|  |
| --- |
|  |
| **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------** |
| **Nhận diện thương hiệu Logo HaUI** |
| **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**  **KHOA HỌC MÁY TÍNH** |
| **XÂY DỰNG HỆ THỐNG PHÂN LOẠI CÁC VÌ SAO**  **BẰNG THUẬT TOÁN HỌC MÁY** |
| **CBHD:** ThS.Phạm Thị Kim Phượng |
| **Sinh viên:**  Trần Thị Ánh Linh |
| **Mã sinh viên:** 2020606785  **Lớp:** KHMT02  **Khóa:** K15 |
|  |
|  |
| **Hà Nội - Năm 2024** |

# LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành nhất đến cô Phạm Thị Kim Phượng về sự hướng dẫn và hỗ trợ trong quá trình làm đồ án tốt nghiệp của em. Dưới sự chỉ dẫn của cô Pham Thị Kim Phượng, em đã có cơ hội học hỏi và áp dụng các kiến thức lý thuyết thành thạo vào thực tế. Quá trình này không chỉ giúp em nắm vững kiến thức mà còn mở ra một cánh cửa mới đối với thế giới của khoa học dữ liệu.

Những kiến thức và kỹ năng mà em đã học được không chỉ giúp em hoàn thành đề tài một cách thành công mà còn là nền tảng quan trọng cho sự phát triển cá nhân và sự nghiệp trong tương lai. Em rất biết ơn vì sự hỗ trợ và khích lệ không ngừng từ phía cô dù em có rất nhiều sự thiếu sót, và em sẽ tiếp tục áp dụng và phát triển những kiến thức này trong những dự án tương lai.

Em xin chân thành cảm ơn!

Trân trọng,

Trần Thị Ánh Linh

# MỞ ĐẦU

1. **Lý do chọn đề tài**

Với sự phát triển mạnh mẽ của công nghệ thông tin và sự gia tăng khối lượng dữ liệu thiên văn học, việc áp dụng các thuật toán học máy để phân loại các vì sao đã trở thành một xu hướng tất yếu. Các hệ thống phân loại truyền thống, dựa vào quan sát và phân tích thủ công, không còn đáp ứng được nhu cầu phân tích dữ liệu khổng lồ và phức tạp. Thuật toán học máy không chỉ giúp tự động hóa quá trình phân loại mà còn nâng cao độ chính xác và hiệu quả. Đề tài "Xây dựng hệ thống phân loại các vì sao bằng thuật toán học máy" được lựa chọn nhằm giải quyết vấn đề này, đồng thời đóng góp vào lĩnh vực thiên văn học hiện đại.

1. **Mục tiêu nghiên cứu**

* Xây dựng mô hình học máy: Phát triển mô hình học máy có khả năng phân loại các vì sao dựa trên các đặc trưng quan sát được.
* Tối ưu hóa hiệu suất mô hình: Nâng cao độ chính xác và khả năng dự đoán của mô hình phân loại thông qua việc tinh chỉnh các tham số và lựa chọn thuật toán phù hợp.
* Ứng dụng thực tế: Áp dụng mô hình đã phát triển vào các dữ liệu thiên văn thực tế để kiểm tra và đánh giá hiệu quả.
* Xây dựng hệ thống tự động: Thiết kế hệ thống phần mềm tự động hóa quá trình phân loại các vì sao, giúp các nhà thiên văn học tiết kiệm thời gian và nguồn lực.

1. **Đối tượng nghiên cứu**

Đề tài tập trung nghiên cứu các thành phần sau:

* Mô hình học máy: Các thuật toán học máy như SVM, Random Forest, KNN, v.v.
* Phương pháp phân loại: Các phương pháp phân loại dựa trên các đặc trưng quang phổ, độ sáng, bán kính, nhiệt độ và màu sắc của các vì sao.
* Công cụ và ngôn ngữ: Sử dụng các công cụ như Python, Scikit-learn và các thư viện hỗ trợ khác.
* Lý thuyết ứng dụng: Áp dụng các lý thuyết về học máy và khai thác dữ liệu để tối ưu hóa quá trình phân loại.

1. **Phạm vi nghiên cứu**

* Thời gian: Nghiên cứu và phát triển hệ thống trong khoảng thời gian từ tháng 3 năm 2024 đến tháng 6 năm 2024.
* Không gian: Phạm vi nghiên cứu tập trung vào dữ liệu thiên văn học từ các cơ quan và tổ chức như NASA, ESA, các đài thiên văn trên toàn cầu và wikipedia.
* Lĩnh vực nghiên cứu: Nghiên cứu tập trung vào lĩnh vực ứng dụng học máy trong thiên văn học, cụ thể là phân loại các vì sao.

1. **Kết quả mong muốn đạt được**

* Mô hình phân loại chính xác: Xây dựng và triển khai mô hình học máy với độ chính xác cao trong việc phân loại các vì sao.
* Báo cáo khoa học: Hoàn thành báo cáo khoa học chi tiết về quá trình nghiên cứu, các kết quả đạt được.

1. **Cấu trúc của báo cáo**

Báo cáo thực tập được cấu trúc thành các phần chính như sau:

* Chương 1: Tìm hiểu về thuật toán học máy được sử dụng
* Chương 2: Giới thiệu bài toán
* Chương 3: Kết quả bài toán

MỤC LỤC

[LỜI CẢM ƠN 2](#_Toc167386818)

[MỞ ĐẦU 3](#_Toc167386819)

[CHƯƠNG 1: TÌM HIỂU VỀ CÁC THUẬT TOÁN HỌC MÁY 8](#_Toc167386820)

[1.1. Phân loại thuật toán học máy 8](#_Toc167386821)

[1.1.1. Supervised Learning (Học có giám sát) 8](#_Toc167386822)

[1.1.2. Unsupervised Learning (Học không giám sát) 10](#_Toc167386823)

[1.1.3. Semi-Supervised Learning (Học bán giám sát) 11](#_Toc167386824)

[1.1.4. Reinforcement Learning (Học Củng Cố) 11](#_Toc167386825)

[1.2. Một số thuật toán học máy 12](#_Toc167386826)

[1.2.1. Decision Trees (Cây quyết định) 12](#_Toc167386827)

[1.2.2. Thuật toán phân lớp Naïve Bayes 15](#_Toc167386828)

[1.2.3 Thuật toán Hồi quy Logistic (Logistic Regression) 22](#_Toc167386829)

[1.2.4. Thuật toán SVM (Support Vector Machine) 30](#_Toc167386830)

[CHƯƠNG 2: GIỚI THIỆU BÀI TOÁN 37](#_Toc167386831)

[2.1. Hoạt động của hệ thống 37](#_Toc167386832)

[2.1.1. Thu thập dữ liệu về các vì sao: 37](#_Toc167386833)

[2.1.2. Tiền xử lý dữ liệu: 37](#_Toc167386834)

[2.1.3. Phân tích dữ liệu 37](#_Toc167386835)

[2.1.4. Xây dựng mô hình phân loại 38](#_Toc167386836)

[2.1.5 Tinh chỉnh mô hình 39](#_Toc167386837)

[2.2. Chi tiết bài toán 39](#_Toc167386838)

[2.2.1. Dữ liệu đầu vào 39](#_Toc167386839)

[2.2.2. Dữ liệu đầu ra 40](#_Toc167386840)

[CHƯƠNG 3: KẾT QUẢ BÀI TOÁN 41](#_Toc167386841)

[3.1 Dữ liệu bài toán 41](#_Toc167386842)

[3.2 Thư viện của chương trình 43](#_Toc167386843)

[3.3 Tiền xử lý 45](#_Toc167386844)

[*3.3.1 Kiểm tra phân phối của các thông số* 45](#_Toc167386845)

[*3.3.2 Kiểm tra các giá trị null* 47](#_Toc167386846)

[3.4 Phân tích dữ liệu 48](#_Toc167386847)

[*3.4.1 Phân bổ ở các loại sao* 48](#_Toc167386848)

[*3.4.2 Mật độ xác suất dựa trên 1 giá trị đặc điểm* 50](#_Toc167386849)

[*3.4.3 Phân loại dựa trên 2 yếu tố* 54](#_Toc167386850)

[3.4.4 Tổng quan 60](#_Toc167386851)

[3.5 Xây dựng mô hình 62](#_Toc167386852)

[3.5.1 Chuẩn hóa dữ liệu 62](#_Toc167386853)

[3.5.2 Huấn luyện mô hình 63](#_Toc167386854)

[3.5.3 Chương trình Test tìm kiếm sao dựa vào đặc điểm 69](#_Toc167386855)

[KẾT LUẬN 72](#_Toc167386856)

[TÀI LIỆU THAM KHẢO 73](#_Toc167386857)

**MỤC LỤC HÌNH ẢNH**

[Hinh1.1Một số activation cho mô hình tuyến tính 22](#_Toc167385984)

[Hinh 1.2 Tại sao Linear Regression không phù hợp? 23](#_Toc167385985)

[Hinh 1.3 Các mặt phân cách hai classes linearly separable. 31](#_Toc167385986)

[Hinh 1.4 Margin của hai classes là bằng nhau và lớn nhất có thể. 32](#_Toc167385987)

[Hinh 1.5 Phân tích bài toán SVM. 34](#_Toc167385988)

[Hinh 1.6 Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn 36](#_Toc167385989)

[Hinh 2.1 Types of star 37](#_Toc167385990)

[Hinh 3.1 Bảng dữ liệu các vì sao 42](#_Toc167385991)

[Hinh 3.2 Thư viện chương trình 43](#_Toc167385992)

[Hinh 3.3 Kiểm tra phân phối các thông số 46](#_Toc167385993)

[Hinh 3.4 Kết quả kiểm tra phân phối các thông số 46](#_Toc167385994)

[Hinh 3.5 Kiểm tra các giá trị Null 47](#_Toc167385995)

[Hinh 3.7 Chương trình phân bổ các loài sao 48](#_Toc167385996)

[Hinh 3.8 Biểu đồ phân bổ của các vì sao 49](#_Toc167385997)

[Hinh 3.9 Biểu đồ mật độ xác suất theo giá trị nhiệt độ 50](#_Toc167385998)

[Hinh 3.13 Biểu đồ Dựa vào độ sáng tuyệt đối 54](#_Toc167385999)

[Hinh 3.11 Biểu đồ Dựa vào độ sáng và nhiệt độ , bán kính và nhiệt độ 56](#_Toc167386000)

[Hinh 3.12 Biểu đồ Dựa vào màu sắc và nhiệt độ, lớp quang phổ và nhiệt độ 59](#_Toc167386001)

[Hinh 3.14 Biểu đồ tổng quan các yếu tố 61](#_Toc167386002)

[Hinh 3.15 Chuẩn hóa dữ liệu 63](#_Toc167386003)

[Hinh 3.16 Huấn luyện mô hình 64](#_Toc167386004)

[Hinh 3.17 Kết quả độ chính xác của mô hình 66](#_Toc167386005)

[Hinh 3.18 Kết quả thực tế và dự đoán 67](#_Toc167386006)

[Hinh 3.19 Đánh giá hiệu suất mô hình 67](#_Toc167386007)

[Hinh 3.20 Chương trình test 69](#_Toc167386008)

# CHƯƠNG 1: TÌM HIỂU VỀ CÁC THUẬT TOÁN HỌC MÁY

## Phân loại thuật toán học máy

Theo phương thức học, các thuật toán Machine Learning thường được chia làm 4 nhóm: Supervised learning, Unsupervised learning, Semi-supervised learning và Reinforcement learning. *Có một số cách phân nhóm không có Semi-supervised learning hoặc Reinforcement learning.*

### Supervised Learning (Học có giám sát)

Supervised learning là thuật toán dự đoán đầu ra (outcome) của một dữ liệu mới (new input) dựa trên các cặp (*input, outcome*) đã biết từ trước. Cặp dữ liệu này còn được gọi là (*data, label*), tức (*dữ liệu, nhãn*). Supervised learning là nhóm phổ biến nhất trong các thuật toán Machine Learning.

Cụ thể, trong supervised learning, mục tiêu là xây dựng một hàm số hoặc mô hình dự đoán đầu ra (nhãn) tương ứng với mỗi đầu vào (đặc trưng). Một cách toán học, chúng ta có thể diễn giải quá trình này như sau:

Khi chúng ra có một tập hợp biến đầu vào X={x1,x2,…,xN}và một tập hợp nhãn tương ứng Y={y1,y2,…,yN}, trong đó xi,yi là các vector. Các cặp dữ liệu biết trước (xi,yi) ∈ X×Y được gọi là tập *training data* (dữ liệu huấn luyện). Từ tập training data này, chúng ta cần tạo ra một hàm số ánh xạ mỗi phần tử từ tập X sang một phần tử (xấp xỉ) tương ứng của tập Y:

Yi ≈ f(xi),  ∀i =1,2,…,N,

Mục đích là xấp xỉ hàm số f thật tốt để khi có một dữ liệu x mới, chúng ta có thể tính được nhãn tương ứng của nó y=f(x).

*Ví dụ 1:* trong nhận dạng chữ viết tay, ta có ảnh của hàng nghìn ví dụ của mỗi chữ số được viết bởi nhiều người khác nhau. Chúng ta đưa các bức ảnh này vào trong một thuật toán và chỉ cho nó biết mỗi bức ảnh tương ứng với chữ số nào. Sau khi thuật toán tạo ra một mô hình, tức một hàm số mà đầu vào là một bức ảnh và đầu ra là một chữ số, khi nhận được một bức ảnh mới mà mô hình chưa nhìn thấy bao giờ, nó sẽ dự đoán bức ảnh đó chứa chữ số nào.

Ví dụ này khá giống với cách học của con người khi còn nhỏ. Ta đưa bảng chữ cái cho một đứa trẻ và chỉ cho chúng đây là chữ A, đây là chữ B. Sau một vài lần được dạy thì trẻ có thể nhận biết được đâu là chữ A, đâu là chữ B trong một cuốn sách mà chúng chưa nhìn thấy bao giờ.

*Ví dụ 2***:** Thuật toán dò các khuôn mặt trong một bức ảnh đã được phát triển từ rất lâu. Thời gian đầu, facebook sử dụng thuật toán này để chỉ ra các khuôn mặt trong một bức ảnh và yêu cầu người dùng *tag friends* - tức gán nhãn cho mỗi khuôn mặt. Số lượng cặp dữ liệu (*khuôn mặt, tên người*) càng lớn, độ chính xác ở những lần tự động *tag* tiếp theo sẽ càng lớn.

*Ví dụ 3:* Bản thân thuật toán dò tìm các khuôn mặt trong 1 bức ảnh cũng là một thuật toán Supervised learning với training data (dữ liệu học) là hàng ngàn cặp (*ảnh, mặt người*) và (*ảnh, không phải mặt người*) được đưa vào. Chú ý là dữ liệu này chỉ phân biệt *mặt người* và *không phải mặt người* mà không phân biệt khuôn mặt của những người khác nhau.

Thuật toán supervised learning còn được tiếp tục chia nhỏ ra thành hai loại chính là Classification (Phân loại) và Regression (Hồi quy).

#### Classification (Phân loại)

Một bài toán được gọi là classification nếu các label của input data được chia thành một số hữu hạn nhóm. Ví dụ: Gmail xác định xem một email có phải là spam hay không; các hãng tín dụng xác định xem một khách hàng có khả năng thanh toán nợ hay không. Ba ví dụ phía trên được chia vào loại này.

#### 1.1.1.2 Regression (Hồi quy)

Nếu *label* không được chia thành các nhóm mà là một giá trị thực cụ thể. Ví dụ: một căn nhà rộng x m2, có y phòng ngủ và cách trung tâm thành phố z km sẽ có giá là bao nhiêu?

### Unsupervised Learning (Học không giám sát)

Trong thuật toán này, chúng ta không biết được *outcome* hay *nhãn* mà chỉ có dữ liệu đầu vào. Thuật toán unsupervised learning sẽ dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó, ví dụ như phân nhóm (clustering) hoặc giảm số chiều của dữ liệu (dimension reduction) để thuận tiện trong việc lưu trữ và tính toán.

Một cách toán học, Unsupervised learning là khi chúng ta chỉ có dữ liệu vào X mà không biết *nhãn* Y tương ứng.

Những thuật toán loại này được gọi là Unsupervised learning vì không giống như Supervised learning, chúng ta không biết câu trả lời chính xác cho mỗi dữ liệu đầu vào. Giống như khi ta học, không có thầy cô giáo nào chỉ cho ta biết đó là chữ A hay chữ B. Cụm *không giám sát* được đặt tên theo nghĩa này.

Các bài toán Unsupervised learning được tiếp tục chia nhỏ thành hai loại:

#### 1.1.2.1 Clustering (phân nhóm)

Một bài toán phân nhóm toàn bộ dữ liệu X thành các nhóm nhỏ dựa trên sự liên quan giữa các dữ liệu trong mỗi nhóm. Ví dụ: phân nhóm khách hàng dựa trên hành vi mua hàng. Điều này cũng giống như việc ta đưa cho một đứa trẻ rất nhiều mảnh ghép với các hình thù và màu sắc khác nhau, ví dụ tam giác, vuông, tròn với màu xanh và đỏ, sau đó yêu cầu trẻ phân chúng thành từng nhóm. Mặc dù không cho trẻ biết mảnh nào tương ứng với hình nào hoặc màu nào, nhiều khả năng chúng vẫn có thể phân loại các mảnh ghép theo màu hoặc hình dạng.

#### 1.1.2.2 Association

Là bài toán khi chúng ta muốn khám phá ra một quy luật dựa trên nhiều dữ liệu cho trước. Ví dụ: những khách hàng nam mua quần áo thường có xu hướng mua thêm đồng hồ hoặc thắt lưng; những khán giả xem phim Spider Man thường có xu hướng xem thêm phim Bat Man, dựa vào đó tạo ra một hệ thống gợi ý khách hàng (Recommendation System), thúc đẩy nhu cầu mua sắm.

### Semi-Supervised Learning (Học bán giám sát)

* 1. Các bài toán khi chúng ta có một lượng lớn dữ liệu X nhưng chỉ một phần trong chúng được gán nhãn được gọi là Semi-Supervised Learning. Những bài toán thuộc nhóm này nằm giữa hai nhóm được nêu bên trên.
  2. Một ví dụ điển hình của nhóm này là chỉ có một phần ảnh hoặc văn bản được gán nhãn (ví dụ bức ảnh về người, động vật hoặc các văn bản khoa học, chính trị) và phần lớn các bức ảnh/văn bản khác chưa được gán nhãn được thu thập từ internet. Thực tế cho thấy rất nhiều các bài toán Machine Learning thuộc vào nhóm này vì việc thu thập dữ liệu có nhãn tốn rất nhiều thời gian và có chi phí cao. Rất nhiều loại dữ liệu thậm chí cần phải có chuyên gia mới gán nhãn được (ảnh y học chẳng hạn). Ngược lại, dữ liệu chưa có nhãn có thể được thu thập với chi phí thấp từ internet.

### Reinforcement Learning (Học Củng Cố)

Reinforcement learning là các bài toán giúp cho một hệ thống tự động xác định hành vi dựa trên hoàn cảnh để đạt được lợi ích cao nhất (maximizing the performance). Hiện tại, Reinforcement learning chủ yếu được áp dụng vào Lý Thuyết Trò Chơi (Game Theory), các thuật toán cần xác định nước đi tiếp theo để đạt được điểm số cao nhất.

## Một số thuật toán học máy

### 1.2.1. Decision Trees (Cây quyết định)

#### 1.2.1.1 Định Nghĩa

Cây quyết định là một mô hình học máy có cấu trúc dạng cây, mỗi nút trong cây đại diện cho một điều kiện kiểm tra trên một thuộc tính đặc trưng, mỗi cành của cây đại diện cho một quytắc quyết định và mỗi lá của cây đại diện cho một dự đoán hoặc một lớp.

#### 1.2.1.2 Chiến Lược Cơ Bản Xây Dựng Cây Quyết Định

Chiến lược xây dựng cây quyết định bao gồm các bước cơ bản để chọn thuộc tính phân chia và xây dựng cây từ đầu đến lá. Dưới đây là chiến lược cơ bản:

* Chọn Thuộc Tính Phân Chia (Splitting Attribute):

Mục Tiêu: Tìm thuộc tính tốt nhất để phân chia tập dữ liệu hiện tại thành các tập con.

Phương Pháp: Sử dụng các phép đo (measure) như Information Gain, Gini Impurity, hoặc Chi-square để đánh giá tính tốt của mỗi thuộc tính để phân chia.

* Tách Rẽ (Splitting):

Tách Dữ Liệu: Tách tập dữ liệu hiện tại thành các tập con dựa trên giá trị của thuộc tính phân chia được chọn.

Lặp Lại: Lặp lại quá trình trên từng nhóm con đến khi đạt được điều kiện dừng hoặc không thể tách tiếp.

* Điều Kiện Dừng (Stopping Criteria):

Các Điều Kiện Dừng Thường Gặp: Số lượng mẫu tối thiểu trong mỗi nhóm con, độ sâu tối đa của cây, không còn thuộc tính nào để phân chia.

* Xác Định Dự Đoán (Prediction):

Dự Đoán ở Lá: Xác định giá trị dự đoán tại mỗi lá của cây, thường là dự đoán phân lớp hoặc giá trị trung bình của các mẫu trong lá.

* Kiểm Soát Overfitting:

Cắt tỉa (Pruning): Loại bỏ các nhánh không cần thiết của cây để giảm overfitting.

Thiết Lập Độ Sâu Tối Đa: Hạn chế độ sâu của cây để tránh việc quá mức hóa.

* Kiểm Tra và Đánh Giá:

Kiểm Tra Hiệu Suất: Sử dụng tập kiểm tra để đánh giá hiệu suất của cây quyết định và xác định liệu cây có bị overfitting hay không.

* Tối Ưu Hóa:

Tinh chỉnh siêu tham số: Tinh chỉnh các siêu tham số của cây như số lượng lá, chiều sâu, hoặc tiêu chí phân chia để tối ưu hóa hiệu suất.

#### Thuận Lợi và Hạn Chế của Mô Hình Cây Quyết Định

* Thuận Lợi:
* Dễ hiểu và giải thích: Cây quyết định tạo ra các quy tắc quyết định dễ hiểu và giải thích.
* Xử lý cả dữ liệu số và dữ liệu phân loại: Cây quyết định có thể xử lý cả dữ liệu dạng số và dữ liệu dạng phân loại.
* Ít yêu cầu tiền xử lý: Không cần phải chuẩn hóa hoặc chuẩn bị dữ liệu quá nhiều trước khi sử dụng.
* Hạn Chế:
* Dễ bị overfitting: Cây quyết định có thể dễ dàng bị overfitting trên tập dữ liệu huấn luyện nếu không được kiểm soát.
* Không ổn định: Những thay đổi nhỏ trong dữ liệu đầu vào có thể dẫn đến những cây quyết định khác nhau.
* Không phù hợp cho dữ liệu không gian lớn: Không phù hợp khi dữ liệu có số lượng biến lớn và có nhiều giá trị bị thiếu.

- Đó là một tổng quan về cây quyết định, bao gồm định nghĩa, chiến lược xây dựng và ưu nhược điểm của mô hình này. Đây là một công cụ hữu ích trong học máy với khả năng tạo ra các quy tắc quyết định dễ hiểu và áp dụng rộng rãi trong nhiều lĩnh vực.

#### 1.2.1.4 Thuật Toán ID3

* **Thuật Toán**

Thuật toán ID3 (Iterative Dichotomiser 3) là một thuật toán xây dựng cây quyết định được phát triển bởi Ross Quinlan vào những năm 1980. Thuật toán này sử dụng phép đo Entropy và lượng thông tin (Information Gain) để chọn thuộc tính phân chia tốt nhất tại mỗi bước.

* + Độ Đo Tính Thuần Nhất

1. Entropy Đo Tính Thuần Nhất của Tập Huấn Luyện

Entropy được sử dụng để đo lường mức độ không chắc chắn trong tập dữ liệu. Entropy được tính bằng công thức:

*Entropy*(*S*)=−∑*i*=1*c*​*pi*​log2​(*pi*​)

Trong đó:

* *S* là tập dữ liệu.
* *c* là số lớp trong tập dữ liệu.
* *pi*​ là tỷ lệ của lớp thứ *i* trong tập dữ liệu.

b. Lượng Thông Tin Thu Được Đo Mức Độ Giảm Entropy Mong Đợi

Lượng thông tin (Information Gain) đo lường mức độ giảm Entropy mong đợi khi một thuộc tính được sử dụng để phân chia tập dữ liệu. Lượng thông tin được tính bằng công thức: *IG*(*S*,*A*)=*Entropy*(*S*)−∑*v*∈*Values*(*A*)​∣*S*∣∣*Sv*​∣​⋅*Entropy*(*Sv*​)

Trong đó:

* *S* là tập dữ liệu.
* *A* là thuộc tính được chọn để phân chia.
* *Values*(*A*) là tập hợp các giá trị có thể của thuộc tính *A*.
* *Sv*​ là tập con của *S* chứa các mẫu có giá trị của thuộc tính *A* là *v*.
  + Tìm Kiếm Không Gian Giả Thuyết trong ID3...

Trong thuật toán ID3, không gian giả thuyết là không gian của tất cả các cây quyết định có thể có dựa trên tập dữ liệu huấn luyện và các thuộc tính có sẵn. Quá trình tìm kiếm không gian giả thuyết bao gồm việc xác định tất cả các thuộc tính có thể được sử dụng để phân chia tập dữ liệu và xác định thuộc tính nào là tốt nhất để phân chia tại mỗi bước.

Thuật toán ID3 lựa chọn thuộc tính tốt nhất cho việc phân chia bằng cách tính toán lượng thông tin (Information Gain) hoặc giảm Entropy mong đợi khi sử dụng mỗi thuộc tính. Sau đó, thuật toán chọn thuộc tính có Information Gain cao nhất hoặc giảm Entropy lớn nhất để phân chia tập dữ liệu hiện tại.

* + Đánh Giá Hiệu Suất của Cây Quyết Định:

Để đánh giá hiệu suất của cây quyết định, chúng ta thường sử dụng các phương pháp đánh giá như độ chính xác (accuracy), độ phủ (recall), độ chính xác tiên nghiệm (precision), và F1-score trên tập kiểm tra hoặc sử dụng kỹ thuật cross-validation để đánh giá trên nhiều tập dữ liệu khác nhau. Điều này giúp đảm bảo rằng cây quyết định không chỉ hoạt động tốt trên tập dữ liệu huấn luyện mà còn tổng quát hóa tốt trên dữ liệu mới.

#### 1.2.1.5 Thuật Toán C4.5

Thuật toán C4.5 là một biến thể cải tiến của ID3, được Ross Quinlan phát triển sau đó. C4.5 giải quyết một số vấn đề của ID3 như xử lý dữ liệu không hoàn chỉnh và giảm overfitting bằng cách sử dụng kỹ thuật cắt tỉa (pruning) sau khi cây quyết định đã được xây dựng.

C4.5 sử dụng lượng thông tin (Information Gain) hoặc tỷ lệ lợi ích của thuộc tính (Gain Ratio) để lựa chọn thuộc tính phân chia tốt nhất. Ngoài ra, C4.5 có khả năng xử lý dữ liệu không hoàn chỉnh bằng cách tạo ra các nút dự đoán thay vì bỏ qua các mẫu có dữ liệu bị thiếu.

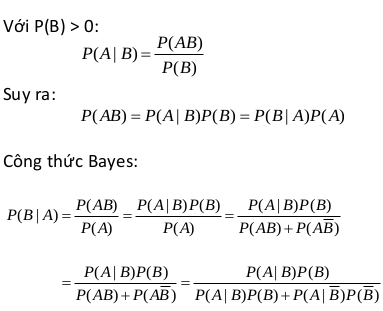
Điều này làm cho C4.5 trở thành một trong những thuật toán cây quyết định phổ biến và mạnh mẽ trong học máy.Top of Form

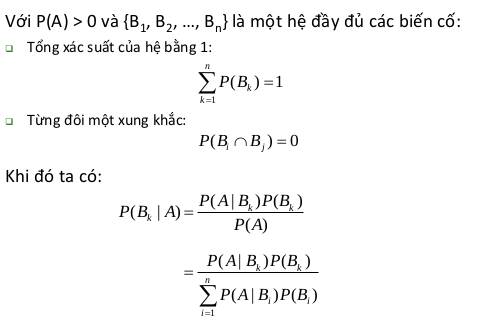
### 1.2.2. Thuật toán phân lớp Naïve Bayes

  Bộ phân lớp Bayes là một giải thuật thuộc lớp giải thuật thống kê, nó có thể dự đoán xác suất của một phần tử dữ liệu thuộc vào một lớp là bao nhiêu. Phân lớp Bayes được dựa trên định lý Bayes (định lý được đặt theo tên tác giả của nó là Thomas Bayes)

#### 1.2.2.1. Định lý Bayes

Gọi A, B là hai biến cố



Công thức Bayes tổng quát

    Trong đó ta gọi A là một chứng cứ (evidence) (trong bài toán phân lớp A sẽ là một phần tử dữ liệu), B là một giả thiết nào để cho A thuộc về một lớp C nào đó. Trong bài toán phân lớp chúng ta muốn xác định giá trị P(B/A) là xác suất để giả thiết B là đúng với chứng cứ A thuộc vào lớp C với điều kiện ra đã biết các thông tin mô tả A. P(B|A) là một xác suất hậu nghiệm (posterior probability hay posteriori probability) của B với điều kiện A.  
    Giả sử tập dữ liệu liệu khách hàng của chúng ta được mô tả bởi các thuộc tính tuổi và thu nhập, và một khách hàng X có tuổi là 25 và thu nhập là 2000$. Giả sử H là giả thiết khách hàng đõ sẽ mua máy tính, thì P(H|X) phản ánh xác xuất người dùng X sẽ mua máy tính với điều kiện ta biết.  
    Ngược lại P(H) là xác suất tiền nghiệm (prior probability hay priori probability) của H. Trong ví dụ trên, nó là xác suất một khách hàng sẽ mua máy tính mà không cần biết các thông tin về tuổi hay thu nhập của họ. Hay nói cách khác, xác suất này không phụ thuộc vào yếu tố X. Tương tự, P(X|H) là xác suất của X với điều kiện H (likelihood), nó là một xác suất hậu nghiệm. VÍ dụ, nó là xác suất người dùng X (có tuổi là 25 và thu nhập là $200) sẽ mua máy tính với điều kiện ta đã biết người đó sẽ mua máy tính. Cuối cùng P(X) là xác suất tiền nghiệm của X. Trong ví dụ trên, nó se là xác xuất một người trong tập dữ liệu sẽ có tuổi 25 và thu nhập $2000.  
**Posterior = Likelihood \* Prior / Evidence**

#### 1.2.2.2 Phân lớp Naive Bayes

    Bộ phân lớp Naive bayes hay bộ phân lớp Bayes (simple byes classifier) hoạt động như sau:

1. Gọi D là tập dữ liệu huấn luyện, trong đó mỗi phần tử dữ liệu X được biểu diễn bằng một vector chứa n giá trị thuộc tính A1, A2,...,An = {x1,x2,...,xn}
2. Giả sử có m lớp C1, C2,..,Cm. Cho một phần tử dữ liệu X, bộ phân lớp sẽ gán nhãn cho X là lớp có xác suất hậu nghiệm lớn nhất. Cụ thể, bộ phân lớp Bayes sẽ dự đoán X thuộc vào lớp Ci nếu và chỉ nếu: **P(Ci|X) > P(Cj|X) (1<= i, j <=m, i != j)**  
   Giá trị này sẽ tính dựa trên định lý Bayes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Tuổi** | **Thu nhập** | **Sinh Viên** | **Đánh giá**  **tín dụng** | **Mua máy tính** |
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle | low | yes | excellent | yes |
| 8 | youth | medum | no | fair | yes |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle | medium | no | excellent | yes |
| 13 | middle | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

1. Để tìm xác suất lớn nhất, ta nhận thấy các giá trị P(X) là giống nhau với mọi lớp nên không cần tính. Do đó ta chỉ cần tìm giá trị lớn nhất của P(X|Ci) \* P(Ci). Chú ý rằng P(Ci)
2. được ước lượng bằng |Di|/|D|, trong đó Di là tập các phần tử dữ liệu thuộc lớp Ci. Nếu xác suất tiền nghiệm P(Ci) cũng không xác định được thì ta coi chúng bằng nhau P(C1) = P(C2) = ... = P(Cm), khi đó ta chỉ cần tìm giá trị P(X|Ci) lớn nhất.
3. Khi số lượng các thuộc tính mô tả dữ liệu là lớn thì chi phí tính toàn P(X|Ci) là rất lớn, dó đó có thể giảm độ phức tạp của thuật toán Naive Bayes giả thiết các thuộc tính độc lập nhau. Khi đó ta có thể tính:  
   **P(X|Ci) = P(x1|Ci)...P(xn|Ci)**

Ví dụ 1:

    Phân các bệnh nhân thành 2 lớp ung thư và không ung thư. Giả sử xác suất để một người bị ung thư là 0.008 tức là P(cancer) = 0.008; và P(nocancer) = 0.992. Xác suất để bệnh nhân ung thư có kết quả xét nghiệm dương tính là 0.98 và xác suất để bệnh nhân không ung thư có kết quả dương tính là 0.03 tức là P(+/cancer) = 0.98, P(+/nocancer) = 0.03. Bây giờ giả sử một bệnh nhân có kết quả xét nghiệm dương tính. Ta có:  
P(+/canncer)P(cancer) = 0.98 \* 0.008 = 0.0078  
P(+/nocancer)P(nocancer) = 0.03 \* 0.992 = 0.0298  
Như vậy, P(+/nocancer)P(nocancer) >> P(+/cancer)P(cancer).  
Do đó ta xét đoán rằng, bệnh nhân là không ung thư.

Ví dụ 2:

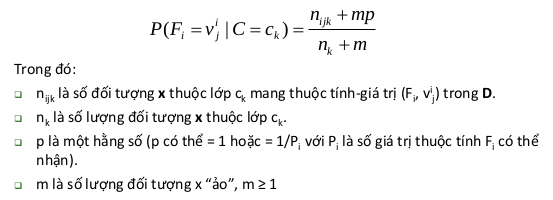
Cơ sở dữ liệu khách hàng:

    Giả sử ta có một khách hàng mới X có các thuộc tính  
X = (age = youth, income = medium, student = yes, credit\_rating = fair)  
Bây giớ cần xác định xem khách hàng X có thuộc lớp Cyes (mua máy tính) hay không, ta tính toán như sau:  
P(Cyes) = 9/14 = 0.357  
Các xác suất thành phần:  
P(age = youth|Cyes) = 2/9 = 0.222  
P(age = youth|Cno) = 3/5 = 0.6  
P(income = medium|Cyes) = 4/9 = 0.444  
P(income = medium|Cno) = 2/5 = 0.4  
P(student = yes|Cyes) = 6/9 = 0.667  
P(student = yes|Cno) = 1/5 = 0.2  
P(credit\_rating = fair|Cyes) = 6/9 = 0.667  
P(credit\_rating = fair|Cno) = 2/5 = 0.2  
Cuối cùng:  
P(X|Cyes) = 0.222 \* 0.444 \* 0.667 \* 0.667 = 0.044  
P(X|Cno) = 0.60.4 \* 0.2 \* 0.4 = 0.019  
P(X|Cyes)\*P(Cyes) = 0.044 \* 0.643  
P(X|Cno)\*P(Cno) =0.019 \* 0.357 = 0.007

    Từ kết quả này ta thấy P(X |Cyes)P(Cyes) có giá trị lớn nhất, do đó thuật toán Bayes sẽ kết luận là khách hàng X sẽ mua máy tính.

#### 1.2.2.3 Khắc phục vấn đề xác suất điều kiện bằng zero

1. Nếu trong dữ liệu huấn luyện không có đối tượng X nào có thuộc tính lớp Ck có thuộc tính Fi nhận một giá trị cụ thể vij, xác suất điều kiện P(Fi = xij | Ck) sẽ bằng 0.
2. Khi phân lớp, nếu có một đối tượng nào mang thuộc tính này thì xác suất phân vào lớp Ck luôn bằng 0.
3. Khắc phục bằng cách ước lượng theo công thức sau:



#### Ưu điểm

* Giả định độc lập: hoạt động tốt cho nhiều bài toán/miền sữ liệu và ứng dụng.  
  Đơn giản nhưng đủ tốt để giải quyết nhiều bài toán như phân lớp văn bản, lọc spam,..
* Cho phép kết hợp tri thức tiền nghiệm (prior knowledge) và dữ liệu quan sát được (obserwed data).  
  Tốt khi có sự chệnh lệch số lượng giữa các lớp phân loại.
* Huấn luyện mô hình (ước lượng tham số) dễ và nhanh.

#### 1.2.2.5 Nhược điểm

* Giả định độc lập (ưu điểm cũng chính là nhược điểm)  
  hầu hết các trường hợp thực tế trong đó có các thuộc tính trong các đối tượng thường phụ thuộc lẫn nhau.
* Vấn đề zero (đã nêu cách giải quyết ở phía trên)
* Mô hình không được huẩn luyện bằng phượng pháp tối ưu mạnh và chặt chẽ.  
  Tham số mủa mô hình là các ước lượng xác suất điều kiện đơn lẻ.  
  Không tính đến sự tương tác giữa các ước lượng này.

### 1.2.3 Thuật toán Hồi quy Logistic (Logistic Regression)

#### 1.2.3.1. Giới thiệu

1. Mô hình Logistic Regression

Đầu ra dự đoán của:

* Linear Regression: f(x)=wTx
* PLA: f(x) = sgn(wTx)

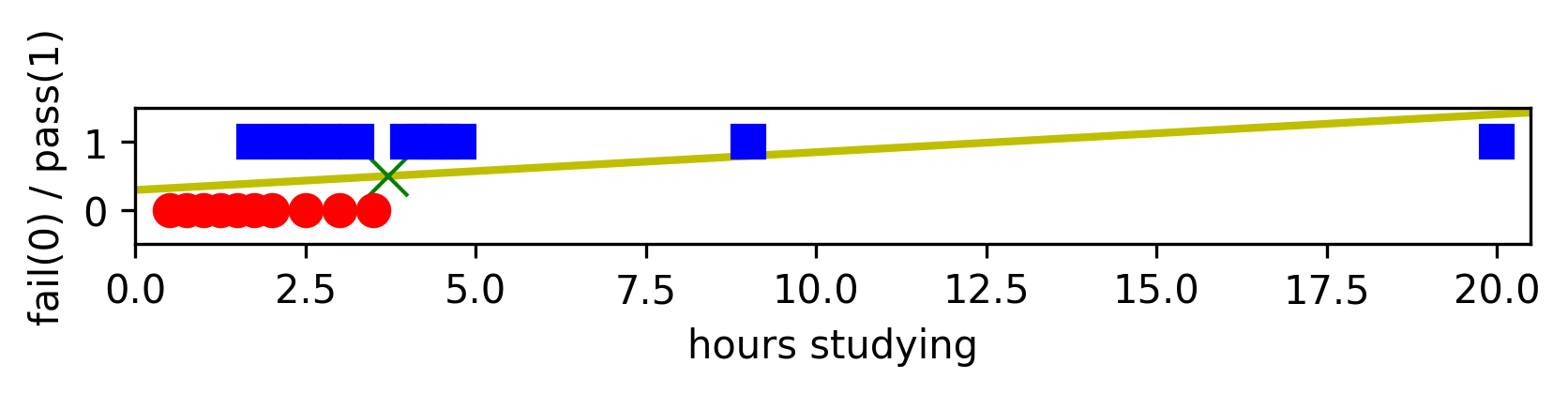
Đầu ra dự đoán của logistic regression thường được viết chung dưới dạng:f(x)=θ(wTx)

Trong đó θ được gọi là logistic function. Một số activation cho mô hình tuyến tính được cho trong hình dưới đây:



Hinh1.1Một số activation cho mô hình tuyến tính

* Đường màu vàng biểu diễn linear regression. Đường này không bị chặn nên không phù hợp cho bài toán này. Có một trick nhỏ để đưa nó về dạng bị chặn: cắt phần nhỏ hơn 0 bằng cách cho chúng bằng 0, cắt các phần lớn hơn 1 bằng cách cho chúng bằng 1. Sau đó lấy điểm trên đường thẳng này có tung độ bằng 0.5 làm điểm phân chia hai class, đây cũng không phải là một lựa chọn tốt. Giả sử có thêm vài bạn sinh viên tiêu biểu ôn tập đến 20 giờ và, tất nhiên, thi đỗ. Khi áp dụng mô hình linear regression như hình dưới đây và lấy mốc 0.5 để phân lớp, toàn bộ sinh viên thi trượt vẫn được dự đoán là trượt, nhưng rất nhiều sinh viên thi đỗ cũng được dự đoán là trượt (nếu ta coi điểm x màu xanh lục là ngưỡng cứng để đưa ra kết luận). Rõ ràng đây là một mô hình không tốt. Anh chàng sinh viên tiêu biểu này đã kéo theo rất nhiều bạn khác bị trượt.



Hinh 1.2 Tại sao Linear Regression không phù hợp?

* Đường màu đỏ (chỉ khác với activation function của PLA ở chỗ hai class là 0 và 1 thay vì -1 và 1) cũng thuộc dạng ngưỡng cứng (hard threshold). PLA không hoạt động trong bài toán này vì dữ liệu đã cho không linearly separable.
* Các đường màu xanh lam và xanh lục phù hợp với bài toán của chúng ta hơn. Chúng có một vài tính chất quan trọng sau:
  + Là hàm số liên tục nhận giá trị thực, bị chặn trong khoảng (0,1)(0,1).
  + Nếu coi điểm có tung độ là 1/2 làm điểm phân chia thì các điểm càng xa điểm này về phía bên trái có giá trị càng gần 0. Ngược lại, các điểm càng xa điểm này về phía phải có giá trị càng gần 1. Điều này khớp với nhận xét rằng học càng nhiều thì xác suất đỗ càng cao và ngược lại.
  + Mượt (smooth) nên có đạo hàm mọi nơi, có thể được lợi trong việc tối ưu.

1. Sigmoid function

Trong số các hàm số có 3 tính chất nói trên thì hàm sigmoid:

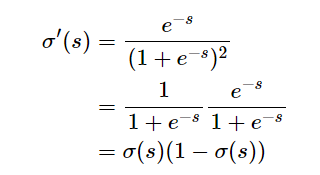


được sử dụng nhiều nhất, vì nó bị chặn trong khoảng

(0,1). Thêm nữa:

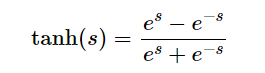


Đặc biệt hơn nữa:



Công thức đạo hàm đơn giản thế này giúp hàm số này được sử dụng rộng rãi. Ở phần sau, tôi sẽ lý giải việc người ta đã tìm ra hàm số đặc biệt này như thế nào.

Ngoài ra, hàm tanh cũng hay được sử dụng:



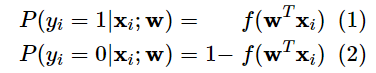
Hàm số này nhận giá trị trong khoảng (−1,1)(−1,1) nhưng có thể dễ dàng đưa nó về khoảng (0,1)(0,1). Bạn đọc có thể chứng minh được:



#### 1.2.3.2. Hàm mất mát và phương pháp tối ưu

1. Xây dựng hàm mất mát

Với mô hình như trên (các activation màu xanh lam và lục), ta có thể giả sử rằng xác suất để một điểm dữ liệu x rơi vào class 1 là f(wTx) và rơi vào class 0 là 1−f(wTx). Với mô hình được giả sử như vậy, với các điểm dữ liệu training (đã biết đầu ra y), ta có thể viết như sau:



trong đó P(yi=1|xi; w) ược hiểu là xác suất xảy ra sự kiện đầu ra yi=1 khi biết tham số mô hình w và dữ liệu đầu vào xi. Mục đích của chúng ta là tìm các hệ số w sao cho f(wTxi) càng gần với 1 càng tốt với các điểm dữ liệu thuộc class 1 và càng gần với 0 càng tốt với những điểm thuộc class 0.

Ký hiệu zi=f(wTxi) và viết gộp lại hai biểu thức bên trên ta có:



Biểu thức này là tương đương với hai biểu thức (1)(1) và (2)(2) ở trên vì khi yi=1, phần thứ hai của vế phải sẽ triệt tiêu, khi yi=0, phần thứ nhất sẽ bị triệt tiêu! Chúng ta muốn mô hình gần với dữ liệu đã cho nhất, tức xác suất này đạt giá trị cao nhất.

Xét toàn bộ training set với X=[x1,x2,…,xN]∈Rd×N và y=[y1,y2,…,yN], chúng ta cần tìm w để biểu thức sau đây đạt giá trị lớn nhất:

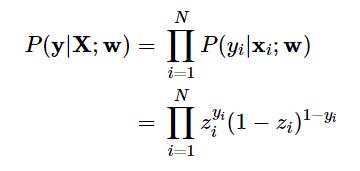


Ở đây, ta cũng ký hiệu X, y như các biến ngẫu nhiên (random variables). Nói cách khác:

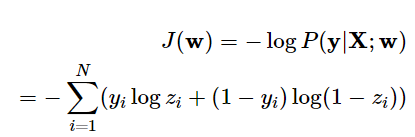


Bài toán tìm tham số để mô hình gần với dữ liệu nhất trên đây có tên gọi chung là bài toán maximum likelihood estimation với hàm số phía sau argmax được gọi là likelihood function. Khi làm việc với các bài toán Machine Learning sử dụng các mô hình xác suất thống kê, chúng ta sẽ gặp lại các bài toán thuộc dạng này, hoặc maximum a posteriori estimation, rất nhiều. Tôi sẽ dành 1 bài khác để nói về hai dạng bài toán này.

Giả sử thêm rằng các điểm dữ liệu được sinh ra một cách ngẫu nhiên độc lập với nhau (independent), ta có thể viết:



Trực tiếp tối ưu hàm số này theo w nhìn qua không đơn giản! Hơn nữa, khi N lớn, tích của N số nhỏ hơn 1 có thể dẫn tới sai số trong tính toán (numerial error) vì tích là một số quá nhỏ. Một phương pháp thường được sử dụng đó là lấy logarit tự nhiên (cơ số e) của *likelihood function* biến phép nhân thành phép cộng và để tránh việc số quá nhỏ. Sau đó lấy ngược dấu để được một hàm và coi nó là hàm mất mát. Lúc này bài toán tìm giá trị lớn nhất (maximum likelihood) trở thành bài toán tìm giá trị nhỏ nhất của hàm mất mát (hàm này còn được gọi là negative log likelihood):



với chú ý rằng zi là một hàm số của w. Bạn đọc tạm nhớ biểu thức vế phải có tên gọi là *cross entropy*, thường được sử dụng để đo *khoảng cách* giữa hai phân phối (distributions). Trong bài toán đang xét, một phân phối là dữ liệu được cho, với xác suất chỉ là 0 hoặc 1; phân phối còn lại được tính theo mô hình logistic regression. *Khoảng cách* giữa hai phân phối nhỏ đồng nghĩa với việc (*có vẻ hiển nhiên là*) hai phân phối đó rất gần nhau. Tính chất cụ thể của hàm số này sẽ được đề cập trong một bài khác mà tầm quan trọng của khoảng cách giữa hai phân phối là lớn hơn.

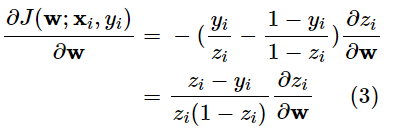
**Chú ý:** Trong machine learning, logarit thập phân ít được dùng, vì vậy loglog thường được dùng để ký hiệu logarit tự nhiên.

1. Tối ưu hàm mất mát

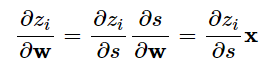
Hàm mất mát với chỉ một điểm dữ liệu (xi,yi) là:



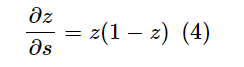
Với đạo hàm:



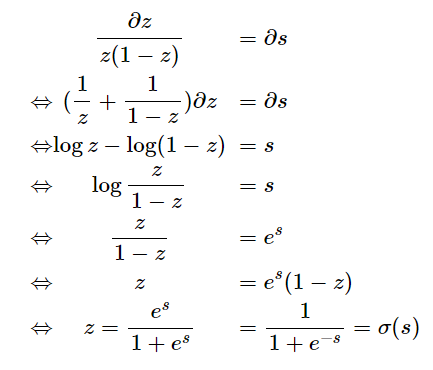
Để cho biểu thức này trở nên *gọn* và *đẹp* hơn, chúng ta sẽ tìm hàm z=f(wTx) sao cho mẫu số bị triệt tiêu. Nếu đặt s=wTx, chúng ta sẽ có:



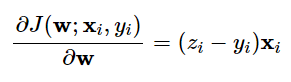
Một cách trực quan nhất, ta sẽ tìm hàm số z = f(s) sao cho:



để triệt tiêu mẫu số trong biểu thức (3)(3). Chúng ta cùng khởi động một chút với phương trình vi phân đơn giản này. Phương trình (4)(4) tương đương với:



1. Công thức cập nhật cho logistic sigmoid regression



#### 1.2.3.3. Một vài tính chất của Logistic Regression

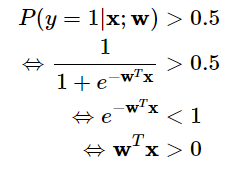
1. Logistic Regression thực ra được sử dụng nhiều trong các bài toán Classification.

Mặc dù có tên là Regression, tức một mô hình cho fitting, Logistic Regression lại được sử dụng nhiều trong các bài toán Classification. Sau khi tìm được mô hình, việc xác định class y cho một điểm dữ liệu x được xác định bằng việc so sánh hai biểu thức xác suất:



Nếu biểu thức thứ nhất lớn hơn thì ta kết luận điểm dữ liệu thuộc class 1, ngược lại thì nó thuộc class 0. Vì tổng hai biểu thức này luôn bằng 1 nên một cách gọn hơn, ta chỉ cần xác định xem P(y=1|x;w) lớn hơn 0.5 hay không. Nếu có, class 1. Nếu không, class 0.

1. Boundary tạo bởi Logistic Regression có dạng tuyến tính



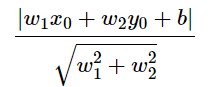
Nói cách khác, boundary giữa hai class là đường có phương trình wTx. Đây chính là phương trình của một siêu mặt phẳng. Vậy Logistic Regression tạo ra boundary có dạng tuyến tính.

### 1.2.4. Thuật toán SVM (Support Vector Machine)

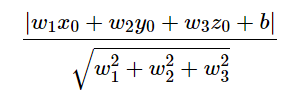
#### 1.2.4.1. Giới thiệu

1. Khoảng cách từ một điểm tới một siêu mặt phẳng

Trong không gian 2 chiều, ta biết rằng khoảng cách từ một điểm có toạ độ (x0,y0) tới *đường thẳng* có phương trình w1x+w2y+b=0 được xác định bởi:

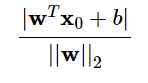


Trong không gian ba chiều, khoảng cách từ một điểm có toạ độ (x0,y0,z0) tới một *mặt phẳng* có phương trình w1x+w2y+w3z+b=0 được xác định bởi:



Hơn nữa, nếu ta bỏ dấu trị tuyệt đối ở tử số, chúng ta có thể xác định được điểm đó nằm về phía nào của *đường thẳng* hay *mặt phẳng* đang xét. Những điểm làm cho biểu thức trong dấu giá trị tuyệt đối mang dấu dương nằm về cùng 1 phía (tôi tạm gọi đây là *phía dương* của đường thẳng), những điểm làm cho biểu thức trong dấu giá trị tuyệt đối mang dấu âm nằm về phía còn lại (tôi gọ là *phía âm*). Những điểm nằm trên *đường thẳng*/*măt phẳng* sẽ làm cho tử số có giá trị bằng 0, tức khoảng cách bằng 0.

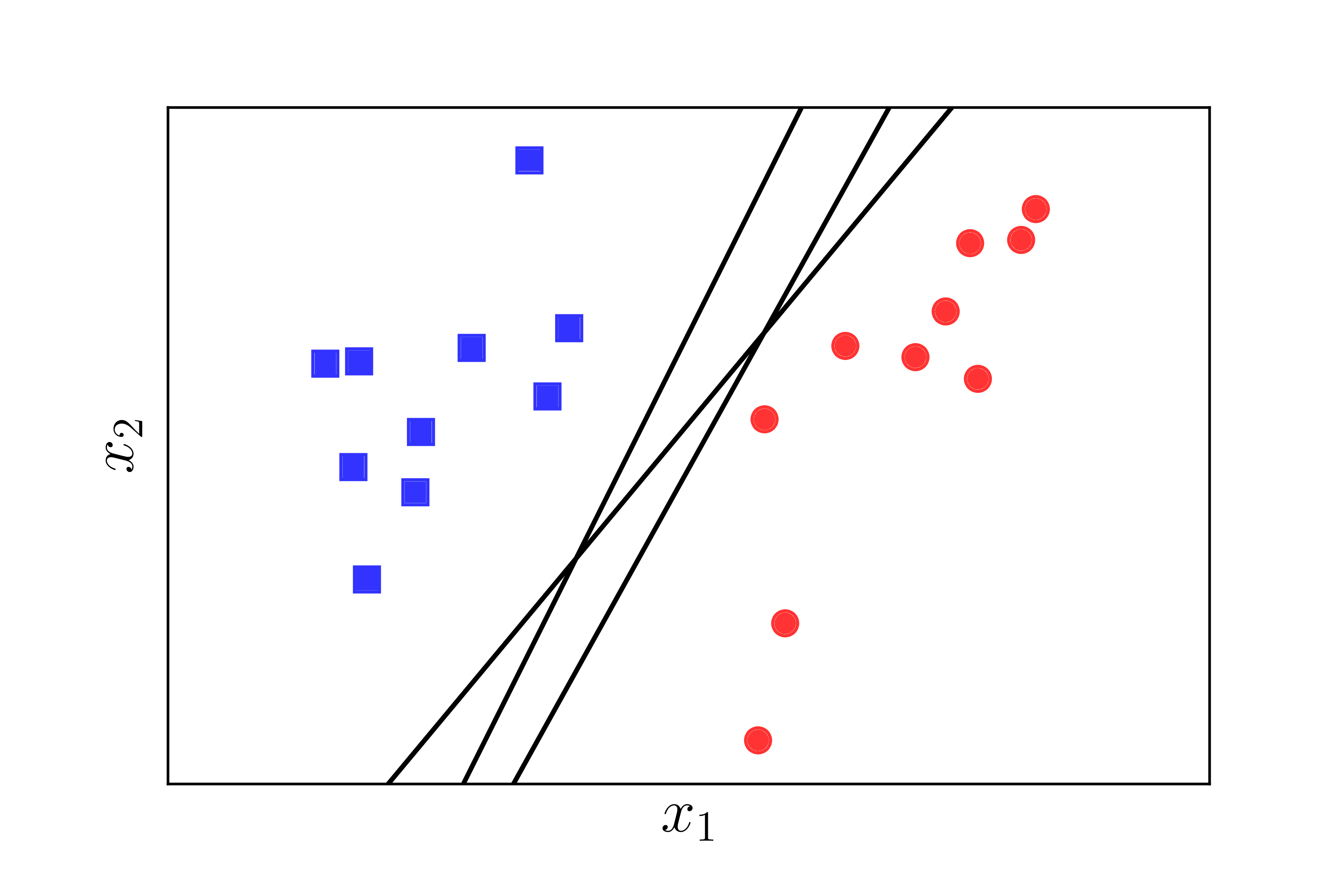
Việc này có thể được tổng quát lên không gian nhiều chiều: Khoảng cách từ một điểm (vector) có toạ độ x0 tới *siêu mặt phẳng* (*hyperplane*) có phương trình wTx+b=0 được xác định bởi:



Với ||w||2=√∑di=1w2ivới d là số chiều của không gian.

1. Nhắc lại bài toán phân chia hai classes

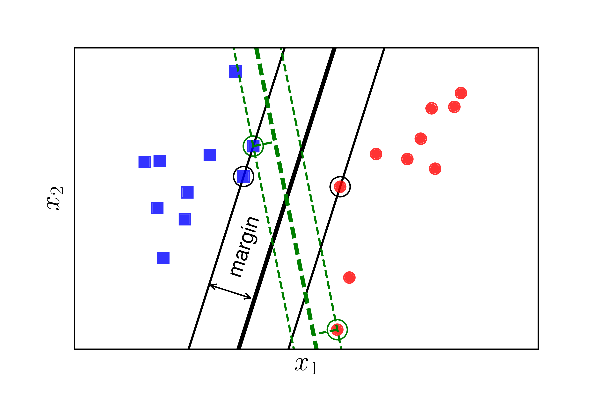
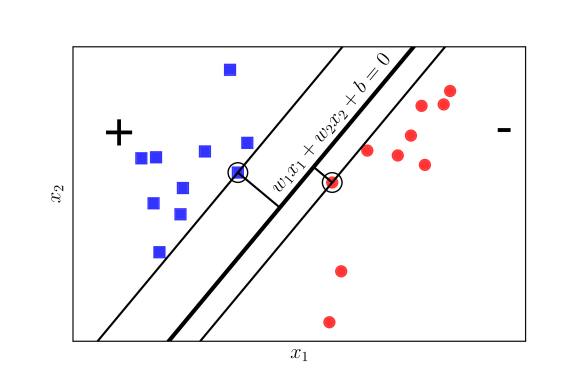
Chúng ta cùng quay lại với bài toán trong Perceptron Learning Algorithm (PLA). Giả sử rằng có hai class khác nhau được mô tả bởi các điểm trong không gian nhiều chiều, hai classes này linearly separable, tức tồn tại một siêu phẳng phân chia chính xác hai classes đó. Hãy tìm một siêu mặt phẳng phân chia hai classes đó, tức tất cả các điểm thuộc một class nằm về cùng một phía của siêu mặt phẳng đó và ngược phía với toàn bộ các điểm thuộc class còn lại. Chúng ta đã biết rằng, thuật toán PLA có thể làm được việc này nhưng nó có thể cho chúng ta vô số nghiệm như Hình 1.3 dưới đây:



Hinh 1.3 Các mặt phân cách hai classes linearly separable.

Câu hỏi đặt ra là: trong vô số các mặt phân chia đó, đâu là mặt phân chia tốt nhất *theo một tiêu chuẩn nào đó*? Trong ba đường thẳng minh họa trong Hình 1 phía trên, có hai đường thẳng khá *lệch* về phía class hình tròn đỏ. Điều này có thể khiến cho lớp màu đỏ *không vui vì lãnh thổ xem ra bị lấn nhiều quá*. Liệu có cách nào để tìm được đường phân chia mà cả hai classes đều cảm thấy *công bằng* và *hạnh phúc* nhất hay không?

Chúng ta cần tìm một tiêu chuẩn để đo sự *hạnh phúc* của mỗi class. Hãy xem Hình 2 dưới đây:



Hinh 1.4 Margin của hai classes là bằng nhau và lớn nhất có thể.

Nếu ta định nghĩa *mức độ hạnh phúc* của một class tỉ lệ thuận với khoảng cách gần nhất từ một điểm của class đó tới đường/mặt phân chia, thì ở Hình 2 trái, class tròn đỏ sẽ *không được hạnh phúc cho lắm* vì đường phân chia gần nó hơn class vuông xanh rất nhiều. Chúng ta cần một đường phân chia sao cho khoảng cách từ điểm gần nhất của mỗi class (các điểm được khoanh tròn) tới đường phân chia là như nhau, như thế thì mới *công bằng*. Khoảng cách như nhau này được gọi là *margin* (*lề*).

Đã có *công bằng* rồi, chúng ta cần *văn minh* nữa. *Công bằng* mà cả hai đều *kém hạnh phúc như nhau* thì chưa phải là *văn mình* cho lắm.

Chúng ta xét tiếp Hình 2 bên phải khi khoảng cách từ đường phân chia tới các điểm gần nhất của mỗi class là như nhau. Xét hai cách phân chia bởi đường nét liền màu đen và đường nét đứt màu lục, đường nào sẽ làm cho cả hai class *hạnh phúc hơn*? Rõ ràng đó phải là đường nét liền màu đen vì nó tạo ra một *margin* rộng hơn.

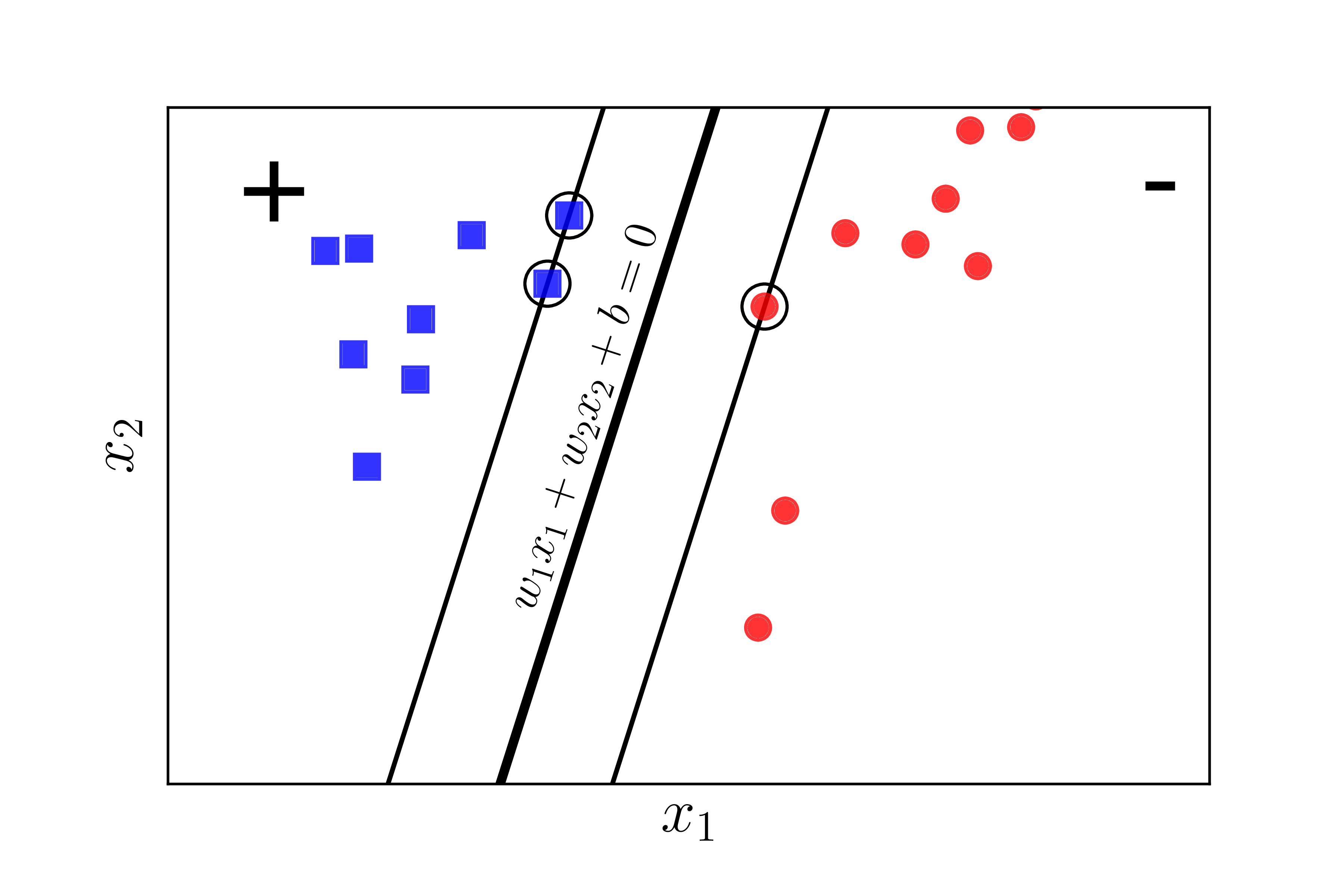
Việc *margin* rộng hơn sẽ mang lại hiệu ứng phân lớp tốt hơn vì *sự phân chia giữa hai classes là rạch ròi hơn*. Việc này, sau này các bạn sẽ thấy, là một điểm khá quan trọng giúp *Support Vector Machine* mang lại kết quả phân loại tốt hơn so với *Neural Network với 1 layer*, tức Perceptron Learning Algorithm.

Bài toán tối ưu trong *Support Vector Machine* (SVM) chính là bài toán đi tìm đường phân chia sao cho *margin* là lớn nhất. Đây cũng là lý do vì sao SVM còn được gọi là *Maximum Margin Classifier*. Nguồn gốc của tên gọi Support Vector Machine sẽ sớm được làm sáng tỏ.

#### 1.2.4.1. Xây dựng bài toán tối ưu cho SVM

Giả sử rằng các cặp dữ liệu của *training set* là (x1, y1),(x2,y2),…,(xN,yN) với vector xi∈Rd thể hiện *đầu vào* của một điểm dữ liệu và yi là *nhãn* của điểm dữ liệu đó. d là số chiều của dữ liệu và N là số điểm dữ liệu. Giả sử rằng *nhãn* của mỗi điểm dữ liệu được xác định bởi yi =1 (class 1) hoặc yi =−1 (class 2) giống như trong PLA.

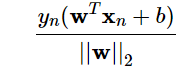
Để giúp các bạn dễ hình dung, chúng ta cùng xét trường hợp trong không gian hai chiều dưới đây. Không gian hai chiều để các bạn dễ hình dung, các phép toán hoàn toàn có thể được tổng quát lên không gian nhiều chiều.



Hinh 1.5 Phân tích bài toán SVM.

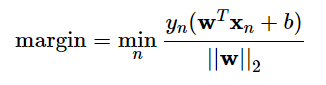
Giả sử rằng các điểm vuông xanh thuộc class 1, các điểm tròn đỏ thuộc class -1 và mặt wTx+b=w1x1+w2x2+b=0 là mặt phân chia giữa hai classes (Hình 3). Hơn nữa, class 1 nằm về *phía dương*, class -1 nằm về *phía âm* của mặt phân chia. Nếu ngược lại, ta chỉ cần đổi dấu của w và b. Chú ý rằng chúng ta cần đi tìm các hệ số w và b.

Ta quan sát thấy một điểm quan trọng sau đây: với cặp dữ liệu (xn,yn) bất kỳ, khoảng cách từ điểm đó tới mặt phân chia là:

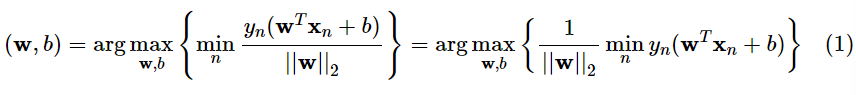


Điều này có thể dễ nhận thấy vì theo giả sử ở trên, yn luôn cùng dấu với *phía* của xn. Từ đó suy ra yn cùng dấu với (wTxn+b), và tử số luôn là 1 số không âm.

Với mặt phần chia như trên, *margin* được tính là khoảng cách gần nhất từ 1 điểm tới mặt đó (bất kể điểm nào trong hai classes):



Bài toán tối ưu trong SVM chính là bài toán tìm w và b sao cho *margin* này đạt giá trị lớn nhất:

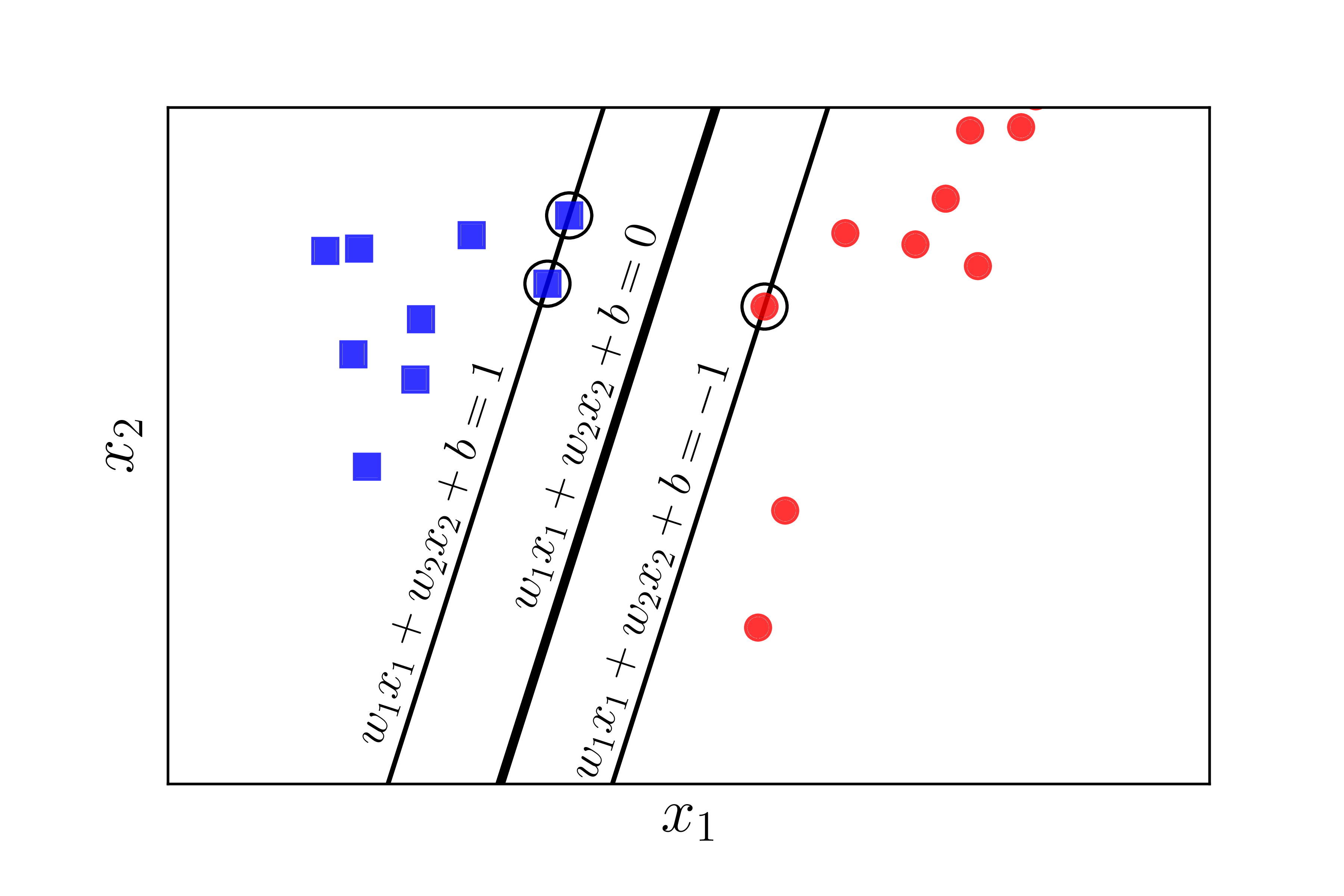


Việc giải trực tiếp bài toán này sẽ rất phức tạp, nhưng các bạn sẽ thấy có cách để đưa nó về bài toán đơn giản hơn.

Nhận xét quan trọng nhất là nếu ta thay vector hệ số w bởi kw và b bởi kb trong đó k là một hằng số dương thì mặt phân chia không thay đổi, tức khoảng cách từ từng điểm đến mặt phân chia không đổi, tức *margin* không đổi. Dựa trên tính chất này, ta có thể giả sử:



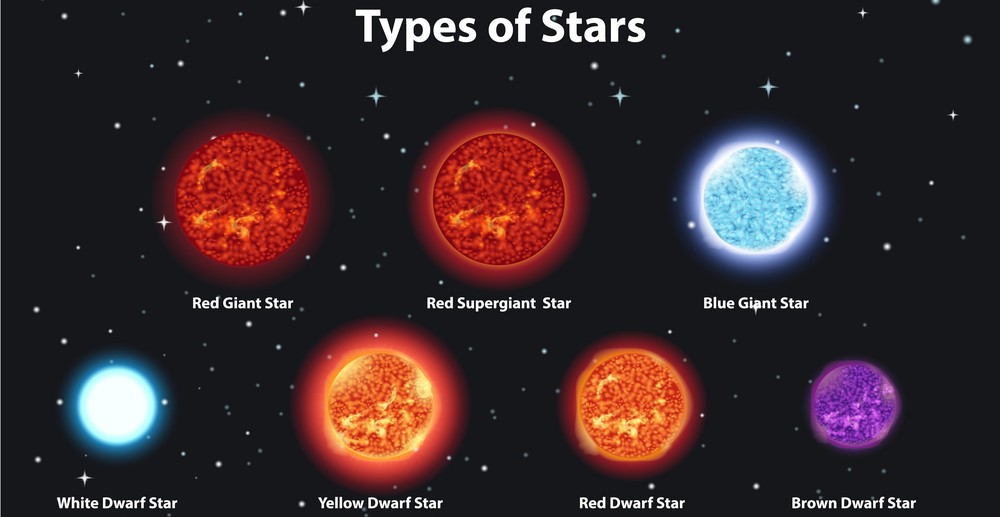
với những điểm nằm gần mặt phân chia nhất như Hình 4 dưới đây:



Hinh 1.6 Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn

# CHƯƠNG 2: GIỚI THIỆU BÀI TOÁN

## 2.1. Hoạt động của hệ thống



Hinh 2.1 Types of star

### Thu thập dữ liệu về các vì sao:

*Xác định nguồn dữ liệu:* Thu thập dữ liệu về các vì sao, bao gồm các hệ thống quan sát thiên văn, cơ sở dữ liệu vì sao ở bộ thiên văn và nguồn Wikipedia.

### Tiền xử lý dữ liệu:

Lọc dữ liệu:

Kiểm tra các giá trị Null và lọc các gia trị Null.

Kiểm tra các giá trị trùng lặp và loại bỏ các giá trị trùng lặp.

Kiểm tra số lượng vì sao.

### Phân tích dữ liệu

#### Mục đích

Mục đích của việc phân tích dữ liệu giúp giải đáp tầm ảnh hưởng của các yếu tố( đặc điểm) củ các vì sao có ảnh hưởng đến việc phân loại các vì sao. Cho thấy việc phân loại các vì sao hoàn toàn phụ thuộc vào các yếu tố trên. Chứng minh các ngôi sao tuân theo một biểu đồ nhất định trong Không gian thiên thể.

#### Phương pháp

Sử dụng biểu đồ hình ảnh để biểu diễn mật độ của các vì sao từ đó tìm ra được yếu tố (đặc điểm) nào ảnh hưởng đến từng vì sao phải phân loại.

#### Phân tích dữ liệu

Kiểm tra mật độ xác xuất của những vì sao dựa trên từng giá trị: Nhiệt độ tuyệt đối (in K), Độ sáng tương đối (L/Lo), Bán kính tương đối (R/Ro), Độ sáng tuyệt đối (Mv), Màu sao (white, Red, Blue, Yellow, yellow-orange etc), Lớp quang phổ (O,B,A,F,G,K,,M)

Kiểm tra mật độ xác xuất của các vì sao dựa vào 2 yếu tố: độ sáng và nhiệt độ , bán kính và nhiệt độ, Dựa vào màu sắc và nhiệt độ, lớp quang phổ và nhiệt độ.

Kiểm tra mật độ xác suất của các vì sao theo toàn bộ 6 yếu tộ trên tập dữ liệu.

* + 1. Xây dựng mô hình phân loại:

a. Chuẩn hóa dữ liệu*:* Mã hóa biến Star color, Spectral Class.

b. Lựa chọn thuật toán

* + - Sử dụng các thuật toán để huấn luyện mô hình như: LogisticRegression, KNN (K-Nearest Neighbors), SVM (Support Vector Machine), RandomForest, DecisionTree, Naïve Bayes.
    - Kiểm tra độ chính xác của thuật toán nào cao nhất.
    - Lựa chọn thuật toán tối ưu nhất.
* *Huấn luyện mô hình*: Dữ liệu được chia thành tập huấn luyện và tập kiểm tra để đánh giá hiệu suất của mô hình.

*- Đánh giá hiệu suất:* Mô hình được đánh giá bằng các phương pháp đánh giá hiệu suất như cross-validation, confusion matrix và các độ đo như accuracy, precision, recall để đảm bảo tính chính xác và hiệu quả của mô hình.

### 2.1.5 Tinh chỉnh mô hình

*a. Tinh chỉnh tham số*: sử dụng các kỹ thuật như grid search, randomized search.

*b. Đánh giá lại mô hình:* đánh giá lại mô hình để đảm bảo rằng hiệu suất đã được cải thiện và đáp ứng được yêu cầu của bài toán phân loại vì sao.

## 2.2. Chi tiết bài toán

### 2.2.1. Dữ liệu đầu vào

a. Nguồn gốc dữ liệu:



Hình 2.2 Dữ liệu đầu vào

Bộ dữ liệu được tạo dựa trên một số phương trình trong vật lý thiên văn. Chúng được đưa ra dưới đây:

* + Định luật bức xạ vật đen của Stefan-Boltzmann (Để tìm độ sáng của một ngôi sao)
  + Định luật dịch chuyển của Wienn (để tìm nhiệt độ bề mặt của ngôi sao bằng bước sóng )
  + Quan hệ độ lớn tuyệt đối
  + Bán kính của một ngôi sao sử dụng thị sai .
  + Tập dữ liệu mất 3 tuần để thu thập được 240 sao, phần lớn được thu thập từ web.
  + Dữ liệu còn thiếu được tính toán thủ công bằng cách sử dụng các phương trình vật lý thiên văn nêu trên.

1. Thông tin tập dữ liệu

Đây là tập dữ liệu bao gồm một số tính năng của các ngôi sao. Có 7 yếu tố trong tập dữ liệu:

* Nhiệt độ tuyệt đối (in K)
* Độ sáng tương đối (L/Lo)
* Bán kính tương đối (R/Ro)
* Cường độ tuyệt đối (Mv)
* Màu sao (white,Red,Blue,Yellow,yellow-orange etc)
* Lớp quang phổ (O,B,A,F,G,K,,M)
* Loại sao (Red Dwarf(0), Brown Dwarf(1), White Dwarf(2), Main Sequence (3), SuperGiants(4), HyperGiants(5))

Lo = 3.828 x 10^26 Watts (Avg Luminosity of Sun)  
Ro = 6.9551 x 10^8 m (Avg Radius of Sun)

### 2.2.2. Dữ liệu đầu ra

* Phân loại các vì sao.
* Người dùng nhập dữ liệu của các yếu tố ( đặc điểm) của 1 vì sao kết quả thu được tên của vì sao.

# CHƯƠNG 3: KẾT QUẢ BÀI TOÁN

## 3.1 Dữ liệu bài toán

Đây là dữ liệu của bài tóa để xử lý dữ liệu. Dữ liệu này gồm 240 bản ghi gồm 7 cột:

Cột Temperate: Nhiệt độ với đơn vị là in K. Nhiệt độ bề mặt của một ngôi sao, thường được gọi là "nhiệt độ hiệu dụng," có thể thay đổi rất nhiều, từ vài nghìn đến hàng chục nghìn Kelvin (K).

Cột Luminoist: Độ sáng (Luminosity - *L*) là một thước đo tổng lượng năng lượng phát ra từ một ngôi sao hoặc đối tượng thiên văn khác trong một đơn vị thời gian. Nó thường được so sánh với độ sáng của Mặt Trời (*L*⊙​), được sử dụng như một điểm tham chiếu tiêu chuẩn. Tỷ lệ giữa độ sáng của một đối tượng so với độ sáng của Mặt Trời được ký hiệu là 𝐿/𝐿⊙​.

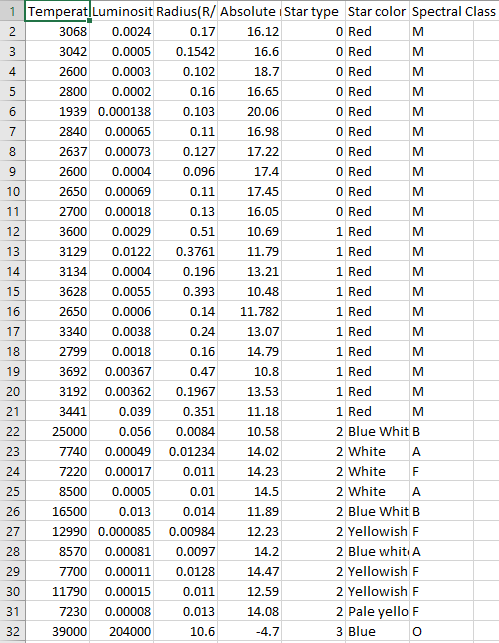
Cột Radius(R/Ro): Bán kính của các vì sao, ký hiệu là 𝑅*R* và thường so sánh với bán kính của Mặt Trời (𝑅⊙​), là một thông số quan trọng trong việc hiểu rõ đặc tính và sự phát triển của chúng. Tỷ lệ bán kính của một ngôi sao so với bán kính của Mặt Trời được ký hiệu là 𝑅/𝑅⊙​.

Cột Absolute magnitude(Mv): Độ sáng tuyệt đối là một thước đo quan trọng trong thiên văn học dùng để đánh giá độ sáng thực sự của một ngôi sao hoặc thiên thể. Độ sáng tuyệt đối là độ sáng của một ngôi sao nếu nó nằm cách Trái Đất 10 parsec (khoảng 32.6 năm ánh sáng).Độ sáng tuyệt đối giúp phân loại các ngôi sao và hiểu rõ hơn về các giai đoạn phát triển của chúng. Sao sáng hơn thường có độ sáng tuyệt đối thấp hơn (ví dụ, sao khổng lồ và siêu khổng lồ), trong khi sao mờ hơn có độ sáng tuyệt đối cao hơn (ví dụ, sao lùn đỏ và sao lùn trắng).

Cột Star type: Red Dwarf(0), Brown Dwarf(1), White Dwarf(2), Main Sequence (3), SuperGiants(4), HyperGiants(5): sao lùn đỏ, sao lùn nâu, sao lùn trắng, sao dãy chính, sao siêu khổng lồ và sao cực khổng lồ.

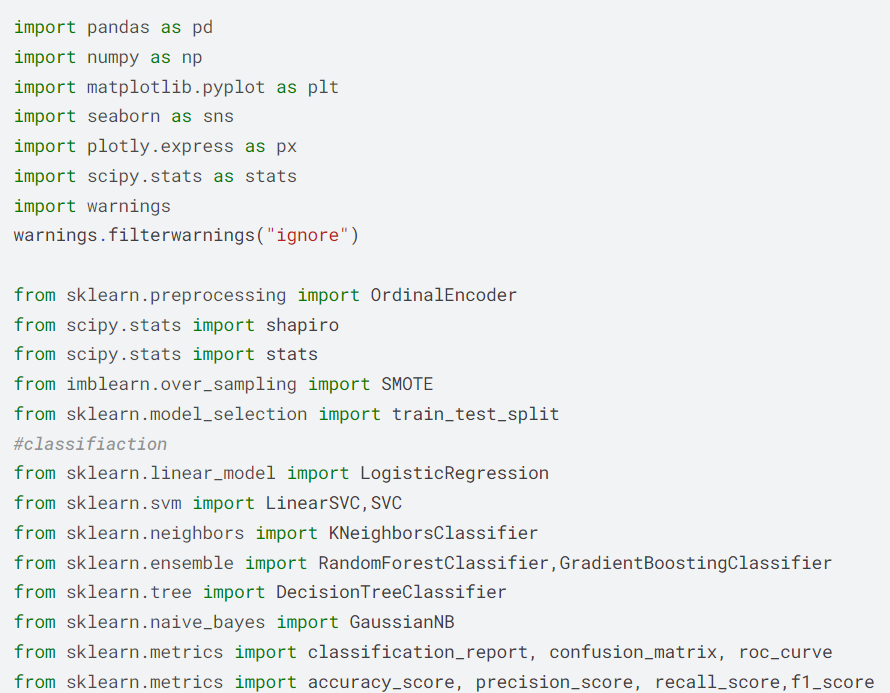
Cột Star color: white, red, blue white, yellow, pale yellow-orange, yellowish white, whitish, orange, yellow-white.

Cột Spectral Class: cách phân loại các ngôi sao dựa trên quang phổ của chúng. Phân loại dựa trên các đặc điểm quang phổ của ánh sáng mà các ngôi sao phát ra khi nhiệt độ của chúng biến đổi. Hệ thống phân loại dựa trên các chữ cái, thường từ A đến M, trong đó mỗi loại đại diện cho một phạm vi nhiệt độ và tính chất khác nhau của các ngôi sao.



Hinh 3.1 Bảng dữ liệu các vì sao

## 3.2 Thư viện của chương trình

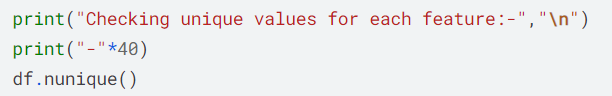


Hinh 3.2 Thư viện chương trình

1. **import pandas as pd:** Import thư viện pandas và đặt tên viết tắt là "pd". Thư viện pandas được sử dụng để làm việc với dữ liệu dạng bảng và chuỗi thời gian.
2. **import numpy as np:** Import thư viện numpy và đặt tên viết tắt là "np". Thư viện numpy cung cấp các công cụ cho tính toán khoa học và toán học trong Python, đặc biệt là khi làm việc với mảng đa chiều.
3. **import matplotlib.pyplot as plt:** Import mô-đun pyplot từ thư viện matplotlib và đặt tên viết tắt là "plt". Matplotlib là một thư viện trực quan hóa dữ liệu trong Python.
4. **import seaborn as sns:** Import thư viện seaborn và đặt tên viết tắt là "sns". Seaborn cung cấp các công cụ trực quan hóa dữ liệu thống kê cao cấp hơn so với matplotlib.
5. **import plotly.express as px:** Import mô-đun express từ thư viện plotly và đặt tên viết tắt là "px". Plotly là một thư viện trực quan hóa dữ liệu tương tác.
6. **import scipy.stats as stats:** Import mô-đun stats từ thư viện scipy và đặt tên viết tắt là "stats". Scipy là một thư viện cho các tính toán và công cụ thống kê.
7. **import warnings:** Import mô-đun warnings từ Python. Mô-đun này được sử dụng để hiển thị hoặc ẩn các cảnh báo trong quá trình thực thi.
8. **warnings.filterwarnings("ignore"):** Tắt cảnh báo để không hiển thị trong quá trình thực thi.
9. **from sklearn.preprocessing import OrdinalEncoder:** Import OrdinalEncoder từ thư viện sklearn. OrdinalEncoder được sử dụng để mã hóa biến đầu vào thành các số nguyên tuần tự.
10. **from scipy.stats import shapiro:** Import shapiro từ scipy.stats. shapiro là một hàm trong thư viện scipy.stats được sử dụng để kiểm tra tính phân phối chuẩn của một mẫu.
11. **from scipy.stats import stats:** Import stats từ scipy.stats. Đây là một lần import trùng lặp, có thể là một lỗi đánh dấu.
12. **from imblearn.over\_sampling import SMOTE:** Import SMOTE từ thư viện imblearn. SMOTE là một kỹ thuật tăng cường dữ liệu để giải quyết vấn đề mất cân bằng lớp trong dữ liệu.
13. **from sklearn.model\_selection import train\_test\_split:** Import train\_test\_split từ thư viện sklearn. Hàm này được sử dụng để chia dữ liệu thành tập huấn luyện và tập kiểm tra.
14. **from sklearn.preprocessing import LabelEncoder:** Import LabelEncoder từ thư viện sklearn. LabelEncoder được sử dụng để chuyển đổi biến mục tiêu (nhãn) thành dạng số.
15. **from sklearn.metrics import ...:** Import một số hàm và lớp từ thư viện sklearn.metrics để đánh giá hiệu suất của mô hình phân loại, bao gồm classification\_report, confusion\_matrix, roc\_curve, accuracy\_score, precision\_score, recall\_score và f1\_score.
16. **from sklearn.linear\_model import LogisticRegression:** Import LogisticRegression từ thư viện sklearn. Logistic Regression là một phương pháp phân loại dựa trên hồi quy tuyến tính.
17. **from sklearn.svm import LinearSVC, SVC:** Import LinearSVC và SVC từ thư viện sklearn. SVC là một mô hình Support Vector Classifier, trong khi LinearSVC là phiên bản tuyến tính của nó.
18. **from sklearn.neighbors import KNeighborsClassifier:** Import KNeighborsClassifier từ thư viện sklearn. KNeighborsClassifier là một mô hình phân loại dựa trên k-nearest neighbors.
19. **from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier:** Import RandomForestClassifier và GradientBoostingClassifier từ thư viện sklearn. Đây là hai mô hình phân loại thuộc loại ensemble, kết hợp nhiều mô hình nhỏ để tạo ra một dự đoán tốt hơn.
20. **from sklearn.tree import DecisionTreeClassifier:** Import DecisionTreeClassifier từ thư viện sklearn. Decision Tree là một phương pháp học máy giúp phân loại dữ liệu dựa trên một loạt các câu hỏi về các tính năng của dữ liệu.
21. **from sklearn.naive\_bayes import GaussianNB:** Import GaussianNB từ thư viện sklearn. Gaussian Naive Bayes là một mô hình phân loại dựa trên nguyên tắc của công thức Bayes với giả định về phân phối Gaussian.

## 3.3 Tiền xử lý

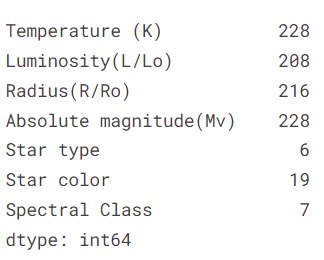
### *3.3.1 Kiểm tra phân phối của các thông số*



Hinh 3.3 Kiểm tra phân phối các thông số

Giải thích: Hàm nunique() là một phương thức trong thư viện pandas của Python, được sử dụng để đếm số lượng giá trị duy nhất trong một Series hoặc DataFrame. Đây là một cách tiện lợi để kiểm tra số lượng các giá trị duy nhất trong một cột dữ liệu.

***Kết quả:***

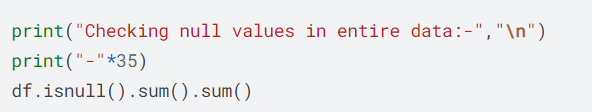


Hinh 3.4 Kết quả kiểm tra phân phối các thông số

* "Temperature (K)", "Absolute magnitude(Mv)": Các cột này có 228 giá trị.
* "Luminosity(L/Lo)": Có 208 giá trị.
* "Radius(R/Ro)": Có 216 giá trị..
* "Star type": Có 6 giá trị.
* "Star color": Có 19 giá trị.
* "Spectral Class": Có 7 giá trị.

Kết luận phân bố thông số không đồng đều. Điều này cung cấp một cái nhìn tổng quan về sự đa dạng của dữ liệu trong mỗi cột.

### *3.3.2 Kiểm tra các giá trị null*

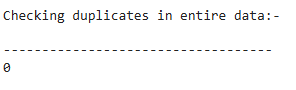


Hinh 3.5 Kiểm tra các giá trị Null

Giải thích: Biểu thức **df.isnull().sum().sum()** được sử dụng để tính tổng số lượng giá trị thiếu trong toàn bộ DataFrame **df**.

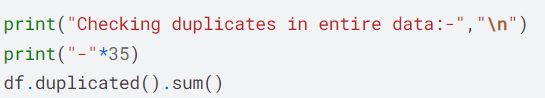
* **df.isnull()** trả về một DataFrame có cùng kích thước như **df**, trong đó mỗi phần tử là True nếu tương ứng với giá trị trong **df** là NaN và False nếu không phải.
* **.sum()** được áp dụng để tính tổng của các giá trị True trong mỗi cột, tức là số lượng giá trị thiếu trong mỗi cột.
* **.sum()** tiếp tục được áp dụng trên kết quả này để tính tổng của tất cả các giá trị thiếu trong toàn bộ DataFrame.

Vậy nên, **df.isnull().sum().sum()** trả về tổng số lượng giá trị thiếu trong toàn bộ DataFrame.



***Kết quả***: Không có giá trị null. Không cần lọc dữ liệu.

#### 3.3.3 Kiểm tra các giá trị trùng lặp



Hình 3.6 Kiểm tra các giá trị trùng lặp

Giải thích: biểu thức **df.duplicated().sum()** được sử dụng để tính tổng số lượng các hàng trùng lặp trong DataFrame **df**.

* **df.duplicated()** trả về một Series với giá trị True cho mỗi hàng trong DataFrame **df** mà là bản sao của một hàng trước đó, và False cho các hàng không trùng lặp.
* **.sum()** được áp dụng để tính tổng số lượng giá trị True trong Series này, tức là số lượng các hàng trùng lặp.

Vì vậy, **df.duplicated().sum()** trả về tổng số lượng các hàng trùng lặp trong DataFrame **df**.

***Kết quả:*** Không có giá trị trùng lặp. Không cần lọc giá trị trùng lặp.

## 3.4 Phân tích dữ liệu

### *3.4.1 Phân bổ ở các loại sao*



Hinh 3.6 Chương trình phân bổ các loài sao

Giải thích: Đoạn mã này được sử dụng để vẽ biểu đồ đếm số lượng mẫu cho mỗi loại sao trong cột 'Star type' của DataFrame `df`. Sau đó, văn bản được thêm vào mỗi cột để hiển thị phần trăm của mỗi loại sao trên biểu đồ.

1. `ax=sns.countplot(data=df, x='Star type', palette="Spectral")`: Vẽ biểu đồ đếm sử dụng Seaborn, đếm số lượng mẫu cho mỗi loại sao trong cột 'Star type' của DataFrame `df`. Màu sắc của biểu đồ được thiết lập bằng cách sử dụng bảng màu "Spectral".

2. `percentages=df['Star type'].value\_counts()/len(df)\*100`: Tính phần trăm của mỗi loại sao trong cột 'Star type' bằng cách chia số lượng mẫu của mỗi loại sao cho tổng số mẫu và nhân với 100.

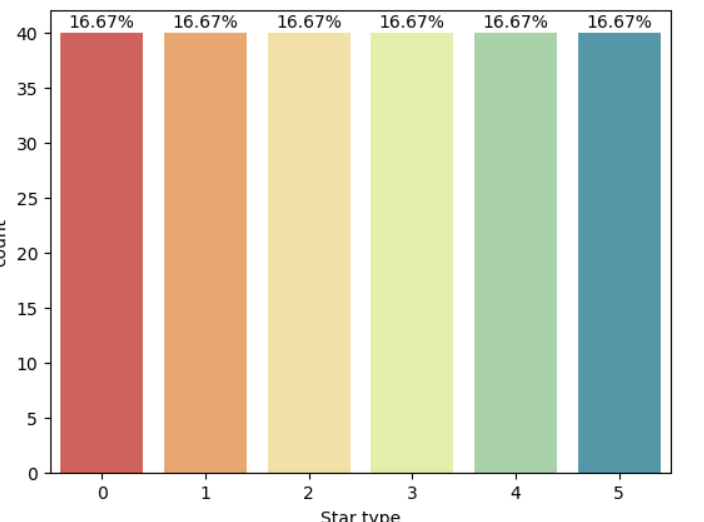
3. Vòng lặp `for i,p in enumerate(ax.patches)`: Duyệt qua từng thanh trong biểu đồ. `ax.patches` là danh sách các thanh trên biểu đồ.

4. `height=p.get\_height()`: Lấy chiều cao của thanh hiện tại.

5. `ax.text(p.get\_x()+p.get\_width()/2, height+0.5, f'{percentages[i]:.2f}%' , ha="center")`: Thêm văn bản vào mỗi thanh để hiển thị phần trăm tương ứng của loại sao. `p.get\_x()+p.get\_width()/2` là vị trí x của văn bản, `height+0.5` là vị trí y của văn bản (cao hơn một chút so với đỉnh của thanh), `f'{percentages[i]:.2f}%'` là chuỗi định dạng để hiển thị phần trăm với 2 chữ số thập phân, và `ha="center"` là căn giữa văn bản.

6. `plt.show()`: Hiển thị biểu đồ.

**Kết quả:**



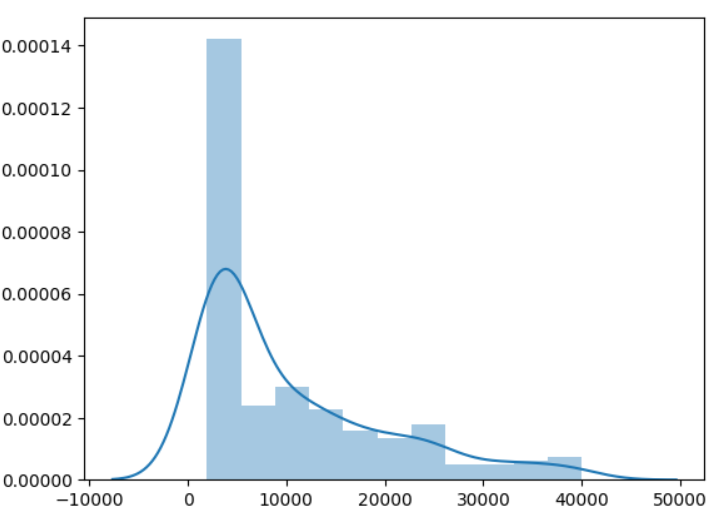
Hinh 3.7 Biểu đồ phân bổ của các vì sao

***Kết luận:*** Tổng công có 6 loại sao, lần lượt các màu đỏ, cam, cam nhạt, vàng nhạt, xanh lá, xanh dương tương ứng ứng loại sao từ 0 đến 5. Số lượng sao đồng đều là 40 sao mỗi loại chiếm 16.67% số lượng mỗi loại sao. Tỉ lệ hoàn hảo giúp sự phân loại trở nên chính xác hơn.

### *3.4.2 Mật độ xác suất dựa trên 1 giá trị đặc điểm*

* + Dựa vào nhiệt độ





Hinh 3.8 Biểu đồ mật độ xác suất theo giá trị nhiệt độ

**Kết luận:**

* Từ biểu đồ trên, chúng ta có thể thấy rằng hầu hết các giá trị nằm ở trên 0 và dưới 10000, và biểu đồ cũng bị lệch phải mạnh mẽ.
* Dải nhiệt độ nằm trong khoảng từ 1939 đến 40000.
* Trung bình của nhiệt độ là 10497,
* Trung vị của nhiệt độ là 5776.
  + Biểu đồ phân bố nhiệt độ theo từng loại sao



Giải thích: Đoạn mã này được sử dụng để vẽ histogram cho phân phối của nhiệt độ (Temperature) của các loại sao khác nhau trong cột "Star type" của DataFrame `df`. Mỗi histogram sẽ hiển thị phân phối của một loại sao cụ thể.

1. `unique\_star\_types = df["Star type"].unique()`: Tạo một mảng chứa tất cả các giá trị duy nhất của cột "Star type" trong DataFrame `df`.

2. `plt.figure(figsize=(12, 9))`: Tạo một hình vẽ với kích thước 12x9 inch (có thể điều chỉnh kích thước nếu cần thiết).

3. Vòng lặp `for i, star\_type in enumerate(unique\_star\_types)`: Duyệt qua mỗi loại sao trong mảng `unique\_star\_types`.

4. `filtered\_df = df[df["Star type"] == star\_type]`: Lọc DataFrame `df` để chỉ chứa các hàng có giá trị trong cột "Star type" là loại sao hiện đang xét.

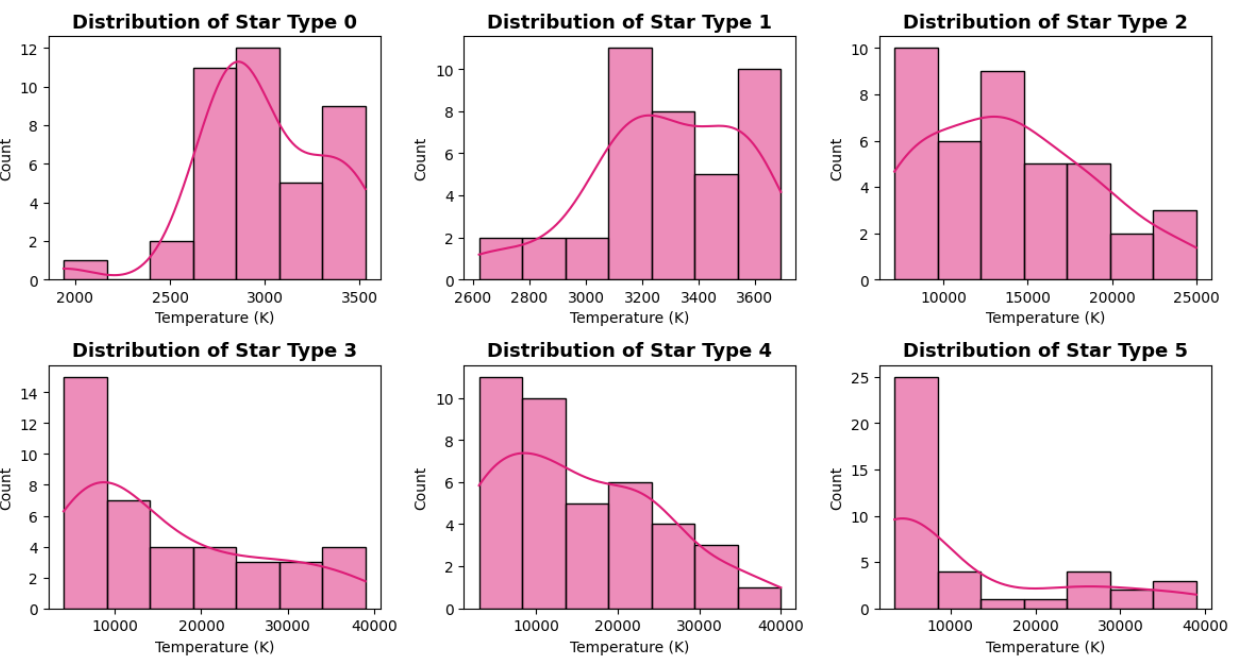
5. `plt.subplot(3, 3, i + 1)`: Tạo một subplot trong hình vẽ với hàng là 3, cột là 3 và chỉ số subplot là `i + 1` (tính từ 1).

6. `sns.histplot(data=filtered\_df, x="Temperature (K)", kde=True, color="#dd1c77")`: Vẽ histogram của cột "Temperature (K)" trong DataFrame đã lọc `filtered\_df` bằng cách sử dụng Seaborn. Tham số `kde=True` được sử dụng để hiển thị dạng phân phối ước lượng mật độ kernel trên histogram. Màu của histogram được thiết lập bằng mã hex "#dd1c77".

7. `plt.title(f"Distribution of Star Type {star\_type}", fontweight="bold", size=13)`: Đặt tiêu đề cho subplot, thể hiện loại sao hiện đang xét.

8. `plt.tight\_layout()`: Tự động điều chỉnh layout của subplot để tránh trùng lặp.

9. `plt.show()`: Hiển thị hình vẽ chứa tất cả các histogram.



Hinh 3.10 Biểu đồ phân bổ nhiệt độ theo từng loại sao

***Kết luận***: Theo biểu đồ trên ta thấy:

Loại sao 0: Có nhiều sao nhất ở nhiệt độ 3000K. Nhưng các giá trị nhiệt độ khác cũng có xuất hiện sao loại 0 từ 0 đến khoảng 3500 K.

Loại sao 1: Ở nhiệt độ 3100 đến 3200 K có nhiều sao loại 1 nhất. Các giá trị khác cũng có sao loại 1 xuất hiện từ 2600 đến 3700 K.

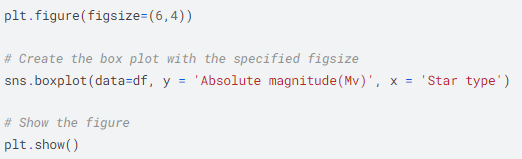
Loại sao 2: Từ 0 đến 10000 K có nhiều sao loại 2 nhất . Ở sao loại 2 số lượng sao xuất hiện ở nhiệt độ từ 0 đến 25000 K, chúng xuất hiện khá đồng đều, số lượng các sao ở các nhiệt độ khác nhau xấp xỉ nhau.

Loại sao 3: Ở loại sao này số lượng sao ở 0 đến 10000 K vượt trội hẳn so với các loại sao ở giá trị nhiệt độ còn lại. Loại sao 3 xuất hiện ở giá trị nhiệt độ từ 0 đến 40000 K.

Loại sao 4: Nhiệt độ 0 đến 15000 K loại sao số 4 phân bổ nhiều nhất, còn lại phân phối đều ở nhiệt độ 15000 đến 40000 K. Sao loại 4 phân bổ từ 0 đến 40000 K.

Loại sao 5: Hầu hết các sao loại 5 phân bổ ở 0 đến 10000 K còn lại phân bổ lẻ tẻ ở 10000 đến 40000 K. Loại sao 5 phân bổ từ 0 đến 40000 K.

Vì vậy ta kết luận không có loại sao nào chỉ được phân loại theo nhiệt độ. Hầu hết các loại sao phân bố ở các giá trị nhiệt độ vì vậy cần phân tích thêm các đặc điểm khác.

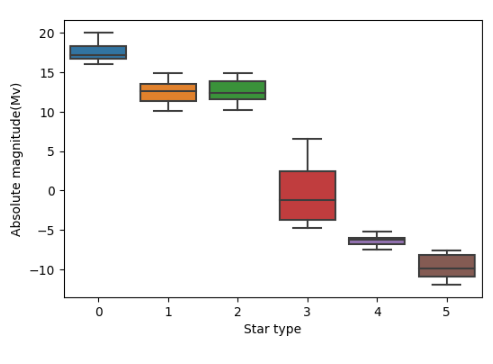
* + Dựa vào độ sáng tuyệt đối

Giải thích: Đoạn mã này được sử dụng để vẽ biểu đồ hộp (box plot) cho cột "Absolute magnitude(Mv)" dựa trên loại sao (cột "Star type") trong DataFrame `df`. Dưới đây là giải thích từng dòng:

1. `plt.figure(figsize=(6,4))`: Tạo một hình vẽ với kích thước 6x4 inch bằng cách sử dụng hàm `plt.figure()`. Kích thước này được chỉ định bởi `figsize=(6,4)` để điều chỉnh kích thước của hình vẽ.

2. `sns.boxplot(data=df, y='Absolute magnitude(Mv)', x='Star type')`: Tạo biểu đồ hộp (box plot) sử dụng Seaborn. Cột "Absolute magnitude(Mv)" được hiển thị trên trục y và loại sao (cột "Star type") được hiển thị trên trục x. Biểu đồ hộp cho phép chúng ta thấy phân phối, phạm vi và các giá trị ngoại lai của cột "Absolute magnitude(Mv)" cho từng loại sao.

3. `plt.show()`: Hiển thị biểu đồ.



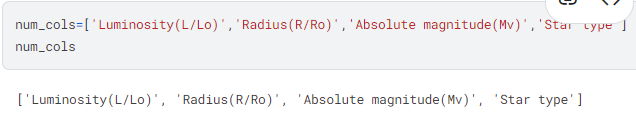
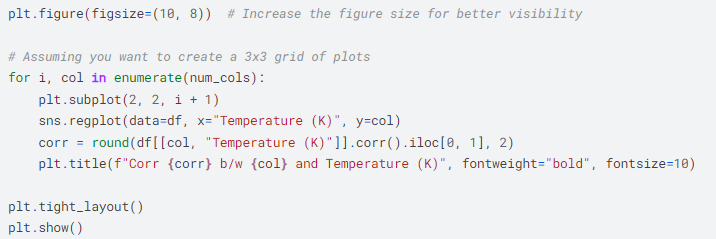
Hinh 3.9 Biểu đồ Dựa vào độ sáng tuyệt đối

***Kết luận***: Ngôi sao Nâu (Brown Dwarf - 0) có độ sáng tuyệt đối cao nhất trong tất cả các loại sao. Ngôi sao cực khổng lồ (HyperGiants - 5) có độ sáng tuyệt đối thấp nhất.

* + Dựa vào

*3.4.3 Phân loại dựa trên 2 yếu tố*

* + Dựa vào độ sáng và nhiệt độ , bán kính và nhiệt độ, độ sáng tuyệt đối và nhiệt độ, loại sao và nhiệt độ

Giải thích: Đoạn mã này tạo một lưới 3x3 của các biểu đồ phân tán (scatter plots) giữa cột "Temperature (K)" và các cột dữ liệu trong danh sách `num\_cols`, đồng thời cũng hiển thị hệ số tương quan giữa chúng trong tiêu đề của mỗi subplot.

1. `num\_cols=['Luminosity(L/Lo)','Radius(R/Ro)','Absolute magnitude(Mv)','Star type']`: Danh sách `num\_cols` chứa các tên cột mà bạn muốn tạo các biểu đồ phân tán với cột "Temperature (K)".

2. Vòng lặp `for i, col in enumerate(num\_cols)`: Duyệt qua từng cột trong `num\_cols`.

3. `plt.subplot(2, 2, i + 1)`: Tạo một subplot trong lưới 2x2, với chỉ số của subplot được tính toán từ biến `i`.

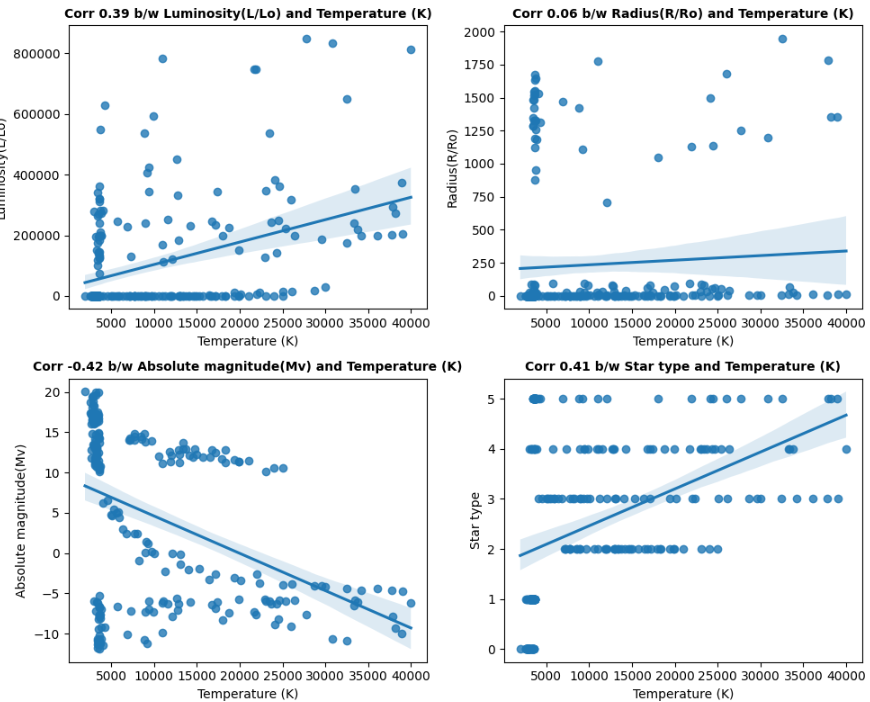
4. `sns.regplot(data=df, x="Temperature (K)", y=col)`: Vẽ biểu đồ phân tán giữa cột "Temperature (K)" và cột hiện đang xét, sử dụng `sns.regplot()` từ thư viện Seaborn. Đây là biểu đồ phân tán với đường hồi quy.

5. `corr = round(df[[col, "Temperature (K)"]].corr().iloc[0, 1], 2)`: Tính toán hệ số tương quan giữa cột hiện đang xét và cột "Temperature (K)" trong DataFrame `df`.

6. `plt.title(f"Corr {corr} b/w {col} and Temperature (K)", fontweight="bold", fontsize=10)`: Đặt tiêu đề cho subplot với thông tin về hệ số tương quan giữa hai cột và tên của hai cột.

7. `plt.tight\_layout()`: Tự động điều chỉnh layout của subplot để tránh trùng lặp.

8. `plt.show()`: Hiển thị biểu đồ.



Hinh 3.10 Biểu đồ Dựa vào độ sáng và nhiệt độ , bán kính và nhiệt độ

***Kết luận:***

Nhìn vào biểu đồ và tính toán ta thấy:

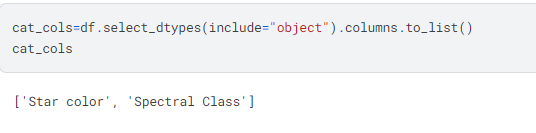
Luminosity(L/Lo) có mối tương quan với nhiệt độ là 0.39 \_ mối tương quan dương trung bình.

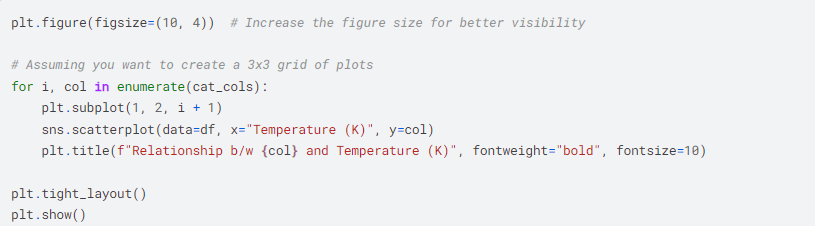
Radius(R/Ro) có mối tương quan với nhiệt độ là 0.06 \_ mối tương quan dương yếu.

Absolute magnitude(Mv) có mối tương quan âm trung bình với nhiệt độ là -0.42.

Loại sao có mối tương quan dương trung bình với đặc điểm nhiệt độ đó là 0.41.

Những ngôi sao nóng nhất là dãy chính (3 Main Sequence ), siêu khổng lồ (4 SuperGiants) và siêu khổng lồ cực lớn (5 HyperGiants).

* Dựa vào màu sắc và nhiệt độ, lớp quang phổ và nhiệt độ



Giải thích: Đoạn mã này được sử dụng để vẽ biểu đồ phân tán (scatter plot) giữa cột "Temperature (K)" và các cột dữ liệu có kiểu dữ liệu là object trong DataFrame `df`.

1. `cat\_cols=df.select\_dtypes(include="object").columns.to\_list()`: Đầu tiên, chúng ta chọn các cột trong DataFrame `df` có kiểu dữ liệu là object bằng cách sử dụng `select\_dtypes(include="object")`. Sau đó, chúng ta chuyển các tên cột này thành một danh sách và lưu vào biến `cat\_cols`.

2. `cat\_cols`: In ra danh sách các cột dữ liệu dạng object trong DataFrame `df`.

3. `plt.figure(figsize=(10, 4))`: Tạo một hình vẽ với kích thước 10x4 inch để tăng kích thước và làm cho biểu đồ dễ nhìn hơn.

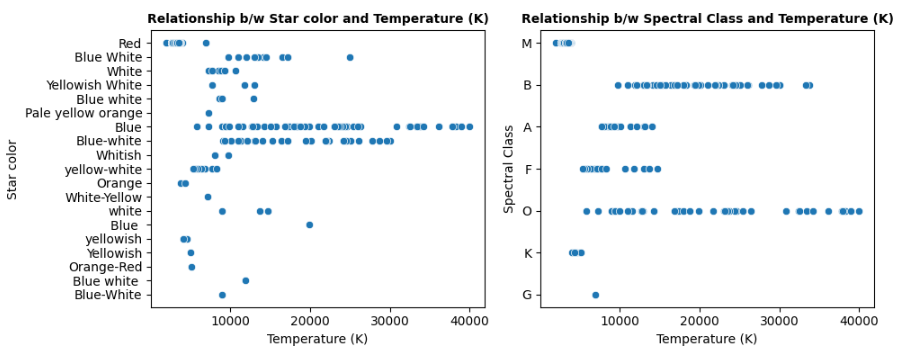
4. `for i, col in enumerate(cat\_cols):`: Duyệt qua từng cột trong danh sách `cat\_cols`, sử dụng `enumerate` để lấy cả chỉ số và giá trị của cột.

5. `plt.subplot(1, 2, i + 1)`: Tạo một subplot trong lưới 1x2, với chỉ số của subplot được tính toán từ biến `i`. Trong trường hợp này, chúng ta tạo một lưới 1x2 với mỗi subplot hiển thị mối quan hệ giữa một cột dữ liệu dạng object và cột "Temperature (K)".

6. `sns.scatterplot(data=df, x="Temperature (K)", y=col)`: Vẽ biểu đồ phân tán giữa cột "Temperature (K)" và cột dữ liệu dạng object hiện đang xét (`col`). Chúng ta sử dụng `sns.scatterplot()` từ thư viện Seaborn để vẽ biểu đồ phân tán.

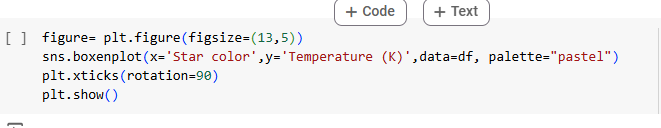
7. `plt.title(f"Relationship b/w {col} and Temperature (K)", fontweight="bold", fontsize=10)`: Đặt tiêu đề cho subplot với thông tin về mối quan hệ giữa cột dữ liệu dạng object và cột "Temperature (K)".

8. `plt.tight\_layout()`: Tự động điều chỉnh layout của subplot để tránh trùng lặp.

9. `plt.show()`: Hiển thị biểu đồ.

Hinh 3.11 Biểu đồ Dựa vào màu sắc và nhiệt độ, lớp quang phổ và nhiệt độ

***Kết luận***: Ngôi sao nóng nhất là blue và lớp quang phổ nóng nhất là O. Sau đó đến màu nóng nhất là Blue-white và lớp quang phổ B. Còn lại nhiệt độ của các sao lớp quang phổ và màu khác là trung bình.

* + Dựa vào màu sắc và nhiệt độ 

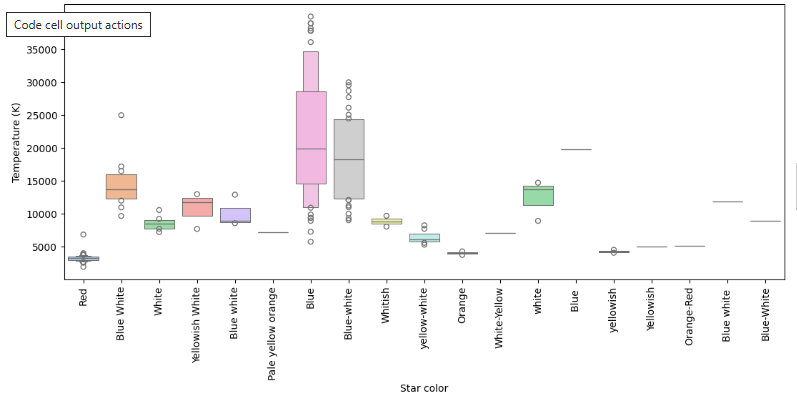
Giải thích: Đoạn mã này được sử dụng để vẽ biểu đồ hộp boxen (boxen plot) cho mỗi mẫu màu sao (cột "Star color") theo nhiệt độ (cột "Temperature (K)") trong DataFrame df.

figure = plt.figure(figsize=(13,5)): Tạo một hình vẽ mới với kích thước 13x5 inch bằng cách sử dụng hàm plt.figure(). Kích thước này được chỉ định bởi figsize=(13,5) để điều chỉnh kích thước của hình vẽ.

sns.boxenplot(x='Star color', y='Temperature (K)', data=df): Vẽ biểu đồ hộp boxen (boxen plot) sử dụng Seaborn. Trong biểu đồ này, cột "Star color" được hiển thị trên trục x và cột "Temperature (K)" được hiển thị trên trục y. Biểu đồ hộp boxen mở rộng biểu đồ hộp thông thường bằng cách thêm các đường thẳng ngắn để biểu diễn các phần phân phối đồng dạng.

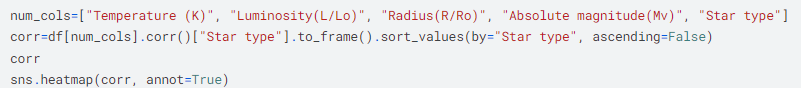
plt.xticks(rotation=90): Xoay các nhãn trên trục x một góc 90 độ để tránh chồng chéo và dễ đọc hơn.

plt.show(): Hiển thị biểu đồ.



***Kết luận:*** biểu đồ hộp cho thấy rằng màu sao "Blue" có nhiệt độ cao nhất so với các màu sao khác. Điều này là rõ ràng khi ta quan sát các hộp trên biểu đồ, hộp của màu "Blue" nằm ở vị trí có giá trị trung vị (median) cao nhất và cả hai đầu hộp đều nằm ở phía trên so với các màu sao khác. Điều này cho thấy rằng phần lớn các sao màu "Blue" có nhiệt độ cao hơn so với các màu sao khác trong dữ liệu.

### 3.4.4 Tổng quan



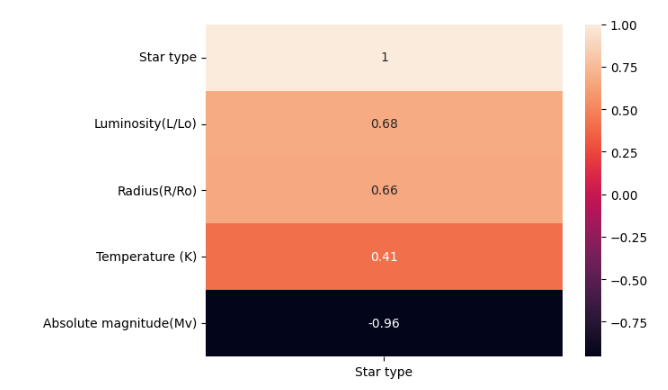
Giải thích: Đoạn mã này thực hiện các bước sau:

1. `num\_cols=["Temperature (K)", "Luminosity(L/Lo)", "Radius(R/Ro)", "Absolute magnitude(Mv)", "Star type"]`: Định nghĩa danh sách `num\_cols` chứa tên của các cột dữ liệu số trong DataFrame `df`, bao gồm "Temperature (K)", "Luminosity(L/Lo)", "Radius(R/Ro)", "Absolute magnitude(Mv)" và "Star type".

2. `corr=df[num\_cols].corr()["Star type"].to\_frame().sort\_values(by="Star type", ascending=False)`: Tính toán ma trận tương quan giữa các cột trong `num\_cols` với cột "Star type". Sau đó, kết quả được chuyển thành một DataFrame, sắp xếp theo giá trị tương quan giảm dần theo cột "Star type". Điều này cho phép chúng ta xem xét mức độ tương quan giữa "Star type" và các biến số khác.

3. `sns.heatmap(corr, annot=True)`: Vẽ biểu đồ heatmap (bản đồ nhiệt) sử dụng Seaborn để hiển thị ma trận tương quan đã tính toán. Tham số `annot=True` được sử dụng để hiển thị các giá trị tương quan trong các ô.

Vì vậy đây là cách chúng ta biểu diễn mức độ tương quan giữa cột "Star type" và các biến số khác trong DataFrame bằng heatmap. Các giá trị tương quan cao được mô tả bằng màu sắc tương ứng.

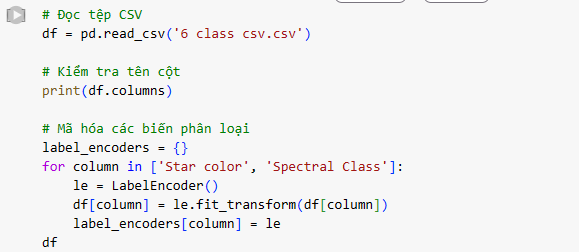


Hinh 3.12 Biểu đồ tổng quan các yếu tố

***Kết luận:*** Các đặc trưng Luminosity và Radius có mối tương quan mạnh với việc phận loại sao. Trong khi, nhiệt độ có mối tương quan trung bình và Độ sáng tuyệt đối có mối tương quan thấp với nó.

## 3.5 Xây dựng mô hình

### 3.5.1 Chuẩn hóa dữ liệu



Giải thích: Đoạn mã này thực hiện các bước sau:

1. `df = pd.read\_csv('6 class csv.csv')`: Đọc dữ liệu từ tệp CSV có tên là '6 class csv.csv' vào DataFrame `df`.

2. `print(df.columns)`: In ra tên của các cột trong DataFrame `df`.

3. Mã hóa các biến phân loại:

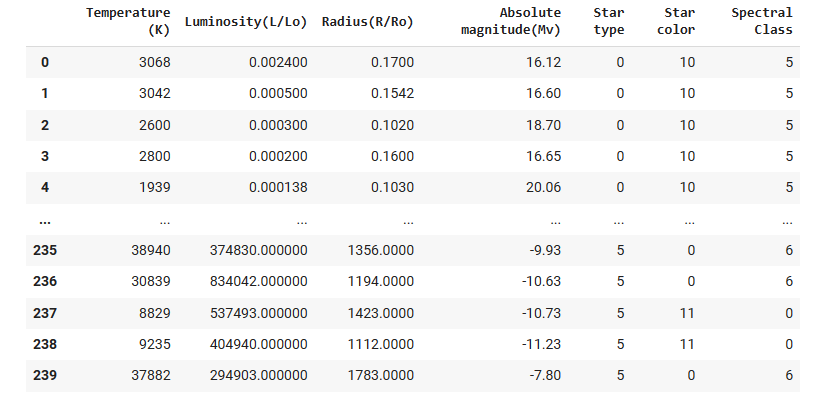
- Tạo một từ điển rỗng `label\_encoders` để lưu trữ các đối tượng LabelEncoder cho mỗi cột phân loại.

- Duyệt qua từng cột phân loại trong danh sách ['Star color', 'Spectral Class'].

- Tạo một đối tượng LabelEncoder mới cho mỗi cột phân loại và gán nó cho cột tương ứng trong DataFrame `df`. LabelEncoder được sử dụng để chuyển đổi các giá trị phân loại thành các số nguyên duy nhất.

- Lưu trữ đối tượng LabelEncoder tương ứng với mỗi cột phân loại vào từ điển `label\_encoders`.

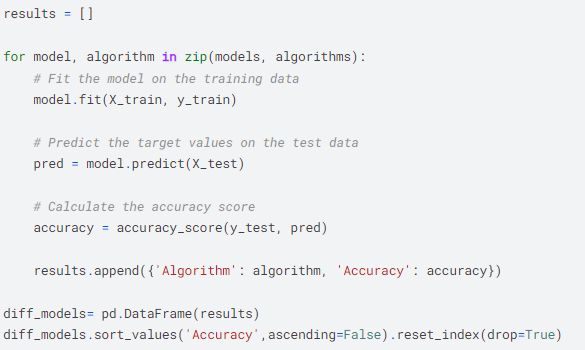
4. Trả về DataFrame `df` sau khi đã mã hóa các biến phân loại.



Hinh 3.13 Chuẩn hóa dữ liệu

Kết luận: Các cột phân loại 'Star color' và 'Spectral Class' đã được chuyển đổi từ các giá trị chuỗi sang các giá trị số nguyên duy nhất, để có thể sử dụng trong việc xây dựng các mô hình học máy.

### 3.5.2 Huấn luyện mô hình



Hinh 3.14 Huấn luyện mô hình

Giải thích: Đoạn mã này thực hiện các bước sau:

1. `X= df.drop(columns=["Star type"], axis=1)`: Tạo DataFrame `X` bằng cách loại bỏ cột "Star type" từ DataFrame `df`. Cột "Star type" sẽ là biến mục tiêu (target variable) trong việc xây dựng mô hình dự đoán.

2. `y=df["Star type"]`: Tạo Series `y` để lưu trữ biến mục tiêu "Star type".

3. `X\_train,X\_test,y\_train,y\_test=train\_test\_split(X, y, test\_size=0.7, random\_state=43)`: Chia dữ liệu thành tập huấn luyện (`X\_train`, `y\_train`) và tập kiểm tra (`X\_test`, `y\_test`) bằng cách sử dụng hàm `train\_test\_split()` từ sklearn. Dữ liệu được chia theo tỷ lệ 70% cho tập kiểm tra và 30% cho tập huấn luyện, và tham số `random\_state=43` được sử dụng để đảm bảo kết quả chia dữ liệu không thay đổi khi chạy mã nhiều lần.

4. `models = [...]`: Tạo danh sách các mô hình máy học được sử dụng để phân loại dữ liệu. Danh sách này bao gồm các mô hình như Logistic Regression, Support Vector Machine, K-Nearest Neighbors, Random Forest, Decision Tree và Gaussian Naive Bayes.

5. `algorithms = [...]`: Tạo danh sách các tên của các mô hình máy học tương ứng với danh sách `models`.

6. `results = []`: Khởi tạo một danh sách rỗng để lưu trữ kết quả độ chính xác của mỗi mô hình.

7. Vòng lặp `for model, algorithm in zip(models, algorithms)`: Duyệt qua từng mô hình và tên của nó.

8. `model.fit(X\_train, y\_train)`: Huấn luyện mô hình trên tập huấn luyện `X\_train` và `y\_train`.

9. `pred = model.predict(X\_test)`: Dự đoán các giá trị mục tiêu trên tập kiểm tra `X\_test`.

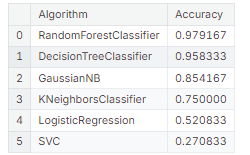
10. `accuracy = accuracy\_score(y\_test, pred)`: Tính toán độ chính xác của mô hình bằng cách so sánh giá trị dự đoán với giá trị thực tế từ tập kiểm tra, sử dụng hàm `accuracy\_score` từ sklearn.metrics.

11. `results.append({'Algorithm': algorithm, 'Accuracy': accuracy})`: Thêm kết quả độ chính xác của mô hình vào danh sách `results`.

12. `diff\_models= pd.DataFrame(results)`: Tạo DataFrame `diff\_models` từ danh sách kết quả.

13.`diff\_models.sort\_values('Accuracy',ascending=False).reset\_index(drop=True)`: Sắp xếp DataFrame `diff\_models` theo độ chính xác giảm dần và đặt lại chỉ số của hàng. Điều này cho phép chúng ta xem xét mô hình nào có độ chính xác cao nhất.

Kết quả độ chính xác của mô hình theo các thuật toán:



Hinh 3.15 Kết quả độ chính xác của mô hình

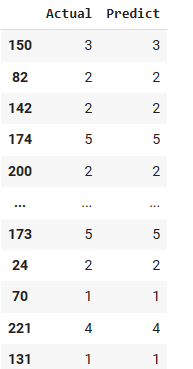
Theo kết quả trên ta thấy mô hình Random Forest Classifier và Decision Tree Classifier có độ chính xác cao nhất trong các mô hình.

Kết quả thực tế và kết quả dự đoán:



***Kết luận:***

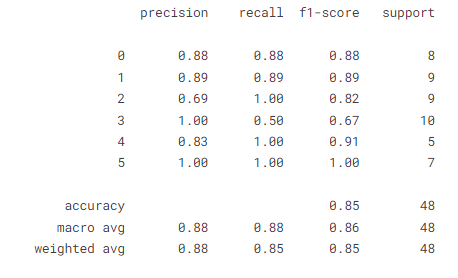
Độ chính xác lên tới 97,9% như kết quả dưới tuy nhiên vẫn có vài trường hợp phân loại sai như TH 78 : thực tế là sao 1 (Brown Dwarf) nhưng dự đoán là sao 0 (Red Dwarf),…



Hinh 3.16 Kết quả thực tế và dự đoán

Kết quả phân loại của các vì sao

*Đánh giá hiệu suất mô hình:*



Hinh 3.17 Đánh giá hiệu suất mô hình

Giải thích:

Các số liệu được hiển thị ở trên là kết quả của một báo cáo phân loại (classification report), mà thường được sử dụng để đánh giá hiệu suất của mô hình phân loại trên một tập dữ liệu kiểm tra. Dưới đây là giải thích từng phần của báo cáo:

- Precision (Chính xác): Là tỷ lệ giữa số lượng dự đoán đúng của một lớp (true positives) và tổng số lượng dự đoán của lớp đó (true positives + false positives). Precision được tính theo công thức: precision = true positives / (true positives + false positives). Trong bảng trên:

* + - Lớp 0 có precision là 0.68, nghĩa là 68% các dự đoán được gán nhãn là lớp 0 là đúng.
    - Lớp 1 có precision là 1.00, tức là 100% các dự đoán được gán nhãn là lớp 1 là đúng.
    - Các lớp khác cũng có precision tương tự.

- Recall: Là tỷ lệ giữa số lượng dự đoán đúng của một lớp (true positives) và tổng số lượng thực tế của lớp đó trong dữ liệu (true positives + false negatives). Recall được tính theo công thức: recall = true positives / (true positives + false negatives). Trong báo cáo trên:

* + Lớp 0 có recall là 0.97, tức là 97% số mẫu thực sự thuộc lớp 0 đã được dự đoán đúng.
  + Lớp 1 có recall là 0.56, tức là chỉ có 56% số mẫu thực sự thuộc lớp 1 đã được dự đoán đúng.
  + Các lớp khác cũng có recall tương tự.

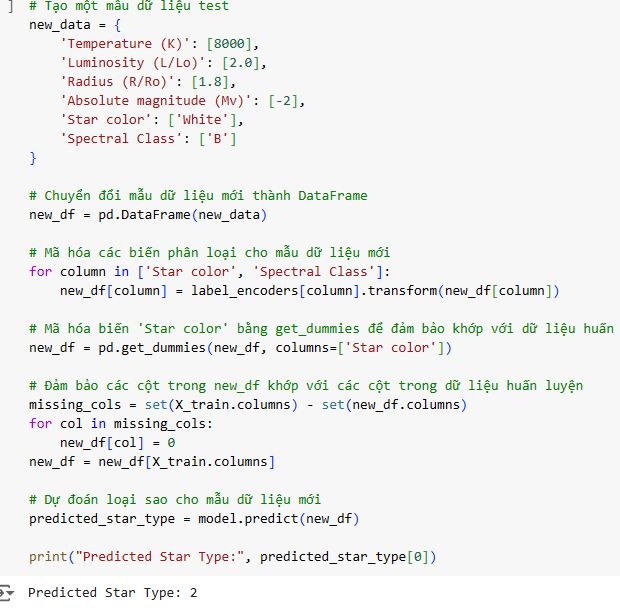
- F1-score: Là một số đo kết hợp giữa precision và recall, được tính theo công thức: F1-score = 2 \* (precision \* recall) / (precision + recall). Nó là trung bình điều hòa của precision và recall, và giúp đánh giá mô hình phân loại một cách tổng quát. Trong báo cáo trên, các lớp đều có F1-score khá cao, cho thấy một sự cân bằng giữa precision và recall.

- Support: Là số lượng mẫu thực tế thuộc vào mỗi lớp trong tập kiểm tra.

- Accuracy (Độ chính xác tổng thể): Là tỷ lệ giữa số lượng dự đoán đúng trên tổng số lượng dự đoán. Trong báo cáo trên, accuracy là 0.82, tức là mô hình dự đoán đúng khoảng 82% số lượng mẫu trong tập kiểm tra.

- Macro avg và weighted avg: Là giá trị trung bình của precision, recall và F1-score qua tất cả các lớp, tính theo hai cách khác nhau. Macro avg trung bình các giá trị của tất cả các lớp một cách không cân nhắc trọng số cho kích thước của mỗi lớp, trong khi weighted avg trung bình các giá trị của tất cả các lớp nhưng cân nhắc trọng số cho kích thước của mỗi lớp. Trong báo cáo trên, macro avg và weighted avg cho thấy mức độ hiệu suất trung bình của mô hình trên tất cả các lớp.

### 3.5.3 Chương trình Test tìm kiếm sao dựa vào đặc điểm



Hinh 3.18 Chương trình test

Giải thích: Đoạn mã này thực hiện các bước sau để dự đoán loại sao cho một mẫu dữ liệu mới:

1. Tạo một mẫu dữ liệu test mới:

- Tạo một từ điển `new\_data` đại diện cho một mẫu dữ liệu mới, với các giá trị cho các thuộc tính như 'Temperature (K)', 'Luminosity (L/Lo)', 'Radius (R/Ro)', 'Absolute magnitude (Mv)', 'Star color', và 'Spectral Class'.

- Chuyển từ điển `new\_data` thành DataFrame `new\_df`.

2. Mã hóa các biến phân loại cho mẫu dữ liệu mới:

- Duyệt qua các cột 'Star color' và 'Spectral Class'.

- Sử dụng `label\_encoders` để chuyển đổi các giá trị trong các cột này thành các số nguyên tương ứng.

3. Mã hóa biến 'Star color' bằng phương pháp One-Hot Encoding:

- Sử dụng `pd.get\_dummies()` để chuyển đổi cột 'Star color' thành các cột dummies để đảm bảo rằng các cột này khớp với dữ liệu huấn luyện.

4. Đảm bảo các cột trong `new\_df` khớp với các cột trong dữ liệu huấn luyện:

- Kiểm tra và thêm các cột cần thiết để `new\_df` có cùng số lượng cột với `X\_train`, đảm bảo rằng mô hình có thể dự đoán trên mẫu dữ liệu mới.

5. Dự đoán loại sao cho mẫu dữ liệu mới:

- Sử dụng mô hình đã được huấn luyện (`model`) để dự đoán loại sao cho mẫu dữ liệu mới (`new\_df`).

- In ra loại sao dự đoán được.

***Kết luận***: Loại sao trên được test là sao số 2\_White Dwarf.

# KẾT LUẬN

* 1. **Kết quả đạt được**

Sau khi hoàn thành đề tài, Em đã đạt được một số kết quả sau:

* Kiểm tra được việc phân loại vì sao dựa trên đặc điểm(Yếu tố) bằng cách sử dụng các loại biểu đồ.
  + - Xây dựng mô hình và huấn luyện mô hình bằng các thuật toán học máy: LogisticRegression, KNN (K-Nearest Neighbors), SVM (Support Vector Machine), RandomForest, DecisionTree, Naïve Bayes.
* Lựa chọn thuật toán tối ưu nhất cho mô hình phân loại.
* Đánh giá được hiệu suất của mô hình.
  1. **Hạn chế đề tài**

Độ chính xác của thuật toán cao tuy nhiên chưa hoàn toàn được 100%.

* 1. **Hướng phát triển**

Kết hợp tạo ra website tìm kiếm phân loại các vì sao.

# 

# TÀI LIỆU THAM KHẢO

1. Trang website tham khảo “[https://machinelearningcoban.com//](https://www.w3schools.com/)”
2. Trang website tham khảo “<https://docs.cocos.com/creator/manual/zh/>”
3. Trang website tham khảo “https://www.geeksforgeeks.org/”
4. Tài liệu Data “https://www.sdss.org/dr18/