ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

XÁC SUẤT THỐNG KÊ

Khảo sát dung lượng bộ nhớ CPU thông qua thông số kỹ thuật và hiệu suất của máy tính theo thời gian

GVHD: Phan Thị Hường
SV thực hiện: Nguyễn Minh Quân – 2212804
Nguyễn Hữu Nhân – 2212362
Nguyễn Trung Tân – 2213063
Nguyễn Bảo Trâm – 2213572
Nguyễn Quỳnh Như – 2212472
Lớp: L13 - Nhóm: 23

Tp. Hồ Chí Minh, Tháng 10/2023

PHÂN CÔNG VIỆC LÀM

Họ và tên	MSSV	Nhiệm vụ	Tỉ lệ đóng góp	Điểm chia
Nguyễn Minh Quân	2212804	Kiểm tra và tổng hợp nội dung	20%	0
Nguyễn Hữu Nhân	2212362	Xử lý dữ liệu	20%	0
Nguyễn Trung Tân	2213063	Soạn cơ sở lý thuyết và thống kê suy diễn	20%	0
Nguyễn Bảo Trâm	2213572	Xử lý dữ liệu	20%	0
Nguyễn Quỳnh Như	2212472	Thực hiện phần thống kê tả và mở rộng	20%	0

Mục lục

1 Tổng quan dữ liệu

1.1 Mô tả tệp dữ liệu

- **Tên tập dữ liệu:** Intel_CPUs
- **Nguồn:** https://www.kaggle.com/datasets/iliassekkaf/computerparts/code?datasetId=2442
- **Số lượng quan trắc:** Intel_CPUs: 2283
- **Số lượng biến:** Intel_CPUS: 45

1.2 Mô tả biến

Danh sách các biến chính cần quan tâm :

Biến	Kiểu dữ liệu	Đơn vị	Mô tả
Product_Collection	Phân loại định tính	không có	Bộ sưu tập sản phẩm
Launch_Date	$\{x \in \mathbb{R} \mid 0.25 \leq x \leq 18.50\}$,liên tục	không có	Ngày ra mắt
Recommended_Customer_Price	$\{x \in \mathbb{R} \mid 2.54 \leq x \leq 13011\}$, liên tục	đô la	Giá bán được đề nghị cho khách hàng
nb_of_Cores	$\{x \in \mathbb{N} \mid 1 \leq x \leq 72\}$, liên tục	nhân	Thuật ngữ phần cứng mô tả số lượng đơn vị xử lý trung tâm độc lập
nb_of_Threads	$\{x \in \mathbb{N} \mid 1 \leq x \leq 56\}$, liên tục	luồng	Luồng (hay luồng thực thi) là một thuật ngữ phần mềm cho chuỗi lệnh cơ bản được sắp xếp có thể được truyền vào
Processor_Base_Frequency	$\{x \in \mathbb{R} \mid 3.2 \times 10^7 \leq x \leq 4.3 \times 10^9\}$, liên tục	Hz	Mô tả tốc độ đóng mở của bóng bán dẫn trong bộ xử lý.
Cache	$\{x \in \mathbb{N} \mid 8000 \leq x \leq 60000000\}$, liên tục	Byte	CPU Cache là vùng bộ nhớ nhanh nằm trên bộ xử lý.
Bus_Speed	$\{x \in \mathbb{N} \mid 0 \leq x \leq 9.6 \times 10^9\}$, liên tục	Hz/s	Bus speed là tốc độ dữ liệu được xử lý trong một giây
Max_Memory_Size	$\{x \in \mathbb{R} \mid 1.00 \times 10^9 \leq x \leq 4.10 \times 10^{12}\}$, liên tục	Byte	Max Memory Size, dùng để biểu thị dung lượng bộ nhớ tối đa, tính bằng byte, mà bộ xử lý có thể hỗ trợ được
Max_Memory_Bandwidth	$\{x \in \mathbb{R} \mid 1.6 \leq x \leq 352\}$, liên tục	GB/s	Max Memory Bandwidth (Băng thông bộ nhớ) là thuật ngữ chỉ khả năng truyền tải dữ liệu tối đa của bộ nhớ

2 Kiến thức nền

2.1 Hồi quy tuyến tính

2.1.1 Khái niệm

Phương pháp hồi quy tuyến tính là một công cụ phân tích thống kê được sử dụng để nghiên cứu mối quan hệ giữa biến độc lập và biến phụ thuộc. Nó giúp chúng ta hiểu rõ cách biến độc lập ảnh hưởng đến biến phụ thuộc và dự đoán giá trị của biến phụ thuộc dựa trên biến độc lập. Hồi quy tuyến tính giả định rằng mối quan hệ giữa các biến là tuyến tính, tức là biến phụ thuộc biến đổi một cách tuyến tính dựa trên biến độc lập.

2.1.2 Mô hình hồi quy tuyến tính đơn

Một mô hình thống kê tuyến tính đơn (Simple linear regression model) liên quan đến một biến ngẫu nhiên Y và một biến giải thích x là phương trình (1) có dạng

$$Y = f(x) + \epsilon = \beta_0 + \beta_1 x + \epsilon$$

Trong đó:

- β_0, β_1 là các tham số chưa biết, gọi là các hệ số hồi quy,
- x là biến độc lập, giải thích cho y,
- ϵ là thành phần sai số, ϵ được giả sử có phân phối chuẩn với $\mathbb{E}(\epsilon) = 0$ và $\mathbb{V}ar(\epsilon) = \sigma^2$.

Trong mô hình (1), sự thay đổi của Y được giả sử ảnh hưởng bởi 2 yếu tố:

- Mối liên hệ tuyến tính của X và Y :

$$\mathbb{E}[Y|x] = \beta_0 + \beta_1 x,$$

  Trong đó, β_0 được gọi là hệ số chặn (intercept) và β_1 gọi là hệ số góc (slope).
- Tác động của các yếu tố khác (không phải X): thành phần sai số ϵ.

Với $(x_1, y_1), ..., (x_n, y_n)$ là n cặp giá trị quan trắc của một mẫu ngẫu nhiên cỡ n, từ (1) ta có

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \qquad i = 1, 2, ..., n$$

Một mô hình hồi quy tuyến tính đơn cần các giả định:

- Các thành phần sai số ϵ_i là độc lập với nhau.
- $\epsilon_i \sim \mathbb{N}(0, \sigma^2)$ hoặc $Y \sim \mathbb{N}(\beta_0 + \beta_1 x, \sigma^2)$

2.2 Hồi quy tuyến tính bội

2.2.1 Khái niệm

Hồi quy tuyến tính bội là một phương pháp trong thống kê để xác định mối quan hệ giữa một biến phụ thuộc và hai hoặc nhiều biến độc lập.

Hồi quy tuyến tính bội: Sử dụng hai hoặc nhiều biến độc lập để dự đoán biến phụ thuộc, Mô hình mối quan hệ tuyến tính giữa nhiều biến độc lập và biến phụ thuộc. Mô hình này có thể được biểu diễn dưới dạng siêu phẳng (plane) hoặc siêu không gian (hyperplane) trong không gian nhiều chiều.

Phương trình Mô hình: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + e$, trong đó biến y là biến phụ thuộc, $x_1, x_2, x_3, ...$ là các biến độc lập, b_0 là hệ số chặn và $\beta_1, \beta_2, .., \beta_n$ là các hệ số góc tương ứng.

β_0: đây là hằng số hồi quy (hệ số chẵn). Hằng số này là đại diện của giá trị Y (biến phụ thuộc) khi tất cả các biến độc lập đều cùng bằng 0. Nói một cách dễ hiểu hơn β_0 là kết quả của Y khi các X bằng 0. Thông thường, trên đồ thị, β_0 chính là điểm nằm trên trục Oy. Tại vị trí mà đường hồi quy giao với trục Oy.

$\beta_1, \beta_2, \beta_n$: đây là hệ số hồi quy (hệ số góc). Cụ thể hơn, hệ số hồi quy là sự phản ánh mức thay đổi của giá trị Y khi X thay đổi. Và mỗi một sự thay đổi của biến X sẽ tương ứng với một giá trị Y khác nhau trên đồ thị. Tóm lại, $\beta_1, \beta_2, \beta_n$ là kết quả của Y khi các biến X tương ứng thay đổi. Và hệ số này có thể tăng, có thể giảm tùy thuộc vào sự biến dổi của X.

ϵ: là sai số. Đây là giá trị biểu hiện cho sự sai lệch trong mối tương quan giữa hai biến là biến độc lập và biến phụ thuộc. Khi giá trị ϵ càng lớn thì khả năng kiểm định của phương pháp hồi quy tuyến tính sẽ trở nên kém đi. Sẽ dẫn tới những sai lệch nhiều trên thực tiễn. Các sai số này sẽ là đại diện của các biến độc lập ngoài mô hình kiểm định hoặc là bất kỳ sai số ngẫu nhiên nào đó trong bài toán nghiên cứu.

Hồi quy tuyến tính bội cung cấp khả năng mô hình hóa mối quan hệ phức tạp hơn giữa biến độc lập và biến phụ thuộc bằng cách sử dụng nhiều biến.

2.2.2 Sự khác biệt giữa hồi quy tuyến tính bội và đơn

Hồi quy tuyến tính bội và đơn là hai phương pháp khác nhau trong mô hình hồi quy, một phần của thống kê và máy học được sử dụng để dự đoán giá trị của một biến phụ thuộc dựa trên một hoặc nhiều biến độc lập.

2.3 Phân tích tương quan

2.3.1 Khái niệm

Phân tích tương quan (Correlation Analysis) dùng để đo độ mạnh của mối liên hệ tuyến tính giữa hai biến ngẫu nhiên. Loại hệ số tương quan được sử dụng rộng rãi nhất là Pearson r. Phân tích này giả định rằng hai biến đang được phân tích được đo lường trên ít nhất các thang đo khoảng cách , có nghĩa là chúng được đo lường trên một phạm vi giá trị tăng dần. Hệ số được tính bằng cách lấy hiệp phương sai của hai biến và chia nó cho tích của độ lệch chuẩn của chúng .

Hệ số tương quan có thể nằm trong khoảng từ -1.00 đến $+1.00$ trong đó giá trị -1.00 đại diện cho mối tương quan âm hoàn hảo, có nghĩa là khi giá trị của một biến tăng lên, biến kia giảm trong khi giá trị $+1.00$ đại diện cho mối quan hệ dương hoàn hảo, có nghĩa là khi một biến tăng giá trị, biến kia cũng vậy. Các giá trị như thế này báo hiệu mối quan hệ tuyến tính hoàn hảo giữa hai biến, vì vậy nếu bạn vẽ kết quả trên biểu đồ, nó sẽ tạo thành một đường thẳng, nhưng giá trị 0,00 có nghĩa là không có mối quan hệ nào giữa các biến đang được kiểm tra và sẽ được vẽ biểu đồ như các dòng riêng biệt hoàn toàn.

Ví dụ, nếu chúng ta muốn biết liệu có mối liên hệ giữa nhiệt độ và sự sinh trưởng chiều cao của thực vật hay không, một hệ số tương quan có thể được tính toán để trả lời câu hỏi này.

2.3.2 Các phương pháp phân tích tương quan

Pearson correlation (r) đo lường sự phụ thuộc tuyến tính giữa hai biến (x và y). Còn được gọi là kiểm tra tương quan tham số vì nó phụ thuộc vào kiểu phân phối dữ liệu. Chỉ được sử dụng khi dữ liệu tuân theo luật phân phối chuẩn (normal distribution). Đồ thị của $y = f(x)$ gọi là đường cong hồi quy tuyến tính.

Kendall's Tau và **Spearman Rho** là các phương pháp tính hệ số tương quan dựa trên thứ hạng (phi tham số)

2.3.3 Ứng dụng của phân tích tương quan

Giúp chúng ta thấy các xu hướng hoặc mô hình khác nhau trong xã hội có thể được kết nối như thế nào, chẳng hạn như thất nghiệp và tội phạm; và họ có thể làm sáng tỏ cách thức mà những trải nghiệm và đặc điểm xã hội định hình những gì xảy ra trong cuộc sống của một người. Phân tích tương quan cho phép chúng ta nói một cách tự tin rằng mối quan hệ có hoặc không tồn tại giữa hai mẫu hoặc biến khác nhau, điều này cho phép chúng ta dự đoán xác suất của một kết quả trong tổng thể được nghiên cứu.

Một nghiên cứu gần đây về hôn nhân và giáo dục cho thấy mối tương quan nghịch giữa trình độ học vấn và tỷ lệ ly hôn. Dữ liệu từ Điều tra Quốc gia về Tăng trưởng Gia đình cho thấy khi trình độ học vấn của phụ nữ tăng lên, tỷ lệ ly hôn đối với các cuộc hôn nhân đầu tiên sẽ giảm xuống.

3 Tiền xử lý số liệu

3.1 Đọc dữ liệu

Dùng lệnh read.csv để đọc dữ liệu vào biến cpu_df và in ra trên terminal để kiểm tra xem data có được nhập thành công hay không:

```
> cpu_df <- read.csv("Intel_CPUs.csv")
> head(cpu_df,10)
                        Product_Collection Vertical_Segment Processor_Number                  Status Launch_Date Lithography
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
7         Intel® Celeron® Processor N Series          Mobile            N2805               Launched       Q3'13       22 nm
8         Intel® Celeron® Processor J Series         Desktop            J1750               Launched       Q3'13       22 nm
9         Intel® Celeron® Processor G Series         Desktop            G1610               Launched       Q1'13       22 nm
10          Legacy Intel® Pentium® Processor          Mobile              518 End of Interactive Support                   90 nm
```

Hình 1: *Đọc dữ liệu của file Intel_CPUs.csv vào biến cpu_df và in ra 10 dòng đầu tiên của mỗi cột*

3.2 Làm sạch dữ liệu

3.2.1 Trích ra dữ liệu con

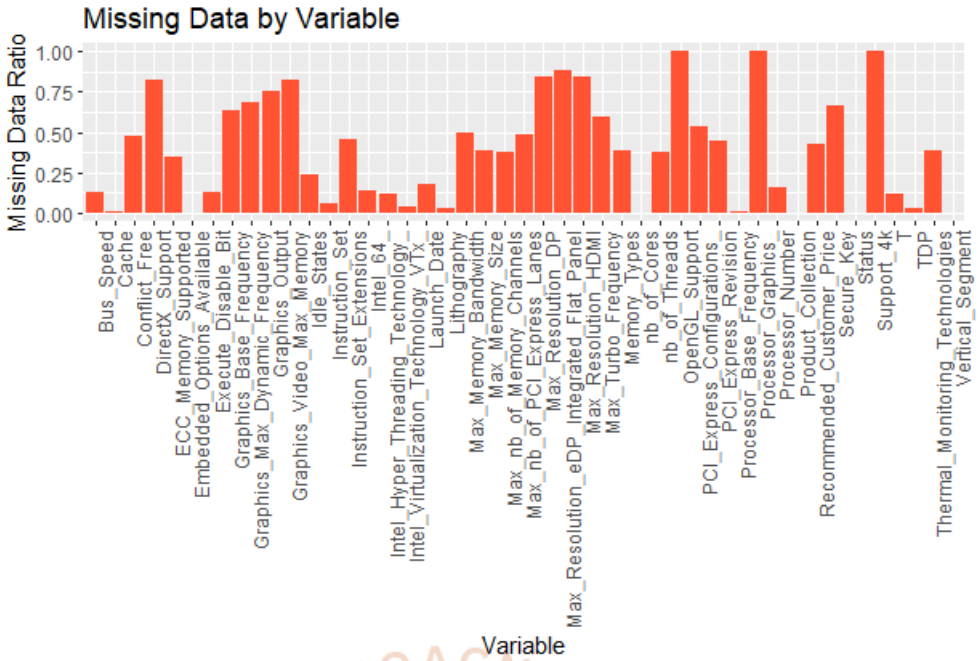
Ta sẽ trích ra dữ liệu con bao gồm các biến chính mà ta quan tâm và lưu vào biến new_cpu_df. Hiện thực bằng R:

```
> new_cpu_df <- cpu_df[,c(1,3,5,7,8,9,10,12,13,14)]
> head(new_cpu_df, 10)
                        Product_Collection Processor_Number Launch_Date
1  7th Generation Intel® Core™ i7 Processors          i7-7Y75       Q3'16
2  8th Generation Intel® Core™ i5 Processors         i5-8250U       Q3'17
3  8th Generation Intel® Core™ i7 Processors         i7-8550U       Q3'17
4           Intel® Core™ X-series Processors          i7-3820       Q1'12
5  7th Generation Intel® Core™ i5 Processors          i5-7Y57       Q1'17
6      Intel® Celeron® Processor 3000 Series            3205U       Q1'15
7         Intel® Celeron® Processor N Series            N2805       Q3'13
8         Intel® Celeron® Processor J Series            J1750       Q3'13
9         Intel® Celeron® Processor G Series            G1610       Q1'13
10          Legacy Intel® Pentium® Processor              518
   Recommended_Customer_Price nb_of_Cores nb_of_Threads Processor_Base_Frequency
1                     $393.00           2             4                1.30 GHz
2                     $297.00           4             8                1.60 GHz
3                     $409.00           4             8                1.80 GHz
4                     $305.00           4             8                3.60 GHz
```

Hình 2: *Lưu dữ liệu con gồm các biến chính từ biến cpu_df vào biến new_cpu_df*

Để giải thích cho việc sử dụng các biến đó, ta có biểu đồ cho tỉ lệ khuyết của từng biến trong dataset ban đầu như sau:

Từ biểu đồ, ta có thể nhìn thấy và chọn ra các biến có tỉ lệ khuyết thấp như Bus_Speed, Cache, nb_of_Cores, nb_of_Threads, Launch_Date, Max_Memory_Bandwidth, Max_Memory_Size, Proccessor_Base_Frequency, Product_Collection, Max_nb_of_Memory_Channels và Recommended_Customer_Price. Ngoài ra, các biến này còn ảnh hưởng đến Cache được chứng minh qua một số tài liệu sách báo (nhóm tác giả sẽ trích dẫn nguồn ở phần tài liệu tham khảo).



Hình 3: *Biểu đồ thể hiện tỉ lệ khuyết của từng biến có trong dataset*

3.2.2 Kiểm tra các dữ liệu bị khuyết

Ta tìm các giá trị bị khuyết trong new_cpu_df.
Hiện thực bằng R:

```
Product_Collection : 0
Launch_Date : 412
Recommended_Customer_Price : 982
nb_of_Cores : 0
nb_of_Threads : 856
Processor_Base_Frequency : 18
Cache : 12
Bus_Speed : 294
Max_Memory_Size : 880
Max_Memory_Bandwidth : 1136
```

Hình 4: *In ra tổng số giá trị bị khuyết trong new_cpu_df*

Nhận xét: Các dữ liệu khuyết được tính trong đoạn code trên bao gồm các dữ liệu rỗng, dữ liệu chứa "-", dữ liệu null và dữ liệu chứa chuỗi "Unknown".

3.2.3 Thay thế các dữ liệu bị khuyết

Đầu tiên, ta sẽ thay đổi format của biến Launch_Date bằng cách lấy số năm + 1 + số quý (quý 1 = 0.0, quý 2 = 0.25, quý 3 = 0.5 và quý 4 = 0.75)

Tiếp theo, ta sẽ xoá bỏ các hàng mang giá trị rỗng trong biến Product_Collection, sau đó chọn ra từ khoá quan trọng trong tên của từng ô:

```
new_cpu_df <- new_cpu_df[complete.cases(new_cpu_df$Processor_Number), ]

new_cpu_df <- new_cpu_df %>%
  mutate(Product_Collection = gsub('.*Core.*', 'Core', Product_Collection),
         Product_Collection = gsub('.*X-series.*', 'X-series', Product_Collection),
         Product_Collection = gsub('.*Celeron.*', 'Celeron', Product_Collection),
         Product_Collection = gsub('.*Pentium.*', 'Pentium', Product_Collection),
         Product_Collection = gsub('.*Quark.*', 'Quark', Product_Collection),
         Product_Collection = gsub('.*Core. [mM].*', 'm', Product_Collection),
         Product_Collection = gsub('.*Atom.*', 'Atom', Product_Collection),
         Product_Collection = gsub('.*Itanium.*', 'Itanium', Product_Collection),
         Product_Collection = gsub('.*Xeon.*', 'Xeon', Product_Collection))

new_cpu_df <- within(new_cpu_df, {
  Processor_Number <- ifelse(Processor_Number == 'Yes', 1,
                             ifelse(Processor_Number == 'No', 0, Processor_Number))
})

new_cpu_df <- new_cpu_df[new_cpu_df$Processor_Number != "", ,drop =  FALSE]
```

Hình 5: *Xoá các ô trống và thay đổi tên thành các từ khoa trong biến Product_Collection*

Tiếp theo, ta sẽ tính giá trị trung bình của biến Recommended_Customer_Price và thay vào các ô dữ liệu bị khuyết. Lưu ý ở đây sẽ có một vài giá trị ở dạng khoảng (ví dụ: $\$70.00 - \$77.00$) nên ta sẽ thay thể các giá trị có khoảng bằng chính giá trị trung bình của nó và mới tiến hành tính trung bình của biến và thay vào chỗ dữ liệu khuyết:

	Product_Collection	Launch_Date	Recommended_Customer_Price
105	Pentium	12.25	$70.00 - $77.00

Hình 6: *Ví dụ về giá trị dạng khoảng ở trong biến Recommended_Customer_Price*

	Product_Collection	Launch_Date	Recommended_Customer_Price
105	Pentium	12.25	73.5

Hình 7: *Giá trị dạng khoảng ở trong biến Recommended_Customer_Price đã được xử lý*

Tiếp theo, ta sẽ thay đổi đơn vị của biến Processor_Base_Frequency thành Hz và sau đó thay thế các dữ liệu bị khuyết thành giá trị trung bình của biến:

Processor_Base_Frequency
2.90 GHz
3.10 GHz
32 MHz
400 MHz
33 MHz
600 MHz
1.60 GHz

Hình 8: *Biến Processor_Base_Frequency trước khi xử lý*

Processor_Base_Frequency
2900000000
3100000000
32000000
400000000
2222320530
33000000
600000000
2222320530
1600000000

Hình 9: *Biến Processor_Base_Frequency sau khi xử lý*

Nhận xét: bởi vì giá trị của biến Processor_Base_Frequency phải luôn dương nên ta sẽ không thay thế số 0 vào các ô bị khuyết mà sẽ thay bằng giá trị trung bình của các ô còn lại.

Với các biến Cache, Bus_Speed, Max_Memory_Size, Max_Memory_Bandwidth, ta tiến hành đổi đơn vị của các biến đó (Cache quy về đơn vị Byte, Bus_Speed quy về đơn vị Hz/s, Max_Memory_Size quy về đơn vị Byte và Max_Memory_Bandwidth quy về đơn vị GB/s):

Cuối cùng, với các biến ta chưa xử lý dữ liệu khuyết, ta sẽ tính toán giá trị trung bình của từng biến đó và thay vào ô dữ liệu bị khuyết thông qua hàm sau :

```
new_cpu_df$Launch_Date <- as.numeric(new_cpu_df$Launch_Date)
for (col in 1: ncol(new_cpu_df)) {
  if (class(new_cpu_df[[col]]) != "character") {
    new_cpu_df[[col]] <- ifelse(is.na(new_cpu_df[[col]]),
                                mean(new_cpu_df[[col]], na.rm = TRUE), new_cpu_df[[col]])
  }

}
```

Hình 10: *đổi đơn vị của các biến*

Sau khi thay thế hết dữ liệu bị khuyết, ta tiến hành kiểm tra xem còn dữ liệu nào bị khuyết không :

```
Product_Collection : 0
Launch_Date : 0
Recommended_Customer_Price : 0
nb_of_Cores : 0
nb_of_Threads : 0
Processor_Base_Frequency : 0
Cache : 0
Bus_Speed : 0
Max_Memory_Size : 0
Max_Memory_Bandwidth : 0
```

Hình 11: *In ra tổng số giá trị bị khuyết*

Nhận xét: Như vậy không còn dữ liệu nào trong new_cpu_df bị khuyết, ta chuyển sang bước tiếp theo.

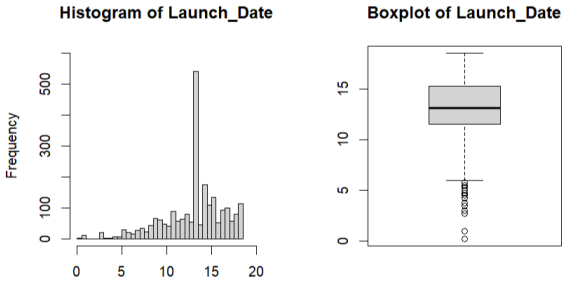
4 Thống kê tả

Sau khi làm sạch dữ liệu, tiến hành thống kê mô tả cho các biến ngẫu nhiên. Kết quả thu được trả về giá trị nhỏ nhất (Min.), điểm phân vị thứ nhất (1st Qu.), trung vị (Median), giá trị trung bình (Mean), điểm phân vị thứ ba (3rd Qu.) và giá trị lớn nhất (Max.) của từng biến (hình 12):

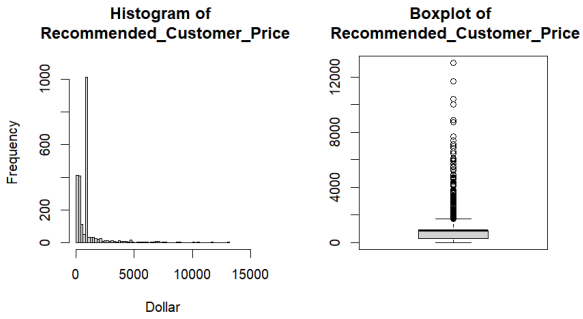
```
Product_Collection  Launch_Date    Recommended_Customer_Price  nb_of_Cores      nb_of_Threads
Atom   :142        Min.   : 0.25   Min.   :    2.54           Min.   : 1.000   Min.   : 1.00
Celeron:290        1st Qu.:11.50   1st Qu.:  278.50           1st Qu.: 1.000   1st Qu.: 4.00
Core   :691        Median :13.12   Median :  850.70           Median : 2.000   Median : 9.00
Itanium: 37        Mean   :13.12   Mean   :  850.70           Mean   : 4.067   Mean   : 8.83
Pentium:401        3rd Qu.:15.25   3rd Qu.:  850.70           3rd Qu.: 4.000   3rd Qu.: 9.00
Quark  : 11        Max.   :18.50   Max.   :13011.00           Max.   :72.000   Max.   :56.00
Xeon   :711
Processor_Base_Frequency     Cache            Bus_Speed         Max_Memory_Size     Max_Memory_Bandwidth
Min.   :3.200e+07        Min.   :    8000   Min.   :0.000e+00   Min.   :1.000e+09   Min.   :  1.60
1st Qu.:1.660e+09        1st Qu.: 2000000   1st Qu.:5.000e+06   1st Qu.:3.200e+10   1st Qu.: 25.60
Median :2.222e+09        Median : 3000000   Median :5.330e+08   Median :2.689e+11   Median : 35.08
Mean   :2.222e+09        Mean   : 7083724   Mean   :1.549e+09   Mean   :2.689e+11   Mean   : 35.08
3rd Qu.:2.800e+09        3rd Qu.: 8000000   3rd Qu.:1.549e+09   3rd Qu.:2.689e+11   3rd Qu.: 35.08
Max.   :4.300e+09        Max.   :60000000   Max.   :9.600e+09   Max.   :4.100e+12   Max.   :352.00
```

Hình 12: *Kết quả thống kê mô tả của từng biến ngẫu nhiên đã chọn trong Intel_CPUs*

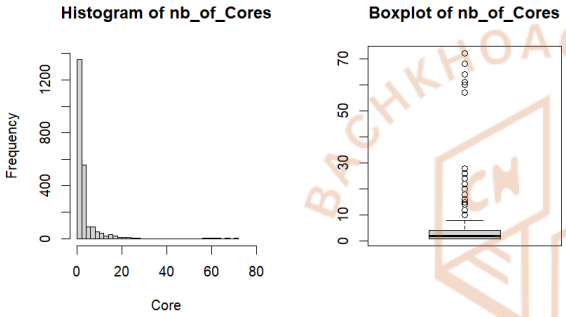
Tiếp theo, nhóm tác giả phân tích dữ liệu của biến bằng cách trực quan hóa dữ liệu qua hai dạng biểu đồ là biểu đồ tần suất (histogram) và biểu đồ hộp (boxplot) cho các biến liên tục. Biểu đồ tần số sẽ cho thấy cái nhìn tổng quan về phân phối của biến. Biểu đồ hộp giúp biểu diễn rõ ràng các đại lượng quan trọng của biến như giá trị lớn nhất, nhỏ nhất, điểm phân vị,...



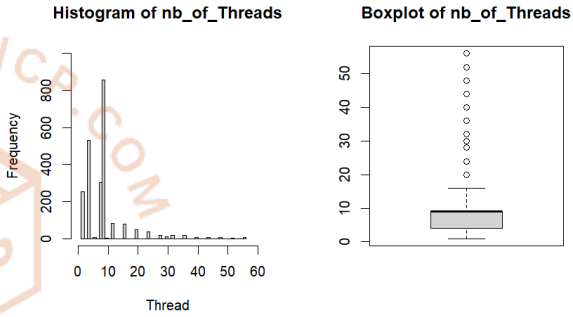
Hình 13: *Launch_ Date*



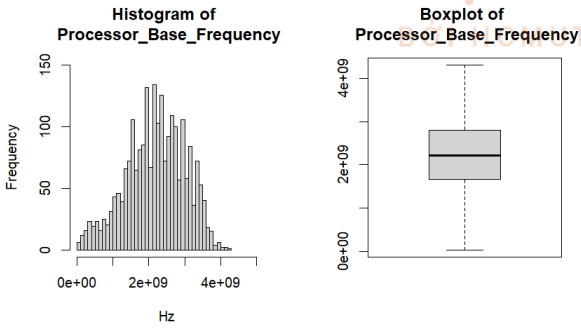
Hình 14: *Recommended_ Customer_ Price*



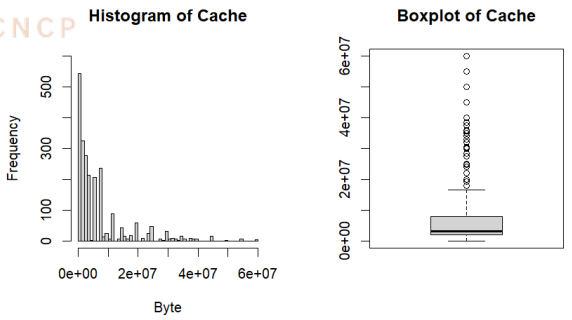
Hình 15: *nb_ of_ Cores*



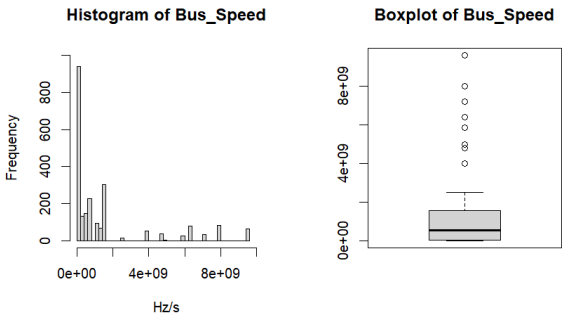
Hình 16: *nb_ of_ Threads*



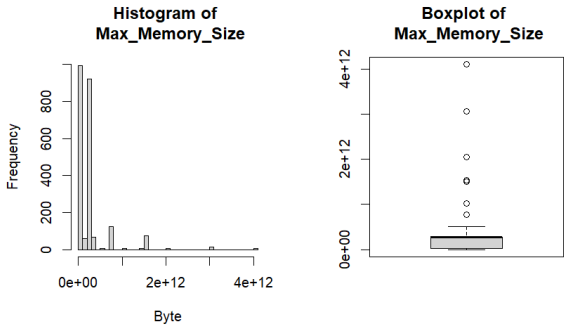
Hình 17: *Processor_ Base_ Frequency*



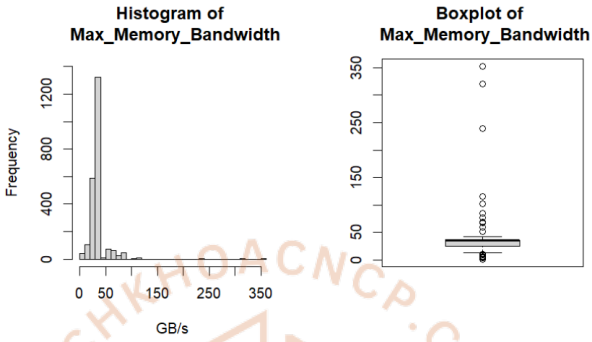
Hình 18: *Cache*



Hình 19: *Bus_Speed*



Hình 20: *Max_Memory_Size*



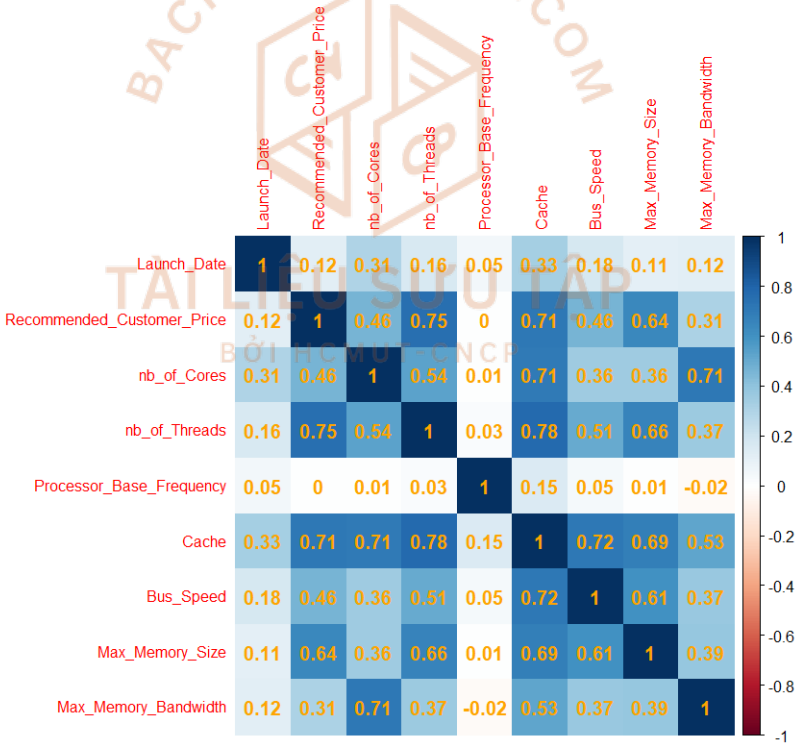
Hình 21: *Max_Memory_Bandwidth*

Qua dữ liệu thu được và hình ảnh từ biểu đồ, nhóm tác giả đưa ra các nhận xét như sau:

- Launch_Date: Thời điểm ra mắt CPU rơi vào khoảng từ năm 2000 đến năm 2018, và trong nửa đầu năm 2013 có lượng CPU được sản xuất ra lớn nhất với khoảng 500 CPU. Giá trị trung bình bằng giá trị trung vị là 13.12 chứng tỏ biến "Launch_Date" có hình dáng phân phối lệch đối xứng.
- Recommended_Customer_Price: Giá bán khuyến nghị của CPU dao động trong khoảng từ $2.54 đến $13011, tuy nhiên phần lớn CPU có giá bán dao động từ $800 đến $1000, với hơn 1000 CPU được đề nghị ở mức giá này. Giá trị trung bình và trung vị đều bằng $850.70, cho thấy rằng biến "Recommended_Customer_Price" có hình dáng phân phối lệch đối xứng.
- nb_of_Cores: Số lượng core dao động từ 1 (nhân) đến 72 (nhân), tập trung nhiều nhất ở khoảng từ 2 (nhân) trở xuống với gần 1300 CPU sử dụng số lượng core này. Giá trị trung bình là 4,067 (nhân) lớn hơn giá trị trung vị là 2 (nhân) chứng tỏ hình dáng phân phối của biến "nb_of_Cores" lệch phải.
- nb_of_Threads: Số lượng thread nằm trong phạm vi từ 1 (luồng) đến 56 (luồng), tuy nhiên, nhiều nhất là hơn 800 CPU sử dụng 9 (luồng). Giá trị trung bình là 8.83 (luồng) nhỏ hơn giá trị trung vị là 9 (luồng) chứng tỏ hình dáng phân phối của biến "nb_of_Threads" lệch trái.
- Processor_Base_Frequency: Tốc độ dao động 3.2×10^7 (Hz) đến 4.3×10^9 (Hz), tốc độ phân bố tập trung trong khoảng 2.2×10^9 - 2.3×10^9 (Hz) với hơn 100 CPU. Giá trị trung bình là 2.22×10^9 (Hz) bằng giá trị trung vị chứng tỏ hình dáng phân phối của biến "Processor_Base_Frequency" đối xứng.
- Cache: Dung lượng của bộ nhớ cache dao động từ 8000 (byte) đến 6×10^7 (byte), tập trung nhiều nhất dưới khoảng 0.1×10^7 (byte) với hơn 500 CPU có dung lượng ở mức này. Giá trị trung bình là 7.08×10^6 (byte) lớn hơn giá trị trung vị là 3×10^6 (byte) chứng tỏ hình dáng phân phối của biến "Cache" lệch phải.

- Bus_Speed: Tốc độ xử lý dữ liệu trong một giây của CPU nằm trong khoảng 0 đến 9.6×10^9 (Hz/s), phân bố nhiều nhất trong khoảng dưới 0.2×10^9 (Hz/s) với gần 900 CPU. Giá trị trung bình là 1.55×10^9 (Hz/s) lớn hơn giá trị trung vị là 5.33×10^8 (Hz/s) chứng tỏ hình dáng phân phối của biến "Bus_Speed" lệch phải.
- Max_Memory_Size: Dung lượng tối đa của bộ nhớ thay đổi trong phạm vi từ 10^9 (byte) đến $4,1 \times 10^{12}$ (byte), và tập trung chủ yếu dưới mức 0.1×10^{12} (byte) với xấp xỉ 900 CPU. Giá trị trung bình là 2.689×10^{11} (byte) bằng với trung vị chứng tỏ hình dáng phân phối của biến "Max_Memory_Size" đối xứng.
- Max_Memory_Bandwidth: Dung lượng tối đa của băng thông thay đổi trong phạm vi từ 1.6 (GB/s) đến 352 (GB/s), và phân bố nhiều nhất từ 30 - 40 (GB/s) với gần 1300 CPU. Giá trị trung bình là bằng với giá trị trung vị là 35.08 (GB/s) chứng tỏ hình dáng phân phối của biến "Max_Memory_Bandwidth" đối xứng.
- Product_Collection: Tất cả các CPU trong tập dữ liệu đều nằm trong 7 bộ sưu tập sản phẩm chính là "Atom", "Celeron", "Core" "Itanium", "Quark" và "Xeon". Tuy nhiên phần lớn các CPU thuộc bộ sưu tập "Xeon", với 711 CPU ở bộ sưu tập này.

Tiếp theo đó, nhóm tiến hành vẽ biểu đồ tương quan và lập bảng hệ số tương quan giữa các biến được chọn từ file Intel_CPUs nhằm trực quan hóa sự phụ thuộc tuyến tính giữa các biến:



Hình 22: *Biểu đồ tương quan ứng với hệ số tương quan giữa từng cặp biến trong file Intel_CPUs*

Qua các dữ liệu về tương quan tuyến tính như trên, nhóm đưa ra nhận xét như sau:

- Launch_Date: Biến "Launch_Date" có tương quan thuận với các biến khác. Hầu hết các hệ số tương quan với các biến khác đều nằm trong khoảng tương quan rất yếu. Do đó, có thể nói rằng khi biến "Launch_Date" tăng sẽ không tác động đáng kể đến các biến còn lại.
- Recommeded_Customer_Price: Biến "Recommeded_Customer_Price" có tương quan thuận với hầu hết các biến còn lại. Biến này có tương quan ở mức yếu và trung bình với các biến, và có tương

quan cao nhất với hai biến "Cache" và "nb_of_Threads". Chính vì thế, giá bán khuyến nghị sẽ càng cao nếu CPU có dung lượng bộ nhớ cache và tốc độ xử lý dữ liệu càng lớn.

- nb_of_Cores: Biến "nb_of_Cores" có tương quan thuận với các biến khác. Hầu hết sự tương quan với các biến khác nằm ở mức từ yếu đến trung bình. Đồng thời, tương quan tốt nhất với hai biến "Cache" và "Max_Memory_Bandwidth". Chính vì thế, nếu số lượng nhân của CPU tăng thì dung lượng của bộ nhớ cache.
- nb_of_Threads: Biến "nb_of_Threads" tương quan thuận với các biến còn lại trong tập dữ liệu. Hầu hết các hệ số tương quan với các biến khác đều nằm trong khoảng tương quan trung bình, và tương quan mạnh nhất với biến "Cache" và "Recommeded_Customer_Price". Như nhóm đã trình bày phía trên, càng nhiều luồng thì giá bán của CPU sẽ càng cao, tương tự với bộ nhớ cache.
- Cache: Biến "Cache" tương quan thuận khá tốt với hầu hết các biến còn lại. Ngoại trừ tương quan yếu với "Launch_Date", thì tương quan với các biến còn lại đều nằm ở mức trung bình. Vì vậy, khi tăng dung lượng bộ nhớ cache thì gần như các thành phần còn lại của CPU cũng tăng theo.
- Processor_Base_Frequency: Biến "Processor_Base_Frequency" gần như không tương quan với tất cả các biến còn lại. Chứng tỏ rằng nếu tăng tốc độ này, thì các thành phần còn lại của CPU cũng không thay đổi.
- Bus_Speed: Biến "Bus_Speed" tương quan thuận với các biến còn lại. Phần lớn biến "Bus_Speed" tương quan với các biến khác ở mức yếu, với biến "Cache" và "Max_Memory_Size" là có tương quan tốt nhất. Nhờ vậy, nếu tăng tốc độ xử lý dữ liệu lên thì dung lượng bộ nhớ cache và kích thước bộ nhớ chính cũng tăng.
- Max_Memory_Size và Max_Memory_Bandwidth: Hai biến này tương quan thuận khá đồng đều với các biến khác. Các hệ số tương quan đều nằm ở mức trung bình yếu. Do đó, khi tăng dung lượng tối đa của bộ nhớ và băng thông bộ nhớ lên thì các thành phần khác của CPU cũng tăng.

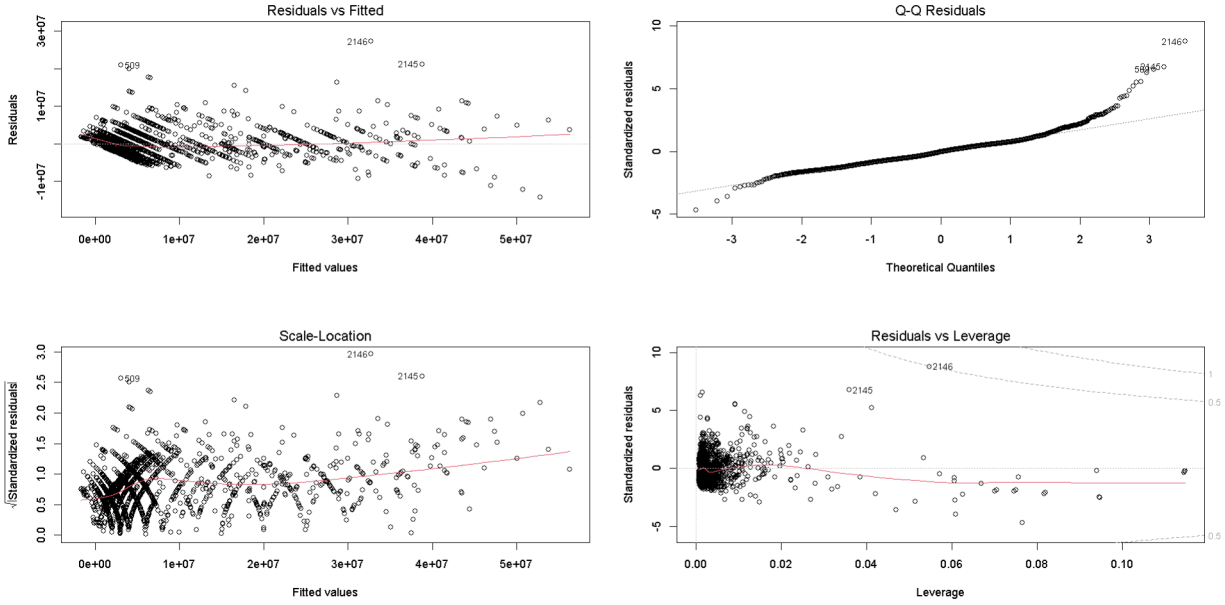
Qua những nhận xét trên, nhóm tác giả nhận thấy biến "Cache" có tương quan tốt nhất với phần lớn các biến khác trong CPU. Chính vì thế, sự phụ thuộc tuyến tính của bộ nhớ của cache với các thành phần còn lại sẽ được tiến hành khảo sát.

5 Thống kê suy diễn

5.1 Giả định

5.1.1 Phân phối chuẩn

Ta tiến hành kiểm định phần dư của mô hình :



Nhận xét: Từ bốn hình trên, ta suy ra được phần dư trong mô hình có dạng phân phối chuẩn. Ta chuyển sang bước tiếp theo.

5.1.2 Quan hệ tuyến tính

```
Call:
lm(formula = new_cpu_df$Cache ~ new_cpu_df$Launch_Date + new_cpu_df$Recommended_Customer_Price
+
    new_cpu_df$nb_of_Cores + new_cpu_df$nb_of_Threads + new_cpu_df$Processor_Base_Frequency +
    new_cpu_df$Bus_Speed + new_cpu_df$Max_Memory_Size + new_cpu_df$Max_Memory_Bandwidth,
    data = df)

Residuals:
      Min        1Q    Median        3Q       Max
-14337351  -2055155    -79194   1739865  27253771

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                     -7.378e+06  3.683e+05 -20.030   <2e-16 ***
new_cpu_df$Launch_Date           2.670e+05  2.131e+04  12.529   <2e-16 ***
new_cpu_df$Recommended_Customer_Price  1.212e+03  9.854e+01  12.301   <2e-16 ***
new_cpu_df$nb_of_Cores           4.946e+05  1.798e+04  27.516   <2e-16 ***
new_cpu_df$nb_of_Threads         3.024e+05  1.620e+04  18.675   <2e-16 ***
new_cpu_df$Processor_Base_Frequency    1.354e-03  8.159e-05  16.594   <2e-16 ***
new_cpu_df$Bus_Speed             1.193e-03  3.493e-05  34.165   <2e-16 ***
new_cpu_df$Max_Memory_Size       2.491e-06  2.419e-07  10.299   <2e-16 ***
new_cpu_df$Max_Memory_Bandwidth -7.177e+03  4.617e+03  -1.554     0.12
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3193000 on 2274 degrees of freedom
Multiple R-squared:  0.8811,    Adjusted R-squared:  0.8807
F-statistic:  2106 on 8 and 2274 DF,  p-value: < 2.2e-16
```

Hình 23: *Sự tuyến tính giữa các biến*

Nhận xét: Từ kết quả trên, ta có thể thấy R-squared = 0.8811 (khá gần 1). Điều đó đồng nghĩa với việc phần lớn sự phụ thuộc của biến Cache có thể được giải thích bởi đa số các biến độc lập trong mô hình.

5.1.3 Kiểm tra đa cộng tuyến

```
                    new_cpu_df$Launch_Date new_cpu_df$Recommended_Customer_Price
                                  1.148612                                2.586646
                    new_cpu_df$nb_of_Cores                 new_cpu_df$nb_of_Threads
                                  2.897664                                3.059692
new_cpu_df$Processor_Base_Frequency                     new_cpu_df$Bus_Speed
                                  1.008028                                1.704480
              new_cpu_df$Max_Memory_Size          new_cpu_df$Max_Memory_Bandwidth
                                  2.454778                                2.324110
```

Hình 24: *Hệ số lạm phát phương sai*

Nhận xét: Đa cộng tuyến là hiện tượng các biến độc lập trong mô hình hồi quy phụ thuộc tuyến tính lẫn nhau. Ta sử dụng hệ số lạm phát phương sai (VIF) để kiểm tra tính đa cộng tuyến giữa các biến. VIF là thước đo mức độ sự biến thiên phương sai của hồi quy tuyến tính dựa vào hiện tượng đa cộng tuyến.

Thông thường, hệ số lạm phát lớn hơn 5 cho thấy mức độ đa cộng tuyến có vấn đề. Trong kết quả trên, không có biến nào có VIF trên 5 nên khả năng xảy ra đa cộng tuyến không phải là vấn đề trong mô hình.

5.2 Hồi quy tuyến tính bội

Trong phần này, nhóm sẽ xây dựng mô hình hồi quy tuyến tính bội với Cache là biến phụ thuộc và Launch_date, Price, Cores, Threads, Frequency, Speed, Max_Memory_size và Max_Memory_Bandwidth là biến độc lập. Mục tiêu của nhóm tác giả là điều tra mối quan hệ giữa Cache và các độc lập này các biến và để phát triển một mô hình dự đoán ước tính chính xác Cache dựa trên các biến này các nhân tố. Phân tích này sẽ cung cấp những hiểu biết có giá trị về các yếu tố tác động đáng kể hiệu suất hệ thống máy tính và tạo điều kiện tối ưu hóa cấu hình hệ thống trong tương lai. Xây dựng mô hình thống kê bằng cách sử dụng hàm Lm trong R. Hàm lm cho phép ước tính mối quan hệ giữa một biến phụ thuộc và một hoặc nhiều biến độc lập.

5.2.1 Xây dựng mô hình hồi quy tuyến tính bội

Kiểm định giả thuyết thống kê:

- H0: Hệ số hồi quy không có ý nghĩa thống kê
- H1: Hệ số hồi quy có ý nghĩa thống kê

Với mức ý nghĩa (significance level) là 5%, nhân tố nào có Pr(>t) lớn hơn 0.05 sẽ không đạt mức ý nghĩa thống kê và không được coi là có ảnh hưởng đáng kể đến giá nhà. Từ bảng kết quả của mô hình hồi quy tuyến tính, Pr(>t) của condition2 lớn hơn 0.05, vì vậy ta chấp nhận giả thuyết H cho nhân tố condition2. Tức là tất cả các nhân tố trừ nhân tố condition2 được giữ lại trong mô hình và được coi là có ảnh hưởng đáng kể đến bộ nhớ đệm.

```
Residuals:
      Min        1Q    Median        3Q       Max
-14337351  -2055155    -79194   1739865  27253771

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                     -7.378e+06  3.683e+05 -20.030   <2e-16 ***
new_cpu_df$Launch_Date           2.670e+05  2.131e+04  12.529   <2e-16 ***
new_cpu_df$Recommended_Customer_Price  1.212e+03  9.854e+01  12.301   <2e-16 ***
new_cpu_df$nb_of_Cores           4.946e+05  1.798e+04  27.516   <2e-16 ***
new_cpu_df$nb_of_Threads         3.024e+05  1.620e+04  18.675   <2e-16 ***
new_cpu_df$Processor_Base_Frequency    1.354e-03  8.159e-05  16.594   <2e-16 ***
new_cpu_df$Bus_Speed             1.193e-03  3.493e-05  34.165   <2e-16 ***
new_cpu_df$Max_Memory_Size       2.491e-06  2.419e-07  10.299   <2e-16 ***
new_cpu_df$Max_Memory_Bandwidth -7.177e+03  4.617e+03  -1.554     0.12
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3193000 on 2274 degrees of freedom
Multiple R-squared:  0.8811,    Adjusted R-squared:  0.8807
F-statistic:  2106 on 8 and 2274 DF,  p-value: < 2.2e-16
```

Hình 25: *Kết quả mô hình hồi quy tuyến tính bội*

Từ kết quả trên ta có thể phân tích :

- Mô hình hồi quy tuyến tính bội đã được trang bị và phần dư nằm trong khoảng từ −14337351 đến 27253771 với Quartile 1: −2055155, Trung vị: −79194 và Quartile 3: 17399865. Điều này ngụ ý rằng có sự phân tán đáng kể trong các sai số dự đoán, bao gồm một số sai số dương và âm đặc biệt lớn.
- Các hệ số ước lượng (hoặc tham số) của mô hình thể hiện hướng và độ lớn của mối quan hệ giữa các biến độc lập (Launch_date, Price, Cores, Threads, Frequency, Speed, Max_Memory_Size, Max_Memory_Bandwidth) và biến phụ thuộc. Trong trường hợp này, tất cả các biến dự đoán đều có mối quan hệ dương với biến phụ thuộc ứng, trừ Max_Memory_Bandwidth có mối quan hệ âm. Tuy nhiên, biến độc lập Max_Memory_Bandwidth không đáng kể (giá trị p = 0,12) vì nó lớn hơn một mức độ thông thường là 0,05. Điều này cho thấy rằng Max_Memory_Bandwidth không có ý nghĩa thống kê trong mô hình.
- Các biến còn lại (Launch_date, Price, Cores, Threads, Frequency, Speed, Max_Memory_Size) có giá trị p thấp 0,05, cho thấy chúng là các yếu tố dự báo có ý nghĩa thống kê.
- Sai số chuẩn dư (RSE) của mô hình là khoảng 3193000. Giá trị này là thước đo chắc chắn về độ lệch điển hình của giá trị thực tế so với giá trị dự đoán.
- Giá trị R-squared bội (0,8811) và R-squared điều chỉnh (0,8807) gợi ý rằng mô hình giải thích một tỷ lệ đáng kể về sự thay đổi trong phản ứng. Đã điều chỉnh R-squared, xử phạt việc bổ sung các yếu tố dự đoán không cần thiết vào mô hình, hơi khó thấp hơn bội số R bình phương, cho thấy mô hình phù hợp tốt.
- Thống kê F tổng thể (2106) và giá trị p tương ứng của nó (nhỏ hơn $2,2 \times 10^{-16}$) kiểm tra giả thuyết rằng ít nhất một trong các yếu tố dự đoán có ích trong việc giải thích phản ứng. Cho giá trị p rất nhỏ, chúng tôi bác bỏ giả thuyết không, chỉ ra rằng ít nhất một yếu tố dự đoán đang đóng góp đáng kể vào việc dự đoán phản ứng.

5.2.2 Hồi quy stepwise

Từ kết quả trên, ta đưa ra hai mô hình thống kê như sau:

Mô hình thứ nhất bao gồm các biến: Launch_date, Price, nb_of_Cores, nb_of_Threads, Frequency, Speed, Max_Memory_Size, Max_Memory_Bandwidth.

```
Start:  AIC=55946.81
Cache ~ Launch_Date + Recommended_Customer_Price + nb_of_Cores +
    nb_of_Threads + Processor_Base_Frequency + Bus_Speed + Max_Memory_Size +
    Max_Memory_Bandwidth

                              Df  Sum of Sq        RSS   AIC
<none>                                      1.8235e+16 55947
- Max_Memory_Bandwidth         1 3.7193e+13 1.8273e+16 55949
- Max_Memory_Size              1 7.9979e+14 1.9035e+16 56025
- Launch_Date                  1 1.4375e+15 1.9673e+16 56087
- Recommended_Customer_Price   1 1.6100e+15 1.9845e+16 56103
- Processor_Base_Frequency     1 2.2988e+15 2.0534e+16 56167
- nb_of_Threads                1 2.9532e+15 2.1189e+16 56225
- nb_of_Cores                  1 6.8847e+15 2.5120e+16 56544
- Bus_Speed                    1 8.6398e+15 2.6875e+16 56670
```

Hình 26: *Kết quả mô hình hồi quy tuyến tính bội thứ nhất*

Mô hình thứ hai bao gồm các biến ban đầu và ta loại bỏ biến Max_Memory_Bandwidth vì biến đó không có ý nghĩa thống kê:

```
Start:  AIC=55948.62
Cache ~ Launch_Date + Recommended_Customer_Price + nb_of_Cores +
    nb_of_Threads + Processor_Base_Frequency + Bus_Speed + Max_Memory_Size

                            Df  Sum of Sq        RSS   AIC
<none>                                    1.8273e+16 55949
- Max_Memory_Size            1 7.6272e+14 1.9035e+16 56023
- Launch_Date                1 1.5434e+15 1.9816e+16 56098
- Recommended_Customer_Price 1 1.6664e+15 1.9939e+16 56110
- Processor_Base_Frequency   1 2.3187e+15 2.0591e+16 56170
- nb_of_Threads              1 3.0493e+15 2.1322e+16 56235
- Bus_Speed                  1 8.6292e+15 2.6902e+16 56670
- nb_of_Cores                1 1.1697e+16 2.9970e+16 56872
```

Hình 27: *Kết quả mô hình hồi quy tuyến tính bội thứ hai*

Như chúng ta có thể thấy từ kết quả trên: Khi so sánh hai mô hình, mô hình với giá trị AIC thấp hơn thì ta ưu tiên. Nếu mô hình mới (sau khi loại bỏ một biến) có giá trị AIC thấp hơn mô hình cũ, điều này có thể được coi là một dấu hiệu tích cực về việc loại bỏ biến đó. Như vậy, sau khi loại bỏ biến Max_Memory_Bandwidth, ta nhận được mô hình có giá trị AIC cao hơn. Vì vậy, ta sẽ thêm biến đó vào lại mô hình mặc dù biến đó không mang ý nghĩa thống kê.

5.2.3 Phương trình tuyến tính

Sau khi áp dụng kiểm tra giả định và hồi quy từng bước, chúng ta có bội số tuyến tính cuối cùng mô hình hồi quy như đã kết luận. Các hệ số hồi quy cho từng biến độc lập trong mô hình như sau: $\text{Cache} = -7378000 + 267000 * \text{Launch\_date} + 1212 * \text{Price} + 494600 * \text{nb\_of\_Cores} + 302400 * \text{nb\_of\_Thread} + 0.001354 * \text{Frequency} + 0.001193 * \text{Speed} + 0.000002 * \text{Max\_Memory\_size} + (-7177) * \text{Max\_Memory\_Bandwidth}$

Để kết luận, các mô hình hồi quy tuyến tính bội được cung cấp gợi ý rằng các biến bao gồm Launch_date, Price, nb_of_Cores, nb_of_Threads, Frequency, Max_Memory_size và Max_Memory_Bandwidth .

5.2.4 Dự đoán mô hình

Tiếp theo nhóm kiểm tra phương trình trong hình để dự đoán các giá trị Cache trong bộ dữ liệu thử nghiệm nhóm đã chia ở trên.

	test_set	prediction
17	2000000	4470947.3
24	2000000	4379477.6
42	2000000	2923916.8
47	2000000	963877.7
50	512000	2047216.8
51	1000000	3058357.2
70	512000	-233623.8
72	512000	1940295.7
75	256000	2181657.2
78	512000	1805855.3

Hình 28: *So sánh giá trị thử nghiệm và dự đoán*

Một số nhận xét:

- Đối với quan sát đầu tiên (chỉ số 17), Cache là 2 000 000 và mô hình dự đoán giá trị 4470947.3
- Trong quan sát thứ hai (chỉ số 24), Cache là 2 000 000 và mô hình dự đoán giá trị 4379477.6
- Có một số giá trị dự đoán âm (chỉ số 70). Điều này có thể là một vấn đề vì biến độc lập không thể có giá trị âm. Cần kiểm tra lại mô hình và dữ liệu để xác định tại sao dự đoán này lại âm.

- Có thể có sự biến động lớn giữa giá trị dự đoán trong một số trường hợp (tại chỉ số 17, 24, 42).
- Chỉ số 50 và chỉ số 78 có sự gần giống đáng kể, điều này cho thấy mô hình hoạt động tốt

Tóm lại, dựa trên thông tin hiển thị các giá trị đem lại số liệu hiệu quả cho thấy mô hình hoạt động tốt , một vài chỉ số của mô hình cần được điều chỉnh hoặc cải thiện để đảm bảo rằng nó hoạt động tốt trên các giá trị khác, một vài giá trị đem lại số liệu hiệu quả.

6 Thảo luận và mở rộng

6.1 Thảo luận về mô hình hồi quy tuyến tính

6.1.1 Ưu điểm

- Đa số các vấn đề thực tế đòi hỏi phân tích của nhiều biến, không chỉ một biến đơn lẻ. Hồi quy tuyến tính bội sử dụng để phân tích mối quan hệ giữa một biến phản hồi với nhiều biến độc lập.
- Sử dụng hồi quy tuyến tính bội cho phép phân tích ảnh hưởng của nhiều yếu tố đến biến phản hồi cùng một lúc, giúp đưa ra kết luận chính xác hơn về mối quan hệ giữa các biến.
- Hồi quy tuyến tính bội cũng cung cấp các thông tin về độ mạnh của mối quan hệ giữa biến phản hồi và các biến độc lập, giúp đánh giá tác động của một biến đến biến phụ thuộc khi các biến khác được giữ ở mức giá trị không đổi.

6.1.2 Nhược điểm

Mô hình hồi quy tuyến tính giả định rằng mối quan hệ giữa các biến là tuyến tính, điều này có thể không phản ánh đúng với các dữ liệu phức tạp trong nhiều tình huống thực tế. Do đó, nếu mối quan hệ thực sự không tuyến tính, mô hình hồi quy tuyến tính có thể không phản ánh chính xác mức độ ảnh hưởng của các biến độc lập lên biến phụ thuộc. Các giá trị ngoại lệ có thể ảnh hưởng lớn đến mô hình, đặc biệt là ảnh hưởng đến các ước lượng của các hệ số, những điểm có thể làm suy giảm chính xác của mô hình và dẫn đến độ chính xác thấp.

6.1.3 Ứng dụng

- Kinh tế học: Hồi quy tuyến tính bội có thể được sử dụng để dự đoán giá cổ phiếu, giá vàng, lợi nhuận doanh nghiệp.
- Y học: Hồi quy tuyến tính bội có thể được sử dụng để xác định mối quan hệ giữa các yếu tố ảnh hưởng đến sức khỏe như thói quen ăn uống, vận động, môi trường sống và bệnh.
- Khoa học xã hội: Hồi quy tuyến tính bội có thể được sử dụng để xác định mối quan hệ giữa các yếu tố tác động đến hạnh phúc, chất lượng cuộc sống và sự hài lòng của con người.
- Khoa học tự nhiên: Hồi quy tuyến tính bội có thể được sử dụng để xác định mối quan hệ giữa các yếu tố ảnh hưởng đến môi trường, khí hậu và sự phát triển của động thực vật.

6.2 Ứng dụng mô hình hồi quy đa thức

Ở phần mở rộng này, nhóm tác giả xây dựng tập dữ liệu trên các mô hình đa thức bậc n. Cụ thể, khi cho bậc của mô hình chạy từ 1 (cũng là mô hình gốc) đến 10 thu được kết quả như sau:

Mô hình hồi quy đa thức bậc n	Residual standard error	Multiple R-squared	Adjusted R-squared	F-statistic	p-value
1	3203000	0.8803	0.8799	2094	$< 2.2e\text{-}16$
2	2588000	0.9222	0.9216	1678	
3	2478000	0.9289	0.9281	1229	
4	2378000	0.9347	0.9338	1007	
5	2339000	0.9371	0.9359	834.7	
6	2291000	0.9389	0.9385	727	
7	2282000	0.9406	0.9391	628.9	
8	2264000	0.9417	0.9400	559.5	
9	2223000	0.9440	0.9421	517	
10	2203000	0.9452	0.9432	474.5	

Bảng 2: *Kết quả của mô hình hồi quy đa thức bậc 1 đến bậc 10*

Tất cả các mô hình ở đây đều có giá trị p-value < 0.05 chứng tỏ kết quả có ý nghĩa thống kê, nghĩa là giả thuyết kiểm định H0 bị bác bỏ. Bên cạnh đó, khi bậc của đa thức càng cao thì biến "Cache" được giải thích càng nhiều bởi các biến độc lập. So với mô hình gốc là $88,03\%$, thì mô hình hồi quy đa thức bậc 10 có đến $94,52\%$ biến "Cache" được giải thích bởi các biến độc lập đã chọn. Tiếp theo, dự đoán giá trị "Cache" bằng mô hình hồi quy tuyến tính và mô hình hồi quy đa thức bậc 10 qua tập dữ liệu "Test" (20% của tập dữ liệu ban đầu), được kết quả sau:

	testing_Cache	predictedbyPoly10	predictedbyLinear	DifferencePoly10	DifferenceLinear	model_type
4	10000000	8514686.6	5511752.22	1.485313e+06	4488247.78	Poly10
6	2000000	2733258.4	543770.12	7.332584e+05	1456229.88	Poly10
11	2000000	2282122.2	1173595.09	2.821222e+05	826404.91	Poly10
13	2000000	2542159.1	1751483.36	5.421591e+05	248516.64	Linear
14	2000000	2786917.9	7038003.96	7.869179e+05	5038003.96	Poly10
16	8000	8786195.4	206879.16	8.778195e+06	198879.16	Linear
19	2000000	4697715.8	5162971.98	2.697716e+06	3162971.98	Poly10
24	2000000	3408513.8	4627385.12	1.408514e+06	2627385.12	Poly10
26	2000000	3836933.0	3841635.98	1.836933e+06	1841635.98	Poly10
28	2000000	3408128.6	6827527.58	1.408129e+06	4827527.58	Poly10

Hình 29: *Dự đoán 10 giá trị đầu tiên bằng hai mô hình*

```
Percentage of observations classified as 10th-degree polynomial regression model: 73.83592 %
Percentage of observations classified as linear regression model: 26.16408 %
```

Hình 30: *Tỷ lệ dự đoán gần giá trị quan sát hơn của hai mô hình*

Kết quả này in ra giá trị kiểm tra, giá trị dự đoán, sai số so với giá trị kiểm tra của hai mô hình và cuối cùng in ra mô hình nào có giá trị dự đoán tốt hơn. Khi dự đoán giá trị "Cache" trong tập dữ liệu "Test" thì tỉ lệ mà mô hình hồi quy đa thức bậc 10 cho giá trị dự đoán gần giá trị quan sát hơn chiếm gần 74%. Chính vì thế, có thể nói rằng mô hình hồi quy tuyến tính có kết quả dự đoán chưa tốt bằng các mô hình khác.

7 Nguồn dữ liệu và nguồn code

Nguồn file Intel_CPUs: https://www.kaggle.com/datasets/iliassekkaf/computerparts/

Nguồn code: https://drive.google.com/file/d/1igl91hhSkc3IQ9yJRrhEjbRrGBzT9KYX/view?usp=sharing

8 Tài liệu tham khảo

- H. Kwak, B. Lee, A. R. Hurson, Suk-Han Yoon and Woo-Jong Hahn, "Effects of multithreading on cache performance," in IEEE Transactions on Computers, truy cập tại: https://ieeexplore.ieee.org/document/752659
- Prisaganec, Milco & Mitrevski, Pece, (2016), *Reducing Competitive Cache Misses in Modern Processor Architectures*, truy cập tại: https://www.researchgate.net/publication/303895846_Reducing_Competitive_Cache_Misses_in_Modern_Processor_Architectures
- Green, O., Fox, J., Young, J., Shirako, J., & Bader, D, (2019), *Performance Impact of Memory Channels on Sparse and Irregular Algorithms*, truy cập tại: https://arxiv.org/pdf/1910.03679.pdf
- Patterson, D. A., & Hennessy, J. L. (2004), *Computer Organization and Design: The Hardware/Software Interface.*
- Algorithmica, *RAM& CPU Caches: Memory Bandwidth*, truy cập tại: https://en.algorithmica.org/hpc/cpu-cache/bandwidth/
- Diana Mindrila, Ph.D & Phoebe Balentyne, M.Ed, *Scatterplots and Correlation*, truy cập tại: https://www.westga.edu/academics/research/vrc/assets/docs/scatterplots_and_correlation_notes.pdf
- Trường Đại học Kinh tế - Luật, ĐHQG, TPHCM, *Khảo sát hình dạng phân phối của tập dữ liệu*, truy cập tại: https://maths.uel.edu.vn/Resources/Docs/SubDomain/maths/TaiLieuHocTap/ToanUngDung/kho_st_hnh_dng_phn_phi_ca_tp_d_liu.html
- ThS. Nguyễn Chí Minh Trung, *Hồi quy và tương quan*, truy cập tại: http://www.ctump.edu.vn/DesktopModules/NEWS/DinhKem/6426_Slide-bai-giang-thuc-tap-SPSS-Buoi-4-ThsTrung.pdf