Trường Đại Học Bách Khoa Tp.Hồ Chí Minh
Khoa Khoa Học & Kỹ Thuật Máy Tính

Xác Suất Thống Kê (MT2013)

BÁO CÁO BÀI TẬP LỚN

&

Giai đoạn 2: Tiểu luận hoàn thiện

GVHD: Nguyễn Thị Mộng Ngọc
Lớp: L15
Nhóm: 13

Họ Tên	MSSV	Đóng góp
Nguyễn Minh Khánh	2111493	100%
Lưu Chí Lập	2211830	100%
Nguyễn Anh Kiệt	2211758	100%
Hàng Nhựt Long	2211874	100%
Trần Minh Khang	2211472	100%

Ho Chi Minh City, 5/2024

Mục lục

I. Tổng quan dữ liệu

1.1 Giới thiệu chung

GPU (Graphic Processing Unit): Là bộ vi xử lý chuyên phân tích những khối dữ liệu hình ảnh. Những tác vụ liên quan tới đồ họa và video. Khác biệt với CPU là GPU chuyên dụng xử lý những tác vụ hình ảnh. Hai hãng sản xuất GPU nổi tiếng trên thị trường hiện nay đó chính là Nvidia và AMD/ATI mỗi hãng đều có những đặc điểm và lợi thế khác nhau.

GPU còn xử lý thông tin đa luồng, song song và bộ nhớ ở tốc độ cao. Kỹ thuật GPU đang dần trở nên dễ lập trình, cung cấp nhiều tiềm năng cho việc tăng tốc xử lý cho nhiều chương trình với nhiều mục đích khác nhau, hơn cả bộ xử lý trung tâm (CPUs).

Trong đề tài này, ta sẽ phân tích tệp dữ liệu chứa thông số kỹ thuật chi tiết, ngày phát hành và giá phát hành của hơn 3000 bộ xử lý đồ họa (GPU) trong máy tính.

Dữ liệu đầu vào có chứa tệp lưu trữ dưới dạng CSV: gpus.csv cho bộ xử lý đồ họa (GPU). Bảng dữ liệu có danh sách các cột biểu thị cho thông tin cần thiết của dữ liệu và một số đặc đặc trưng sẽ bao gồm: tốc độ xung nhịp, nhiệt độ tối đa, độ phân giải màn hình, mức tiêu thụ năng lượng, số luồng, ngày phát hành, giá phát hành, kích thước khuôn, hỗ trợ ảo hóa và nhiều đề mục tương tự khác.

Dữ liệu được trích từ: Computer Parts (CPUs and GPUs).

1.2 Các biến trong bảng dữ liệu

STT	TÊN BIẾN	LOẠI BIẾN	ĐẶC TẢ
1	Best_resolution	Ngẫu nhiên	Kích thước tối ưu nhất cho màn hình để giảm thiểu tình trạng màn hình bị ngưng/dừng đột ngột.
2	Boost_Clock (MHz)	Liên tục	Tốc độ xử lý của card đồ họa sau khi boost clock.
3	Core_Speed (MHz)	Liên tục	Tốc độ xử lý của card đồ họa.
4	DVI_Connection	Rời rạc	Số lượng cổng hỗ trợ kết nối với các thiết bị điện tử như máy tính, tivi, laptop,... giúp truyền hình ảnh, phát video trực tiếp đến màn hình thông qua một kết nối duy nhất mà không cần phải chuyển về analog.
5	DisplayPort_Connecti on	Rời rạc	Số lượng cổng có chức năng trích xuất hình ảnh và âm thanh chất lượng cao từ thiết bị nguồn sang màn hình TV, laptop, máy chiếu, màn hình máy tính,...
6	HDMI_Connection	Rời rạc	Số lượng cổng dùng để truyền tải âm thanh và hình ảnh với những thiết bị khác như tivi, laptop... cùng độ phân giải cao.
7	Memory	Liên tục	Bộ nhớ của card đồ họa
8	Memory_Bandwidth	Liên tục	Tốc độ mà bộ xử lý có thể đọc hoặc lưu trữ dữ liệu vào bộ nhớ bán dẫn. Băng thông bộ nhớ thường được biểu thị bằng đơn vị byte/giây, mặc dù điều này có thể thay đổi đối với các hệ thống có kích thước dữ liệu tự nhiên không phải là bội số của byte 8 bit thường được sử dụng.
9	Memory_Bus	Liên tục	Một loại bus máy tính, thường ở dạng một bộ dây hoặc dây dẫn kết nối các thành phần điện và cho phép truyền dữ liệu và địa chỉ từ bộ nhớ chính đến bộ xử lý trung tâm (CPU) hoặc bộ điều khiển bộ nhớ.
10	Memory_Speed	Liên tục	Là tốc độ bộ nhớ RAM của card được tính bằng MHz, hiểu đơn giản là tốc độ mà card có thể truy cập dữ liệu được lưu trữ trên RAM.
11	PSU	Liên tục	Nguồn điện (Watt và Ampe) để card đồ họa hoạt động.
12	Pixels_Rate	Liên tục	Số pixel mỗi giây mà GPU có thể tạo ra.
13	Process	Liên tục	Loại công nghệ nano mà GPU sử dụng.
14	ROPs	Liên tục	Số lượng đường ống vận hành Raster trong GPU
15	Resolution_WxH	Liên tục	Độ phân giải tối đa được hỗ trợ.
16	TMUs	Liên tục	Số đơn vị ánh xạ kết cấu trong GPU.

STT	TÊN BIẾN	LOẠI BIẾN	ĐẶC TẢ
17	Texture_Rate	Liên tục	Số pixel kết cấu mỗi giây GPU có thể tạo ra.
18	VGA Connection	Rời rạc	Số lượng đầu nối tiêu chuẩn được sử dụng cho đầu ra hình ảnh.
19	Release_Price	Liên tục	Giá bán ra của GPU

1.3 Công cụ và môi trường

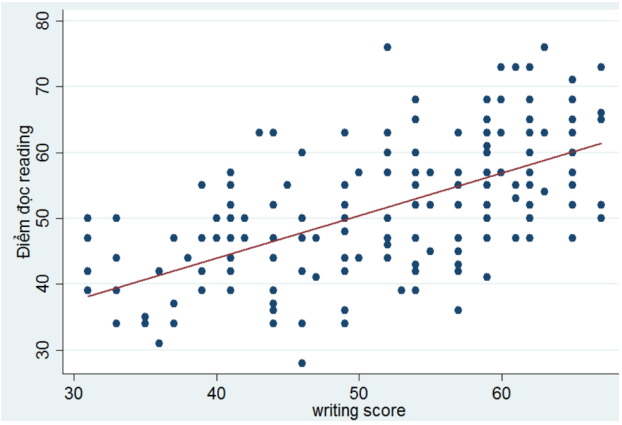
Trong bài tập lớn này, nhóm chủ yếu sử dụng R/R studio như là công cụ và môi trường để phân tích dữ liệu

II. Kiến thức nền

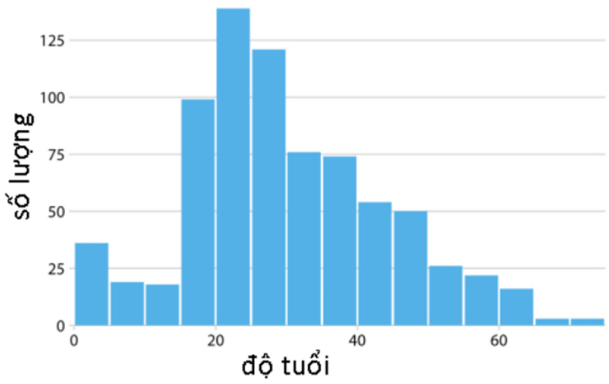
2.1 Phân tích và khám phá dữ liệu

Phân tích và khám phá dữ liệu (EDA) là một phương pháp để trích xuất thông tin từ dữ liệu. Nó thường bao gồm việc sử dụng các mẫu thông kê dạng bảng và các đồ thị để mô tả mối quan hệ giữa các biến. Mục tiêu là xác định các mô hình và xu hướng của mẫu dữ liệu, đồng thời phát hiện các giá trị ngoại lệ. EDA rất quan trọng trong việc trích xuất các đặc điểm quan trọng cho các mô hình dự đoán và giúp hiểu rõ hơn về dữ liệu gốc thông qua việc biểu thị. Điều này giúp người phân tích có cái nhìn sâu sắc hơn về dữ liệu, từ đó đưa ra quyết định chính xác hơn trong việc xây dựng mô hình hoặc giải quyết vấn đề cụ thể.

- Biểu đồ tần suất (Histograms) là biểu đồ cột sử dụng để biểu diễn phân phối tần suất của một tập dữ liệu liên tục. Thông qua biểu đồ này, chúng ta có thể dễ dàng nhận ra xu hướng phân phối, biên độ, độ tập trung của dữ liệu và sự phân tán của chúng. Điều này giúp cho việc đánh giá dữ liệu trở nên dễ dàng và hiệu quả hơn.
- Biểu đồ phân tán (Scatter plots) là dạng biểu đồ thống kê được sử dụng để biểu diễn mối quan hệ giữa hai biến trong một tập dữ liệu. Mỗi điểm trên biểu đồ đại diện cho một quan sát và được đặt tại tọa độ tương ứng với giá trị của hai biến. Scatterplot giúp người đọc nhận biết mối quan hệ giữa hai biến là tương quan dương ,tương quan âm hoặc không có mối quan hệ. Mục đích của scatterplot là: hiển thị mối quan hệ giữa hai biến, phát hiện xu hướng hoặc mẫu số học trong dữ liệu, đánh giá sự tương quan giữa các biến.



Hình 1: Minh hoạ biểu đồ histogram



Hình 2: Minh hoạ biểu đồ scatterplot

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

Trung bình cộng: là một đại lượng mô tả thống kê được tính ra bằng cách lấy tổng giá trị của toàn bộ các quan sát trong tập và chia cho số lượng các quan sát trong tập đó.

Độ lệch chuẩn: Độ lệch chuẩn, hay độ lệch tiêu chuẩn, là một đại lượng thống kê mô tả dùng để đo mức độ phân tán của một tập dữ liệu. Nó là căn bậc hai của phương sai. Độ lệch chuẩn biểu thị độ biến động trung bình của các giá trị từ giá trị trung bình của tập dữ liệu. Một độ lệch chuẩn lớn thường cho thấy rằng các giá trị trong tập dữ liệu phân tán rộng hơn so với giá trị trung bình, trong khi độ lệch chuẩn nhỏ hơn cho thấy rằng các giá trị gần nhau hơn với giá trị trung bình.

Trung vị: Trung vị là số nằm giữa trong một tập dữ liệu có các số được sắp xếp. Để xác định giá trị trung vị trong một chuỗi số, trước tiên các số phải được sắp xếp theo thứ tự giá trị từ thấp nhất đến cao nhất hoặc cao nhất đến thấp nhất. Trung vị có thể được sử dụng để xác định giá trị trung bình gần đúng hoặc giá trị trung bình, tuy nhiên không được nhầm lẫn trung vị với giá trị trung bình thực tế. Trung vị được sử dụng thay cho giá trị trung bình khi có các điểm ngoại lai trong chuỗi dữ liệu, các điểm ngoại lai có thể làm lệch giá trị trung bình của các giá trị. Trung vị của một chuỗi ít bị ảnh hưởng bởi các điểm ngoại lai hơn giá trị trung bình.

1. Giá trị lớn nhất: là giá trị lớn nhất trong toàn bộ các giá trị của một tập mẫu.
2. Giá trị nhỏ nhất: là giá trị nhỏ nhất trong toàn bộ các giá trị của một tập mẫu

2.2 Hồi quy tuyến tính bội và phân tích phương sai ANOVA đa nhân tố

2.2.1 Hồi quy tuyến tính đa biến

Hồi quy tuyến tính đa biến là một phương pháp trong thống kê dùng để dự đoán một biến phụ thuộc dựa trên nhiều biến độc lập. Trong hồi quy tuyến tính đa biến, chúng ta giả định rằng mối quan hệ giữa biến phụ thuộc và các biến độc lập là tuyến tính.

Công thức của mô hình hồi quy tuyến tính đa biến có thể được biểu diễn như sau:

$Y = \beta_0 + \beta_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon$ (1)

Trong đó:

- Y là biến phụ thuộc.
- $X_1, X_2, ..., X_n$ là các biến giải thích hay các biến độc lập.
- $\beta_0, \beta_1, \beta_2, ..., \beta_n$ là các tham số.
- Mục tiêu của hồi quy đa biến là đi ước lượng giá trị tối ưu của các hệ số $\beta_0, \beta_1, \beta_2, ..., \beta_n$ sao cho mô hình có khả năng phản ánh quan hệ giữa các biến tốt nhất.

Các phương pháp giải mô hình hồi quy tuyến tính đa biến bao gồm:

- Phương pháp bình phương cực tiểu.
- Phương pháp sử dụng các công cụ tính toán như R, Excel.

2.2.2 Phân tích phương sai ANOVA đa nhân tố

Phân tích phương sai ANOVA (Analysis of Variance) đa nhân tố là một phương pháp thống kê được sử dụng để phân tích sự khác biệt trong trung bình của nhiều nhóm theo nhiều biến độc lập đồng thời. Nó được sử dụng khi có ít nhất hai biến độc lập và một biến phụ thuộc là liên tục. ANOVA đa nhân tố giúp xác định xem có sự ảnh hưởng nào của các biến độc lập đến biến phụ thuộc không và xác định mức độ ảnh hưởng đó là bao nhiêu. Phân tích phương sai đa yếu tố giúp chúng ta đưa thêm yếu tố nguyên nhân vào phân tích làm cho kết quả nghiên cứu càng có giá trị.

III. Tiền xử lí dữ liệu

3.1 Đọc dữ liệu

Dùng câu lệnh sau để đọc dữ liệu từ file All GPUs.csv, rồi xem kết quả 5 dòng đầu tiên của bộ dữ liệu.

```
> data_GPU = read.csv("C:/Users/Admin/Downloads/All_GPUs.csv")
> head(data_GPU,5)
  Architecture Best_Resolution Boost_Clock Core_Speed DVI_Connection Dedicated Direct_X DisplayPort_Connection HDMI_Connection Integrated L2_Cache
1  Tesla G92b                                738 MHz              2       Yes  DX 10.0                     NA              0          No      0KB
2     R600 XT       1366 x 768                   \n-              2       Yes     DX 10                     NA              0          No      0KB
3    R600 PRO       1366 x 768                   \n-              2       Yes     DX 10                     NA              0          No      0KB
4       RV630       1024 x 768                   \n-              2       Yes     DX 10                     NA              0          No      0KB
5       RV630       1024 x 768                   \n-              2       Yes     DX 10                     NA              0          No      0KB
  Manufacturer Max_Power   Memory Memory_Bandwidth Memory_Bus Memory_Speed Memory_Type                          Name Notebook_GPU Open_GL
1       Nvidia 141 Watts 1024 MB          64GB/sec   256 Bit      1000 MHz       GDDR3                GeForce GTS 150          No      3.3
2          AMD 215 Watts  512 MB         106GB/sec   512 Bit       828 MHz       GDDR3          Radeon HD 2900 XT 512MB          No      3.1
3          AMD 200 Watts  512 MB        51.2GB/sec   256 Bit       800 MHz       GDDR3              Radeon HD 2900 Pro           No      3.1
4          AMD           256 MB        36.8GB/sec   128 Bit      1150 MHz       GDDR4 Radeon HD 2600 XT Diamond Edition          No      3.3
5          AMD  45 Watts  256 MB        22.4GB/sec   128 Bit       700 MHz       GDDR3               Radeon HD 2600 XT           No      3.1
                PSU  Pixel_Rate Power_Connector Process ROPs   Release_Date Release_Price Resolution_WxH SLI_Crossfire Shader TMUs Texture_Rate
1 450 Watt & 38 Amps 12 GPixel/s            None     55nm   16 \n01-Mar-2009                2560x1600           Yes      4   64  47 GTexel/s
2 550 Watt & 35 Amps 12 GPixel/s            None     80nm   16 \n14-May-2007                2560x1600           Yes      4   16  12 GTexel/s
3 550 Watt & 35 Amps 10 GPixel/s            None     80nm   16 \n07-Dec-2007                2560x1600           Yes      4   16  10 GTexel/s
4                    3 GPixel/s            None     65nm    4 \n01-Jul-2007                2560x1600           Yes      4    8   7 GTexel/s
5 400 Watt & 25 Amps  3 GPixel/s            None     65nm    4 \n28-Jun-2007                2560x1600           Yes      4    8   6 GTexel/s
  VGA_Connection
1              0
2              0
3              0
4              0
5              0
```

Hình 3: 5 dòng đầu tiên của dữ liệu

3.2 Làm sạch dữ liệu

Để tạo một bộ dữ liệu mới chỉ chứa các cột cần thiết như Max_Power, Core_Speed, L2_Cache, Memory, Memory_Bandwidth, Memory_Bus, Memory_Speed, Pixel_Rate từ bộ dữ liệu gốc, ta sử dụng cú pháp sau.

```
> new_data <- data_GPU[,c("Max_Power", "Core_Speed", "L2_Cache", "Memory", "Memory_Bandwidth", "Memory_Bus", "Memory_Speed","Pixel_Rate")]
> head(new_data,10)
   Max_Power Core_Speed L2_Cache   Memory Memory_Bandwidth Memory_Bus Memory_Speed  Pixel_Rate
1  141 Watts    738 MHz      0KB 1024 MB          64GB/sec   256 Bit      1000 MHz 12 GPixel/s
2  215 Watts        \n-      0KB  512 MB         106GB/sec   512 Bit       828 MHz 12 GPixel/s
3  200 Watts        \n-      0KB  512 MB        51.2GB/sec   256 Bit       800 MHz 10 GPixel/s
4                   \n-      0KB  256 MB        36.8GB/sec   128 Bit      1150 MHz  3 GPixel/s
5   45 Watts        \n-      0KB  256 MB        22.4GB/sec   128 Bit       700 MHz  3 GPixel/s
6   50 Watts        \n-      0KB  256 MB        35.2GB/sec   128 Bit      1100 MHz  3 GPixel/s
7  190 Watts    870 MHz      0KB 2048 MB       134.4GB/sec   256 Bit      1050 MHz 14 GPixel/s
8  150 Watts        \n-      0KB  256 MB        51.2GB/sec   256 Bit       800 MHz  7 GPixel/s
9  150 Watts        \n-      0KB 2048 MB         160GB/sec   256 Bit      1250 MHz 25 GPixel/s
10  32 Watts        \n-      0KB   64 MB         2.9GB/sec    64 Bit       366 MHz
```

Hình 4: Câu lệnh tạo bộ dữ liệu và 10 dòng đầu của bộ dữ liệu mới

Lúc này, dữ liệu vẫn còn đang ở dạng chuỗi, ta sẽ tiến hành xóa bỏ đơn vị và chuyển dữ liệu về dạng số để làm việc.

```
> new_data <- new_data %>%
+     mutate(
+         Max_Power = as.numeric(gsub("[^0-9.]", "", Max_Power)),
+         Core_Speed = as.numeric(gsub("[^0-9.]" , "", Core_Speed)),
+         L2_Cache = as.numeric(gsub("[^0-9.]", "", L2_Cache)),
+         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)),
+         Pixel_Rate = as.numeric(gsub("[^0-9.]", "", Pixel_Rate))
+     )
>
> head(new_data,10)
   Max_Power Core_Speed L2_Cache Memory Memory_Bandwidth Memory_Bus Memory_Speed Pixel_Rate
1        141        738        0   1024             64.0        256         1000         12
2        215         NA        0    512            106.0        512          828         12
3        200         NA        0    512             51.2        256          800         10
4         NA         NA        0    256             36.8        128         1150          3
5         45         NA        0    256             22.4        128          700          3
6         50         NA        0    256             35.2        128         1100          3
7        190        870        0   2048            134.4        256         1050         14
8        150         NA        0    256             51.2        256          800          7
9        150         NA        0   2048            160.0        256         1250         25
10        32         NA        0     64              2.9         64          366         NA
```

Hình 5: Kết quả 10 dòng đầu tiên của bộ dữ liệu mới đã loại bỏ đơn vị và chỉnh sửa định dạng

Tiếp theo, ta sẽ tiến hành thống kê số lượng dữ liệu của từng biến.

```
> head(new_data,10)
   Max_Power Core_Speed L2_Cache Memory Memory_Bandwidth Memory_Bus Memory_Speed Pixel_Rate
1        141        738        0   1024             64.0        256         1000         12
2        215         NA        0    512            106.0        512          828         12
3        200         NA        0    512             51.2        256          800         10
4         NA         NA        0    256             36.8        128         1150          3
5         45         NA        0    256             22.4        128          700          3
6         50         NA        0    256             35.2        128         1100          3
7        190        870        0   2048            134.4        256         1050         14
8        150         NA        0    256             51.2        256          800          7
9        150         NA        0   2048            160.0        256         1250         25
10        32         NA        0     64              2.9         64          366         NA
```

Hình 6: Thống kê số dữ liệu khuyết của từng biến

Dựa vào quan sát, nhóm đã xử lý dữ liệu thiếu trong bộ dữ liệu bằng cách thay thế tất cả các giá trị thiếu của các biến bằng giá trị trung bình tương ứng.

```
mean_values <- sapply(new_data, function(x) mean(x,na.rm=TRUE))
mean_values
new_data <- new_data %>%
  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),
    L2_Cache=ifelse(is.na(L2_Cache),mean_values["L2_Cache"],L2_Cache),
    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),
    Pixel_Rate=ifelse(is.na(Pixel_Rate),mean_values["Pixel_Rate"],Pixel_Rate),
    )
```

```
> head(new_data,10)
   Max_Power Core_Speed L2_Cache Memory Memory_Bandwidth Memory_Bus Memory_Speed Pixel_Rate
1   141.0000   738.0000        0   1024             64.0        256         1000   12.00000
2   215.0000   946.8939        0    512            106.0        512          828   12.00000
3   200.0000   946.8939        0    512             51.2        256          800   10.00000
4   125.5987   946.8939        0    256             36.8        128         1150    3.00000
5    45.0000   946.8939        0    256             22.4        128          700    3.00000
6    50.0000   946.8939        0    256             35.2        128         1100    3.00000
7   190.0000   870.0000        0   2048            134.4        256         1050   14.00000
8   150.0000   946.8939        0    256             51.2        256          800    7.00000
9   150.0000   946.8939        0   2048            160.0        256         1250   25.00000
10   32.0000   946.8939        0     64              2.9         64          366   34.96541
```

Hình 7: Kết quả 10 dòng đầu tiên của bộ dữ liệu sau khi được xử lý

3.3 Xử lý các giá trị ngoại lai

Sau khi quan sát bộ dữ liệu, nhóm nhận thấy có nhiều giá trị ngoại lai ảnh hưởng đến chất lượng của mô hình. Để xử lý vấn đề này, nhóm đã áp dụng phương pháp chặn trên hoặc chặn dưới để giới hạn các giá trị ngoại lai. Khi một giá trị vượt quá ngưỡng cao hoặc thấp, nhóm sẽ điều chỉnh nó về giá trị lớn nhất hoặc nhỏ nhất mà được coi là bình thường.

```r
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)
}

new_data$Max_Power <- BoxplotOutlierClipper(whisker_coeff = 1.5, X = new_data$Max_Power)
new_data$Core_Speed <- BoxplotOutlierClipper(whisker_coeff = 1.5, X = new_data$Core_Speed)
new_data$L2_Cache <- BoxplotOutlierClipper(whisker_coeff = 1.5, X = new_data$L2_Cache)
new_data$Memory <- BoxplotOutlierClipper(whisker_coeff = 1.5, X = new_data$Memory)
new_data$Memory_Bandwidth <- BoxplotOutlierClipper(whisker_coeff = 1.5, X = new_data$Memory_Bandwidth)
new_data$Memory_Bus <- BoxplotOutlierClipper(whisker_coeff = 1.5, X = new_data$Memory_Bus)
new_data$Memory_Speed <- BoxplotOutlierClipper(whisker_coeff = 1.5, X = new_data$Memory_Speed)
new_data$Pixel_Rate <- BoxplotOutlierClipper(whisker_coeff = 1.5, X = new_data$Pixel_Rate)

head(new_data,10)
```

	Max_Power	Core_Speed	L2_Cache	Memory	Memory_Bandwidth	Memory_Bus	Memory_Speed	Pixel_Rate
	141.0000	738.0000	0	1024	64.0	256	1000	12.00000
	215.0000	946.8939	0	512	106.0	448	828	12.00000
	200.0000	946.8939	0	512	51.2	256	800	10.00000
	125.5987	946.8939	0	256	36.8	128	1150	3.00000
	45.0000	946.8939	0	256	22.4	128	700	3.00000
	50.0000	946.8939	0	256	35.2	128	1100	3.00000
	190.0000	870.0000	0	2048	134.4	256	1050	14.00000
	150.0000	946.8939	0	256	51.2	256	800	7.00000
	150.0000	946.8939	0	2048	160.0	256	1250	25.00000
0	32.0000	946.8939	0	64	2.9	64	366	34.96541

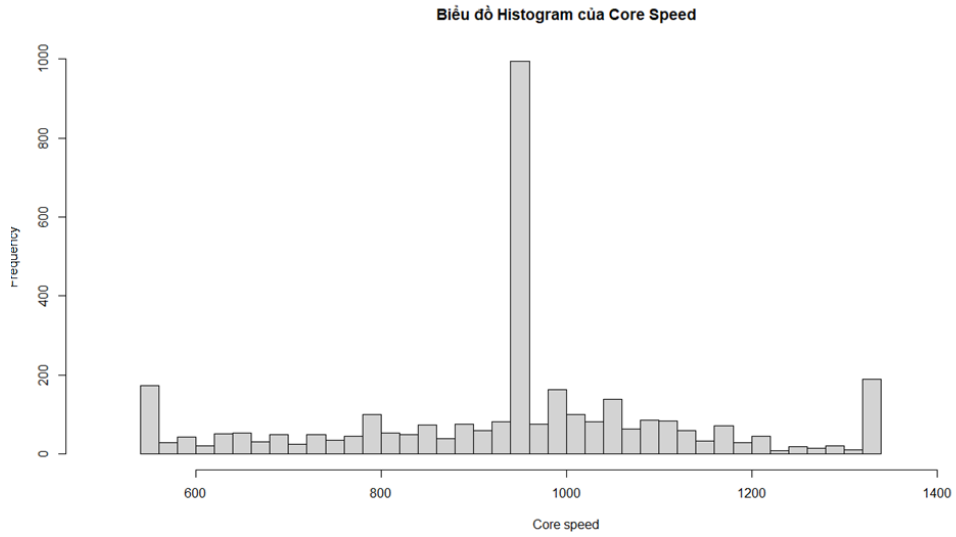
Hình 8: Đoạn code R xử lý các giá trị ngoại lai và kết quả 10 dòng đầu tiên của bộ dữ liệu sau khi xử lý các giá trị ngoại lai

IV. Thống kê mô tả

4.1 Biểu đồ Histogram của biến Core_Speed

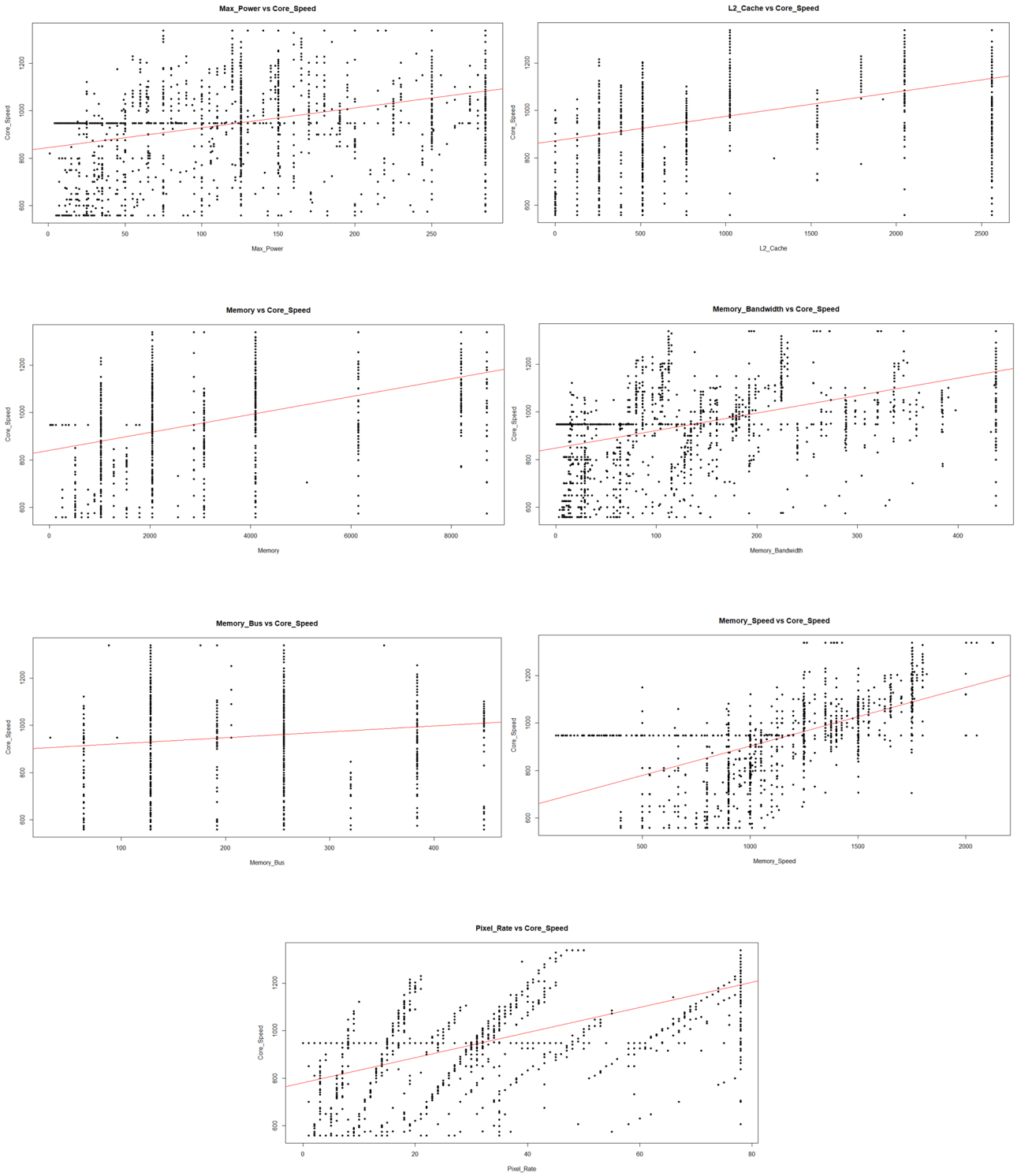
Dưới đây là biểu đồ Histogram thể hiện sự phân bố cho biến Core_Speed.

```
hist(new_data$Core_Speed,main="Core_Speed",xlab="", breaks = 100)
```



Hình 9: Biểu đồ Histogram cho biến Core_Speed

4.2 Biểu đồ Scatterplot



Nhận xét: Từ các đồ thị trên, ta thấy rằng đa số các biến đều có quan hệ tuyến tính yếu với biến Core_Speed, ngoại trừ biến Memory_Speed.

4.3 Tính toán các giá trị thống kê mô tả

Sau đây nhóm sẽ đi tính toán các giá trị thống kê mô tả của các biến, bao gồm: Trung bình, độ lệch chuẩn, trung vị, các điểm phân vị và giá trị lớn nhất, giá trị nhỏ nhất.

Đoạn code R thực hiện tính toán các giá trị thống kê mô tả:

```
> mean <- apply(new_data,2,mean)
> sd <- apply(new_data,2,sd)
> median <- apply(new_data,2,median)
> Q1 <- apply(new_data,2,quantile,probs=0.25)
> Q3 <- apply(new_data,2,quantile,probs=0.75)
> min <- apply(new_data,2,min)
> max <- apply(new_data,2,max)
> t(data.frame(mean,sd,median,Q1,Q3,min,max))
       Max_Power Core_Speed  L2_Cache   Memory
mean   120.00710   945.4024  716.5452 2767.768
sd      74.65553   187.9565  814.0761 2212.579
median 125.59871   946.8939  512.0000 2048.000
Q1      60.00000   851.0000    0.0000 1024.000
Q3     150.00000  1046.0000 1024.0000 4096.000
min      1.00000   558.5000    0.0000   16.000
max    285.00000  1338.5000 2560.0000 8704.000
       Memory_Bandwidth Memory_Bus Memory_Speed Pixel_Rate
mean           132.7456   192.6917    1175.6556   31.12689
sd             115.5229   101.0420     439.5549   22.52131
median         112.0000   128.0000    1175.6556   31.00000
Q1              28.8000   128.0000     840.2500   13.00000
Q3             192.3000   256.0000    1502.0000   39.00000
min              1.0000    32.0000     100.0000    0.00000
max            437.5500   448.0000    2127.0000   78.00000
```

Hình 10: Bảng giá trị thống kê mô tả của các biến

V. Thống kê suy diễn

5.1 Xây dựng mô hình hồi quy tuyến tính đa biến để đánh giá các nhân tố ảnh hưởng đến tốc độ nhân GPUs

5.1.1 Xây dựng mô hình hồi quy tuyến tính đa biến

Chúng ta sẽ bắt đầu xây dựng mô hình hồi quy tuyến tính đa biến bằng cách thực hiện các bước sau:

Biến phụ thuộc: Core_Speed

Biến độc lập: Max_Power, L2_Cache, Memory, Memory_Bandwidth, Memory_Bus, Memory_Speed, Pixel_Rate

Ta có mô hình:

$$\text{Core\_Speed} = \beta_0 + \beta_1 \times \text{Max\_Power} + \beta_2 \times \text{L2\_Cache} + \beta_3 \times \text{Memory} + \beta_4 \times \text{Memory\_Bandwidth} + \beta_5 \times \text{Memory\_Bus} + \beta_6 \times \text{Memory\_Speed} + \beta_7 \times \text{Pixel\_Rate} + \epsilon$$

Ta tiến hành ước lượng các hệ số β_i với $0 \leq i \leq 7$ dựa vào mô hình trên.

```
model1=lm(Core_Speed ~ Max_Power+ L2_Cache + Memory + Memory_Bandwidth + Memory_Bus + Memory_Speed + Pixel_Rate, data = new_data)
summary(model1)
```

Hình 11: Đoạn code R xây dựng mô hình

```
> model1=lm(Core_Speed ~ Max_Power+ L2_Cache + Memory + Memory_Bandwidth + Memory_Bus + Memory_Speed + Pixel_Rate, data = new_data)
> summary(model1)

Call:
lm(formula = Core_Speed ~ Max_Power + L2_Cache + Memory + Memory_Bandwidth +
    Memory_Bus + Memory_Speed + Pixel_Rate, data = new_data)

Residuals:
    Min      1Q  Median      3Q     Max
-442.08  -73.01   13.72   84.01  356.19

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     694.549650   8.289625  83.785  < 2e-16 ***
Max_Power         0.217726   0.053808   4.046 5.32e-05 ***
L2_Cache          0.007505   0.003854   1.947 0.051596 .
Memory           -0.005720   0.001646  -3.475 0.000518 ***
Memory_Bandwidth -0.235116   0.051858  -4.534 5.99e-06 ***
Memory_Bus       -0.554086   0.034676 -15.979  < 2e-16 ***
Memory_Speed      0.172291   0.006595  26.125  < 2e-16 ***
Pixel_Rate        5.480798   0.160492  34.150  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 125.3 on 3398 degrees of freedom
Multiple R-squared:  0.5566,    Adjusted R-squared:  0.5557
F-statistic: 609.3 on 7 and 3398 DF,  p-value: < 2.2e-16
```

Hình 12: Kết quả thu được sau khi chạy đoạn code R cung cấp trên

Dựa vào bảng số liệu trên, chúng ta thu được các ước lượng cho các hệ số β như sau:

$\beta_0 = 694.549650$

$\beta_1 = 0.217726$

$\beta_2 = 0.007505$

$\beta_3 = -0.005720$

$\beta_4 = -0.235116$

$\beta_5 = -0.554086$

$\beta_6 = 0.172291$

$\beta_7 = 5.480798$

Do đó, phương trình đường thẳng ước lượng được xác định như sau:

$$\text{Core\_Speed} = 694.549650 + 0.217726 \times \text{Max\_Power} + 0.007505 \times \text{L2\_Cache} - 0.005720 \times \text{Memory} - 0.235116 \times \text{Memory\_Bandwidth} - 0.554086 \times \text{Memory\_Bus} + 0.172291 \times \text{Memory\_Speed} + 5.480798 \times \text{Pixel\_Rate}$$

5.1.2 Kiểm định các hệ số hồi quy

Dựa vào kết quả kiểm định giả thuyết, ta nhận thấy:

- Các biến Max_Power, Memory, Memory_Bandwidth, Memory_Bus, Memory_Speed, Pixel_Rate, giá trị p-value của hệ số tương ứng đều nhỏ hơn mức ý nghĩa $\alpha = 0.05$. Nên ta có thể bác bỏ giả thuyết H_0 và chấp nhận giả thuyết H_1. Tức là, hệ số hồi quy của các biến này có ý nghĩa thống kê.
- Biến L2_Cache, giá trị p-value lớn hơn mức ý nghĩa $\alpha = 0.05$. Nên ta không đủ cơ sở để bác bỏ giả thuyết H_0 của hệ số tương ứng với biến này. Tức là, hệ số hồi quy của biến L2_Cache không có ý nghĩa thống kê.

5.1.3 Cải tiến mô hình hồi quy tuyến tính đa biến

Tiếp theo, chúng ta sẽ xây dựng lại một mô hình hồi quy tuyến tính mới bằng cách loại bỏ các biến mà hệ s không có ý nghĩa thống kê. Chúng ta sẽ loại bỏ biến L2_Cache và xây dựng mô hình hồi quy tuyến tính với các biến còn lại: Max_Power, Memory, Memory_Bandwidth, Memory_Bus, Memory_Speed, Pixel_Rate.

Xây dựng mô hình tuyến tính bao gồm:

Biến phụ thuộc: Core_Speed

Biến độc lập: Max_Power, Memory, Memory_Bandwidth, Memory_Bus, Memory_Speed, Pixel_Rate

```
> model2=lm(Core_Speed ~ Max_Power + Memory + Memory_Bandwidth + Memory_Bus + Memory_Speed + Pixel_Rate, data = new_data)
> summary(model2)

Call:
lm(formula = Core_Speed ~ Max_Power + Memory + Memory_Bandwidth +
    Memory_Bus + Memory_Speed + Pixel_Rate, data = new_data)

Residuals:
    Min      1Q  Median      3Q     Max
-443.62  -71.85   12.70   83.76  353.35

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     693.787738   8.283785  83.753  < 2e-16 ***
Max_Power         0.209387   0.053659   3.902 9.72e-05 ***
Memory           -0.005052   0.001611  -3.137  0.00172 **
Memory_Bandwidth -0.213605   0.050689  -4.214 2.57e-05 ***
Memory_Bus       -0.558551   0.034614 -16.136  < 2e-16 ***
Memory_Speed      0.173701   0.006558  26.488  < 2e-16 ***
Pixel_Rate        5.533499   0.158258  34.965  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 125.3 on 3399 degrees of freedom
Multiple R-squared:  0.5561,    Adjusted R-squared:  0.5553
F-statistic: 709.7 on 6 and 3399 DF,  p-value: < 2.2e-16
```

Hình 13: Kết quả sau khi xây dựng được mô hình mới

5.1.4 So sánh hai mô hình

Chúng ta thực hiện kiểm định giả thuyết để so sánh sự hiệu quả của hai mô hình hồi quy tuyến tính đa biến đã được xây dựng.

- Giả thuyết H0: Hai mô hình có độ hiệu quả giống nhau.
- Giả thuyết H1: Hai mô hình có độ hiệu quả khác nhau.

```
150  anova(model1, model2)
```

Hình 14: Câu lệnh R thực hiện việc so sánh hai mô hình

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

Model 1: Core_Speed ~ Max_Power + L2_Cache + Memory + Memory_Bandwidth +
    Memory_Bus + Memory_Speed + Pixel_Rate
Model 2: Core_Speed ~ Max_Power + Memory + Memory_Bandwidth + Memory_Bus +
    Memory_Speed + Pixel_Rate
  Res.Df      RSS Df Sum of Sq      F Pr(>F)
1   3398 53337998
2   3399 53397512 -1    -59515 3.7915 0.0516 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Hình 15: Kết quả so sánh hai mô hình

Dựa vào kết quả kiểm định, chúng ta nhận thấy giá trị p-value $Pr(> F)$ lớn hơn mức ý nghĩa $\alpha = 0.05$. Điều này cho thấy chúng ta không đủ cơ sở để bác bỏ giả thuyết H_0, tức là hai mô hình có độ hiệu quả như nhau.

Tuy nhiên, cần lưu ý rằng mô hình thứ nhất có R2 hiệu chỉnh cao hơn so với mô hình thứ hai. Do đó, dù hai mô hình có độ hiệu quả tương đương, mô hình thứ nhất có khả năng giải thích phương sai của dữ liệu tốt hơn. Dựa trên điều này, ta quyết định chọn mô hình thứ nhất làm mô hình để dự báo tốc độ nhân dựa trên các yếu tố liên quan khác.

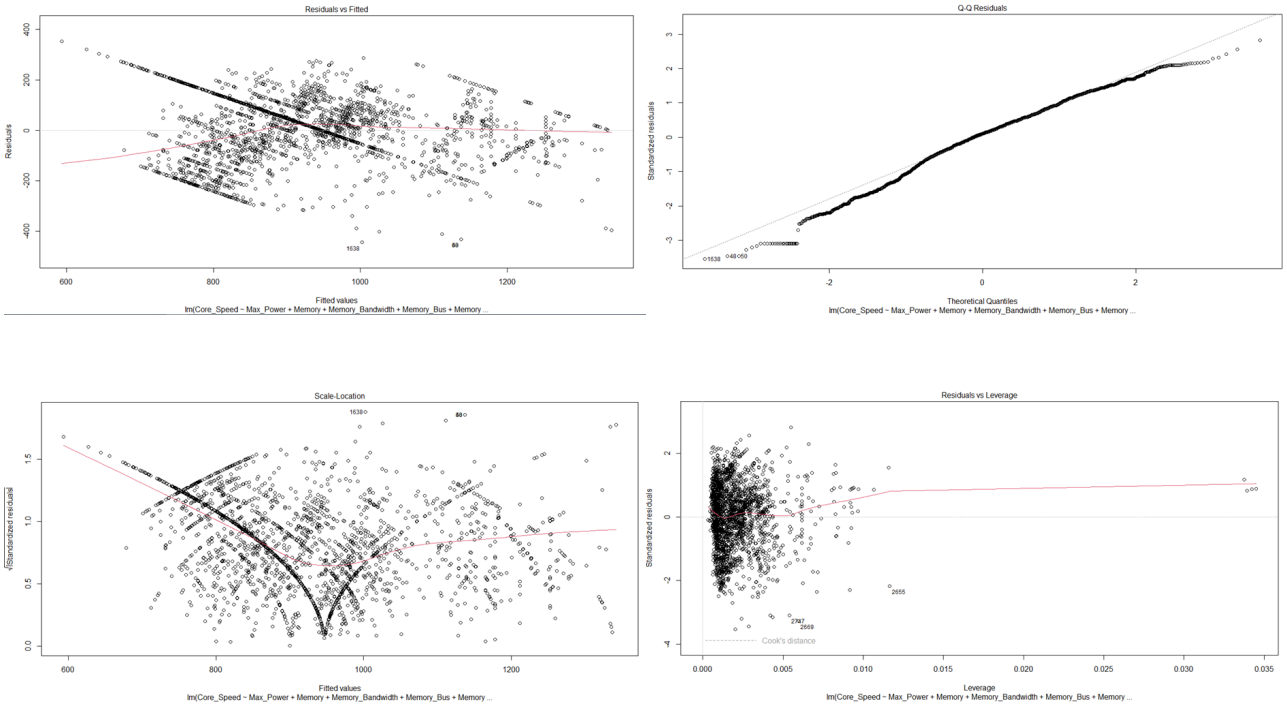
5.1.5 Kiểm tra các giả định của mô hình

Ta nhắc lại các giả định của mô hình hồi quy truyến tính, giả sử rằng, ta có n quan sát của mô hình, khi đó:

- Tính tuyến tính của dữ liệu, mối quan hệ giữa biến dự báo X và biến phụ thuộc Y được giả sử là tuyến tính.
- Các sai số ϵ_1, ϵ_2, ϵ_3, ..., ϵ_n có phân phối chuẩn $N(0, \sigma^2)$.
- Các sai số ngẫu nhiên ϵ_1, ϵ_2, ϵ_3, ..., ϵ_n thì độc lập với nhau.

Ta thực hiện phân tích thặng dư để kiểm tra các giả định của mô hình. Sử dụng câu lệnh trong R như dưới đây:

```
162 par(mfrow=c(2,2))
163 plot(model2)
```



Việc đánh giá đồ thị thực hiện phân tích thặng dư của mô hình hồi quy tuyến tính là vô cùng quan trọng để xác nhận tính chính xác và đáng tin cậy của mô hình.

- Trên đồ thị Residuals vs Fitted, chúng ta thấy rằng đường màu đỏ không phải là một đường thẳng tuyến tính, cho thấy tính tuyến tính chưa được đạt. Tuy nhiên, các điểm dữ liệu gần với đường residuals = 0 cho thấy giả định về kì vọng của sai số có thể chấp nhận được, tuy nhiên, phương sai của sai số không ổn định, không tuân theo giả định của mô hình.
- - Trên đồ thị Q-Q Residuals, chúng ta thấy hầu hết các điểm sai số nằm gần đường thẳng kì vọng của phân phối chuẩn, ngụ ý rằng giả định về phân phối chuẩn của sai số có thể chấp nhận.
- Đối với đồ thị Scale-Location, chúng ta thấy đường màu đỏ không phải là một đường thẳng ngang và các điểm không phân tán đều xung quanh đường màu đỏ, ngụ ý rằng giả định về phương sai không đổi của sai số không được đáp ứng.
- Cuối cùng, trên đồ thị về các điểm ảnh hưởng lớn, không xuất hiện điểm nào có giá trị Cook's distance cao, ngụ ý rằng không có điểm dữ liệu nào ảnh hưởng lớn đến mô hình.
- Tổng cộng, mặc dù mô hình hồi quy tuyến tính không hoàn toàn thỏa mãn các giả định, tuy vẫn có thể sử dụng để dự báo tốc độ nhân của GPU trong nghiên cứu. Cần phải xem xét kỹ lưỡng và áp dụng phương pháp điều chỉnh hoặc phân tích sâu hơn để cải thiện độ chính xác của mô hình.

5.1.6 Dự báo tốc độ nhân GPU

Ta tiến hành thử dự đoán tốc độ nhân GPU bằng mô hình hồi quy tuyến tính đa biến đã thiết lập ở trên với giá trị của các biến phụ thuộc được cho như sau: Max_Power = 285, Memory = 1792, Memory_Bandwidth = 225.8, Memory_Speed = 448, Pixel_Rate = 32.

```
> X <- data.frame("Max_Power" = 285, "L2_Cache" = 768, "Memory" = 1792, "Memory_Bandwidth" = 225.8,"Memory_Bus"=448, "Memory_Speed" = 1008, "Pixel_Rate" = 32)
> predictX <- predict(model1,X,interval="confidence", level=0.95)
> head(predictX)
       fit      lwr      upr
1 799.8512 784.4841 815.2183
```

Hình 16: Đoạn code R thực hiện dự báo

Vậy tốc độ nhân dự báo là 799.8512 MHz với khoảng tin cậy là (784,4841; 815,2183).

5.2 Xây dựng mô hình ANOVA đa nhân tố để đánh giá sự ảnh hưởng của các yếu tố định tính lên tốc độ nhân GPU

5.2.1 Xây dựng mô hình ANOVA

Ta tiến hành xây dựng mô hình ANOVA hai nhân tố để đánh giá sự ảnh hưởng của các yếu tố định tính liên quan đến tốc độ nhân của GPU. Chúng tôi quyết định sử dụng thêm hai biến định tính là Manufacturer và SLI_Crossfire từ bộ dữ liệu gốc và phân tích sự tác động của chúng tôi đối với tốc độ nhân của GPU(Core_Speed).

```
data_anova <- new_data
data_anova$Manufacturer <-  data_GPU$Manufacturer
data_anova$SLI_Crossfire <- data_GPU$SLI_Crossfire
anova_model <- aov(Core_Speed ~ Manufacturer * SLI_Crossfire, data = data_anova)
summary(anova_model)
```

Hình 17: Đoạn code R thực hiện phân tích

5.2.2 Phân tích kết quả thu được

Sau khi xây dựng mô hình ANOVA và chạy lệnh phân tích mô hình, ta thu được kết quả như dưới đây, ta sẽ tiến hành phân tích kết quả thu được này.

```
> data_anova <- new_data
> data_anova$Manufacturer <-  data_GPU$Manufacturer
> data_anova$SLI_Crossfire <- data_GPU$SLI_Crossfire
> anova_model <- aov(Core_Speed ~ Manufacturer * SLI_Crossfire, data = data_anova)
> summary(anova_model)
                            Df    Sum Sq Mean Sq F value  Pr(>F)
Manufacturer                 3  12121551 4040517 127.905 < 2e-16 ***
SLI_Crossfire                1    659136  659136  20.865 5.1e-06 ***
Manufacturer:SLI_Crossfire   2    135973   67986   2.152   0.116
Residuals                 3399 107374008   31590
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Hình 18: Kết quả phân tích

Từ bảng trên, ta thực hiện việc kiểm định:

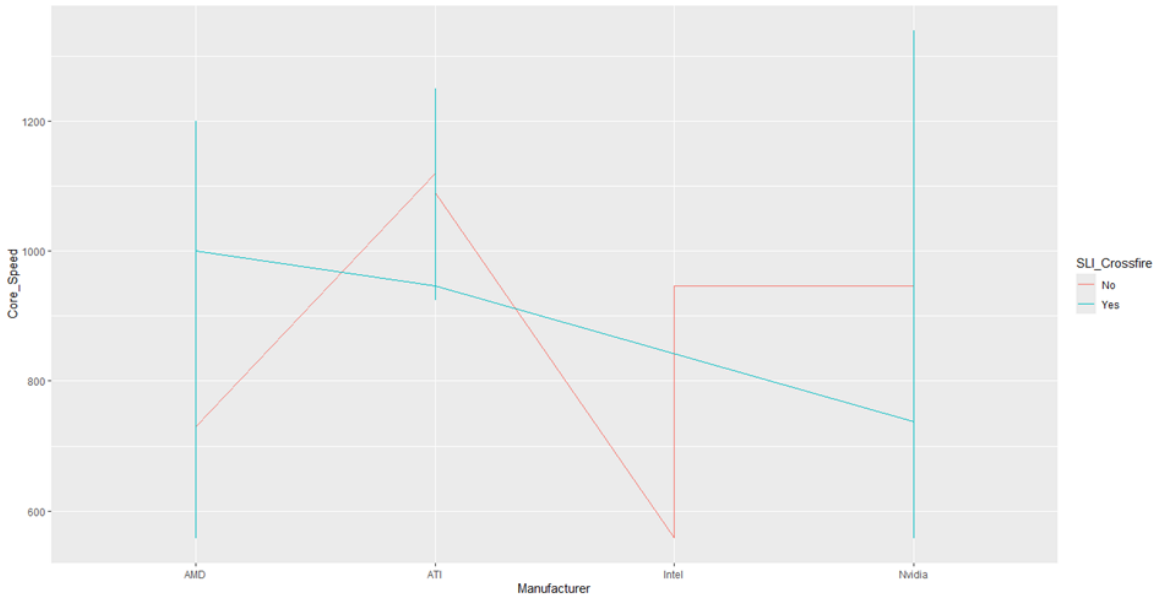
a. Đối với biến Manufacturer, chúng ta thực hiện kiểm định giả thuyết để xem liệu có sự khác biệt về tốc độ nhân GPU trung bình giữa các hãng hay không.

Giả thuyết H_0**:** Tốc độ nhân trung bình của GPU giữa các hãng là như nhau.
Giả thuyết H_1**:** Có ít nhất hai hãng có tốc độ nhân trung bình của GPU khác nhau.
Dựa vào giá trị p-value $Pr(> F)$ ứng với biến Manufacturer, chúng ta thấy rằng giá trị p-value nhỏ hơn mức ý nghĩa $\alpha = 0.05$. Do đó, chúng ta bác bỏ giả thuyết H_0 và chấp nhận giả thuyết H_1. Điều này cho thấy có ít nhất hai hãng có tốc độ nhân trung bình của GPU khác nhau.
Đối với biến SLI_Crossfire, ta tiến hành kiểm định giả thuyết để xem liệu có sự khác biệt về tốc độ nhân trung bình giữa GPU có SLI_Crossfire và GPU không có SLI_Crossfire hay không.
Giả thuyết H_0**:** Tốc độ nhân trung bình của GPU giữa GPU có SLI_Crossfire và GPU không có SLI_Crossfire là như nhau.

Giả thuyết H_1: Tốc độ nhân trung bình của GPU giữa GPU có SLI_Crossfire và GPU không có SLI_Crossfire là khác nhau.
Dựa vào giá trị p-value $Pr(> F)$ ứng với biến SLI_Crossfire, ta thấy rằng giá trị p-value nhỏ hơn mức ý nghĩa $\alpha = 0.05$. Do đó, chúng ta bác bỏ giả thuyết H_0 và chấp nhận giả thuyết H_1. Điều này cho thấy có sự khác biệt về tốc độ nhân giữa GPU có SLI_Crosfire và GPU không có SLI_Crossfire.

b. Tiếp theo, chúng ta sẽ xem xét sự tương tác giữa hai biến **Manufacturer** và **SLI_Crossfire** đối với tốc độ nhân **Core_Speed**. Chúng ta tiến hành kiểm định giả thuyết sau:

Giả thuyết H_0: Không có sự tương tác giữa hai nhân tố **Manufacturer** và **SLI_Crossfire**.
Giả thuyết H_1: Có sự tương tác giữa hai nhân tố **Manufacturer** và **SLI_Crossfire**.
Dựa vào giá trị p-value $Pr(> F)$ tương ứng, chúng ta thấy rằng giá trị p-value lớn hơn mức ý nghĩa $\alpha = 0.05$. Vì vậy, chúng ta không có đủ cơ sở để bác bỏ giả thuyết H_0. Từ đó, chúng ta kết luận rằng không có sự tương tác giữa hai nhân tố **Manufacturer** và **SLI_Crossfire**.



Hình 19: Biểu đồ thể hiện sự tương tác giữa Manufacturer và SLI_Crosfire

Nhận xét: Từ biểu đồ, ta nhận thấy rằng tốc độ nhân trung bình của GPU khi có và không có SLI_Crossfire ở các hãng là gần bằng nhau, do đó không có sự tương tác giữa hai nhân tố Manufacturer và SLI_Crossfire.

5.2.3 Thực hiện phân tích sâu ANOVA

Sau khi phân tích các kết quả thu được ở trên, ta nhận thấy rằng, tốc độ nhân của GPU(Core_Speed) là khác nhau giữa các phân loại của biến Manufacturer và SLI_Crossfire. Trong phần này, ta sẽ thực hiện so sánh bội để làm rõ sự khác biệt đó.

```
TukeyHSD(anova_model)
```

Hình 20: Câu lệnh so sánh bội trong R

Sau đây ta sẽ lần lượt phân tích các kết quả của câu lệnh so sánh bội trên.

a. So sánh bội với biến Manufacturer:

```
$Manufacturer
                  diff        lwr        upr     p adj
ATI-AMD       143.89779   94.63409  193.16148 0.0000000
Intel-AMD    -135.26505 -166.57170 -103.95839 0.0000000
Nvidia-AMD     70.12115   53.44187   86.80044 0.0000000
Intel-ATI    -279.16284 -334.75142 -223.57425 0.0000000
Nvidia-ATI    -73.77664 -122.64567  -24.90760 0.0006147
Nvidia-Intel  205.38620  174.70433  236.06807 0.0000000
```

Xét cặp AMD-ATI, chúng ta sẽ thực hiện kiểm định giả thuyết để xem liệu có sự khác biệt về tốc độ nhân trung bình giữa hai hãng ATI và AMD hay không.
Giả thuyết H_0: Không có sự khác biệt về tốc độ nhân trung bình giữa hai hãng ATI và AMD.
Giả thuyết H_1: Có sự khác biệt về tốc độ nhân trung bình giữa hai hãng ATI và AMD.
Quan sát giá trị p-value (p_{adj}), chúng ta thấy giá trị p-value tương ứng là 0, nhỏ hơn mức ý nghĩa $\alpha = 0.05$. Do đó, chúng ta bác bỏ giả thuyết H_0 và chấp nhận giả thuyết H_1. Điều này cho thấy có sự khác biệt về tốc độ nhân giữa hai hãng ATI và AMD.
Ngoài ra, quan sát giá trị diff tương ứng, ta thấy diff có giá trị dương (143.89779), và giá trị diff này chính là hiệu tốc độ nhân trung bình của mẫu giữa hãng ATI và AMD. Từ đó, ta có thể kết luận rằng GPU do hãng ATI sản xuất có tốc độ nhân trung bình lớn hơn so với hãng AMD.
Tiếp tục thực hiện kiểm định trên các cặp còn lại, chúng ta thu được các kết luận sau:

- Tốc độ nhân GPU trung bình của hãng Intel nhỏ hơn so với hãng AMD.
- Tốc độ nhân GPU trung bình của hãng Nvidia lớn hơn so với hãng AMD.
- Tốc độ nhân GPU trung bình của hãng Intel nhỏ hơn so với hãng ATI.
- Tốc độ nhân GPU trung bình của hãng Nvidia nhỏ hơn so với hãng ATI.
- Tốc độ nhân GPU trung bình của hãng Nvidia lớn hơn so với hãng Intel.

b. So sánh bội với biến SLI_Crossfire:

```
$SLI_Crossfire
            diff      lwr      upr    p adj
Yes-No 26.123 14.07988 38.16611 2.17e-05
```

Xét hai phân loại của biến SLI_Crossfire, chúng ta sẽ thực hiện kiểm định giả thuyết để xem liệu có sự khác biệt về tốc độ nhân trung bình giữa GPU có SLI_Crossfire và GPU không có SLI_Crossfire hay không.
Giả thuyết H_0: Không có sự khác biệt về tốc độ nhân trung bình giữa GPU có SLI_Crossfire và GPU không có SLI_Crossfire.
Giả thuyết H_1: Có sự khác biệt về tốc độ nhân trung bình giữa GPU có SLI_Crossfire và GPU không có SLI_Crossfire.
Quan sát giá trị p-value, chúng ta thấy giá trị p-value tương ứng nhỏ hơn mức ý nghĩa $\alpha = 0.05$. Vì vậy, chúng ta bác bỏ giả thuyết H_0 và chấp nhận giả thuyết H_1. Điều này cho thấy có sự khác biệt về tốc độ nhân trung bình giữa GPU có SLI_Crossfire và GPU không có SLI_Crossfire.
Ngoài ra, quan sát giá trị diff tương ứng, ta thấy diff có giá trị dương (26.213), và giá trị diff này chính là hiệu tốc độ nhân trung bình của mẫu giữa các GPU có SLI_Crossfire và các GPU không có SLI_Crossfire. Từ đó, ta có thể kết luận rằng tốc độ nhân trung bình của các GPU có SLI_Crossfire lớn hơn so với tốc độ nhân trung bình của các GPU không có SLI_Crossfire
Nhận xét: Từ kết quả của việc so sánh bội, ta kết luận về sự so sánh tốc độ nhân GPU của các hãng từ cao đến thấp như sau: ATI -> Nvidia -> AMD -> Intel. Các GPU có SLI_Crossfire thì sẽ có tốc độ nhân trung bình lớn hơn so với các GPU không có SLI_Croosfire.

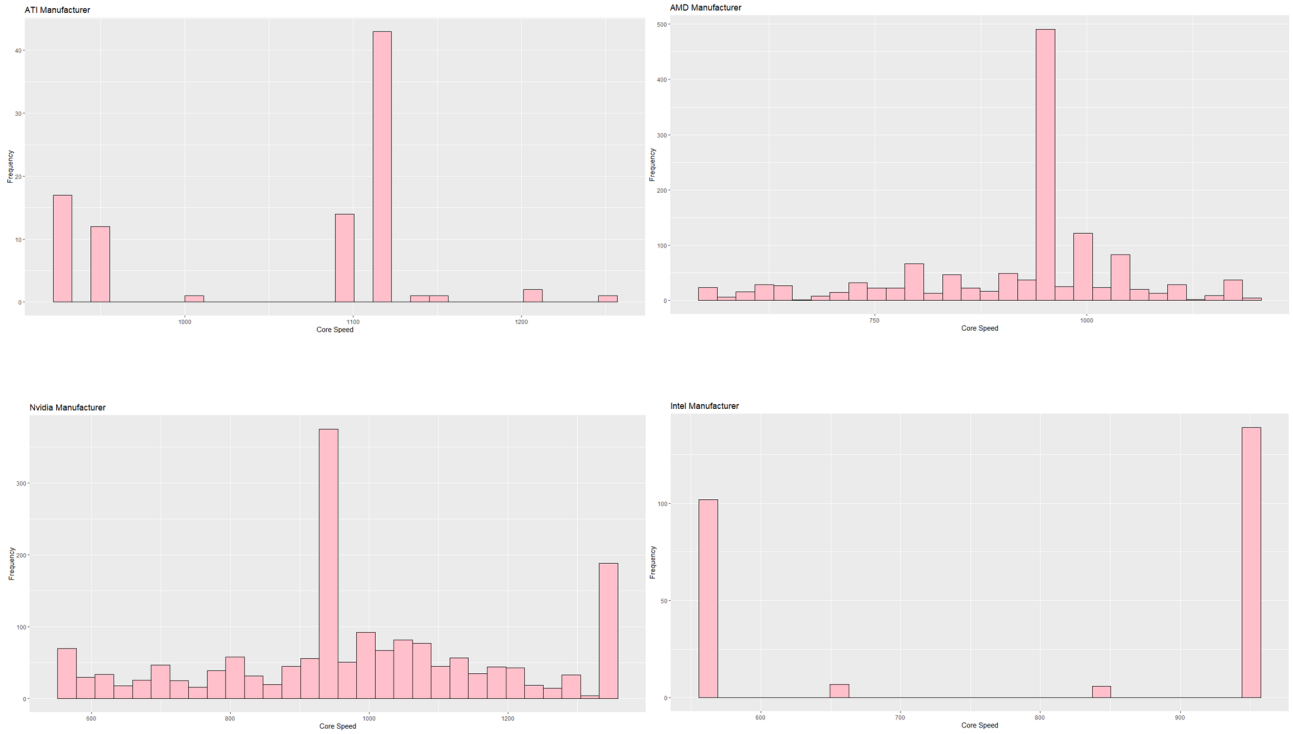
5.2.4 Kiểm tra các giả định của mô hình

Điều kiện về phân phối chuẩn: Tốc độ nhân GPU(Core_Speed) ở 4 hãng sản xuất, của hai nhóm có và không có SLI_Crossfire có phân phối chuẩn.

Điều kiện về phương sai: Phương sai tốc độ nhân(Core_Speed) ở 4 hãng, của hai nhóm có và không có SLI_Crossfire là bằng nhau

Ta sẽ sử dụng đồ thị Histogram để kiểm tra điều kiện về phân phối chuẩn như dưới đây:

1. Đối với biến phân loại Manufacturer:



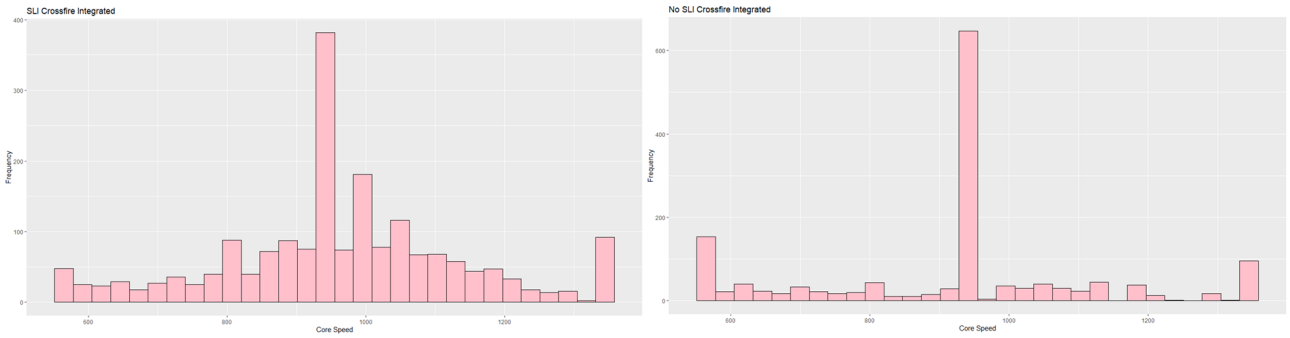

```
install.packages("car")
library(car)
data_anova$Manufacturer <- factor(data_anova$Manufacturer)
leveneTest(Core_Speed ~ Manufacturer, data=data_anova)
```

```
> data_anova$Manufacturer <- factor(data_anova$Manufacturer)
> leveneTest(Core_Speed ~ Manufacturer, data=data_anova)
Levene's Test for Homogeneity of Variance (center = median)
       Df F value    Pr(>F)
group   3  94.672 < 2.2e-16 ***
      3402
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Hình 21: Kết quả thu được

Ta kiểm định các giả thuyết sau đây:
Giả thuyết H_0: Phương sai của tốc độ nhân (*Core_Speed*) ở 4 hãng là bằng nhau.
Giả thuyết H_1: Có ít nhất hai hãng có phương sai tốc độ nhân (*Core_Speed*) khác nhau.
Dựa vào kết quả thu được từ R, quan sát giá trị cột p-value (Pr(>F)) tương ứng, ta thấy rằng p-value bé hơn mức ý nghĩa $\alpha = 0.05$ nên ta bác bỏ giả thuyết H_0 và chấp nhận giả thuyết H_1. Điều này có nghĩa là có ít nhất hai hãng có phương sai tốc độ nhân (*Core_Speed*) khác nhau. Vậy điều kiện phương sai tốc độ xử lý ở 4 hãng là bằng nhau không được thỏa mãn.

2. Đối với biến phân loại SLI_Crossfire:




```
data_anova$SLI_Crossfire <- factor(data_anova$SLI_Crossfire)
leveneTest(Core_Speed ~ SLI_Crossfire, data=data_anova)
```

```
> data_anova$SLI_Crossfire <- factor(data_anova$SLI_Crossfire)
> leveneTest(Core_Speed ~ SLI_Crossfire, data=data_anova)
Levene's Test for Homogeneity of Variance (center = median)
        Df F value  Pr(>F)
group    1  3.9502 0.04694 *
      3404
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Hình 22: Kết quả thu được

Ta thực hiện kiểm định giả thuyết sau đây:
Giả thuyết H_0: Phương sai của tốc độ nhân (*Core_Speed*) của GPU có SLI_Crossfire và GPU không có SLI_Crossfire là bằng nhau.
Giả thuyết H_1: Phương sai của tốc độ nhân (*Core_Speed*) của GPU có SLI_Crossfire và GPU không có SLI_Crossfire là khác nhau.

Dựa vào kết quả thu được từ R, quan sát giá trị cột p-value (Pr(>F)) tương ứng, ta thấy rằng p-value bé hơn mức ý nghĩa $\alpha = 0.05$ nên ta bác bỏ giả thuyết H_0 và chấp nhận giả thuyết H_1. Điều này có nghĩa là phương sai tốc độ nhân (*Core_Speed*) của GPU có SLI_Crossfire và GPU không có SLI_Crossfire là khác nhau.

Vậy điều kiện phương sai tốc độ xử lý ở hai nhóm GPU có và không có SLI_Crossfire bằng nhau không được thỏa mãn.

Sau khi kiểm tra các gia định của mô hình, ta thấy rằng, vẫn còn những điều kiện chưa được thỏa mãn, tuy nhiên để phục vụ cho mục đích của đề tài, chúng ta vẫn sử dụng ANOVA đa nhân tố như một phương án tham khảo.

VI. Thảo luận và mở rộng

Trong việc thảo luận về ưu và nhược điểm của phương pháp hồi quy tuyến tính bội, chúng ta nhận thấy rằng có nhiều điểm tích cực. Phương pháp này cho phép chúng ta phân tích tác động của nhiều biến độc lập lên một biến phụ thuộc cùng một lúc, giúp chúng ta hiểu rõ hơn về tầm quan trọng của mỗi biến độc lập trong việc dự đoán biến phụ thuộc. Đồng thời, đánh giá 7 yếu tố chính ảnh hưởng đến biến phụ thuộc "Core_Speed" giúp chúng ta dự đoán tốc độ nhân của GPU. Phân tích phương sai ANOVA cũng hữu ích để kiểm tra sự khác biệt giữa các nhóm của biến phụ thuộc và hiểu rõ hơn về ảnh hưởng của các yếu tố

Tuy nhiên, phương pháp này chỉ áp dụng được cho dữ liệu tuyến tính và cần loại bỏ các điểm ngoại lai để không làm sai lệch mô hình. ANOVA chỉ mô tả sự khác biệt giữa các nhóm mà không xác định được mối quan hệ nhân quả giữa các biến, và một số giả định của mô hình cần được kiểm tra kỹ.

Để mở rộng nghiên cứu, chúng ta cần thu thập nguồn dữ liệu đa dạng hơn để tăng tính linh hoạt và khách quan cho dự đoán. Và để đạt hiệu quả tốt nhất, chúng ta có thể kết hợp nhiều phương pháp dự đoán khác nhau như hồi quy tuyến tính, hồi quy phi tuyến, hồi quy lựa chọn, hồi quy đa cấp, v.v.

VII. Tài liệu tham khảo

1. Nguyễn Đình Huy (chủ biên), Nguyễn Bá Thi, Giáo trình Xác suất và Thống kê, 2018.
2. Nguyễn Văn Tuấn, Phân tích số liệu và biểu đồ bằng R, 2014
3. What Is Memory Bus In GPU?.
4. GPU Memory Clock Speed vs Core Clock Speed – Difference Explained.
5. Why GPU Memory Matters More Than You Think?.