

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN - ĐIỆN TỬ



BÁO CÁO BÀI TẬP LỚN
MÔN HỌC XÁC SUẤT & THỐNG KÊ (MT2013)
ĐỀ TÀI

"DỰ BÁO BĂNG THÔNG BỘ NHỚ GPUS
BẰNG MÔ HÌNH HỒI QUY TUYẾN TÍNH BỘI"

LỚP: N1HT - NHÓM: 10

Danh sách thành viên:

STT	Họ và tên	MSSV
1	Trần Ngọc Lan Anh	2348002
2	Nguyễn Thị Thanh Duyên	2348007
3	Mai Thanh Nhã	2348036
4	Trương Tường Như	2348037

Giảng viên hướng dẫn: ThS. Phan Thị Khánh Vân

Tp. Hồ Chí Minh, ngày 01 tháng 01 năm 2025

BẢNG PHÂN CÔNG VÀ ĐÁNH GIÁ CÁC THÀNH VIÊN NHÓM

Họ và tên	Nhiệm vụ	Dánh giá
Trần Ngọc Lan Anh	Tổng quan dữ liệu, kiến thức nền code R và nhận xét phần kết quả tiền xử lý số liệu	100%
Nguyễn Thị Thanh Duyên	Tổng hợp code và viết báo cáo, thảo luận mở rộng	100%
Mai Thanh Nhã	Code R và nhận xét phần thống kê mô tả	100%
Trương Tường Như	Code R và nhận xét phần thống kê suy diễn	100%

Mục lục

1 TỔNG QUAN DỮ LIỆU	4
2 KIẾN THỨC NỀN	5
2.1 Giả định cơ bản	5
2.2 Ước lượng các hệ số	5
2.3 Đánh giá mô hình	6
3 TIỀN XỬ LÝ SỐ LIỆU	7
3.1 Đọc dữ liệu	7
3.2 Kiểm tra dữ liệu khuyết	7
3.3 Làm sạch các biến và xử lý định dạng các biến	9
3.4 Chọn lọc các biến phân tích	9
4 THỐNG KÊ MÔ TẢ	10
4.1 Tính toán các thống kê mô tả	10
4.2 Trực quan hoá dữ liệu bằng các đồ thị	10
5 THỐNG KÊ SUY DIỄN	15
6 THẢO LUẬN VÀ MỞ RỘNG	20
7 NGUỒN DỮ LIỆU VÀ NGUỒN CODE	21
8 TÀI LIỆU THAM KHẢO	24

Danh sách hình vẽ

3.1	Kết quả 6 dòng đầu tiên của dữ liệu	7
3.2	Kết quả kiểm tra dữ liệu khuyết của tệp tin GPU_data	8
3.3	Kết quả kiểm tra cấu trúc tệp tin new_GPU_data sau khi xử lý dữ liệu khuyết	9
4.4	Kết quả thống kê mô tả cho các biến liên tục	10
4.5	Kết quả thống kê số lượng cho các biến phân loại	10
4.6	Kết quả đồ thị histogram của biến Memory_Bandwidth	11
4.7	Kết quả đồ thị histogram của biến Memory_Bandwidth	11
4.8	Kết quả đồ thị boxplot của log(Memory Bandwidth) theo Dedicated	12
4.9	Kết quả đồ thị boxplot của log(Memory Bandwidth) theo Shader	12
4.10	Kết quả đồ thị phân tán của log(Memory Bandwidth) theo log(Memory_Speed), log(L2_Cache), log(Memory_Bus).	14
4.11	Kết quả đồ thị ma trận tương quan	14
5.12	Kết quả mô hình hồi quy tuyến tính bội	15
5.13	Kết quả đồ thị Residuals and Fitted	17
5.14	Kết quả đồ thị Normal Q-Q	17
5.15	Kết quả đồ thị Scale-Location	18
5.16	Kết quả đồ thị Residuals và Leverage	18
5.17	Kết quả chỉ số VIF	19
5.18	Kết quả dự báo	19
5.19	Kết quả so sánh giữa thực tế và dự báo	19

1 TỔNG QUAN DỮ LIỆU

Trong bối cảnh công nghệ hiện đại, GPU đã trở thành phần quan trọng trong nhiều lĩnh vực, từ chơi game đến trí tuệ nhân tạo. GPU (Graphics processing unit – Đơn vị xử lý đồ họa) là vi xử lý chuyên dụng để xử lý các tác vụ liên quan đến đồ họa và tính toán song song. Việc phân tích các thông số kỹ thuật của GPU trở nên quan trọng, đặc biệt là đối với người tiêu dùng và nhà nghiên cứu. Các yếu tố như bộ nhớ và tốc độ xung nhịp ảnh hưởng đến hiệu năng của GPU. Nghiên cứu này sẽ kết hợp phương pháp phân tích thống kê để hiểu rõ hơn về các thông số kỹ thuật của GPU.

Bộ dữ liệu được sử dụng để phân tích trong đề tài này từ tập tin "All_GPUs.csv". Những dữ liệu này được thu thập dựa trên các GPU từ một số công ty sản xuất với mục đích so sánh các hiệu năng giữa các GPU với nhau.

Dưới đây là một số thông tin của bộ dữ liệu:

- Tác giả: ilissek
- Tổng thể: GPUs từ Intel, Game-Debate và các công ty sản xuất linh kiện
- Kích cỡ mẫu: 3406
- Số biến: 34

Dữ liệu gốc được cung cấp tại: <https://www.kaggle.com/datasets/iliassekkaf/computerparts>

Dưới đây là một số biến đã được chọn lọc phân tích (sau bước tiền xử lý) trong bộ dữ liệu:

STT	Tên biến	Loại biến	Đơn vị	Mô tả
1	Memory_Bandwidth	liên tục	GB/s	Băng thông bộ nhớ
2	Memory_Speed	liên tục	MHz	Tốc độ của bộ nhớ
3	L2_Cache	liên tục	KB	Kích thước bộ nhớ đệm cấp 2 của GPU
4	Memory_Bus	liên tục	Bit	Số bit có thể truy cập vào bộ nhớ cùng lúc
5	Dedicated	phân loại	-	Chỉ GPU rời hoặc GPU tích hợp (No: GPU tích hợp / Yes: GPU rời)
6	Shader	phân loại	-	Chỉ phiên bản Shader (trình đồ họa, tính toán các mức độ sáng tối với các phiên bản 1, 1.4, 2, 3, 4, 4.1 và 5)

2 KIẾN THỨC NỀN

Mô hình hồi quy bội (Multiple Regression Model) là một công cụ mạnh mẽ trong thống kê và phân tích dữ liệu, giúp nghiên cứu mối quan hệ giữa một biến phụ thuộc và nhiều biến độc lập.

Mô hình này mở rộng từ hồi quy tuyến tính đơn giản, trong đó chỉ có một biến độc lập.

Biến phụ thuộc (Dependent Variable): Là biến mà bạn muốn dự đoán hoặc giải thích.

Biến độc lập (Independent Variables): Là các biến được sử dụng để dự đoán hoặc giải thích biến phụ thuộc.

Công thức của mô hình hồi quy bội là:

$$Y = \beta_0 + \beta_1.X_1 + \beta_2.X_2 + \dots + \beta_k.X_k + \varepsilon$$

Trong đó:

- + Y: biến phụ thuộc
- + X_1, X_2, \dots, X_k : các biến độc lập
- + β_0 : hệ số chặn.
- + $\beta_1, \beta_2, \dots, \beta_k$: hệ số hồi quy của các biến độc lập.

2.1 Giả định cơ bản

Để mô hình hồi quy bội có thể đưa ra các kết quả đáng tin cậy, một số giả định cơ bản cần được thỏa mãn:

- Tuyến tính: Mọi quan hệ giữa biến phụ thuộc và các biến độc lập là tuyến tính. Điều này có thể được kiểm tra bằng cách vẽ đồ thị phân phối của biến phụ thuộc so với các biến độc lập và kiểm tra các phần dư (residuals).
- Độc lập: Các quan sát phải độc lập với nhau. Vi phạm giả thuyết này có thể dẫn đến các kết quả không chính xác.
- Homoscedasticity: Sai số ngẫu nhiên có phương sai đồng nhất trên toàn bộ phạm vi của các biến độc lập. Điều này có nghĩa là phương sai của sai số không thay đổi khi giá trị của biến độc lập thay đổi. Vi phạm giả thuyết này có thể dẫn đến các vấn đề trong phân tích hồi quy.
- Phân phối chuẩn của sai số: Sai số ngẫu nhiên được phân phối chuẩn. Điều này cần thiết cho các kiểm định thống kê về các hệ số hồi quy.
- Không có đa cộng tuyến (Multicollinearity): Các biến độc lập không có mối quan hệ tuyến tính mạnh với nhau. Đa cộng tuyến có thể làm cho việc ước lượng các hệ số hồi quy trở nên không chính xác và khó giải thích.
- Không có tự tương quan: Sai số ngẫu nhiên không có mối liên hệ với nhau (không có tự tương quan). Vi phạm giả thuyết này có thể xảy ra trong các dữ liệu chuỗi thời gian.

2.2 Ước lượng các hệ số

Các hệ số hồi quy β được ước lượng bằng cách sử dụng phương pháp bình phương tối thiểu (Ordinary Least Squares - OLS), nhằm giảm thiểu tổng bình phương của sai số giữa giá trị quan sát và giá trị dự đoán. Phương pháp này giúp tìm ra các hệ số β sao cho tổng của bình phương các phần dư (residuals) là nhỏ nhất.

2.3 Đánh giá mô hình

Sau khi xây dựng mô hình, việc đánh giá tính phù hợp của mô hình và ý nghĩa của các hệ số hồi quy là rất quan trọng. Một số chỉ số thường được sử dụng bao gồm:

R^2 : Tỷ lệ phương sai của biến phụ thuộc được giải thích bởi mô hình. Giá trị của R^2 nằm trong khoảng từ 0 đến 1, với giá trị càng gần 1 cho thấy mô hình giải thích tốt hơn.

$P - value$: Kiểm tra tính đáng kể của các hệ số hồi quy. Một p-value nhỏ (thường nhỏ hơn 0.05) cho thấy hệ số hồi quy có ý nghĩa thống kê.

Kiểm tra phần dư: Kiểm tra các giả thuyết về sai số để đảm bảo rằng các giả thuyết của mô hình hồi quy không bị vi phạm.

3 TIỀN XỬ LÝ SỐ LIỆU

3.1 Đọc dữ liệu

Đọc file dữ liệu tên là "GPU_data" vào R bằng câu lệnh read.csv và dùng lệnh head để in kết quả 6 dòng đầu tiên của dữ liệu.

```
1 GPU_data <- read.csv("~/Desktop/All_GPUs.csv")
```

	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
	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
	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		
	Power_Connector	Process	ROPs	Release_Date	Release_Price	Resolution_WxH	SLI_Crossfire	Shader	TMUs
1	None	55nm	16	\n01-Mar-2009		2560x1600		Yes	4 64
2	None	80nm	16	\n14-May-2007		2560x1600		Yes	4 16
3	None	80nm	16	\n07-Dec-2007		2560x1600		Yes	4 16
4	None	65nm	4	\n01-Jul-2007		2560x1600		Yes	4 8
5	None	65nm	4	\n28-Jun-2007		2560x1600		Yes	4 8
6	None	65nm	4	\n26-Jun-2007		2560x1600		Yes	4 8
	Texture_Rate	VGA_Connection							
1	47 GPixel/s	0							
2	12 GPixel/s	0							
3	10 GPixel/s	0							
4	7 GPixel/s	0							
5	6 GPixel/s	0							
6	6 GPixel/s	0							

Hình 3.1 Kết quả 6 dòng đầu tiên của dữ liệu

3.2 Kiểm tra dữ liệu khuyết

Tìm các dòng trống (hàng có toàn bộ các giá trị là rỗng hoặc "") trong dữ liệu và chuyển chúng thành giá trị NA:

```
1 GPU_data[GPU_data == ""] <- NA
```

Tiếp theo, ta tìm các giá trị trong dataframe có định dạng \n (các giá trị bắt đầu bằng ký tự xuống dòng \n) hoặc \n- (các giá trị bắt đầu bằng ký tự xuống dòng \n theo sau là dấu -) và chuyển chúng thành NA.

```
1 GPU_data[] <- lapply(GPU_data, function(x) gsub("^\\n$", NA, x))
2 GPU_data[] <- lapply(GPU_data, function(x) gsub("^\\n- $", NA, x))
```

Thực hiện in bảng thống kê số lượng và tỷ lệ dữ liệu khuyết trong tệp tin bằng lệnh freq.na trong thư viện questionr. Ta thu được kết quả như hình 3.2.

```
1 library(questionr)
2 freq.na(GPU_data)
```

	missing	%
Release_Price	2850	84
DisplayPort_Connection	2549	75
Boost_Clock	1960	58
PSU	1175	34
Core_Speed	936	27
HDMI_Connection	763	22
VGA_Connection	758	22
DVI_Connection	750	22
Power_Connector	709	21
Best_Resolution	642	19
Max_Power	625	18
Pixel_Rate	544	16
Texture_Rate	544	16
ROPs	538	16
TMUs	538	16
Process	463	14
Memory	420	12
Resolution_WxH	195	6
Memory_Bandwidth	121	4
Shader	107	3
Memory_Speed	105	3
Architecture	62	2
Memory_Bus	62	2
Memory_Type	56	2
Open_GL	40	1
Dedicated	14	0
Integrated	14	0
Direct_X	6	0
L2_Cache	0	0
Manufacturer	0	0
Name	0	0
Notebook_GPU	0	0
Release_Date	0	0
SLI_Crossfire	0	0

Hình 3.2 Kết quả kiểm tra dữ liệu khuyết của tệp tin GPU_data

Dựa trên kết quả thu được, ta thấy có nhiều biến chứa dữ liệu khuyết (ví dụ biến Release_Price chứa 2850 NA và chiếm tỷ lệ khoảng 84% dữ liệu, biến DisplayPort_Connection chứa 2549 NA và chiếm tỷ lệ khoảng 75% dữ liệu, biến Boost_Clock chứa 1960 NA, chiếm tỷ lệ khoảng 58% dữ liệu,...).

Ta nhận thấy tỷ lệ dữ liệu khuyết ở các biến có sự khác biệt và chênh lệch nhau lớn. Do đó, ta thực hiện lọc bỏ các biến có tỷ lệ dữ liệu khuyết cao ($>=10$) và chỉ giữ lại các biến có dữ liệu khuyết dưới 10%. Ngoài ra, đối với các biến có dữ liệu khuyết dưới 10%, ta sẽ thực hiện xoá các quan sát chứa dữ liệu khuyết

```

1 NA_summary <- data.frame(freq.na(GPU_data))
2 colnames(NA_summary) <- c("Missing", "Percent")
3 selected_columns <- rownames(NA_summary[NA_summary$Percent < 10, ])
4 new_GPU_data <- GPU_data[, selected_columns]
5 new_GPU_data<-na.omit(new_GPU_data)

```

Sau khi thực hiện xử lý dữ liệu khuyết, ta kiểm tra lại cấu trúc tệp tin lại bằng câu lệnh str.

```
1 str(new_GPU_data)
```

```
'data.frame': 3036 obs. of 17 variables:
 $ Resolution_WxH : chr "2560x1600" "2560x1600" "2560x1600" "2560x1600" ...
 $ Memory_Bandwidth: chr "64GB/sec" "106GB/sec" "51.2GB/sec" "36.8GB/sec" ...
 $ Shader           : chr "4" "4" "4" "4" ...
 $ Memory_Speed     : chr "1000 MHz" "828 MHz" "800 MHz" "1150 MHz" ...
 $ Architecture    : chr "Tesla G92b" "R600 XT" "R600 PRO" "RV630" ...
 $ Memory_Bus       : chr "256 Bit" "512 Bit" "256 Bit" "128 Bit" ...
 $ Memory_Type      : chr "GDDR3" "GDDR3" "GDDR3" "GDDR4" ...
 $ Open_GL          : chr "3.3" "3.1" "3.1" "3.3" ...
 $ Dedicated        : chr "Yes" "Yes" "Yes" "Yes" ...
 $ Integrated       : chr "No" "No" "No" "No" ...
 $ Direct_X         : chr "DX 10.0" "DX 10" "DX 10" "DX 10" ...
 $ L2_Cache         : chr "0KB" "0KB" "0KB" "0KB" ...
 $ Manufacturer     : chr "Nvidia" "AMD" "AMD" "AMD" ...
 $ Name              : chr "GeForce GTS 150" "Radeon HD 2900 XT 512MB" "Radeon HD 2900 Pro" "Radeon HD 2600 XT Diamond Edition" ...
 $ Notebook_GPU      : chr "No" "No" "No" "No" ...
 $ Release_Date     : chr "\n01-Mar-2009" "\n14-May-2007" "\n07-Dec-2007" "\n01-Jul-2007" ...
 $ SLI_Crossfire    : chr "Yes" "Yes" "Yes" "Yes" ...
```

Hình 3.3 Kết quả kiểm tra cấu trúc tệp tin new_GPU_data sau khi xử lý dữ liệu khuyết

Từ kết quả hình 3.3, tệp tin còn lại 3036 quan sát với 17 biến, tức là đã loại bỏ 17 biến và 370 quan sát (chiếm tỷ lệ khoảng 10.86%).

3.3 Làm sạch các biến và xử lý định dạng các biến

Trong bộ dữ liệu có một vài biến "L2_Cache", "Memory_Bandwidth", "Memory_Bus", "Memory_Speed" có chứa đơn vị, ta thực hiện loại bỏ các đơn vị của các biến này.

```
1 columns_to_clean <- c("L2_Cache", "Memory_Bandwidth", "Memory_Bus", "Memory_Speed")
2
3 remove_units <- function(column) {
4   cleaned_column <- gsub("[^0-9.]", "", column)
5   cleaned_column <- as.numeric(cleaned_column)
6   return(cleaned_column)
7 }
8
9 new_GPU_data[columns_to_clean] <- lapply(new_GPU_data[columns_to_clean], remove_units)
```

3.4 Chọn lọc các biến phân tích

Ta tiếp tục thực hiện chọn lọc các biến quan trọng để thực hiện dự báo băng thông bộ nhớ (Memory_Bandwidth) của GPUs. Và mô hình Hồi quy là lựa chọn phù hợp và đơn giản nhất cho việc phân tích những nhân tố ảnh hưởng đến Memory_Bandwidth.

Khi xây dựng mô hình hồi quy với Memory_Bandwidth là biến phụ thuộc, ta xem xét các biến độc lập (sau khi đã làm sạch dữ liệu) như: Memory_Speed, L2_Cache, Architecture, Dedicated, Memory_Bus, Shader, Direct_X và Resolution_WxH .

Tuy nhiên vì các biến Architecture, Direct_X, Resolution_WxH là các biến dạng chuỗi và nhận nhiều giá trị, và việc đưa các biến này vào mô hình hồi quy sẽ rất khó khăn cho việc phân tích nên do đó, ta sẽ lựa chọn các biến độc lập Memory_Speed, L2_Cache, Dedicated, Memory_Bus, Shader.

Ta tạo bộ dữ liệu bao gồm các biến trong nội dung phân tích như sau:

```
1 main_data <- new_GPU_data[c("Memory_Bandwidth", "Memory_Speed", "L2_Cache", "Memory_Bus",
2                               "Shader", "Dedicated")]
2 head(main_data)
```

4 THỐNG KÊ MÔ TẢ

4.1 Tính toán các thống kê mô tả

Thực hiện tính toán các thống kê mô tả bao gồm cỡ mẫu, trung bình mẫu, độ lệch chuẩn, các phân vị, giá trị nhỏ nhất, giá trị lớn nhất cho các biến liên tục.

```

1 numeric_vars <- sapply(main_data, is.numeric)
2 numeric_data <- main_data[, numeric_vars]
3
4 summary_stats <- sapply(numeric_data, function(x) {
5   c(
6     Mean = mean(x),
7     SD = sd(x),
8     Min = min(x),
9     Q1 = quantile(x, 0.25),
10    Median = median(x),
11    Q3 = quantile(x, 0.75),
12    Max = max(x)
13  )
14 })
15
16 t(as.data.frame(summary_stats))

```

	Mean	SD	Min	Q1.25%	Median	Q3.75%	Max
Memory_Bandwidth	144.7002	135.4090	2	32	112.1	223.85	1280
Memory_Speed	1211.1077	420.2349	200	900	1247.5	1502.00	2127
L2_Cache	1335.0817	3209.3938	0	128	512.0	1024.00	30722
Memory_Bus	210.6746	221.9971	32	128	128.0	256.00	4096

Hình 4.4 Kết quả thống kê mô tả cho các biến liên tục

Đối với các biến phân loại, ta lập bảng thống kê số lượng.

```

1 table(main_data$Dedicated)
2 table(main_data$Shader)

```

No	Yes
214	2822
1	1.4
1	7
39	82
194	126
2587	

Hình 4.5 Kết quả thống kê số lượng cho các biến phân loại

Từ kết quả hình 4.4, cho thấy với biến Dedicated: có 2822 mẫu quan sát là GPU rời (card đồ họa rời) và 241 mẫu quan sát là GPU tích hợp (card đồ họa tích hợp).

Đối với biến Shader: có 1 quan sát sử dụng phiên bản Shader (trình đồ bóng, tính toán các mức độ sáng tối, màu sắc) 1, có 7 quan sát sử dụng phiên bản Shader 1.4, có 39 quan sát sử dụng phiên bản Shader 2, có 82 quan sát sử dụng phiên bản Shader 3, có 194 quan sát sử dụng phiên bản Shader 4, có 126 quan sát sử dụng phiên bản Shader 4.1, có 2587 quan sát sử dụng phiên bản Shader 5.

Ta nhận thấy Shader có 2 phân loại có số lượng quan sát rất nhỏ so với các nhóm còn lại, điều này có thể gây ảnh hưởng trong việc dự báo (có thể dự báo tốt đối với các phân loại 2, 3, 4, 4.1 và 5, dự báo kém đối với phân loại 1 và 1.4). Do đó, ta sẽ loại bỏ các quan sát mà Shader nhận giá trị 1 và 1.4.

```

1 main_data <- subset(main_data, !(Shader %in% c(1, 1.4)))

```

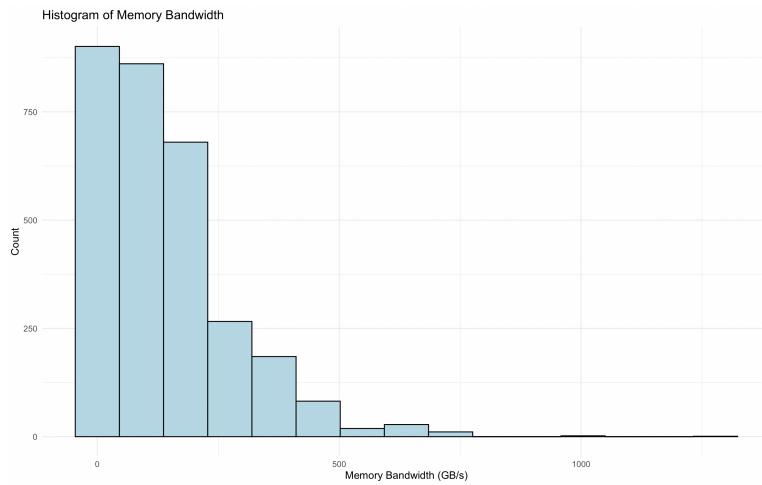
4.2 Trực quan hóa dữ liệu bằng các đồ thị

Ta dùng lệnh ggplot trong thư viện ggplot2 để vẽ đồ thị histogram thể hiện phân bố tần số cho biến Memory_Bandwidth.

```

1 ggplot(main_data, aes(x = Memory_Bandwidth)) +
2 geom_histogram(fill = "lightblue", color = "black", bins = 15) +
3 labs(title = "Histogram of Memory Bandwidth", x = "Memory Bandwidth (GB/s)", y = "Count") +
4 theme_minimal()

```



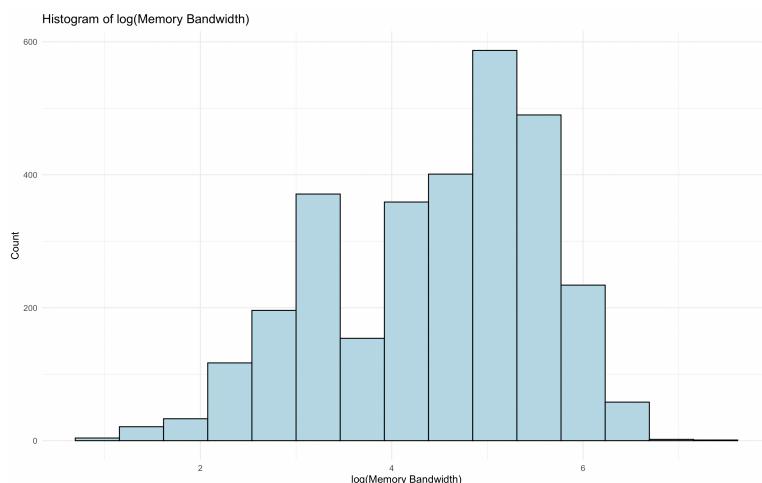
Hình 4.6 Kết quả đồ thị histogram của biến Memory_Bandwidth

Đồ thị của biến Memory_Bandwidth có phân phối lệch. Điều này dẫn đến việc vi phạm điều kiện của mô hình hồi quy (biến phụ thuộc Memory_Bandwidth cần tuân theo phân phối chuẩn). Do đó, để cải thiện hiệu quả của mô hình hồi quy tuyến tính dự báo Memory_Bandwidth, ta sẽ thực hiện chuyển đổi các biến liên tục sang dạng logagit tự nhiên (dạng $\log(x+1)$), sau đó vẽ lại đồ thị histogram cho biến Memory_Bandwidth.

```

1 main_data$Memory_Bandwidth<-log(main_data$Memory_Bandwidth)
2 main_data$Memory_Speed<-log(main_data$Memory_Speed)
3 main_data$L2_Cache<-log(main_data$L2_Cache)
4 main_data$Memory_Bus<-log(main_data$Memory_Bus)
5
6 ggplot(main_data, aes(x = Memory_Bandwidth)) +
7 geom_histogram(fill = "lightblue", color = "black", bins = 15) +
8 labs(title = "Histogram of log(Memory Bandwidth)", x = "log(Memory Bandwidth)", y = "Count") +
9 theme_minimal()

```



Hình 4.7 Kết quả đồ thị histogram của biến Memory_Bandwidth

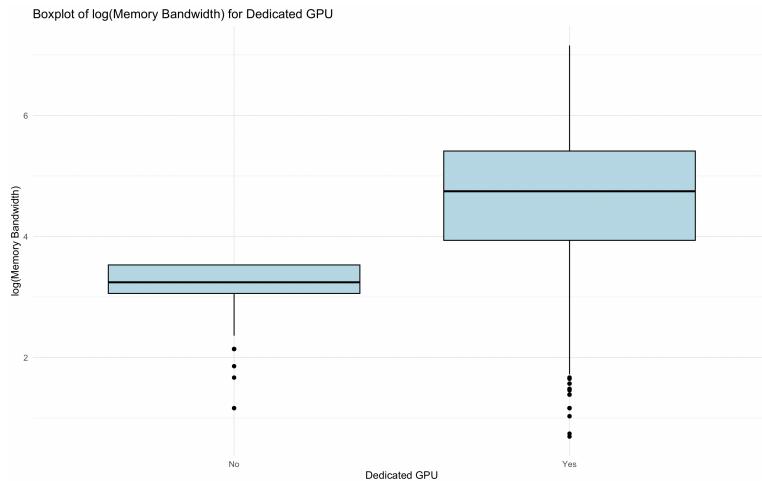
Việc lấy logarit tự nhiên của Memory_Bandwidth cho thấy biến này trở nên có phân phối đối xứng, gần phân phối chuẩn hơn. Điều đó dẫn đến sẽ cải thiện hiệu quả mô hình.

Tiếp theo, ta vẽ đồ thị boxplot của log(Memory Bandwidth) theo các biến phân loại. Đối với biến Dedicated, ta thu được kết quả như hình 4.8.

```

1 ggplot(main_data, aes(x = Dedicated, y = Memory_Bandwidth)) +
2 geom_boxplot(fill = "lightblue", color = "black") +
3 labs(title = "Boxplot of log(Memory Bandwidth) for Dedicated GPU",
4 x = "Dedicated GPU", y = "log(Memory Bandwidth)") +
5 theme_minimal()

```



Hình 4.8 Kết quả đồ thị boxplot của log(Memory Bandwidth) theo Dedicated

Đồ thị hình 4.8 cho ta các giá trị (min, max, Q1, Q2, Q3, và các điểm ngoại lai) của log(Memory Bandwidth) ở hai nhóm GPU rời và GPU tích hợp. Ta nhận thấy có sự khác biệt trong log(Memory Bandwidth), cụ thể:

- Miền giá trị của log(Memory Bandwidth) ở nhóm No thấp hơn so với nhóm Yes.
- Độ dao động của log(Memory Bandwidth) ở nhóm No nhỏ hơn so với nhóm Yes.

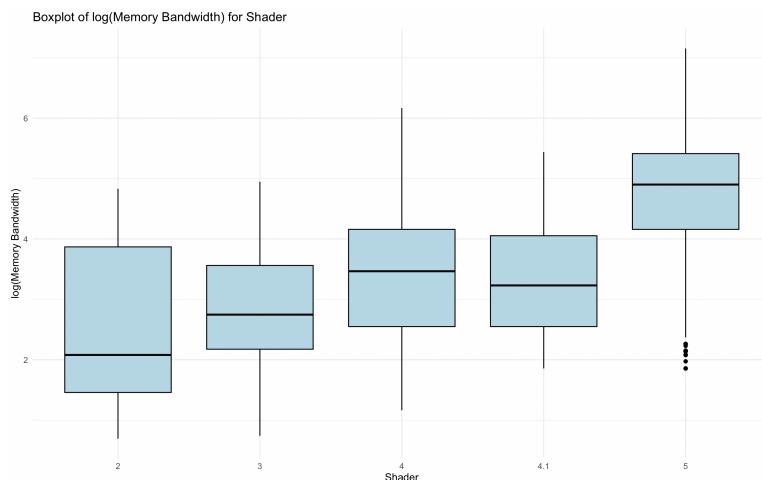
Những điều này cho ta dự đoán Dedicated có ý nghĩa trong việc dự báo Memory Bandwidth.

Đối với biến Shader, ta thu được kết quả như hình 4.9.

```

1 ggplot(main_data, aes(x = Shader, y = Memory_Bandwidth)) +
2 geom_boxplot(fill = "lightblue", color = "black") +
3 labs(title = "Boxplot of log(Memory Bandwidth) for Shader",
4 x = "Shader", y = "log(Memory Bandwidth)") +
5 theme_minimal()

```



Hình 4.9 Kết quả đồ thị boxplot của log(Memory Bandwidth) theo Shader

Đồ thị hình 4.9 cho ta các giá trị (min, max, Q1, Q2, Q3, và các điểm ngoại lai) của log(Memory Bandwidth) ở các nhóm phiên bản Shader. Ta nhận thấy có sự khác biệt trong log(Memory Bandwidth), cụ thể:

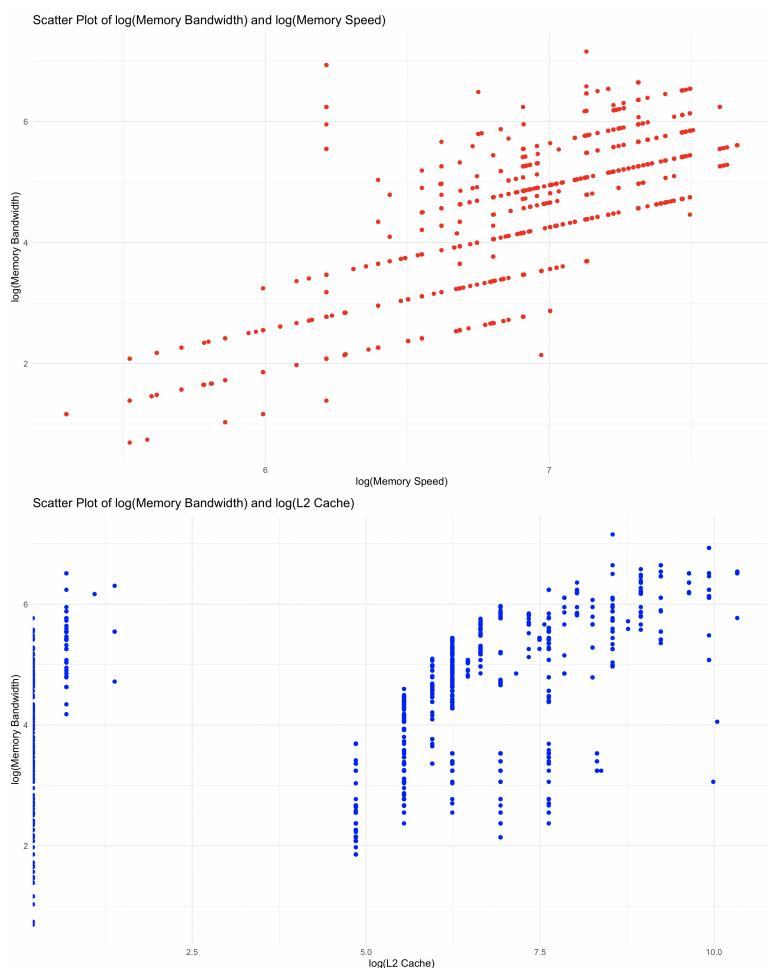
- Miền giá trị của log(Memory Bandwidth) ở có sự tăng dần theo các phiên bản Shader.
- Độ dao động của log(Memory Bandwidth) ở nhóm Shader 2 cao hơn các nhóm khác.

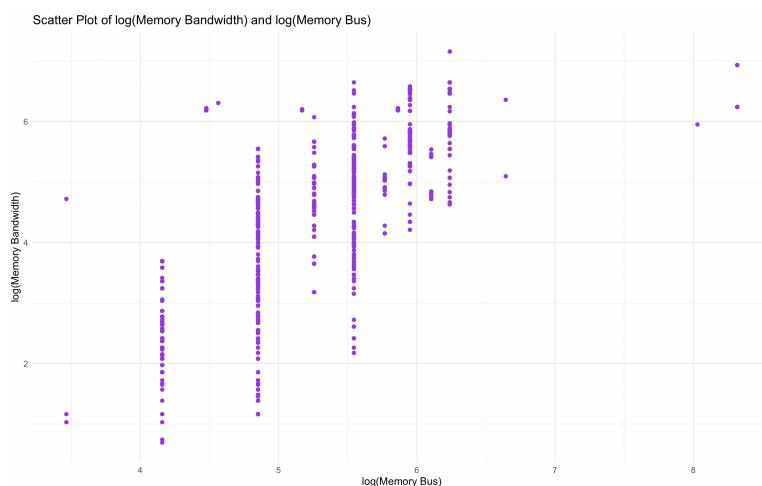
Những điều này cho ta dự đoán Shader có ý nghĩa trong việc dự báo Memory Bandwidth.

Tiếp theo, ta sẽ vẽ các đồ thị phân tán giữa log(Memory_Bandwidth) theo log(Memory_Speed), log(L2_Cache), log(Memory_Bus).

```

1 ggplot(main_data, aes(x = Memory_Speed, y = Memory_Bandwidth)) +
2 geom_point(color = "red") +
3 labs(title = "Scatter Plot of log(Memory Bandwidth) and log(Memory Speed)",
4 x = "log(Memory Speed)", y = "log(Memory Bandwidth)") +
5 theme_minimal()
6
7 ggplot(main_data, aes(x = L2_Cache, y = Memory_Bandwidth)) +
8 geom_point(color = "blue") +
9 labs(title = "Scatter Plot of log(Memory Bandwidth) and log(L2 Cache)",
10 x = "log(L2 Cache)", y = "log(Memory Bandwidth)") +
11 theme_minimal()
12
13 ggplot(main_data, aes(x = Memory_Bus, y = Memory_Bandwidth)) +
14 geom_point(color = "purple") +
15 labs(title = "Scatter Plot of log(Memory Bandwidth) and log(Memory Bus)",
16 x = "log(Memory Bus)", y = "log(Memory Bandwidth)") +
17 theme_minimal()
```



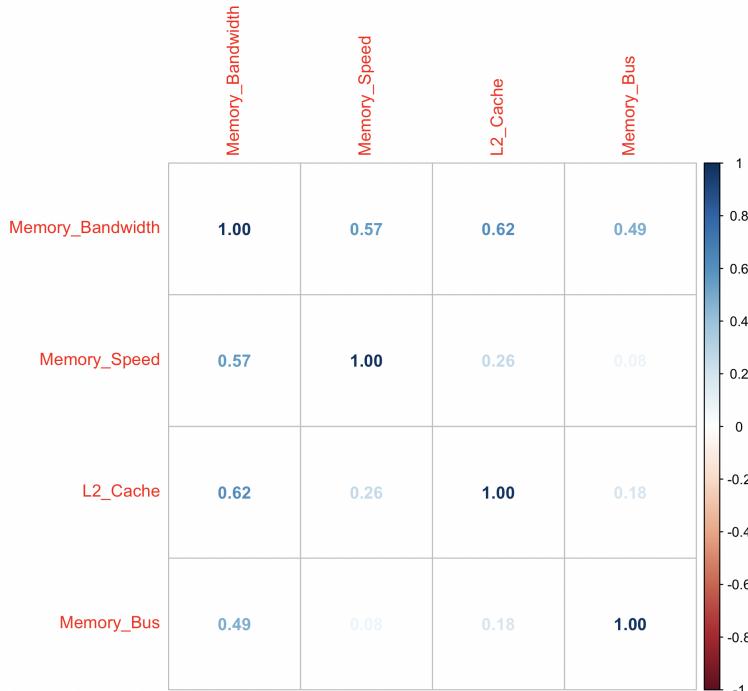


Hình 4.10 Kết quả đồ thị phân tán của $\log(\text{Memory Bandwidth})$ theo $\log(\text{Memory_Speed})$, $\log(\text{L2_Cache})$, $\log(\text{Memory_Bus})$.

Dựa trên các đồ thị phân tán, ta nhận thấy $\log(\text{Memory Bandwidth})$ có quan hệ tuyến tính ở mức độ trung bình khá với $\log(\text{Memory_Speed})$, $\log(\text{L2_Cache})$, $\log(\text{Memory_Bus})$.

Tiếp theo, ta sẽ thực hiện vẽ đồ thị ma trận tương quan để nhận xét mối tương quan giữa các biến liên tục (đã được lấy logarit tự nhiên).

```
1 library(corrplot)
2 corrplot(cor(numeric_data), method = "number")
```



Hình 4.11 Kết quả đồ thị ma trận tương quan

Đồ thị hình 4.11 đánh giá mối tương quan giữa các biến thông qua hệ số tương quan, cụ thể:

- Đối với các cặp biến độc lập, trị tuyệt đối hệ số tương quan $|R|$ ở các cặp biến đều dưới 0.65, cho thấy các biến độc lập có quan hệ tuyến tính ở mức độ trung bình hoặc yếu. Do vậy không có hiện tượng đa cộng tuyến xảy ra.
- Đối với biến phụ thuộc và các biến độc lập, trị tuyệt đối hệ số tương quan $|R|$ ở các cặp biến đều từ 0.49 trở lên, cho thấy biến phụ thuộc và các biến độc lập có quan hệ tuyến tính ở mức độ trung bình, khá.

5 THỐNG KÊ SUY DIỄN

Mô hình tối ưu và đơn giản để dự báo cho băng thông bộ nhớ GPUs là mô hình hồi quy tuyến tính bội.

Trước khi thực hiện mô hình hồi quy tuyến tính bội, ta sẽ chia dữ liệu của mình thành 2 phần gọi là tập huấn luyện và tập kiểm tra theo tỉ lệ 8:2. Ta sử dụng dữ liệu huấn luyện (chứa 2422 quan sát) để xây dựng mô hình của mình và sử dụng dữ liệu từ tập kiểm tra (chứa 606 quan sát) thực hiện dự báo và đánh giá xem mô hình của mình có dự báo tốt hay không.

```
1 set.seed(10)
2 train.rows <- sample(rownames(main_data), dim(main_data)[1] * 0.8)
3 train_data <- main_data[train.rows, ]
4 test.rows <- setdiff(rownames(main_data), train.rows)
5 test_data <- main_data[test.rows, ]
```

Ta xây dựng mô hình với:

- Biến phụ thuộc: $\log(\text{Memory_Bandwidth} + 1)$
- Biến độc lập: $\log(\text{Memory Speed} + 1)$, $\log(\text{L2_Cache} + 1)$, $\log(\text{Memory_Bus} + 1)$, DedicatedYes, Shader3, Shader4, Shader4.1, Shader5.

Mô hình tổng quát có dạng như sau:

$$\log(\text{Memory_Bandwidth} + 1) = \beta_0 + \beta_1.\log(\text{Memory_Speed} + 1) + \beta_2.\log(\text{L2_Cache} + 1) + \beta_3.\log(\text{Memory_Bus} + 1) + \beta_4.\text{DedicatedYes} + \beta_5.\text{Shader3} + \beta_6.\text{Shader4} + \beta_7.\text{Shader4.1} + \beta_8.\text{Shader5} + \varepsilon$$

Thực hiện ước lượng các hệ số β_i :

```
1 model<-lm(Memory_Bandwidth~Memory_Speed+L2_Cache+Memory_Bus+Dedicated+Shader ,train_
  data)
2 summary(model)
```

```
Call:
lm(formula = Memory_Bandwidth ~ Memory_Speed + L2_Cache + Memory_Bus +
    Dedicated + Shader, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.76645 -0.18755 -0.07595  0.09153  1.94423 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -12.108702   0.160279 -75.548 < 2e-16 ***
Memory_Speed  1.481960   0.025383  58.383 < 2e-16 ***
L2_Cache     0.037218   0.004044   9.203 < 2e-16 ***
Memory_Bus   1.031544   0.014609  70.612 < 2e-16 ***
DedicatedYes 0.414839   0.029783  13.929 < 2e-16 ***
Shader3      0.272968   0.076037   3.590 0.000337 ***
Shader4      0.176184   0.071022   2.481 0.013180 *  
Shader4.1     0.246763   0.075227   3.280 0.001052 ** 
Shader5      0.260276   0.072080   3.611 0.000311 *** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3591 on 2413 degrees of freedom
Multiple R-squared:  0.8965, Adjusted R-squared:  0.8961 
F-statistic: 2612 on 8 and 2413 DF, p-value: < 2.2e-16
```

Hình 5.12 Kết quả mô hình hồi quy tuyến tính bội

Phương trình ước lượng thu được như sau:

$$\log(\text{Memory_Bandwidth} + 1) = -12.108702 + 1.48196.\log(\text{Memory_Speed} + 1) + 0.037218.\log(\text{L2_Cache} + 1) + 1.031544.\log(\text{Memory_Bus} + 1) + 0.414839.\text{DedicatedYes} + 0.272968.\text{Shader3} + 0.176184.\text{Shader4} + 0.246763.\text{Shader4.1} + 0.260276.\text{Shader5}$$

Kiểm định các hệ số β_i với các giả thuyết sau:

Giả thuyết $H_0 : \beta_i = 0$

Giả thuyết đối $H_1 : \beta_i \neq 0$

Ta thấy rằng p_value ứng với các biến đều bé hơn mức ý nghĩa 5% nên ta bác bỏ giả thuyết H_0 . Vậy các hệ số β_i này khác 0, hay nói cách khác các biến này có ý nghĩa trong việc dự báo Memory Bandwidth.

Ngoài ra, ta có thể kiểm định sự phù hợp của mô hình hồi quy với giả thuyết:

Giả thuyết $H_0: \beta_1 = \beta_2 = \beta_3 = \dots = \beta_7 = \beta_8 = 0$

Giả thuyết đối $H_1: \exists \beta_i \neq 0$

Ta thấy rằng p-value < 2.2e-16 ở kiểm định F, rất bé hơn mức ý nghĩa 5% nên ta bác bỏ giả thuyết H_0 . Điều này cho thấy mô hình hồi quy phù hợp (có ít nhất 1 hệ số $\beta_i \neq 0$)

R^2 hiệu chỉnh của mô hình là 0.8961, thể hiện có 89.61% sự biến thiên của log(Memory Bandwidth + 1) được giải thích bởi các biến độc lập trong mô hình.

Như vậy, ta suy ra được mô hình dự báo băng thông bộ nhớ Memory Bandwidth như sau:

$$\begin{aligned} \text{Memory_Bandwidth} = & \exp(-12.108702 + 1.48196 \log(\text{Memory_Speed} + 1) + \\ & 0.037218 \cdot \log(\text{L2_Cache} + 1) + 1.031544 \cdot \log(\text{Memory_Bus} + 1) + 0.414839 \cdot \text{DedicatedYes} + \\ & 0.272968 \cdot \text{Shader3} + 0.176184 \cdot \text{Shader4} + 0.246763 \cdot \text{Shader4.1} + 0.260276 \cdot \text{Shader5}) - 1 \end{aligned}$$

Ta đánh giá tác động của các biến đối với Memory Bandwidth:

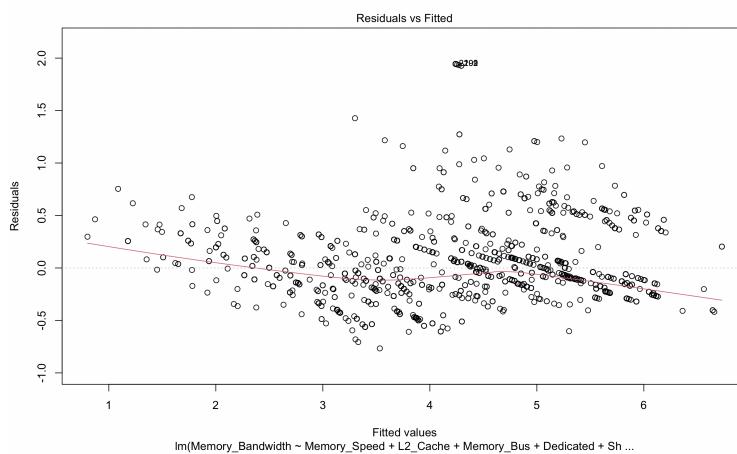
- Hệ số $\hat{\beta}_1 = 1.48196$, cho thấy nếu ta tăng Memory Speed thêm 1 đơn vị (ví dụ, từ 1 MHz lên 2 MHz), Memory Bandwidth sẽ được kỳ vọng tăng lên nhưng với tỷ lệ khoảng 1.48196 lần sự thay đổi trong logarit của tốc độ bộ nhớ.
- Hệ số $\hat{\beta}_2 = 0.037218$, cho thấy nếu ta tăng L2 Cache thêm 1 đơn vị (ví dụ, từ 1 MB lên 2 MB), Memory Bandwidth sẽ được kỳ vọng tăng lên nhưng với tỷ lệ khoảng 0.037218 lần sự thay đổi trong logarit của tốc độ bộ nhớ.
- Hệ số $\hat{\beta}_3 = 1.031544$, cho thấy nếu ta tăng Memory Bus thêm 1 đơn vị (ví dụ, từ 1 MHz lên 2 MHz), Memory Bandwidth sẽ được kỳ vọng tăng lên nhưng với tỷ lệ khoảng 1.031544 lần sự thay đổi trong logarit của tốc độ bộ nhớ.
- Hệ số $\hat{\beta}_{4.1} = 0.414839$ cho thấy khi DedicatedYes = 1, Memory Bandwidth sẽ tăng lên trung bình 0.414839 đơn vị.
- Hệ số $\hat{\beta}_{5.1} = 0.272968$ cho thấy khi Shader3 = 1 (tức là hệ thống có hỗ trợ Shader 3), Memory Bandwidth sẽ tăng lên trung bình 0.272968 đơn vị.
- Hệ số $\hat{\beta}_{5.2} = 0.176184$ cho thấy khi Shader4 = 1 (tức là hệ thống có hỗ trợ Shader 4), Memory Bandwidth sẽ tăng lên trung bình 0.176184 đơn vị.
- Hệ số $\hat{\beta}_{5.3} = 0.246763$ cho thấy khi Shader4.1 = 1 (tức là hệ thống có hỗ trợ Shader 4.1), Memory Bandwidth sẽ tăng lên trung bình 0.246763 đơn vị.
- Hệ số $\hat{\beta}_{5.4} = 0.260276$ cho thấy khi Shader5 = 1 (tức là hệ thống có hỗ trợ Shader 5), Memory Bandwidth sẽ tăng lên trung bình 0.260276 đơn vị.

Các giả định của mô hình hồi quy tuyến tính bao gồm:

- Sai số có phân phối chuẩn.
- Sai số có kỳ vọng bằng 0.
- Phương sai của các sai số là hằng số
- Các sai số $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$ độc lập với nhau.
- Không có hiện tượng đa cộng tuyến xảy ra giữa những biến độc lập.

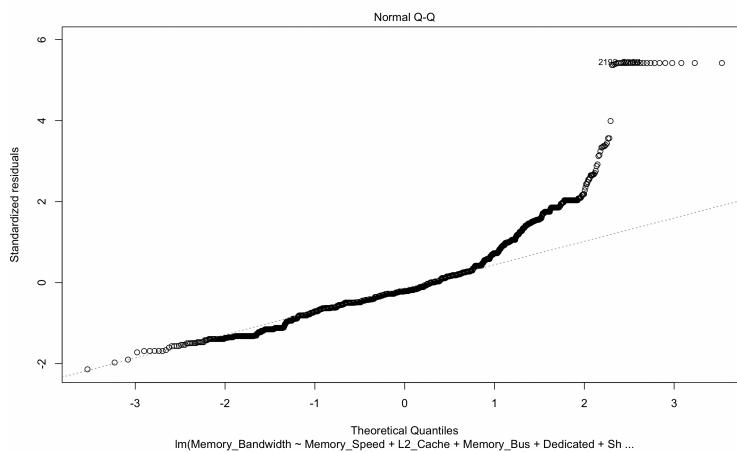
Ta sẽ thực hiện vẽ các đồ thị kiểm tra các điều kiện.

```
1 plot(model, which=1)
2 plot(model, which=2)
3 plot(model, which=3)
4 plot(model, which=5)
```



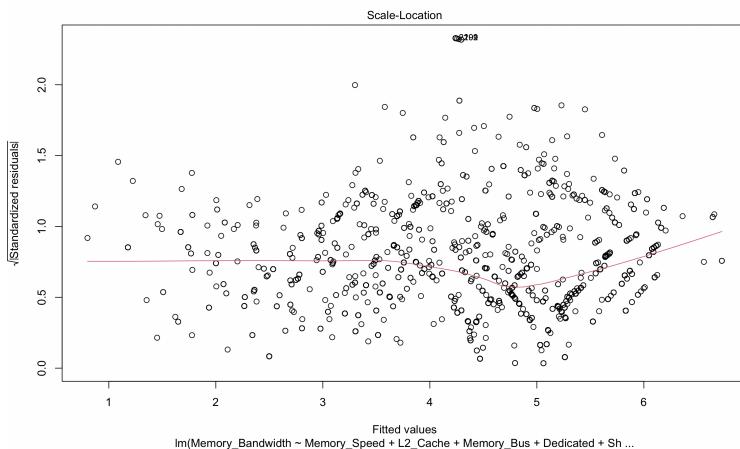
Hình 5.13 Kết quả đồ thị Residuals and Fitted

Đồ thị Residuals and Fitted vẽ ra những điểm sai số tương ứng với các giá trị dự báo. Đồ thị này dùng để kiểm tra giả định sai số có kỳ vọng bằng 0. Ta nhìn thấy đường màu đỏ chưa nằm sát đường thẳng $y = 0$ nên giả định sai số có kỳ vọng bằng 0 chưa thỏa mãn.



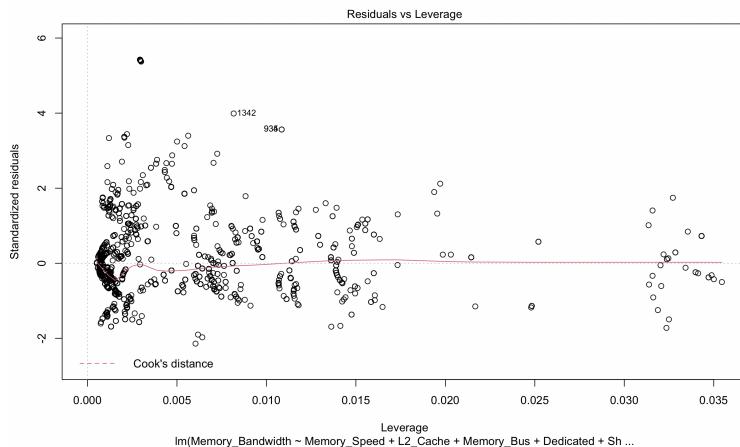
Hình 5.14 Kết quả đồ thị Normal Q-Q

Đồ thị Normal Q-Q vẽ ra các sai số được chuẩn hoá, và dùng để kiểm tra giả định sai số có phân phối chuẩn. Ta nhận thấy có nhiều quan trắc nằm lệch khỏi đường thẳng kỳ vọng phân phối chuẩn, cho thấy giả định này chưa thỏa mãn.



Hình 5.15 Kết quả đồ thị Scale-Location

Đồ thị Scale-Location vẽ căn bậc hai các sai số được chuẩn hoá, dùng để kiểm tra giả định phương sai các sai số là hằng số. Dựa trên đồ thị, ta thấy đường thẳng màu đỏ gấp khúc và không nằm ngang và các quan trắc lại không phân tán ngẫu nhiên dọc theo đường màu đỏ (tập tập nhiều ở phần giữa và thừa thoát ở hai đầu đường thẳng). Điều này cho thấy giả định phương sai các sai số là hằng số bị vi phạm.



Hình 5.16 Kết quả đồ thị Residuals và Leverage

Đồ thị Residuals và Leverage sẽ chỉ ra các điểm có ảnh hưởng cao đến mô hình hồi quy, ví dụ như các quan trắc 1342, 934, 935. Tuy nhiên vì trong đồ thị, ta không thấy đường Cook's distance nên các điểm này chưa vượt ra khỏi đường Cook's distance, do đó các quan trắc này không phải là các giá trị ngoại lai nên ta sẽ không cần phải loại bỏ khi phân tích.

Đối với hiện tượng đa cộng tuyến, ta thực hiện kiểm tra bằng cách tính toán chỉ số vif trong thư viện car.

```

1 library(car)
2 vif(model)

```

	GVIF	Df	GVIF^(1/(2*Df))
Memory_Speed	1.906564	1	1.380784
L2_Cache	2.483188	1	1.575813
Memory_Bus	1.265630	1	1.125002
Dedicated	1.093050	1	1.045490
Shader	2.855470	4	1.140144

Hình 5.17 Kết quả chỉ số VIF

Từ kết quả hình 5.17 cho thấy chỉ số VIF thuộc $(1,5]$ nên có hiện tượng đa cộng tuyến vừa phải giữa các biến độc lập. Do đó ta không cần loại bỏ các biến trong mô hình.

Cuối cùng thực hiện dự báo strength trung bình dựa trên tệp test_data bằng lệnh predict.

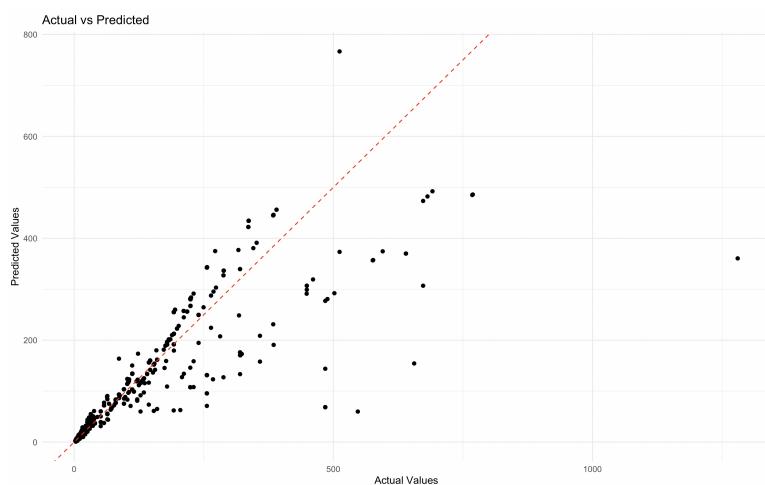
```
1 test_data$predicted <- predict(model, test_data)
2 head(test_data, 10)
```

	Memory_Bandwidth	Memory_Speed	L2_Cache	Memory_Bus	Shader	Dedicated	predicted
4	3.632309	7.048386	0.000000	4.859812	4	Yes	3.940858
6	3.589059	7.003974	0.000000	4.859812	4	Yes	3.875040
27	4.976734	6.621406	6.645091	5.953243	5	Yes	4.767422
44	3.430756	6.839476	6.932448	4.859812	5	No	3.558529
46	6.306640	7.262629	1.609438	4.574711	5	Yes	4.108254
58	5.667810	7.315218	7.337588	5.953243	5	Yes	5.821399
63	5.667810	7.315218	7.337588	5.953243	5	Yes	5.821399
64	5.667810	7.315218	7.337588	5.953243	5	Yes	5.821399
65	5.667810	7.315218	7.337588	5.953243	5	Yes	5.821399
77	1.435085	5.303305	0.000000	4.859812	2	Yes	1.178532

Hình 5.18 Kết quả dự báo

Ta thực hiện chuyển đổi giá trị logarit tự nhiên về dạng nguyên bản để so sánh giá trị thực tế và giá trị dự báo, sau đó vẽ đồ thị để nhận xét mối quan hệ tuyến tính giữa giá trị thực tế và dự báo bằng hàm ggplot trong thư viện ggplot2.

```
1 actual_value <- exp(test_data$Memory_Bandwidth)-1
2 predicted_value <- exp(test_data$predicted)-1
3 compare <- data.frame(actual_value, predicted_value)
4 library(ggplot2)
5 ggplot(compare, aes(x = actual_value, y = predicted_value)) + geom_point() +
6 geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
7 labs(title = "Actual vs Predicted", x = "Actual Values", y = "Predicted Values") +
8 theme_minimal()
```



Hình 5.19 Kết quả so sánh giữa thực tế và dự báo

Đồ thị cho thấy có sự phân tán lớn xung quanh đường chéo, điều này cho thấy việc dự báo Memory_Bandwidth có sự sai lệch lớn, do đó mô hình ta dự báo chưa thực sự tốt.

6 THẢO LUẬN VÀ MỞ RỘNG

Kết quả của mô hình hồi quy ta vừa xây dựng trên train_data cho thấy các biến giải thích đều có ý nghĩa thống kê, đóng góp vào việc dự báo băng thông bộ nhớ (Memory Bandwidth). Cụ thể, các hệ số ước lượng cho từng biến cho thấy sự tương quan mạnh mẽ với Memory_Bandwidth, với các yếu tố như Memory Speed, Memory Bus, và Dedicated có ảnh hưởng đáng kể đến dự báo băng thông. Điều này khẳng định rằng các yếu tố này thực sự quan trọng trong việc dự đoán hiệu suất bộ nhớ của hệ thống.

Ưu điểm của mô hình:

- **Khả năng giải thích tốt:** Mô hình này sử dụng các yếu tố phần cứng như Memory Speed, L2 Cache, Memory Bus, và các yếu tố phần mềm như Shader, giúp làm rõ cách các yếu tố này ảnh hưởng đến Memory Bandwidth.
- **Dễ hiểu và thực tế:** Các biến đầu vào của mô hình đều là các chỉ số mà người dùng hoặc kỹ sư có thể đo đạc và kiểm tra trong thực tế, do đó mô hình dễ dàng được áp dụng vào các tình huống thực tế.
- **Dảm bảo tính chính xác:** Việc sử dụng các yếu tố như Dedicated Memory và Shader cung cấp thêm các chỉ số quan trọng có thể cải thiện độ chính xác của mô hình khi dự báo hiệu suất bộ nhớ.

Nhược điểm của mô hình:

- **Mối quan hệ phi tuyến tính:** Mô hình hồi quy tuyến tính giả định một mối quan hệ tuyến tính giữa các yếu tố giải thích và Memory Bandwidth, trong khi thực tế có thể có mối quan hệ phi tuyến tính. Điều này có thể làm giảm độ chính xác của dự báo trong những trường hợp hệ thống phức tạp hơn.
- **Không xét đến các yếu tố ngoại lai:** Mô hình không tính đến các yếu tố khác có thể ảnh hưởng đến Memory Bandwidth, chẳng hạn như nhiệt độ hệ thống, phần mềm đang chạy, hoặc các yếu tố phần cứng khác chưa được đưa vào mô hình.
- **Giới hạn về các biến đầu vào:** Mặc dù các biến hiện tại có thể đã bao quát được phần lớn yếu tố ảnh hưởng đến băng thông bộ nhớ, nhưng mô hình vẫn có thể bị giới hạn nếu thiếu các thông tin bổ sung quan trọng khác về phần cứng hoặc phần mềm.

Tóm lại, mô hình hồi quy có hiệu quả trong việc dự báo và giải thích các yếu tố ảnh hưởng đến băng thông bộ nhớ, tuy nhiên cần cân nhắc điều chỉnh hoặc bổ sung các yếu tố phi tuyến tính và các yếu tố ngoại lai để cải thiện độ chính xác trong các tình huống phức tạp hơn.

7 NGUỒN DỮ LIỆU VÀ NGUỒN CODE

Nguồn dữ liệu: <https://www.kaggle.com/datasets/iliassekkaf/computerparts>
 Nguồn code:

```

1 #1 Doc du lieu
2 GPU_data <- read.csv("~/Desktop/All_GPUs.csv")
3 head(GPU_data)
4
5 GPU_data[GPU_data == ""] <- NA
6 GPU_data[] <- lapply(GPU_data, function(x) gsub("^\\n-", NA, x))
7 GPU_data[] <- lapply(GPU_data, function(x) gsub("^\\n$", NA, x))
8
9 library(questionr)
10 freq.na(GPU_data)
11
12 NA_summary <- data.frame(freq.na(GPU_data))
13 colnames(NA_summary) <- c("Missing", "Percent")
14 selected_columns <- rownames(NA_summary[NA_summary$Percent < 10, ])
15 new_GPU_data <- GPU_data[, selected_columns]
16 new_GPU_data<-na.omit(new_GPU_data)
17
18 str(new_GPU_data)
19
20 columns_to_clean <- c("L2_Cache", "Memory_Bandwidth", "Memory_Bus", "Memory_Speed")
21
22 remove_units <- function(column) {
23   cleaned_column <- gsub("[^0-9.]", "", column)
24   cleaned_column <- as.numeric(cleaned_column)
25   return(cleaned_column)
26 }
27
28 new_GPU_data[columns_to_clean]<-lapply(new_GPU_data[columns_to_clean],remove_units)
29
30
31 main_data<-new_GPU_data[c("Memory_Bandwidth", "Memory_Speed", "L2_Cache", "Memory_Bus",
32 "Shader", "Dedicated")]
32 head(main_data)
33
34 numeric_vars <- sapply(main_data, is.numeric)
35 numeric_data <- main_data[, numeric_vars]
36
37 summary_stats <- sapply(numeric_data, function(x) {
38   c(
39     Mean = mean(x),
40     SD = sd(x),
41     Min = min(x),
42     Q1 = quantile(x, 0.25),
43     Median = median(x),
44     Q3 = quantile(x, 0.75),
45     Max = max(x)
46   )
47 })
48
49 t(as.data.frame(summary_stats))
50
51 table(main_data$Dedicated)
52 table(main_data$Shader)
53
54 main_data <- subset(main_data, !(Shader %in% c(1, 1.4)))
55
56 library(ggplot2)
57 ggplot(main_data, aes(x = Memory_Bandwidth)) +
58 geom_histogram(fill = "lightblue", color = "black", bins = 15) +
59 labs(title = "Histogram of Memory Bandwidth", x = "Memory Bandwidth (GB/s)", y =
      "Count") + theme_minimal()

```

```

60 main_data$Memory_Bandwidth<-log1p(main_data$Memory_Bandwidth)
61 main_data$Memory_Speed<-log1p(main_data$Memory_Speed)
62 main_data$L2_Cache<-log1p(main_data$L2_Cache)
63 main_data$Memory_Bus<-log1p(main_data$Memory_Bus)
64
65 ggplot(main_data, aes(x = Memory_Bandwidth)) +
66 geom_histogram(fill = "lightblue", color = "black",bins = 15) +
67 labs(title = "Histogram of log(Memory Bandwidth)", x = "log(Memory Bandwidth)", y =
68     "Count") +
69 theme_minimal()
70
71 ggplot(main_data, aes(x = Dedicated, y = Memory_Bandwidth)) +
72 geom_boxplot(fill = "lightblue", color = "black") +
73 labs(title = "Boxplot of log(Memory Bandwidth) for Dedicated GPU",
74 x = "Dedicated GPU",
75 y = "log(Memory Bandwidth)") +
76 theme_minimal()
77
78 ggplot(main_data, aes(x = Shader, y = Memory_Bandwidth)) +
79 geom_boxplot(fill = "lightblue", color = "black") +
80 labs(title = "Boxplot of log(Memory Bandwidth) for Shader",
81 x = "Shader",
82 y = "log(Memory Bandwidth)") +
83 theme_minimal()
84
85 ggplot(main_data, aes(x = Memory_Speed, y = Memory_Bandwidth)) +
86 geom_point(color = "red") +
87 labs(title = "Scatter Plot of log(Memory Bandwidth) and log(Memory Speed)",
88 x = "log(Memory Speed)",
89 y = "log(Memory Bandwidth)") +
90 theme_minimal()
91
92 ggplot(main_data, aes(x = L2_Cache, y = Memory_Bandwidth)) +
93 geom_point(color = "blue") +
94 labs(title = "Scatter Plot of log(Memory Bandwidth) and log(L2 Cache)",
95 x = "log(L2 Cache)",
96 y = "log(Memory Bandwidth)") +
97 theme_minimal()
98
99 ggplot(main_data, aes(x = Memory_Bus, y = Memory_Bandwidth)) +
100 geom_point(color = "purple") +
101 labs(title = "Scatter Plot of log(Memory Bandwidth) and log(Memory Bus)",
102 x = "log(Memory Bus)",
103 y = "log(Memory Bandwidth)") +
104 theme_minimal()
105
106 library(corrplot)
107 corrplot(cor(numeric_data),method = "number")
108
109 set.seed(10)
110 train.rows <- sample(rownames(main_data), dim(main_data)[1] * 0.8)
111 train_data <- main_data[train.rows, ]
112 test.rows <- setdiff(rownames(main_data), train.rows)
113 test_data <- main_data[test.rows, ]
114
115 model<-lm(Memory_Bandwidth~Memory_Speed+L2_Cache+Memory_Bus+Dedicated+Shader ,
116             train_data)
117 summary(model)
118
119 plot(model,which=1)
120 plot(model,which=2)
121 plot(model,which=3)
122 plot(model,which=5)
123
124 library(car)
125 vif(model)

```

```
124  
125 test_data$predicted <- predict(model,test_data)  
126 head(test_data,10)  
127  
128 actual_value <- exp(test_data$Memory_Bandwidth)-1  
129 predicted_value <- exp(test_data$predicted)-1  
130 compare<-data.frame(actual_value,predicted_value)  
131 library(ggplot2)  
132 ggplot(compare, aes(x = actual_value, y = predicted_value)) + geom_point() +  
133 geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +  
134 labs(title = "Actual vs Predicted", x = "Actual Values", y = "Predicted Values") +  
135 theme_minimal()
```

8 TÀI LIỆU THAM KHẢO

- [1] Stallings, W. (2019). Computer Organization and Architecture: Designing for Performance (10th ed.). Pearson.
- [2] Patterson, D. A., & Hennessy, J. L. (2017). Computer Organization and Design: The Hardware/-Software Interface (5th ed.). Morgan Kaufmann.
- [3] Bitar, A., & Pan, L. (2015). Memory Architecture and Performance in High-End GPUs: A Survey. Journal of Computer Architecture, 42(3), 234-245.
- [4] AMD. (2019). Memory Bus Architecture and Bandwidth. AMD.
- [5] NVIDIA Developer Blog. (2018). The Effect of Shaders on Graphics and Memory Bandwidth. NVIDIA.
- [6] TechSpot. (2016). The Impact of Memory Speed on System Performance. TechSpot.
- [7] Introductory Statistics with R, J.Jambers - D.Hand - W.Hardle.
- [8] Peter Dalgaard. 2008. Introductory Statistics with R. Springer.
- [9] Douglas C. Montgomery, George C. Runger (2018). Applied Statistics and Probability]