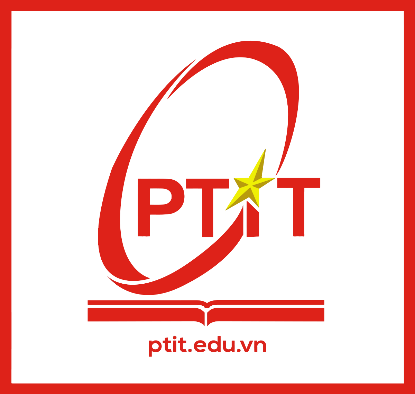
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**



**KHOA CÔNG NGHỆ THÔNG TIN 1**

****

BÁO CÁO BÀI TẬP LỚN:

**HỌC PHẦN KHAI PHÁ DỮ LIỆU**

**Đề tài:**

**Khai phá dữ liệu bằng phương pháp phân cụm và phân lớp để dự báo giá điện thoại**

**Nhóm bài tập lớn : 03**

**Lớp: : E22HTTT**

**Giảng viên giảng dạy : Phan Thị Hà**

**Thành viên:**

Nguyễn Đức Trí – B22DCAT302 (Nhóm trưởng)

Khuất Quang Đông – B22DCVT146

Nguyễn Đại Phát – B22DCVT393

Trần Văn Hoàng – B22DCAT129

Phạm Anh Minh – B22DCAT192

Phạm Việt Bách – B22DCVT043

Nguyễn Ngọc Long – B22DCVT319

Nguyễn Công Minh – B22DCVT343

Mục lục

[*LỜI CẢM ƠN* 3](#_Toc214686954)

[CHƯƠNG 1: TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU 4](#_Toc214686955)

[1.1. Phát hiện tri thức và khai phá dữ liệu. 4](#_Toc214686956)

[1.2. Quy trình khám phá tri thức trong CSDL. 4](#_Toc214686957)

[1.3. Mô tả bài toán dự báo giá điện thoại. 4](#_Toc214686958)

[1.3.1. Tổng quan bài toán. 4](#_Toc214686959)

[1.3.2. Phân tích dữ liệu thô 4](#_Toc214686960)

[CHƯƠNG 2: TIỀN XỬ LÝ DỮ LIỆU 6](#_Toc214686961)

[2.1. Làm sạch dữ liệu 6](#_Toc214686962)

[2.2. Tích hợp dữ liệu (Data Integration) 9](#_Toc214686963)

[2.3. Biến đổi dữ liệu (Data Transformation) 9](#_Toc214686964)

[CHƯƠNG 3: ỨNG DỤNG THUẬT TOÁN K-MEANS ĐỂ PHÂN CỤM SẢN PHẨM ĐIỆN THOẠI 12](#_Toc214686965)

[3.1. Giới thiệu về Thuật toán K-means 12](#_Toc214686966)

[3.2. Cách thức Hoạt động của K-means 12](#_Toc214686967)

[3.3. Phương pháp Elbow: Chọn K hợp lý 14](#_Toc214686968)

[3.4. Giải pháp lựa chọn centroid thông minh 16](#_Toc214686969)

[3.4.1. Khởi tạo thông minh (K-means++) 17](#_Toc214686970)

[3.4.2. Chạy nhiều lần với khởi tạo ngẫu nhiên (n\_init). 17](#_Toc214686971)

[3.5. Ứng dụng: Dự án "Phân loại Giá Điện thoại" 18](#_Toc214686972)

[CHƯƠNG 4: KHAI PHÁ DỮ LIỆU BẰNG THUẬT TOÁN PHÂN LỚP 20](#_Toc214686973)

[4.1. Giới thiệu về bài toán phân lớp 20](#_Toc214686974)

[4.2. Thuật toán phân lớp 20](#_Toc214686975)

[4.2.1. Thuật toán Cây quyết định (Decision Tree) – Regressor 20](#_Toc214686976)

[4.2.2. Thuật toán Rừng ngẫu nhiên (Random Forest) 22](#_Toc214686977)

[4.2.3. Thuật toán KNN (K-Nearest Neighbors) 24](#_Toc214686978)

[4.2.4. Ví dụ minh họa dựa trên dataset 25](#_Toc214686979)

[4.3. Các bước thực hiện 29](#_Toc214686980)

[4.4. Đánh giá mô hình 29](#_Toc214686981)

[CHƯƠNG 5: DEMO 31](#_Toc214686982)

[CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 32](#_Toc214686983)

[6.1. Kết luận 32](#_Toc214686984)

[6.2. Hướng phát triển 32](#_Toc214686985)

[TÀI LIỆU THAM KHẢO 34](#_Toc214686986)

# *LỜI CẢM ƠN*

Trong những năm gần đây cùng với phát triển nhanh chóng của khoa học kỹ thuật là sự bùng nổ về tri thức. Kho dữ liệu, nguồn tri thức của nhân loại cũng trở nên đồ sộ, vô tận làm cho vấn đề khai thác các nguồn tri thức đó ngày càng trở nên nóng bỏng và đặt ra thách thức lớn cho nền công nghệ thông tin thế giới.

Nhu cầu về tìm kiếm và xử lý thông tin, cùng với yêu cầu về khả năng kịp thời khai thác chúng để mạng lại những năng suất và chất lượng cho công tác quản lý, hoạt động kinh doanh... đã trở nên cấp thiết trong xã hội hiện đại. Để đáp ứng phần nào yêu cầu này, người ta đã xây dựng các công cụ tìm kiếm và xử lý thông tin nhằm giúp cho người dùng tìm kiếm được các thông tin cần thiết cho mình.

Với các phương pháp khai thác cơ sở dữ liệu truyền thống chưa đáp ứng được các yêu cầu đó. Để giải quyết vấn đề này, một hướng đi mới đó là nghiên cứu và áp dụng kỹ thuật khai phá dữ liệu và khám phá tri thức trong môi trường Web. Do đó, việc nghiên cứu các mô hình dữ liệu mới và áp dụng các phương pháp khai phá dữ liệu trong khai phá tài nguyên Web là một xu thế tất yếu vừa có ý nghĩa khoa học vừa mang ý nghĩa thực tiễn cao.

Vì vậy chúng em chọn đề tài: “Khai phá dữ liệu bằng phương pháp phân cụm và phân lớp để dự báo giá điện thoại”, để làm báo cáo kết thúc môn học của mình.

Báo cáo gồm 5 chương:

Chương 1: Tổng quan về khai phá dữ liệu

Chương 2: Tiền xử lý dữ liệu

Chương 3: Ứng dụng thuật toán K-means để phân cụm sản phẩm điện thoại

Chương 4: Khai phá dữ liệu bằng thuật toán phân lớp

Chương 5: Kết luận và hướng phát triển

***Nhóm 3***

# TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU

## Phát hiện tri thức và khai phá dữ liệu.

* Phát hiện tri thức (Knowledge Discovery) trong các cơ sở dữ liệu là một qui trình nhận biết các mẫu hoặc các mô hình trong dữ liệu với các tính năng: hợp thức, mới, khả ích, và có thể hiểu được.
* Khai phá dữ liệu (Data mining) được định nghĩa như sau: “Data mining là một quá trình tìm kiếm, phát hiện các tri thức mới, tiềm ẩn, hữu dụng trong CSDL lớn”.
* Khai phá dữ liệu có thể được sử dụng cho các lĩnh vực y tế, phân tích thị trường, xây dựng ... có thể được xem như là kết quả của sự tiến triển tự nhiên của công nghệ thông tin.

## Quy trình khám phá tri thức trong CSDL.

A diagram of a diagram

AI-generated content may be incorrect.

Hình 1.1 Quá trình khai phá dữ liệu từ cơ sở dữ liệu.

## Mô tả bài toán dự báo giá điện thoại.

### Tổng quan bài toán.

* Dataset gồm các mô tả về các thuộc tính tương ứng với dự đoán giá điện thoại.
* Áp dụng các thuật toán để xác định giá của điện thoại.

### Phân tích dữ liệu thô

* Nguồn dữ liệu thô: <https://www.kaggle.com/datasets/abdulmalik1518/mobiles-dataset-2025/>
* Hiểu dữ liệu: Dữ liệu gồm các thông số liên quan đến cấu hình phần cứng của điện thoại và hãng điện thoại. Phân loại giá dựa trên các giá trị của từng thuộc tính.
* Dữ liệu gồm: Dữ liệu bao gồm 930 bản ghi cùng 11 thuộc tính về các đặc trưng của điện thoại.

A screenshot of a phone number

AI-generated content may be incorrect.

Hình 1.2 Dữ liệu ban đầu.

* Hiểu các thuộc tính:

|  |  |  |
| --- | --- | --- |
| STT | Thuộc tính | Ý nghĩa |
| 1 | Company Name | Tên công ty (Có thể tác động đến giá do thương hiệu) |
| 2 | Model Name | Mẫu máy (Có thể kèm theo bộ nhớ trong(ROM)) |
| 3 | Mobile Weight | Trọng lượng |
| 4 | RAM | RAM |
| 5 | Front Camera | Thông tin về độ phân giải camera trước |
| 6 | Back Camera | Thông tin về độ phân giải camera sau |
| 7 | Processor | Tên vi xử lý |
| 8 | Battery Capacity | Dung lượng pin |
| 9 | Screen Size | Kích thước màn hình |
| 10 | Launched Price (USA) | Giá tại thị trường Mỹ tại thời điểm ra mắt |
| 11 | Launched Year | Năm ra mắt |

# TIỀN XỬ LÝ DỮ LIỆU

## Làm sạch dữ liệu

* Là quá trình nhận dạng dữ liệu có thể có để tiến hành xử lý các dữ liệu bị nhiễu (noisy data), không nhất quán (inconsistent data) và các dữ liệu bị thiếu (missing data).
* Xử lý các dữ liệu nhiễu (noisy data), không nhất quán (Inconsistent Data).

|  |  |
| --- | --- |
| Cột | Quy trình làm sạch |
| Launched Price (USA) | Loại bỏ ký tự tiền tệ ($, USD) và dấu phân cách hàng nghìn (,). Chuyển đổi sang kiểu số thực (float). |
| RAM, Front Camera, Back Camera | Loại bỏ đơn vị (GB, MP, GHz) và các ký tự không phải số. Trích xuất chỉ giá trị số đầu tiên để xử lý dữ liệu kép/đa (ví dụ: 48MP + 8MP -> 48). |
| Battery Capacity, Screen Size | Loại bỏ đơn vị đo lường (mAh, inches) và chuyển đổi sang kiểu số. |
| ROM (Bộ nhớ trong) | Trích xuất thông tin từ cột Model Name. Thực hiện chuyển đổi đơn vị GB sang TB (chia 1024) để đảm bảo tính nhất quán của đơn vị. |

**Ví dụ 1:** Launched Price (USD):

499 USD 🡺 499.0

1,500 USD 🡺 1500.0

1,000,000 USD 🡺 1000000.0

**Ví dụ 2:** Camera:

48MP + 8MP 🡺 48

12MP / 4K 🡺 12

60MP (ultrawide) + 8MP (telephoto) 🡺 60

**Ví dụ 3:** RAM: 8GB / 12GB 🡺 8

**Ví dụ 4:** Screen Size: 6.1 inches 🡺 6.1

A table of numbers with numbers

AI-generated content may be incorrect.

* Xử lý các dữ liệu bị thiếu (Missing Data).
* Sau quá trình làm sạch và trích xuất, các giá trị thiếu (NaN) còn lại được xử lý như sau:

+) Thuộc tính ROM: Các giá trị thiếu được điền bằng trung vị (Median) của cột, một phương pháp phù hợp cho dữ liệu liên tục.

**Ví dụ Median:**

**Ví dụ 1:**

Tập dữ liệu A = {2, 7, 5, 1, 9}, n = 5(lẻ).

Sắp xếp: {1, 2, 5, 7, 9}.

Vị trí: (5+1)/2 = 3.

Median (Trung vị) là 5.

**Ví dụ 2:**

Tập dữ liệu B = {10, 20, 30, 40, 50, 60}, n = 6 (chẵn).

Sắp xếp: {10, 20, 30, 40, 50, 60}.

Vị trí hai giá trị giữa: 6/2 = 3 (giá trị 30) và 6/2 + 1 = 4 (giá trị 40).

Median (Trung vị) là (30 + 40)/2 = 70/2 = 35.

A table of numbers and text

AI-generated content may be incorrect.

## Tích hợp dữ liệu (Data Integration)

* Là quá trình trộn dữ liệu từ các nguồn khác nhau vào một kho dữ liệu sẵn sàng cho quá trình khai phá dữ liệu.
* Trong dự án này, dữ liệu được lấy từ một nguồn duy nhất (tập tin CSV) nên không cần thực hiện quá trình tích hợp dữ liệu phức tạp.

## Biến đổi dữ liệu (Data Transformation)

* Giai đoạn Biến đổi dữ liệu nhằm mục đích chuyển đổi các thuộc tính đã được làm sạch thành định dạng số phù hợp với mô hình học máy, đồng thời thực hiện chuẩn hóa để tối ưu hóa hiệu suất mô hình.
* **Company Name (Tên hãng):**
* Các hãng được chia thành 6 nhóm bao gồm: apple, honor, oppo, samsung, vivo và các hãng khác (other).
* Áp dụng kỹ thuật Mã hóa One-Hot (One-Hot Encoding) để chuyển các hãng sản xuất chính (ví dụ: Apple, Samsung, Xiaomi) thành các biến nhị phân (0 hoặc 1). Điều này giúp mô hình nhận biết sự khác biệt giữa các hãng mà không áp đặt mối quan hệ thứ tự.

A screenshot of a data

AI-generated content may be incorrect.

* **Processor (Chip xử lý):**
* Tìm các điện thoại sử dụng cùng 1 loại chip ở Mỹ, sau đó tính giá trị trung bình của các điện thoại này, rồi chia cho 100.

Ví dụ:

|  |  |  |
| --- | --- | --- |
| **Name** | **Processor** | **Launched Price (USA)** |
| **Xiaomi 14 Ultra** | **Snap 8 Gen 3** | 1200 |
| **Samsung Galaxy S24 Ultra** | **Snap 8 Gen 3** | 1300 |
| **Xiaomi 14T Pro** | **Dimensity 9300** | 900 |
| **Vivo X100 Pro** | **Dimensity 9300** | 1000 |
| **Samsung Galaxy A54** | **Exynos 1380** | 400 |

Giá trị trung bình cho mỗi chip (Processor\_Avg\_Price) sẽ là:

* Snap 8 Gen 3: (1200 + 1300)/2 = 1250
* Dimensity 9300: (900 + 1000)/2 = 950
* Exynos 1380: 400

Chúng ta chia giá trị trung bình cho 100 để chuẩn hóa:

|  |  |
| --- | --- |
| **Processor** | **Launched Price (USA)** |
| **Snap 8 Gen 3** | 1250 / 100 = 12.5 |
| **Dimensity 9300** | 950 / 100 = 9.5 |
| **Exynos 1380** | 400 / 100 = 4.0 |

* **ROM (Bộ nhớ trong):**
* Giá trị GB được chuyển đổi thành TB (chia 1024) để đảm bảo tính đồng nhất đơn vị sau đó điền khuyết bằng Median.

Ví dụ: ROM:

512GB 🡺 512 🡺 512/1024 = 0.5

* **Battery Capacity:**
* Đổi từ mAh thành Ah, mục đích là làm gọn số tránh sai số dữ liệu sau này. Chia cho 1000

Ví dụ: 4,200mAh 🡺 4200 🡺 4.2

A table of data with numbers

AI-generated content may be incorrect.

* **Launched Price (USA):**
* Chuẩn hóa dấu phẩy (,) để xử lý cả định dạng hàng nghìn và thập phân. Sau đó, nó xác thực giá trị đó phải nằm trong phạm vi hợp lý ($99 đến $2000), nếu không sẽ trả về NaN (giá trị thiếu).

A screenshot of a computer

AI-generated content may be incorrect.

# ỨNG DỤNG THUẬT TOÁN K-MEANS ĐỂ PHÂN CỤM SẢN PHẨM ĐIỆN THOẠI

## Giới thiệu về Thuật toán K-means

* **K-means** là một trong những thuật toán **phân cụm (Clustering)** phổ biến và cơ bản nhất thuộc nhóm **Học không giám sát (Unsupervised Learning).**
* **Mục tiêu:** Để phân chia một tập hợp gồm N điểm dữ liệu vào K cụm (cluster) khác nhau.
* **Nguyên tắc:** Thuật toán sẽ gán mỗi điểm dữ liệu vào cụm có **tâm (centroid)** gần nó nhất. Mục tiêu cuối cùng là tối ưu hóa sao cho tổng bình phương khoảng cách từ các điểm đến tâm cụm của chúng là nhỏ nhất (gọi là **Inertia** hoặc **WCSS).**

## Cách thức Hoạt động của K-means

* Thuật toán K-means hoạt động theo một quy trình lặp đi lặp lại để tìm ra các tâm cụm tối ưu.

1. **Bước 1: Khởi tạo (Initialize).**
   * Bạn (người dùng) chọn một số K (số cụm bạn muốn).

* Thuật toán sẽ chọn ra K điểm dữ liệu ban đầu làm K tâm cụm (centroid). Cách chọn thông minh nhất hiện nay là **K-means++** (đảm bảo các tâm ban đầu được trải rộng, không co cụm một chỗ).

1. **Bước 2: Gán nhãn (Assign).**

* Với *mọi* điểm dữ liệu, thuật toán sẽ tính khoảng cách (thường là Euclide) từ điểm đó đến K tâm cụm.
* Mỗi điểm sẽ được "gán" vào cụm của tâm mà nó **gần nhất.**
* Ký hiệu quy ước:
* **:** Tập dữu liệu gồm m điểm.
* : Số lượng cụm.
* : Các tâm cụm (centroids).
* **Với công thức để tính khoảng cách Euclide:** Khoảng cách giữa 1 điểm x và tâm cụm u được tính bằng bình phương khoảng cách Euclide:
* **Áp dụng với bài toán giả sử ta chọn được K = 3, các tâm cụm được chọn đó là:**

u(1): (6.0,12.0,48.0,3.6,6.1,799.0,0.125,1,**0,**0,0,0,0,8.99)

u(2):(6.0,12.0,48.0,4.2,6.7,949.0,0.25,1,**0,**0,0,0,0,8.99)

u(3):(4.0,12.0,12.0,2.227,5.4,699.0,0.0625,1,**0,**0,0,0,0,9.597142857142856)

Ta có x(1) là điện thoại có các thuộc tính : (6.0,12.0,48.0,3.6,6.1,799.0,0.125,1,**0,**0,0,0,0,8.99)

* D(x(1), u(1)) = (799-799)^2 + …. + (8.99-8.99)^2 = 0(Do vô tình lấy x(1) làm tâm u(1))

Tương tự ta có:

D(x(1), u(2)) = 40,009

D(x(1), u(3)) = 11,302

* **Công thức gắn nhãn:**

**Công thức tổng quát**: Tìm chỉ số j (nhãn cụm) sao cho khoảng cách là nhỏ nhất

* Vậy ta có:
* Từ đó ta thấy sẽ gần với tâm 1 nhất =>

1. **Bước 3: Cập nhật (Update)**

* Sau khi tất cả các điểm đã được gán, vị trí của K tâm cụm sẽ được tính toán lại.
* Tâm cụm mới của mỗi cụm sẽ được *di chuyển* đến vị trí **trung bình cộng** (mean) của tất cả các điểm dữ liệu nằm trong cụm đó.
* Ta có công thức tổng quát

**Chú thích**

**:** Tập hợp các điểm dữ liệu đang được gán nhãn thuộc cụm j.

: Số lượng điểm dữ liệu trong cụm j.

: Tổng vector của tất cả các điểm trong cụm j.

* Tiếp tục với data trên nhưng ta thêm khoảng 20 dữ liệu vào để có thể rõ ràng hơn về cách tìm tâm cụm mới.
* Dựa theo cách tính khoảng cách từ các điểm tới tâm cụm ta có thể tính, biết được điểm nào sẽ thuộc cụm nào.
* Ví dụ về 20 điểm dữ liệu đầu tiên trong dataset ta có:
* 4 chiếc điện thoại thuộc cụm u(1).
* 13 chiếc điện thoại thuộc u(2).
* 3 chiếc điện thoại thuộc u(3).
* Áp dụng công thức tổng quát trên ta tính được:
* u(1) new có tâm là (824.0$, 6.0 GB, 3.4 Pin trung bình,...)
* Tương tự ta tính được:

u(2) new có tâm là (1076$, 6.77GB, 4.28 Pin trung bình)

u(3) new(899$, 6GB,...)

1. **Bước 4: Lặp lại**

* Thuật toán lặp lại **Bước 2** và **Bước 3.**
* (Gán nhãn lại): Vì tâm cụm đã di chuyển, một số điểm bây giờ có thể gần tâm cụm khác hơn -> chúng đổi cụm.
* Quá trình này cứ thế tiếp diễn cho đến khi các tâm cụm không còn di chuyển đáng kể nữa, nghĩa là các cụm đã "hội tụ" (ổn định).

## Phương pháp Elbow: Chọn K hợp lý

* Một trong những thách thức lớn nhất của K-means là bạn phải tự chọn K. Nếu chọn K quá nhỏ, các cụm sẽ quá chung chung (under-clustering); nếu chọn K quá lớn, các cụm sẽ bị phân mảnh vô nghĩa (over-clustering).
* **Phương pháp Elbow (Khuỷu tay)** giúp chúng ta tìm ra K tối ưu bằng cách trực quan hóa chỉ số **Inertia.**
* **Inertia (WCSS):** Là tổng bình phương khoảng cách từ các điểm đến tâm cụm của chúng. Inertia thấp nghĩa là các cụm "chặt chẽ" -> Tốt.
* **Cách hoạt động:**

1. Chúng ta chạy thuật toán K-means nhiều lần với các giá trị K khác nhau (ví dụ: K chạy từ 1 đến 10).
2. Với mỗi K, chúng ta ghi lại giá trị Inertia (Inertia luôn giảm khi K tăng).
3. Chúng ta vẽ đồ thị của Inertia (trục Y) theo K (trục X).

* **Cách đọc đồ thị:**

1. Ban đầu, khi tăng K, Inertia sẽ **giảm rất nhanh** (dốc đứng).
2. Đến một lúc nào đó, đồ thị sẽ bắt đầu "là" ra, gần như nằm ngang. Điều này có nghĩa là việc thêm một cụm mới không còn mang lại lợi ích giảm Inertia nhiều nữa.
3. Điểm mà đồ thị **bị gãy** (thay đổi độ dốc rõ rệt nhất), trông giống như một **khuỷu tay**, chính là giá trị K tốt nhất được gợi ý.

* Ngoài việc phải đọc K tốt bằng cách đọc biểu đồ ta còn có thể sử dụng công thức để có thể trực quan hơn.
* **Ý tưởng:**

1. Vẽ một đường thẳng nối điểm đầu tiên K(min) và điểm cuối cùng K(max) của đồ thị.
2. Tính khoảng cách từ từng điểm trên đường cong Inertia đến đường thẳng này.
3. Điểm K nào có **khoảng cách lớn nhất** chính là khuỷu tay (Elbow Point).

* **Công thức hình học:** Giả sử đường thẳng nối điểm (1,J(K)) và (10, J(10)) có dạng **ax + by + c = 0**. Khoảng cách từ điểm (k, J(k))đến đường thẳng là:
* Áp dụng vào bài toán thực tế phân cụm giá điện thoại, dựa vào dataset ta có:

A graph with a blue and red line

AI-generated content may be incorrect.

* Ta thấy tại K = 4 thì có khoảng cách tới đường cơ sở là lớn nhất, vậy theo tính toán ta có thể xác định được K tốt = 4.
* **Trong code (ví dụ với sklearn).**

1. Tạo một list rỗng: inertia\_list = []
2. Viết vòng lặp for k in range(1, 11):
3. Bên trong vòng lặp:

* Tạo model: model = KMeans(n\_clusters=k)
* Viết vòng lặp for k in range(1, 11)
* Lấy Inertia: inertia\_list.append(model.inertia\_)

1. Bên ngoài vòng lặp: Vẽ đồ thị plt.plot(range(1, 11), inertia\_list)

**\*\* Chú ý**: Ngoài phương pháp Elbow giúp tìm được K hợp lý, việc pick centroid cũng là 1 vấn đề, Áp dụng vào bài toán thực tế “Phân cụm giá điện thoại”. Nếu như pick bừa 5 chiếc điện thoại đầu tiên và vô tình 5 chiếc này đều thuộc loại phân khúc “thấp”(ROM,RAM,... đều kém) thì sẽ có ảnh hưởng lớn đến việc phân cụm. Vì:

* **Vòng 1 (Gán nhãn):** Cả 5 centroid ban đầu đều ở khu vực "thấp". Các điểm "thấp" sẽ tự chia nhau gán vào 5 centroid này. Tuy nhiên, *tất cả* các điểm "cận cao cấp" và "cao cấp" đều ở rất xa cả 5 centroid. Chúng sẽ cùng bị gán vào *một* centroid duy nhất (cái centroid "thấp" mà gần chúng nhất, dù vẫn là rất xa).
* **Vòng 2 (Cập nhật):**
* 4 centroid "thấp" sẽ di chuyển loanh quanh trong khu vực "thấp”.
* 1 centroid "thấp" còn lại (cái đã vô tình "bắt" được tất cả các điểm "cao cấp") sẽ đột ngột di chuyển đến vị trí trung bình của *toàn bộ* khu vực "cao cấp" và "cận cao cấp”.

🡺 **Kết quả:** Sẽ nhận được một kết quả phân cụm rất tệ: 4 cụm nhỏ chen chúc nhau trong phân khúc thấp, và 1 cụm khổng lồ bao gồm tất cả các phân khúc còn lại

## Giải pháp lựa chọn centroid thông minh

* Chúng ta sẽ sử dụng vòng lặp kết hợp với phương pháp lựa chọn centroid thông minh.

### Khởi tạo thông minh (K-means++)

* Đây là giải pháp phổ biến và hiệu quả hơn, và hiện là **mặc định** trong hầu hết các thư viện. Thay vì "pick bừa" 5 centroid cùng lúc, K-means++ chọn chúng một cách có chiến lược:
* Với công thức:
* K-means++ sẽ lựa chọn điểm làm centroid có xác suất tỉ lệ thuận với bình phương khoảng cách tới tâm để làm centroid. Cụ thể các bước như sau:

1. **Centroid đầu tiên:** Chọn 1 điểm dữ liệu *bất kỳ* làm centroid đầu tiên (C1).
2. **Centroid thứ hai:** Tính khoảng cách từ *mọi* điểm khác đến C1. Chọn điểm **xa nhất** (hoặc chọn ngẫu nhiên với xác suất tỉ lệ với bình phương khoảng cách) làm centroid thứ hai (C2).
3. **Centroid thứ ba:** Với *mọi* điểm, tìm khoảng cách *ngắn nhất* của nó đến *bất kỳ* centroid nào đã được chọn (tức là khoảng cách đến C1 hoặc C2, tùy cái nào gần hơn).
4. Chọn điểm có "khoảng cách ngắn nhất" này là **lớn nhất** làm centroid thứ ba (C3).
5. **Lặp lại:** Cứ tiếp tục chọn điểm tiếp theo sao cho nó ở *xa* tất cả các centroid đã được chọn trước đó, cho đến khi đủ K centroid.

* Tuy nhiên việc nếu chỉ sử dụng K-means++ 1 lần chúng ta rất dễ rơi vào việc lựa chọn điểm ngoại lai outlier làm 1 centroid điều này dẫn đến centroid này kéo các điểm ra quá xa so với cụm mà nó lẽ ra nên thuộc về. Chính vì thế chúng ta cần phải lặp lại liên tục quá trình này thông qua.

### Chạy nhiều lần với khởi tạo ngẫu nhiên (n\_init).

1. Thay vì chỉ chạy thuật toán 1 lần, chúng ta sẽ chạy nó (ví dụ) 10 lần.
2. Mỗi lần, chúng ta chạy 5 centroid ở 5 vị trí ngẫu nhiên có xác suất mà đã tính thông qua phương pháp K-means++.
3. Sau 10 lần chạy, chúng ta sẽ có 10 kết quả phân cụm khác nhau.
4. Chúng ta sẽ chọn ra kết quả **tốt nhất**—là kết quả có **Inertia** (tổng bình phương khoảng cách từ các điểm đến tâm cụm của chúng) **thấp nhất.**

* Trong thư viện Scikit-learn, tham số này tên là n\_init (mặc định thường là 10).

## Ứng dụng: Dự án "Phân loại Giá Điện thoại"

* **Làm rõ thuật ngữ:** Điều quan trọng cần lưu ý: K-means là **Phân cụm (Clustering)**, không phải **Phân loại (Classification).**
* **Phân loại (bạn dùng):** Là *Học có giám sát*. Bạn cần dữ liệu có nhãn từ trước (ví dụ: "giá rẻ", "cao cấp") để huấn luyện mô hình dự đoán nhãn cho điện thoại mới.
* **Phân cụm (chúng ta làm):** Là *Học không giám sát*. Chúng ta *không biết* có bao nhiêu phân khúc. Chúng ta yêu cầu K-means: "Hãy nhìn vào dữ liệu Pin, Giá, Hãng... và **tự khám phá** xem có bao nhiêu nhóm tự nhiên trong đó”.
* **Quy trình ứng dụng:**

1. **Thu thập dữ liệu:** 'RAM','Front Camera', 'Back Camera', 'Battery Capacity','Screen Size', 'ROM','Launched Price (USA)','Company\_Apple', 'Company\_Honor','Company\_Oppo','Company\_Other','Company\_Samsung','Company\_Vivo', 'Processor\_Avg\_Price\_Scaled'.
2. **Tìm K:** Chạy phương pháp Elbow trên dữ liệu đã xử lý để tìm K tối ưu (ví dụ: K=4).
3. **Chạy K-means:** Chạy mô hình K-means cuối cùng với K=4.



**A screenshot of a computer screen

AI-generated content may be incorrect.**

1. **Diễn giải kết quả:**

* Gán nhãn cụm (0, 1, 2) vào lại bảng dữ liệu gốc.
* Sử dụng df.groupby('Cluster').mean() để xem giá trị trung bình của Pin, Launched Price ... của từng cụm.
* **Ví dụ diễn giải:**

A blue square with white text

AI-generated content may be incorrect.

# KHAI PHÁ DỮ LIỆU BẰNG THUẬT TOÁN PHÂN LỚP

## Giới thiệu về bài toán phân lớp

* Hồi quy là quá trình dự đoán một giá trị số thực (liên tục) cho các đối tượng dữ liệu dựa trên các đặc trưng đầu vào.
* Mục tiêu của bài toán là xây dựng một mô hình học có khả năng dự đoán chính xác giá trị mục tiêu (target value) của các đối tượng mới. Trong bài toán này, đó là giá điện thoại.
* Quá trình phân lớp bao gồm hai bước chính:
* **Bước 1:** Xây dựng mô hình (Training): Chuẩn bị tập dữ liệu huấn luyện bao gồm các đặc trưng (cấu hình điện thoại) và giá trị thực tế (giá tiền). Sử dụng thuật toán (như Decision Tree Regressor) để học mối quan hệ giữa đặc trưng và giá.
* **Bước 2:** Vận hành mô hình (Prediction): Sử dụng mô hình để dự đoán giá cho các mẫu điện thoại mới và đánh giá sai số bằng các độ đo như MAE, MSE hoặc RMSE.

## Thuật toán phân lớp

### Thuật toán Cây quyết định (Decision Tree) – Regressor

* + - 1. **Logic dự đoán**
* Khi xây dựng cây hồi quy, chúng ta chia không gian đặc trưng thành các vùng hình chữ nhật rời rạc R1, R2, R3, …Rm. Mỗi vùng này tương ứng với một **nút lá (leaf node)** của cây.
* Công thức dự đoán:
* Giá trị dự đoán cho một điểm dữ liệu mới rơi vào vùng (nút lá) R chính là trung bình cộng các giá trị mục tiêu của tất cả các điểm dữ liệu huấn luyện nằm trong vùng đó.

Trong đó:

* **:** Giá trị dự đoán cho vùng R
* **:** Số lượng điểm dữ liệu huấn luyện nằm trong vùng R
* **:** Giá trị thực tế của điểm dữ liệu thứ i trong vùng R
  + - 1. **Tiêu chí tách nhánh**
* Để xây dựng cây, tại mỗi bước, thuật toán phải quyết định biến (feature) nào và ngưỡng (threshold) nào để chia một nút cha thành hai nút con. Mục tiêu là làm cho các giá trị y trong mỗi nút con càng giống nhau (ít biến động) càng tốt.Chúng ta sử dụng Tổng Bình phương Sai số (Sum of Squared Errors - SSE) hoặc Sai số Bình phương Trung bình (Mean Squared Error - MSE) làm thước đo "độ vẩn đục" (impurity).
* Độ đo sai số tại một nút (Node Impurify)
* Giả sử tại một nút t, chúng ta có Nt điểm dữ liệu. Giá trị trung bình tại nút đó là:
* Độ đo lỗi (thường là MSE) tại nút t được tính bằng:

(Đây chính là phương sai - Variance của các giá trị y tại nút đó).

* Hàm mất mát cho phép tách (Cost Function for Splitting)
* Khi xem xét một phép tách chia nút cha thành hai nút con: Trái (Left) và Phải (Right), thuật toán sẽ tìm cách tối thiểu hóa tổng MSE có trọng số của hai nút con.Công thức hàm mất mát J cho một phép tách (feature, threshold) là:

Trong đó:

* **:**Tổng số điểm ở nút cha
* **, :**Số điểm rơi vào nút con trái và phải
* **:** Số điểm rơi vào nút con trái và phải

**Mục tiêu:** Thuật toán sẽ thử tất cả các đặc trưng và tất cả các ngưỡng, sau đó chọn cặp (đặc trưng, ngưỡng) làm cho J(split) **nhỏ nhất**

### Thuật toán Rừng ngẫu nhiên (Random Forest)

Mô hình này là một phương pháp Ensemble (Hợp tấu), cụ thể là kỹ thuật Bagging (Bootstrap Aggregating). Thay vì phụ thuộc vào một cây quyết định duy nhất (dễ bị quá khớp), nó xây dựng một "rừng" gồm nhiều cây và lấy kết quả trung bình

* + - 1. **Quy trình tổng quát**
* Giả sử chúng ta có bộ dữ liệu huấn luyện D = {(x1, y1), (x2, y2), ..., (xN, yN)} với N điểm dữ liệu.
* Quy trình tạo ra dự đoán gồm 3 bước chính:
* Bootstrapping: Tạo ra B tập dữ liệu con khác nhau.
* Xây dựng cây: Huấn luyện B cây quyết định riêng biệt trên các tập con đó.
* Aggregation (Tổng hợp): Lấy trung bình kết quả của B cây.

A diagram of a network

AI-generated content may be incorrect.

* + - 1. Các công thức chi tiết

**Bước 1: Bootstrapping (Lấy mẫu có hoàn lại)**

* Chúng ta cần tạo ra B cây (trong thư viện lập trình thường gọi là .Với mỗi cây thứ b (từ 1 đến B), ta tạo ra một tập dữ liệu huấn luyện bằng cách lấy mẫu ngẫu nhiên N lần từ tập gốc D có hoàn lại (tức là một điểm dữ liệu có thể xuất hiện nhiều lần trong ).

**Bước 2: Dự đoán của từng Cây đơn lẻ (Individual Tree Prediction)**

* Mỗi tập được dùng để huấn luyện một cây quyết định .
* **Lưu ý quan trọng**: Tại mỗi nút phân tách của cây, thuật toán chỉ xem xét một tập hợp ngẫu nhiên các đặc trưng (random features) chứ không phải toàn bộ đặc trưng
* Giả sử cây đã được xây dựng xong. Khi đưa một điểm dữ liệu mới x vào, nó sẽ rơi vào một vùng (nút lá) .
* Công thức dự đoán của cây thứ b, ký hiệu là .

Trong đó:

* **:**Giá trị dự đoán của cây thứ b
* **:** Tập hợp các điểm dữ liệu từ tập huấn luyện rơi vào cùng nút lá với x.
* **:** Số lượng điểm dữ liệu trong nút lá đó.
* **:** Giá trị thực tế của các điểm dữ liệu trong nút lá đó.

**Bước 3: Tổng hợp kết quả (Aggregation) - CÔNG THỨC CHÍNH**

* Đây là công thức quan trọng nhất phân biệt Random Forest với Decision Tree. Dự đoán cuối cùng của Rừng ngẫu nhiên, ký hiệu là , là trung bình cộng các dự đoán của tất cả các cây con
* Công thức:

Trong đó:

* **B**: Tổng số cây trong rừng.
* **:** Kết quả dự đoán của cây thứ b.
  + - 1. Tại sao công thức lại hiệu quả.
* Về mặt toán học, nếu các cây con có sai số (variance) cao nhưng không tương quan với nhau (do bootstrapping và random features), thì việc lấy trung bình theo công thức trên sẽ làm giảm Phương sai (Variance) của mô hình tổng thể đi B lần.
* Điều này giúp Random Forest dự đoán "mượt mà" hơn và tránh bị quá khớp (overfitting) so với một cây quyết định đơn lẻ.

### Thuật toán KNN (K-Nearest Neighbors)

* + - 1. Quy trình tổng quát
* Giả sử chúng ta có tập dữ liệu huấn luyện D = {(, ), (,), ..., (,)}.Khi có một điểm dữ liệu mới cần dự đoán giá trị , quy trình gồm 3 bước:
* Tính khoảng cách: Đo khoảng cách từ đến tất cả các điểm trong tập huấn luyện.
* Tìm lân cận: Chọn ra K điểm có khoảng cách nhỏ nhất. Gọi tập hợp này là .
* Dự đoán: Tính giá trị trung bình (có thể có trọng số) của các trong tập **.**
  + - 1. Các công thức chi tiết

**Bước 1: Tính Khoảng cách (Distance Metric)**

* Để xác định "lân cận", ta cần một công thức đo khoảng cách. Phổ biến nhất là Khoảng cách Euclid.
* Với hai vector x = (, ...,) và z = (, ...,) trong không gian d chiều, khoảng cách Euclid d(x,z) được tính bằng:

*(Trong bài toán điện thoại, d là số lượng đặc trưng như RAM, ROM, Pin...)*

**Bước 2: Xác định Tập Lân cận ()**

* Sắp xếp các khoảng cách theo thứ tự tăng dần. Chọn K điểm đầu tiên để đưa vào tập .

**Bước 3: Tính Giá trị Dự đoán (Prediction Formulas)**

* Có hai cách chính để tính giá trị dự đoán :

**Cách 1: Trung bình Đơn giản (Uniform Weights)**

* Đây là cách mặc định. Tất cả K hàng xóm đều có tiếng nói như nhau, bất kể khoảng cách.

Trong đó:

* **:** Giá trị dự đoán.
* **:** Giá trị thực tế của hàng xóm thứ i.
* **K :** Số lượng lân cận.

**Cách 2: Trung bình Có trọng số (Distance-Weighted)**

* Cách này cho rằng những hàng xóm gần hơn thì đáng tin cậy hơn và nên có ảnh hưởng lớn hơn tới kết quả.
* Công thức tổng quát:

*Trong đó là trọng số của điểm thứ i trong tập lân cận*

* Có hai cách phổ biến để tính trọng số :

1. Nghịch đảo khoảng cách (Inverse Distance):

( là một số rất nhỏ để tránh lỗi chia cho 0 nếu khoảng cách bằng 0).

1. Hàm Gaussian (Gaussian Kernel):

(Cách này làm trọng số giảm rất nhanh khi khoảng cách tăng lên, là tham số điều chỉnh độ rộng).

### Ví dụ minh họa dựa trên dataset

* Trong chúng ta sử dụng 3 cột đầu trong dataset để làm ví dụ mẫu như sau:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Company Name | Model Name | RAM | Front Camera | Back Camera | Processor | Battery Capacity | Screen Size | Launched Price (USA) | Launched Year |
| Apple | iPhone 16 128GB | 6GB | 12MP | 48MP | A17 Bionic | 3,600mAh | 6.1 inches | USD 799 | 2024 |
| Apple | iPhone 16 256GB | 6GB | 12MP | 48MP | A17 Bionic | 3,600mAh | 6.1 inches | USD 849 | 2024 |
| Apple | iPhone 16 512GB | 6GB | 12MP | 48MP | A17 Bionic | 3,600mAh | 6.1 inches | USD 899 | 2024 |
| Apple | iPhone 16 Plus 128GB | 6GB | 12MP | 48MP | A17 Bionic | 4,200mAh | 6.7 inches | USD 899 | 2024 |

* Vì bảng quá dài nên chúng ta sẽ lọc ra 4 giá trị Battery Capacity(x1), Screen Size(x2), ROM(x3), Launched Price (USA)(y) để thu gọn tính toán. Chúng ta sẽ áp dụng các bước chuẩn hóa dữ liệu như ở preprocessor:

x1 = x1old / 1000

x2 = x2old

x3 = x3old / 64

* Từ đó chúng ta có bảng data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Battery (x1​)** | **Screen (x2​)** | **ROM (x3​)** | **Price (y)** |
| **Row 1** | 3.6 | 6.1 | 2 | **7.99** |
| **Row 2** | 3.6 | 6.1 | 4 | **8.49** |
| **Row 3** | 3.6 | 6.1 | 8 | **8.99** |
| **Row 4** | 4.2 | 6.7 | 2 | **8.99** |

* Input sẽ là các giá trị ở các cột: Battery, Screen, ROM
* Output sẽ là giá trị của cột Price.
* Kịch bản dự đoán:
* Giả sử chúng ta có một chiếc điện thoại mới (Query Point - ) cần định giá với thông số:
* Battery: 3.6.
* Screen: 6.1.
* ROM: 6 (Nằm giữa 4 và 8).
* Vector = [3.6, 6.1, 6].
* Chúng ta sẽ dùng 3 thuật toán để dự đoán cho ví dụ này:
  + - 1. **Thuật toán KNN**

Chọn K = 2 (Dựa vào 2 hàng xóm gần nhất). Dùng khoảng cách Euclid.

**Bước 1: Tính khoảng cách từ đến các điểm dữ liệu**

* Công thức:

.

* Với Row 1:

= = 4

* Với Row 2:

= = 2

* Với Row 3:

= = 2

* Với Row 4:

=

**Bước 2: Chọn K lân cận**

* Hai điểm có khoảng cách nhỏ nhất là Row 2 (d=2) và Row 3 (d=2).

**Bước 3: Dự đoán (Lấy trung bình giá trị y)**

* Giá của Row 2 là 8.49.
* Giá của Row 3 là 8.99.
  + - 1. **Thuật toán Decision Tree**
* Cây quyết định sẽ cố gắng tìm một đặc trưng và một ngưỡng để chia dữ liệu sao cho các nhóm con có giá trị y giống nhau nhất (giảm phương sai/MSE).
* Nhìn vào dữ liệu:
* Nếu chia theo Battery hoặc Screen: Row 4 sẽ bị tách riêng ra.
* Nếu chia theo ROM: Có thể tách Row 1, 2, 4 (ROM nhỏ) và Row 3 (ROM lớn).
* Giả sử thuật toán chọn ROM làm tiêu chí tách đầu tiên (Nút gốc).
* Ngưỡng tách tối ưu (ví dụ): ROM 3
* Cấu trúc cây:
* Nút Gốc: ROM <= 3?
* Nhánh TRÁI (Đúng): Gồm [Row 1, Row 4] (ROM là 2).
* Giá trị dự đoán tại lá này = Trung bình(Row 1, Row 4) = (7.99 + 8.99) / 2 = 8.49
* Nhánh PHẢI (Sai): Gồm [Row 2, Row 3] (ROM là 4 và 8).
* Giá trị dự đoán tại lá này = Trung bình(Row 2, Row 3) = (8.49 + 8.99) / 2 = 8.74
* Dự đoán cho :
* Chiếc điện thoại mới có ROM = 6.
* Kiểm tra: 6 <= 3 -> Sai.
* Đi sang Nhánh PHẢI.
  + - 1. **Thuật toán Random Forest**
* Giả sử ta xây dựng một rừng gồm 2 cây.
* Kỹ thuật cốt lõi: Bootstrapping (Lấy mẫu ngẫu nhiên có hoàn lại).

**Cây 1:**

* *Dữ liệu huấn luyện (random):* [Row 2, Row 3, Row 3, Row 4] (Row 1 bị bỏ qua ngẫu nhiên).
* Cây 1 học được rằng: ROM càng cao giá càng cao.
* Giả sử Cây 1 tách dữ liệu sao cho (ROM=6) rơi vào cùng nhóm với Row 2 và Row 3.
* Dự đoán của Cây 1: .

**Cây 2:**

* *Dữ liệu huấn luyện (random):* [Row 1, Row 2, Row 4, Row 1].
* Cây 2 thấy ROM=2 có giá 7.99, ROM=4 có giá 8.49.
* Với (ROM=6), Cây 2 nhận thấy nó lớn hơn các mẫu có sẵn (lớn hơn ROM 4). Nó sẽ gán giá trị của điểm dữ liệu có ROM lớn nhất mà nó biết (là Row 2).
* Dự đoán của Cây 2: 8.49.

Tổng hợp kết quả (Ensemble):

* Rừng ngẫu nhiên sẽ lấy trung bình cộng dự đoán của 2 cây.

## Các bước thực hiện

**Bước 1: Mở file dữ liệu .csv đã tiền xử lý**

* Dữ liệu sau khi đã được làm sạch và mã hóa (ví dụ xử lý giá trị thiếu, chuyển đổi nhãn văn bản sang số, chuẩn hóa thuộc tính) được lưu trong file .csv.

A screenshot of a table

AI-generated content may be incorrect.

* File được nạp vào chương trình bằng thư viện pandas trong Python:
* # Load data
* FILE\_PATH = "/media/hoang/HDD\_Code/Tài liệu học tập/AI/Code\_example/Mobile\_Price\_Prediction/mobiles\_dataset\_2025\_processed.csv"
* df = pd.read\_csv(FILE\_PATH)
* print(f"Loaded {len(df)} rows and {len(df.columns)} columns")
* df.head()

**Bước 2: Chia tập dữ liệu**

* Dữ liệu được chia thành hai phần:
* Tập huấn luyện (train): 80% dữ liệu.
* Tập kiểm tra (test): 20% dữ liệu.

**Bước 3: Huấn luyện mô hình**

# Train models from scratch with new processed data

import os

print("TRAINING MODELS WITH TARGET-ENCODED DATA")

# Initialize fresh models for REGRESSION

best\_knn = KNeighborsRegressor(

*n\_neighbors*=7,

*weights*='distance',

*metric*='euclidean'

)

best\_dt = DecisionTreeRegressor(

*max\_depth*=15,

*min\_samples\_split*=10,

*min\_samples\_leaf*=4,

*random\_state*=RANDOM\_STATE

)

rf = RandomForestRegressor(

*n\_estimators*=200,

*max\_depth*=20,

*min\_samples\_split*=5,

*min\_samples\_leaf*=2,

*max\_features*='sqrt',

*bootstrap*=True,

*random\_state*=RANDOM\_STATE,

*n\_jobs*=-1

)

print("FITTING MODELS")

# KNN needs scaled data

print("Fitting KNN with scaled data...")

best\_knn.fit(X\_train\_prep, y\_train)

print("KNN fitted")

# Decision Tree needs original data with feature names

print("Fitting Decision Tree with original data...")

best\_dt.fit(X\_train, y\_train)

print("Decision Tree fitted")

# Random Forest needs original data with feature names

print("Fitting Random Forest with original data...")

rf.fit(X\_train, y\_train)

print("Random Forest fitted")

# Build pipelines that include the fitted numeric\_pipeline (so preprocessing is saved together)

from sklearn.pipeline import Pipeline as SKPipeline

knn\_pipe = SKPipeline([('preprocessor', numeric\_pipeline), ('model', best\_knn)])

dt\_pipe = SKPipeline([('preprocessor', numeric\_pipeline), ('model', best\_dt)])

rf\_pipe = SKPipeline([('preprocessor', numeric\_pipeline), ('model', rf)])

**Bước 4: Dự đoán và đánh giá hiệu quả mô hình**

**Bước 4.1: Đánh giá hiệu quả mô hình**

A screenshot of a graph

AI-generated content may be incorrect.

Dựa vào hiệu quả của từng mô hình trên tập test chúng ta thấy được Random Forest là thuật toán tối ưu nhất => Chọn thuật toán Random Forest làm thuật toán dự đoán

**Bước 4.2: Dự đoán bằng mô hình**

* Dựa vào thuật toán Random Forest chúng ta tiến hành dự đoán giá điện thoại.

def predict\_price(*phone\_data*, *model\_path*='random\_forest\_model.pkl'):

print("PHONE SPECIFICATIONS:")

for col, val in *phone\_data*.items():

print(f" {col}: {val[0]}")

* # Initialize preprocessor
* preprocessor = MobilePreprocessor()
* preprocessor.load\_preprocessor()
* processed\_data = preprocessor.preprocess\_input(*phone\_data*)
* print("PROCESSED FEATURES:")
* print(processed\_data.to\_string())
* # Load and predict with model
* try:
* with open(*model\_path*, 'rb') as f:
* model\_pipeline = pickle.load(f)
* prediction = model\_pipeline.predict(processed\_data)
* predicted\_price = float(prediction[0]) \* 100
* return predicted\_price
* except Exception as e:
* print(f"\nPrediction failed: {e}")
* import traceback
* traceback.print\_exc()
* return None
* if \_\_name\_\_ == "\_\_main\_\_":
* print("MOBILE PRICE PREDICTION")
* # Test case: Samsung Galaxy Z Fold 6
* phone\_data = pd.DataFrame({
* 'Model Name': ['Galaxy Z Fold 6 256GB'],
* 'RAM': ['12GB'],
* 'Front Camera': ['10MP'],
* 'Back Camera': ['50MP'],
* 'Battery Capacity': ['4400mAh'],
* 'Screen Size': ['7.6 inches'],
* 'Company Name': ['Samsung'],
* 'Processor': ['Snapdragon 8 Gen 3']
* })
* # Predict using Random Forest
* predicted\_price = predict\_price(phone\_data, *model\_path*='random\_forest\_model.pkl')
* if predicted\_price:
* print(f"PREDICTED PRICE: ${predicted\_price:.2f}")
* phone\_data\_2 = pd.DataFrame({
* 'Model Name': ['iPhone 15 Pro Max 512GB'],
* 'RAM': ['8GB'],
* 'Front Camera': ['12MP'],
* 'Back Camera': ['48MP'],
* 'Battery Capacity': ['4685mAh'],
* 'Screen Size': ['6.9 inches'],
* 'Company Name': ['Apple'],
* 'Processor': ['Apple A18 Pro']
* })
* predicted\_price\_2 = predict\_price(phone\_data\_2, *model\_path*='random\_forest\_model.pkl')
* if predicted\_price\_2:
* print(f"PREDICTED PRICE: ${predicted\_price\_2:.2f}")

Code test để dự đoán thử với mô hình

MOBILE PRICE PREDICTION

PHONE SPECIFICATIONS:

Model Name: Galaxy Z Fold 6 256GB

RAM: 12GB

Front Camera: 10MP

Back Camera: 50MP

Battery Capacity: 4400mAh

Screen Size: 7.6 inches

Company Name: Samsung

Processor: Snapdragon 8 Gen 3

Loaded Target Encoder (target\_encoder\_fitted.pkl)

PREDICTED PRICE: $416.44

PHONE SPECIFICATIONS:

Model Name: iPhone 15 Pro Max 512GB

RAM: 8GB

Front Camera: 12MP

Back Camera: 48MP

Battery Capacity: 4685mAh

Screen Size: 6.9 inches

Company Name: Apple

Processor: Apple A18 Pro

Loaded Target Encoder (target\_encoder\_fitted.pkl)

PREDICTED PRICE: $517.15

## Đánh giá mô hình

* Hiệu quả mô hình được đánh giá bằng ba chỉ số chính:
* Trong các công thức:
* n: Tổng số điểm dữ liệu.
* : Giá trị thực tế (Actual value) của điểm thứ i.
* : Giá trị dự đoán (Predicted value) của điểm thứ i.
* : Giá trị trung bình của toàn bộ dữ liệu thực tế ().

|  |  |  |
| --- | --- | --- |
| Chỉ số | Ý nghĩa | Công thức |
| MAE (Mean Absolute Error) | Sai số tuyệt đối trung bình |  |
| Root Mean Square Error | Căn bậc hai sai số bình phương trung bình |  |
| R² (R-squared) | Hệ số xác định, phản ánh mức độ phù hợp của mô hình |  |

* MAE và RMSE càng nhỏ → mô hình càng chính xác.
* R² càng gần 1 → mô hình càng tốt.
* **Kết quả thu được:** Từ kết quả trên, có thể thấy Rừng ngẫu nhiên (Random Forest) cho độ chính xác cao nhất, sai số thấp nhất. Do đó, mô hình này được lựa chọn cho bước dự đoán cuối cùng.

# DEMO

**Ảnh giao diện chính của trang web dự báo giá điện thoại:**

A screenshot of a computer

AI-generated content may be incorrect.

Phần dự đoán giá điện thoại:

A screenshot of a computer

AI-generated content may be incorrect.

Giao diện khi vào trang dự đoán

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả của dự báo giá điện thoại

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả khi thử thay đổi phần Chip/vi xử lý khi thay thế bằng một con chip giá rẻ

Các ví dụ khác về phần dự đoán giá điện thoại:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

* Phân cụm và phân lớp là 2 lĩnh vực khá quan trọng trong khai phá dữ liệu xu hướng trong tương lai, nó được ứng dụng trong nhiều ngành như y tế, thương mại... Hoàn thành đề tài “Khai phá dữ liệu bằng phương pháp phân cụm và phân lớp để dự báo giá điện thoại”, nhóm em đã đạt được một số kết quả như sau:
* Tìm hiểu tổng quan về khai phá dữ liệu, bài toán phân lớp, phân cụm để từ đó xây dựng mô hình phân cụm và mô hình phân lớp hỗ trợ dự đoán giá điện thoại.
* Thu thập dữ liệu điện thoại tiền xử lý dữ liệu bằng Python.
* So sánh kết quả tỷ lệ train/test để lựa chọn tỷ lệ đánh giá mô hình tốt nhất.
* Đánh giá mô hình phân lớp so sánh giữa 3 thuật toán Cây quyết định,KNN và random forrest.
* Tuy nhiên bài tập nhóm vẫn còn một số hạn chế:
* Việc đánh giá chất lượng camera chỉ dựa trên độ phân giải khá phiến diện, cụ thể như sau:
* Một số thiết bị sử dụng kĩ thuật nội suy để tăng độ phân giải
* Kích thước cảm biến quan trọng hơn( Tuy nhiên nhiều mẫu máy thường không tiết lộ điều này)
* Ngoài ra cần những thông tin sau để đánh giá camera:

+) Tốc độ màn chụp

+) Khẩu độ

+) ISO

+) Chất lượng thấu kính

+) Khả năng tối ưu phần mềm + AI

* Nếu có dữ liệu thời gian sử dụng, sẽ tốt hơn dung lượng pin vì khả năng tiêu thụ và tối ưu mỗi máy khác nhau.
* Kết quả dự đoán tương đối cao nhưng vẫn chưa được tốt nhất.

## Hướng phát triển

* Áp dụng mô hình học máy vào một số lĩnh vực khác và đi vào áp dụng thực tế.
* Xây dựng, cải tiến mô hình dự đoán giá điện thoại với phương pháp học máy khác như Naives Bayes…
* Trong quá trình hoàn thành bài tập lớn, nhóm em đã cố gắng tìm hiểu và tham khảo các tài liệu liên quan. Tuy nhiên, thời gian có hạn nên chúng em sẽ không tránh khỏi những thiếu sót, rất mong nhận được sự đóng góp ý kiến của quý thầy cô và các bạn để báo cáo và kỹ năng của chúng em ngày được hoàn thiện hơn và có thế áp dụng được trong thực tiễn.

Link github code + model + dataset + báo cáo + web: https://github.com/nguyenductripro/Data-mining-group-3

# TÀI LIỆU THAM KHẢO

[1] <https://en.wikipedia.org/wiki/Random_forest>

[2] <https://cuongndh.blogspot.com/p/khai-pha-du-lieu.html>

[3] <https://en.wikipedia.org/wiki/K-means_clustering>

[4]<https://vi.wikipedia.org/wiki/C%C3%A2y_quy%E1%BA%BFt_%C4%91%E1%BB%8Bnh>

[5] <https://machinelearningcoban.com/>

[6] TS.Đặng Thị Thu Hiền, (2019), Bài giảng Khai Phá dữ liệu.

[7] <https://www.gsmarena.com/>