**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: HỌC MÁY**

**ĐỀ TÀI: BREAST CANCER PRIDICTION**

**Giáo viên hướng dẫn: Giảng viên Ngô Thị Bích Thúy**

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Stt** | **Họ và tên** | **Mã sv** | **Lớp** |
| 1 | Vũ Đức Anh | 1671020036 | CNTT 16-01 |
| 2 | Võ Vĩnh Thái | 1671020289 | CNTT 16-01 |
| 3 | Nguyễn Quang Hiệp | 1671020115 | CNTT 16-01 |

**Hà Nội, năm 2024**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: HỌC MÁY**

**ĐỀ TÀI: BREAST CANCER PRIDICTION**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Họ và Tên | Mã Sinh Viên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 1 | Vũ Đức Anh | 1671020036 | 09/02/04 |  |  |
| 2 | Võ Vĩnh Thái | 1671020289 | 20/10/04 |  |  |
| 3 | Nguyễn Quang Hiệp | 1671020115 | 23/04/04 |  |  |

**CÁN BỘ CHẤM THI**

**Hà Nội, năm 2024**

**LỜI MỞ ĐẦU**

Ung thư vú hiện đang là một trong những căn bệnh nguy hiểm hàng đầu đối với phụ nữ trên toàn thế giới. Theo Tổ chức Y tế Thế giới (WHO), ung thư vú chiếm tỷ lệ cao nhất trong các loại ung thư ở phụ nữ, với hàng triệu ca mắc mới mỗi năm. Việc phát hiện sớm và điều trị kịp thời có thể cứu sống hàng triệu người, nhưng không phải lúc nào quá trình chẩn đoán cũng dễ dàng. Các phương pháp truyền thống thường gặp khó khăn trong việc phát hiện và phân loại chính xác các tổn thương, dẫn đến sự chậm trễ trong việc điều trị và giảm hiệu quả điều trị.

Trong bối cảnh này, sự phát triển của công nghệ học máy (machine learning) đã mở ra những cơ hội mới trong việc cải thiện quá trình chẩn đoán ung thư vú. Các thuật toán học máy có khả năng phân tích và xử lý lượng dữ liệu lớn từ hình ảnh y tế, giúp tăng độ chính xác trong việc phát hiện bệnh, đồng thời hỗ trợ bác sĩ trong việc ra quyết định lâm sàng.

Đặc biệt, hai thuật toán như Cây quyết định (Decision Tree) và k láng giềng gần nhất (k-Nearest Neighbors) đã chứng tỏ hiệu quả vượt trội trong việc phân loại và dự đoán tình trạng bệnh nhân. Cây quyết định giúp xác định các yếu tố quan trọng liên quan đến nguy cơ mắc bệnh, trong khi k láng giềng gần nhất sử dụng thông tin từ các bệnh nhân tương tự để đưa ra chẩn đoán chính xác hơn.

Bài báo cáo này sẽ tập trung vào việc trình bày các ứng dụng của học máy trong chẩn đoán ung thư vú, những kết quả đạt được từ việc áp dụng các thuật toán này, cũng như triển vọng trong tương lai của công nghệ này trong việc nâng cao hiệu quả chẩn đoán và điều trị căn bệnh này.

**MỤC LỤC**

[**CHƯƠNG 1. LÝ THUYẾT HỌC MÁY** 2](#_Toc179579886)

[1.1. Giới thiệu về học máy 2](#_Toc179579887)

[1.2. Ứng dụng của Học máy 3](#_Toc179579888)

[1.3. Các Dạng Học Máy 5](#_Toc179579889)

[1.4. Thuật toán cây quyết định 6](#_Toc179579890)

[1.4.1. Mô Tả Chi Tiết về Cây Quyết Định 6](#_Toc179579891)

[1.4.2. Ưu và Nhược Điểm của Cây Quyết Định 7](#_Toc179579892)

[1.5. Thuật toán KNN 7](#_Toc179579893)

[1.5.1. Thuật toán KNN là gì? 7](#_Toc179579894)

[1.5.2. Ưu điểm và nhược điểm của KNN 7](#_Toc179579895)

[**CHƯƠNG 2. THỰC HIỆN BÀI TOÁN** 9](#_Toc179579896)

[2.1. Importing Libraries 9](#_Toc179579897)

[2.2. Data Preprocessing 11](#_Toc179579898)

[2.3. Missing Value 13](#_Toc179579899)

[2.4. EDA 16](#_Toc179579900)

[2.5. CORR 17](#_Toc179579901)

[2.6. Buliding Model 20](#_Toc179579902)

[2.7. Apply Machine learning Algo 21](#_Toc179579903)

[2.7.1. Thuật toán KNN 21](#_Toc179579904)

[2.7.2. Thuật toán DT 23](#_Toc179579905)

[2.8. Model Comparison 26](#_Toc179579906)

[**CHƯƠNG 3. KẾT LUẬN** 30](#_Toc179579907)

[**TÀI LIỆU THAM KHẢO** 32](#_Toc179579908)

# **LÝ THUYẾT HỌC MÁY**

## Giới thiệu về học máy

* **Học máy (machine learning)** là khả năng của chương trình máy tính trong việc sử dụng kinh nghiệm, quan sát hoặc dữ liệu quá khứ để cải thiện hiệu suất công việc trong tương lai, thay vì chỉ thực hiện theo những quy tắc đã được lập trình sẵn. Ví dụ, máy tính có thể học cách dự đoán dựa trên các mẫu đã thấy hoặc phát triển hành vi thích hợp dựa trên quan sát trước đó.

**Ví dụ**:

1. **Chơi cờ**: Một chương trình máy tính có thể quan sát các ván cờ cùng với kết quả (thắng hay thua) để nâng cao khả năng chơi cờ, từ đó gia tăng tỷ lệ chiến thắng trong tương lai. Ở đây, kinh nghiệm được rút ra từ các ván cờ trước đó (có thể là do chương trình tự chơi) để cải thiện kỹ năng chơi cờ, với tiêu chí đánh giá là số ván thắng.
2. **Nhận dạng ký tự**: Một chương trình học nhận dạng ký tự sẽ được cung cấp dữ liệu dưới dạng hình ảnh của các ký tự cùng với mã UNICODE tương ứng. Sau khi hoàn thành quá trình học, chương trình có khả năng nhận dạng và xác định mã UNICODE của các hình ảnh ký tự mới chưa được học.

Giống như quá trình học của con người, một hệ thống học máy cần có khả năng ghi nhớ, thích ứng, và đặc biệt là tổng quát hóa. Tổng quát hóa là khả năng của hệ thống ra quyết định chính xác trong các tình huống mới, dựa trên kinh nghiệm học được từ dữ liệu hoặc quan sát trước đó.

* **Tại sao cần học máy?**

Học máy là một lĩnh vực nghiên cứu quan trọng trong trí tuệ nhân tạo với nhiều ứng dụng thành công trong thực tế. Một số lý do giải thích cho sự cần thiết và phát triển của học máy bao gồm:

1. **Khó khăn trong việc xây dựng hệ thống thông minh**: Việc phát triển hệ thống thực hiện các công việc liên quan đến trí tuệ, như thị giác máy hay xử lý ngôn ngữ tự nhiên, thường gặp nhiều khó khăn nếu không dựa vào kinh nghiệm và quá trình học. Trong nhiều trường hợp, việc tạo ra một thuật toán rõ ràng để chuyển đổi đầu vào thành đầu ra là rất phức tạp. Ví dụ, mặc dù con người có khả năng nhận diện chữ cái rất tốt, nhưng lại khó khăn trong việc giải thích cơ chế dẫn đến kết luận đó. Học máy giúp tìm ra giải pháp cho các trường hợp này bằng cách dựa trên dữ liệu và xác định các điểm chung và riêng giữa nhiều hình ảnh.
2. **Khả năng thích nghi**: Nhiều ứng dụng yêu cầu chương trình máy tính có khả năng thích ứng với các biến đổi trong hành vi của người dùng, chẳng hạn như hành vi mua sắm có thể thay đổi theo thời gian trong ngày, trong năm hoặc theo độ tuổi. Việc xây dựng một thuật toán cố định cho các ứng dụng cần tính linh hoạt là không khả thi. Học máy cung cấp khả năng thích ứng thông qua việc phân tích dữ liệu thu thập được.
3. **Khó khăn trong việc tìm kiếm chuyên gia**: Việc tìm kiếm chuyên gia và thu thập tri thức cần thiết để thiết kế thuật toán giải quyết các vấn đề phức tạp là một thách thức. Trong khi đó, dữ liệu ngày càng nhiều và dễ dàng thu thập hơn. Khả năng lưu trữ và tính toán của máy tính cũng đang gia tăng, cho phép thực hiện các thuật toán học máy trên các tập dữ liệu lớn.
4. **Học là một hoạt động trí tuệ quan trọng**: Khả năng học hỏi là một trong những hoạt động trí tuệ cốt lõi của con người, do đó, học tự động hay học máy luôn thu hút sự quan tâm khi xây dựng các hệ thống thông minh.

## Ứng dụng của Học máy

Học máy có rất nhiều ứng dụng thực tế, với hai lĩnh vực chính là **khai phá dữ liệu (data mining)** và **nhận dạng mẫu (pattern recognition)**.

**Khai phá dữ liệu** là ứng dụng các kỹ thuật học máy vào cơ sở dữ liệu hoặc các tập dữ liệu lớn nhằm phát hiện quy luật hay tri thức trong dữ liệu, cũng như dự đoán thông tin quan tâm trong tương lai. Chẳng hạn, từ một tập hợp hóa đơn bán hàng, có thể phát hiện quy luật “những người mua bánh mì thường mua bơ”.

**Nhận dạng mẫu** là việc áp dụng các kỹ thuật học máy để tìm ra các mẫu có tính quy luật trong dữ liệu, thường liên quan đến hình ảnh và âm thanh. Một bài toán cụ thể trong nhận dạng mẫu là xác định nhãn cho đầu vào, ví dụ, trong trường hợp ảnh chụp khuôn mặt, cần xác định danh tính của người trong ảnh.

Cần lưu ý rằng khai phá dữ liệu và nhận dạng mẫu có nhiều điểm chung cả trong nghiên cứu và ứng dụng. Sự khác biệt chủ yếu liên quan đến lĩnh vực ứng dụng và kỹ thuật sử dụng: khai phá dữ liệu thường liên quan đến dữ liệu thương mại, trong khi nhận dạng mẫu tập trung vào dữ liệu hình ảnh và âm thanh, thường được sử dụng trong các lĩnh vực kỹ thuật.

* **Ứng dụng Cụ thể**

Dưới đây là một số ví dụ cụ thể về ứng dụng của học máy:

1. **Nhận dạng ký tự**: Phân loại hình ảnh ký tự thành các loại, mỗi loại tương ứng với một ký tự cụ thể.
2. **Phát hiện và nhận dạng khuôn mặt**: Nhận diện vùng có chứa mặt người trong ảnh và xác định danh tính của người đó từ những bức ảnh đã biết trước.
3. **Lọc thư rác và phân loại văn bản**: Dựa trên nội dung của thư điện tử để phân loại thành “thư rác” hoặc “thư bình thường”; hoặc phân chia tin tức thành các thể loại như “xã hội”, “kinh tế”, “thể thao”, v.v.
4. **Dịch tự động**: Hệ thống học cách dịch từ ngôn ngữ này sang ngôn ngữ khác dựa trên dữ liệu huấn luyện là các văn bản song ngữ. Google Translate là một ví dụ tiêu biểu.
5. **Chẩn đoán y tế**: Hệ thống học cách dự đoán khả năng mắc bệnh dựa trên các triệu chứng quan sát được.
6. **Phân loại khách hàng và dự đoán sở thích**: Sắp xếp khách hàng vào các loại khác nhau và dự đoán sở thích tiêu dùng của họ.
7. **Dự đoán chỉ số thị trường**: Dựa vào giá trị của một số tham số hiện tại và trong quá khứ để đưa ra dự đoán, chẳng hạn như dự đoán giá chứng khoán hoặc giá vàng.
8. **Hệ thống khuyến nghị**: Cung cấp danh sách các sản phẩm, phim, video, hoặc tin tức mà người dùng có khả năng quan tâm. Ví dụ như phần khuyến nghị trên YouTube hay Amazon.
9. **Ứng dụng lái xe tự động**: Dựa vào các mẫu học từ thông tin về các tình huống trên đường, hệ thống học máy có khả năng tự quyết định điều khiển xe mà không cần người lái. Google đang có kế hoạch thương mại hóa xe ô tô tự lái dựa trên công nghệ này.

## Các Dạng Học Máy

Khi thiết kế và xây dựng hệ thống học máy, cần chú ý đến một số yếu tố quan trọng:

* **Thứ nhất**, hình thức mà kinh nghiệm hoặc dữ liệu được cung cấp cho hệ thống học máy.
* **Thứ hai**, cách thức biểu diễn hàm đích. Hàm đích có thể được biểu diễn dưới dạng hàm đại số, nhưng cũng có thể được thể hiện dưới các dạng khác như cây, mạng nơron, hay công thức xác suất.

Việc sử dụng các dạng kinh nghiệm và cách biểu diễn khác nhau dẫn đến các dạng học máy khác nhau. Dưới đây là ba dạng học máy chính:

1. **Học có giám sát (Supervised Learning)**: Đây là dạng học máy trong đó tập dữ liệu huấn luyện được cung cấp dưới dạng các ví dụ kèm theo giá trị đầu ra hoặc giá trị đích. Dựa trên dữ liệu huấn luyện này, thuật toán học sẽ xây dựng một mô hình hoặc hàm đích để dự đoán giá trị đầu ra cho các trường hợp mới.
   * Nếu giá trị đầu ra là rời rạc, quá trình học có giám sát được gọi là **phân loại (Classification)**.
   * Nếu giá trị đầu ra là liên tục (số thực), quá trình này được gọi là **hồi quy (Regression)**. Phần sau của báo cáo sẽ đi sâu vào chi tiết về học có giám sát.
2. **Học không giám sát (Unsupervised Learning)**: Trong dạng học này, các ví dụ được cung cấp mà không có giá trị đầu ra. Thay vì xác định giá trị đích, thuật toán học máy sẽ dựa vào độ tương tự giữa các ví dụ để phân nhóm chúng thành những cụm có đặc điểm tương đồng. Quá trình phân nhóm này được gọi là **phân cụm (Clustering)**. Ví dụ, chỉ bằng cách quan sát chiều cao của mọi người, ta có thể học được khái niệm “người cao” và “người thấp”, từ đó phân chia mọi người thành hai nhóm tương ứng.
   * Một dạng học không giám sát phổ biến khác là **phát hiện luật kết hợp (Association Rule)**. Luật này được thể hiện dưới dạng P(A | B), cho thấy xác suất của hai thuộc tính A và B xuất hiện cùng nhau. Ví dụ, từ việc phân tích dữ liệu mua hàng tại siêu thị, ta có thể có luật P(Bơ | Bánh mì) = 80%, có nghĩa là 80% những người mua bánh mì cũng mua bơ.
3. **Học tăng cường (Reinforcement Learning)**: Đối với dạng học này, kinh nghiệm không được cung cấp trực tiếp dưới dạng đầu vào/đầu ra cho mỗi trạng thái hoặc hành động. Thay vào đó, hệ thống nhận một giá trị khuyến khích (reward) như là kết quả của một chuỗi hành động. Thuật toán cần học cách hành động để tối đa hóa giá trị khuyến khích. Một ví dụ điển hình cho học tăng cường là học đánh cờ, trong đó hệ thống không được chỉ dẫn cụ thể nước đi nào là hợp lý mà chỉ biết kết quả toàn bộ ván cờ. Các chỉ dẫn về nước đi được cung cấp một cách gián tiếp và có độ trễ dưới dạng giá trị thưởng. Nước đi tốt là nước đi nằm trong chuỗi các nước dẫn đến chiến thắng toàn bộ ván cờ.

Trong số các dạng học máy, học có giám sát là phổ biến nhất, với nhiều thuật toán liên quan và ứng dụng thực tế rộng rãi.

## Thuật toán cây quyết định

### Mô Tả Chi Tiết về Cây Quyết Định

Cây quyết định là một kỹ thuật học máy có giám sát. Tại gốc của cây, chúng ta có thể tìm hiểu về tần suất của các yếu tố mà chúng ta đang cố gắng dự đoán. Cây quyết định phân chia dữ liệu thành các nhóm dựa trên các biến quan trọng nhất để đưa ra kết quả dự đoán. Quá trình này tiếp tục phân nhánh bằng cách sử dụng thêm nhiều biến cho đến khi thuật toán quyết định dừng lại. Cuối cùng, cây sẽ đạt đến các nút lá, đại diện cho một phần nhỏ của tập dữ liệu tổng thể, nơi có sự tập trung cao hoặc thấp về yếu tố mà chúng ta đang cố gắng dự đoán. Các nút lá này có thể được chuyển đổi thành các câu lệnh điều kiện if-then, giúp dễ dàng diễn giải.

### Ưu và Nhược Điểm của Cây Quyết Định

Cây quyết định mang lại nhiều lợi ích, bao gồm khả năng giảm thiểu dữ liệu, khám phá thông tin và xử lý đa dạng các vấn đề liên quan đến dữ liệu. Chúng cũng dễ dàng triển khai, cho phép biến các nút lá thành chuỗi các câu lệnh điều kiện if-then. Tuy nhiên, cây quyết định cũng tồn tại một số nhược điểm, như sự tham lam trong thuật toán, kích thước lớn và độ phức tạp của cây, cũng như độ chính xác có thể thấp hơn so với các phương pháp hiện đại khác.

Theo quan điểm của tôi, mặc dù cây quyết định có những nhược điểm nhất định, chúng vẫn là công cụ hữu ích trong quá trình khám phá dữ liệu và xây dựng các mô hình dự đoán đơn giản. Việc áp dụng cây quyết định là bước quan trọng để hiểu rõ hơn về học máy (machine learning) trước khi chuyển sang các kỹ thuật phức tạp hơn.

## Thuật toán KNN

### Thuật toán KNN là gì?

K-Thuật toán hàng xóm gần nhất (KNN) là một trong những thuật toán học máy phổ biến nhất nhờ vào tính đơn giản của nó. KNN là một thuật toán học có giám sát, hoạt động dựa trên nguyên tắc rằng mọi điểm dữ liệu gần nhau đều thuộc về cùng một lớp. Giả định cơ bản của thuật toán này là các đối tượng gần nhau thường có sự tương đồng. Hầu hết, thuật toán KNN được ưa chuộng vì dễ giải thích và thời gian tính toán nhanh chóng.

### Ưu điểm và nhược điểm của KNN

* Ưu điểm của KNN:
  + Tính toán nhanh chóng.
  + Thuật toán đơn giản và dễ hiểu.
  + Đa năng, hữu ích cho cả phân loại và hồi quy.
  + Độ chính xác cao.
  + Không yêu cầu giả định về dữ liệu, không cần xây dựng mô hình bổ sung.
* Nhược điểm của KNN:
  + Độ chính xác phụ thuộc vào chất lượng của dữ liệu.
  + Dự đoán có thể trở nên chậm khi xử lý dữ liệu lớn.
  + Không phù hợp cho tập dữ liệu có kích thước lớn.
  + Cần phải lưu trữ toàn bộ dữ liệu đào tạo, dẫn đến yêu cầu bộ nhớ cao.
  + Tốn kém về mặt tính toán do phải lưu trữ tất cả các khóa đào tạo.

# **THỰC HIỆN BÀI TOÁN**

## Importing Libraries

* **Mô tả tổng quát**

Đoạn mã trên sử dụng các thư viện phổ biến trong Python để thực hiện các bước đầu tiên trong quá trình phân tích dữ liệu ung thư vú từ tệp CSV có tên breast\_cancer.csv. Mục tiêu chính của đoạn mã là đọc dữ liệu, xử lý các cảnh báo, và hiển thị một số thông tin cơ bản về tập dữ liệu này.

* **Chi tiết các bước thực hiện**

1. **Nhập Thư Viện**:
   * **pandas**: Được sử dụng để thao tác và phân tích dữ liệu. Nó cung cấp các cấu trúc dữ liệu linh hoạt như DataFrame, giúp dễ dàng quản lý và xử lý dữ liệu dạng bảng.
   * **numpy**: Cung cấp các công cụ cho các phép toán số học. Thư viện này hỗ trợ việc xử lý mảng và thực hiện các phép toán hiệu quả trên chúng.
   * **seaborn**: Một thư viện trực quan hóa dữ liệu dựa trên Matplotlib, giúp tạo ra các biểu đồ và hình ảnh đẹp hơn với cú pháp dễ sử dụng.
   * **matplotlib.pyplot**: Thư viện đồ họa mạnh mẽ cho phép vẽ biểu đồ, đồ thị và hình ảnh.
   * **missingno**: Thư viện giúp trực quan hóa các giá trị thiếu trong DataFrame, giúp người dùng dễ dàng xác định các vấn đề liên quan đến dữ liệu không đầy đủ.
2. **Xử Lý Cảnh Báo**:
   * Sử dụng warnings.filterwarnings('ignore') để bỏ qua các cảnh báo có thể phát sinh trong quá trình chạy mã, giúp đầu ra sạch sẽ hơn mà không bị làm rối bởi các thông báo không cần thiết.
3. **Thiết Lập Kiểu Đồ Thị**:
   * plt.style.use('ggplot') thiết lập kiểu vẽ đồ thị theo phong cách ggplot, mang lại cho biểu đồ một cái nhìn hiện đại và chuyên nghiệp hơn.
4. **Đọc Dữ Liệu**:
   * df = pd.read\_csv("breast\_cancer.csv") đọc dữ liệu từ tệp CSV có tên breast\_cancer.csv và lưu nó vào biến df dưới dạng một DataFrame. Điều này cho phép người dùng dễ dàng thao tác với dữ liệu trong Python.
5. **Hiển Thị Thông Tin Cơ Bản**:
   * print(df.head()) sẽ in ra 5 dòng đầu tiên của DataFrame, cho phép người dùng xem nhanh cấu trúc và nội dung của dữ liệu, bao gồm các cột và giá trị tương ứng. Đây là một bước quan trọng trong việc khám phá dữ liệu, giúp hiểu rõ hơn về các biến và kiểu dữ liệu.

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import missingno as msno

import warnings

warnings.filterwarnings('ignore')

plt.style.use('ggplot')

df = pd.read\_csv("breast\_cancer.csv")

print(df.head())

id diagnosis radius\_mean texture\_mean perimeter\_mean area\_mean \

0 842302 M 17.99 10.38 122.80 1001.0

1 842517 M 20.57 17.77 132.90 1326.0

2 84300903 M 19.69 21.25 130.00 1203.0

3 84348301 M 11.42 20.38 77.58 386.1

4 84358402 M 20.29 14.34 135.10 1297.0

smoothness\_mean compactness\_mean concavity\_mean concave points\_mean \

0 0.11840 0.27760 0.3001 0.14710

1 0.08474 0.07864 0.0869 0.07017

2 0.10960 0.15990 0.1974 0.12790

3 0.14250 0.28390 0.2414 0.10520

4 0.10030 0.13280 0.1980 0.10430

... radius\_worst texture\_worst perimeter\_worst area\_worst \

0 ... 25.38 17.33 184.60 2019.0

1 ... 24.99 23.41 158.80 1956.0

2 ... 23.57 25.53 152.50 1709.0

3 ... 14.91 26.50 98.87 567.7

4 ... 22.54 16.67 152.20 1575.0

smoothness\_worst compactness\_worst concavity\_worst concave points\_worst \

0 0.1622 0.6656 0.7119 0.2654

1 0.1238 0.1866 0.2416 0.1860

2 0.1444 0.4245 0.4504 0.2430

3 0.2098 0.8663 0.6869 0.2575

4 0.1374 0.2050 0.4000 0.1625

symmetry\_worst fractal\_dimension\_worst

0 0.4601 0.11890

1 0.2750 0.08902

2 0.3613 0.08758

3 0.6638 0.17300

4 0.2364 0.07678

[5 rows x 32 columns]

## Data Preprocessing

df.diagnosis.unique()

df.describe()

id radius\_mean texture\_mean perimeter\_mean area\_mean \

count 5.690000e+02 569.000000 569.000000 569.000000 569.000000

mean 3.037183e+07 14.127292 19.289649 91.969033 654.889104

std 1.250206e+08 3.524049 4.301036 24.298981 351.914129

min 8.670000e+03 6.981000 9.710000 43.790000 143.500000

25% 8.692180e+05 11.700000 16.170000 75.170000 420.300000

50% 9.060240e+05 13.370000 18.840000 86.240000 551.100000

75% 8.813129e+06 15.780000 21.800000 104.100000 782.700000

max 9.113205e+08 28.110000 39.280000 188.500000 2501.000000

smoothness\_mean compactness\_mean concavity\_mean concave points\_mean \

count 569.000000 569.000000 569.000000 569.000000

mean 0.096360 0.104341 0.088799 0.048919

std 0.014064 0.052813 0.079720 0.038803

min 0.052630 0.019380 0.000000 0.000000

25% 0.086370 0.064920 0.029560 0.020310

50% 0.095870 0.092630 0.061540 0.033500

75% 0.105300 0.130400 0.130700 0.074000

max 0.163400 0.345400 0.426800 0.201200

symmetry\_mean ... radius\_worst texture\_worst perimeter\_worst \

count 569.000000 ... 569.000000 569.000000 569.000000

mean 0.181162 ... 16.269190 25.677223 107.261213

std 0.027414 ... 4.833242 6.146258 33.602542

min 0.106000 ... 7.930000 12.020000 50.410000

25% 0.161900 ... 13.010000 21.080000 84.110000

50% 0.179200 ... 14.970000 25.410000 97.660000

75% 0.195700 ... 18.790000 29.720000 125.400000

max 0.304000 ... 36.040000 49.540000 251.200000

area\_worst smoothness\_worst compactness\_worst concavity\_worst \

count 569.000000 569.000000 569.000000 569.000000

mean 880.583128 0.132369 0.254265 0.272188

std 569.356993 0.022832 0.157336 0.208624

min 185.200000 0.071170 0.027290 0.000000

25% 515.300000 0.116600 0.147200 0.114500

50% 686.500000 0.131300 0.211900 0.226700

75% 1084.000000 0.146000 0.339100 0.382900

max 4254.000000 0.222600 1.058000 1.252000

concave points\_worst symmetry\_worst fractal\_dimension\_worst

count 569.000000 569.000000 569.000000

mean 0.114606 0.290076 0.083946

std 0.065732 0.061867 0.018061

min 0.000000 0.156500 0.055040

25% 0.064930 0.250400 0.071460

50% 0.099930 0.282200 0.080040

75% 0.161400 0.317900 0.092080

max 0.291000 0.663800 0.207500

[8 rows x 31 columns]

Đoạn mã trên thực hiện một số thao tác để khám phá và phân tích dữ liệu trong DataFrame df. Đầu tiên, nó kiểm tra các giá trị duy nhất trong cột diagnosis, nhằm xác định các loại chẩn đoán khác nhau có trong dữ liệu, từ đó giúp phân loại bệnh nhân thành các nhóm ác tính và lành tính. Điều này cho thấy rằng cột diagnosis sẽ là mục tiêu (target) trong bối cảnh học máy có giám sát (supervised learning).

Tiếp theo, mã sử dụng phương thức describe() để cung cấp thống kê mô tả cho các cột số trong DataFrame. Thống kê này bao gồm số lượng, giá trị trung bình, độ lệch chuẩn, giá trị nhỏ nhất, giá trị lớn nhất và các phần trăm phân vị, cho phép người dùng có cái nhìn tổng quan về phân phối và đặc điểm của các biến trong tập dữ liệu. Qua đó, người dùng có thể đánh giá chất lượng dữ liệu và xác định các yếu tố quan trọng cần xem xét trong quá trình phân tích và xây dựng mô hình học máy.

## Missing Value

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 569 entries, 0 to 568

Data columns (total 32 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 id 569 non-null int64

1 diagnosis 569 non-null object

2 radius\_mean 569 non-null float64

3 texture\_mean 569 non-null float64

4 perimeter\_mean 569 non-null float64

5 area\_mean 569 non-null float64

6 smoothness\_mean 569 non-null float64

7 compactness\_mean 569 non-null float64

8 concavity\_mean 569 non-null float64

9 concave points\_mean 569 non-null float64

10 symmetry\_mean 569 non-null float64

...

30 symmetry\_worst 569 non-null float64

31 fractal\_dimension\_worst 569 non-null float64

dtypes: float64(30), int64(1), object(1)

memory usage: 142.4+ KB

----------------------------------------------

df.isnull().sum()

id 0

diagnosis 0

radius\_mean 0

texture\_mean 0

perimeter\_mean 0

area\_mean 0

smoothness\_mean 0

compactness\_mean 0

radius\_se 0

texture\_se 0

perimeter\_se 0

area\_se 0

smoothness\_se 0

compactness\_se 0

concavity\_se 0

radius\_worst 0

texture\_worst 0

...

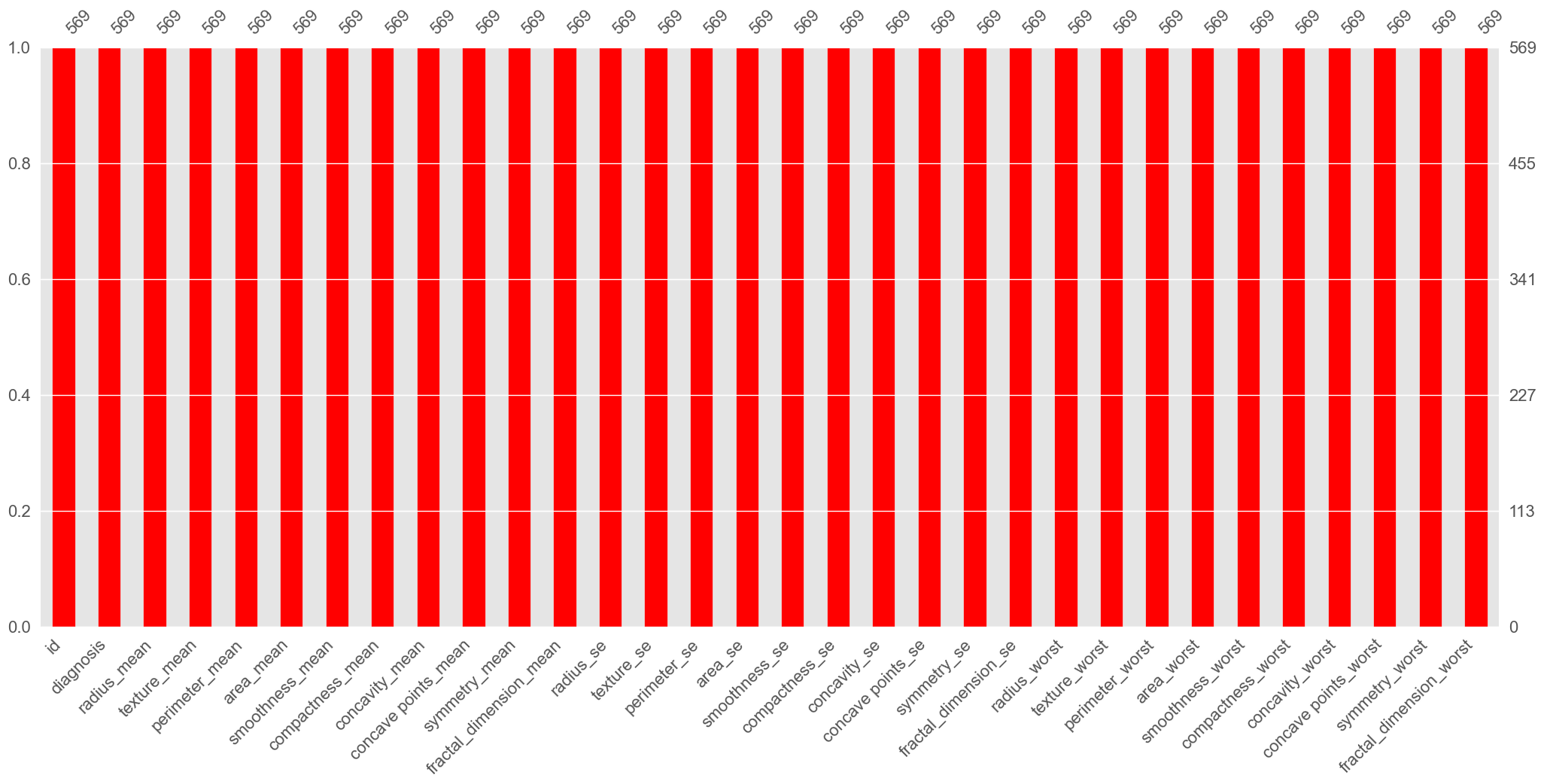
symmetry\_worst 0

fractal\_dimension\_worst 0

dtype: int64

----------------------------------------------

msno.bar(df, color="red")



----------------------------------------------

df['diagnosis'] = df['diagnosis'].apply(lambda val:1 if val=='M' else 0)

plt.hist(df['diagnosis'])

plt.title('Diagnosis(M=1, B=0)')

plt.show()

Ảnh có chứa văn bản, ảnh chụp màn hình, Sơ đồ, biểu đồ

Mô tả được tạo tự động

Đoạn mã trên thực hiện một số thao tác để phân tích và trực quan hóa dữ liệu trong DataFrame df. Đầu tiên, hàm info() được sử dụng để cung cấp thông tin tổng quan về DataFrame, bao gồm số lượng các mục, kiểu dữ liệu của từng cột và số lượng giá trị không null trong mỗi cột. Điều này giúp đánh giá cấu trúc và tính toàn vẹn của dữ liệu.

Tiếp theo, mã sử dụng phương thức isnull().sum() để kiểm tra số lượng giá trị null trong từng cột của DataFrame, từ đó xác định các biến nào có dữ liệu thiếu và cần xử lý.

Để trực quan hóa số lượng giá trị null, mã sử dụng thư viện missingno với hàm bar(), cho phép người dùng dễ dàng nhận thấy mức độ thiếu dữ liệu trong DataFrame thông qua biểu đồ cột. Màu sắc đỏ được sử dụng để nhấn mạnh các giá trị null.

Sau đó, cột diagnosis được chuyển đổi từ các ký tự (M cho ác tính và B cho lành tính) thành các giá trị số (1 cho ác tính và 0 cho lành tính) để phục vụ cho quá trình phân tích và xây dựng mô hình học máy. Cuối cùng, một biểu đồ histogram được vẽ để hiển thị phân bố của các chẩn đoán, với tiêu đề rõ ràng thể hiện ý nghĩa của các giá trị trong biểu đồ (M=1, B=0). Biểu đồ này giúp người dùng nhanh chóng nhận diện số lượng bệnh nhân thuộc các loại chẩn đoán khác nhau.

## EDA

Đoạn mã trên tạo ra các biểu đồ phân bố (density plots) cho từng cột trong DataFrame df bằng cách sử dụng thư viện Seaborn (sns.distplot) và Matplotlib. Dưới đây là tóm tắt các thao tác trong đoạn mã:

1. **Kích thước của biểu đồ**: Hàm plt.figure(figsize=(20,15)) thiết lập kích thước cho hình ảnh đầu ra với chiều rộng là 20 và chiều cao là 15, nhằm tạo không gian đủ để hiển thị các biểu đồ phân bố.
2. **Vòng lặp for**: Đoạn mã sử dụng một vòng lặp for để duyệt qua từng cột trong DataFrame df. Biểu đồ phân bố được vẽ cho mỗi cột dữ liệu.
3. **Giới hạn số lượng biểu đồ**: Vòng lặp cho phép tạo tối đa 30 biểu đồ (5 hàng và 6 cột) bằng cách kiểm tra điều kiện if plotnumber<=30. Điều này đảm bảo rằng chỉ có 30 biểu đồ được vẽ.
4. **Vẽ biểu đồ phân bố**: Trong mỗi lần lặp, sns.distplot(df[column]) vẽ biểu đồ phân bố cho từng cột trong DataFrame. Biểu đồ này giúp hiển thị phân bố giá trị của cột đó.
5. **Hiển thị tên cột**: plt.xlabel(column) đặt nhãn cho trục x của mỗi biểu đồ tương ứng với tên của cột.
6. **Cấu trúc biểu đồ gọn gàng**: plt.tight\_layout() đảm bảo rằng các biểu đồ không bị chồng lấn lên nhau, tối ưu hóa không gian giữa các biểu đồ.
7. **Hiển thị biểu đồ**: Cuối cùng, plt.show() được gọi để hiển thị toàn bộ các biểu đồ đã vẽ trên màn hình.

plt.figure(figsize=(20,15))

plotnumber=1

for column in df:

    if plotnumber<=30:

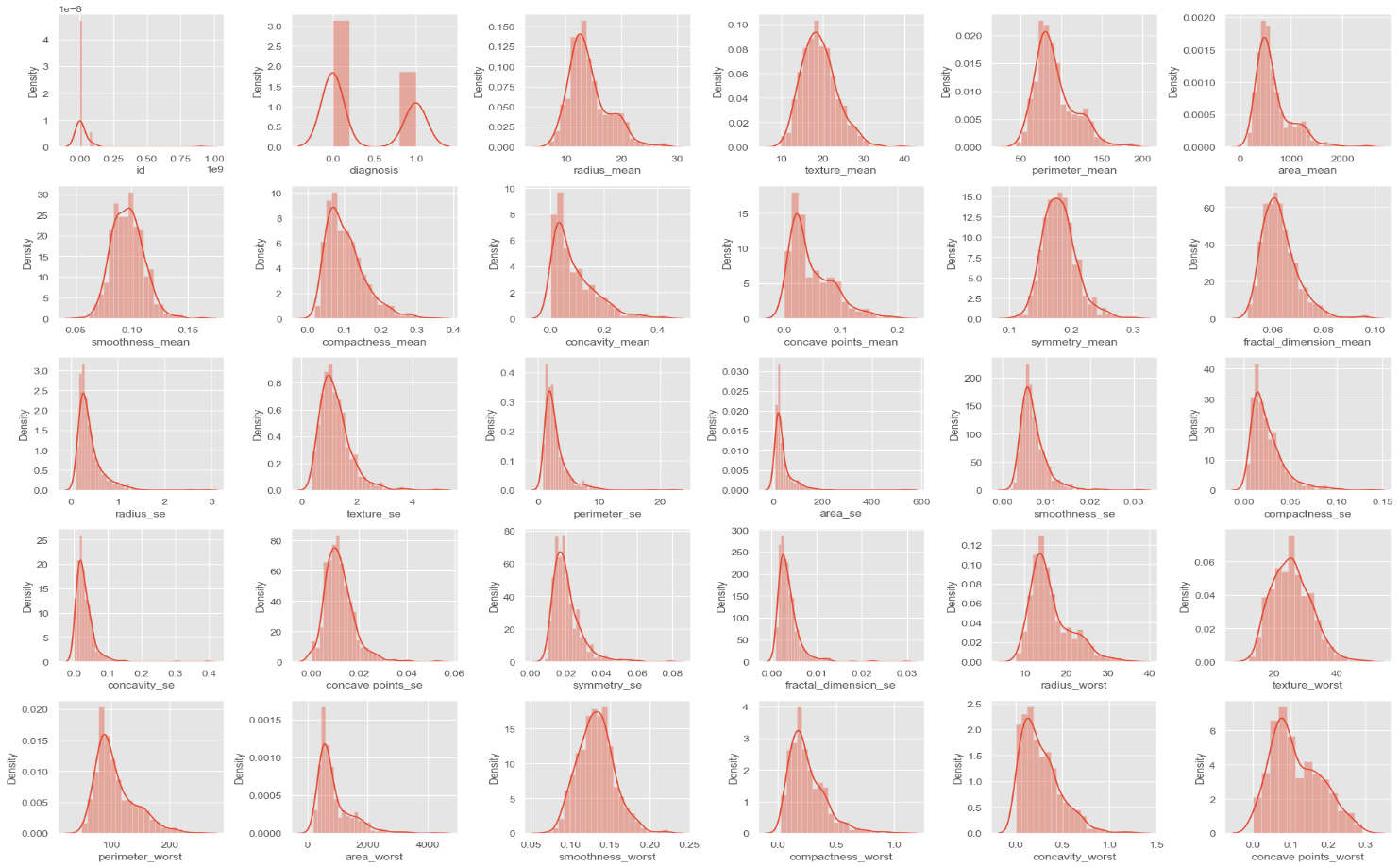
        ax = plt.subplot(5,6, plotnumber)

        sns.distplot(df[column])

        plt.xlabel(column)

    plotnumber+=1

plt.tight\_layout()

plt.show()

## CORR

df.corr()

id diagnosis radius\_mean texture\_mean \

id 1.000000 0.039769 0.074626 0.099770

diagnosis 0.039769 1.000000 0.730029 0.415185

radius\_mean 0.074626 0.730029 1.000000 0.323782

texture\_mean 0.099770 0.415185 0.323782 1.000000

perimeter\_mean 0.073159 0.742636 0.997855 0.329533

area\_mean 0.096893 0.708984 0.987357 0.321086

smoothness\_mean -0.012968 0.358560 0.170581 -0.023389

compactness\_mean 0.000096 0.596534 0.506124 0.236702

concavity\_mean 0.050080 0.696360 0.676764 0.302418

concave points\_mean 0.044158 0.776614 0.822529 0.293464

symmetry\_mean -0.022114 0.330499 0.147741 0.071401

fractal\_dimension\_mean -0.052511 -0.012838 -0.311631 -0.076437

radius\_se 0.143048 0.567134 0.679090 0.275869

texture\_se -0.007526 -0.008303 -0.097317 0.386358

perimeter\_se 0.137331 0.556141 0.674172 0.281673

area\_se 0.177742 0.548236 0.735864 0.259845

smoothness\_se 0.096781 -0.067016 -0.222600 0.006614

compactness\_se 0.033961 0.292999 0.206000 0.191975

concavity\_se 0.055239 0.253730 0.194204 0.143293

concave points\_se 0.078768 0.408042 0.376169 0.163851

symmetry\_se -0.017306 -0.006522 -0.104321 0.009127

fractal\_dimension\_se 0.025725 0.077972 -0.042641 0.054458

radius\_worst 0.082405 0.776454 0.969539 0.352573

texture\_worst 0.064720 0.456903 0.297008 0.912045

...

concave points\_worst 0.511114

symmetry\_worst 0.537848

fractal\_dimension\_worst 1.000000

------------------------------------------------------------

df.corr()

# heatmap

plt.figure(figsize=(20,12))

corr=df.corr()

mask = np.triu(np.ones\_like(corr, dtype=bool))

sns.heatmap(corr, mask=mask, linewidths=1, annot=True, fmt = ".2f")

plt.show()

Ảnh có chứa văn bản, ảnh chụp màn hình, Sơ đồ, mũi khâu

Mô tả được tạo tự động

df.drop('id', axis=1, inplace=True)

# feature selection

corr\_matrix = df.corr().abs()

mask = np.triu(np.ones\_like(corr\_matrix, dtype=bool))

tri\_df = corr\_matrix.mask(mask)

to\_drop = [x for x in tri\_df.columns if any(tri\_df[x]>0.92)]

df = df.drop(to\_drop, axis=1)

print(df.shape[1])

df.head()

diagnosis texture\_mean smoothness\_mean compactness\_mean \

0 1 10.38 0.11840 0.27760

1 1 17.77 0.08474 0.07864

2 1 21.25 0.10960 0.15990

3 1 20.38 0.14250 0.28390

4 1 14.34 0.10030 0.13280

concave points\_mean symmetry\_mean fractal\_dimension\_mean texture\_se \

0 0.14710 0.2419 0.07871 0.9053

1 0.07017 0.1812 0.05667 0.7339

2 0.12790 0.2069 0.05999 0.7869

3 0.10520 0.2597 0.09744 1.1560

4 0.10430 0.1809 0.05883 0.7813

area\_se smoothness\_se ... symmetry\_se fractal\_dimension\_se \

0 153.40 0.006399 ... 0.03003 0.006193

1 74.08 0.005225 ... 0.01389 0.003532

2 94.03 0.006150 ... 0.02250 0.004571

3 27.23 0.009110 ... 0.05963 0.009208

4 94.44 0.011490 ... 0.01756 0.005115

texture\_worst area\_worst smoothness\_worst compactness\_worst \

0 17.33 2019.0 0.1622 0.6656

1 23.41 1956.0 0.1238 0.1866

2 25.53 1709.0 0.1444 0.4245

...

2 0.08758

3 0.17300

4 0.07678

Đoạn mã trên thực hiện các thao tác sau:

1. **Tính toán ma trận tương quan** (df.corr()): Mã tính toán ma trận tương quan cho tất cả các cột trong tập dữ liệu df, giúp xác định mối liên hệ giữa các biến. Giá trị tương quan dao động từ -1 đến 1, với 1 là tương quan hoàn hảo, -1 là tương quan âm hoàn hảo và 0 là không có tương quan.
2. **Vẽ Heatmap**: Sử dụng thư viện seaborn, mã tạo ra biểu đồ heatmap để trực quan hóa ma trận tương quan:
   * Kích thước biểu đồ là 20x12.
   * Mặt nạ được áp dụng để chỉ hiển thị một nửa ma trận nhằm tránh lặp lại thông tin tương quan.
   * Các giá trị tương quan được hiển thị bằng số trong biểu đồ.
3. **Lựa chọn đặc trưng (Feature Selection)**:
   * Mã loại bỏ cột id vì không có giá trị trong việc phân tích mô hình.
   * Tính ma trận tương quan với các giá trị tuyệt đối, sau đó áp dụng mặt nạ tương tự để chỉ hiển thị nửa dưới của ma trận.
   * Các cột có độ tương quan lớn hơn 0.92 được đánh dấu và loại bỏ để giảm sự đa cộng tuyến trong mô hình.
4. **Kết quả**: Cuối cùng, mã in ra số lượng cột còn lại trong tập dữ liệu và hiển thị những hàng đầu tiên để kiểm tra sau khi loại bỏ các cột có tương quan cao.

Tóm lại, đoạn mã thực hiện các bước tính toán và trực quan hóa mối quan hệ giữa các biến, sau đó áp dụng lựa chọn đặc trưng để tối ưu hóa tập dữ liệu cho các mô hình học máy.

## Buliding Model

X=df.drop('diagnosis', axis=1)

y=df['diagnosis']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train ,y\_test =train\_test\_split(X,y, test\_size=0.2, random\_state=0)

# scaling data

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

X\_train.shape

(455, 22)

Đoạn mã trên thực hiện các bước chuẩn bị dữ liệu cho mô hình học máy như sau:

1. **Tách dữ liệu**:
   * X chứa tất cả các đặc trưng của tập dữ liệu (các cột không bao gồm diagnosis).
   * y là biến mục tiêu (diagnosis), được sử dụng để dự đoán kết quả.
2. **Chia dữ liệu**:
   * Sử dụng hàm train\_test\_split từ thư viện sklearn.model\_selection để chia dữ liệu thành tập huấn luyện (X\_train, y\_train) và tập kiểm tra (X\_test, y\_test). Tỷ lệ chia là 80% cho tập huấn luyện và 20% cho tập kiểm tra, với random\_state=0 để đảm bảo tính lặp lại.
3. **Chuẩn hóa dữ liệu**:
   * Sử dụng StandardScaler từ sklearn.preprocessing để chuẩn hóa các đặc trưng trong tập huấn luyện. Phép chuẩn hóa này giúp dữ liệu có trung bình bằng 0 và độ lệch chuẩn bằng 1, cải thiện hiệu suất của nhiều thuật toán học máy.
   * Sau đó, áp dụng cùng một phép biến đổi này cho tập kiểm tra (X\_test).
4. **Kiểm tra kích thước của tập huấn luyện**:
   * Cuối cùng, mã in ra kích thước của X\_train, cho biết số lượng mẫu và số đặc trưng mà tập huấn luyện chứa.

Tóm lại, đoạn mã thực hiện việc tách dữ liệu thành tập huấn luyện và kiểm tra, sau đó chuẩn hóa các đặc trưng để chuẩn bị cho quá trình xây dựng mô hình học máy.

## Apply Machine learning Algo

### Thuật toán KNN

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

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

KNeighborsClassifier()

------------------------------------------------------------

y\_pred = knn.predict(X\_test)

y\_pred

array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,

0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0,

0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,

1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,

1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,

0, 1, 1, 0])

------------------------------------------------------------

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

print(accuracy\_score(y\_train, knn.predict(X\_train)))

knn\_acc = accuracy\_score(y\_test, knn.predict(X\_test))

print(knn\_acc)

y\_pred = knn.predict(X\_test)

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

0.967032967032967

0.956140350877193

[[66 1]

[ 4 43]]

precision recall f1-score support

0 0.94 0.99 0.96 67

1 0.98 0.91 0.95 47

accuracy 0.96 114

macro avg 0.96 0.95 0.95 114

weighted avg 0.96 0.96 0.96 114

Đoạn mã trên thực hiện các bước liên quan đến việc xây dựng và đánh giá mô hình phân loại sử dụng thuật toán K-Nearest Neighbors (KNN) như sau:

1. **Khởi tạo mô hình KNN**:
   * Sử dụng lớp KNeighborsClassifier từ thư viện sklearn.neighbors để tạo một đối tượng knn, đại diện cho mô hình phân loại KNN.
2. **Huấn luyện mô hình**:
   * Gọi phương thức fit để huấn luyện mô hình trên tập dữ liệu huấn luyện (X\_train, y\_train).
3. **Dự đoán**:
   * Sử dụng mô hình đã huấn luyện để dự đoán nhãn cho tập kiểm tra (X\_test) bằng cách gọi phương thức predict, kết quả dự đoán được lưu vào biến y\_pred.
4. **Đánh giá mô hình**:
   * Tính toán và in ra độ chính xác (accuracy) của mô hình trên cả tập huấn luyện và tập kiểm tra bằng cách sử dụng hàm accuracy\_score.
   * In ra ma trận nhầm lẫn (confusion matrix) để xem chi tiết về số lượng dự đoán đúng và sai cho từng lớp.
   * In ra báo cáo phân loại (classification report) để cung cấp các chỉ số như độ chính xác, độ nhạy (recall), độ đặc hiệu (precision) và điểm F1 cho từng lớp.

Tóm lại, đoạn mã thực hiện việc xây dựng mô hình KNN, huấn luyện nó trên dữ liệu, thực hiện dự đoán, và cuối cùng đánh giá hiệu suất của mô hình trên tập kiểm tra.

### Thuật toán DT

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import GridSearchCV

dtc = DecisionTreeClassifier()

parameters = {

    'criterion':['gini','entropy'],

    'max\_depth':range(2,32,1),

    'min\_samples\_leaf':range(1,10,1),

    'min\_samples\_split':range(2,10,1),

    'splitter':['best','random']

}

grid\_search\_dt = GridSearchCV(dtc, parameters, cv=5, n\_jobs=-1, verbose=1)

grid\_search\_dt.fit(X\_train, y\_train)

Fitting 5 folds for each of 8640 candidates, totalling 43200 fits

GridSearchCV(cv=5, estimator=DecisionTreeClassifier(), n\_jobs=-1,

param\_grid={'criterion': ['gini', 'entropy'],

'max\_depth': range(2, 32),

'min\_samples\_leaf': range(1, 10),

'min\_samples\_split': range(2, 10),

'splitter': ['best', 'random']},

verbose=1)

------------------------------------------------------------

grid\_search\_dt.best\_params\_

{'criterion': 'entropy',

'max\_depth': 21,

'min\_samples\_leaf': 3,

'min\_samples\_split': 2,

'splitter': 'random'}

------------------------------------------------------------

grid\_search\_dt.best\_score\_

np.float64(0.9670329670329669)

------------------------------------------------------------

dtc = DecisionTreeClassifier(criterion='entropy', max\_depth=29, min\_samples\_leaf=3, min\_samples\_split=4, splitter = 'random')

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

DecisionTreeClassifier(criterion='entropy', max\_depth=29, min\_samples\_leaf=3,

min\_samples\_split=4, splitter='random')

------------------------------------------------------------

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

print(accuracy\_score(y\_train, dtc.predict(X\_train)))

dtc\_acc = accuracy\_score(y\_test, dtc.predict(X\_test))

print(dtc\_acc)

y\_pred = dtc.predict(X\_test)

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

0.9758241758241758

0.956140350877193

[[67 0]

[ 5 42]]

precision recall f1-score support

0 0.93 1.00 0.96 67

1 1.00 0.89 0.94 47

accuracy 0.96 114

macro avg 0.97 0.95 0.95 114

weighted avg 0.96 0.96 0.96 114

Đoạn mã trên thực hiện các bước liên quan đến việc xây dựng và tối ưu hóa mô hình phân loại cây quyết định (Decision Tree) như sau:

1. **Khởi tạo mô hình cây quyết định**:
   * Sử dụng lớp DecisionTreeClassifier từ thư viện sklearn.tree để tạo một đối tượng dtc, đại diện cho mô hình cây quyết định.
2. **Thiết lập tham số cho Grid Search**:
   * Tạo một từ điển parameters chứa các tham số mà mô hình sẽ được tối ưu hóa. Các tham số bao gồm:
     + criterion: tiêu chí phân tách, có thể là 'gini' hoặc 'entropy'.
     + max\_depth: độ sâu tối đa của cây, nằm trong khoảng từ 2 đến 31.
     + min\_samples\_leaf: số lượng mẫu tối thiểu trong một nút lá, từ 1 đến 9.
     + min\_samples\_split: số lượng mẫu tối thiểu để chia một nút, từ 2 đến 9.
     + splitter: cách chọn nút phân tách, có thể là 'best' hoặc 'random'.
3. **Tối ưu hóa mô hình**:
   * Sử dụng GridSearchCV để tìm kiếm các tham số tối ưu cho mô hình, sử dụng 5-fold cross-validation (cv=5). Đối tượng grid\_search\_dt sẽ chứa các thông tin về kết quả tìm kiếm.
   * Gọi phương thức fit để thực hiện quá trình tối ưu hóa trên tập huấn luyện (X\_train, y\_train).
4. **Lấy tham số tốt nhất**:
   * Lưu các tham số tốt nhất và điểm số tốt nhất từ quá trình tìm kiếm.
5. **Huấn luyện mô hình với tham số tối ưu**:
   * Tạo một đối tượng cây quyết định mới với các tham số tốt nhất và huấn luyện nó trên dữ liệu huấn luyện.
6. **Đánh giá mô hình**:
   * Tính toán và in ra độ chính xác (accuracy) của mô hình trên cả tập huấn luyện và tập kiểm tra bằng cách sử dụng hàm accuracy\_score.
   * In ra ma trận nhầm lẫn (confusion matrix) để xem chi tiết về số lượng dự đoán đúng và sai cho từng lớp.
   * In ra báo cáo phân loại (classification report) để cung cấp các chỉ số như độ chính xác, độ nhạy (recall), độ đặc hiệu (precision) và điểm F1 cho từng lớp.

Tóm lại, đoạn mã này thực hiện việc xây dựng mô hình cây quyết định, tối ưu hóa các tham số của mô hình bằng cách sử dụng Grid Search, và cuối cùng đánh giá hiệu suất của mô hình trên tập kiểm tra.

## Model Comparison

from sklearn import metrics

plt.figure(figsize=(8,5))

models = [

{

    'label': 'DT',

    'model': dtc,

},

{

    'label': 'KNN',

    'model': knn,

}

]

for m in models:

    model = m['model']

    model.fit(X\_train, y\_train)

    y\_pred=model.predict(X\_test)

    fpr1, tpr1, thresholds = metrics.roc\_curve(y\_test, model.predict\_proba(X\_test)[:,1])

    auc = metrics.roc\_auc\_score(y\_test,model.predict(X\_test))

    plt.plot(fpr1, tpr1, label='%s - ROC (area = %0.2f)' % (m['label'], auc))

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([-0.01, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('1 - Specificity (False Positive Rate)', fontsize=12)

plt.ylabel('Sensitivity (True Positive Rate)', fontsize=12)

plt.title('ROC - Breast Cancer Prediction', fontsize=12)

plt.legend(loc="lower right", fontsize=12)

plt.savefig("roc\_breast\_cancer.jpeg", format='jpeg', dpi=400, bbox\_inches='tight')

plt.show()

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, Sơ đồ

Mô tả được tạo tự động

Đoạn mã trên thực hiện việc vẽ đồ thị ROC (Receiver Operating Characteristic) cho hai mô hình phân loại: cây quyết định (Decision Tree - DT) và K-Nearest Neighbors (KNN). Các bước cụ thể như sau:

1. **Thiết lập không gian vẽ**:
   * Tạo một hình vẽ mới với kích thước 8x5 inch bằng cách sử dụng plt.figure.
2. **Định nghĩa các mô hình**:
   * Tạo một danh sách chứa các từ điển, mỗi từ điển chứa nhãn (label) và mô hình (model). Trong trường hợp này, danh sách bao gồm hai mô hình: cây quyết định (dtc) và KNN (knn).
3. **Vòng lặp qua các mô hình**:
   * Đối với mỗi mô hình trong danh sách models, thực hiện các bước sau:
     + Huấn luyện mô hình trên dữ liệu huấn luyện (X\_train, y\_train) bằng phương thức fit.
     + Dự đoán giá trị cho tập kiểm tra (X\_test) bằng cách sử dụng phương thức predict.
     + Tính toán đường cong ROC bằng hàm roc\_curve, thu được tỷ lệ dương tính giả (false positive rate - FPR) và tỷ lệ dương tính thật (true positive rate - TPR).
     + Tính toán diện tích dưới đường cong ROC (AUC) bằng cách sử dụng roc\_auc\_score.
4. **Vẽ đường ROC**:
   * Sử dụng plt.plot để vẽ đường ROC cho mỗi mô hình với nhãn tương ứng và giá trị AUC.
5. **Thiết lập đồ thị**:
   * Vẽ một đường chéo đỏ từ điểm (0, 0) đến (1, 1) để biểu thị cho mô hình ngẫu nhiên.
   * Thiết lập giới hạn cho các trục x và y.
   * Đặt nhãn cho các trục, tiêu đề cho đồ thị, và thêm chú thích cho các mô hình.
6. **Lưu và hiển thị đồ thị**:
   * Lưu hình ảnh đồ thị dưới dạng tệp JPEG với độ phân giải cao bằng cách sử dụng plt.savefig.
   * Cuối cùng, gọi plt.show() để hiển thị đồ thị trên màn hình.

Tóm lại, đoạn mã này nhằm mục đích so sánh hiệu suất của hai mô hình phân loại thông qua đồ thị ROC, cho thấy khả năng phân loại chính xác của chúng trong việc dự đoán ung thư vú.

🡪 **Kết luận:** Trong chương này, chúng tôi đã tiến hành phân tích dữ liệu và xây dựng mô hình dự đoán cho ung thư vú. Chúng tôi đã xử lý dữ liệu, loại bỏ các cột không cần thiết và kiểm tra sự thiếu hụt dữ liệu. Sử dụng ma trận tương quan và biểu đồ nhiệt, chúng tôi đã khám phá mối liên hệ giữa các đặc trưng và mục tiêu chẩn đoán.

Sau đó, dữ liệu được chia thành tập huấn luyện và kiểm tra, đồng thời chuẩn hóa để nâng cao độ chính xác. Chúng tôi áp dụng các thuật toán K-Nearest Neighbors (KNN) và cây quyết định (Decision Tree), đánh giá hiệu suất qua độ chính xác và ma trận nhầm lẫn. Cuối cùng, biểu đồ ROC cho thấy cả hai mô hình đều có khả năng phân loại tốt, mở ra triển vọng ứng dụng học máy trong chẩn đoán ung thư vú.

# **KẾT LUẬN**

Trong bài viết này, chúng tôi đã nghiên cứu về việc ứng dụng công nghệ học máy trong chẩn đoán ung thư vú, một căn bệnh nguy hiểm đối với sức khỏe phụ nữ toàn cầu. Qua quá trình phân tích dữ liệu và xây dựng các mô hình dự đoán, chúng tôi đã sử dụng hai thuật toán chính: K-Nearest Neighbors (KNN) và cây quyết định (Decision Tree).

**Ưu điểm** của các mô hình này bao gồm khả năng phân loại chính xác và khả năng xử lý lượng dữ liệu lớn. KNN, với sự đơn giản và dễ hiểu, đã cho kết quả đáng khích lệ, trong khi cây quyết định cung cấp khả năng giải thích rõ ràng thông qua cấu trúc nếu-thì. Cả hai phương pháp đều cho thấy hiệu suất tốt trong việc phân loại và dự đoán kết quả chẩn đoán.

**Nhược điểm** của KNN là độ chính xác phụ thuộc vào chất lượng dữ liệu, và với tập dữ liệu lớn, thời gian dự đoán có thể chậm. Cây quyết định, mặc dù dễ diễn giải, có thể dẫn đến hiện tượng overfitting nếu không được điều chỉnh cẩn thận.

Kết quả đạt được từ việc áp dụng các thuật toán học máy này cho thấy khả năng nâng cao độ chính xác trong chẩn đoán ung thư vú, từ đó hỗ trợ bác sĩ trong việc ra quyết định lâm sàng và cải thiện quy trình điều trị. Việc phát hiện sớm và chính xác sẽ giúp cứu sống hàng triệu người, đồng thời giảm thiểu gánh nặng cho hệ thống y tế.

**Ứng dụng trong đời sống** của công nghệ học máy trong y tế không chỉ dừng lại ở ung thư vú, mà còn mở rộng ra nhiều lĩnh vực khác như chẩn đoán bệnh tim mạch, phân tích hình ảnh y tế và nghiên cứu gen. Việc tích hợp học máy vào quy trình chăm sóc sức khỏe không chỉ mang lại lợi ích cho bệnh nhân mà còn nâng cao hiệu quả hoạt động của các cơ sở y tế, góp phần tạo ra một hệ thống y tế thông minh hơn và hiệu quả hơn trong tương lai.

# **TÀI LIỆU THAM KHẢO**

1. [Breast Cancer Prediction with 7+ Models 98% ACC](lmhttps://www.kaggle.com/code/chanchal24/breast-cancer-prediction-with-7-models-98-acc)
2. [Khám Phá Sâu Về Cây Quyết Định Trong Khoa Học Dữ Liệu và Machine Learning](https://ceoduyho.com/ai/machine-learning/kham-pha-sau-ve-cay-quyet-dinh-trong-khoa-hoc-du-lieu-va-machine-learning.html#:~:text=C%C3%A2y%20quy%E1%BA%BFt%20%C4%91%E1%BB%8Bnh%20l%C3%A0%20m%E1%BB%99t%20k%E1%BB%B9%20thu%E1%BA%ADt%20machine,quan%20tr%E1%BB%8Dng%20nh%E1%BA%A5t%20%C4%91%E1%BB%83%20d%E1%BB%B1%20%C4%91o%C3%A1n%20k%E1%BA%BFt%20qu%E1%BA%A3.)
3. [Hướng dẫn nhanh để hiểu thuật toán KNN](https://www.unite.ai/vi/h%C6%B0%E1%BB%9Bng-d%E1%BA%ABn-nhanh-v%E1%BB%81-thu%E1%BA%ADt-to%C3%A1n-knn/#:~:text=Thu%E1%BA%ADt%20to%C3%A1n%20KNN%20l%C3%A0%20g%C3%AC%3F%20K-Thu%E1%BA%ADt%20to%C3%A1n%20h%C3%A0ng,n%E1%BA%B1m%20g%E1%BA%A7n%20nhau%20%C4%91%E1%BB%81u%20thu%E1%BB%99c%20c%C3%B9ng%20m%E1%BB%99t%20l%E1%BB%9Bp.)