BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

ĐỒ ÁN MÔN HỌC

ĐỀ TÀI:

PHÂN TÍCH DỮ LIỆU PHÂN HẠNG GIÁ TIỀN ĐIỆN THOẠI DỰA TRÊN BỘ DỮ LIỆU ĐƯỢC LẤY TỪ CÁC CÔNG TY CÔNG NGHỆ HÀNG ĐẦU TRÊN TOÀN THẾ GIỚI

Học phần: Biểu diễn trực quan dữ liệu
Nhóm Sinh Viên:

1. Phạm Nhật Hải
2. Đồng Đan Hoài
3. Ngô Thị Huyền
4. Nguyễn Đức Huy
5. Lê Minh Hoàng

Chuyên Ngành: Khoa học dữ liệu
Khóa: K46

Giảng Viên: TS. Nguyễn An Tế

TP. Hồ Chí Minh, Ngày xx tháng xx năm 2022

Mục Lục

CHƯƠNG 1. TỔNG QUAN

1.1 Lời giới thiệu

Điện thoại thông minh đóng một vai trò quan trọng trong xã hội hiện đại và đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày của nhiều người. Chúng đóng vai trò là phương tiện liên lạc chính, cho phép người dùng thực hiện cuộc gọi điện thoại, gửi tin nhắn văn bản và email cũng như truy cập internet. Điện thoại thông minh cũng cung cấp nhiều tính năng và khả năng khác, bao gồm điều hướng GPS, phát lại nhạc và video cũng như khả năng chạy nhiều ứng dụng cho các tác vụ như mạng xã hội, ngân hàng và mua sắm.

Điện thoại thông minh cũng đã trở thành một công cụ quan trọng đối với nhiều doanh nghiệp, vì chúng cung cấp một cách thuận tiện để nhân viên duy trì kết nối và cộng tác khi đang di chuyển. Ngoài ra, điện thoại thông minh đã cách mạng hóa các ngành như bán lẻ, vận tải và chăm sóc sức khỏe bằng cách cung cấp những cách mới để các công ty tương tác với khách hàng và hợp lý hóa hoạt động.

Nhìn chung, điện thoại thông minh đã mở rộng đáng kể khả năng và sự tiện lợi của giao tiếp hiện đại, đồng thời có tác động đáng kể đến các khía cạnh khác nhau của cuộc sống hàng ngày và trong cả kinh doanh.

1.2 Giới thiệu bài toán

Với sự phát triển bùng nổ của điện thoại thông minh thì cùng lúc đó cũng chính là một cuộc chạy đua trong lĩnh vực sản xuất điện thoại thông minh từ các nhà sản xuất điện thoại lớn và cũng như là nhỏ, đưa ra cho ta hàng tá lựa chọn khác nhau từ ngoài hình, mức giá, cấu hình, hiệu năng… Tuy có nhiều tiêu chí như vậy, nhưng trên hết chắc lẽ thứ mà người mua quan tâm đến nhất sẽ là mức giá tiền. Để giải được bài toán này, nhóm chúng tôi dựa vào những gì đã được học từ môn Biểu diễn Trực quan Dữ liệu cũng như trước đó là môn Lập trình Phân tích Dữ liệu để phân tích và đánh giá với những nhu cầu cụ thể của người mua thì mức giá phù hợp cho chiếc điện thoại sẽ là bao nhiêu. Từ đó cung cấp thêm thông tin giúp cho người mua dễ dàng hơn trong việc lựa chọn mua một chiếc điện thoại cho mình.

Trong báo cáo đồ án này, chúng em đã thu thập bộ dữ liệu được tổng hợp từ trên trang web Kaggle với bộ dữ liệu mang tên "***train.xlsx***" để sử dụng phục vụ cho việc thực hiện đề tài "***Phân tích dữ liệu phân hạng giá tiền điện thoại dựa trên bộ dữ liệu được lấy từ các công ty công nghệ hàng đầu trên toàn thế giới***" làm đồ án cho học phần Biểu diễn Trực quan Dữ liệu do thầy Nguyễn An Tế phụ trách và giảng dạy. Do sự giới hạn về mặt thời gian cũng những thiếu sót về mặt kiến thức nên đồ án có thể sẽ không tránh khỏi có phần sai sót. Kính mong quý giảng viên nhiệt tình góp ý để giúp chúng tôi có những tiếp thu tốt hơn trong môn học.

Qua đồ án, chúng em xin cảm ơn giảng viên phụ trách bộ môn là thầy Tế đã tâm huyết truyền đạt những kiến thức của thầy cho sinh viên chúng em suốt thời gian vừa qua.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Biểu diễn trực quan dữ liệu

2.1.1. Khái niệm

Trực quan hóa dữ liệu là biểu diễn đồ họa của thông tin và dữ liệu. Bằng cách sử dụng các yếu tố trực quan như biểu đồ, đồ thị và bản đồ, các công cụ trực quan hóa dữ liệu cung cấp một cách dễ tiếp cận để xem và hiểu các xu hướng, ngoại lệ và mẫu trong dữ liệu. Ngoài ra, nó cung cấp một cách tuyệt vời để nhân viên hoặc chủ doanh nghiệp trình bày dữ liệu cho những đối tượng không có kỹ thuật mà không bị nhầm lẫn.

2.2.2. Lợi ích và hạn chế

a. Lợi ích

Đôi mắt của chúng ta bị thu hút bởi **màu sắc và hoa văn**. Chúng ta có thể nhanh chóng xác định màu đỏ từ màu xanh lam và hình vuông từ hình tròn. Trực quan hóa dữ liệu là một hình thức nghệ thuật thị giác khác mà nó thu hút sự quan tâm và khiến chú ý đến thông điệp. Khi chúng ta nhìn thấy một biểu đồ, chúng ta nhanh chóng thấy các **xu hướng và giá trị ngoại lệ**. Nếu chúng ta có thể nhìn thấy điều gì đó, chúng ta sẽ tiếp thu nó một cách nhanh chóng. Đó là cách kể chuyện có mục đích.

Một số ưu điểm khác của trực quan hóa dữ liệu bao gồm:

- Dễ dàng chia sẻ thông tin.
- Tương tác khám phá các cơ hội.
- Trực quan hóa các mẫu và mối quan hệ.

b. Hạn chế

Mặc dù có nhiều ưu điểm, nhưng một số nhược điểm có vẻ ít rõ ràng hơn. Ví dụ: khi xem một hình ảnh trực quan với nhiều điểm dữ liệu khác nhau, rất dễ đưa ra giả định không chính xác. Hoặc đôi khi hình ảnh trực quan được thiết kế sai khiến nó bị sai lệch hoặc khó hiểu.

Một số nhược điểm khác bao gồm:

- Thông tin sai lệch hoặc không chính xác.
- Mối tương quan không phải lúc nào cũng có nghĩa là nhân - quả.
- Thông điệp cốt lõi có thể bị mất trong bản dịch.

2.2.3. Sự quan trọng

Tầm quan trọng của trực quan hóa dữ liệu rất đơn giản: nó giúp mọi người nhìn thấy, tương tác và hiểu rõ hơn về dữ liệu. Cho dù đơn giản hay phức tạp, hình ảnh trực quan phù hợp có thể đưa mọi người vào cùng một trang, bất kể trình độ chuyên môn của họ.

2.2.4. Ứng dụng

Khi "thời đại của Dữ liệu lớn" bắt đầu phát triển mạnh mẽ, trực quan hóa là một công cụ ngày càng quan trọng để hiểu được hàng nghìn tỷ hàng dữ liệu được tạo ra mỗi ngày. Trực quan hóa dữ liệu giúp kể chuyện bằng cách sắp xếp dữ liệu thành một dạng dễ hiểu hơn, làm nổi bật các xu hướng và điểm khác biệt. Một hình ảnh trực quan tốt sẽ kể một câu chuyện, loại bỏ nhiễu khỏi dữ liệu và làm nổi bật thông tin hữu ích.

Tuy nhiên, nó không chỉ đơn giản là chỉnh sửa một biểu đồ để làm cho nó trông đẹp hơn hoặc thêm vào phần "thông tin" của một đồ họa thông tin. Trực quan hóa dữ liệu hiệu quả là một hành động cân bằng tinh tế giữa hình thức và chức năng. Biểu đồ đơn giản nhất có thể quá nhàm chán để thu hút bất kỳ thông báo nào hoặc nó có thể nói lên một điểm mạnh mẽ; hình ảnh trực quan tuyệt đẹp nhất có thể hoàn toàn thất bại trong việc truyền tải đúng thông điệp hoặc nó có thể nói lên nhiều điều. Dữ liệu và hình ảnh cần kết hợp với nhau và có một nghệ thuật để kết hợp phân tích tuyệt vời với cách kể chuyện tuyệt vời.

2.2. Kiểm định giả thuyết

a. Khái niệm

Kiểm định giả thuyết là một hành động trong thống kê, theo đó một nhà phân tích kiểm tra một giả định liên quan đến tham số tổng số (population). Phương pháp được nhà phân tích sử dụng phụ thuộc vào bản chất của dữ liệu được sử dụng và lý do phân tích.

Thử nghiệm giả thuyết được sử dụng để đánh giá tính hợp lý của một giả thuyết bằng cách sử dụng dữ liệu mẫu (sample). Dữ liệu như vậy có thể đến từ một nhóm dân số lớn hơn hoặc từ một quy trình tạo dữ liệu.

b. Cách kiểm định giả thuyết hoạt động

Trong thử nghiệm giả thuyết, một nhà phân tích kiểm tra một mẫu thống kê, với mục tiêu cung cấp bằng chứng về tính hợp lý của giả thuyết khống(thay thế).

Các nhà phân tích thống kê kiểm tra một giả thuyết bằng cách đo lường và kiểm tra một mẫu ngẫu nhiên của quần thể đang được phân tích. Tất cả các nhà phân tích sử dụng một mẫu dân số ngẫu nhiên để kiểm tra hai giả thuyết khác nhau: giả thuyết không và giả thuyết thay thế.

Giả thuyết không thường là một giả thuyết về sự bình đẳng giữa các tham số dân số; ví dụ: một giả thuyết không có thể nói rằng lợi nhuận trung bình tổng thể bằng không. Giả thuyết thay thế thực sự trái ngược với giả thuyết không (ví dụ: lợi tức trung bình dân số không bằng 0). Vì vậy, chúng loại trừ lẫn nhau, và chỉ một cái có thể đúng. Tuy nhiên, một trong hai giả thuyết sẽ luôn đúng.

c. Các bước kiểm định giả thuyết

Tất cả các giả thuyết đều được kiểm tra bằng quy trình bốn bước:

- **Bước đầu tiên** là nhà phân tích phải đưa ra hai giả thuyết để chỉ có một giả thuyết đúng.
- **Bước tiếp theo** là xây dựng một kế hoạch phân tích, trong đó phác thảo cách dữ liệu sẽ được đánh giá.
- **Bước thứ ba** là thực hiện kế hoạch và phân tích vật lý dữ liệu mẫu.
- **Bước thứ tư** và cũng là bước cuối cùng là phân tích kết quả và bác bỏ giả thuyết không, hoặc tuyên bố rằng giả thuyết không là hợp lý, dựa trên dữ liệu.

2.3. Phân tích dữ liệu

2.3.1. Đơn biến (Univariate Analysis):

Phân tích đơn biến là một hình thức phân tích chỉ liên quan đến một biến duy nhất. Trong môi trường thực tế, phân tích đơn biến có nghĩa là phân tích một biến (hoặc cột) duy nhất trong tập dữ liệu (bảng dữ liệu).

2.3.2. Hai biến (Bivariate Analysis):

a. Khái niệm

Phân tích hai biến có nghĩa là phân tích dữ liệu hai biến. Đây là một phân tích thống kê duy nhất được sử dụng để tìm ra mối quan hệ tồn tại giữa hai bộ giá trị. Các biến có liên quan là X và Y.

b. Cách tiến hành một phân tích 2 biến

Đây là cách phân tích 2 biến được thực hiện.

- Biểu đồ phân tán - Điều này đưa ra ý tưởng về các mẫu có thể được hình thành bằng cách sử dụng hai biến

- Phân tích hồi quy – Điều này sử dụng nhiều công cụ để xác định xem bài đăng dữ liệu có thể liên quan như thế nào. Bài đăng có thể theo một đường cong hàm mũ. Phân tích hồi quy đưa ra phương trình cho một đường hoặc đường cong. Nó cũng giúp tìm ra hệ số tương quan.
- Hệ số tương quan – Hệ số cho biết liệu dữ liệu được đề cập có liên quan hay không. Khi hệ số tương quan bằng 0, các biến không liên quan với nhau. Nếu hệ số tương quan là dương hoặc âm 1, thì điều này có nghĩa là các biến có tương quan hoàn hảo.

c. Có 3 loại tương quan hai chiều:

- Số và số: Trong loại biến này, cả hai biến của dữ liệu hai chiều, bao gồm biến phụ thuộc và biến độc lập, đều có giá trị số.
- Phân loại và phân loại: Khi cả hai biến trong dữ liệu hai chiều đều ở dạng tĩnh, dữ liệu sẽ được diễn giải, đồng thời đưa ra các tuyên bố và dự đoán về nó. Trong quá trình nghiên cứu, phân tích sẽ giúp xác định nguyên nhân và tác động để kết luận biến đưa ra là biến phân loại.
- Số và phân loại: Đây là khi một trong các biến là số và biến kia là phân loại. Phân tích hai biến là một loại phân tích thống kê trong đó hai biến được quan sát đối lập với nhau. Một trong các biến sẽ phụ thuộc và biến còn lại là độc lập. Các biến được ký hiệu là X và Y. Các thay đổi được phân tích giữa hai biến để hiểu mức độ thay đổi đã xảy ra.

2.3.3. Đa biến (Multivariate analysis):

a. Khái niệm

Phân tích đa biến bao gồm tất cả các kỹ thuật thống kê được sử dụng để phân tích nhiều hơn hai biến cùng một lúc. Mục đích là để tìm ra các mẫu và mối tương quan giữa một số biến đồng thời—cho phép hiểu sâu hơn, phức tạp hơn về một kịch bản nhất định so với những gì sẽ nhận được với phân tích hai biến.

b. Kỹ thuật phân tích dữ liệu đa biến

Có nhiều kỹ thuật khác nhau để phân tích đa biến và chúng có thể được chia thành hai loại:

b1. **kỹ thuật phụ thuộc**

Có hai loại kỹ thuật phân tích đa biến: Kỹ thuật phụ thuộc, xem xét mối quan hệ nhân quả giữa các biến.

b2. **kỹ thuật phụ thuộc lẫn nhau**

Khám phá cấu trúc của tập dữ liệu

Một số kỹ thuật phân tích đa biến hữu ích

- Hồi quy tuyến tính bội

- Nhiều hồi quy logistic
- Phân tích phương sai đa biến (MANOVA)
- Phân tích nhân tố
- Phân tích cluster

2.4. PCA

a. Khái niệm

Phân tích thành phần chính, hay **PCA**, là một **phương pháp giảm kích thước** thường được sử dụng để giảm kích thước của các tập **dữ liệu lớn**, bằng cách chuyển đổi một tập hợp lớn các biến thành một biến nhỏ hơn mà vẫn chứa hầu hết thông tin trong tập hợp lớn.

Việc giảm số lượng biến của một tập dữ liệu đương nhiên phải **trả giá bằng độ chính xác**, nhưng mẹo trong việc giảm kích thước là đánh đổi một chút độ chính xác để **lấy sự đơn giản**. Bởi vì các tập dữ liệu nhỏ hơn sẽ dễ khám phá và trực quan hóa hơn, đồng thời giúp việc phân tích dữ liệu trở nên dễ dàng và nhanh hơn nhiều đối với các thuật toán máy học mà **không cần xử lý các biến ngoại lai**.

Vì vậy, tóm lại, ý tưởng về PCA rất đơn giản — giảm số lượng biến của một tập dữ liệu, đồng thời bảo toàn càng nhiều thông tin càng tốt.

b. Giải thích từng bước về PCA

BƯỚC 1: TIÊU CHUẨN HÓA

Mục đích của bước này là **chuẩn hóa phạm vi** của các biến ban đầu liên tục để mỗi biến đóng góp như nhau vào phân tích.

Cụ thể hơn, lý do tại sao việc thực hiện tiêu chuẩn hóa trước PCA lại quan trọng là vì **PCA khá nhạy cảm đối với phương sai của các biến ban đầu.** Nghĩa là, nếu có sự khác biệt lớn giữa phạm vi của các biến ban đầu, thì những biến có phạm vi lớn hơn sẽ chiếm ưu thế so với những biến có phạm vi nhỏ (ví dụ: một biến nằm trong khoảng từ 0 đến 100 sẽ chiếm ưu thế so với biến nằm trong khoảng từ 0 đến 1), điều này sẽ dẫn đến kết quả sai lệch. Vì vậy, việc chuyển đổi dữ liệu thành các tỷ lệ có thể so sánh được có thể ngăn chặn vấn đề này.

Về mặt toán học, điều này có thể được thực hiện bằng cách trừ đi giá trị trung bình và chia cho độ lệch chuẩn cho mỗi giá trị của mỗi biến.

$$z = \frac{value - mean}{standard\ deviation}$$

Sau khi chuẩn hóa xong, tất cả các biến sẽ được chuyển đổi về cùng một tỷ lệ.

BƯỚC 2: TÍNH TOÁN MA TRẬN COVARIANCE

Mục đích của bước này là để hiểu các biến của tập dữ liệu đầu vào thay đổi như thế nào so với giá trị trung bình đối với nhau, hay nói cách khác, để xem liệu có bất kỳ mối quan hệ nào giữa chúng hay không. Bởi vì đôi khi, các biến có mối tương quan cao đến mức chúng chứa thông tin dư thừa. Vì vậy, để **xác định các mối tương quan** này, nhà phát triển cần tính toán ma trận hiệp phương sai.

Ma trận hiệp phương sai là một **ma trận đối xứng p × p** (trong đó p là số chiều) có các mục nhập là hiệp phương sai liên quan đến tất cả các cặp biến ban đầu có thể có. Ví dụ: đối với tập dữ liệu 3 chiều có 3 biến x, y và z, ma trận hiệp phương sai là ma trận 3×3 của ma trận này từ:

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

Vì hiệp phương sai của một biến với chính nó là phương sai của nó (Cov(a,a)=Var(a)), trong đường chéo chính (Trên cùng bên trái xuống dưới cùng bên phải), chúng ta thực sự có phương sai của từng biến ban đầu. Và vì hiệp phương sai có tính chất giao hoán (Cov(a,b)=Cov(b,a)), nên các phần tử của ma trận hiệp phương sai là đối xứng với đường chéo chính, có nghĩa là phần trên và phần dưới của tam giác bằng nhau.

Các hiệp phương sai mà có dưới dạng các mục của ma trận cho biết điều gì về mối tương quan giữa các biến?

Nó thực sự là dấu hiệu của hiệp phương sai quan trọng:

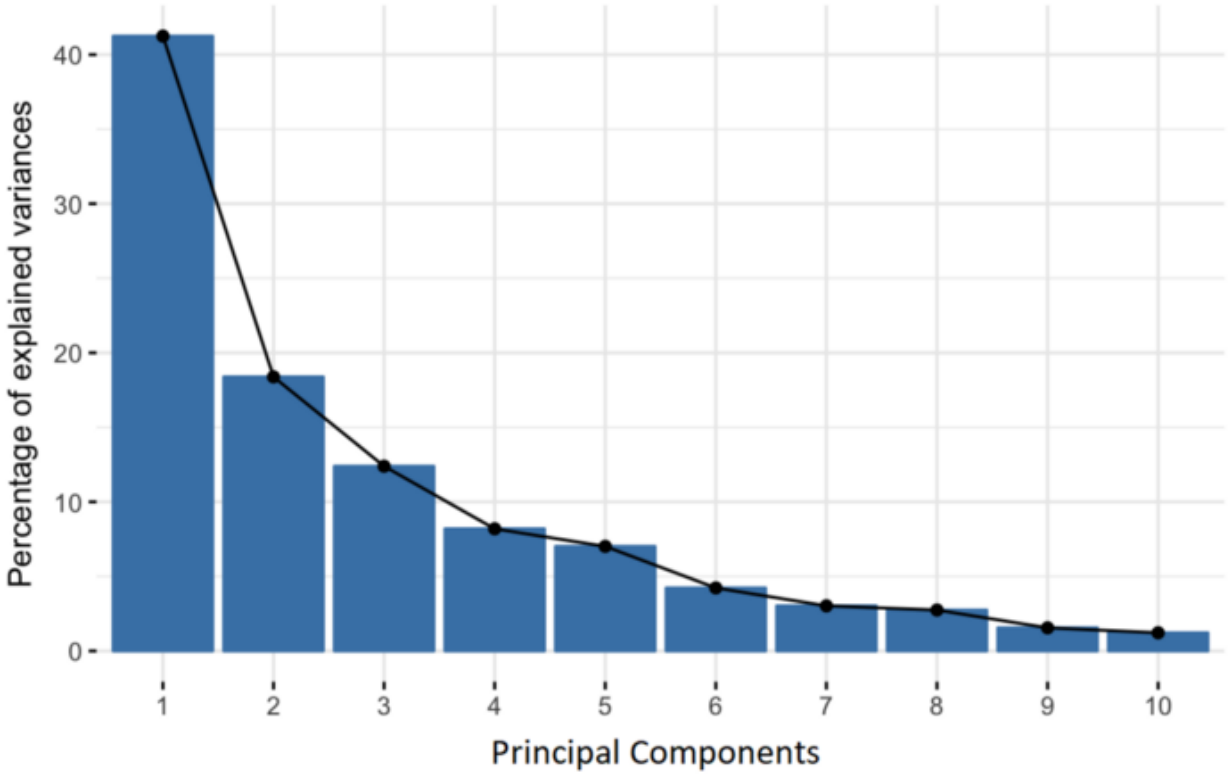
- Nếu dương thì: 2 biến cùng tăng giảm (có tương quan)
- Nếu âm thì: cái này tăng khi cái kia giảm (Tương quan nghịch)

BƯỚC 3: Tính toán các **VECTƠ RIÊNG** và **(GIÁ) TRỊ RIÊNG** của ma trận hiệp lực để xác định các **THÀNH PHẦN CHÍNH**

Các **vectơ riêng** và **(giá) trị riêng** là các khái niệm đại số tuyến tính mà người dùng cần tính toán từ ma trận hiệp phương sai để **xác định** các **thành phần chính** của dữ liệu. Trước khi giải thích các khái niệm này, trước tiên hãy hiểu ý nghĩa của các thành phần chính.

Các thành phần chính là các biến mới được xây dựng dưới dạng kết hợp tuyến tính hoặc hỗn hợp của các biến ban đầu. Các kết hợp này được thực hiện theo cách sao cho các biến mới (nghĩa là các thành phần chính) không tương quan với nhau và hầu hết thông tin trong các biến ban đầu được ép hoặc nén vào các thành phần đầu tiên. Vì vậy, ý tưởng là dữ liệu 10 chiều cung cấp cho người dùng 10 thành phần chính, nhưng PCA cố gắng đưa thông tin tối đa có thể vào thành phần đầu tiên, sau đó là thông tin còn lại tối đa

trong thành phần thứ hai, v.v., cho đến khi có thứ gì đó giống như được hiển thị trong biểu đồ màn hình bên dưới.



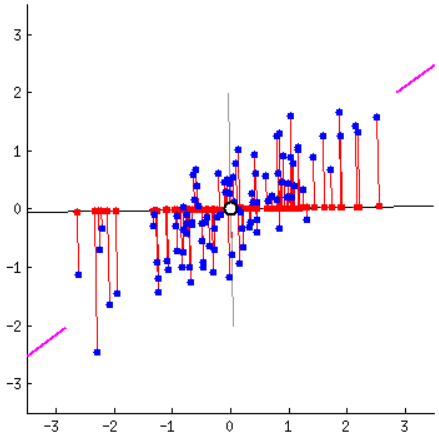
Tổ chức thông tin trong các thành phần chính theo cách này sẽ cho phép người dùng **giảm kích thước** mà **không làm mất nhiều thông tin** và điều này bằng cách **loại bỏ** các **thành phần có ít thông tin** và coi các thành phần còn lại là các biến mới của người dùng.

Một điều quan trọng cần nhận ra ở đây là các thành phần chính khó diễn giải hơn và không có bất kỳ ý nghĩa thực sự nào vì chúng được xây dựng dưới dạng kết hợp tuyến tính của các biến ban đầu.

Về mặt hình học, các thành phần chính đại diện cho các hướng của dữ liệu giải thích lượng phương sai tối đa, nghĩa là các đường thu được hầu hết thông tin của dữ liệu. Mối quan hệ giữa phương sai và thông tin ở đây là, phương sai của một dòng càng lớn thì độ phân tán của các điểm dữ liệu dọc theo nó càng lớn và độ phân tán dọc theo một dòng càng lớn thì càng có nhiều thông tin. Nói một cách đơn giản, chỉ cần nghĩ về các thành phần chính như **các trục mới** cung cấp góc tốt nhất để xem và **đánh giá dữ liệu**, để sự khác biệt giữa các quan sát được nhìn thấy **rõ hơn**.

Cách PCA xây dựng các thành phần chính

Vì có nhiều thành phần chính cũng như có nhiều biến trong dữ liệu, nên các thành phần chính được **xây dựng** theo cách sao cho **thành phần chính đầu tiên chiếm phương sai lớn nhất** có thể có trong tập dữ liệu. Ví dụ: giả sử rằng biểu đồ phân tán của tập dữ liệu như được hiển thị bên dưới, liệu rằng có thể đoán thành phần chính đầu tiên không? Vâng, đó gần như là đường trùng với các dấu màu tím vì nó đi qua gốc toạ độ và là đường mà hình chiếu của các điểm (chấm đỏ) trải rộng nhất. Hay nói một cách toán học, đó là **đường tối đa hóa phương sai** (trung bình của bình phương khoảng cách từ các điểm được chiếu (chấm đỏ) đến gốc tọa độ).



Thành phần chính thứ hai được tính theo cách tương tự, với điều kiện là nó **không tương quan** với (nghĩa là vuông góc với) **thành phần chính thứ nhất** và **nó chiếm phương sai cao nhất tiếp theo**.

Điều này **tiếp tục** cho đến khi **tổng số p thành phần chính** được tính toán, bằng với số biến ban đầu.

Bây giờ chúng ta đã hiểu ý nghĩa của các thành phần chính, hãy quay lại với các **vectơ riêng** và **(giá) trị riêng**. Điều đầu tiên bạn cần biết về chúng là chúng **luôn đi theo cặp**, do đó mọi **vectơ riêng** đều **có một giá trị riêng**. Và số của chúng bằng với số chiều của dữ liệu. Ví dụ, đối với tập dữ liệu 3 chiều, có 3 biến nên có 3 vector riêng với 3 giá trị riêng tương ứng.

Không cần bàn cãi thêm, chính các vectơ riêng và giá trị riêng đứng đằng sau tất cả những điều kỳ diệu được giải thích ở trên, bởi vì các vectơ riêng của ma trận Hiệp phương sai thực sự là hướng của các trục nơi có nhiều phương sai nhất (hầu hết thông tin) và nó được gọi là Thành phần chính. Và các giá trị riêng chỉ đơn giản là các hệ số được gắn với các vectơ riêng, cho biết lượng phương sai có trong mỗi Thành phần chính.

Bằng cách xếp hạng các vectơ riêng của bạn theo thứ tự các giá trị riêng của chúng, từ cao nhất đến thấp nhất, bạn sẽ có được các thành phần chính theo thứ tự có ý nghĩa.

Thí dụ:

Giả sử rằng tập dữ liệu của chúng ta là 2 chiều với 2 biến x, y và các vectơ riêng và giá trị riêng của ma trận hiệp phương sai như sau:

$$v1 = \begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \qquad \lambda_1 = 1.284028$$

$$v2 = \begin{bmatrix} -0.7351785 \\ 0.6778736 \end{bmatrix} \qquad \lambda_2 = 0.04908323$$

Ví dụ phân tích thành phần chính

Nếu xếp hạng các giá trị riêng theo thứ tự giảm dần, người dùng sẽ nhận được λ1>λ2, có nghĩa là véc tơ riêng tương ứng với thành phần chính đầu tiên (PC1) là v1 và véc tơ riêng tương ứng với thành phần thứ hai (PC2) là v2.

Sau khi có các thành phần chính, để tính phần trăm phương sai (thông tin) chiếm bởi từng thành phần, người dùng chia giá trị riêng của từng thành phần cho tổng các giá trị riêng. Nếu người áp dụng điều này vào ví dụ trên, có thể thấy rằng PC1 và PC2 tương ứng mang 96% và 4% phương sai của dữ liệu.

BƯỚC 4: Vectơ ĐẶC TRƯNG

Như đã thấy ở bước trước, tính toán các vectơ riêng và sắp xếp chúng theo giá trị riêng của chúng theo thứ tự giảm dần, cho phép tìm các thành phần chính theo thứ tự có ý nghĩa. Trong bước này, những gì cần làm là chọn giữ lại tất cả các thành phần này hay loại bỏ những thành phần có ý nghĩa thấp hơn (có giá trị riêng thấp) và tạo với những thành phần còn lại một ma trận vectơ mà được gọi là vectơ đặc trưng.

Vì vậy, **vectơ đặc trưng** chỉ đơn giản **là một ma trận có các cột** là **vectơ riêng** của các thành phần mà **được quyết định giữ lại**. Đây là bước đầu tiên hướng tới việc giảm kích thước, bởi vì nếu chọn chỉ giữ lại các vectơ riêng p (thành phần) trong số n, tập dữ liệu cuối cùng sẽ chỉ có p kích thước.

Thí dụ:

Tiếp tục với ví dụ từ bước trước, có thể tạo một vectơ đặc trưng với cả hai vectơ riêng v1 và v2:

$$\begin{bmatrix} 0.6778736 & -0.7351785 \\ 0.7351785 & 0.6778736 \end{bmatrix}$$

Hoặc loại bỏ véc tơ riêng v2, véc tơ riêng ít quan trọng hơn và tạo thành véc tơ đặc trưng chỉ với v1:

$$\begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix}$$

Việc loại bỏ véc tơ riêng v2 sẽ làm giảm số chiều đi 1 và do đó sẽ làm mất thông tin trong tập dữ liệu cuối cùng. Nhưng cho rằng v2 chỉ mang **4% thông tin**, do đó, **sự mất mát** sẽ **không quan trọng** và **vẫn sẽ có 96% thông tin** được mang bởi v1.

Vì vậy, như đã thấy trong ví dụ, người dùng có quyền chọn giữ lại tất cả các thành phần hay loại bỏ những thành phần ít quan trọng hơn, tùy thuộc vào những gì người dùng đang tìm kiếm. Bởi vì nếu người dùng chỉ muốn mô tả dữ liệu của mình theo các biến mới (các thành phần chính) không tương quan mà không tìm cách giảm kích thước, thì không cần thiết phải loại bỏ các thành phần ít quan trọng hơn.

BƯỚC CUỐI: GHI LẠI DỮ LIỆU THEO CÁC TRỤC THÀNH PHẦN CHÍNH

Ở các bước trước, ngoài việc chuẩn hóa, không thực hiện bất kỳ thay đổi nào trên dữ liệu, mà chỉ chọn các thành phần chính và tạo thành vectơ đặc trưng, nhưng **bộ dữ liệu đầu vào luôn luôn theo trục ban đầu** (tức là theo trục các biến ban đầu).

Trong bước này, là **bước cuối cùng**, mục đích là **sử dụng vectơ đặc trưng** được tạo bằng cách sử dụng các vectơ riêng của ma trận hiệp phương sai, để **định hướng lại dữ liệu** từ các trục ban đầu sang các trục được đại diện bởi các thành phần chính (do đó có tên là Phân tích thành phần chính). Điều này có thể được thực hiện bằng cách nhân chuyển vị của tập dữ liệu gốc với chuyển vị của vectơ đặc trưng.

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

2.5. Mô hình phân lớp

2.5.1. Mô hình KNN

a. Khái niệm

Một cách ngắn gọn, KNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách *chỉ* dựa trên thông tin của K điểm dữ liệu trong training set gần nó nhất (K-lân cận), *không quan tâm đến việc có một vài điểm dữ liệu trong những điểm gần nhất này là nhiễu*.

b. Ưu và nhược điểm

Ưu điểm của KNN

- Độ phức tạp tính toán của quá trình training là bằng 0.
- Việc dự đoán kết quả của dữ liệu mới rất đơn giản.
- Không cần giả sử gì về phân phối của các class.

Nhược điểm của KNN

- KNN rất nhạy cảm với nhiễu khi K nhỏ.
- Như đã nói, KNN là một thuật toán mà mọi tính toán đều nằm ở khâu test. Trong đó việc tính khoảng cách tới *từng* điểm dữ liệu trong training set sẽ tốn rất nhiều thời gian, đặc biệt là với các cơ sở dữ liệu có số chiều lớn và có nhiều điểm dữ liệu. Với K càng lớn thì độ phức tạp cũng sẽ tăng lên. Ngoài ra, việc lưu toàn bộ dữ liệu trong bộ nhớ cũng ảnh hưởng tới hiệu năng của KNN.

2.5.2. Mô Hình Logistic Regression

a. Khái niệm

Loại mô hình thống kê này (còn được gọi là mô hình logit) thường được sử dụng để phân loại và phân tích dự đoán. Hồi quy logistic ước tính xác suất xảy ra một sự kiện, chẳng hạn như đã bỏ phiếu hoặc không bỏ phiếu, dựa trên tập dữ liệu nhất định gồm các biến độc lập. Vì kết quả là một xác suất nên biến phụ thuộc có giới hạn từ 0 đến 1. Trong hồi quy logistic, phép biến đổi logit được áp dụng trên tỷ lệ cược—nghĩa là xác suất thành công chia cho xác suất thất bại. Điều này còn thường được gọi là tỷ lệ cược log hoặc logarit tự nhiên của tỷ lệ cược và hàm logistic này được biểu thị bằng các công thức sau:

$$logit(pi) = \frac{1}{1+e^{-pi}}$$

$$\ln\left(\frac{pi}{1-pi}\right) = \beta_0 + \beta_1 \times x_1 + \cdots + \beta_k \times x_k$$

Trong phương trình hồi quy logistic này, logit(pi) là biến phụ thuộc hoặc biến phản hồi và x là biến độc lập. Tham số beta hoặc hệ số trong mô hình này thường được ước tính thông qua ước tính khả năng tối đa (MLE). Phương pháp này kiểm tra các giá trị khác nhau của beta thông qua nhiều lần lặp lại để tối ưu hóa cho phù hợp nhất với tỷ lệ chênh lệch nhật ký. Tất cả các lần lặp này tạo ra hàm khả năng ghi nhật ký và hồi quy logistic tìm cách tối đa hóa hàm này để tìm ước tính tham số tốt nhất. Sau khi tìm thấy hệ số tối ưu (hoặc các hệ số nếu có nhiều hơn một biến độc lập), xác suất có điều kiện cho từng quan sát có thể được tính toán, ghi lại và cộng lại với nhau để tạo ra xác suất dự đoán. Đối với phân loại nhị phân, xác suất nhỏ hơn 0,5 sẽ dự đoán 0 trong khi xác suất lớn hơn 0 sẽ dự đoán 1. Sau khi mô hình được tính toán, cách tốt nhất là đánh giá mức độ mô hình dự đoán biến phụ thuộc, được gọi là mức độ tốt của Phù hợp. Phép thử Hosmer–Lemeshow là một phương pháp phổ biến để đánh giá mức độ phù hợp của mô hình.

b. Giải thích hồi quy logistic

Tỷ lệ chênh lệch nhật ký có thể khó hiểu trong phân tích dữ liệu hồi quy logistic. Do đó, việc lũy thừa các ước tính beta là phổ biến để chuyển đổi kết quả thành tỷ lệ chênh lệch (Odds Ratio-OR), giúp dễ dàng giải thích kết quả. OR đại diện cho tỷ lệ cược mà một kết quả sẽ xảy ra với một sự kiện cụ thể, so với tỷ lệ cược của kết quả xảy ra khi không có sự kiện đó. Nếu OR lớn hơn 1, thì sự kiện đó có khả năng tạo ra một kết quả cụ thể cao hơn. Ngược lại, nếu OR nhỏ hơn 1, thì sự kiện đó có tỷ lệ xảy ra kết quả đó thấp hơn. Dựa trên phương trình ở trên, việc giải thích tỷ lệ chênh lệch có thể được biểu thị như sau: tỷ lệ cược thành công thay đổi theo $e^{c\beta_1}$ lần cho mỗi lần tăng đơn vị c trong x. Để sử dụng một ví dụ, giả sử chúng ta ước tính tỷ lệ sống sót trên tàu Titanic với điều kiện người đó là nam giới và tỷ lệ chênh lệch đối với nam giới là 0,0810. Có thể giải thích tỷ lệ chênh lệch là tỷ lệ sống sót của nam giới giảm theo hệ số 0,0810 so với nữ giới, giữ nguyên tất cả các biến số khác.

c. Các loại hồi quy logistic

Có ba loại mô hình hồi quy logistic, được xác định dựa trên phản ứng phân loại.

- Hồi quy logistic nhị phân (**Binary logistic regression**): Theo cách tiếp cận này, biến phản hồi hoặc biến phụ thuộc có bản chất phân đôi—tức là nó chỉ có hai kết quả có thể xảy ra (ví dụ: 0 hoặc 1). Một số ví dụ phổ biến về việc sử dụng nó bao gồm dự đoán xem một e-mail có phải là thư rác hay không phải thư rác hoặc khối u ác tính hay không ác tính. Trong hồi quy logistic, đây là cách tiếp cận được sử dụng phổ biến nhất và nói chung, nó là một trong những cách phân loại phổ biến nhất để phân loại nhị phân.
- Hồi quy logistic đa thức (**Multinomial logistic regression**): Trong loại mô hình hồi quy logistic này, biến phụ thuộc có ba hoặc nhiều kết quả có thể xảy ra; tuy nhiên, các giá trị này không có thứ tự cụ thể. Ví dụ: các hãng phim muốn dự đoán thể loại phim mà khán giả có khả năng xem để tiếp thị phim hiệu quả hơn. Mô hình hồi quy logistic đa thức có thể giúp hãng phim xác định mức độ ảnh hưởng của tuổi tác, giới tính và tình trạng hẹn hò của một người đối với loại phim họ thích. Sau đó, hãng phim có thể định hướng chiến dịch quảng cáo của một bộ phim cụ thể tới một nhóm người có khả năng sẽ đi xem bộ phim đó.
- Hồi quy logistic thông thường (**Ordinal logistic regression**): Loại mô hình hồi quy logistic này được tận dụng khi biến phản hồi có ba hoặc nhiều hơn kết quả có thể xảy ra, nhưng trong trường hợp này, các giá trị này có một thứ tự xác định. Ví dụ về các câu trả lời theo thứ tự bao gồm thang điểm từ A đến F hoặc thang đánh giá từ 1 đến 5.

2.5.3. Mô hình SVM

a. Khái niệm

Support Vector Machine (SVM) là một hệ thống học có giám sát và được sử dụng cho các bài toán phân loại và hồi quy. Máy vectơ hỗ trợ được nhiều người cực kỳ ưa chuộng vì nó tạo ra độ chính xác đáng chú ý với sức mạnh tính toán ít hơn. Nó chủ yếu được sử dụng trong các vấn đề phân loại. Chúng tôi có ba loại học tập được giám sát, không giám sát và học tập tăng cường. Máy vectơ hỗ trợ là một bộ phân loại chọn lọc được xác định chính thức bằng cách chia siêu phẳng.

Đưa ra dữ liệu đào tạo được gắn nhãn, thuật toán tạo ra siêu phẳng tốt nhất để phân loại các ví dụ mới. Trong không gian hai chiều, siêu phẳng này là một đường chia mặt phẳng thành hai phần mà mỗi lớp nằm ở hai bên. Mục đích của thuật toán máy vectơ hỗ trợ là tìm một siêu phẳng trong không gian N chiều phân loại riêng các điểm dữ liệu.

b. Ưu và nhược điểm

Ưu điểm:

- SVM hoạt động tương đối tốt khi có một biên độ phân ly dễ hiểu giữa các lớp.
- Nó hiệu quả hơn trong không gian nhiều chiều.
- Nó có hiệu quả trong trường hợp số lượng kích thước lớn hơn số lượng mẫu vật.
- SVM có hệ thống bộ nhớ tương đối.

Nhược điểm:

- Thuật toán SVM không được chấp nhận đối với các tập dữ liệu lớn.
- Nó không thực thi tốt khi tập dữ liệu có nhiều âm thanh hơn, tức là các lớp mục tiêu đang chồng chéo.
- Trong trường hợp số lượng thuộc tính cho mỗi điểm dữ liệu vượt quá số lượng mẫu dữ liệu đào tạo, máy vectơ hỗ trợ sẽ hoạt động kém.
- Vì phân loại SVM hoạt động bằng cách đặt các điểm dữ liệu, bên trên và bên dưới siêu phẳng phân loại nên không có sự làm rõ xác suất nào cho việc phân loại.

c. Ứng dụng

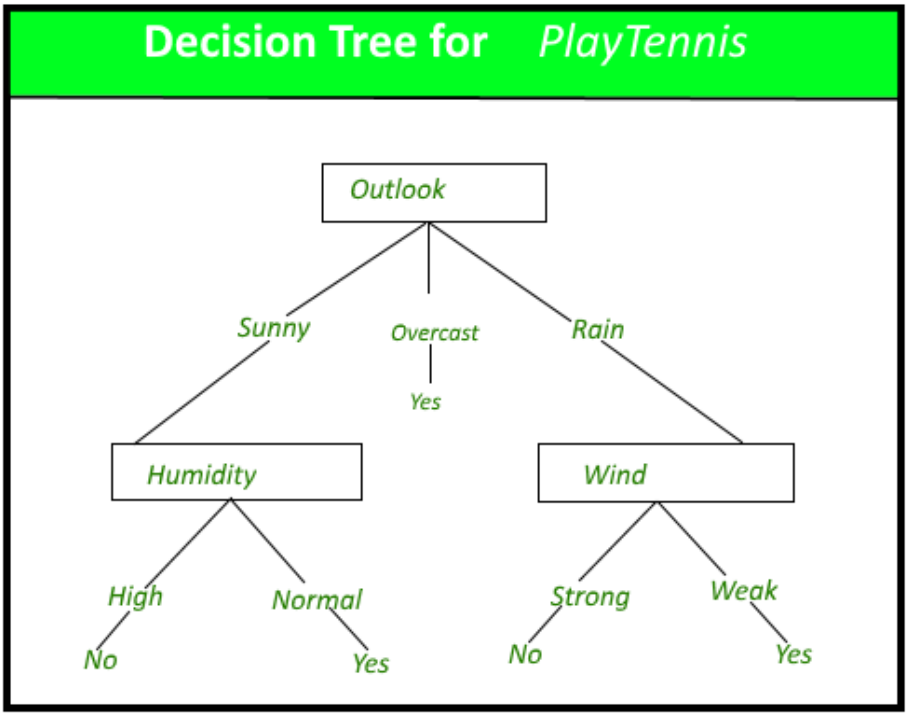
- Quan sát khuôn mặt
- Sắp xếp văn bản và siêu văn bản
- Nhóm các bức chân dung
- Ghi nhớ chữ viết tay
- Kiểm soát dự đoán tổng quát

2.5.4. Mô hình Decision Tree

a. Khái niệm

Cây quyết định (Decision Tree) là công cụ mạnh mẽ và phổ biến nhất để phân loại và dự đoán. Cây quyết định là một cấu trúc cây giống như lưu đồ, trong đó mỗi nút bên trong

biểu thị một phép thử trên một thuộc tính, mỗi nhánh biểu thị một kết quả của phép thử và mỗi nút lá (nút đầu cuối) giữ một nhãn lớp.



b. Xây dựng cây quyết định

Một cây có thể được "học" bằng cách chia tập hợp nguồn thành các tập con dựa trên kiểm tra giá trị thuộc tính. Quá trình này được lặp lại trên mỗi tập con dẫn xuất theo cách đệ quy được gọi là phân vùng đệ quy. Đệ quy được hoàn thành khi tất cả các tập hợp con tại một nút có cùng giá trị của biến mục tiêu hoặc khi việc tách không còn thêm giá trị cho các dự đoán. Việc xây dựng bộ phân loại cây quyết định không yêu cầu bất kỳ cài đặt tham số hoặc kiến thức miền nào và do đó phù hợp với khám phá kiến thức khám phá. Cây quyết định có thể xử lý dữ liệu nhiều chiều. Nhìn chung, bộ phân loại cây quyết định có độ chính xác tốt. Quy nạp cây quyết định là một phương pháp quy nạp điển hình để học kiến thức về phân loại.

c. Biểu diễn cây quyết định

Cây quyết định phân loại các thể hiện bằng cách sắp xếp chúng xuống cây từ gốc đến một số nút lá, cung cấp sự phân loại của thể hiện. Một thể hiện được phân loại bằng cách bắt đầu từ nút gốc của cây, kiểm tra thuộc tính được chỉ định bởi nút này, sau đó di chuyển

xuống nhánh cây tương ứng với giá trị của thuộc tính như trong hình trên. Quá trình này sau đó được lặp lại cho cây con bắt nguồn từ nút mới.

d. Ưu và nhược điểm

Ưu điểm:

- Cây quyết định có thể tạo ra các quy tắc dễ hiểu.
- Cây quyết định thực hiện phân loại mà không cần tính toán nhiều.
- Cây quyết định có thể xử lý cả biến liên tục và phân loại.
- Cây quyết định cung cấp một dấu hiệu rõ ràng về trường nào là quan trọng nhất để dự đoán hoặc phân loại.

Nhược điểm:

- Cây quyết định ít thích hợp hơn cho các nhiệm vụ ước tính trong đó mục tiêu là dự đoán giá trị của một thuộc tính liên tục.
- Cây quyết định dễ mắc lỗi trong các bài toán phân loại với nhiều lớp và một số ví dụ huấn luyện tương đối nhỏ.
- Cây quyết định có thể tốn kém về mặt tính toán để đào tạo. Quá trình phát triển một cây quyết định rất tốn kém về mặt tính toán. Tại mỗi nút, mỗi trường phân tách ứng cử viên phải được sắp xếp trước khi có thể tìm thấy phân tách tốt nhất của nó. Trong một số thuật toán, sự kết hợp của các trường được sử dụng và phải thực hiện tìm kiếm để có trọng số kết hợp tối ưu. Các thuật toán cắt tỉa cũng có thể tốn kém vì nhiều cây con ứng cử viên phải được hình thành và so sánh.

CHƯƠNG 3. CÁC KẾT QUẢ THỰC NGHIỆM

3.1. Bộ dữ liệu

Trong bài dự án này, nhóm chúng tôi chọn bộ dữ liệu có tên Mobile Price Classification. Bộ dữ liệu mô tả về cấu tạo và tính năng của điện thoại (RAM, Bộ nhớ trong, chiều dài chiều cao, có 4G hay không,...)

Bộ dữ liệu này được đăng tải trên trang Kaggle bởi tác giả ABHISHEK SHARMA có 2000 dòng và 22 cột biến. Mỗi dòng chứa thông tin của từng chiếc điện thoại giúp phân tích mối tương quan giữa cấu tạo, chức năng của điện thoại với giá tiền của chúng.

Các biến trong bộ dữ liệu ***train.xlxs***:

Tên biến	Mô tả
battery_power	Mức pin tối đa (mAh)
blue	Có bluetooth hay không
clock_speed	Tốc độ xung nhịp vi xử lí (GHz: gigahertz)
dual_sim	Có hỗ trợ sim kép hay không
fc	độ phân giải camera trước (MP: megapixel)

four_g	có 4G hay không
int_memor	Dung lượng bộ nhớ trong (GB: gigabytes)
m_dep	Độ sâu (dày) của điện thoại (cm)
mobile_wt	Khối lượng của điện thoại
n_cores	Số lượng core của bộ xử lý
pc	độ phân giải của camera chính (sau) (MP: megapixel)
px_height	Số pixel theo chiều dài màn hình
px_width	Số pixel theo chiều ngang màn hình
ram	dung lượng của RAM (MB: megabyte)
sc_h	chiều dài màn hình (cm)
sc_w	chiều rộng màn hình (cm)
talk_time	thời gian đàm thoại (tiếng)

three_g	có 3G hay không
touch_screen	điện thoại cảm ứng hay không
wifi	có wifi hay không
price_range	biến mục tiêu với các mức chi phí có giá trị 0 (thấp), 1 (trung bình), 2 (cao) và 3 (rất cao)

3.2. Tiền xử lý dữ liệu

- Kiểm tra bộ dữ liệu

Input:

```
df = pd.read_excel('train.xlsx')
df
```

Output:

	Unnamed: 0	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g
0	0	842	no	2.2	no	1	no	7	0.6	188	...	20.0	756	2549	9	7	19	no
1	1	1021	yes	0.5	yes	0	yes	53	0.7	136	...	905.0	1988	2631	17	3	7	yes
2	2	563	yes	0.5	yes	2	yes	41	0.9	145	...	1263.0	1716	2603	11	2	9	yes
3	3	615	yes	2.5	no	0	no	10	0.8	131	...	1216.0	1786	2769	16	8	11	yes
4	4	1821	yes	1.2	no	13	yes	44	0.6	141	...	1208.0	1212	1411	8	2	15	yes
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	1995	794	yes	0.5	yes	0	yes	2	0.8	106	...	1222.0	1890	668	13	4	19	yes
1996	1996	1965	yes	2.6	yes	0	no	39	0.2	187	...	915.0	1965	2032	11	10	16	yes
1997	1997	1911	no	0.9	yes	1	yes	36	0.7	108	...	868.0	1632	3057	9	1	5	yes
1998	1998	1512	no	0.9	no	4	yes	46	0.1	145	...	336.0	670	869	18	10	19	yes
1999	1999	510	yes	2.0	yes	5	yes	45	0.9	168	...	483.0	754	3919	19	4	2	yes

2000 rows × 22 columns

Input:

```
print('Tổng các giá trị null theo từng cột\n')
df.isnull().sum().sort_values()
```

Output:

```
Tổng các giá trị null theo từng cột

battery_power    0
touch_screen     0
three_g          0
talk_time        0
sc_w             0
sc_h             0
ram              0
px_width         0
wifi             0
price_range      0
mobile_wt        0
int_memory       0
four_g           0
fc               0
dual_sim         0
blue             0
n_cores          0
px_height        1
m_dep            1
clock_speed      2
pc               3
dtype: int64
```

- Loại bỏ cột Unnamed, và xóa bỏ những dòng bị thiếu dữ liệu

Input:

```
df = df.drop(['Unnamed: 0'], axis = 1)
```

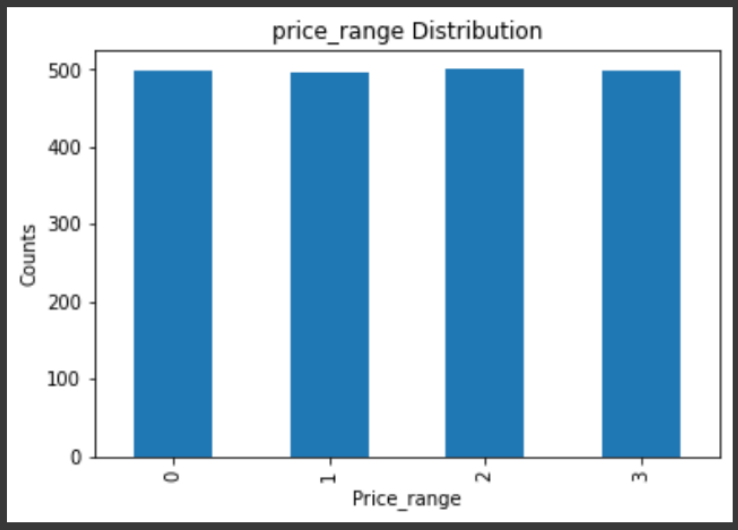
```
df = df.dropna(inplace=False)
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1993 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  1993 non-null   int64  
 1   blue           1993 non-null   object 
 2   clock_speed    1993 non-null   float64
 3   dual_sim       1993 non-null   object 
 4   fc             1993 non-null   int64  
 5   four_g         1993 non-null   object 
 6   int_memory     1993 non-null   int64  
 7   m_dep          1993 non-null   float64
 8   mobile_wt      1993 non-null   int64  
 9   n_cores        1993 non-null   int64  
 10  pc             1993 non-null   float64
 11  px_height      1993 non-null   float64
 12  px_width       1993 non-null   int64  
 13  ram            1993 non-null   int64  
 14  sc_h           1993 non-null   int64  
 15  sc_w           1993 non-null   int64  
 16  talk_time      1993 non-null   int64  
 17  three_g        1993 non-null   object 
 18  touch_screen   1993 non-null   object 
 19  wifi           1993 non-null   object 
 20  price_range    1993 non-null   int64  
dtypes: float64(4), int64(11), object(6)
memory usage: 342.5+ KB
```

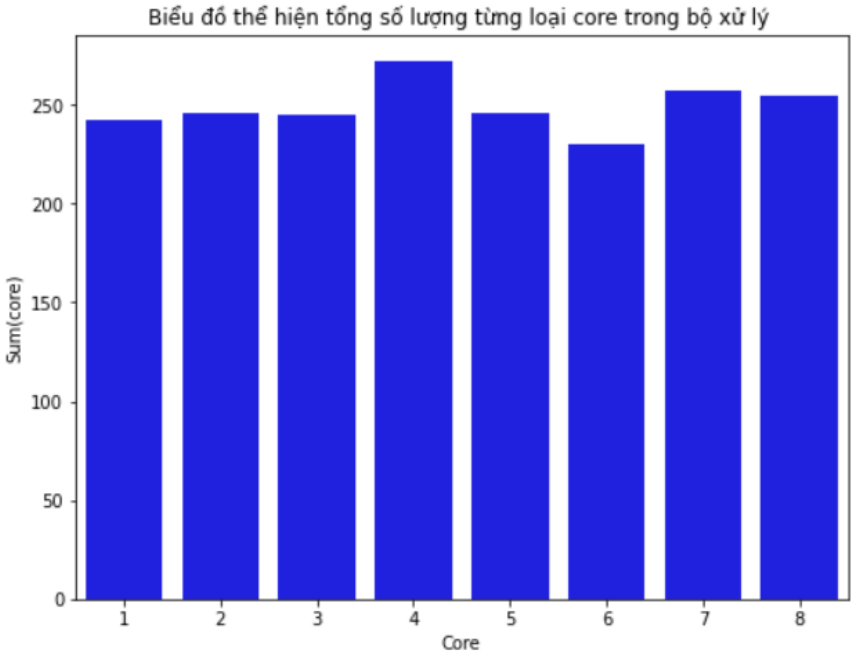
3.3. Biểu diễn trực quan dữ liệu

```
[ ]  plt.title('price_range Distribution')
     df['price_range'].value_counts().sort_index().plot(kind='bar')
     plt.xlabel('Price_range')
     plt.ylabel ('Counts')
     plt.show()
```



Dữ liệu đưa vào phân chia đều cho các loại điện thoại (chi phí thấp, trung bình, cao, rất cao) với mỗi loại đều có khoảng 500 cái.

```
plt.figure(figsize=(8, 6))

sbn.countplot(df['n_cores'], color ='b')

plt.title('Biểu đồ thể hiện tổng số lượng từng loại core trong bộ xử lý ',fontsize = 12)

plt.xlabel('Core')

plt.ylabel('Sum(core)')

plt.show()
```

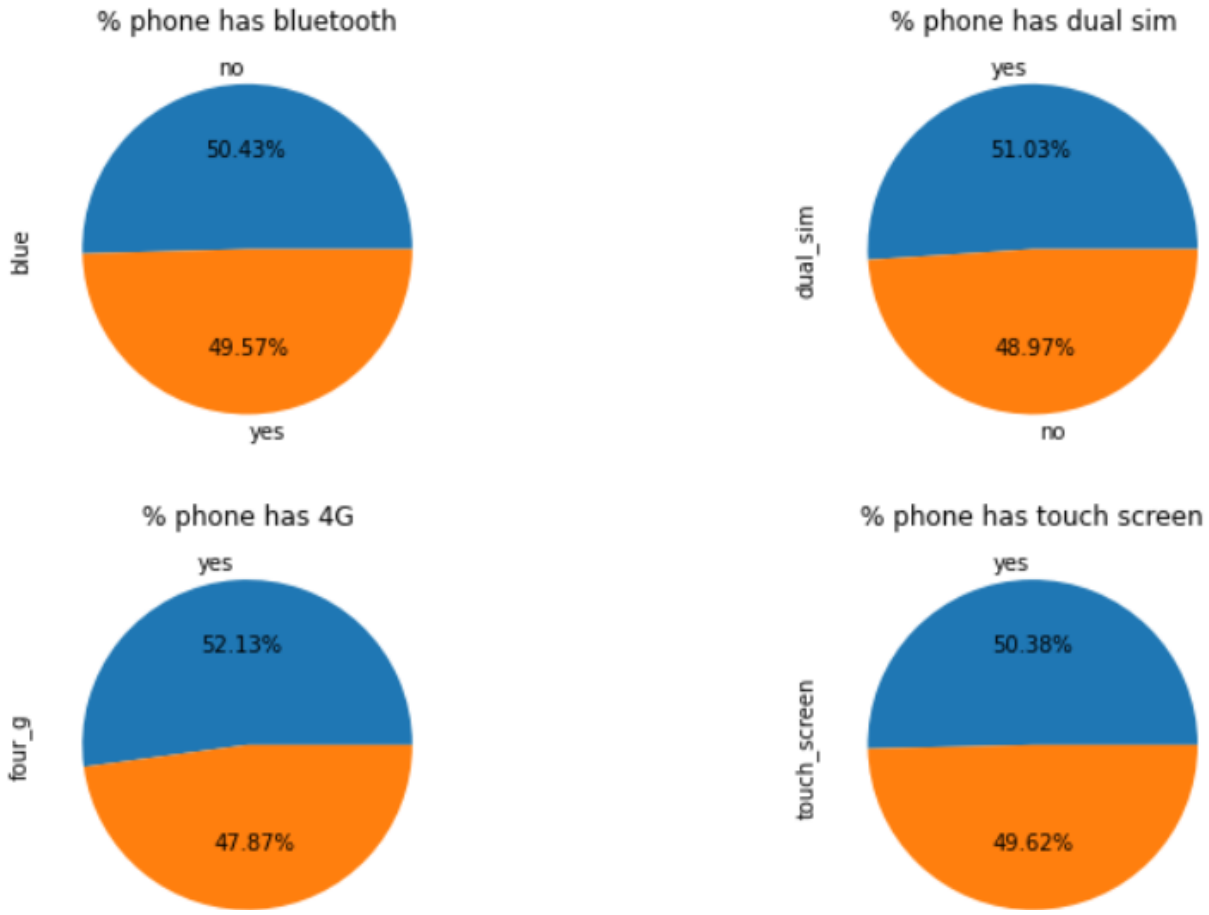


Nhìn vào biểu đồ ta thấy loại điện có 4 core chiếm số lượng nhiều nhất và loại điện có 6 core chiếm số lượng ít nhất trong tổng số 8 loại điện thoại có từ 1-> 8 core.

Input:

```
plt.subplot(3, 2, 1)
ax2 = df['blue'].value_counts().plot(kind = 'pie',figsize=(12,12),autopct='%1.2f%%')
ax2.set_title('% phone has bluetooth', fontsize = 12)

plt.subplot(3, 2, 2)
ax3 = df['dual_sim'].value_counts().plot(kind = 'pie',figsize=(12,12),autopct='%1.2f%%')
ax3.set_title('% phone has dual sim', fontsize = 12)

plt.subplot(3, 2, 3)
ax4 = df['four_g'].value_counts().plot(kind = 'pie',figsize=(12,12),autopct='%1.2f%%')
ax4.set_title('% phone has 4G', fontsize = 12)

plt.subplot(3, 2, 4)
ax5 = df['touch_screen'].value_counts().plot(kind = 'pie',figsize=(12,12),autopct='%1.2f%%')
ax5.set_title('% phone has touch screen', fontsize = 12)
plt.show()
```

Output:

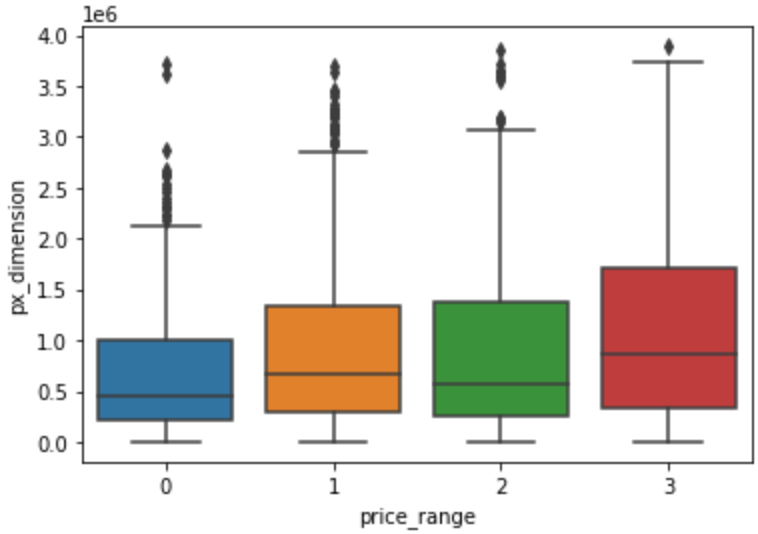


Bốn biểu đồ tròn thể hiện tỷ lệ điện thoại có tích hợp bluetooth, sim ghép, 4G và cảm ứng hay không

Đầu tiên nhóm sẽ tạo thêm một cột nữa, cột này tên là px-dimension, nó thể hiện kích thước của độ phân giải px, được tính bằng cách lấy px_heigh(chiều cao px) nhân với px_width(độ rộng của px).

Sau đó nhóm sẽ dùng biểu đồ hộp để xem thử có xuất hiện outliers giữa các mức độ phạm vi giá so với kích thước độ phân giải.

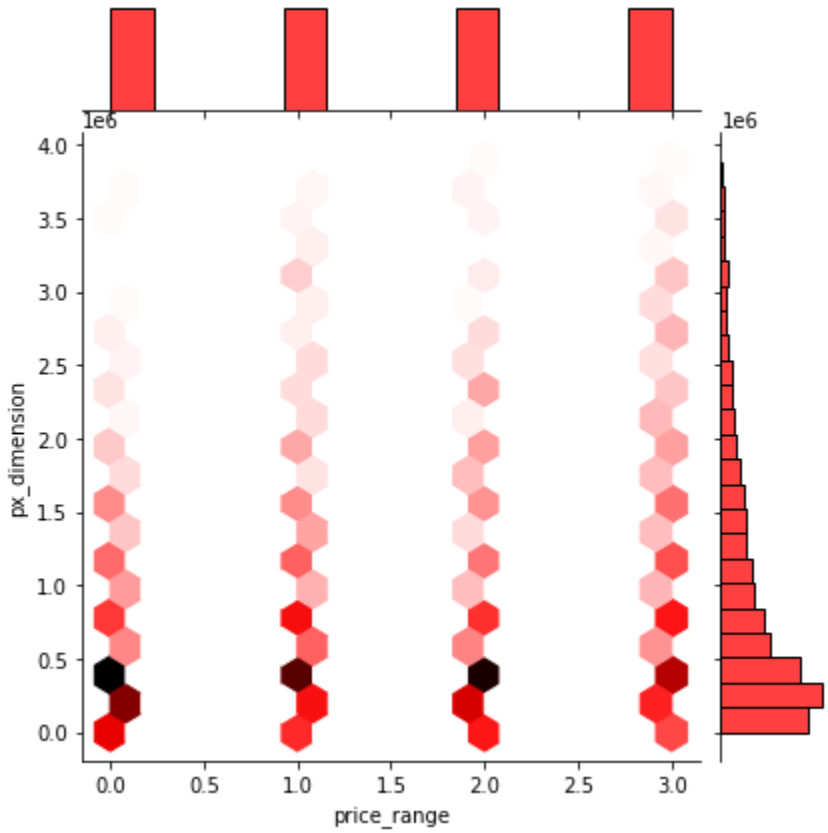
```
df["px_dimension"] = df_["px_height"] * df["px_width"]
sns.boxplot(x = "price_range", y = "px_dimension", data = df_train)
plt.show()
```



Chúng tôi nhận thấy được xuất hiện khá nhiều outlier ở cả 4 mức độ.

Tiếp tới chúng tôi sẽ xét tính tương quan giữa các mức độ phạm vi giá so với kích thước độ phân giải hay không

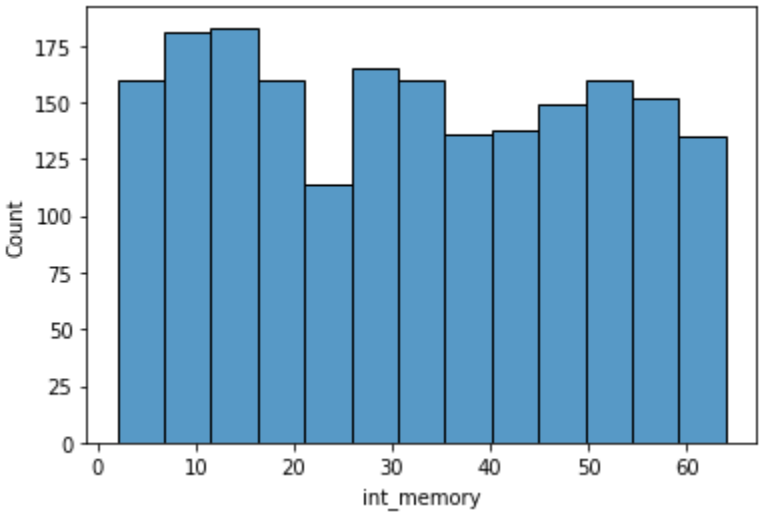
```
sns.jointplot(x = "price_range", y = "px_dimension", data = df_train,kind ='hex',color ='r')
plt.show()
```



Nhóm thấy rằng kích thước của độ phân giải không ảnh hưởng tới mức độ phạm vi giá.

Tiếp tới chúng tôi sẽ xem thử độ phân bổ giữa các giá trị trong bộ nhớ

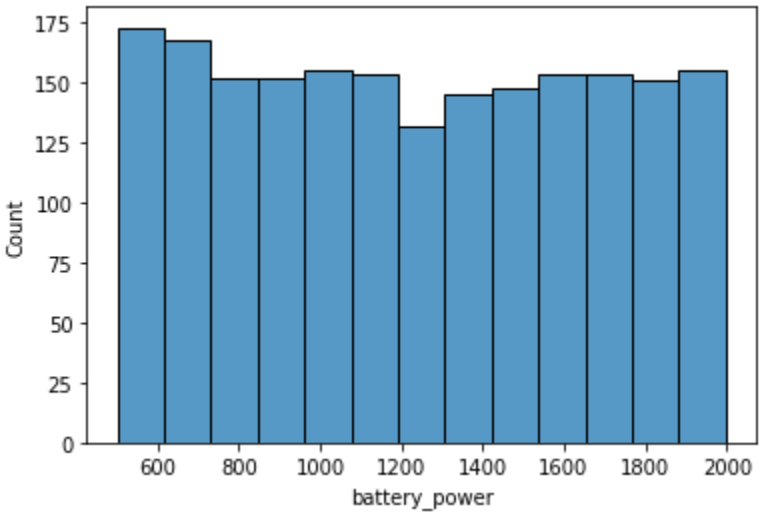
```
sbn.histplot(x="int_memory", data = df)
```



Có thể thấy rằng lượng người dùng bộ nhớ phân phối khá đều, chỉ có trong khoảng 0-20 thì số lượng người dùng có vẻ nhỉnh hơn 1 xíu so với các khoảng còn lại.

Tiếp tới chúng tôi sẽ xem thử độ phân bổ của các giá trị năng lượng pin

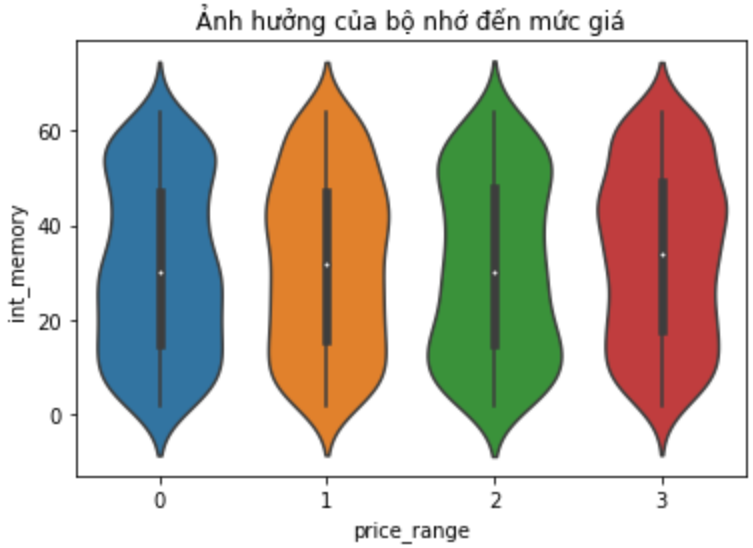
```
sbn.histplot(x="battery_power", data = df)
```



Cũng như bộ nhớ, tổng năng lượng pin cũng được phân phối khá đều nhau và khoảng 0-800 là khoảng có số lượng nhiều hơn so một chút so với các khoảng còn lại.

Tiếp theo chúng tôi sẽ xét tới ảnh hưởng của bộ nhớ đến giá

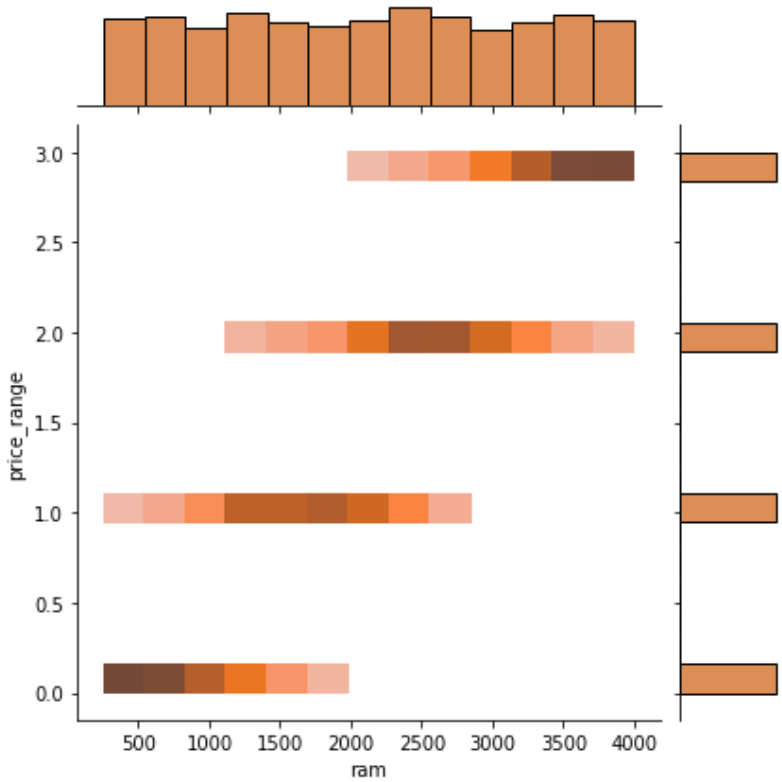
```
sbn.violinplot(y="int_memory", x="price_range", data=df)
plt.title('Ảnh hưởng của bộ nhớ đến mức giá')
```



Có thể thấy rằng ở cả 4 mức độ, lượng bộ nhớ đều phân bổ khá đều nhau. Điều này cho thấy rằng lượng bộ nhớ có thể không ảnh hưởng tới mức độ giá.

Và chúng tôi xem thử lượng ram có ảnh hưởng tới mức độ phạm vi giá hay không

```
sbn.jointplot(x='ram',y='price_range',data=df,kind ='hist',color ='chocolate')
```



Có thể thấy rằng ram ảnh hưởng khá nhiều tới mức độ phạm vi giá

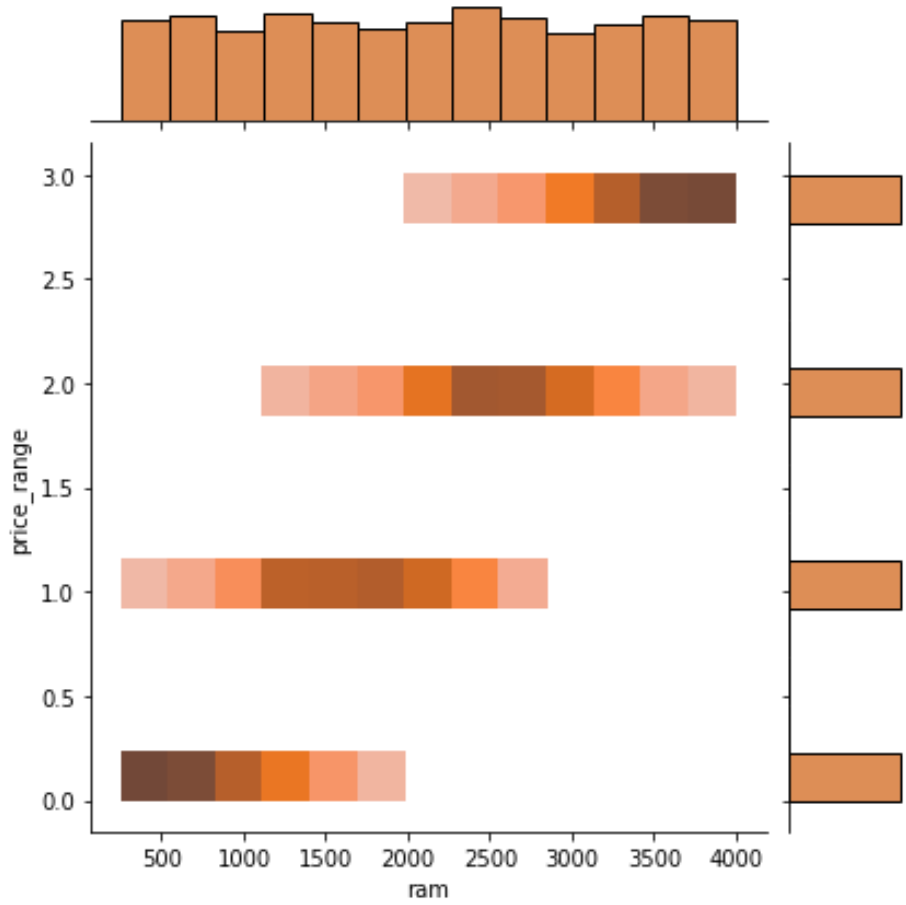
- Ở mức độ 0 thì lượng ram hoàn toàn khá thấp(nhiều nhất là trong khoảng từ 500 tới 1000 và không vượt quá 2000)

- Sang tới mức độ 1 thì lượng ram sẽ có cải thiện hơn, lớn hơn so với mức độ 0 (nhiều nhất trong khoảng 1000 tới 2500 và không vượt quá 3000)
- Tới mức độ 2 thì lại hơn so với mức 1(nhiều nhất nằm trong khoảng 2000 đến 3000)
- Tới mức độ 3 là mức độ mà lượng ram nhiều nhất trong tất cả các mức độ

Từ đó có thể thấy rằng mức độ phạm vi giá càng tăng thì lượng ram cũng tăng lên hay nói cách khác ram và mức phạm vi giá có tương quan tuyến tính thuận

Sau đó chúng tôi xem thử liệu lượng ram có ảnh hưởng đến mức độ phạm vi giá hay không

```
sbn.jointplot(x='ram',y='price_range',data=df,kind ='hist',color ='chocolate')
```



Có thể thấy rằng ram ảnh hưởng khá nhiều tới mức độ phạm vi giá.

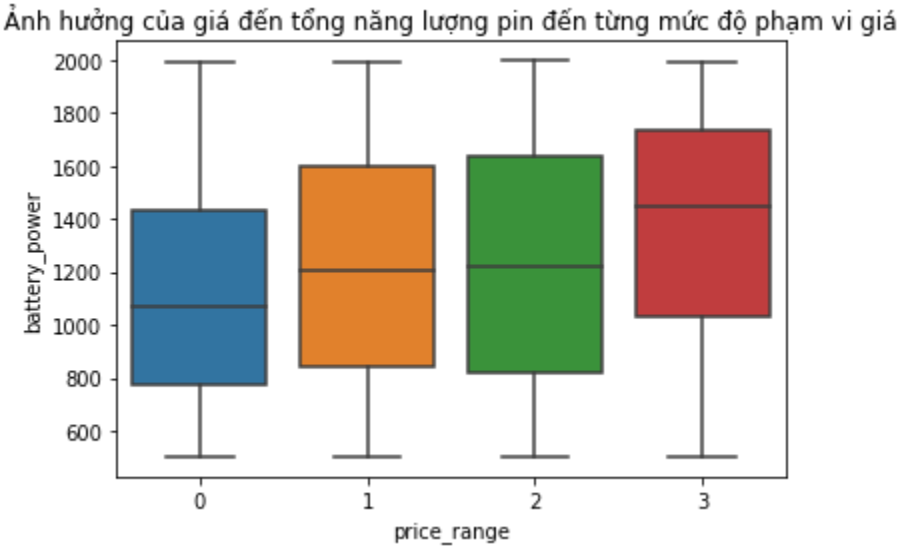
- Ở mức độ 0 thì lượng ram hoàn toàn khá thấp(nhiều nhất là trong khoảng từ 500 tới 1000 và không vượt quá 2000).
- Sang tới mức độ 1 thì lượng ram sẽ có cải thiện hơn, lớn hơn so với mức độ 0 (nhiều nhất trong khoảng 1000 tới 2500 và không vượt quá 3000).

- Tới mức độ 2 thì lại hơn so với mức 1 (nhiều nhất nằm trong khoảng 2000 đến 3000).
- Tới mức độ 3 là mức độ mà lượng ram nhiều nhất trong tất cả các mức độ.

Từ đó có thể thấy rằng mức độ phạm vi giá càng tăng thì lượng ram cũng tăng lên hay nói cách khác ram và mức phạm vi giá có tương quan tuyến tính thuận

Tiếp tới chúng tôi sẽ xem thử ảnh hưởng của pin đến từng mức độ phạm vi giá

```
sbn.boxplot(x="price_range", y="battery_power", data=df)
plt.title('Ảnh hưởng của giá đến tổng năng lượng pin đến từng mức độ phạm vi giá')
```



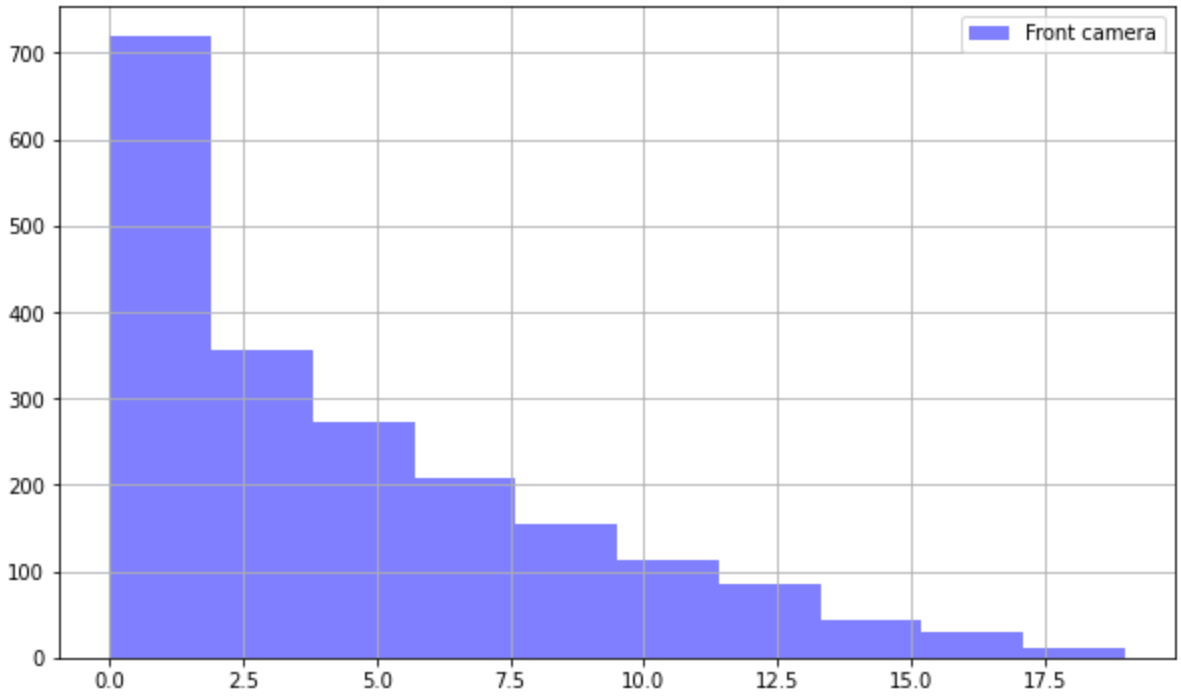
Có thể thấy ở cả 4 mức độ phạm vi đều không xuất hiện outlier.

- Ở mức độ 0 mức năng lượng pin nằm trong khoảng từ 800 đến 1400.
- Sang tới mức độ 1 mức năng lượng pin lớn hơn so với mức độ 0: nằm trong khoảng từ hơn 800 đến 1600.
- Ở mức độ 2 mức năng lượng pin nhỉnh hơn một chút so với mức độ 1: nằm trong khoảng từ hơn 800 đến hơn 1600.
- Tới mức độ 3 mức năng lượng pin hơn hoàn toàn so với 3 mức độ còn lại: nằm trong khoảng từ hơn 1000 đến 1700

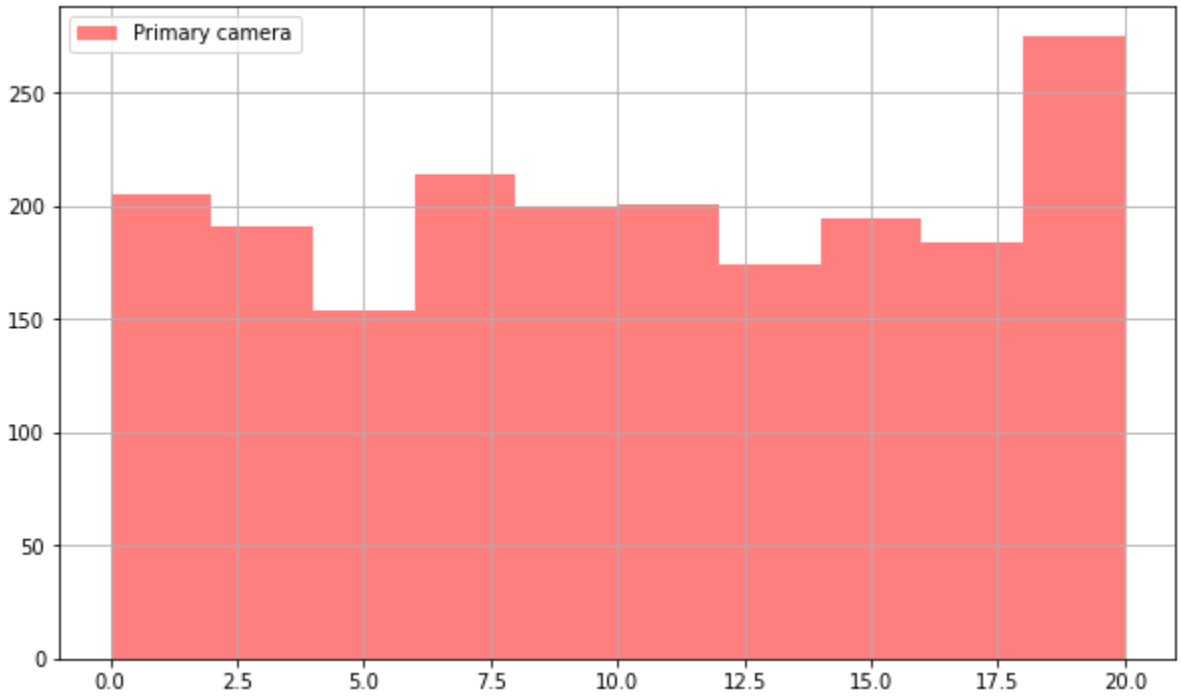
Từ đó chúng tôi kết luận rằng mức năng lượng pin và phạm vi giá cũng có tương quan tuyến tính thuận

Bây giờ chúng tôi sẽ so sánh thử 2 cam trước và cam chính

```
plt.figure(figsize=(10,6))
df['fc'].hist(alpha=0.5,color='blue',label='Front camera')
plt.legend()
plt.show()
```



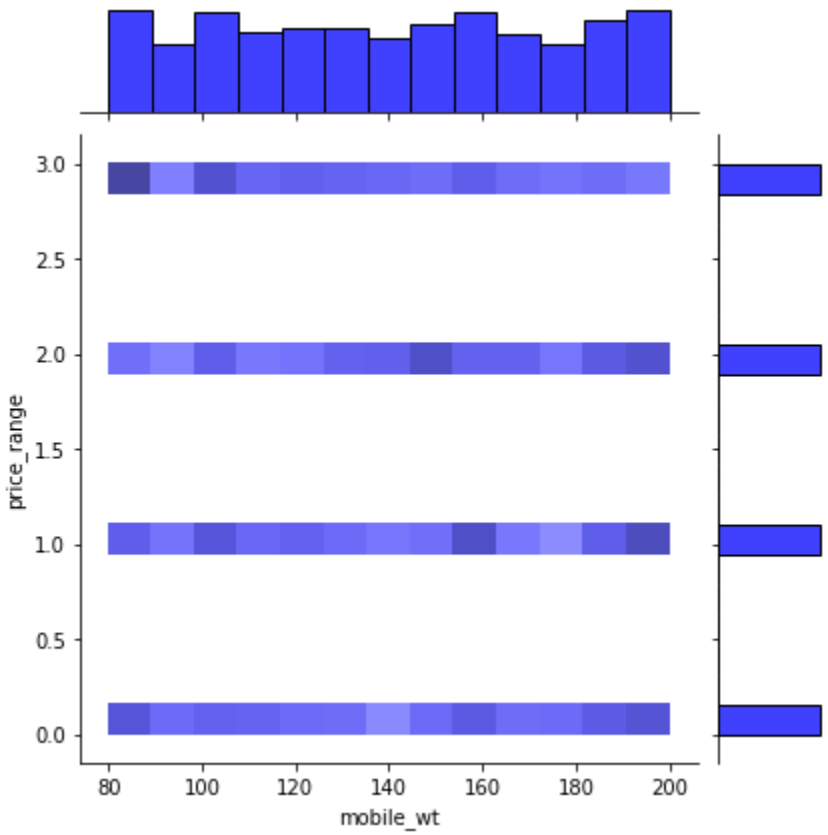

```
plt.figure(figsize=(10,6))
df['pc'].hist(alpha=0.5,color='red',label='Primary camera')
plt.legend()
```



Từ 2 hình trên chúng tôi có thể kết luận rằng độ phân giải của cam chính hoàn toàn lớn hơn, thậm chí là lớn hơn rất nhiều so với cam trước.

Chúng tôi cũng sẽ xem thử trọng lượng pin có làm ảnh hưởng tới mức giá hay không

```
sbn.jointplot(x='mobile_wt',y='price_range',data=df,kind='hist',color='blue')
```



Bằng mắt thường sau khi vẽ biểu đồ, nhóm em có thể kết luận rằng trọng lượng của điện thoại không ảnh hưởng đến mức độ phạm vi giá.

Và chúng tôi cũng sẽ xem thử độ phân phối của tốc độ mà bộ vi xử lý thực hiện

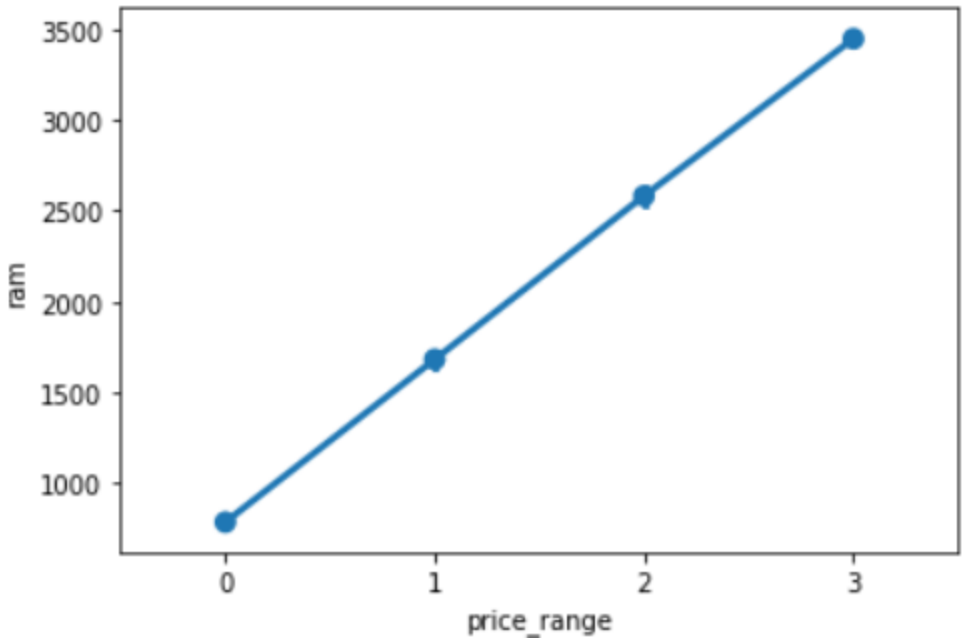
```
plt.figure(figsize=(10,6))
df['clock_speed'].value_counts().plot(kind='bar')
plt.title('Tốc độ mà bộ vi xử lý thực hiện các hướng dẫn')
```

Có thể thấy rằng tốc độ mà bộ vi xử lý ở mức độ 0.5 là nhiều nhất, điều này cho thấy rằng tốc độ 0.5 là một tốc độ khá đại trà. Sang tới các tốc độ khác thì phân phối khá đều nhau. Chỉ có duy nhất tốc độ xử lý 3.0 thì ít hơn, điều này cho thấy rằng tốc độ xử lý 3.0 có thể hiếm hơn so với các mức độ khác.

Input:

```
[165] sbn.pointplot(y="ram", x="price_range", data=df) # mối tương quan giữa ram và price_range
      plt.show()
```

Output:

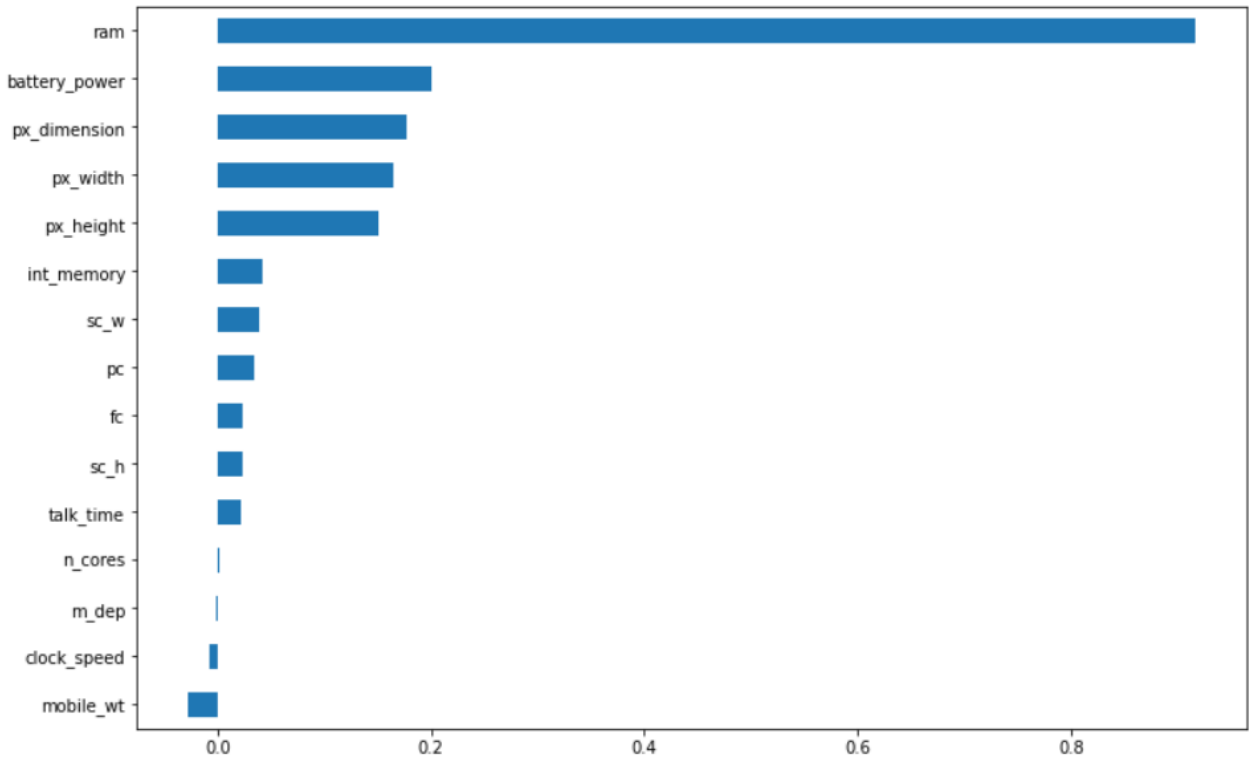


Hình trên cho thấy, với dung lượng ram tại khoảng gần 1000 thì giá trị điện thoại nằm ở mức 0, và dung lượng ram cùng lớn thì mức giá càng tăng và đạt cao tại 3500.

Input:

```
[166] df.corr().price_range.sort_values()[:-1].plot(kind='barh', figsize = (12,8))
      # mối tương quan giữa price_range và các biến
```

Output:



Biểu đồ so sánh và xếp hạng độ tương quan của các biến độc lập với biến phụ thuộc.

3.4 Phân tích đa biến

Chúng tôi sẽ kiểm định theo phương sai giữa 2 biến px_height và px-width để xem thử phương sai của chiều cao và độ rộng của px có gì khác nhau không và chúng có sự độc lập với nhau không

```
# kiểm định phương sai chiều cao và độ rộng của px
#    H0: Muy['px_height'] =  Muy['px_width']
#    Ha: Muy['px_height'] #  Muy['px_width']
```

Đầu tiên chúng tôi sẽ chuyển đổi 2 cột px_height và px_width sang dạng dữ liệu array, riêng với cột px_height, vì dữ liệu số trong cột ở dạng 0.0 nên phải thêm 1 bước nữa thì mới chuyển được tất cả dữ liệu của cột đó về dạng số nguyên

```
height1 = df['px_height'].values
height = np.array(list(map(np.int_, height1)))
width  = df['px_width'].values
```

Tiếp tới chúng tôi sẽ xem thử phương sai của 2 biến có giống nhau hay khác nhau

```
print(f'Var(life_1) = {height.var()}, Var(life_2) = {width.var():.2f}')
print('==> Phương sai 2 mẫu', 'BẰNG NHAU' if (height.var() == width.var()) else 'KHÁC NHAU')
```

```
Var(life_1) = 4.251402821718476e+34, Var(life_2) = 186702.96
==> Phương sai 2 mẫu KHÁC NHAU
```

Tiếp theo chúng tôi sẽ dùng ttest_ind đã có sẵn trong thư viện của python tìm giá trị t-test giữa 2 biến

```
from scipy.stats import ttest_ind
df   = len(life1_array) - 1
t, p = ttest_ind(life1_array, life2_array, equal_var = False)
```

Sau khi tìm t- test thì chúng tôi sẽ tính hàm phân phối tích lũy nghịch đảo (nhóm đặt tên biến đó là cv) dựa theo độ dài của biến height, biến t-test vừa tìm được, khoảng tin cậy và bậc tự do

Sau đó chúng tôi sẽ xét: Nếu trị tuyệt đối của t lớn hơn hoặc bằng cv thì nhóm bác bỏ H0. Ngược lại thì nhóm chấp nhận H0

Từ đó chúng tôi kiểm định được rằng là không bác bỏ H0, tức là

Không có sự khác biệt giữa chiều cao và độ rộng của px, tức là nói một cách đơn giản là trung bình giữa chiều cao và độ rộng của các máy là giống nhau. Nhưng chiều cao và

```
import scipy.stats as stats
cv = stats.t.ppf(confidence_level, len_height)
print(f'Giá trị tới hạn = {cv:.2f}; Trị thống kê t = {t:2f}')

if (abs(t) >= cv):
    print('Vì trị thống kê t NẰM TRONG miền bác bỏ cho nên BÁC BỎ H0 ==> Muy[''px_height''] #  Muy[''px_width'']')
else:
    print('Vì trị thống kê t NẰM NGOÀI miền bác bỏ cho nên KHÔNG BÁC BỎ H0 ==> H0: Muy[''px_height''] =  Muy[''px_width'']')
```

```
Giá trị tới hạn = 1.65; Trị thống kê t = -1.000000
Vì trị thống kê t NẰM NGOÀI miền bác bỏ cho nên KHÔNG BÁC BỎ H0 ==> H0: Muy[px_height] =  Muy[px_width]
```

độ rộng của px này hoàn toàn không có mối liên hệ với nhau, tức là chiều cao của px có thay đổi thì độ rộng chưa chắc đã thay đổi và ngược lại

Tiếp theo chúng tôi sẽ dùng kiểm định ANOVA để xem thử có sự khác biệt giữa cam trước và cam chính hay không

```
## Các giả thuyết kiểm định
##    H0: Muy[fc] = Muy[pc]
##    Ha: Có sự khác biệt giữa các cam trước và cam chính
```

Nhóm dùng ANOVA f_oneway để kiểm định, dùng hàm f_oneway có sẵn trên python

```
import scipy.stats as stats
f, p = stats.f_oneway(fc1, pc1)
alpha = .05
```

Sau đó sẽ lấy ra giá trị p để so sánh với alpha và đưa ra kết luận về việc có bác bỏ hay chấp nhận H0

```
if (p < alpha):
    print('Trị số p < alpha cho nên bác bỏ H0 ==> hậu kiểm Tukey HSD')
else:
    print('KHÔNG bác bỏ H0 ==> Muy[fc] = Muy[pc](sự ĐỘC LẬP)')
```

```
Trị số p = 0.08, alpha = 0.05
KHÔNG bác bỏ H0 ==> Muy[fc] = Muy[pc](sự ĐỘC LẬP)
```

Từ đó chúng tôi kết luận được rằng: không có sự khác biệt giữa 2 cam, tức là nói một cách đơn giản là trung bình giữa cam trước và cam chính của các máy là giống nhau. Nhưng 2 camera này hoàn toàn không có mối liên hệ với nhau, tức là cam trước có bị vấn đề gì thì cũng không quá ảnh hưởng tới cam chính và ngược lại

Chúng tôi sẽ dùng kiểm định z để xem thử lượng pin trung bình của điện thoại có lớn hơn 1250mAh hay không

```
# Kiểm định z
# H0: Muy_babattery_power <= 1250
# Ha: Muy_babattery_power > 1250
```

Trước khi kiểm định được z thì chúng tôi sẽ tiến hành tính z

```
x = df['battery_power'].mean()
dolechchuan = df['battery_power'].std()
n = len(df['battery_power'])
z= (x - 1250)/(dolechchuan/math.sqrt(n))
```

Sau đó tạo hàm để xét bảng z table và xuất giá trị z ra

```
def zDistribution(side, lower, upper):
    import scipy.stats as st

    side = side.lower()

    if (side == 'left'):
        p = st.norm.cdf(lower)
    elif (side == 'right'):
        p = 1 - st.norm.cdf(upper)
    else:
        p = st.norm.cdf(upper) - st.norm.cdf(lower)
    return (p)

## Hàm hiển thị xác suất
def displayZ(side, lower, upper, p):
    side = side.lower()
    lower = str(lower)
    upper = str(upper)
    if (side == 'left'):
        s = 'P(Z < ' + lower + ') = '
    elif (side == 'right'):
        s = 'P(' + upper + ' < Z) = '
    else:
        s = 'P(' + lower + ' < Z < ' + upper + ') = '
    return (s + str('%.4f' %p))
```

Tiếp tới tính p-value của z và so sánh với alpha để kiểm định

```
side  = 'right'
lower = None
upper = z
p = zDistribution(side, lower, upper)
print(displayZ(side, lower, upper, p))
if (p < alpha):
  print('Bác bỏ H0')
else:
  print('Chấp nhận H0')
```

```
P(-1.1685184695685393 < Z) = 0.8787
Chấp nhận H0
```

Vậy trung bình lượng pin của các điện thoại không vượt quá 1250 mAh

3.5. PCA

- Chuyển đổi dữ liệu sang kiểu số và phân chia bộ dữ liệu thành bộ dữ liệu các biến độc lập (X) và chuỗi target (y) - bỏ đi cột target trong bộ dữ liệu gốc, và lưu target vào một mảng mới.

Input:

```
df =  pd.get_dummies(df)
y = list(df['price_range'])
X = df.drop(['price_range'],axis=1)
X
```

Output:

	battery_power	clock_speed	fc	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	...	dual_sim_no	dual_sim_yes	four_g_no	four_g_yes	three_g_no
0	842	2.2	1	7	0.6	188	2	2.0	20.0	756	...	1	0	1	0	1
1	1021	0.5	0	53	0.7	136	3	6.0	905.0	1988	...	0	1	0	1	0
2	563	0.5	2	41	0.9	145	5	6.0	1263.0	1716	...	0	1	0	1	0
3	615	2.5	0	10	0.8	131	6	9.0	1216.0	1786	...	1	0	1	0	0
4	1821	1.2	13	44	0.6	141	2	14.0	1208.0	1212	...	1	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	794	0.5	0	2	0.8	106	6	14.0	1222.0	1890	...	0	1	0	1	0
1996	1965	2.6	0	39	0.2	187	4	3.0	915.0	1965	...	0	1	1	0	0
1997	1911	0.9	1	36	0.7	108	8	3.0	868.0	1632	...	0	1	0	1	0
1998	1512	0.9	4	46	0.1	145	5	5.0	336.0	670	...	1	0	0	1	0
1999	510	2.0	5	45	0.9	168	6	16.0	483.0	754	...	0	1	0	1	0

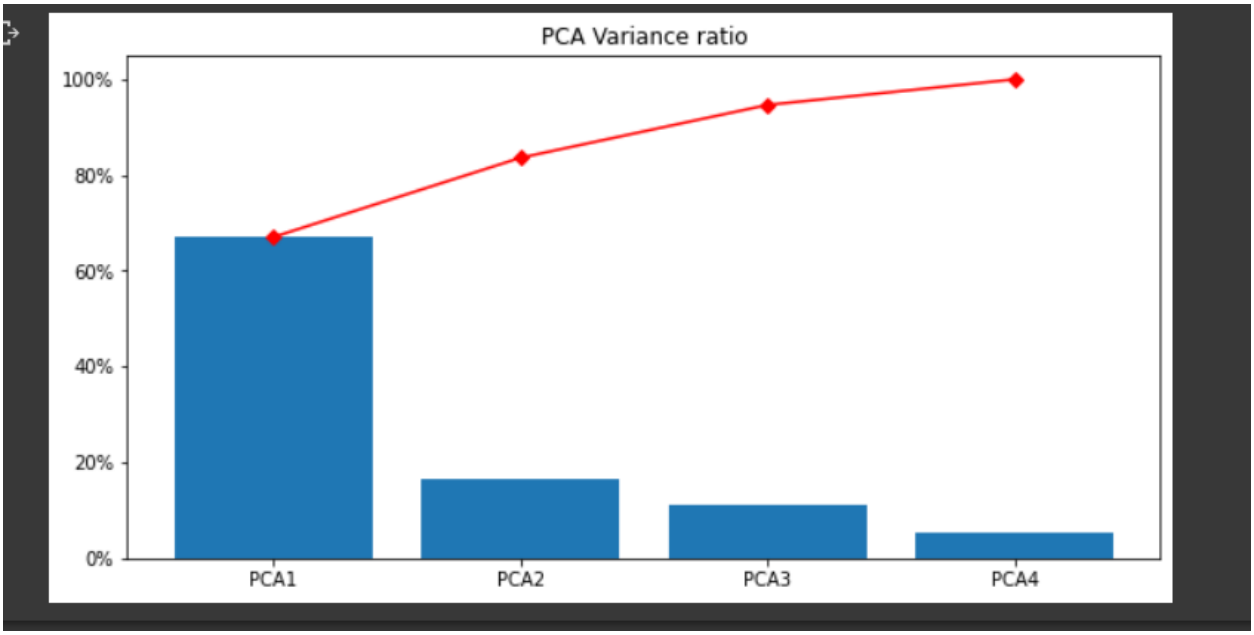
1993 rows × 26 columns

- Tìm số feafures tốt bằng cách sử dụng độ đo phương sai tích lũy

Input:

```
variance_ratio_cumul = list(pca.explained_variance_ratio_.cumsum() * 100)
variance_ratio = list(pca.explained_variance_ratio_ * 100)
fig, ax = plt.subplots(figsize=(10,5))
ax.plot([f"PCA{i}" for i in range(1,len(variance_ratio_cumul)+1)], variance_ratio_cumul, "r", marker="D", label="Variance ratio cumul")
plt.bar(range(len(variance_ratio_cumul)), variance_ratio, label="Variance ratio")
plt.title("PCA Variance ratio")
ax.yaxis.set_major_formatter(PercentFormatter())
plt.show()
```

Output:



-> Để lấy được ~100% thông tin từ bộ dữ liệu thì cần dùng từ 4 features

- Áp dụng giảm chiều PCA với k = 4

```
pca = PCA(n_components=4)
pca.fit(X)
```

- Gán columns cho các cột và nối thêm cột phân lớp

Input:

```
X_pca = pca.transform(X)

df_pca = pd.DataFrame(X_pca, columns=['PC 1','PC 2','PC 3','PC 4'])
df_pca['price_range'] = y
df_pca
```

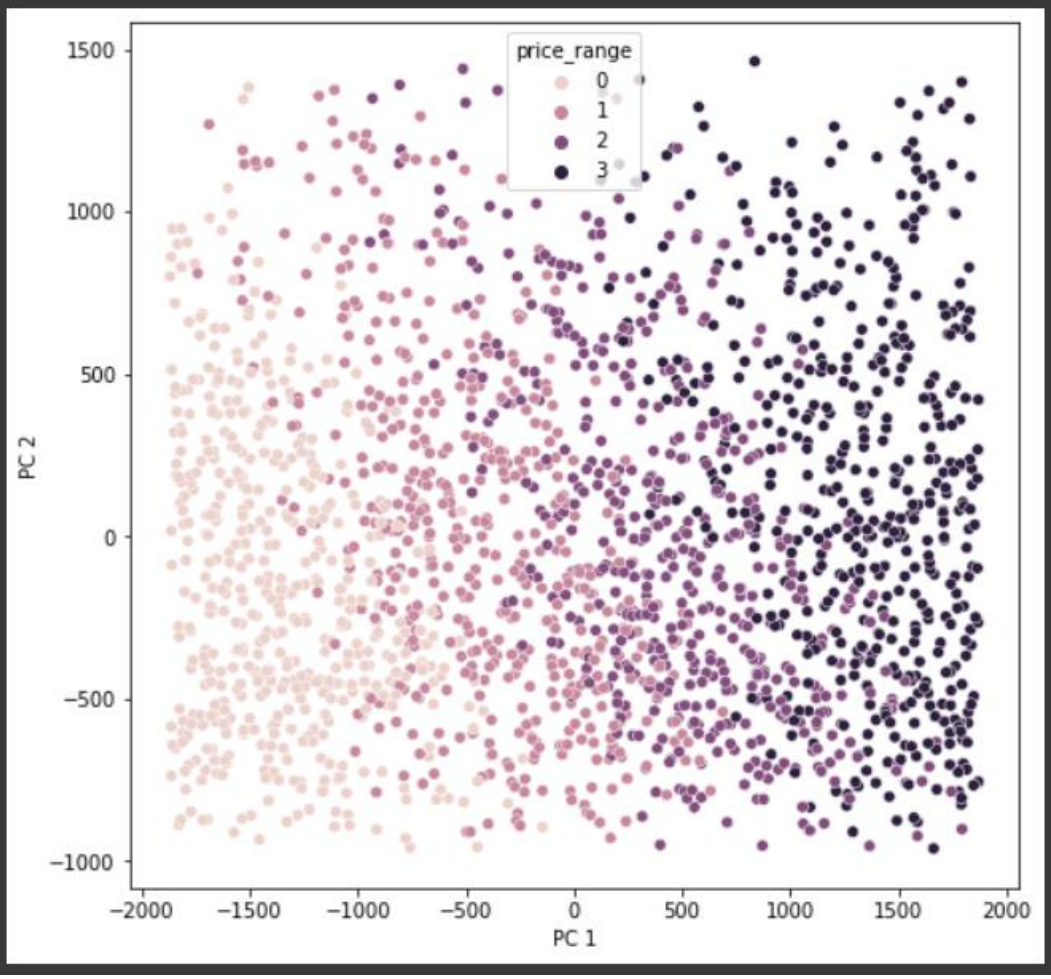
Output:

	PC 1	PC 2	PC 3	PC 4	price_range
0	428.923059	-795.371636	-391.587768	56.546944	1
1	503.710962	696.569301	-232.920260	344.859751	2
2	472.410578	765.078638	-678.109412	-112.090537	2
3	638.852022	780.625463	-628.612370	-28.562571	2
4	-719.967620	380.879197	591.407734	-393.939618	1
...	...	...	...	...	...
1988	-1462.113665	845.152495	-453.646187	62.426556	0
1989	-95.788637	691.820032	711.061953	353.473637	2
1990	929.359604	434.295278	665.979778	135.868990	3
1991	-1254.237601	-630.386607	284.055066	-191.933613	0
1992	1794.784073	-455.263858	-715.787441	-282.934123	3

1993 rows × 5 columns

- Biểu diễn trực quan dữ liệu sau khi áp dụng PCA

```
plt.figure(figsize = (8, 8))
sbn.scatterplot(x = "PC 1", y = "PC 2", data = df_pca, hue = 'price_range')
plt.show()
```



3.6. Phân lớp

3.6.1 Mục tiêu

Ở bước phân lớp này, mục tiêu của đồ án là dự đoán điểm phạm vi giá của các thiết bị trong bộ dữ liệu. Kết quả của giá trị dự đoán là các giá trị 0,1,2,3 tương ứng với các loại điên thoại chi phí thấp, trung bình, cao và rất cao.

Sau khi đã phân lớp dữ liệu bằng các thuật toán khác nhau, chúng tôi sẽ tiến hành đánh giá các thuật toán phân lớp này để xem thuật toán nào là phù hợp và cho ra kết qủa phân lớp tốt.

3.6.2. Tiến hành

```
X=  df_pca.drop('price_range',axis=1)
y = df_pca['price_range']
```

Tiến hành train bộ dữ liệu, dùng train_test_split để chia bộ dữ liệu gốc mà đã tách biến target ra khỏi, và chuỗi biến target ra thành hai bộ train và test với tỉ lệ 80:20 (**test_size = 0.2**).

```
# Bắt đầu lấy mẫu phân tầng
X_train,X_test,y_train,y_test  = train_test_split(X,y,test_size=0.2,random_state= 42)
```

a. K-Nearest Neighbors

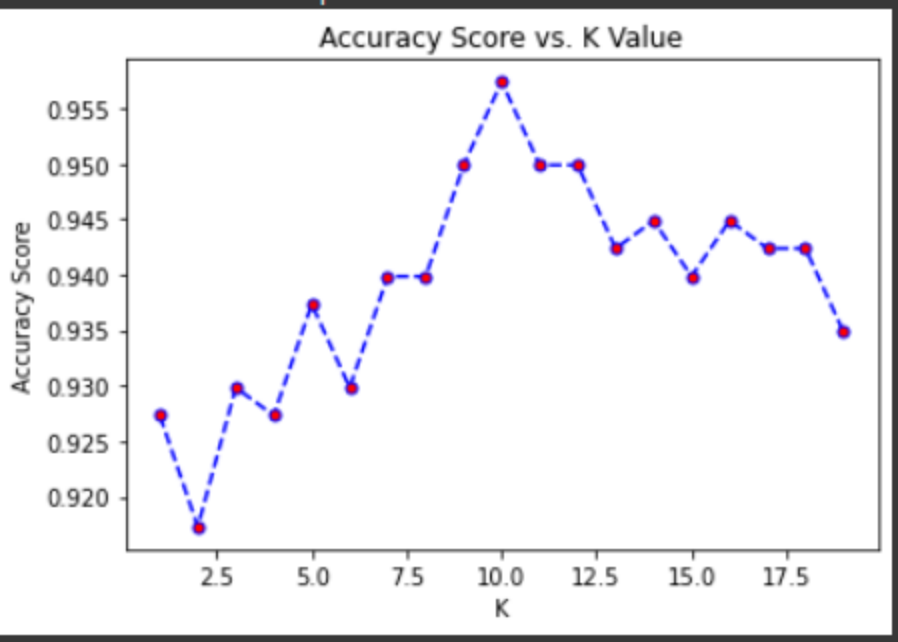
- Tiến hành tìm ra k tốt nhất cho thuật toán KNN

Input:

```
accuracy_knn = []

for i in range(1,20):
  model_KN = KNeighborsClassifier(n_neighbors=i)
  model_KN.fit(X_train,y_train)
  y_pred = model_KN.predict(X_test)
  accuracy_knn.append(accuracy_score(y_test, y_pred))

print(f'k = {accuracy_knn.index(max(accuracy_knn))+1} cho ra kết quả tốt nhất {max(accuracy_knn)} ')

plt.plot(range(1,20),accuracy_knn,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=5)
plt.title('Accuracy Score vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy Score')
plt.show()
```

Output:

```
k = 10 cho ra kết quả tốt nhất 0.9573934837092731
```



b. Logistic regression

Input:

```
#Logistic
model_LG = LogisticRegression()
model_LG.fit(X_train,y_train)
y_pred = model_LG.predict(X_test)

print('Accuracy score:',accuracy_score(y_test, y_pred))
```

Output:

```
Accuracy score: 0.9147869674185464
```

c. **SVM (Support Vector Machine)**

Input:

```
model_SVC = SVC()
model_SVC.fit(X_train,y_train)
y_pred = model_SVC.predict(X_test)

print('Accuracy score:',accuracy_score(y_test, y_pred))
```

Output:

```
Accuracy score: 0.9598997493734336
```

d. Decision tree

Input:

```
model_dec = DecisionTreeClassifier()
dTree = model_dec.fit(X_train, y_train)
y_pred = dTree.predict(X_test)

print('Accuracy score:',accuracy_score(y_test, y_pred))
```

Output:

```
Accuracy score: 0.9072681704260651
```

3. Kết quả độ chính xác mô hình

- KNN(k=10) : 95,74%
- Logistic regression : 91,47%
- **SVM : 95,99%**
- Decision tree : 90,72%

4. Kết luận

Trong 4 mô hình máy học được huấn luyện thì mô hình SVM cho ra điểm cho tốt nhất với 95.9% độ chính xác

Xem kết quả SVM mang lại

Input:

```
df_SVC = pd.DataFrame({'label':y_test,'predict_label':y_pred})
df_SVC
```

Output:

	label	predict_label
889	2	2
1672	3	3
414	2	2
1599	1	1
849	2	2
...	...	...
909	0	0
261	1	1
576	0	0
963	1	1
240	3	3

399 rows × 2 columns

CHƯƠNG 4. KẾT LUẬN

4.1. Các Kết Quả Đạt Được

Qua nghiên cứu này, chúng tôi đã thực hiện các kỹ thuật phân lớp để phân loại điện thoại theo các mức giá thấp, trung bình, cao và rất cao từ đó đề xuất các thuật toán phù hợp nhất cho việc phân loại giá điện thoại. Dữ liệu thực nghiệm là thông tin mô tả về cấu tạo và tính năng của điện thoại (RAM, Bộ nhớ trong, chiều dài chiều cao, có 4G hay không,...). Để thực hiện phân lớp, chúng tôi đã tiến hành biểu diễn trực quan các dữ liệu để rút ra những thông tin cần thiết cho việc phân lớp, sử dụng các loại kiểm định để kiểm tra mỗi tương độc lập giữa 1 số biến trong dữ liệu, dùng phép phân tích thành phần chính (PCA) để giảm chiều dữ liệu để rút gọn dữ liệu, dễ dàng hơn cho việc phân lớp. Cuối cùng chúng tôi tiến hành phân lớp biến mục tiêu 'price_range' qua các thuật toán: KNeighborsRegressor, Logistic regression, SVM (Support Vector Machine), Decision tree. Kết quả cả 4 mô hình này đều cho độ chính xác khá cao (>90%) và đều phù hợp để sử dụng cho việc phân lớp, trong đó thuật toán SVM có độ chính xác cao nhất (95,99%).

TÀI LIỆU THAM KHẢO

1. https://builtin.com/data-science/step-step-explanation-principal-component-analysis
2. https://www.investopedia.com/terms/h/hypothesistesting.asp
3. https://www.tableau.com/learn/articles/data-visualization
4. https://www.ibm.com/topics/logistic-regression
5. https://www.analytixlabs.co.in/blog/univariate-analysis/#What_is_Univariate_Analysis
6. https://www.jigsawacademy.com/blogs/business-analytics/bivariate-analysis/
7. https://careerfoundry.com/en/blog/data-analytics/multivariate-analysis/
8. https://machinelearningcoban.com/2017/01/08/knn/#:~:text=M%E1%BB%99t%20c%C3%A1ch%20ng%E1%BA%AFn%20g%E1%BB%8Dn%2C%20KNN,g%E1%BA%A7n%20nh%E1%BA%A5t%20n%C3%A0y%20l%C3%A0%20nhi%E1%BB%85u.
9. https://www.geeksforgeeks.org/support-vector-machine-in-machine-learning/
10. https://www.geeksforgeeks.org/decision-tree/?ref=leftbar-rightbar