 0.000392 ***
Cache_TypeSmartCache             -6.540e-02  6.644e-02  -0.984 0.325199
Instruction_Set                   7.597e-03  7.051e-03   1.078 0.281502
TDP                              -5.807e-05  1.035e-03  -0.056 0.955286
Max_Memory_Size                  -7.610e-02  2.234e-02  -3.406 0.000686 ***
Max_nb_of_Memory_Channels         4.097e-01  3.471e-02  11.805  < 2e-16 ***
Max_Memory_Bandwidth             -1.880e-02  1.598e-03 -11.767  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4424 on 1036 degrees of freedom
Multiple R-squared:  0.7993, Adjusted R-squared:  0.7933
F-statistic: 133.1 on 31 and 1036 DF,  p-value: < 2.2e-16
```

Table 6: Multivariate Linear Regression's result (insignificant variables included)

Interpretation: From top to down, the Table 6 gives us the following information.

- **Residuals:** The differences between observed values and targeted values. In the relevant context, the residuals represent the deviations or errors of the model's predictions compared to the ground truth. What we focus on these results is that the model underpredicts 25% of the targeted variable with trivial margin (-0.23401) and overpredicts another 25% with nearly the same margin of 0.2243. Overall, the model predicts nearly exact with merely an error of 0.01936.
- **Coefficients**: The listed coefficients correspond to the predetermined independent variables above. 4 of which are not defined because they are highly colinear with each other, which cause singularities. We will later discuss the reason and solution for the singularities issue, now let's focus on the result. The results give information about the estimate results, standard error, t-value and p-value. For the intercept, all the predictors are set to 0, and the estimate result is -108.3, the t-value will be -3.034 and p-value ($0.002471 < 0.05$) suggests that we can confirm the intercept is different from zero. The other coefficients can also be interpret in the similar way.

- **Training results:** The residual standard error of 0.4424 shows that the model still needs some improvements to get a smaller error. For the next row, R-squared R^2 is the proportion of the variance in the dependent variable that is explained by the independent variables in the model while adjust R^2 rely on the number of predictors. R^2 of 0.7993 and adjusted R^2 of 0.7933 determines the performance of the model to not be perform quite well and the predictors are mostly meaningful to the model.

There are several insignificant variables symbolized by the number of stars and blanks. Those variables are *Lithography*, *Instruction Set*, *TDP*. We eliminate these variables as they are irrelevant to the main model and removing these predictors will help improve the model performance, making it easier to understand. The remaining variables are those with p-value < 0.05 or i.e. significant to the model.

```
Call:
lm(formula = Recommended_Customer_Price ~ Product_Collection +
    Vertical_Segment + Year + Status + nb_of_Cores + nb_of_Threads +
    Processor_Base_Frequency + Cache + Cache_Type + Max_Memory_Size +
    Max_nb_of_Memory_Channels + Max_Memory_Bandwidth, data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-2.42426 -0.23231  0.01991  0.22792  1.84000

Coefficients:
                                   Estimate Std. Error t value Pr(>|t|)
(Intercept)                      -9.232e+01  2.517e+01  -3.668 0.000256 ***
Product_CollectionCeleron        -7.937e-02  1.173e-01  -0.677 0.498668
Product_CollectionCore            6.550e-01  1.203e-01   5.445 6.46e-08 ***
Product_CollectionLegacy Atom     2.317e-01  1.560e-01   1.485 0.137865
Product_CollectionLegacy Celeron -1.676e-01  1.368e-01  -1.225 0.220983
Product_CollectionLegacy Core     5.398e-01  1.215e-01   4.443 9.84e-06 ***
Product_CollectionLegacy Pentium -1.241e-01  1.349e-01  -0.920 0.357850
Product_CollectionLegacy Xeon     7.591e-01  1.454e-01   5.219 2.17e-07 ***
Product_CollectionPentium         3.031e-02  1.225e-01   0.247 0.804658
Product_CollectionQuark           7.753e-01  2.914e-01   2.661 0.007910 **
Product_CollectionXeon            6.571e-01  1.330e-01   4.939 9.14e-07 ***
Vertical_SegmentEmbedded          4.015e-01  7.146e-02   5.618 2.48e-08 ***
Vertical_SegmentMobile            5.991e-01  5.203e-02  11.514  < 2e-16 ***
Vertical_SegmentServer            3.087e-01  8.915e-02   3.463 0.000556 ***
```

```
Year                              4.525e-02  1.247e-02   3.630 0.000297 ***
StatusEnd of Interactive Support  8.894e-01  1.872e-01   4.752 2.30e-06 ***
StatusEnd of Life                 9.142e-01  1.823e-01   5.014 6.28e-07 ***
StatusLaunched                    9.501e-01  1.769e-01   5.369 9.75e-08 ***
nb_of_Cores                      -1.397e-02  3.494e-03  -4.000 6.80e-05 ***
nb_of_Threads                     1.880e-02  3.465e-03   5.425 7.19e-08 ***
Processor_Base_Frequency          1.412e-04  2.867e-05   4.926 9.76e-07 ***
Cache                             5.311e-01  5.267e-02  10.084  < 2e-16 ***
Cache_TypeL3                     -1.602e-01  9.684e-02  -1.654 0.098375 .
Cache_TypeLast Level Cache       -1.099e+00  1.714e-01  -6.408 2.23e-10 ***
Cache_TypeNormal                 -2.694e-01  7.255e-02  -3.714 0.000215 ***
Cache_TypeSmartCache             -7.527e-02  6.487e-02  -1.160 0.246179
Max_Memory_Size                  -7.378e-02  2.214e-02  -3.332 0.000894 ***
Max_nb_of_Memory_Channels         4.075e-01  3.386e-02  12.034  < 2e-16 ***
Max_Memory_Bandwidth             -1.873e-02  1.593e-03 -11.756  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4421 on 1039 degrees of freedom
Multiple R-squared:  0.799, Adjusted R-squared:  0.7936
F-statistic: 147.5 on 28 and 1039 DF,  p-value: < 2.2e-16
```

Table 7: Multivariate Linear Regression's result (insignificant variables removed)

The result in Table 7 gives a nearly similar adjusted R-squared compared to the former model, plus, nearly identical RSE. These contribute to prove that removing these variables does not make significant changes to the overall train result. Let's evaluate the performance to see whether the latter model performs better than the former or not.

5.1.2 Model Evaluation

To test the newly trained model on the test data, the performance of the train data and test data is evaluated based on metrics, which are **Mean Squared Error**, **Root Mean Squared Error**, and **R-squared**. The results for train data and test data are shown in Table 8

Result

```
Training Data of LM1:
 MSE: 0.1898761
 RMSE: 0.4357478
 R-squared: 0.799289

Test Data of LM1:
 MSE: 0.2065096
 RMSE: 0.4544332
 R-squared: 0.795135
```

Result

```
Training Data of LM2:
 MSE: 0.1901643
 RMSE: 0.4360783
 R-squared: 0.7989844

Test Data of LM2:
 MSE: 0.2056761
 RMSE: 0.4535152
 R-squared: 0.7959619
```

Table 8: Performance Result of LM1 and LM2, with and without irrelevant variables

What we draw from Table 8 is that MSE and RMSE of both models are relatively low on both train data and test data, with only a small increase in test data. This indicates that the model is not overfitting. Moreover, R-squared value of approximate 0.8 suggests that the model can substantially explains the large amount of variance. Therefore, we can draw out that the efficiency of both model is nearly identical, indicating that removed variables *Lithography*, *Instruction Set* and *TDP* do not meaningfully contribute to the predictive performance of the model. Therefore, eliminating these variables will help maintain the simplicity of the model.

5.1.3 Assumptions of Multivariate Linear Regression

After fitting the model, the next crucial step is to ensure that the model is valid and the results can be trusted. The 5 main assumptions of Multivariate Linear Regression are:

- **Linearity:** There exists a linear relationship between each predictor variable and the response variable.
- **No Multicolinearity:** None of the predictor variable are highly correlated with each other.
- **Independence:** The observations are independent.
- **Homoscedasticity:** The residuals have constant variance at every point in linear model.
- **Multivariate Normality:** The residuals of the model are normally distributed.

If one or more of these assumptions are violated, then the results of the multivariate linear regression model may be unreliable. From the previous results, we can see that the Assumption 2 is violated as there are 4 singularities in the coefficients. Let's check for assumption 4 and assumption 5.

Assumption 4 Testing: Homoscedasticity
One obvious way to determine whether the residuals have constant variance at every point in linear model is that we create a plot of standardized residuals versus predicted values.

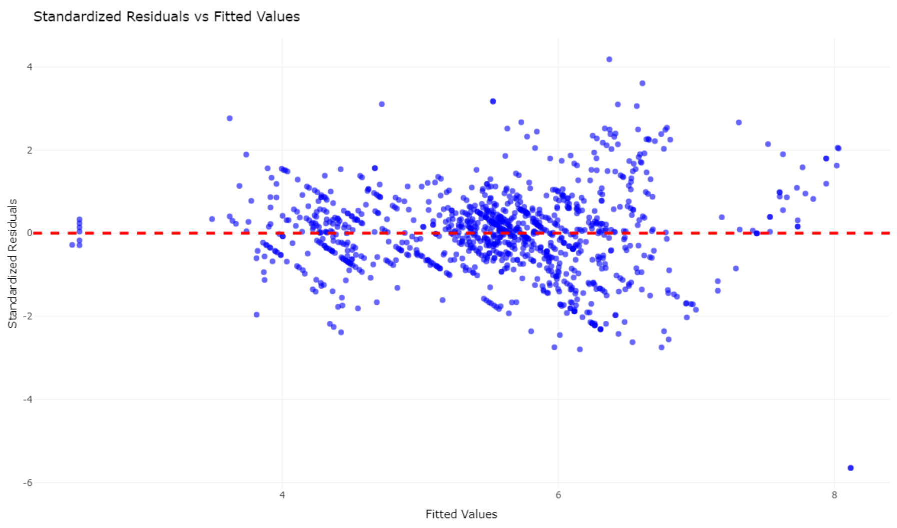


Figure 24: Standardized Residuals vs Fitted Values

What we can tell from Figure is that the residuals are spread across the range of the fitted values. Ideally, the residuals suppose to have a uniform spread (randomly scattered) across all fitted values. However, in this plot, they show a slightly funnel-shaped pattern which spread a minimal larger range as the fitted values increase. This suggests the variance of the residuals is not constant. In conclusion, the assumption 4 is violated.

Assumption 5 Testing: Multivariate Normality
Testing for multivariate normality is important when working with multivariate data, and the most optimal way is to create Q-Q plots to visually inpsect the normality. The Q-Q plot of the residuals are shown below:

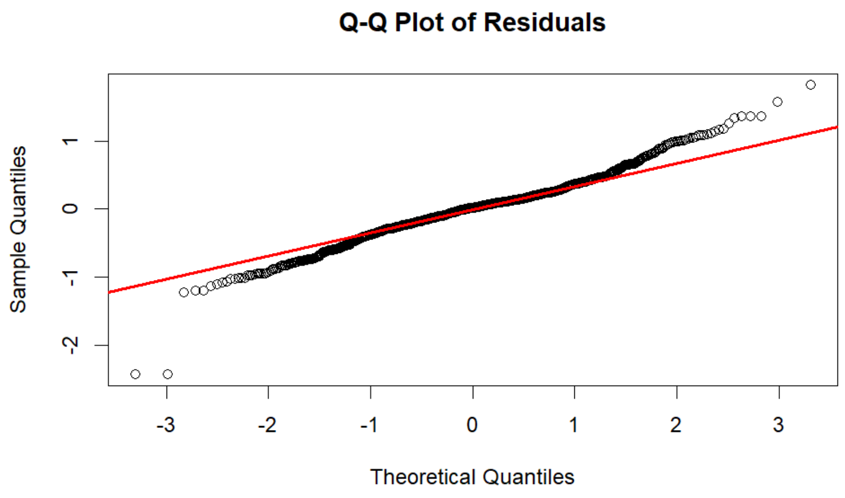


Figure 25: Q-Q Plot of residuals

From the graph shown in Figure 25, the residuals follow a nearly normal distribution for the majority of the data, indicated by their alignment with the red line in the center of the plot. Small deviations at the tails suggest minor violations of normality and the model struggle to fit the outliers. However, this detail is trivial compared to the overall picture. This means that the assumption 5 is valid.

In general, after evaluating the assumptions of the linear regression model, it is evident that two critical assumptions are violated. Therefore, the results of the regression model cannot be fully trusted.

5.2 Model Insights

The Multivariate Linear Regression model assists us evaluating the impact of the CPU factors on the recommended price in market. To consider the effect of each factor, we analyze the regression coefficients, which quantify the relationship between each independent variable and the dependent variable. This allows us to determine how much a specific feature, such as the number of cores, cache size, or memory bandwidth could influence the recommended price, while controlling for other variables. Particularly, we can see that:

- In product type, the **Core** category has a significant positive impact on price, contributing to a rise of **0.665** units. This is also correct in real world matter as Core CPUs are a versatile, high-performance product line targeting a broad range of consumers. For example, we always opt for Intel Core CPU to get the best performance. The same goes for vertical segment, **Mobile** seems to dominate the industry (with a contribution of **0.6** units increase in price). The other variables can also be explained in the same way.
- For the continuous variables, let take **Cache** into consideration. The result implies the impact of Cache on the general price as Cache represents the memory storage closer to the CPU. The increase in Cache can boost the CPU performance, which answers the question why it is reflected aby higher recommended prices as larger cache sizes are often associated with premium CPUs.

The pricing of CPUs is heavily influenced by the **market segment** they target and **product line**. Budget products like Celeron and Pentium have negligible impacts on overall price variation, while premium CPUs such as Core or Xeon are strong drivers to the market price. This interpretation can help simulate the real-life analysis which helps manufacturers and marketers set pricing strategies to specific customer needs.

5.3 Price Prediction

In this last section, we focus on the task of predicting CPU prices based on various technical and categorical features. Price prediction is a critical aspect of understanding market trends and consumer behavior. Let consider 2 CPUs (with given values) with the following attributes in Table 9 below:

Table 9: Features and Sample CPU Attributes

Feature	**CPU 1**	**CPU 2**
Product Collection	Core	Xeon
Vertical Segment	Mobile	Server
Year	2013	2022
Lithography	24 nm	14 nm
Number of Cores	5	24
Number of Threads	10	48
Processor Base Frequency	2420 MHz	3000 MHz
Cache	2980 KB	35.75 MB
Cache Type	L3 Cache	L3 Cache
Instruction Set	64-bit	64-bit
Thermal Design Power (TDP)	60 W	205 W
Maximum Memory Size	64 GB	1 TB
Maximum Number of Memory Channels	3	6
Maximum Memory Bandwidth	35 GB/s	60 GB/s

In Table 9, the **CPU 1** is the average CPU with each value of attribute is the mean value, while **CPU 2** is a Xeon Server CPU with high specifications. The CPU 2 is built based on the **Intel Xeon Gold 6248R Processor** from [8]. These two samples will be inserted into the model 2 to predict the future prices. The results are shown in Table 10, in which the price for CPU 2 is approximately within the range of the exact price ($3178.00) (the price is drawn from the official Intel site).

Result

```
CPU 1 predicted price:  382.8002
CPU 2 predicted price:  3979.8
```

Table 10: CPU 1 and 2 Predicted Price

6 Conclusion and Source Code

6.1 Conclusion

Through our analysis, we've explored the dataset, theoretical frameworks, and implemented various predictive models to estimate CPU prices based on specific characteristics. Our careful feature selection process focused on attributes that demonstrate significant correlation and impact on pricing, providing valuable insights for both manufacturers and consumers in understanding price determinants.
Leveraging R programming, we've developed comprehensive skills in data collection, processing, and long-term analysis, enabling us to make informed predictions about market trends. While implementation may have minor limitations and predictions aren't perfect, our comparative analysis of Multiple Linear Regression reveal that the analysis taken in this study will point out several challenges associated with this model and its assumptions, as well as potential solutions to address these issues. This model shows promising practical applications for real-world CPU pricing scenarios.

6.2 Source Code

The code is written in R_Markdown and R and displayed as notebook for better visualization and neat display.
Source : Source Code Link

References

[1] Dionysia Lemonaki. *What is CPU? Meaning, Definition, and What CPU Stands For*. URL: %5Curl%7Bhttps://www.freecodecamp.org/news/what-is-cpu-meaning-definition-and-what-cpu-stands-for%7D.

[2] Sebastian Taylor. *Multiple Linear Regression, Overview, Formula, How It Works*. URL: %5Curl%7Bhttps://corporatefinanceinstitute.com/resources/data-science/multiple-linear-regression%7D.

[3] The R Graph Gallery. *Correlogram*. URL: %5Curl%7Bhttps://r-graph-gallery.com/correlogram.html%7D.

[4] Geeksforgeeks. *Quantile-Quantile Plot*. URL: %5Curl%7Bhttps://www.geeksforgeeks.org/quantile-quantile-plots%7D.

[5] James Chen. *How a Histogram Works to Display Data*. URL: %5Curl%7Bhttps://www.investopedia.com/terms/h/histogram.asp%7D.

[6] Geeksforgeeks. *Scatter plot*. URL: %5Curl%7Bhttps://www.geeksforgeeks.org/scatter-plot/%7D.

[7] Douglas C. Montgomery and George C. Runger. *Applied Statistics and Probability for Engineers*. 6th. Wiley, 2013.

[8] Intel Corporation. *Intel Xeon Gold 6248R Processor*. URL: %5Curl%7Bhttps://www.intel.com/content/www/us/en/products/sku/199351/intel-xeon-gold-6248r-processor-35-75m-cache-3-00-ghz/specifications.html%7D.