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

BÁO CÁO BÀI TẬP LỚN
MÔN HỌC: XÁC SUẤT THỐNG KÊ

ĐỀ TÀI 2:

PHÂN TÍCH DỮ LIỆU CỦA GPU VÀ CPU VÀ XÂY DỰNG MÔ HÌNH DỰ ĐOÁN GIÁ BÁN BẰNG MÔ HÌNH HỒI QUY TUYẾN TÍNH

Giảng viên hướng dẫn : NGUYỄN THỊ MỘNG NGỌC

Nhóm sinh viên thực hiện : 07

Lớp : L11

TP Hồ Chí Minh, ngày 09 tháng 12 năm 2023

BÁO CÁO KẾT QUẢ LÀM VIỆC NHÓM

MSSV	Họ	Tên	Nhiệm vụ được phân công	Tỷ lệ % tham gia BTL	Ký tên	Điểm chia
2210737	Nguyễn Huỳnh Hải	Đăng	Thống kê suy diễn Nguồn dữ liệu và code	100%		
2210882	Hà Thiên	Hải	Tổng hợp Tổng quan dữ liệu	100%		
2211937	Trần Văn	Lộc	Thống kê tả Thảo luận và mở rộng Nguồn dữ liệu và code	100%		
2212003	Trần Văn	Mạnh	Tiền xử lý Slide powerpoint	100%		
2213061	Nguyễn Thanh	Tân	Kiến thức nền Thảo luận và mở rộng	100%		

Nhận xét của giảng viên:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

GIẢNG VIÊN

(Ký và ghi rõ họ, tên)

MỤC LỤC

TỔNG QUAN DỮ LIỆU

1.1 Mô tả tóm tắt dữ liệu

Bộ dữ liệu này chứa thông số kỹ thuật chi tiết, ngày phát hành và giá phát hành của các bộ phận máy tính.

Bộ dữ liệu chứa hai tệp CSV: All_GPUs.csv cho Đơn vị Xử lý Đồ Họa (GPUs) và Intel_CPUs.csv cho Đơn vị Xử lý Trung Tâm (CPUs). Mỗi bảng có danh sách của các mục riêng, nhưng về đặc điểm chung thì bao gồm: tốc độ đồng hồ, nhiệt độ tối đa, độ phân giải màn hình, công suất tiêu thụ, số luồng, ngày phát hành, giá phát hành, kích thước chip, hỗ trợ ảo hóa và nhiều trường tương tự khác. Và nhóm đã thống nhất chọn dữ liệu GPU.

1.2 Mục đích của bộ dữ liệu

Hiệu suất so với tỷ lệ giá đã thay đổi như thế nào theo thời gian?

Sức mạnh tính toán nói chung đang như thế nào?

Có nhà sản xuất nào được biết đến với một số phạm vi hiệu suất và giá cả cụ thể không?

1.3 Graphics Processing Unit (GPU)

1.3.1 Khái niệm

GPU là bộ xử lý những tác vụ liên quan đến đồ hoạ cho vi xử lý trung tâm CPU. Rất nhiều tính năng trên GPU vượt xa so với trình điều khiển đồ họa cơ bản như GPU của Intel. GPU được dùng trong các hệ thống nhúng, máy tính cá nhân, máy trạm workstation, máy tính chơi game...GPU dễ nhận biết nhất là trong máy tính cá nhân, CPU xuất hiện ở Card màn hình hoặc có thể gắn trên Mainboard.

1.3.2 Các biến sử dụng

Max_Power: năng lượng cần để chạy được graphics card (Watts).

Boost_Clock: tốc độ của graphics card (MHz).

Core_Speed: tốc độ lõi của graphics card (MHz).
Memory: Bộ nhớ của card đồ họa.
Memory_Bandwidth: băng thông bộ nhớ của graphic card (GB/sec).
Memory_Bus: bus bộ nhớ của graphic card (Bit).
Memory_Speed: tốc độ bộ nhớ RAM của graphic card (MHz).
Release_Price: Giá bán của GPU.
Texture_Rate: Số pixel kết cấu mỗi giây GPU có thể tạo ra.

1.3.3 Thống kê phần mềm RStudio

Bộ dữ liệu All_GPUs.csv có 3406 giá trị quan trắc, 34 biến.

1.4 Nguồn của dữ liệu

Dữ liệu được cung cấp ở đây chủ yếu thuộc về Intel, Game-Debate và các công ty liên quan đến việc sản xuất các bộ phận này.

KIẾN THỨC NỀN

2.1 Khái niệm về một số đại lượng cơ bản của thống kê

Kì vọng (Expectation/Mean): của biến ngẫu nhiên X là giá trị trung bình theo xác suất của X. Kí hiệu E(X).

Phương sai (Variance): của biến ngẫu nhiên X được định nghĩa bằng trung bình bình phương sai lệch giữa biến ngẫu nhiên với kì vọng của nó. Kí hiệu V(X).

Độ lệch chuẩn (Standard Deviation): của biến ngẫu nhiên có giá trị bằng p V(X).

Trung vị (Median): của X là giá trị nằm chính giữa , chia tập giá trị X ra thành 2 phần bằng nhau.

Cực tiểu (Min): là giá trị nhỏ nhất trong toàn bộ các giá trị của một tập mẫu.

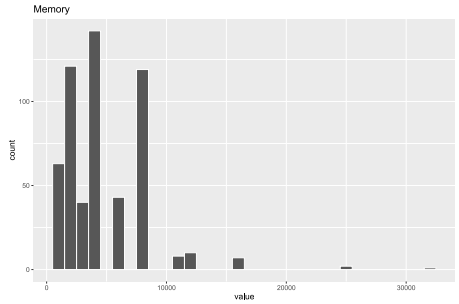
Cực đại (Max): là giá trị lớn nhất trong toàn bộ các giá trị của một tập mẫu.

Tứ phân vị: là đại lượng mô tả sự phân bố và sự phân tán của tập dữ liệu. Tứ phân vị có 3 giá trị, đó là tứ phân vị thứ nhất, thứ nhì, và thứ ba. Ba giá trị này chia một tập hợp dữ liệu (đã sắp xếp dữ liệu theo trật từ từ bé đến lớn) thành 4 phần có số lượng quan sát đều nhau.

- Giá trị tứ phân vị thứ hai Q_2 chính bằng giá trị trung vị.
- Giá trị tứ phân vị thứ nhất Q_1 bằng trung vị phần dưới.
- Giá trị tứ phân vị thứ ba Q_3 bằng trung vị phần trên.
- Giá trị outlines là phần dữ liệu bất thường.

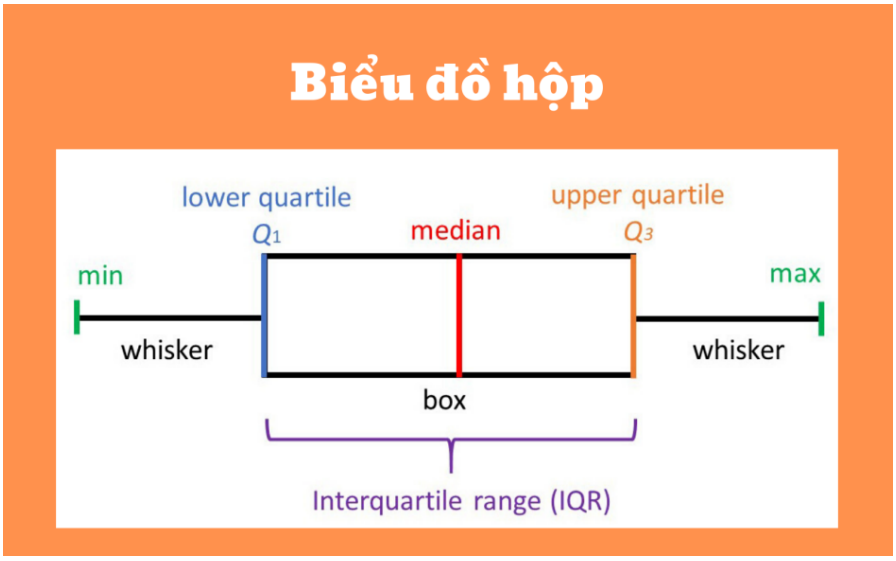
2.2 Một số loại biểu đồ được sử dụng.

Histogram: là biểu đồ tần số dùng cho biến định lượng liên tục nhằm biểu diễn phân phối của tập dữ liệu.



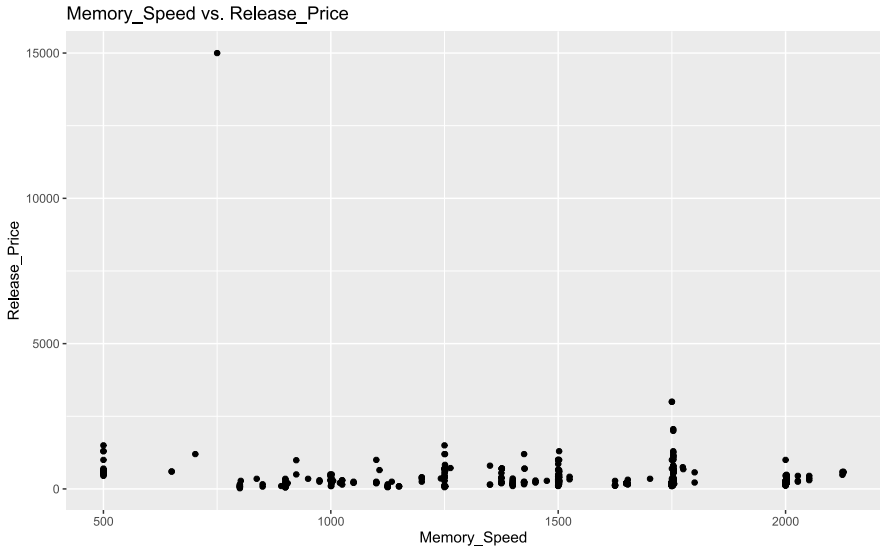
Hình 1: Hình ảnh minh họa Histogram.

Biểu đồ boxplots: là biểu đồ diễn tả 5 vị trí phân bổ của dữ liệu, đó là giá trị nhỏ nhất(Min), tứ phân vị thứ nhất (Q_1), trung vị (Q_2), tứ phân vị thứ ba (Q_3), giá trị lớn nhất (Max).



Hình 2: Hình ảnh minh họa Boxplots.

Scatterplots: Là biểu đồ dạng chấm nhằm thể hiện mối quan hệ giữa hai biến liên tục.



Hình 3: Hình ảnh minh họa Scatterplots.

2.3 Hồi quy tuyến tính bội và phân tích phương sai ANOVA.

Hồi quy tuyến tính bội là một phương pháp được sử dụng khi ta muốn dự đoán giá trị của một biến phản hồi dựa trên giá trị của hai hoặc nhiều biến giải thích khác. Biến mà chúng ta muốn dự đoán được gọi là biến phản hồi (biến phụ thuộc). Các biến mà ta đang sử dụng để dự đoán giá trị của biến phản hồi được gọi là các biến giải thích (biến dự báo, biến phụ thuộc).

Hồi quy bội cũng cho phép chúng ta xác định sự phù hợp tổng thể của mô hình và đóng góp tương đối của từng yếu tố dự báo và tổng phương sai được giải thích.

Mô hình hồi quy tuyến tính bội liên quan đến biến Y và $p-1$ biến dự báo X có dạng như sau:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + ... + \beta_{p-1} X_{i(p-1)} + \varepsilon_i$$

Hay:

$$Y_i = \beta_0 + \sum_{k=1}^{p-1} \beta_k X_{ik} + \varepsilon_i$$

Trong đó:

- Y_i là biến phản hồi hay biến phụ thuộc (response variable / dependent variable).
- X_{ik} là các biến giải thích hay biến độc lập (explanatory variables / independent variables).
- β_k là tham số của các biến độc lập trong đó β_0 là hệ số hồi quy (hệ số chặn).
- $\varepsilon_i \sim N(0, \sigma^2)$ là số hạng nhiễu hay sai số ngẫu nhiên.
- Y_i $i = 1, 2, ..., n$.

Khi $p-1 = 1$ mô hình hồi quy là: $Y_i = \beta_0 + \beta_1 X_{i1} + \varepsilon_i$ là mô hình hồi quy tuyến tính đơn.

Giả định:

- ε_i có phân phối chuẩn và phương sai không đổi.
- Các biến X độc lập với nhau.

Kiểm định từng tham số hồi quy tổng thể:

- Trường hợp $\beta_k = 0$ thì X_i và Y không có mối quan hệ nào.
- Trường hợp $\beta_k > 0$ thì X_i và Y có mối quan hệ thuận.
- Trường hợp $\beta_k < 0$ thì X_i và Y có mối quan hệ nghịch.
- Ở mức ý nghĩa α, giả thuyết vô hiệu H_0 được kiểm định ở các trường hợp sau:
  (1) $H_0 : \beta_k \leq 0$ và $H_1 : \beta_k > 0$
  (2) $H_0 : \beta_k \geq 0$ và $H_1 : \beta_k < 0$
- Giá trị kiểm định: $t = \dfrac{\beta_k}{SE(\beta_k)}$.
- Quyết định bác bỏ giả thuyết H_0:
  - Bác bỏ giả thuyết (1) khi $t > t_{n-p,\alpha}$
  - Bác bỏ giả thuyết (1) khi $t < -t_{n-p,\alpha}$

Ma trận hồi quy:

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1(p-1)} \\ 1 & x_{21} & x_{22} & \dots & x_{2(p-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{n(p-1)} \end{bmatrix}, \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}, \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Với dữ liệu: $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

Ước tính β bằng cực tiểu hóa:

$$f(\beta_0, \beta_1, \dots, \beta p - 1 = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_{p-1} X_{i(p-1)})$$

Khi đó: $\beta = (X^T X)^{-1} X^T y$

Ý nghĩa hồi quy:

- ε_i Trường hợp: $H_0 : \beta_0 = \beta_1 = \ldots = \beta_{p-1} = 0 \rightarrow Y_i = \beta_0 + \varepsilon_i$, kí hiệu các dữ liệu thỏa là y_{0i}. Khi đó: $\overline{y} = y_{0i}$

- Trường hợp: H_1 : có ít nhất một $\beta_j \neq 0, j = 1,2,\ldots,(p-1)$

Biến thiên	Tổng độ lệnh bình phương.	Bậc tự do	Phương sai	Giá trị kiểm định.
Hồi quy	SSR = $\sum_{i=1}^{n}(y_{1i}-\bar{y})^2$	$p-1$	MSR = $\frac{SSR}{p-1}$	F = $\frac{MSR}{MSE}$
Sai số	SSE = $\sum_{i=1}^{n}(y_i-y_{1i})^2$	$n-p$	MSE = $\frac{SSE}{n-p}$	
Tổng	SST = $\sum_{i=1}^{n}(y_i-\bar{y})^2$	$n-1$		

Bảng 1: Bảng ANOVA cho mô hình hồi quy tuyến tính tổng quát.

Nếu $F \leq F_{p-1,n-p,\alpha}$ chấp nhận H_0.

Nếu $F > F_{p-1,n-p,\alpha}$ bác bỏ giả thuyết H_0.

Hệ số xác định $R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$

Đại diện cho tỷ lệ biến thiên theo biến Y_i với các biến dự báo $X_1i, X_2i, \ldots$ Hệ số R^2 nói lên tính chặt chẽ giữa biến phản hồi Y_i và các biến dự báo X_ki.

Ta có: $0 \leq R^2 \leq 1$. Khi $R^2 = 0$ thì chấp nhận H_0. Khi $R^2 = 1$ thì tất cả các quan sát nằm trên mặt đáp ứng.

Nếu chúng ta bắt đầu với một mô hình hồi quy tuyến tính đơn giản với biến dự báo X_1i sau đó thêm biến dự báo X_2i, SSE sẽ giảm (hoặc giữ nguyên) trong khi SST không đổi và do đó R^2 sẽ tăng (hoặc giữ nguyên). Nói cách khác, R^2 luôn tăng (hoặc giữ nguyên) khi có nhiều yếu tố dự báo vào mô hình hồi quy tuyến tính bội, ngay cả khi các yếu tố dự báo được thêm vào không liên quan đến biến phản hồi. Do đó bản thân R^2 không thể được sử dụng để giúp chúng ta xác định những yếu tố dự báo nào nên được đưa vào một mô hình và yếu tố nào nên được loại bỏ.

Hệ số xác định đã điều chỉnh:

$$R^2_\alpha = \frac{\frac{SSR}{n-p}}{\frac{SST}{n-1}} = 1 - (\frac{n-1}{1-p})(1-R^2)$$

TIỀN XỬ LÝ GPU

3.1 Cài đặt thư viện

Để có thể sử dụng các đoạn code được đề cập bên dưới chúng ta phải cài đặt một số thư viện sau:

R Code

```r
install.packages("dplyr")
library(dplyr)
install.packages("ggplot2")
library(ggplot2)
install.packages("patchwork")
library(patchwork)
```

Trong bài chúng tôi đã sử dụng các thư viện trên để:

- dplyr: Thư viện dplyr được sử dụng để thực hiện các hoạt động xử lý dữ liệu, như biến đổi dữ liệu, lọc dữ liệu, tóm tắt dữ liệu và sắp xếp dữ liệu.
- ggplot2: Thư viện ggplot2 được sử dụng để tạo đồ thị và biểu đồ dựa trên dữ liệu.
- patchwork: Thư viện patchwork được sử dụng để tạo và kết hợp các đồ thị ggplot2 thành một đồ thị tổng thể.

3.2 Đọc dữ liệu

Ta dùng lệnh sau để đọc dữ liệu từ file All_GPU.csv và để xem 10 dòng dữ liệu đầu tiên.

R Code

```r
all_gpus <- read.csv("ALL_GPUs.csv")
head(all_gpus,10)
```

	Architecture	Best_Resolution	Boost_Clock	Core_Speed	DVI_Connection	Dedicated	Direct_X	DisplayPort_Connection
1	Tesla G92b			738 MHz	2	Yes	DX 10.0	NA
2	R600 XT	1366 x 768		\n-	2	Yes	DX 10	NA
3	R600 PRO	1366 x 768		\n-	2	Yes	DX 10	NA
4	RV630	1024 x 768		\n-	2	Yes	DX 10	NA
5	RV630	1024 x 768		\n-	2	Yes	DX 10	NA
6	RV630	1024 x 768		\n-	2	Yes	DX 10	NA
7	R700 RV790 XT	1920 x 1080		870 MHz	1	Yes	DX 10.1	NA
8	R600 GT	1024 x 768		\n-	2	Yes	DX 10	NA
9	Pitcairn XT GL	1920 x 1080		\n-	0	Yes	DX 11.2	NA
10	RV100			\n-	NA	Yes	DX 7	NA

	HDMI_Connection	Integrated	L2_Cache	Manufacturer	Max_Power	Memory	Memory_Bandwidth	Memory_Bus	Memory_Speed
1	0	No	0KB	Nvidia	141 Watts	1024 MB	64GB/sec	256 Bit	1000 MHz
2	0	No	0KB	AMD	215 Watts	512 MB	106GB/sec	512 Bit	828 MHz
3	0	No	0KB	AMD	200 Watts	512 MB	51.2GB/sec	256 Bit	800 MHz
4	0	No	0KB	AMD		256 MB	36.8GB/sec	128 Bit	1150 MHz
5	0	No	0KB	AMD	45 Watts	256 MB	22.4GB/sec	128 Bit	700 MHz
6	0	No	0KB	AMD	50 Watts	256 MB	35.2GB/sec	128 Bit	1100 MHz
7	1	No	0KB	AMD	190 Watts	2048 MB	134.4GB/sec	256 Bit	1050 MHz
8	0	No	0KB	AMD	150 Watts	256 MB	51.2GB/sec	256 Bit	800 MHz
9	0	No	0KB	AMD	150 Watts	2048 MB	160GB/sec	256 Bit	1250 MHz
10	NA	No	0KB	AMD	32 Watts	64 MB	2.9GB/sec	64 Bit	366 MHz

	Memory_Type	Name	Notebook_GPU	Open_GL	PSU	Pixel_Rate
1	GDDR3	GeForce GTS 150	No	3.3	450 Watt & 38 Amps	12 GPixel/s
2	GDDR3	Radeon HD 2900 XT 512MB	No	3.1	550 Watt & 35 Amps	12 GPixel/s
3	GDDR3	Radeon HD 2900 Pro	No	3.1	550 Watt & 35 Amps	10 GPixel/s
4	GDDR4	Radeon HD 2600 XT Diamond Edition	No	3.3		3 GPixel/s
5	GDDR3	Radeon HD 2600 XT	No	3.1	400 Watt & 25 Amps	3 GPixel/s
6	GDDR4	Radeon HD 2600 XT 256MB GDDR4	No	3.3	400 Watt & 26 Amps	3 GPixel/s
7	GDDR5	Radeon HD 4890 Sapphire Vapor-X OC 2GB Edition	No	3.3	500 Watt & 37 Amps	14 GPixel/s
8	GDDR3	Radeon HD 2900 GT	No	3.1	550 Watt & 30 Amps	7 GPixel/s
9	GDDR5	FirePro D300	No	4.3	500 Watt & 20 Amps	25 GPixel/s
10	DDR	Radeon 7000 64mb	No	1.4		

	Power_Connector	Process	ROPs	Release_Date	Release_Price	Resolution_WxH	SLI_Crossfire	Shader	TMUs	Texture_Rate
1	None	55nm	16	\n01-Mar-2009		2560x1600	Yes	4.0	64	47 GTexel/s
2	None	80nm	16	\n14-May-2007		2560x1600	Yes	4.0	16	12 GTexel/s
3	None	80nm	16	\n07-Dec-2007		2560x1600	Yes	4.0	16	10 GTexel/s
4	None	65nm	4	\n01-Jul-2007		2560x1600	Yes	4.0	8	7 GTexel/s
5	None	65nm	4	\n28-Jun-2007		2560x1600	Yes	4.0	8	6 GTexel/s
6	None	65nm	4	\n26-Jun-2007		2560x1600	Yes	4.0	8	6 GTexel/s
7	2x 6-pin	55nm	16	\n13-Jul-2009		2560x1600	Yes	4.1	40	35 GTexel/s
8	None	80nm	12	\n06-Nov-2007		2560x1600	Yes	4.0	12	7 GTexel/s
9	None	28nm	32	\n18-Jan-2014		4096x2160	Yes	5.0	80	62 GTexel/s
10	None			\n02-Jan-2001		1600x1200	No	1.0	NA	

	VGA_Connection
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	NA

Hình 4: 10 dòng đầu tiên của dữ liệu

3.3 Làm sạch dữ liệu

Dựa trên nguồn sau có tính tham khảo tốt và phù hợp với nhu cầu sử dụng của đề tài: Card màn hình (VGA) và các thông số quan trọng thường gặp. Nhóm đã thống nhất bằng cách tạo một bảng dữ liệu mới gồm những dữ liệu cần sử dụng là: Max_Power, Boost_Clock, Core_Speed, Memory, Memory_Bandwidth, Memory_Bus, Memory_Speed, Release_Price, Texture_Rate

R Code

```
all_gpus_news <- all_gpus[, c("Max_Power","Boost_
    Clock", "Core_Speed", "Memory", "Memory_Bandwidth"
    , "Memory_Bus", "Memory_Speed", "Release_Price", "
    Texture_Rate")]
head(all_gpus_news, n=10)
```

	Max_Power	Boost_Clock	Core_Speed	Memory	Memory_Bandwidth	Memory_Bus	Memory_Speed	Release_Price	Texture_Rate
1	141 Watts		738 MHz	1024 MB	64GB/sec	256 Bit	1000 MHz		47 GTexel/s
2	215 Watts		\n-	512 MB	106GB/sec	512 Bit	828 MHz		12 GTexel/s
3	200 Watts		\n-	512 MB	51.2GB/sec	256 Bit	800 MHz		10 GTexel/s
4			\n-	256 MB	36.8GB/sec	128 Bit	1150 MHz		7 GTexel/s
5	45 Watts		\n-	256 MB	22.4GB/sec	128 Bit	700 MHz		6 GTexel/s
6	50 Watts		\n-	256 MB	35.2GB/sec	128 Bit	1100 MHz		6 GTexel/s
7	190 Watts		870 MHz	2048 MB	134.4GB/sec	256 Bit	1050 MHz		35 GTexel/s
8	150 Watts		\n-	256 MB	51.2GB/sec	256 Bit	800 MHz		7 GTexel/s
9	150 Watts		\n-	2048 MB	160GB/sec	256 Bit	1250 MHz		62 GTexel/s
10	32 Watts		\n-	64 MB	2.9GB/sec	64 Bit	366 MHz		

Hình 5: 10 dòng đầu tiên của dữ liệu sau khi lọc dữ liệu

Lúc này dữ liệu chúng ta vẫn còn đơn vị và các các dữ liệu đang còn ở định dạng string, ta cần xóa đơn vị và chuyển về kiểu number:

R Code

```r
all_gpus_news <- all_gpus_news %>%
mutate(
  Max_Power = as.numeric(gsub("[^0-9.]","", Max_Power)),
  Boost_Clock = as.numeric(gsub("[^0-9.]", "", Boost_
    Clock)),
  Core_Speed = as.numeric(gsub("[^0-9.]","", Core_Speed)
    ),
  Memory = as.numeric(gsub("[^0-9.]","", Memory)),
  Memory_Bandwidth = as.numeric(gsub("[^0-9.]", "",
    Memory_Bandwidth)),
  Memory_Bus = as.numeric(gsub("[^0-9.]", "", Memory_Bus
    )),
  Memory_Speed = as.numeric(gsub("[^0-9.]", "", Memory_
    Speed)),
  Release_Price = as.numeric(gsub("[^0-9.]", "", Release
    _Price)),
  Texture_Rate = as.numeric(gsub("[^0-9.]", "", Texture_
    Rate))
 )
 head(all_gpus_news,n = 10)
```

	Max_Power	Boost_Clock	Core_Speed	Memory	Memory_Bandwidth	Memory_Bus	Memory_Speed	Release_Price	Texture_Rate
1	141	NA	738	1024	64.0	256	1000	NA	47
2	215	NA	NA	512	106.0	512	828	NA	12
3	200	NA	NA	512	51.2	256	800	NA	10
4	NA	NA	NA	256	36.8	128	1150	NA	7
5	45	NA	NA	256	22.4	128	700	NA	6
6	50	NA	NA	256	35.2	128	1100	NA	6
7	190	NA	870	2048	134.4	256	1050	NA	35
8	150	NA	NA	256	51.2	256	800	NA	7
9	150	NA	NA	2048	160.0	256	1250	NA	62
10	32	NA	NA	64	2.9	64	366	NA	NA

Hình 6: 10 dòng đầu tiên của dữ liệu sau khi dữ liệu chuyển thành dạng số

Sau khi hoàn tất dữ liệu ta thống kê số dữ liệu bị khuất của từng biến:

R Code

```
missing_counts <- colSums(is.na(all_gpus_news))
missing_counts
```

```
  Max_Power   Boost_Clock  Core_Speed      Memory Memory_Bandwidth 
        625        1960         936         420         121 
 Memory_Bus Memory_Speed Release_Price Texture_Rate 
         62         105        2850         544 
```

Hình 7: Thống kê dữ liệu bị khuyết ở các cột

Quan sát bảng thống kê ta thấy được như sau:

- 625 dữ liệu thuộc biến Max_Power bị khuyết.
- 1960 dữ liệu thuộc biến Boost_Clock bị khuyết.
- 936 dữ liệu thuộc biến Core_Speed bị khuyết.
- 420 dữ liệu thuộc biến Memory bị khuyết.
- 121 dữ liệu thuộc biến Memory_Bandwidth bị khuyết.
- 62 dữ liệu thuộc biến Memory_Bus bị khuyết.
- 105 dữ liệu thuộc biến Memory_Speed bị khuyết.
- 2850 dữ liệu thuộc biến Release_Price bị khuyết.
- 544 dữ liệu thuộc biến Textur_Rate bị khuyết.

Từ đó nhóm chúng tôi đã đưa ra một số phương pháp xử lí dữ liệu bị khuyết:

- Ngoài biến Release_Price là yếu tố chính để dự đoán số liệu trong đề tài này, tất cả các dữ liệu khuyết của các biến còn lại sẽ được thay thế bằng giá trị trung bình tương ứng với từng biến.
- Vì số lượng dữ liệu khuyết của biến Boost_Clock chiếm hơn 50% tổng số dữ liệu của biến nên nhóm xin đề xuất loại bỏ biến này để không gây ảnh hưởng đến dữ liệu chung.

Qua đó nhóm sẽ lọc lại dữ liệu bằng cách:

- Xóa dữ liệu thuộc biến Boost_Clock

R Code

```
all_gpus_news <- all_gpus_news[, -2]
```

- Thay thế các giá trị NA bằng các giá trị trung bình sau đó lọc và bỏ các dòng chứa dữ liệu bị khiếm khuyết của biến Release_Price.

R Code

```
mean_values <- sapply(all_gpus_news[, -7],
    function(x) mean(x, na.rm = TRUE))
mean_values
all_gpus_news <- all_gpus_news %>%
 mutate(
  Max_Power = ifelse(is.na(Max_Power), mean_values
     ["Max_Power"], Max_Power),
    Core_Speed = ifelse(is.na(Core_Speed), mean_
       values["Core_Speed"], Core_Speed),
    Memory = ifelse(is.na(Memory), mean_values["
       Memory"], Memory),
    Memory_Bandwidth = ifelse(is.na(Memory_
       Bandwidth), mean_values["Memory_Bandwidth"
       ], Memory_Bandwidth),
    Memory_Bus = ifelse(is.na(Memory_Bus), mean_
       values["Memory_Bus"], Memory_Bus),
    Memory_Speed = ifelse(is.na(Memory_Speed),
       mean_values["Memory_Speed"], Memory_Speed),
    Texture_Rate = ifelse(is.na(Texture_Rate),
       mean_values["Texture_Rate"], Texture_Rate)
 )
 head(all_gpus_news, n=10)
 all_gpus_news_clean <- na.omit(all_gpus_news)
 head(all_gpus_news_clean, n=10)
```

	Max_Power	Core_Speed	Memory	Memory_Bandwidth	Memory_Bus	Memory_Speed	Release_Price	Texture_Rate
1	141.0000	738.0000	1024	64.0	256	1000	NA	47.00000
2	215.0000	946.8939	512	106.0	512	828	NA	12.00000
3	200.0000	946.8939	512	51.2	256	800	NA	10.00000
4	125.5987	946.8939	256	36.8	128	1150	NA	7.00000
5	45.0000	946.8939	256	22.4	128	700	NA	6.00000
6	50.0000	946.8939	256	35.2	128	1100	NA	6.00000
7	190.0000	870.0000	2048	134.4	256	1050	NA	35.00000
8	150.0000	946.8939	256	51.2	256	800	NA	7.00000
9	150.0000	946.8939	2048	160.0	256	1250	NA	62.00000
10	32.0000	946.8939	64	2.9	64	366	NA	90.27463

Hình 8: Thống kê dữ liệu bị khuyết ở các cột

- Xóa dữ các dòng bị khuyết của biến Release_Price.

R Code

```
all_gpus_news_clean <- na.omit(all_gpus_news)
head(all_gpus_news_clean)
```

	Max_Power	Core_Speed	Memory	Memory_Bandwidth	Memory_Bus	Memory_Speed	Release_Price	Texture_Rate
43	250.0000	946.8939	2872.769	480.0	384	1250	1199.00	343
46	250.0000	946.8939	12288.000	547.2	96	1425	1199.00	354
48	375.0000	705.0000	12288.000	672.0	384	1750	2999.00	420
50	375.0000	705.0000	12288.000	672.0	384	1750	2999.00	420
52	250.0000	1140.0000	12288.000	336.6	384	1753	1099.00	240
53	125.5987	1127.0000	24576.000	673.2	384	1753	2059.00	467
55	450.0000	1000.0000	24576.000	673.2	384	1753	1998.00	418
56	250.0000	1000.0000	12288.000	336.6	384	1753	999.00	209
57	250.0000	1127.0000	12288.000	336.6	384	1753	1029.99	233
60	300.0000	1000.0000	12288.000	384.0	3072	500	999.00	320

Hình 9: Dữ liệu sau khi lọc bỏ các hàng bị khuyết của biến Release_Price

- Đánh lại số thứ tự các dòng dữ liệu.

R Code

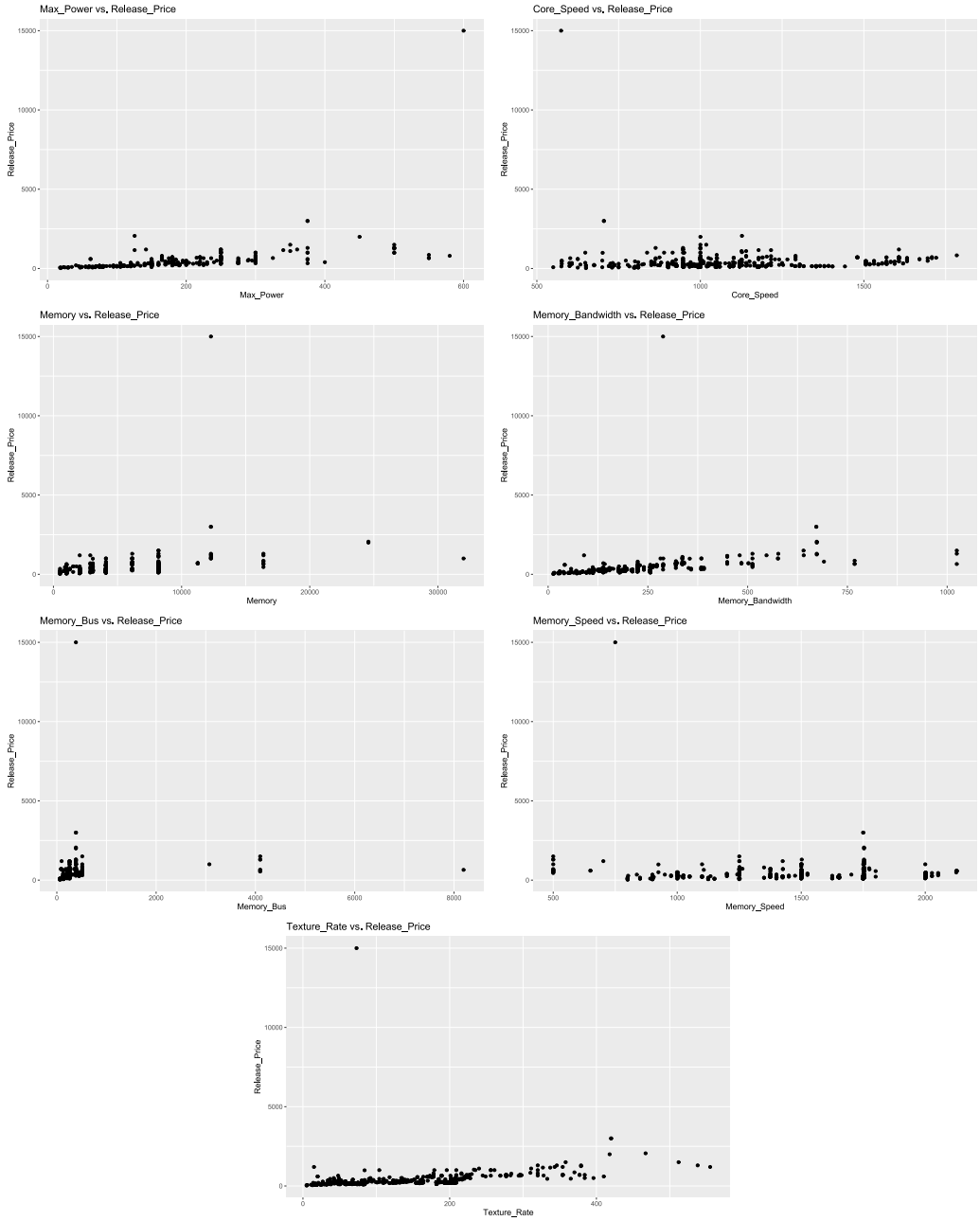
```
rownames(all_gpus_news_clean) <- NULL
head(all_gpus_news_clean)
```

	Max_Power	Core_Speed	Memory	Memory_Bandwidth	Memory_Bus	Memory_Speed	Release_Price	Texture_Rate
1	250.0000	946.8939	2872.769	480.0	384	1250	1199.00	343
2	250.0000	946.8939	12288.000	547.2	96	1425	1199.00	354
3	375.0000	705.0000	12288.000	672.0	384	1750	2999.00	420
4	375.0000	705.0000	12288.000	672.0	384	1750	2999.00	420
5	250.0000	1140.0000	12288.000	336.6	384	1753	1099.00	240
6	125.5987	1127.0000	24576.000	673.2	384	1753	2059.00	467
7	450.0000	1000.0000	24576.000	673.2	384	1753	1998.00	418
8	250.0000	1000.0000	12288.000	336.6	384	1753	999.00	209
9	250.0000	1127.0000	12288.000	336.6	384	1753	1029.99	233
10	300.0000	1000.0000	12288.000	384.0	3072	500	999.00	320

Hình 10: Dữ liệu sau khi đánh lại số thứ tự

THỐNG KÊ TẢ GPU

4.1 Scatter plots

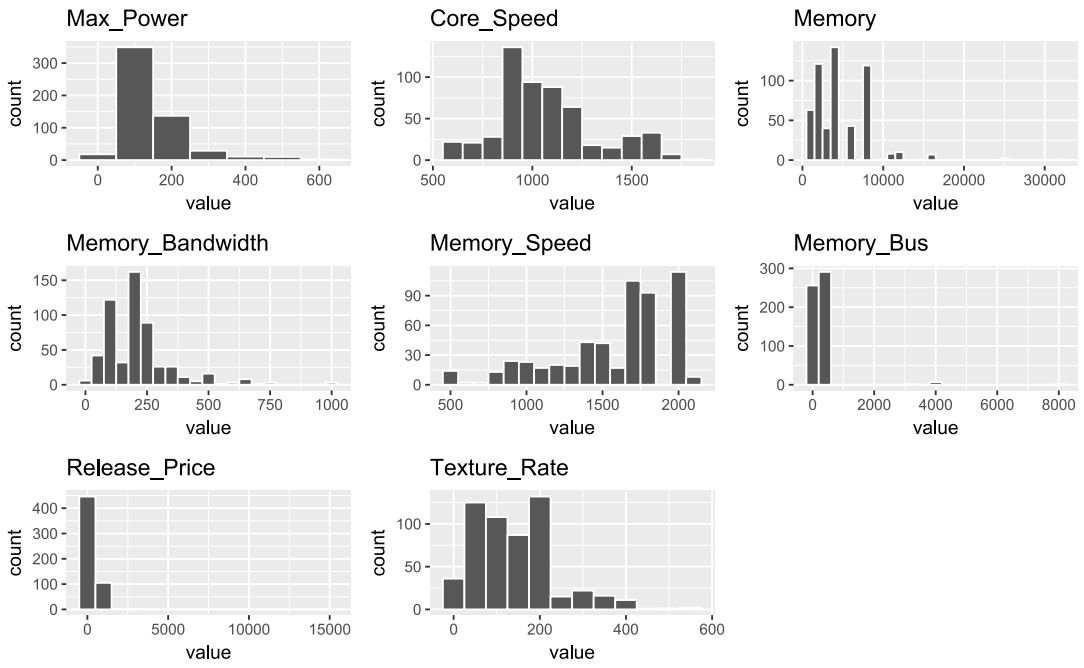


Hình 11: Biểu đồ phân tán của 7 biến đối với biến Release_Price

Nhận xét: Dựa vào các biểu đồ, giá trị "Release_Price" dao động trong khoảng 49$ tới 15$. Tuy nhiên, ta chỉ có thể vẽ đường hồi quy cho các biến "Texture_Rate", "Memory_Bandwidth". Nghĩa là, "Release_Price" có mối quan hệ tuyến tính mạnh với các biến này. Vì thế, ta có thể kết luận rằng mối quan hệ giữa các biến này là mối quan hệ tuyến tính giữa các biến độc lập là thuận và tương đối mạnh khi mà ta có thể vẽ được đường thẳng hồi quy với xu hướng đi lên.

4.2 Histogram

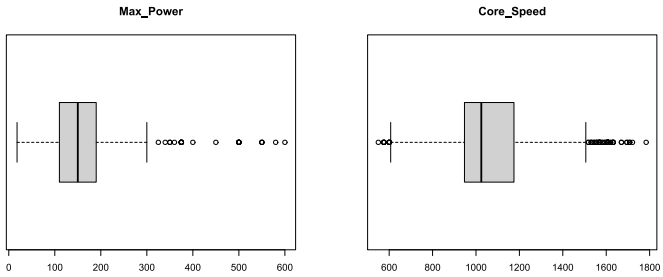
Nhóm đã tiến hành vẽ biểu đồ tần suất (Histogram) nhằm biểu thị mức độ phân phối của bộ dữ liệu được cung cấp.



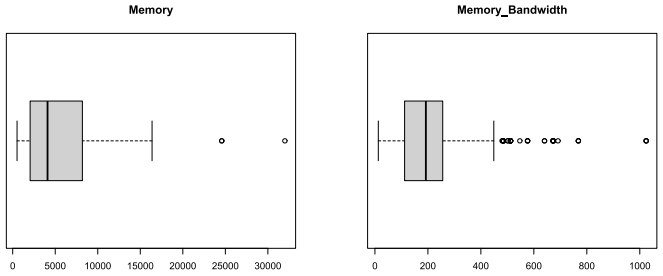
Hình 12: Biểu đồ cột của 8 biến

Nhận xét: Dựa trên các Histogram, ta có thể thấy rằng giá trị của biến "Release_Price" chủ yếu giao động ở mức 0$ đến 500$ và tập trung chủ yếu trong khoảng 50$ đến 250$. Giá trị có số lượng ít nhất là trong khoảng 800$ đến 900$.

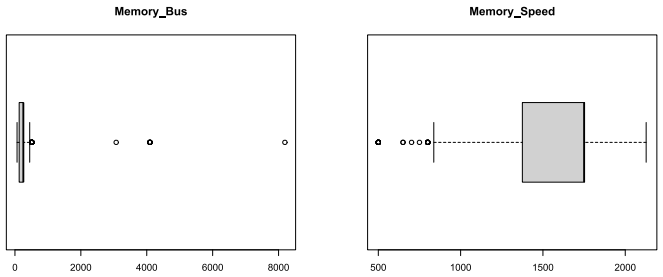
4.3 Boxplot



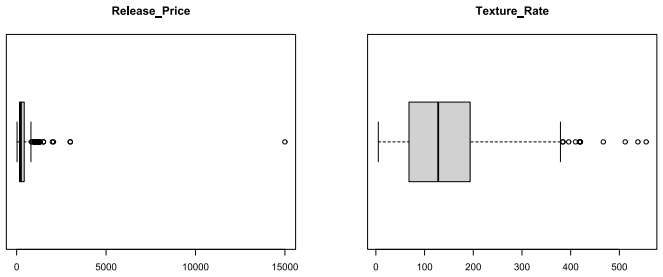
Hình 13: Biểu đồ boxplot của 2 biến Max_Power và Core_Speed.



Hình 14: Biểu đồ boxplot của 2 biến Memory và Memory_Bandwidth.



Hình 15: Biểu đồ boxplot của 2 biến Memory_Bus và Memory_Speed.



Hình 16: Biểu đồ boxplot của 2 biến Release_Price và Textur_Rate.

Nhận xét: Nhìn chung, có nhiều điểm ngoại lai nên khiến ta không thể nhìn rõ điểm phân phối, vậy nên chúng ta cần xử lí và vẽ lại đồ thị".

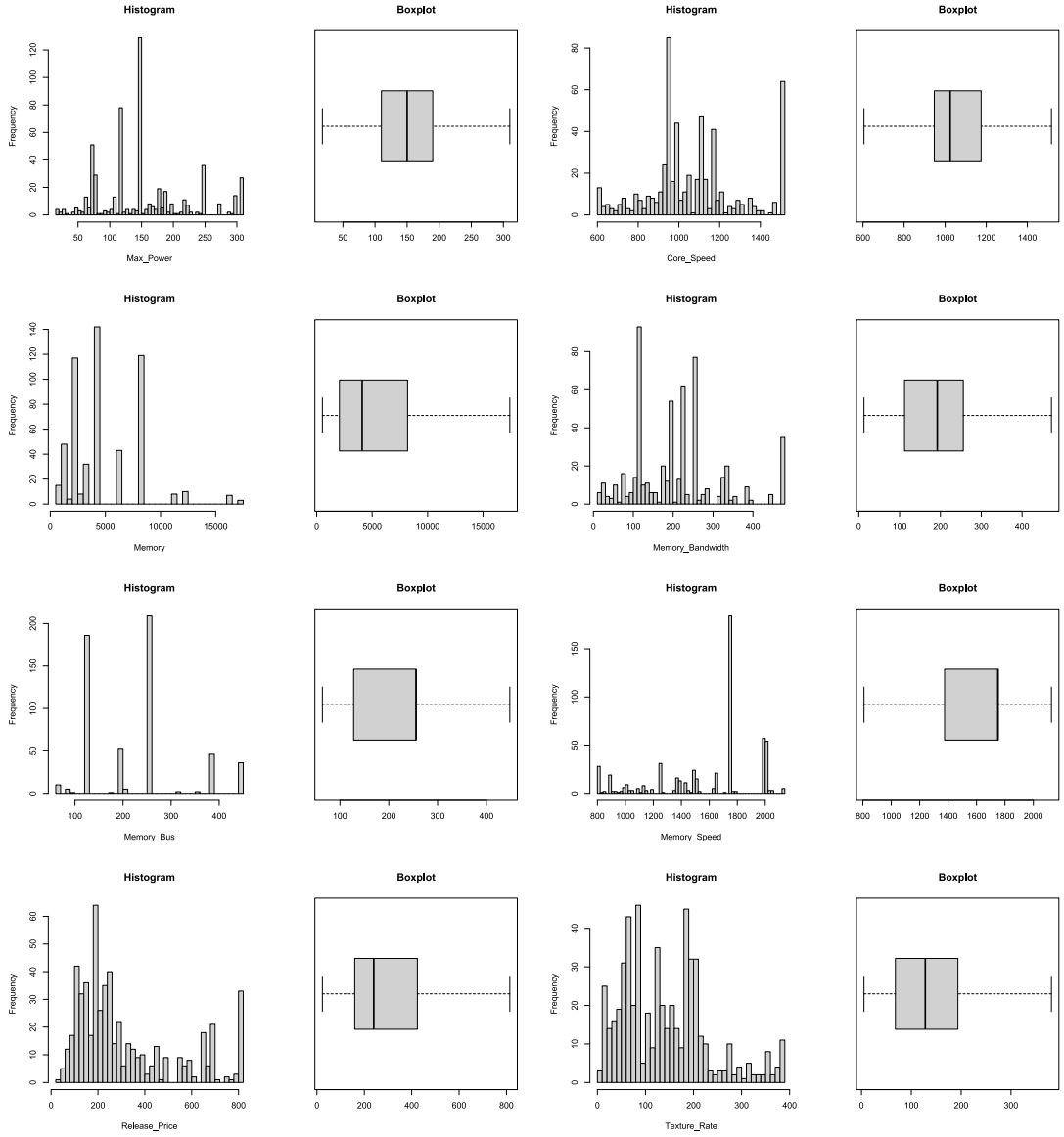
Các điểm ngoại lệ có thể được xử lý bằng cách clip về giá trị cực tiểu và cực đại của Box plot. Tức là khi một giá trị quá lớn hoặc quá nhỏ, ta đưa nó về giá trị lớn nhất/nhỏ nhất được coi là những điểm bình thường.

R Code

```
find_boxplot_boundaries<-function(col,whisker_coeff
   =1.5){
  Q1 <- quantile(col, 0.25)
  Q3 <- quantile(col, 0.75)
  IQR <- Q3 - Q1
  lower <- Q1 - whisker_coeff * IQR
  upper <- Q3 + whisker_coeff * IQR
  return(list(lower = lower, upper = upper))
  }

BoxplotOutlierClipper<-function(whisker_coeff = 1.5,
   X){
  boundaries <- find_boxplot_boundaries(X, whisker_
     coeff)
  clipped_X <- pmax(pmin(X, boundaries$upper),
     boundaries$lower)
  return(clipped_X)
  }
```

Áp dụng lại vào dữ liệu của cột của các biến ta có histogram và boxplot mới như sau:



Hình 17: Biểu đồ histogram và boxplot của 8 biến sau xử lí các giá trị ngoại lai

Nhận xét: Sau khi xử lí dữ liệu theo cực tiểu và cực đại của box plot, ta thấy rằng dữ liệu đỡ bị lệch đi. Box plot cũng cho thấy không còn điểm dữ liệu ngoại lệ nào. Các biểu đồ histogram hầu như không thay đổi quá lớn so với lúc chưa xử lí các giá trị ngoại lai và ta có thể nhìn rõ sự phân bố của các biến, qua đó có thể thấy rằng giá của GPU trải dài và đa dạng. Các biểu đồ box plot hầu như lệch trái cũng thể hiện giá của GPU thường ở mức trung bình, thấp tiếp cận được với đại đa số khách hàng.

4.4 Tính giá trị thống kê mô tả

Tính các giá trị thống kê mô tả gồm: trung bình, trung vị, độ lệch chuẩn, giá trị lớn nhất và giá trị nhỏ nhất, xuất kết quả dưới dạng bảng ta thu được như sau:

R Code

```r
thongkemota <- data.frame(
 cbind(
   apply(all_gpus_news_clean, MARGIN=2, min),
   apply(all_gpus_news_clean, MARGIN=2, sd),
   apply(all_gpus_news_clean, MARGIN=2, mean),
   apply(all_gpus_news_clean, MARGIN=2, median),
   apply(all_gpus_news_clean, MARGIN=2, max)
 )
)

colnames(thongkemota) <- c("GTNN(min)", "Do lech chuan (
    sd)", "Trung binh (mean)", "Trung vi (median)", "GTLN
     (max)")
head(thongkemota)
```

Bảng 2: Bảng giá trị thống kê mô tả các của các biến

	GTNN(min)	Độ lệch chuẩn (sd)	Trung bình (mean)	Trung vị (median)	GTLN(max)
Max_Power	18.0000	70.31991	153.6982	150.0	310.000
Core_Speed	604.7348	229.16461	1072.8082	1024.5	1517.159
Memory	512.0000	3238.46282	4724.6154	4096.0	17408.000
Memory_Bandwidth	12.8000	109.80099	208.1493	192.3	471.850
Memory_Bus	64.0000	98.51261	224.8251	256.0	448.000
Memory_Speed	808.0000	366.05475	1583.0270	1750.0	2127.000

THỐNG KÊ SUY DIỄN GPU

5.1 Xây dựng mô hình hồi quy tuyến tính

Xét mô hình hổi quy gồm:

- Biến phụ thuộc: Release_Price.
- Biến độc lập: Max_Power, Core_Speed, Memory, Memory_Bandwidth, Memory_Bus, Memory_Speed, Texture_Rate.

Ta có mô hình:

$$Release\_Price = \beta_0 + \beta_1 \cdot Max\_Power + \beta_2 \cdot Core\_Speed + \beta_3 \cdot Memory + \beta_4 \cdot Memory\_Bandwidth + \beta_5 \cdot Memory\_Bus + \beta_6 \cdot Memory\_Speed + \beta_7 \cdot Texture\_Rate$$

```
> model1 = lm(Release_Price~Max_Power+Core_Speed+Memory+Memory_Bandwidth+Memory_Bus+Memory_Speed+Texture_Rate, data = all_gpus_news_clean)
> summary(model1)

Call:
lm(formula = Release_Price ~ Max_Power + Core_Speed + Memory +
    Memory_Bandwidth + Memory_Bus + Memory_Speed + Texture_Rate,
    data = all_gpus_news_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-300.00  -65.28   -6.24   40.30  541.34

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     -7.537e+01  3.299e+01  -2.285  0.02272 *
Max_Power        9.547e-01  1.379e-01   6.921 1.26e-11 ***
Core_Speed       1.995e-01  2.513e-02   7.936 1.18e-14 ***
Memory          -5.705e-04  2.659e-03  -0.215  0.83023
Memory_Bandwidth 4.290e-01  1.302e-01   3.294  0.00105 **
Memory_Bus       3.416e-01  7.581e-02   4.505 8.10e-06 ***
Memory_Speed    -1.365e-01  1.565e-02  -8.719  < 2e-16 ***
Texture_Rate     6.083e-01  1.287e-01   4.726 2.91e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 105.9 on 548 degrees of freedom
Multiple R-squared:  0.7603,    Adjusted R-squared:  0.7573
F-statistic: 248.4 on 7 and 548 DF,  p-value: < 2.2e-16
```

Hình 18: Ước lượng hệ số β_i $(0 \leq i \leq 7)$ dựa trên tệp all_gpus_news_clean (model1)

Do $P_r(>|t|)$ của tất cả các biến trừ Memory và hệ số tự do đều bé hơn mức ý nghĩa 0.05, nên ta bác bỏ giả thuyết H_0 của các biến trừ Memory. Tuy nhiên, do chưa đủ căn cứ để khẳng định giả thuyết H_0 đối với Memory, nhóm quyết định xây dựng thêm một mô hình model2 bằng cách loại bớt biến Memory khỏi mô hình model1.

```
> model2 = lm(Release_Price~Max_Power+Core_Speed+Memory_Bandwidth+Memory_Bus+Memory_Speed+Texture_Rate, data = all_gpus_news_clean)
> summary(model2)

Call:
lm(formula = Release_Price ~ Max_Power + Core_Speed + Memory_Bandwidth +
    Memory_Bus + Memory_Speed + Texture_Rate, data = all_gpus_news_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-301.14  -65.64   -6.37   40.33  540.24

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      -74.40664   32.65524  -2.279 0.023077 *
Max_Power          0.95820    0.13689   7.000 7.50e-12 ***
Core_Speed         0.19907    0.02504   7.949 1.07e-14 ***
Memory_Bandwidth   0.42069    0.12429   3.385 0.000763 ***
Memory_Bus         0.34244    0.07564   4.527 7.33e-06 ***
Memory_Speed      -0.13706    0.01538  -8.909  < 2e-16 ***
Texture_Rate       0.59922    0.12137   4.937 1.05e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 105.8 on 549 degrees of freedom
Multiple R-squared:  0.7603,    Adjusted R-squared:  0.7577
F-statistic: 290.3 on 6 and 549 DF,  p-value: < 2.2e-16
```

Hình 19: mô hình model2 loại bớt biến Memory khỏi mô hình model1

Tiến hành so sánh hai mô hình với giả thuyết:

- H_0: Hai mô hình có độ hiệu quả như nhau.
- H_1: Hai mô hình có độ hiệu quả khác nhau.

Phân tích phương sai ANOVA:

```
> anova(model1, model2)
Analysis of Variance Table

Model 1: Release_Price ~ Max_Power + Core_Speed + Memory + Memory_Bandwidth +
    Memory_Bus + Memory_Speed + Texture_Rate
Model 2: Release_Price ~ Max_Power + Core_Speed + Memory_Bandwidth + Memory_Bus +
    Memory_Speed + Texture_Rate
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1    548 6142938
2    549 6143454 -1    -515.8 0.046 0.8302
```

Hình 20: Phương sai ANOVA

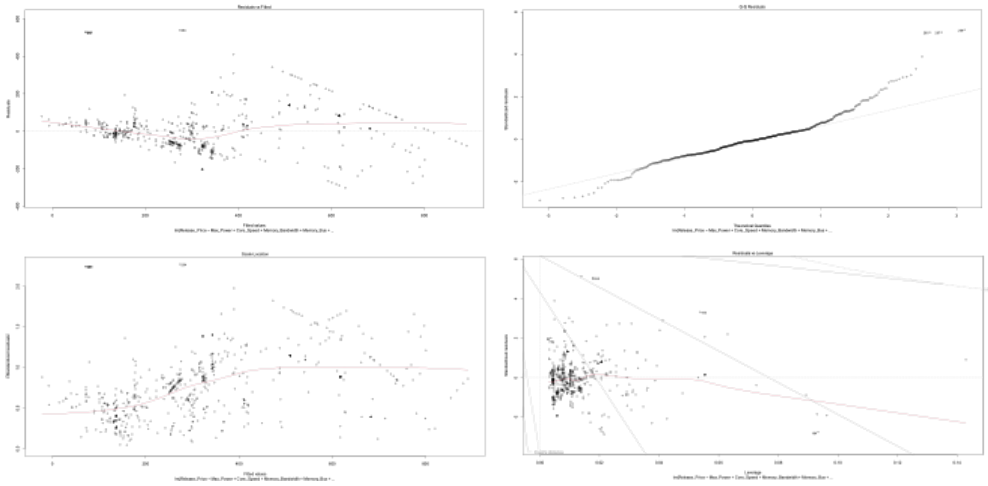
Vì $P_r(> T) = 0.8302 > 0.05$ nên giả thuyết H_0 là phù hợp. Do cả hai mô hình đều có độ hiệu quả như nhau, nên việc loại bỏ biến Memory không gây ảnh hưởng tới mô hình ta đã xây dựng, vì vậy nhóm quyết định chọn mô hình model2 là mô hình phù hợp hơn.

Như vậy đường thẳng hồi quy ước lượng được cho bởi phương trình:

$$\begin{aligned} Release\_Price = &-74.40664 + 0.95820 \cdot Max\_Power + 0.19907 \cdot Core\_Speed - \\ &0.42069 \cdot Memory\_Bandwidth + 0.34244 \cdot Memory\_Bus - \\ &0.13706 \cdot Memory\_Speed + 0.59922 \cdot Texture\_Rate \end{aligned}$$

Ngoài ra ta cũng thấy được hệ số R^2 hiệu chỉnh là 0.7577 cho thấy đến 75.77% sự biến thiên trong giá máy tính được giải thích bằng các biến độc lập, 24.23% còn lại có thể do sai số hoặc các biến độc lập khác chưa được đưa vào mô hình.

Cuối cùng, ta đi kiểm tra tính phù hợp của mô hình vừa thu được thông qua các đồ thị:



Nhận xét:

- Đồ thị thứ 1 (Residuals vs Fitted) vẽ các giá trị sai số so với dự báo, dùng để kiểm tra giả thiết tuyến tính của dữ liệu và giả thiết sai số có kỳ vọng bằng 0. Đường màu đỏ gần với đường thẳng $y = 0$, tuy có sai lệch nhưng vẫn trong khoảng chấp nhận được, chứng tỏ bộ dữ liệu có tính tuyến tính, và giả thiết sai số có kỳ vọng bằng 0 thỏa mãn.
- Đồ thị thứ 2 (Q-Q Residuals) vẽ sai số được chuẩn hóa, dùng để kiểm tra giả thiết sai số có phân phối chuẩn. Các điểm trên đồ thị được phân bố theo đường thẳng màu đỏ, chỉ sai lệch ở hai đầu nên giả thiết sai số có phân phối chuẩn thỏa mãn.
- Đồ thị thứ 3 (Scale – Location) vẽ căn bậc hai của sai số được chuẩn hóa dùng để kiểm tra giả định phương sai của sai số là hằng số. Các điểm phân tán một cách ngẫu nhiên, không tập trung nhiều quanh đường màu đỏ nên giả định về phương sai của sai số là hằng số được thỏa mãn.
- Đồ thị thứ 4 (Residuals vs Leverage) dùng để xác định các điểm có ảnh hưởng cao. Các điểm đó là 184, 265 và 183.

5.2 Dự báo cho giá GPU

- Dựa trên mô hình hồi quy đã xây dựng, thử dự báo giá một GPU có: $Max\_Power = 141W, Core\_Speed = 738MHz, Memory = 1024MB, Memory\_Bandwidth = 64GB/sec, Memory\_Bus = 256bit, Memory\_Speed = 1000MHz, Texture\_Rate = 47Gtexel/s$.

```
> X <- data.frame("Max_Power" = 141, "Core_Speed" = 738, "Memory_Bandwidth" = 64, "Memory_Bus" = 256, "Memory_Speed" = 1000, "Texture_Rate" = 47)
> head(X)
  Max_Power Core_Speed Memory_Bandwidth Memory_Bus Memory_Speed Texture_Rate
1       141         738               64        256         1000           47
> predictX <- predict(model2, X, interval = "confidence")
> head(predictX)
       fit      lwr      upr
1 213.3062 190.1539 236.4586
```

- Giá dự báo 213.3062 với khoảng tin cậy so với giá trị dự báo $(190.1539, 236.4586)$.

THẢO LUẬN VÀ MỞ RỘNG

6.1 Ưu điểm

Hồi quy tuyến tính bội giúp phân tích đồng thời tác động của nhiều biến độc lập lên một biến phụ thuộc. Điều này giúp chúng ta xác định được tầm quan trọng tương đối của từng biến độc lập trong việc dự đoán biến phụ thuộc.

Bằng cách sử dụng 6 yếu tố chính tác động đến biến phụ thuộc "Release_Price", chúng ta đã có thể dự đoán giá bán ra của các GPU.

Phân tích phương sai ANOVA cho phép kiểm định sự khác biệt giữa các nhóm của biến phụ thuộc, giúp hiểu rõ hơn về ảnh hưởng của các yếu tố. Đồng thời, phương pháp này giúp phân tích sự biến thiên của dữ liệu và xác định xem có yếu tố nào ảnh hưởng đáng kể đến giá bán GPU hay không.

6.2 Hạn chế

Chỉ có thể sử dụng cho kiểu dữ liệu tuyến tính.

Để mô hình hoạt động hiệu quả, cần phải loại bỏ các điểm ngoại lai là những giá trị quan trắc có sự khác biệt đáng kể so với các giá trị quan trắc khác trong bộ dữ liệu vì chúng có thể làm sai lệch mối quan hệ giữa biến dự đoán và biến phụ thuộc và dẫn đến dự đoán không chính xác.

Phân tích phương sai ANOVA chỉ mô tả sự khác biệt giữa các nhóm mà không xác định được mối quan hệ nhân quả giữa các biến.

6.3 Mở rộng

Nguồn dữ liệu cần được thu thập và mở rộng để có nguồn dữ liệu toàn diện hơn, nhiều đặc điểm nghiên cứu hơn. Từ đó, đưa ra dự đoán một cách khách quan hơn, tăng tỉ lệ chính xác cho dự đoán.

Bên cạnh đó, để mang lại hiệu quả dự đoán tốt nhất thì các phương pháp dự đoán cũng cần sử dụng một cách linh hoạt. Chúng ta có thể sự dụng một số phương pháp khác như: Mô hình hồi quy tuyến tính, Mô hình hồi quy phi tuyến, Mô hình hồi quy lựa chọn, Mô hình hồi quy đa cấp,...

Tuy nhiên, hạn chế về mặt thời gian và kiến thức nên chúng em chỉ sử dụng Mô hình hồi quy bội để dự đoán kết quả.

NGUỒN DỮ LIỆU VÀ NGUỒN CODE

7.1 Nguồn dữ liệu

https://www.kaggle.com/datasets/iliassekkaf/computerparts

7.2 Nguồn code

https://drive.google.com/drive/folders/1O10XfZgub6dYam8kqlon-0puwhi4C7Ef?usp=sharing

TÀI LIỆU THAM KHẢO

[1] Nguyễn Kiều Dung, *Bài giảng Xác suất Thống Kê*.

[2] Nguyễn Tiến Dũng (chủ biên), Nguyễn Đình Huy, 2019, *Xác suất - Thống kê & Phân tích số liệu*.

[3] Hoàng Văn Hà, *Bài giảng Xác suất Thống Kê*.

[4] Lê Hồng Thái, *Phương pháp tính toán và đánh giá độ tin cậy trong hệ thống điện*.

[5] Nguyễn Đình Thắng, *Giáo trình Vật liệu điện*.

[6] George C.Runner.Hoboken, Douglas C.Montgomery, 2007, *Applied Statistic and Probability for Engineers*, NJ: Wiley.

[7] David M. Rocke, 2020, Geoffrey J. Thompson, *Applied Statistics with R*, NY: Springer.