A black and white rectangular frame

Description automatically generated**TRƯỜNG ĐẠI HỌC KINH TẾ - KỸ THUẬT CÔNG NGHIỆP**

-----🙞🙜🕮🙞🙜-----

A blue and white logo

Description automatically generated

**THỰC TẬP LẬP TRÌNH**

**Đề tài**

**Phân tích và so sánh SVM, Logistic Regresion, KNN và Naive Bayes**

**Giảng viên hướng dẫn: Nguyễn Anh Thư**

**Sinh viên thực hiện: Khuất Thanh Phương**

**Lê Tuấn Thành**

**Lớp: DHKL16A2HN**

**Khoa: Khoa học ứng dụng**

**HÀ NỘI - 2025**

**MỤC LỤC**

[Lời mở đầu 4](#_Toc198155991)

[PHẦN 1. SVM VÀ CÁC MÔ HÌNH TƯƠNG TỰ. 5](#_Toc198155992)

[CHƯƠNG 1. Lịch sử và Phát triển của SVM 5](#_Toc198155993)

[I . Lịch sử ra đời của SVM 5](#_Toc198155994)

[II . Những bước tiến quan trọng trong sự phát triển của SVM 5](#_Toc198155995)

[CHƯƠNG 2. Khái niệm về SVM (Support vector machine) và nguyên lý hoạt động của SVM 6](#_Toc198155996)

[I . Khái niệm về SVM (Support Vector Machine) 6](#_Toc198155997)

[II . Nguyên lí hoạt động của SVM. 6](#_Toc198155998)

[III . Đặc điểm của SVM và cấu trúc của SVM. 7](#_Toc198155999)

[IV . Hàm mục tiêu và hàm mất mát trong SVM. 10](#_Toc198156000)

[V . Hard – margin và Soft – margin trong SVM. 11](#_Toc198156001)

[VI . KENEL TRICK VÀ ỨNG DỤNG TRONG SVM. 14](#_Toc198156002)

[VII . Ưu và Nhược điểm trong thuật toán SVM. 17](#_Toc198156003)

[VIII . Ứng dụng thực tế của thuật toán Support Vector Machine (SVM). 20](#_Toc198156004)

[CHƯƠNG 3. Logistic regression 22](#_Toc198156005)

[I . Logistic Regression là gì? 22](#_Toc198156006)

[II . Cách hoạt động của Logistic Regression. 23](#_Toc198156007)

[III . Tối ưu hóa trọng số trong học máy. 25](#_Toc198156008)

[IV . Newton’s Method. 27](#_Toc198156009)

[CHƯƠNG 4. K – Nearest neighbors (KNN). 27](#_Toc198156010)

[I . Khái niệm cơ bản về KNN. 27](#_Toc198156011)

[II . Cách thức hoạt động của KNN. 28](#_Toc198156012)

[III . Đặc điểm của KNN. 29](#_Toc198156013)

[IV . Ưu điểm của KNN. 29](#_Toc198156014)

[V . Nhược điểm của KNN. 30](#_Toc198156015)

[VI . Ứng dụng của KNN. 30](#_Toc198156016)

[CHƯƠNG 5. Naïve Bayes. 31](#_Toc198156017)

[I . Khái niệm Naïve Bayes 31](#_Toc198156018)

[II . Định lí Bayes. 31](#_Toc198156019)

[III . Giả định độc lập (Naïve Assumption). 32](#_Toc198156020)

[IV . Cách thức hoạt động của Naïve Bayes. 32](#_Toc198156021)

[V . Các loại Naïve Bayes. 32](#_Toc198156022)

[VI . Ưu và nhược điểm của Naïve Bayes. 33](#_Toc198156023)

[VII . Ứng dụng của Naïve Bayes. 33](#_Toc198156024)

[PHẦN 2. thực nghiệm 35](#_Toc198156025)

[PHẦN 3. tổng kết và kết luận 50](#_Toc198156026)

[I . So sánh và đánh giá. 50](#_Toc198156027)

[II . Kết luận. 52](#_Toc198156028)

# Lời mở đầu

Trong kỷ nguyên của cách mạng công nghiệp 4.0, dữ liệu đã trở thành một tài nguyên vô giá, đóng vai trò trung tâm trong quá trình ra quyết định, tự động hóa và tối ưu hóa hệ thống trong nhiều lĩnh vực như y tế, tài chính, marketing, sản xuất và trí tuệ nhân tạo. Để khai thác hiệu quả nguồn dữ liệu ngày càng phong phú và phức tạp, các kỹ thuật học máy (machine learning) đã được phát triển mạnh mẽ, trong đó các mô hình phân loại là một trong những thành phần quan trọng nhất.

Bài báo cáo này tập trung tìm hiểu một số mô hình phân loại nền tảng trong học máy, bao gồm Support Vector Machine (SVM), Logistic Regression, K-Nearest Neighbors (KNN) và Naïve Bayes. Đây là những mô hình được sử dụng phổ biến không chỉ vì tính hiệu quả trong nhiều bài toán thực tiễn mà còn vì sự đơn giản trong cấu trúc và khả năng diễn giải tốt. Mỗi mô hình mang trong mình những đặc điểm riêng biệt về cách xây dựng, giả định, ưu nhược điểm cũng như phạm vi ứng dụng.

Thông qua việc phân tích nguyên lý hoạt động, cơ chế huấn luyện và khả năng dự đoán của từng mô hình, bài báo cáo nhằm mục tiêu giúp người đọc có được cái nhìn tổng quan và hệ thống về các thuật toán phân loại cơ bản. Ngoài ra, chúng tôi cũng sẽ so sánh các mô hình này trên một số tiêu chí quan trọng như độ chính xác, khả năng mở rộng, tốc độ huấn luyện và khả năng xử lý dữ liệu nhiễu. Việc nắm vững những kiến thức này sẽ là tiền đề quan trọng để lựa chọn thuật toán phù hợp với từng loại dữ liệu và bài toán cụ thể, đồng thời mở rộng sang các phương pháp học máy nâng cao hơn trong tương lai.

Qua báo cáo này, chúng tôi mong muốn cung cấp một nền tảng lý thuyết vững chắc cho việc lựa chọn và ứng dụng các thuật toán phân loại một cách hiệu quả trong thực tế.

# SVM VÀ CÁC MÔ HÌNH TƯƠNG TỰ.

## Lịch sử và Phát triển của SVM

### Lịch sử ra đời của SVM

Bắt đầu từ những năm 1960, với nền tảng lý thuyết đầu tiên được đặt ra bởi Vladimir Vapnik và Alexey Chervonenkis. Ban đầu, SVM được phát triển như một phương pháp trong lý thuyết học thống kê, với mục tiêu tạo ra một cách tiếp cận mới trong việc phân loại và dự đoán.

### Những bước tiến quan trọng trong sự phát triển của SVM

#### Những năm đầu (1960s – 1980s)

Trong giai đoạn này, Vapnik và Chervonenkis đã xây dựng nên những nguyên tắc cơ bản của SVM, bao gồm khái niệm về “margin” và “hyperplane” trong không gian đặc trưng. Tuy nhiên, SVM chưa được chú ý nhiều trong cộng đồng học máy.

#### Sự phát triển và nhận diện (1990s)

Thập kỷ 90 chứng kiến sự bùng nổ của SVM trong lĩnh vực học máy. Với việc giới thiệu các kỹ thuật như Kernel Trick, SVM trở nên mạnh mẽ và linh hoạt hơn, có khả năng xử lý dữ liệu phi tuyến tính.

#### Ứng dụng rông rãi (2000s – Hiện nay)

SVM đã trở thành một trong những thuật toán phổ biến và được ưa chuộng trong nhiều lĩnh vực ứng dụng khác nhau. Sự linh hoạt và hiệu suất cao của nó trong việc giải quyết các bài toán phân loại và hồi quy đã làm nổi bật SVM so với các thuật toán khác.

#### Sự phát triển của SVM không chỉ góp phần vào lĩnh vực Machine Learning

Sự phát triển của SVM không chỉ góp phần vào lĩnh vực Machine Learning mà còn ảnh hưởng đến các ngành khoa học khác như thống kê, tối ưu hóa và logic. Với sự phát triển của công nghệ và sự gia tăng về khả năng tính toán, SVM tiếp tục phát triển và thích ứng, chứng tỏ sức mạnh và độ linh hoạt của nó trong một loạt các ứng dụng thực tế.

## Khái niệm về SVM (Support vector machine) và nguyên lý hoạt động của SVM

### Khái niệm về SVM (Support Vector Machine)

SVM (Support Vector Machine) là một thuật toán học máy giám sát dùng để phân loại và hồi quy. Tuy nhiên, SVM chủ yếu được sử dụng cho bài toán phân loại, đặc biệt là phân loại nhị phân (hai lớp). Mục tiêu của SVM là tìm ra một siêu phẳng (hyperplane) tối ưu, giúp phân chia các lớp dữ liệu sao cho khoảng cách giữa các điểm gần nhất của mỗi lớp và siêu phẳng là lớn nhất.

### Nguyên lí hoạt động của SVM.

#### Siêu phẳng tối ưu.

Trong không gian hai chiều, SVM tìm một đường thẳng (siêu phẳng) để phân chia hai lớp dữ liệu sao cho khoảng cách giữa hai lớp (margins) là lớn nhất. Siêu phẳng này được gọi là siêu phẳng phân cách.

Trong không gian nhiều chiều, siêu phẳng sẽ trở thành một không gian (hyperplane) chia không gian thành hai phần.

#### Các điểm hỗ trợ (Support Vectors).

Các điểm dữ liệu nằm gần siêu phẳng phân cách và có ảnh hưởng quyết định đến vị trí của siêu phẳng được gọi là support vectors. Chúng là những điểm quan trọng nhất trong quá trình huấn luyện SVM, vì chúng xác định ranh giới giữa các lớp.

#### Tối ưu hóa.

SVM sẽ tìm cách tối đa hóa margins, tức là khoảng cách từ các điểm dữ liệu của mỗi lớp đến siêu phẳng phân cách. Khoảng cách này càng lớn thì mô hình càng có khả năng phân loại chính xác các dữ liệu mới.

SVM sử dụng các phương pháp tối ưu hóa (như phương pháp Lagrange) để tìm ra siêu phẳng với margin lớn nhất.

#### Linh hoạt với dữ liệu không tuyến tính (kernel trick).

Trong trường hợp dữ liệu không thể phân chia một cách tuyến tính, SVM có thể sử dụng một kỹ thuật gọi là kernel trick để chuyển đổi dữ liệu vào không gian có chiều cao hơn, nơi dữ liệu trở nên phân chia được bằng một siêu phẳng. Các kernel phổ biến bao gồm:

* Kernel tuyến tính: dùng khi dữ liệu có thể phân chia bằng một đường thẳng trong không gian gốc.
* Kernel Gauss (RBF): dùng khi dữ liệu không thể phân chia tuyến tính.

#### Cải tiến với tham số C.

C là tham số điều chỉnh trong SVM, xác định độ rộng của margin và sự chấp nhận các sai số. Nếu C quá lớn, mô hình sẽ cố gắng phân loại chính xác tất cả các điểm dữ liệu (rất dễ bị overfitting). Nếu C quá nhỏ, mô hình sẽ cho phép các điểm bị phân loại sai nhiều hơn (dễ bị underfitting).

### Đặc điểm của SVM và cấu trúc của SVM.

#### Đặc điểm của SVM.

##### Phân loại nhị phân (binary classification).

SVM chủ yếu được sử dụng để phân loại dữ liệu thành hai lớp (như phân loại email spam hay không spam). Mặc dù SVM cũng có thể được mở rộng để phân loại nhiều lớp (multi-class), nhưng phân loại nhị phân là ứng dụng chính.

##### Tìm siêu phẳng phân cách.

Mục tiêu chính của SVM là tìm một siêu phẳng (hyperplane) phân chia hai lớp dữ liệu sao cho khoảng cách giữa các điểm dữ liệu của mỗi lớp và siêu phẳng là lớn nhất, từ đó tăng khả năng phân loại chính xác của mô hình.

##### Hỗ trợ cho dữ liệu không tuyến tính.

Khi dữ liệu không thể phân chia bằng một siêu phẳng trong không gian 2D hay 3D, SVM có thể áp dụng kernel trick để đưa dữ liệu vào không gian có chiều cao hơn, nơi mà việc phân chia có thể thực hiện được.

#### Đặc điểm cứng (hard margin) và mềm (soft margin).

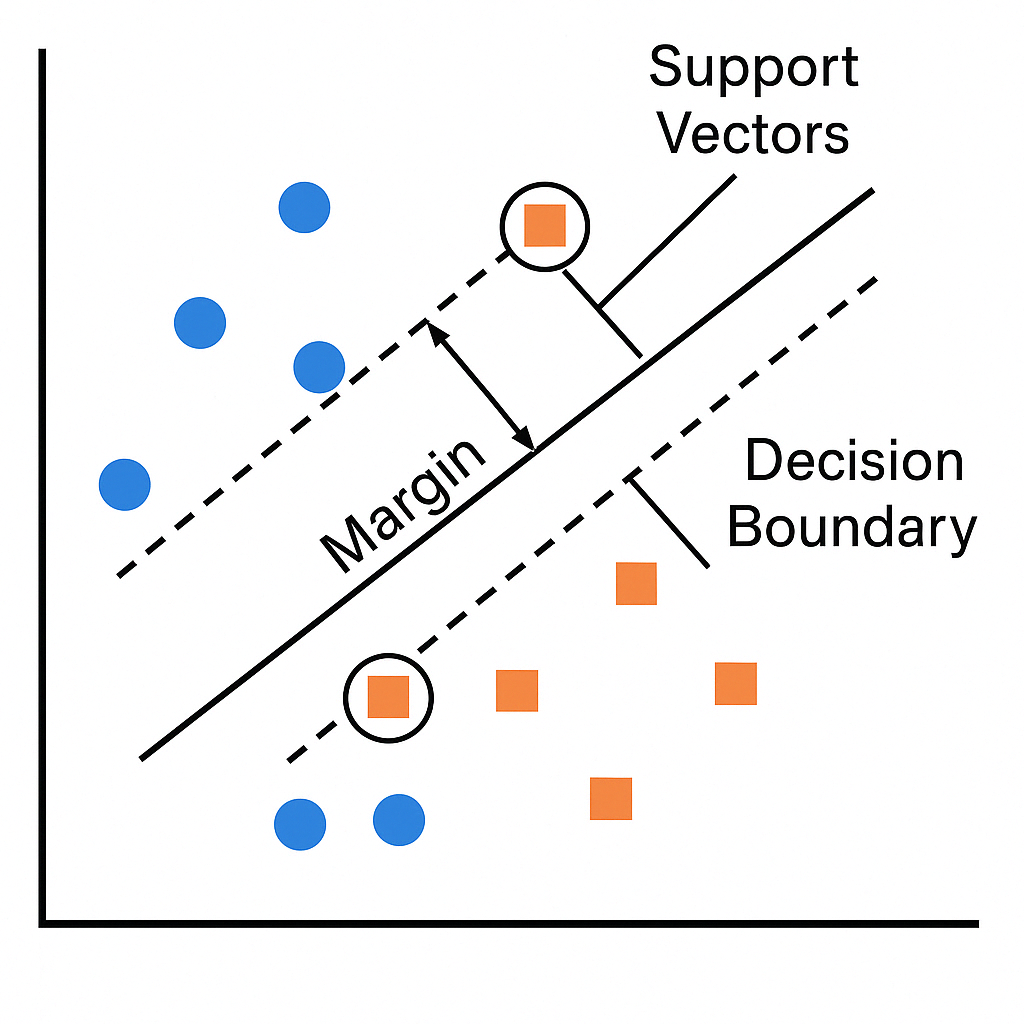
**Hard margin:** SVM cố gắng tìm một siêu phẳng mà không có điểm dữ liệu nào nằm ở phía sai của siêu phẳng. Điều này chỉ có thể thực hiện được khi dữ liệu là hoàn hảo và có thể phân chia tuyến tính.

**Soft margin:** Trong trường hợp dữ liệu có nhiễu hoặc không thể phân chia hoàn toàn tuyến tính, SVM cho phép một số điểm dữ liệu bị phân loại sai để làm cho mô hình có tính tổng quát tốt hơn. Điều này được điều chỉnh bởi tham số **C**.

**Support vectors:** Các điểm dữ liệu quan trọng nhất trong việc xác định siêu phẳng phân cách. Support vectors là các điểm nằm gần siêu phẳng và ảnh hưởng trực tiếp đến việc tối ưu hóa siêu phẳng.

**Tối ưu hóa margin:** SVM tìm cách tối ưu hóa margin, tức là khoảng cách từ các điểm dữ liệu của lớp này đến siêu phẳng phân cách. Một margin lớn hơn có nghĩa là mô hình có khả năng phân loại chính xác hơn

#### Cấu trúc của SVM.

*****Hình 1: Hình ảnh minh họa về cấu trúc của SVM trong 2D*

SVM có thể được mô tả bằng một số thành phần chính trong cấu trúc của nó:

**Siêu phẳng phân cách (Hyperplane).**

* Là một không gian trong không gian đặc trưng (feature space) mà tại đó dữ liệu được phân chia thành hai lớp. Trong không gian hai chiều, siêu phẳng này là một đường thẳng. Trong không gian ba chiều, siêu phẳng là một mặt phẳng, và trong không gian cao chiều, nó là một hyperplane.
* Siêu phẳng này được chọn sao cho khoảng cách giữa siêu phẳng và các điểm dữ liệu gần nhất của mỗi lớp là lớn nhất.

**Margin (Khoảng cách giữa các lớp).**

* Là khoảng cách từ các điểm dữ liệu của mỗi lớp đến siêu phẳng phân cách. SVM sẽ tìm cách tối đa hóa margin này. Khoảng cách này càng lớn, mô hình càng ít có khả năng bị overfitting.

**Các điểm hỗ trợ (Support Vectors).**

* Là các điểm dữ liệu gần siêu phẳng nhất. Những điểm này ảnh hưởng đến việc xác định siêu phẳng phân cách. Nếu các điểm này được thay đổi, siêu phẳng cũng sẽ thay đổi theo. SVM chỉ quan tâm đến những điểm này trong việc huấn luyện.

**Kernel Functions.**

* Khi dữ liệu không thể phân chia một cách tuyến tính, kernel trick sẽ được áp dụng. Kernel function là một phép biến đổi dữ liệu vào không gian chiều cao hơn để có thể phân chia tuyến tính được. Một số loại kernel thường gặp là.
* **Linear Kernel:** Dùng cho dữ liệu có thể phân chia tuyến tính.
* **Polynomial Kernel:** Dùng cho dữ liệu có sự phân chia phức tạp hơn.
* **Radial Basis Function (RBF) Kernel:** Dùng cho dữ liệu không tuyến tính.

### Hàm mục tiêu và hàm mất mát trong SVM.

#### Hàm mục tiêu (Objective Function) trong SVM.

Hàm mục tiêu trong SVM là hàm mà chúng ta muốn tối ưu hóa trong quá trình huấn luyện mô hình. Mục tiêu của SVM là tìm siêu phẳng phân cách sao cho khoảng cách giữa các điểm dữ liệu gần nhất của hai lớp là lớn nhất (hay nói cách khác là tối đa hóa margin). Điều này giúp mô hình phân loại chính xác hơn và giảm nguy cơ overfitting.

Trong SVM, bài toán tối ưu hóa được mô tả như sau:

* **Hàm mục tiêu chính** là tối đa hóa margin, nhưng điều này không đủ khi dữ liệu có nhiễu hoặc không thể phân chia hoàn hảo.
* Để biểu diễn hàm mục tiêu này, ta sử dụng các tham số w (vector trọng số) và b (hệ số dịch chuyển):

Maximize (Tối đa hóa margin )

Tuy nhiên, thay vì tối đa hóa trực tiếp , ta thường chuyển sang bài toán tối thiểu hóa (bởi vì việc tối thiểu hóa một hàm bậc hai dễ dàng hơn). Do đó, hàm mục tiêu sẽ trở thành:

minimize

#### Hàm mất mát (Loss Function) trong SVM.

Hàm mất mát trong SVM phản ánh sự sai lệch giữa dự đoán và thực tế của mô hình. SVM sử dụng hàm mất mát loại hinge loss, đặc biệt cho bài toán phân loại nhị phân.

Trong SVM, hàm mất mát được thiết kế để tính toán mức độ sai lệch khi một điểm dữ liệu bị phân loại sai hoặc nằm gần siêu phẳng phân cách.

* Giả sử chúng ta có một điểm dữ liệu thuộc lớp
* **W ⋅ :** là giá trị mà mô hình đưa ra cho điểm dữ liệu

Hàm mất mát loại hinge được định nghĩa như sau:

L() = max (0,1 - ⋅ (W ⋅ ))

**Trong đó:**

- là nhãn thực tế của điểm dữ liệu (1 hoặc -1).

**-= W ⋅** : là giá trị dự đoán của mô hình cho điểm dữ liệu

Nếu điểm dữ liệu được phân loại chính xác và khoảng cách của nó đến siêu phẳng phân cách đủ lớn, hàm mất mát sẽ là 0. Ngược lại, nếu điểm dữ liệu bị phân loại sai hoặc gần siêu phẳng, hàm mất mát sẽ lớn hơn 0 và tăng dần khi điểm dữ liệu càng gần siêu phẳng hoặc bị phân loại sai.

### Hard – margin và Soft – margin trong SVM.

#### Hard margin trong SVM.

##### Khái niệm.

Hard-margin trong SVM là thuật ngữ dùng để chỉ một dạng phân loại trong đó yêu cầu dữ liệu có thể phân chia hoàn hảo mà không có bất kỳ điểm dữ liệu nào bị sai phân loại (tức là không có lỗi). Đây là một khái niệm cơ bản của SVM, dùng để tìm kiếm siêu phẳng tối ưu phân chia hai lớp dữ liệu.

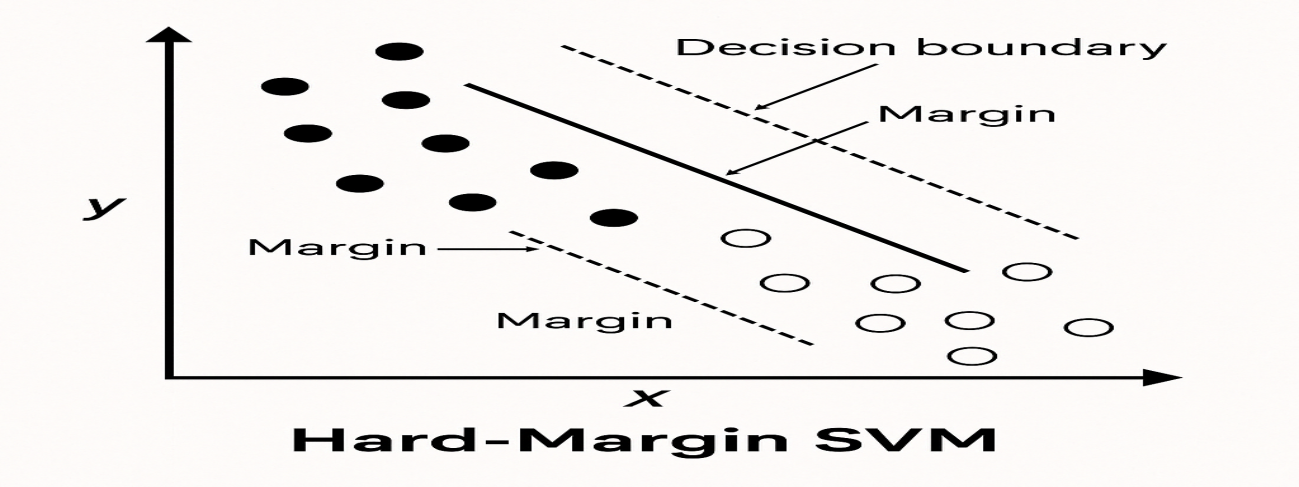
##### Điều kiện áp dụng.

Dữ liệu phải hoàn toàn phân tách được, tức là không có điểm dữ liệu nào nằm giữa các lớp hoặc bị sai phân.

Các điểm dữ liệu phải có thể được chia ra bởi một hyperplane mà không có bất kỳ sai sót nào.

**Ví dụ:** Trong bài toán phân loại hình ảnh hoặc văn bản, nếu mỗi lớp dữ liệu hoàn toàn tách biệt, chúng ta có thể sử dụng Hard-margin SVM.

Không có điểm dữ liệu bị sai phân loại: Mô hình Hard-margin yêu cầu rằng tất cả các điểm dữ liệu đều phải được phân loại chính xác và không có điểm nào nằm trong khu vực "lỗi" của siêu phẳng.



*Hình 2 : Hình ảnh minh họa về Hard-margin SVM*

##### Công thức tối ưu hóa.

Trong Hard-margin SVM, mục tiêu là tối đa hóa khoảng cách giữa hai lớp, tức là tìm một hyperplane sao cho khoảng cách từ hyperplane đến các điểm dữ liệu gần nhất (các vector hỗ trợ) là lớn nhất. Công thức tối ưu hóa là:

* Mục tiêu: Tối đa hóa khoảng cách giữa các lớp, hay là tối đa hóa độ rộng của "margin".
* Công thức: Tìm một hyperplane với phương trình w⋅x+b=0 sao cho khoảng cách từ hyperplane đến các điểm dữ liệu gần nhất là
* **Phương trình tối ưu hóa là:**

Với điều kiện rằng tất cả các điểm dữ liệu phải nằm ở một phía của hyperplane và cách nó ít nhất một khoảng cách nhất định (tức là không có điểm nào sai phân).

Điều kiện này có thể được viết dưới dạng:

Yi​(w⋅xi​+b) ≥ 1∀i

trong đó:

* yi là nhãn lớp của điểm dữ liệu xi, với **∈{+1,−1}**

##### Hậu quả.

**Ưu điểm:** Đảm bảo phân tách rõ ràng giữa hai lớp.

**Nhược điểm:** Phụ thuộc vào điều kiện dữ liệu. Nếu dữ liệu không phân tách hoàn toàn được, Hard-margin sẽ không hoạt động được và dẫn đến lỗi phân loại.

#### Soft – margin trong SVM.

##### Khái niệm.

Soft-margin SVM là phiên bản mở rộng của Hard-margin, được thiết kế để xử lý trường hợp dữ liệu không phân tách hoàn toàn (dữ liệu có thể bị nhiễu hoặc chồng lấn). Thay vì yêu cầu phân tách hoàn toàn, Soft-margin cho phép có một số điểm dữ liệu bị sai phân.

##### Điều kiện áp dụng.

Dữ liệu có thể không phân tách hoàn toàn (tức là có thể có một số điểm nằm sai phía của hyperplane).

Có một tham số điều chỉnh C để kiểm soát mức độ phạt đối với các điểm sai phân.

##### Công thức tối ưu hóa.

Mục tiêu của Soft-margin SVM là tối ưu hóa một hàm mục tiêu bao gồm hai yếu tố:

* Tối đa hóa margin (khoảng cách giữa các lớp).
* Phạt các điểm dữ liệu bị sai phân (tức là những điểm nằm ở phía sai của hyperplane).
* Công thức tối ưu hóa của Soft-margin SVM là:

**Trong đó:**

* là độ lệch (slack variable) của điểm dữ liệu thứ i, đại diện cho sự sai phân (nếu có) của điểm này so với hyperplane.
* C là tham số điều chỉnh mức độ phạt đối với các điểm sai phân. Giá trị của Ccàng lớn thì mức phạt đối với các điểm sai phân càng cao.

**Điều kiện ràng buộc cho Soft-margin là:**

​(w⋅+b)≥1−ξi ​∀i

và ξi≥0, với ξi là độ lệch của điểm dữ liệu xi

##### Hậu quả.

**Ưu điểm:** Có thể xử lý dữ liệu không phân tách được hoặc dữ liệu nhiễu. Cung cấp sự linh hoạt khi dữ liệu không hoàn toàn phân tách.

**Nhược điểm:** Việc điều chỉnh tham số CCC là quan trọng. Nếu CCC quá lớn, mô hình có thể quá khớp (overfitting) với dữ liệu huấn luyện. Nếu CCC quá nhỏ, mô hình có thể không học được cấu trúc dữ liệu đúng.

### KENEL TRICK VÀ ỨNG DỤNG TRONG SVM.

#### Kenel Trick là gì?

Kernel trick là một kỹ thuật trong học máy giúp biến đổi không gian đặc trưng của dữ liệu từ không gian đầu vào (input space) vào một không gian có chiều cao hơn, sao cho vấn đề phân loại phi tuyến có thể trở thành một bài toán tuyến tính trong không gian mới này.

Thay vì tính toán trực tiếp trong không gian đặc trưng cao hơn (thường là không gian vô cùng lớn hoặc vô hạn), ta sử dụng một hàm kernel để tính toán sản phẩm vô hướng (dot product) giữa các điểm dữ liệu trong không gian đặc trưng mà không cần phải thực hiện phép ánh xạ trực tiếp. Điều này tiết kiệm rất nhiều tài nguyên tính toán và vẫn mang lại kết quả chính xác.

#### Phương pháp không gian đặc trưng.

Giả sử chúng ta có một bài toán phân loại dữ liệu phi tuyến, trong đó các điểm dữ liệu không thể phân tách rõ ràng bởi một hyperplane trong không gian ban đầu (input space). SVM tìm kiếm một hyperplane phân tách các lớp trong không gian đặc trưng (feature space) sao cho khoảng cách giữa các điểm của các lớp là lớn nhất.

Không gian đặc trưng (Feature space) là không gian cao hơn mà các điểm dữ liệu được ánh xạ vào đó.

Kernel là hàm ánh xạ dữ liệu vào không gian đặc trưng cao hơn mà không cần phải tính toán trực tiếp sự chuyển đổi.

**Ví dụ:** Nếu không gian đầu vào có dữ liệu 2D (x1, x2), ta có thể sử dụng kernel để ánh xạ dữ liệu này vào một không gian ba chiều hoặc thậm chí cao hơn, nơi các điểm dữ liệu có thể phân tách tuyến tính dễ dàng hơn.

#### Cách thức hoạt động của Kernel Trick.

Trong SVM, mục tiêu là tìm ra một hyperplane phân tách các lớp trong không gian đặc trưng. Tuy nhiên, thay vì ánh xạ trực tiếp dữ liệu vào không gian đặc trưng, ta sử dụng một hàm kernel K(x,x′) để tính toán sản phẩm vô hướng giữa các điểm trong không gian đặc trưng mà không phải ánh xạ dữ liệu vào đó.

Sản phẩm vô hướng trong không gian đặc trưng: Khi chúng ta tính toán sản phẩm vô hướng ⟨ϕ(x),ϕ(x′)⟩ nơi ϕ(x) là ánh xạ của điểm x vào không gian đặc trưng, ta không cần phải biết chi tiết cách ánh xạ mà chỉ cần tính K(x,x′). Kernel trick giúp tính toán K(x,x′) mà không cần phải tính toán trực tiếp ϕ(x) và ϕ(x′).

#### Các loại Kernel phổ biến.

##### Linear Kernel (Kernel tuyến tính).

Kernel tuyến tính là kernel đơn giản nhất và thực tế không thực hiện bất kỳ chuyển đổi nào vào không gian đặc trưng (không gian mới). Nó chỉ tính sản phẩm vô hướng giữa hai điểm trong không gian đầu vào:

K(x,x′)=x⋅x′

Kernel này thường được sử dụng khi dữ liệu có thể phân tách được bằng một hyperplane trong không gian đầu vào.

##### Polynomial Kernel (Kernel đa thức).

Kernel đa thức ánh xạ dữ liệu vào không gian có các thành phần bậc cao hơn:

Trong đó ccc là một hằng số và d là bậc của đa thức.

Kernel này phù hợp với các bài toán mà các mối quan hệ giữa các điểm dữ liệu có thể được mô tả bằng các đa thức.

##### Gaussian Radial Basic Function (RBF) Kernel (Kernel hàm cơ sở bán kính).

Đây là kernel phổ biến nhất trong các ứng dụng thực tế. Nó ánh xạ dữ liệu vào không gian vô cùng cao (hoặc vô hạn) và có khả năng xử lý tốt các bài toán phân loại phi tuyến:

trong đó σ là tham số điều chỉnh, xác định mức độ "làm mềm" của kernel.

Kernel RBF rất mạnh mẽ trong các tình huống mà dữ liệu có thể không phân tách rõ ràng trong không gian ban đầu.

##### Sigmoid Kernel (Kernel Sigmoid).

Kernel này giống như hàm kích hoạt trong mạng nơ-ron nhân tạo, sử dụng hàm sigmoid để chuyển đổi dữ liệu:

K(x,x′)=tanh(αx⋅x′+c)

trong đó α và c là các tham số điều chỉnh.

Kernel sigmoid có thể được sử dụng cho các bài toán mà các mối quan hệ phi tuyến có thể được mô tả bằng hàm sigmoid.

#### Ứng dụng của Kernel Trick trong SVM.

Kernel trick giúp mở rộng khả năng của SVM khi xử lý các bài toán phân loại phi tuyến. Một số ứng dụng điển hình của Kernel trick trong SVM bao gồm:

Phân loại dữ liệu phi tuyến:

* Nếu dữ liệu không thể phân tách được trong không gian ban đầu, kernel trick cho phép SVM phân tách dữ liệu trong không gian đặc trưng có chiều cao hơn mà không cần phải tính toán các tọa độ trong không gian đó.

Phát hiện lỗi và phát hiện bất thường:

* Kernel SVM có thể được sử dụng để phát hiện các điểm dữ liệu bất thường trong các tập dữ liệu lớn, bằng cách chuyển đổi các điểm dữ liệu vào không gian đặc trưng và tìm kiếm những điểm khác biệt rõ rệt.

Ứng dụng trong hình ảnh và nhận dạng chữ viết:

* Kernel SVM có thể được áp dụng trong các bài toán nhận dạng chữ viết tay, nhận dạng đối tượng, và phân tích hình ảnh khi dữ liệu có sự phức tạp cao và không thể phân tách được bằng một đường thẳng.

Chẩn đoán y tế:

* Kernel SVM có thể được sử dụng trong các bài toán phân loại y tế, chẳng hạn như phân loại các bệnh hoặc chẩn đoán từ hình ảnh y tế, nơi các mối quan hệ giữa các yếu tố có thể là phi tuyến.

#### Lợi ích và hạn chế của Kernel Trick.

* Lợi ích:
  + Tính linh hoạt: Có thể áp dụng kernel cho nhiều loại bài toán khác nhau và xử lý dữ liệu phi tuyến một cách hiệu quả
  + Tiết kiệm tính toán: Kernel trick giúp tránh phải tính toán trực tiếp trong không gian đặc trưng cao, giảm bớt sự phức tạp tính toán.
  + Tính hiệu quả: Dù không gian đặc trưng có thể rất lớn (hoặc vô hạn), kernel trick cho phép SVM giải quyết bài toán mà không cần phải tính toán rõ ràng không gian đó.
* Hạn chế:
  + Lựa chọn kernel: Việc chọn kernel phù hợp là một yếu tố quan trọng, và có thể khó khăn nếu không có sự hiểu biết về dữ liệu.
  + Chi phí tính toán: Mặc dù kernel trick giúp giảm bớt chi phí tính toán trong không gian đặc trưng, nhưng tính toán sản phẩm vô hướng giữa tất cả các cặp điểm trong tập dữ liệu vẫn có thể tốn kém về mặt tài nguyên tính toán.

### Ưu và Nhược điểm trong thuật toán SVM.

#### Ưu điểm của thuật toán SVM.

**Hiệu quả trong không gian đặc trưng cao:**

* SVM là một trong những thuật toán học máy hiệu quả khi làm việc với dữ liệu có không gian đặc trưng cao hoặc khi số chiều của dữ liệu lớn (ví dụ, trong phân loại văn bản, nhận dạng hình ảnh, v.v.).
* SVM có thể sử dụng các hàm kernel để ánh xạ dữ liệu vào không gian đặc trưng cao hơn, nơi có thể phân tách các lớp dữ liệu phi tuyến.

**Tính chính xác cao (Khi lựa chọn kernel tốt):**

* Khi được tối ưu hóa và chọn lựa cẩn thận, SVM có thể đạt được độ chính xác cao trong các bài toán phân loại.
* Đặc biệt, SVM có thể cung cấp hiệu quả tốt khi số điểm dữ liệu ít nhưng dữ liệu có tính chất phức tạp (ví dụ: phân loại văn bản, nhận dạng hình ảnh).

**Tối ưu hóa tối đa biên (Maximum Margin):**

* SVM tìm kiếm hyperplane phân tách với biên (margin) lớn nhất giữa các lớp, giúp tăng khả năng tổng quát của mô hình.
* Điều này giúp SVM có khả năng phân loại tốt các dữ liệu chưa thấy (dữ liệu test), giảm thiểu khả năng overfitting.

**Hỗ trợ phân loại phi tuyến nhờ Kernel Trick:**

* SVM có khả năng xử lý các bài toán phân loại phi tuyến nhờ vào Kernel trick. Thông qua việc sử dụng các hàm kernel, SVM có thể chuyển các vấn đề phi tuyến thành các vấn đề tuyến tính trong không gian đặc trưng cao hơn mà không cần phải thực hiện phép chiếu vào không gian đó.

**Đặc biệt hiệu quả trong các bài toán phân loại hai lớp (Binary Classification):**

* SVM rất mạnh mẽ trong các bài toán phân loại hai lớp, đặc biệt khi dữ liệu là tuyến tính hoặc có thể phân tách được trong không gian đặc trưng.

**Khả năng làm việc tốt với các bộ dữ liệu nhỏ:**

* SVM có thể đạt được hiệu quả cao ngay cả khi số lượng dữ liệu huấn luyện tương đối nhỏ, điều này làm cho SVM là một lựa chọn phổ biến trong các ứng dụng thực tế có sẵn bộ dữ liệu huấn luyện không quá lớn.

#### Nhược điểm của thuật toán SVM.

**Chi phí tính toán cao.**

* SVM có thể yêu cầu tài nguyên tính toán lớn, đặc biệt khi số lượng điểm dữ liệu và số chiều của không gian đặc trưng rất lớn.
* Việc tính toán ma trận kernel cho tất cả các cặp điểm trong bộ dữ liệu có thể trở nên tốn thời gian và đòi hỏi bộ nhớ lớn, đặc biệt đối với các bộ dữ liệu rất lớn.

**Khó khăn trong việc lựa chọn kernel và các tham số:**

* Một trong những nhược điểm lớn của SVM là sự lựa chọn kernel phù hợp và điều chỉnh tham số CCC và γ\gammaγ (đối với kernel RBF).
* Việc lựa chọn một kernel phù hợp có thể ảnh hưởng rất lớn đến hiệu suất của mô hình. Việc tìm các tham số tốt (thông qua cross-validation hoặc grid search) có thể tốn thời gian và công sức.

**Không phù hợp với bộ dữ liệu rất lớn:**

* Mặc dù SVM có thể đạt hiệu suất rất cao với dữ liệu nhỏ, nhưng khi dữ liệu có kích thước rất lớn (ví dụ: hàng triệu điểm dữ liệu), thuật toán này có thể gặp khó khăn về mặt thời gian tính toán và bộ nhớ.
* SVM không phải là lựa chọn tối ưu trong những bài toán yêu cầu huấn luyện trên một lượng dữ liệu cực lớn.

**Độ nhạy với dữ liệu nhiễu (Noise):**

* Mặc dù SVM có khả năng phân tách rõ ràng các lớp, nhưng nó có thể bị ảnh hưởng mạnh mẽ bởi dữ liệu nhiễu, đặc biệt nếu như có điểm ngoại lai (outliers) hoặc dữ liệu không đồng nhất.
* Điều này có thể dẫn đến việc lựa chọn một hyperplane không tối ưu hoặc mô hình overfitting, làm giảm khả năng tổng quát của mô hình.

**Không dễ dàng cho bài toán phân loại đa lớp:**

* Mặc dù có thể sử dụng SVM trong các bài toán phân loại đa lớp thông qua các kỹ thuật như "one-vs-all" hoặc "one-vs-one", nhưng SVM bản thân nó chỉ hỗ trợ phân loại hai lớp (binary classification).
* Các phương pháp này có thể phức tạp và không phải lúc nào cũng hiệu quả trong các bài toán phân loại đa lớp có số lượng lớp lớn.

**Không trực quan và dễ hiểu:**

* SVM là một mô hình "hộp đen" (black-box), nghĩa là khó giải thích và trực quan hóa được kết quả mô hình, đặc biệt khi sử dụng các kernel phức tạp.
* Điều này làm cho SVM ít được ưa chuộng trong các ứng dụng yêu cầu giải thích rõ ràng về cách thức mô hình đưa ra quyết định.

### Ứng dụng thực tế của thuật toán Support Vector Machine (SVM).

#### Nhận dạng hình ảnh và phân loại đối tượng,

SVM được sử dụng rộng rãi trong các bài toán nhận dạng hình ảnh và phân loại đối tượng, nhờ vào khả năng xử lý tốt các bài toán phân loại phi tuyến và hiệu quả với không gian đặc trưng cao.

**Nhận dạng chữ viết tay:** SVM thường được sử dụng trong các hệ thống nhận dạng chữ viết tay như **MNIST**, nơi SVM có thể phân loại các chữ số viết tay với độ chính xác cao.

**Nhận diện khuôn mặt:** SVM được sử dụng để nhận diện khuôn mặt trong các hệ thống bảo mật, phân biệt các đặc điểm của khuôn mặt người dùng trong một cơ sở dữ liệu.

**Phân loại đối tượng trong ảnh:** Trong các bài toán phân loại các loại đối tượng trong hình ảnh (ví dụ, phân biệt giữa các loại xe cộ hoặc các loại cây trong ảnh), SVM có thể được sử dụng để phân loại các đối tượng vào các nhóm khác nhau.

#### Phân loại văn bản và phân tích cảm xúc.

SVM cũng rất mạnh mẽ trong các bài toán phân loại văn bản, đặc biệt khi làm việc với các văn bản lớn và phức tạp.

**Phân loại văn bản:** SVM có thể được sử dụng để phân loại các tài liệu văn bản thành các thể loại khác nhau (ví dụ: phân loại email thành spam hoặc không spam, phân loại bài viết theo thể loại tin tức, v.v.).

**Phân tích cảm xúc (Sentiment Analysis):** SVM có thể giúp phân tích cảm xúc trong các bài đăng trên mạng xã hội hoặc các nhận xét sản phẩm, phân loại chúng thành các nhóm như "tiêu cực", "tích cực" và "trung lập".

#### Phân loại trong y tế và sinh học.

SVM đã được sử dụng rộng rãi trong lĩnh vực y tế và sinh học để phân tích các tập dữ liệu phức tạp, bao gồm dữ liệu gene, hình ảnh y tế, và các chỉ số sinh học khác.

**Chẩn đoán bệnh:** SVM có thể giúp chẩn đoán các bệnh như ung thư, bệnh tim, và các bệnh lý khác dựa trên dữ liệu y tế. Ví dụ, SVM có thể phân loại các bệnh ung thư vú (ví dụ, thông qua hình ảnh mô học) thành lành tính và ác tính.

**Phân tích gene:** SVM cũng được áp dụng trong phân tích gene để phân biệt các loại gene khác nhau, hoặc để tìm kiếm các đặc điểm gene liên quan đến một bệnh lý nhất định.

**Phân loại các mô hình sinh học:** SVM được sử dụng để phân tích các dữ liệu sinh học, chẳng hạn như phân loại các loại tế bào, mô, hoặc các protein.

#### Phát hiện gian lận (Fraud Detection).

SVM là một công cụ hiệu quả trong phát hiện gian lận trong các lĩnh vực tài chính và ngân hàng. Bằng cách sử dụng dữ liệu giao dịch, SVM có thể phân loại các giao dịch "bình thường" và "nghi ngờ" (gian lận).

**Phát hiện gian lận trong giao dịch thẻ tín dụng:** SVM có thể phân loại các giao dịch thẻ tín dụng thành hợp pháp và gian lận dựa trên các đặc điểm giao dịch như giá trị, thời gian, địa điểm, v.v.

**Phát hiện gian lận trong bảo hiểm:** SVM cũng có thể được sử dụng trong lĩnh vực bảo hiểm để phát hiện các yêu cầu bồi thường gian lận dựa trên dữ liệu yêu cầu và hồ sơ khách hàng.

#### Dự báo tài chính và chứng khoán.

SVM có thể được sử dụng trong các mô hình dự báo tài chính để dự đoán xu hướng của thị trường chứng khoán, giá trị tài sản, hoặc các yếu tố kinh tế khác.

**Dự đoán giá cổ phiếu:** SVM có thể được sử dụng để phân tích dữ liệu lịch sử và dự đoán xu hướng tăng hoặc giảm của giá cổ phiếu.

**Phân loại các loại chứng khoán:** SVM có thể phân loại các loại chứng khoán theo các nhóm rủi ro khác nhau, giúp nhà đầu tư lựa chọn các khoản đầu tư phù hợp.

#### Tự động hóa và điều khiển.

SVM cũng có thể được áp dụng trong các hệ thống tự động hóa và điều khiển, đặc biệt trong các tình huống yêu cầu phân loại quyết định nhanh chóng.

**Hệ thống điều khiển robot:** SVM có thể được sử dụng để điều khiển robot trong các môi trường không gian phức tạp, giúp phân loại các đối tượng trong môi trường và đưa ra các quyết định dựa trên phân loại đó.

**Phân loại tín hiệu trong tự động hóa:** SVM có thể phân loại các tín hiệu cảm biến trong các hệ thống tự động hóa (ví dụ, phân loại tín hiệu từ các cảm biến nhiệt độ, độ ẩm trong các nhà máy).

#### Xử lí tín hiệu và nhận dạng giọng nói.

SVM có thể được sử dụng trong các hệ thống nhận dạng giọng nói và xử lý tín hiệu để phân loại các tín hiệu âm thanh.

**Nhận dạng giọng nói:** SVM được sử dụng để nhận dạng các mẫu giọng nói khác nhau và phân biệt giữa các người nói, hoặc nhận diện các lệnh giọng nói trong các hệ thống như trợ lý ảo (ví dụ: Siri, Google Assistant).

**Nhận dạng âm thanh:** SVM có thể phân loại các loại âm thanh, chẳng hạn như phân biệt tiếng động trong môi trường (tiếng động giao thông, tiếng bước chân, v.v.).

#### Xử lí ngữ nghĩa và phân loại ngữ nghĩa.

Trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), SVM có thể được sử dụng để phân loại các câu hoặc văn bản dựa trên ngữ nghĩa hoặc mục đích của chúng.

**Phân loại câu theo mục đích:** SVM có thể phân loại câu trong các hệ thống đối thoại tự động (chatbot) hoặc các hệ thống dịch máy, phân biệt các câu hỏi, yêu cầu, hay phản hồi.

**Phân tích ngữ nghĩa:** SVM có thể giúp phân tích ngữ nghĩa của các văn bản và phân loại chúng theo các chủ đề khác nhau.

## Logistic regression

### Logistic Regression là gì?

#### Khái niệm về Logistic Regression.

Logistic Regression dùng để **dự đoán xác suất** của một đối tượng thuộc về một lớp (ví dụ: lớp "1" hoặc "0") trong bài toán phân loại nhị phân. Đặc biệt, trong bài toán phân loại nhị phân, mô hình Logistic Regression trả về một giá trị từ 0 đến 1, biểu diễn xác suất của đối tượng thuộc về một lớp nhất định.

#### Mục tiêu.

**Phân loại nhị phân:** Dự đoán nhãn lớp y cho các điểm dữ liệu x, với y∈{0,1}

**Đầu ra:** Xác suất của một điểm dữ liệu thuộc về lớp "1". Nếu xác suất lớn hơn một ngưỡng (thường là 0.5), điểm dữ liệu được phân loại vào lớp "1", ngược lại phân vào lớp "0".

### Cách hoạt động của Logistic Regression.

#### Hồi quy tuyến tính.

Trong hồi quy tuyến tính, mô hình sẽ tính toán một giá trị liên tục dựa trên hàm số của các biến độc lập ,...,:

z= + +.....+ +b

Trong đó:

,..., ​ là các đặc trưng (features) của dữ liệu.

,...,là các trọng số (weights) tương ứng với mỗi đặc trưng.

b là bias (hệ số dịch).

=> Kết quả z sẽ là một giá trị liên tục và có thể bất kỳ giá trị nào trên dải vô hạn. Tuy nhiên, trong bài toán phân loại nhị phân, chúng ta cần một giá trị từ 0 đến 1 (xác suất). Để đạt được điều này, chúng ta sử dụng hàm sigmoid.

#### Sử dụng hàm Sigmoid.

Hàm sigmoid (hay còn gọi là hàm logistic) là một hàm toán học được sử dụng phổ biến trong các mô hình học máy, đặc biệt là trong hồi quy logistic và mạng nơ-ron nhân tạo. Hàm này có tính chất quan trọng trong việc chuyển đổi giá trị đầu vào thành một giá trị trong khoảng từ 0 đến 1, rất phù hợp để sử dụng trong các bài toán phân loại, nơi kết quả đầu ra là xác suất.

Công thức :

σ(z)=

Trong đó:

**z** là đầu vào của hàm sigmoid, chính là kết quả của hồi quy tuyến tính z= + +.....+ +b

**e** là cơ số của logarit tự nhiên.

Hàm sigmoid có đặc điểm là biến giá trị đầu vào (z) thành một giá trị trong khoảng từ 0 đến 1. Khi đầu vào z tăng, giá trị đầu ra của hàm sigmoid càng gần 1, và ngược lại, khi z giảm, giá trị đầu ra càng gần 0.

Kết quả của hàm sigmoid có thể được hiểu là xác suất của điểm dữ liệu thuộc về lớp "1". Cụ thể:

P(y=1∣x) = σ(z) =

Khi giá trị xác suất này lớn hơn hoặc bằng 0.5, ta phân loại đối tượng vào lớp "1", nếu nhỏ hơn 0.5, ta phân loại vào lớp "0".

Ý nghĩa:

* **Xác suất phân loại:**

Trong bài toán **phân loại nhị phân**, đầu ra của mô hình (thường là kết quả của hồi quy logistic hoặc mạng nơ-ron) có thể không phải là một giá trị nhị phân (0 hoặc 1), mà là một giá trị thực. Hàm sigmoid sẽ chuyển đổi giá trị này thành một xác suất thuộc vào một trong hai lớp:

* Xác suất lớp "1" là σ(z)
* Xác suất lớp "0" là 1-ơ(z)

=> Điều này giúp mô hình đưa ra quyết định phân loại dựa trên ngưỡng xác suất. Thông thường, nếu xác suất của lớp "1" lớn hơn 0.5, mô hình sẽ phân loại đối tượng vào lớp "1", ngược lại, nếu xác suất nhỏ hơn 0.5, mô hình sẽ phân loại đối tượng vào lớp "0".

* **Giải thích mô hình phân loại:**
* Hàm sigmoid cho phép các mô hình phân loại đưa ra các xác suất thay vì chỉ đưa ra dự đoán "0" hoặc "1". Điều này mang lại khả năng **giải thích mô hình** rõ ràng hơn, vì ta có thể hiểu rằng một điểm dữ liệu có xác suất là 0.7 thuộc về lớp "1" (tức là mô hình tin tưởng 70% vào việc điểm này thuộc lớp "1").

* **Hỗ trợ tối ưu hóa (Gradient Descent):**
* Trong quá trình huấn luyện các mô hình học máy (như **hồi quy logistic** hay **mạng nơ-ron**), hàm sigmoid có vai trò quan trọng trong việc tối ưu hóa các trọng số của mô hình thông qua việc tính toán **gradient**. Đạo hàm của hàm sigmoid (tính theo trọng số) giúp trong việc tính toán gradient để cập nhật các trọng số sao cho mô hình giảm thiểu hàm mất mát.
* **Chuyển đổi giá trị thành xác suất:**
* Một trong những tính năng mạnh mẽ của hàm sigmoid là nó có thể chuyển đổi các giá trị thực thành xác suất. Điều này cực kỳ hữu ích trong các bài toán phân loại, đặc biệt là khi ta cần xác định khả năng mà một điểm dữ liệu thuộc về một lớp nào đó, thay vì chỉ dự đoán "nhãn lớp".

#### Hàm mất mát (Loss Function) và tối ưu hóa.

Để huấn luyện mô hình Logistic Regression, chúng ta cần tối ưu hóa các trọng số ....., sao cho mô hình có thể dự đoán chính xác nhất. Điều này được thực hiện thông qua việc tối ưu hóa một hàm mất mát (loss function), thường là Hàm mất mát Cross-Entropy (hay còn gọi là log loss).

Hàm mất mát Cross-Entropy cho một điểm dữ liệu là:

L(y,​) = - [y.log() + (1- y).log(1-)]

Trong đó:

y là nhãn thực tế của điểm dữ liệu (0 hoặc 1).

​ là xác suất mà mô hình dự đoán cho lớp "1", tức là giá trị đầu ra của hàm sigmoid.

Mục tiêu của Logistic Regression là tối thiểu hóa hàm mất mát này trên toàn bộ dữ liệu huấn luyện. Quá trình tối ưu này thường được thực hiện bằng **Gradient Descent** hoặc các phương pháp tối ưu hóa khác.

### Tối ưu hóa trọng số trong học máy.

#### Gradient Descent (GD).

##### Khái niệm.

**Gradient Descent** là một thuật toán tối ưu hóa phổ biến và dễ hiểu được sử dụng trong nhiều bài toán học máy, đặc biệt trong các mạng nơ-ron và hồi quy tuyến tính. Mục tiêu của Gradient Descent là tối thiểu hóa hàm mất mát (loss function) bằng cách điều chỉnh trọng số theo chiều của gradient (đạo hàm) của hàm mất mát đối với các trọng số.

##### Cách thức hoạt động.

**Bước 1** : Khởi tạo trọng số: Bắt đầu với các giá trị ngẫu nhiên cho trọng số ,..., và bias b.

**Bước 2 :** Tính toán gradient: Tính đạo hàm của hàm mất mát đối với các trọng số. Đạo hàm này sẽ cho biết hướng và độ lớn của độ dốc (sự thay đổi nhanh nhất) của hàm mất mát tại mỗi điểm.

**Bước 3 :** Cập nhật trọng số: Cập nhật các trọng số bằng cách di chuyển theo hướng ngược lại với gradient (hướng giảm dần giá trị hàm mất mát). Cập nhật này được thực hiện theo công thức:

trong đó:

​ là trọng số thứ i

là **learning rate** (tốc độ học), điều chỉnh độ lớn của bước nhảy.

là gradient của hàm mất mát LLL đối với trọng số

**Bước 4:** Lặp lại quá trình: Quá trình cập nhật trọng số tiếp tục cho đến khi hàm mất mát hội tụ, tức là không còn thay đổi lớn qua các bước lặp.

Các loại Gradient Descent.

Batch Gradient Descent: Tính toán gradient trên toàn bộ tập dữ liệu huấn luyện. Phương pháp này thường chậm và yêu cầu bộ nhớ lớn khi tập dữ liệu rất lớn.

Stochastic Gradient Descent (SGD): Tính toán gradient dựa trên từng điểm dữ liệu riêng lẻ. Phương pháp này nhanh hơn nhưng có độ dao động lớn trong quá trình học.

Mini-batch Gradient Descent: Kết hợp giữa Batch và Stochastic Gradient Descent, tính toán gradient dựa trên một nhóm con các điểm dữ liệu (mini-batch), giúp cân bằng giữa tốc độ và độ chính xác.

##### Ưu và nhược điểm của Gradient Descent.

###### Ưu điểm của Gradient Descent.

Dễ dàng triển khai và áp dụng cho các bài toán có số lượng tham số lớn.

Có thể sử dụng cho các mô hình phức tạp như mạng nơ-ron sâu.

###### Nhược điểm của Gradient Descent.

Độ hội tụ có thể chậm nếu không chọn đúng learning rate.

Có thể mắc phải vấn đề local minima hoặc saddle points, nơi hàm mất mát có thể không đạt giá trị tối thiểu toàn cục.

### Newton’s Method.

#### Khái niệm.

Newton's Method là một phương pháp tối ưu hóa sử dụng đạo hàm cấp hai của hàm mất mát để tìm cực tiểu. Phương pháp này nhanh chóng hội tụ so với các phương pháp như Gradient Descent vì nó sử dụng thông tin về độ cong của hàm để đưa ra các bước cập nhật chính xác hơn.

#### Cách thức hoạt động.

**\* Bước 1:** Tính gradient và Hessian:

* + Tính gradient (đạo hàm cấp nhất) của hàm mất mát
  + Tính ma trận Hessian, là ma trận đạo hàm cấp hai của hàm mất mát

H =

**\* Bước 2:** Cập nhật trọng số: Cập nhật trọng số theo công thức:

w:=w−∇ L

trong đó là ma trận nghịch đảo của Hessian.

#### Ưu và nhược điểm của Newton’s Method.

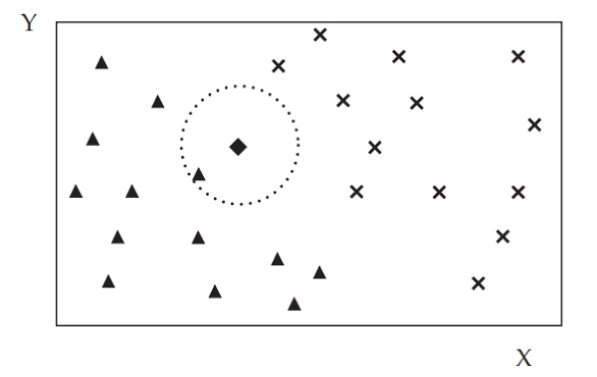
* Ưu điểm:
  + Hội tụ nhanh chóng: Phương pháp này thường hội tụ nhanh hơn Gradient Descent vì nó sử dụng cả gradient và thông tin về độ cong của hàm.
  + Chính xác cao hơn: Newton’s Method có khả năng tìm cực tiểu chính xác hơn.
* Nhược điểm:
  + Chi phí tính toán cao: Việc tính toán và nghịch đảo ma trận Hessian yêu cầu nhiều tài nguyên tính toán, đặc biệt đối với bài toán có số lượng tham số lớn.
  + Yêu cầu bộ nhớ lớn: Ma trận Hessian có thể rất lớn, yêu cầu bộ nhớ đáng kể khi xử lý các mô hình phức tạp.

## K – Nearest neighbors (KNN).

### Khái niệm cơ bản về KNN.

KNN là một thuật toán học có giám sát (supervised learning), nơi thuật toán học từ dữ liệu huấn luyện để phân loại hoặc dự đoán giá trị cho các điểm dữ liệu mới. Cơ bản của KNN là tìm kiếm K điểm dữ liệu gần nhất với điểm mới cần phân loại hoặc dự đoán giá trị, và sau đó quyết định nhãn hoặc giá trị dự đoán cho điểm dữ liệu đó dựa trên các điểm láng giềng gần nhất.

* Phân loại (Classification): Nếu bài toán là phân loại, KNN sẽ gán nhãn của điểm dữ liệu mới theo đa số nhãn của K điểm gần nhất trong không gian đặc trưng.
* Hồi quy (Regression): Nếu bài toán là hồi quy, giá trị dự đoán cho điểm dữ liệu mới sẽ là trung bình hoặc trung vị của các giá trị mục tiêu của K điểm gần nhất.



*Hình 3 : Hình ảnh minh họa thuật toán KNN (K-Nearest Neighbors)*

### Cách thức hoạt động của KNN.

KNN hoạt đông qua một số bước cơ bản sau:

* **Bước 1 :** Chọn số K: Người dùng cần xác định một số nguyên KKK, là số lượng láng giềng gần nhất mà thuật toán sẽ sử dụng để dự đoán nhãn hoặc giá trị cho điểm dữ liệu mới. Số K phải được chọn sao cho không quá nhỏ (dễ bị ảnh hưởng bởi nhiễu) và không quá lớn (có thể dẫn đến quá khớp, overfitting).
* **Bước 2 :** Tính khoảng cách: Để tìm K láng giềng gần nhất, thuật toán KNN tính toán khoảng cách giữa điểm dữ liệu mới và tất cả các điểm trong tập huấn luyện. Các thước đo khoảng cách phổ biến bao gồm:
* Khoảng cách Euclidean (thường được sử dụng trong các bài toán phân loại và hồi quy):

d(x,y) =

* Khoảng cách Manhattan: Tính tổng các giá trị tuyệt đối của sự chênh lệch giữa các đặc trưng.
* Khoảng cách Minkowski: Là một sự tổng quát của khoảng cách Euclidean và Manhattan.
* **Bước 3 :** Tìm K láng giềng gần nhất: Dựa trên khoảng cách tính được, thuật toán KNN tìm K điểm dữ liệu gần nhất với điểm cần phân loại hoặc dự đoán.
* **Bước 4 :** Phân loại hoặc dự đoán giá trị:
* Trong phân loại, thuật toán sẽ gán nhãn cho điểm dữ liệu mới bằng cách lấy nhãn của đa số K láng giềng gần nhất (kỹ thuật "đa số" hoặc majority voting).
* Trong hồi quy, giá trị của điểm dữ liệu mới sẽ là trung bình (hoặc trung vị) của các giá trị mục tiêu của K láng giềng gần nhất.
* **Bước 5 :** Gán nhãn hoặc giá trị: Kết quả phân loại hoặc giá trị dự đoán được gán cho điểm dữ liệu mới.

### Đặc điểm của KNN.

Không có mô hình (Instance-based): KNN là một thuật toán không có mô hình, tức là không học một hàm hay mô hình chính thức từ dữ liệu. Thay vào đó, nó lưu trữ tất cả các điểm huấn luyện và tính toán kết quả trực tiếp khi có dữ liệu mới.

Đơn giản và dễ hiểu: KNN rất đơn giản, dễ triển khai và hiểu. Đây là một trong những thuật toán cơ bản trong học máy, nhưng có thể hiệu quả nếu số lượng điểm dữ liệu không quá lớn.

Thuật toán có giám sát: KNN là thuật toán học giám sát, tức là yêu cầu dữ liệu huấn luyện có nhãn hoặc giá trị mục tiêu.

Không yêu cầu huấn luyện (Lazy Learning): KNN không có bước huấn luyện thực sự. Dữ liệu huấn luyện chỉ được sử dụng khi có yêu cầu phân loại hoặc dự đoán cho điểm dữ liệu mới. Điều này làm cho thuật toán trở nên "lười" trong việc huấn luyện nhưng có thể trở nên rất chậm khi dự đoán nếu dữ liệu huấn luyện rất lớn.

### Ưu điểm của KNN.

Đơn giản và dễ triển khai: KNN rất dễ hiểu và triển khai. Thuật toán này không yêu cầu quá nhiều bước phức tạp trong quá trình học máy.

Không cần huấn luyện: Không giống như các mô hình học máy khác, KNN không cần một quá trình huấn luyện riêng biệt. Bạn chỉ cần lưu trữ dữ liệu huấn luyện và tính toán khoảng cách khi cần thiết.

Hiệu quả với dữ liệu nhỏ: KNN hoạt động tốt khi bộ dữ liệu huấn luyện không quá lớn và không có quá nhiều đặc trưng.

Xử lý tốt với các bài toán phân loại phi tuyến: Vì KNN không phụ thuộc vào một mô hình tuyến tính, nó có thể hoạt động tốt với các bài toán phân loại phi tuyến.

### Nhược điểm của KNN.

Chi phí tính toán cao: Vì KNN cần tính toán khoảng cách giữa điểm dữ liệu mới và tất cả các điểm huấn luyện, việc này có thể trở nên tốn kém về mặt tính toán khi số lượng dữ liệu huấn luyện rất lớn.

Khả năng làm việc kém với dữ liệu lớn: Khi số lượng dữ liệu huấn luyện rất lớn, thuật toán có thể trở nên chậm và không hiệu quả, đặc biệt trong các ứng dụng yêu cầu tốc độ nhanh.

Nhạy cảm với nhiễu: KNN rất nhạy cảm với các điểm ngoại lai (outliers) trong dữ liệu, vì các điểm ngoại lai có thể ảnh hưởng đáng kể đến việc tính toán khoảng cách và kết quả phân loại.

Cần chọn K phù hợp: Chọn giá trị K quá nhỏ có thể dẫn đến việc mô hình bị ảnh hưởng bởi nhiễu (overfitting), trong khi chọn K quá lớn có thể làm mất đi các đặc điểm phân biệt của lớp (underfitting).

Không hiệu quả với dữ liệu có nhiều đặc trưng: Khi số chiều của không gian đặc trưng quá lớn (dữ liệu "high-dimensional"), KNN có thể gặp khó khăn vì khoảng cách giữa các điểm dữ liệu trở nên không đáng kể (vấn đề "curse of dimensionality").

### Ứng dụng của KNN.

Phân loại văn bản: KNN được sử dụng để phân loại văn bản trong các bài toán như phân loại email spam hay phân loại tài liệu theo chủ đề.

Nhận dạng hình ảnh: KNN có thể được áp dụng trong nhận dạng hình ảnh, đặc biệt là khi số lượng hình ảnh không quá lớn.

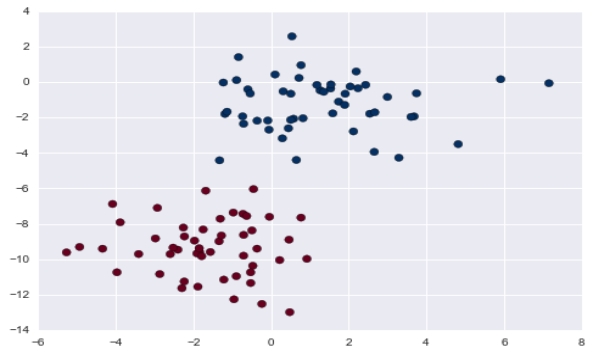
Phát hiện gian lận: KNN có thể được sử dụng trong các bài toán phát hiện gian lận, chẳng hạn như trong các giao dịch tài chính hoặc bảo hiểm.

Dự đoán giá trị: Trong các bài toán hồi quy, KNN có thể được sử dụng để dự đoán giá trị dựa trên các láng giềng gần nhất, ví dụ như dự đoán giá nhà hoặc nhiệt độ.

## Naïve Bayes.

### Khái niệm Naïve Bayes

Naïve Bayes là một nhóm các thuật toán phân loại dựa trên Định lý Bayes và giả định rằng các đặc trưng (features) trong dữ liệu là độc lập với nhau. Thuật toán Naïve Bayes thường được sử dụng trong các bài toán phân loại văn bản, nhận dạng email spam, phân loại tài liệu, và các ứng dụng khác trong học máy. Thuật toán này rất đơn giản, hiệu quả và có thể làm việc tốt ngay cả với dữ liệu có kích thước lớn.



*Hình 4 : Hình ảnh mình họa về thuật toán Naïve Bayes .*

### Định lí Bayes.

Naïve Bayes dựa trên Định lý Bayes, một công thức trong lý thuyết xác suất dùng để tính xác suất có điều kiện. Định lý Bayes nói rằng:

P(C|X) =

**Trong đó:**

* P(C|X) là xác suất có điều kiện, tức là xác suất của lớp C khi biết các đặc trưng X
* P(X|C) là xác suất có điều kiện của các đặc trưng X với lớp C
* P(C) là xác suất tiên nghiệm (prior probability) của lớp C
* P(X) là xác suất tiên nghiệm của các đặc trưng X (thường là hằng số trong quá trình tính toán).

Mục tiêu của Naïve Bayes là tính toán xác suất P(C|X) để phân loại điểm dữ liệu X vào lớp có xác suất cao nhất.

### Giả định độc lập (Naïve Assumption).

Naïve Bayes đi kèm với một giả định đơn giản nhưng mạnh mẽ: Các đặc trưng (features) trong dữ liệu là độc lập với nhau, điều này có nghĩa là sự xuất hiện của một đặc trưng không ảnh hưởng đến sự xuất hiện của các đặc trưng khác.

Với giả định này, công thức P(X|C) có thể được phân tách thành:

P(X|C) = P() P() ... P()

**Trong đó:**

* ​ là các đặc trưng của điểm dữ liệu X, và P()là xác suất của đặc trưng khi lớp là C

### Cách thức hoạt động của Naïve Bayes.

Naïve Bayes phân loại điểm dữ liệu X vào lớp Csao cho xác suất có điều kiện P(C|X) là cao nhất . Bởi vì P(X)là hằng số, nên thuật toán chỉ cần tối đa hóa P(C|X), tương đương với tối đa hóa P(X|C)P(C)

Để phân loại, Naïve Bayes tính toán:

= arg P(C)

**Trong đó:**

* P(C) là xác suất tiên nghiệm của lớp C
* là xác suất của đặc trưng với lớp C
* n là số lượng đặc trưng trong dữ liệu.

### Các loại Naïve Bayes.

Có ba loại Naïve Bayes phổ biến dựa trên kiểu dữ liệu và cách tính xác suất:

* Gaussian Naïve Bayes: Giả sử các đặc trưng có phân phối chuẩn (Gaussian). Đặc trưng của mỗi lớp được mô hình hóa bằng phân phối chuẩn với các tham số (trung bình và phương sai) tính từ dữ liệu huấn luyện.
* Multinomial Naïve Bayes: Thường được sử dụng trong các bài toán phân loại văn bản, nơi các đặc trưng là số lần xuất hiện của các từ trong tài liệu. Phân phối xác suất được mô hình hóa dưới dạng phân phối multinomial.
* Bernoulli Naïve Bayes: Dành cho dữ liệu nhị phân, nơi các đặc trưng có thể có giá trị 0 hoặc 1 (ví dụ: từ có xuất hiện trong văn bản hay không). Phân phối xác suất của các đặc trưng được mô hình hóa bằng phân phối Bernoulli.

### Ưu và nhược điểm của Naïve Bayes.

* Ưu điểm:
* Đơn giản và dễ triển khai: Naïve Bayes dễ dàng triển khai và có thể làm việc tốt với bộ dữ liệu lớn mà không cần nhiều tài nguyên tính toán.
* Nhanh chóng và hiệu quả: Vì nó chỉ tính toán xác suất, Naïve Bayes rất nhanh và hiệu quả, đặc biệt khi dữ liệu lớn.
* Làm việc tốt với dữ liệu có nhiễu: Naïve Bayes có thể hoạt động tốt ngay cả khi có một số nhiễu trong dữ liệu, vì việc giả định độc lập giúp làm giảm ảnh hưởng của các đặc trưng không liên quan.
* Hiệu quả với dữ liệu không đầy đủ: Naïve Bayes có thể xử lý tốt với dữ liệu thiếu, vì có thể bỏ qua các đặc trưng thiếu khi tính toán xác suất.
* Nhược điểm:
* Giả định độc lập mạnh mẽ: Mặc dù giả định rằng các đặc trưng là độc lập là rất đơn giản và giúp giảm độ phức tạp tính toán, nhưng nó không luôn đúng trong thực tế. Các đặc trưng có thể phụ thuộc vào nhau, điều này có thể làm giảm hiệu suất của mô hình.
* Không thích hợp với dữ liệu có đặc trưng liên tục và phức tạp: Trong trường hợp các đặc trưng có mối quan hệ phức tạp, Naïve Bayes có thể không đạt được hiệu quả tốt như các thuật toán khác như SVM hoặc Decision Trees.

### Ứng dụng của Naïve Bayes.

Phân loại văn bản: Naïve Bayes được sử dụng phổ biến trong phân loại văn bản, chẳng hạn như phân loại email spam hay không spam, phân loại tài liệu theo chủ đề, phân loại cảm xúc trong các bài đăng trên mạng xã hội, v.v.

Nhận dạng chữ viết tay: Naïve Bayes có thể được sử dụng trong nhận dạng chữ viết tay, chẳng hạn như phân loại các chữ số trong bộ dữ liệu MNIST.

Phát hiện gian lận: Naïve Bayes cũng có thể được ứng dụng trong việc phát hiện gian lận trong các hệ thống tài chính hoặc bảo hiểm.

Chẩn đoán y tế: Naïve Bayes có thể giúp trong các bài toán chẩn đoán bệnh dựa trên các đặc trưng của bệnh nhân (như các triệu chứng, kết quả xét nghiệm, v.v.).

# thực nghiệm

#### Giới thiệu về bộ dữ liệu giá nhà.

Bộ dữ liệu kc\_house\_data.csv là tập dữ liệu về giá nhà ở khu vực King County, bang Washington, Mỹ (bao gồm thành phố Seattle). Đây là một tập dữ liệu phổ biến được sử dụng trong các bài toán học máy để dự đoán giá nhà dựa trên các đặc trưng của bất động sản.

##### Tổng qua về bộ dữ liệu.

Chúng ta sử dụng pd.read\_csv() để đọc bộ dữ liệu và các phương thức như info() và describe() để kiểm tra thông tin tổng quan của bộ dữ liệu.

BẮT ĐẦU

ĐỌC file 'kc\_house\_data.csv' vào biến data

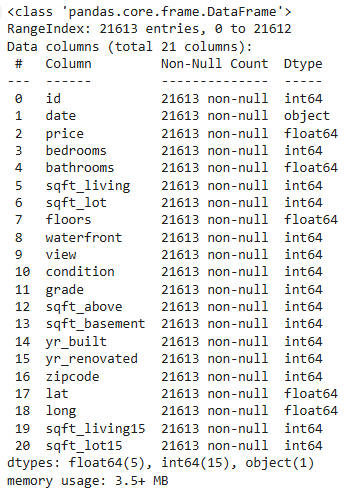
LỌC data, chỉ giữ các hàng có giá (price) lớn hơn 0

HIỂN\_THỊ thông tin tổng quan về data

HIỂN\_THỊ thống kê mô tả của data

KẾT THÚC

Thông tin tổng quan của bộ dữ liệu:



Bộ dữ liệu gồm 21 cột, 31613 dòng và không có dữ liệu bị thiếu. Có 5 cột thuộc kiểu dữ liệu float, 15 cột kiểu int, và 1 cột thuộc kiểu object.

Giá nhà trung bình là 540088.2 USD, giá thấp nhất là 75000 USD, và nhà có giá cao nhất là 7700000 USD.

Trung vị (50%): Trung vị là 450,000 USD, thấp hơn trung bình khoảng 90,000 USD. Điều này cho thấy phân phối giá nhà có thể lệch phải (right-skewed), tức là có một số ít căn nhà có giá rất cao làm kéo giá trị trung bình lên.

Độ lệch chuẩn (std): Độ lệch chuẩn là 367,127.2 USD, khá lớn so với giá trị trung bình, cho thấy giá nhà có sự biến động mạnh. Điều này phản ánh sự đa dạng trong giá nhà, từ những căn nhà giá rẻ đến những căn nhà rất đắt.

#### Xử lí dữ liệu.

Để đảm bảo tính nhất quán cho phân tích tương quan và xây dựng mô hình học máy ta cần đảm bảo dữ liệu sạch nên sẽ cần có bước tiền xử lí dữ liệu.

 Tiền xử lý dữ liệu

BẮT ĐẦU

LOẠI\_BỎ các cột 'id', 'date', 'zipcode', 'lat', 'long' khỏi data

LOẠI\_BỎ các dòng trùng lặp trong data

LỌC data, chỉ giữ các hàng có giá (price) lớn hơn 0

VỚI mỗi hàng trong data

NẾU yr\_renovated bằng 0

GÁN yr\_renovated = yr\_built

NGƯỢC LẠI

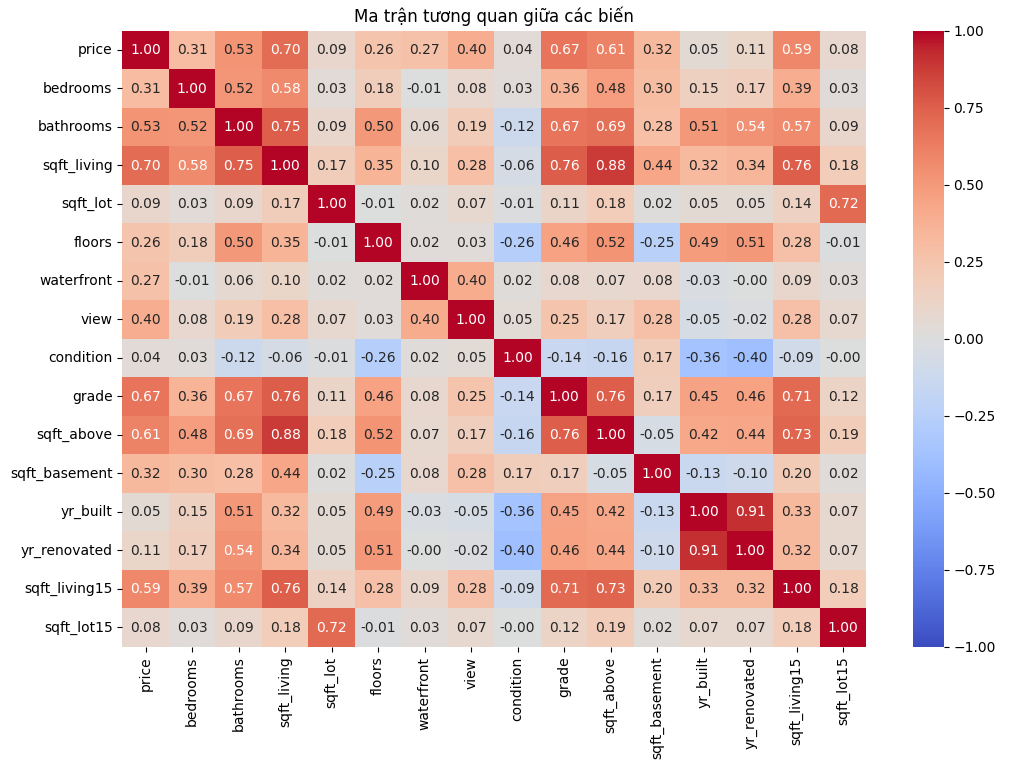
GIỮ nguyên giá trị yr\_renovated

KẾT THÚC NẾU

KẾT THÚC VỚI

KẾT THÚC

#### Tính ma trận tương quan.



Các biến có tương quan mạnh với price:

sqft\_living (0.70): Diện tích sinh hoạt có tương quan dương mạnh với giá nhà. Điều này hợp lý vì nhà có diện tích lớn thường có giá cao hơn.

grade (0.67): Chất lượng xây dựng (grade) cũng có tương quan mạnh với giá nhà. Nhà có chất lượng xây dựng cao (grade cao) thường đắt hơn.

sqft\_above (0.61): Diện tích trên mặt đất có tương quan mạnh, vì phần lớn diện tích sinh hoạt nằm ở trên mặt đất.

#### Trực quan hóa mối quan hệ giữa price và các biến số.

BẮT ĐẦU

ĐẶT danh\_sach\_dac\_trung = ['sqft\_living', 'grade', 'sqft\_above']

VỚI mỗi dac\_trung trong danh\_sach\_dac\_trung

TẠO biểu đồ phân tán với kích thước 8x6

VẼ điểm phân tán với trục X là dac\_trung và trục Y là price

ĐẶT tiêu đề biểu đồ là "Price vs dac\_trung"

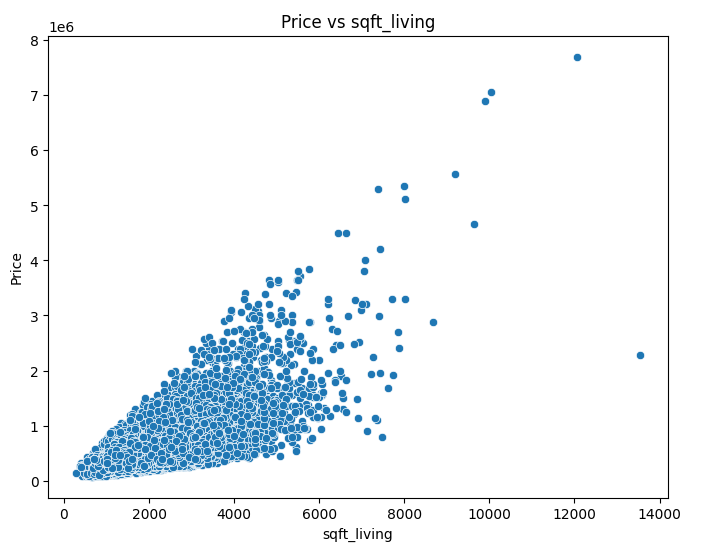
ĐẶT nhãn trục X là dac\_trung

ĐẶT nhãn trục Y là "Price"

HIỂN\_THỊ biểu đồ

KẾT THÚC VỚI

KẾT THÚC



***1.Nhận xét về mối quan hệ giữa price và sqft\_living***

**Xu hướng tổng quát:**

Có một xu hướng tăng tuyến tính giữa sqft\_living và price: Khi diện tích sinh hoạt (sqft\_living) tăng, giá nhà (price) cũng có xu hướng tăng. Điều này phù hợp với thực tế, vì nhà có diện tích lớn hơn thường có giá cao hơn do cung cấp không gian sống rộng rãi hơn.

Tuy nhiên, mối quan hệ không hoàn toàn tuyến tính hoàn hảo, vì có nhiều điểm dữ liệu phân tán khá rộng, đặc biệt ở các giá trị sqft\_living lớn.

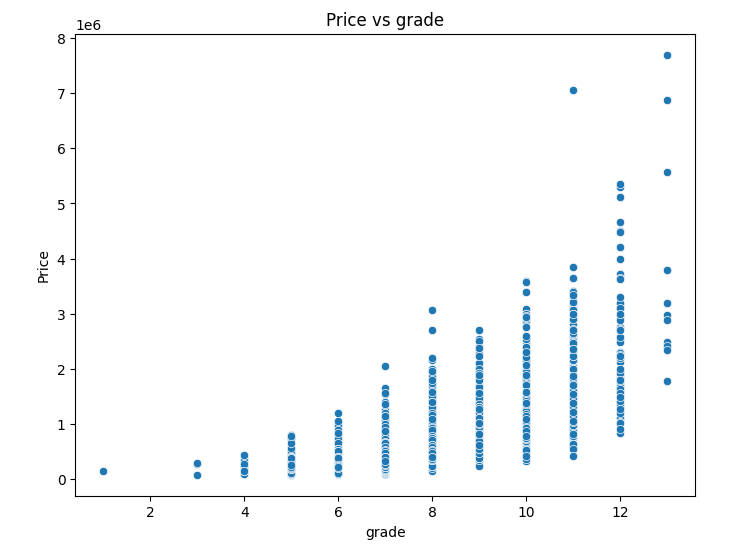
**Biến động giá nhà:**

Ở cùng một mức diện tích (sqft\_living), giá nhà có sự biến động lớn. Ví dụ: Với sqft\_living khoảng 2000 feet vuông, giá nhà dao động từ dưới 1 triệu USD đến khoảng 2.5 triệu USD. Với sqft\_living khoảng 4000 feet vuông, giá nhà dao động từ khoảng 1 triệu USD đến gần 4 triệu USD.

Sự biến động này cho thấy ngoài diện tích sinh hoạt, còn có các yếu tố khác ảnh hưởng đến giá nhà, như vị trí, chất lượng xây dựng (grade), tầm nhìn (view), mặt nước (waterfront), hoặc tình trạng nhà (condition).

**Ý nghĩa thực tế:**

Diện tích là yếu tố quan trọng: sqft\_living có ảnh hưởng rõ rệt đến giá nhà, và đây là một trong những yếu tố quan trọng nhất khi định giá bất động sản. Điều này phù hợp với hệ số tương quan cao (0.70) đã tính toán trước đó.



***2. Nhận xét về mối quan hệ giữa price và grade***

**Xu hướng tổng quát:**

Có một xu hướng tăng giữa grade và price: Khi chất lượng xây dựng (grade) tăng, giá nhà (price) cũng có xu hướng tăng. Điều này hợp lý vì nhà có chất lượng xây dựng cao hơn (grade cao) thường được xây dựng với vật liệu tốt hơn, thiết kế đẹp hơn, và có giá trị cao hơn.

Mối quan hệ này không hoàn toàn tuyến tính, nhưng có thể thấy rõ rằng các căn nhà có grade cao (từ 9 trở lên) thường có giá cao hơn đáng kể so với các căn nhà có grade thấp (dưới 6).

**Biến động giá nhà:**

Ở cùng một mức grade, giá nhà có sự biến động lớn. Ví dụ: Với grade = 7, giá nhà dao động từ khoảng 0.5 triệu USD đến gần 2 triệu USD. Với grade = 10, giá nhà dao động từ khoảng 1 triệu USD đến gần 5 triệu USD.

Sự biến động này cho thấy ngoài chất lượng xây dựng, còn có các yếu tố khác ảnh hưởng đến giá nhà, như diện tích (sqft\_living), vị trí, tầm nhìn (view), hoặc mặt nước (waterfront).

**Ý nghĩa thực tế:**

Chất lượng xây dựng là yếu tố quan trọng: grade có ảnh hưởng rõ rệt đến giá nhà, và đây là một trong những yếu tố quan trọng khi định giá bất động sản. Điều này phù hợp với hệ số tương quan cao (0.67) đã tính toán trước đó.



***3. Nhận xét về mối quan hệ giữa price và sqft\_above***

**Xu hướng tổng quát:**

Có một xu hướng tăng tuyến tính giữa sqft\_above và price: Khi diện tích trên mặt đất (sqft\_above) tăng, giá nhà (price) cũng có xu hướng tăng. Điều này hợp lý vì diện tích trên mặt đất thường chiếm phần lớn diện tích sinh hoạt của căn nhà, và nhà có diện tích lớn hơn thường có giá cao hơn.

Mối quan hệ này tương tự như mối quan hệ giữa price và sqft\_living (đã phân tích trước đó), vì sqft\_above là một phần của sqft\_living (diện tích sinh hoạt = diện tích trên mặt đất + diện tích tầng hầm).

**Biến động giá nhà:**

Ở cùng một mức sqft\_above, giá nhà có sự biến động lớn. Ví dụ: Với sqft\_above khoảng 2000 feet vuông, giá nhà dao động từ dưới 1 triệu USD đến khoảng 2.5 triệu USD. Với sqft\_above khoảng 4000 feet vuông, giá nhà dao động từ khoảng 1 triệu USD đến gần 4 triệu USD.

Sự biến động này cho thấy ngoài diện tích trên mặt đất, còn có các yếu tố khác ảnh hưởng đến giá nhà, như chất lượng xây dựng (grade), tầm nhìn (view), mặt nước (waterfront), hoặc vị trí.

**Ý nghĩa thực tế**

Diện tích trên mặt đất là yếu tố quan trọng: sqft\_above có ảnh hưởng rõ rệt đến giá nhà, và đây là một trong những yếu tố quan trọng khi định giá bất động sản. Tuy nhiên, nó kém quan trọng hơn một chút so với sqft\_living, vì không bao gồm diện tích tầng hầm.

#### Tạo biến phân loại dựa trên giá nhà.

Mục tiêu của bài toán là dự đoán giá nhà (price) dựa trên các đặc trưng (features) trong tập dữ liệu như số phòng ngủ (bedrooms), số phòng tắm (bathrooms), diện tích (sqft\_living, sqft\_lot), số tầng (floors), năm xây dựng (yr\_built), v.v. Đây là một bài toán hồi quy (regression) vì biến mục tiêu (price) là một giá trị liên tục. Tuy nhiên, để áp dụng các mô hình phân loại như Logistic Regression và Naive Bayes, ta có thể chuyển bài toán thành phân loại bằng cách phân loại giá nhà thành các nhóm (ví dụ: giá thấp, trung bình, cao).

BẮT ĐẦU

ĐẶT danh\_sach\_dac\_trung = ['bedrooms', 'bathrooms', 'sqft\_living', 'sqft\_lot', 'floors', 'waterfront', 'view',

'condition', 'grade', 'sqft\_above', 'sqft\_basement', 'yr\_built', 'yr\_renovated',

'sqft\_living15', 'sqft\_lot15']

CHỌN các cột trong danh\_sach\_dac\_trung từ data, gán vào X

TÍNH ngưỡng\_giá\_thấp = phân vị 0.33 của cột price trong data

TÍNH ngưỡng\_giá\_cao = phân vị 0.66 của cột price trong data

TẠO biến phân loại y\_classification bằng cách:

CHIA cột price thành 3 khoảng:

từ 0 đến ngưỡng\_giá\_thấp, gán nhãn 'low'

từ ngưỡng\_giá\_thấp đến ngưỡng\_giá\_cao, gán nhãn 'medium'

từ ngưỡng\_giá\_cao đến vô cực, gán nhãn 'high'

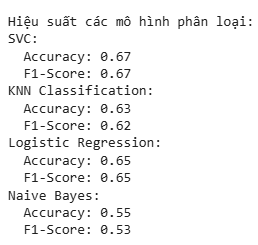
BAO\_GỒM giá trị thấp nhất trong khoảng

ĐẶT y\_regression = cột price trong data

KẾT THÚC

#### Chia tập dữ liệu và huấn luyện các mô hình SVM, Logistic Regression, KNN, Naïve Bayes.

Sau khi chia tập dữ luyện và huấn luyện các mô hình ta có được hiệu suất của các mô hình phân loại.

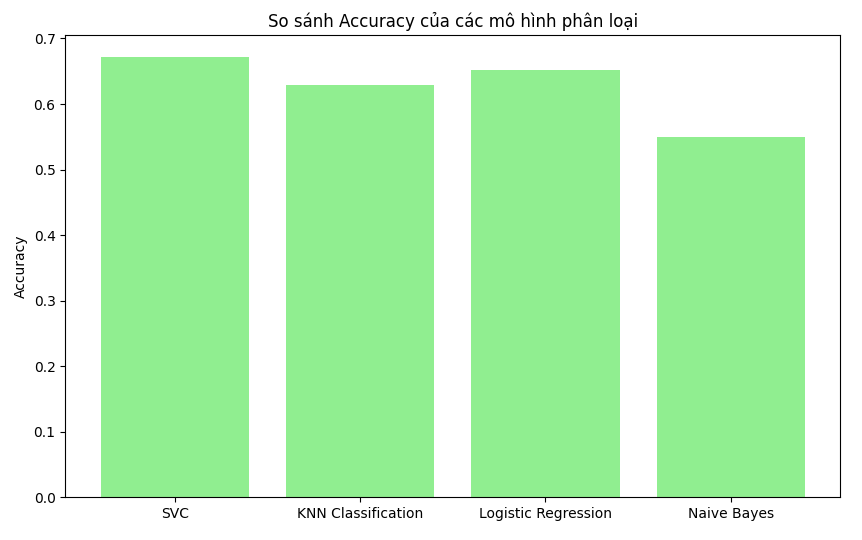


Mô hình tốt nhất: KNN Regression vượt trội với R² Score cao nhất (~0.65), cho thấy đây là mô hình phù hợp nhất trong các mô hình được thử nghiệm. KNN tận dụng được tính chất cục bộ của dữ liệu, dự đoán giá nhà dựa trên các ngôi nhà "giống" nhất trong không gian đặc trưng.

Mô hình trung bình: Logistic Regression có R² ~0.4, nhưng không nên sử dụng trong bài toán hồi quy liên tục. Hiệu suất của nó chỉ mang tính tham khảo do cách mô phỏng không phù hợp.

Mô hình kém: Naive Bayes và SVR có hiệu suất rất thấp. Naive Bayes không phù hợp cho bài toán hồi quy, còn SVR cần được tinh chỉnh thêm để cải thiện hiệu suất.

* So sánh accuracy của các mô hình phân loại.



***Biểu đồ so sánh độ chính xác (Accuracy) của các mô hình phân loại:***

SVC đạt độ chính xác cao nhất, khoảng 0.7.

KNN Classification và Logistic Regression có độ chính xác tương đương nhau, khoảng 0.68.

Naïve Bayes có độ chính xác thấp nhất, khoảng 0.62.

Nhận xét: SVC là mô hình hiệu quả nhất trong các mô hình được so sánh phân loại ở bài toán này, trong khi Naïve Bayes có hiệu suất thấp nhất. KNN và Logistic Regression có hiệu suất tương đối ổn nhưng không vượt trội bằng SVC.

Trục X: Giá thực tế (Actual Price), với thang giá trị từ 0 đến 5e6 (5 triệu).

Trục Y: Giá dự đoán (Predicted Price), cũng với thang giá trị từ 0 đến 5e6.

Điểm dữ liệu: Các điểm màu xanh dương thể hiện cặp giá trị (giá thực tế, giá dự đoán) cho từng ngôi nhà trong tập kiểm tra.

Đường đỏ chéo (đường lý tưởng): Đường thẳng với phương trình y = x, biểu thị trường hợp giá dự đoán hoàn toàn khớp với giá thực tế. Các điểm càng gần đường này, mô hình càng chính xác.

**Phân bố điểm:**

Hầu hết các điểm tập trung ở vùng giá thấp, dưới 1 triệu USD (giá thực tế và giá dự đoán đều nhỏ hơn 1e6).

Một số ít điểm nằm ở vùng giá cao hơn, từ 1 triệu đến 5 triệu USD, nhưng rất thưa thớt.

Điều này cho thấy tập dữ liệu có thể bị lệch (skewed), với phần lớn giá nhà ở mức thấp và chỉ một số ít nhà có giá cao (có thể là các ngôi nhà cao cấp hoặc ngoại lệ).

**Hiệu suất của SVR:**

Ở vùng giá thấp (<1 triệu USD): Các điểm tập trung sát đường đỏ chéo, cho thấy SVR dự đoán khá chính xác ở phân khúc giá thấp.

Tuy nhiên, các điểm không nằm hoàn toàn trên đường lý tưởng, vẫn có sự sai lệch nhẹ.

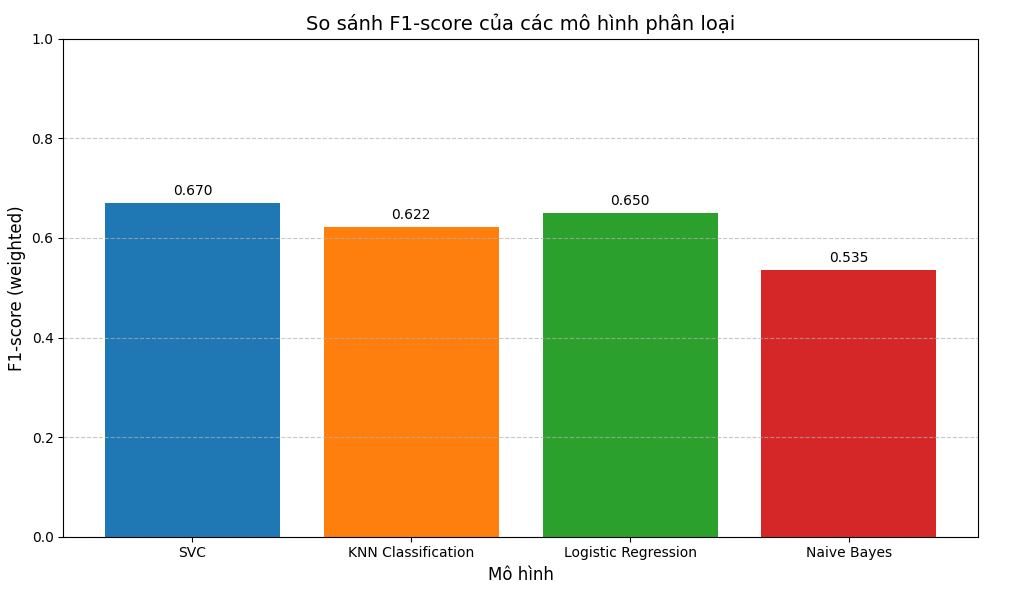
Ở vùng giá cao (>1 triệu USD): Các điểm phân tán xa đường đỏ chéo, với giá dự đoán thường thấp hơn giá thực tế. Ví dụ, một ngôi nhà có giá thực tế 5 triệu USD nhưng SVR chỉ dự đoán khoảng 1 triệu USD. Điều này cho thấy SVR không hiệu quả trong việc dự đoán giá nhà ở phân khúc cao cấp.

**Xu hướng dự đoán:**

SVR có xu hướng dự đoán thấp (underpredict), đặc biệt ở các mức giá cao. Các điểm trên biểu đồ thường nằm dưới đường đỏ chéo ở vùng giá cao, nghĩa là giá dự đoán nhỏ hơn giá thực tế.

Ở vùng giá thấp, SVR dự đoán tương đối sát với giá thực tế, nhưng vẫn có sự phân tán nhẹ, cho thấy mô hình không hoàn toàn chính xác.

* So sánh F1-score của các mô hình phân loại.



Nhận xét F1-score của các mô hình:

* **Mô hình SVC (F1-score: 0.670):**

SVC đạt F1-score cao nhất trong số các mô hình, cho thấy hiệu suất tốt nhất trong việc cân bằng giữa precision và recall. Điều này có thể do SVC với kernel 'rbf' (Radial Basis Function) đã học tốt các mẫu dữ liệu phi tuyến tính trong tập dữ liệu nhà ở, vốn có nhiều đặc trưng phức tạp như diện tích, số phòng, và năm xây dựng.

* **Mô hình Logistic Regression (F1-score: 0.650):**

Logistic Regression có hiệu suất khá tốt, chỉ thấp hơn SVC một chút (0.020). Điều này cho thấy mô hình tuyến tính như Logistic Regression vẫn có thể hoạt động hiệu quả trên bài toán phân loại giá nhà thành các nhóm (thấp, trung bình, cao), dù dữ liệu có thể không hoàn toàn tuyến tính.

* **Mô hình KNN Classification (F1-score: 0.622):**

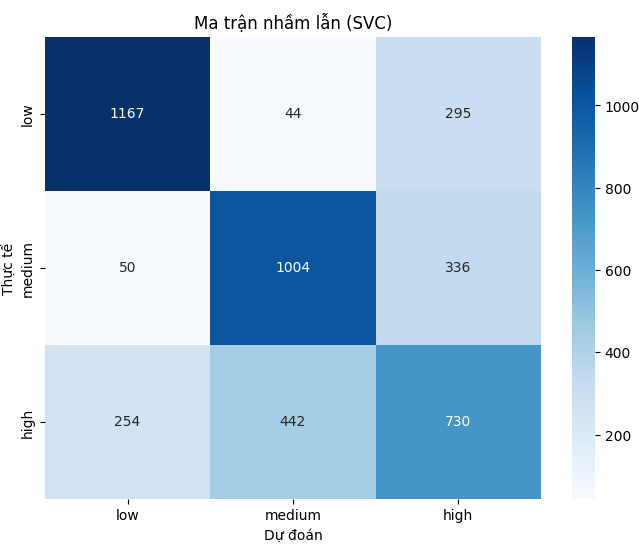
KNN có F1-score thấp hơn SVC và Logistic Regression, nhưng vẫn ở mức chấp nhận được. Hiệu suất của KNN phụ thuộc nhiều vào tham số n\_neighbors=5 và cách chuẩn hóa dữ liệu (dữ liệu đã được chuẩn hóa bằng StandardScaler trong code). Tuy nhiên, KNN có thể gặp khó khăn khi dữ liệu có nhiều nhiễu hoặc các lớp không được phân tách rõ ràng, như được chỉ ra trong ma trận nhầm lẫn (có xu hướng overpredict "low" thành "high" và underpredict "high" thành "medium").

* Mô hình Naive Bayes (F1-score: 0.535):

Naive Bayes có F1-score thấp nhất, chỉ đạt 0.535, kém hơn đáng kể so với các mô hình khác. Điều này có thể do giả định của Naive Bayes (các đặc trưng độc lập với nhau) không phù hợp với tập dữ liệu này. Trong dữ liệu nhà ở, các đặc trưng như diện tích sinh hoạt (sqft\_living), số phòng ngủ (bedrooms), và năm xây dựng (yr\_built) có thể có mối tương quan mạnh với nhau, làm giảm hiệu quả của Naive Bayes.

#### Ma trận nhầm lẫn.

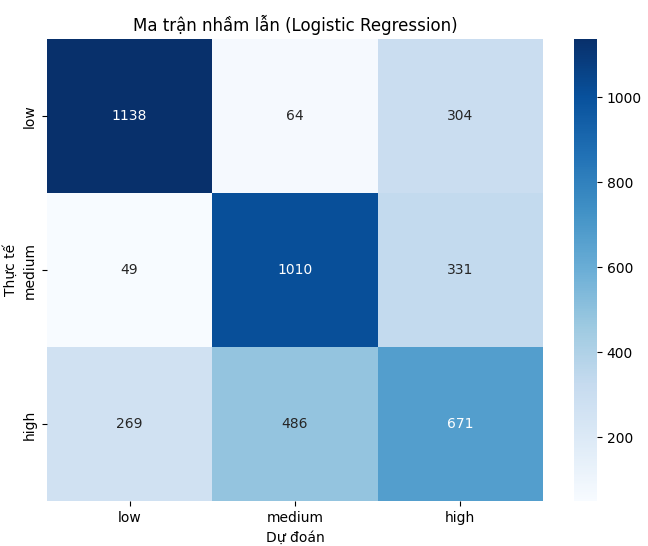
* Ma trận nhầm lẫn của SVC.



**Nhận xét:**

Ma trận nhầm lẫn của SVC cho thấy mô hình đạt độ chính xác tổng thể cao, hoạt động tốt trên lớp "high" (1489) và "medium" (1233), nhưng kém hơn trên lớp "low" (403) với nhiều mẫu bị dự đoán sai thành "high". Xu hướng overpredict cho "low" và "medium", underpredict cho "high" có thể do dữ liệu không cân bằng và chồng lấn đặc trưng.

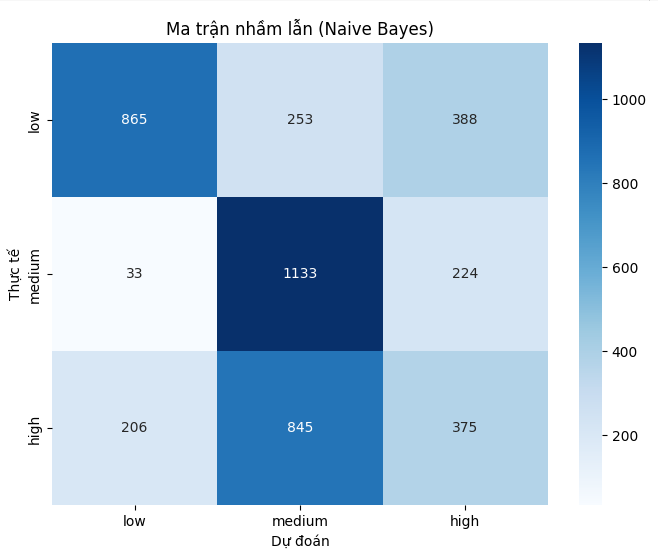
* Ma trận nhầm lẫn Logistic Regression.



**Nhận xét:**

Ma trận nhầm lẫn của Logistic Regression cho thấy độ chính xác tổng thể là 70.1% (3129/4462 mẫu dự đoán đúng). Mô hình hoạt động tốt trên lớp "medium" (72.9%) và "high" (69.2%), nhưng kém trên lớp "low" (61.2%) với 222 mẫu bị dự đoán sai thành "high". Có xu hướng overpredict cho "low" và "medium" (dự đoán thành "high") và underpredict cho "high" (dự đoán thành "medium").

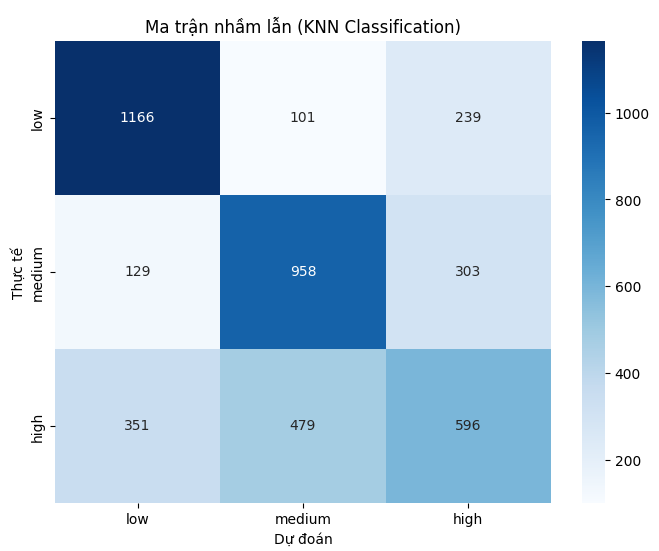
* Ma trận nhầm lẫn Naïve Bayes.



**Nhận xét:**

Ma trận nhầm lẫn của Naive Bayes cho thấy độ chính xác tổng thể là 62.6% (2635/4205 mẫu dự đoán đúng). Mô hình hoạt động tốt trên lớp "medium" (81.5%), nhưng kém trên lớp "high" (40.2%) và "low" (64.8%) với nhiều mẫu bị dự đoán sai thành "medium" (972 mẫu "high" và 27 mẫu "low"). Có xu hướng overpredict thành "medium".

* Ma trận nhầm lẫn KNN.



**Nhận xét:**

Ma trận nhầm lẫn của KNN Classification cho thấy độ chính xác tổng thể là 68.5% (2889/4218 mẫu dự đoán đúng). Mô hình hoạt động tốt trên lớp "medium" (73.4%) và "low" (66.3%), nhưng kém hơn trên lớp "high" (67.2%) với 508 mẫu bị dự đoán sai thành "medium". Có xu hướng overpredict "low" thành "high" và underpredict "high" thành "medium".

# tổng kết và kết luận

### So sánh và đánh giá.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tiêu chí | SVM | KNN | Logistic Regression | Naïve Bayes |
| Loại thuật toán | Phân loại (có thể dùng cho hồi quy) | Phân loại (có thể dùng cho hồi quy) | Phân loại (hồi quy tuyến tính cho xác suất) | Phân loại (dựa trên xác suất) |
| Nguyên lý | Tìm siêu phẳng tối ưu để phân tách lớp với khoảng cách lớn nhất | Phân loại dựa trên k láng giềng gần nhất (khoảng cách Euclidean,...) | Dự đoán xác suất lớp bằng hàm sigmoid trên tổ hợp tuyến tính | Dựa trên định lý Bayes, giả định các đặc trưng độc lập |
| Hiệu quả với dữ liệu lớn | Tốn tài nguyên tính toán, chậm với tập dữ liệu lớn | Chậm khi dự đoán (tính khoảng cách cho toàn bộ dữ liệu) | Hiệu quả, nhanh với dữ liệu lớn | Rất nhanh, phù hợp với dữ liệu lớn |
| Hiệu quả với dữ liệu nhỏ | Hiệu quả, đặc biệt với dữ liệu phi tuyến tính (dùng kernel) | Hiệu quả, dễ triển khai | Hiệu quả, đơn giản | Hiệu quả, đặc biệt với dữ liệu văn bản |
| Xử lý dữ liệu phi tuyến tính | Tốt (với kernel như RBF, polynomial) | Tốt (dựa trên khoảng cách) | Kém (yêu cầu biến đổi đặc trưng) | Kém (giả định tuyến tính trong phân phối) |
| Giả định | Không có giả định cụ thể, tối ưu hóa margin | Không có giả định, dựa trên khoảng cách | Giả định dữ liệu có thể phân tách tuyến tính (log-odds) | Giả định đặc trưng độc lập (thường không đúng trong thực tế) |
| Tham số cần điều chỉnh | C (độ lỗi), kernel, gamma (nếu dùng kernel) | k (số láng giềng), metric khoảng cách | Regularization (L1/L2), learning rate | Loại phân phối (Gaussian, Multinomial,...) |
| Ưu điểm | - Hiệu quả với dữ liệu phi tuyến tính.  - Tốt với dữ liệu chiều cao | - Đơn giản, không cần huấn luyện  - Linh hoạt với dữ liệu phức tạp | - Nhanh, dễ hiểu  - Tốt cho bài toán nhị phân | - Rất nhanh  - Tốt cho dữ liệu văn bản, phân loại đa lớp |
| Nhược điểm | - Chậm với dữ liệu lớn.  - Cần chọn kernel và tham số cẩn thận | - Chậm khi dự đoán.  - Nhạy cảm với nhiễu và dữ liệu không cân bằng | - Kém với dữ liệu phi tuyến tính.  - Cần tiền xử lý tốt | - Giả định độc lập thường không thực tế.  - Kém với dữ liệu phức tạp |
| Ứng dụng phổ biến | Phân loại văn bản, nhận diện hình ảnh, bioinformatics | Nhận diện mẫu, hệ thống gợi ý | Phân loại nhị phân (spam, bệnh lý), scoring tín dụng | Phân loại văn bản, phát hiện spam, phân tích cảm xúc |

### Kết luận.

Qua phân tích và so sánh, mỗi thuật toán phân loại đều có những điểm mạnh và hạn chế riêng, phù hợp với các loại bài toán và tập dữ liệu khác nhau. Việc lựa chọn thuật toán phụ thuộc vào đặc điểm của bài toán, kích thước dữ liệu, tính chất của đặc trưng và yêu cầu về hiệu suất:

* **SVM** là lựa chọn lý tưởng khi cần độ chính xác cao và dữ liệu có kích thước vừa phải, đặc biệt với các bài toán phi tuyến.
* **Logistic Regression** phù hợp cho các bài toán phân loại nhị phân đơn giản, yêu cầu diễn giải rõ ràng và dữ liệu tuyến tính.
* **KNN** hữu ích cho các bài toán nhỏ, phi tuyến và không yêu cầu mô hình hóa phức tạp, nhưng không phù hợp với dữ liệu lớn.
* **Naïve Bayes** là lựa chọn tối ưu cho các bài toán phân loại văn bản hoặc dữ liệu lớn, nơi tốc độ và khả năng xử lý nhiễu là ưu tiên.

Phần thực nghiệm trên bộ dữ liệu giá nhà đã cung cấp cái nhìn thực tiễn về cách các đặc trưng ảnh hưởng đến giá nhà, làm nền tảng cho việc áp dụng các thuật toán học máy để dự đoán giá nhà trong tương lai. Các thuật toán như SVM, Logistic Regression, KNN và Naïve Bayes có thể được thử nghiệm trên bộ dữ liệu này để so sánh hiệu suất thực tế, từ đó xác định mô hình phù hợp nhất.

Báo cáo này không chỉ cung cấp một nền tảng lý thuyết vững chắc về các thuật toán phân loại mà còn đặt ra định hướng cho các nghiên cứu tiếp theo, bao gồm việc thử nghiệm thực tế các mô hình trên bộ dữ liệu giá nhà, tối ưu hóa tham số và khám phá các kỹ thuật học máy nâng cao hơn như học sâu hoặc ensemble learning. Việc hiểu rõ ưu nhược điểm của từng thuật toán sẽ giúp các nhà khoa học dữ liệu đưa ra quyết định sáng suốt trong việc xây dựng các hệ thống phân loại hiệu quả và phù hợp với nhu cầu thực tiễn.