BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC GIA ĐỊNH**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN TỐT NGHIỆP**

**TÊN ĐỀ TÀI**

**XÂY DỰNG HỆ THỐNG NHẬN DẠNG CHỮ SỐ VIẾT TAY SỬ DỤNG BỘ DỮ LIỆU MNIST**

Giảng viên hướng dẫn: **TRẦN HOÀI THUẬN**

Sinh viên thực hiện: **TRỊNH NGỌC MINH**

MSSV: **22150318** Lớp: **221521** Khóa: **K16**

TP.Hồ Chí Minh, tháng 7 năm 2025

**Khoa/Viện: Công Nghệ Thông Tin**

**NHẬN XÉT VÀ CHẤM ĐIỂM CỦA GIẢNG VIÊN**

1. **Họ và tên sinh viên: Trịnh Ngọc Minh**
2. **Tên đề tài**: Xây dựng hệ thống nhận dạng chữ số viết tay sử dụng bộ dữ liệu MNIST
3. **Nhận xét**:

***a. Những kết quả đạt được:***

***b. Những hạn chế:***

1. **Điểm đánh giá** *(theo thang điểm 10, làm tròn đến 0.5):*

Điểm số:

Điểm chữ:

|  |  |
| --- | --- |
|  | *TP. HCM, ngày 20 tháng 12 năm 2024*  **Giảng viên chấm thi**  *(Ký và ghi rõ họ tên)* |

# LỜI MỞ ĐẦU

Hệ thống nhận dạng chữ số viết tay là một ứng dụng quan trọng trong lĩnh vực thị giác máy tính, với nhiều ứng dụng thực tiễn trong việc tự động hóa quá trình nhận diện dữ liệu số từ các tài liệu viết tay, ngân hàng, dịch vụ bưu chính, hay các hệ thống nhận dạng chữ viết tự động. Một trong những bài toán nổi bật trong nhận dạng chữ viết tay là nhận diện các chữ số, đặc biệt là các chữ số được viết tay trong các mẫu dữ liệu có cấu trúc phong phú và đa dạng như bộ dữ liệu MNIST.

MNIST (Modified National Institute of Standards and Technology) là một trong những bộ dữ liệu nổi tiếng nhất trong lĩnh vực học máy, bao gồm 60.000 hình ảnh huấn luyện và 10.000 hình ảnh kiểm tra của các chữ số viết tay từ 0 đến 9. Bộ dữ liệu này đã trở thành chuẩn mực để thử nghiệm và so sánh các thuật toán học máy trong việc nhận dạng hình ảnh.

Tiểu luận này sẽ trình bày quá trình xây dựng một hệ thống nhận dạng chữ số viết tay sử dụng bộ dữ liệu MNIST. Hệ thống sẽ áp dụng các kỹ thuật học máy, đặc biệt là mô hình SVM (Support Vector Machine), nhằm đạt được hiệu suất dự đoán cao nhất. Cũng như nhiều ứng dụng học máy khác, hệ thống này không chỉ giúp cải thiện khả năng tự động hóa mà còn mở ra tiềm năng ứng dụng trong các lĩnh vực như nhận dạng biển số xe, chữ viết tay trong hóa đơn, và nhiều ứng dụng thực tế khác.

Bằng cách triển khai các phương pháp học máy hiện đại, tiểu luận này sẽ phân tích các phương pháp tiền xử lý dữ liệu, xây dựng mô hình, huấn luyện và đánh giá kết quả nhằm đạt được hệ thống nhận dạng chữ số chính xác và hiệu quả.

# MỤC LỤC

[LỜI MỞ ĐẦU](#_Toc182512637)

[MỤC LỤC](#_Toc182512638)

[MỤC LỤC HÌNH ẢNH](#_Toc182512639)

[CHƯƠNG 1: TỔNG QUAN MACHINE LEARNING VÀ DATA MINING 1](#_Toc182512640)

[1.1.Machine Learning: 1](#_Toc182512641)

[1.1.1.Support Vector Machine(SVM) 3](#_Toc182512642)

[1.2.Data Mining 10](#_Toc182512643)

[1.3.Mục tiêu và phạm vi của tiểu luận 13](#_Toc182512644)

[CHƯƠNG 2: PHƯƠNG PHÁP VÀ CÔNG CỤ 15](#_Toc182512645)

[2.1.Phương pháp phổ biến trong Machine Learning 15](#_Toc182512646)

[2.2.Các công cụ hỗ trợ 17](#_Toc182512647)

[CHƯƠNG 3: THỰC HIỆN ĐỀ TÀI 21](#_Toc182512648)

[3.1.Giới thiệu đề tài 21](#_Toc182512649)

[3.2.Thực hiện đề tài 22](#_Toc182512650)

[3.2.1.Import thư viện 22](#_Toc182512651)

[3.2.2.Load dữ liệu 23](#_Toc182512652)

[3.2.3.Tiền xử lý dữ liệu 23](#_Toc182512653)

[3.2.4.Trực quan hóa dữ liệu 26](#_Toc182512654)

[3.2.5.Xây dựng mô hình SVM 27](#_Toc182512655)

[3.2.5.1.Hyperlane 27](#_Toc182512656)

[3.2.5.2.Loss Function 28](#_Toc182512657)

[3.2.5.3.Regularization 29](#_Toc182512658)

[3.2.5.4.Gradient decent 30](#_Toc182512659)

[3.2.6.Dự đoán 33](#_Toc182512660)

[3.2.7.Đánh giá mô hình 33](#_Toc182512661)

[3.2.8.Triển khai 35](#_Toc182512662)

[CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 36](#_Toc182512663)

[4.1.Kết luận 36](#_Toc182512664)

[4.2.Hướng phát triển 36](#_Toc182512665)

[TÀI LIỆU THAM KHẢO 38](#_Toc182512666)

# MỤC LỤC HÌNH ẢNH

[Hình 1.1: Điểm dữ liệu phân tách tuyến tính 4](#_Toc182512734)

[Hình 1.2: Siêu phẳng tách dữ liệu khỏi hai lớp 5](#_Toc182512735)

[Hình 1.3: Siêu phẳng cho dữ liệu có giá trị ngoại lệ 5](#_Toc182512736)

[Hình 1.4: Siêu phẳng tối ưu hóa phân loại 6](#_Toc182512737)

[Hình 1.5: Ánh xạ dữ liệu 1D sang 2D để có thể tách biệt hai lớp 7](#_Toc182512738)

[Hình 1.6: Công thức siêu phẳng 7](#_Toc182512739)

[Hình 1.7: Công thức ranh giới quyết định 8](#_Toc182512740)

[Hình 1.8: Công thức phân loại tuyến tính 8](#_Toc182512741)

[Hình 1.9: Công thức Hard margin 8](#_Toc182512742)

[Hình 1.10: Công thức Soft margin 8](#_Toc182512743)

[Hình 1.11: Công thức các loại kernel 10](#_Toc182512744)

[Hình 3.1 : Machine learning pipeline 22](#_Toc182512745)

[Hình 3.2: Hình ảnh các label 26](#_Toc182512746)

[Hình 3.3: Số lượng các label 27](#_Toc182512747)

[Hình 3.4: Công thức Hyperlane 27](#_Toc182512748)

[Hình 3.5: Công thức Loss Function 28](#_Toc182512749)

[Hình 3.6: Công thức Regularization 29](#_Toc182512750)

[Hình 3.7: Công thức Gradient Decent 30](#_Toc182512751)

[Hình 3.8: Update Gradient Descent 31](#_Toc182512752)

[Hình 3.9: Đánh giá mô hình bằng heatmap 34](#_Toc182512753)

[Hình 3.10: Đánh giá mô hình bằng f1-score 34](#_Toc182512754)

[Hình 3.11: Giao diện người dùng 35](#_Toc182512755)

# CHƯƠNG 1: TỔNG QUAN MACHINE LEARNING VÀ DATA MINING

## 1.1.Machine Learning:

**Supervised learning:** học có giám sát, là việc cho máy tính học trên dữ liệu đã được gán nhãn (label), hay nói cách khác, với mỗi đầu vào Xi, chúng ta sẽ có nhãn Yi tương ứng.

**Unsupervised learning:** học không giám sát, là cho máy tính học trên dữ liệu mà không được gán nhãn, các thuật toán machine learning sẽ tìm ra sự tương quan dữ liệu, mô hình hóa dữ liệu hay chính là làm cho máy tính có kiến thức, hiểu về dữ liệu, từ đó chúng có thể phân loại các dữ liệu về sau thành các nhóm, lớp (clustering) giống nhau mà chúng đã được học hoặc giảm số chiều dữ liệu (dimension reduction).

**Semi-supervised learning:** Học bán giám sát là một loại học máy nằm giữa học có giám sát và học không giám sát. Đây là phương pháp sử dụng một lượng nhỏ dữ liệu có nhãn và một lượng lớn dữ liệu không có nhãn để huấn luyện mô hình. Mục tiêu của học bán giám sát là xây dựng một hàm có khả năng dự đoán chính xác biến đầu ra dựa trên các biến đầu vào, tương tự như học có giám sát. Tuy nhiên, khác với học có giám sát, thuật toán này được huấn luyện trên một tập dữ liệu chứa cả dữ liệu có nhãn và không có nhãn.

**Reinforcement Learning (RL):** Học tăng cường, là một phương pháp học máy mà trong đó một tác nhân (agent) học cách tối ưu hóa hành vi của mình thông qua việc tương tác với môi trường. Thay vì có sẵn nhãn cho mỗi tình huống, tác nhân nhận được phần thưởng (reward) hoặc hình phạt (penalty) dựa trên hành động của mình, và học cách tối ưu hóa các quyết định để tối đa hóa tổng phần thưởng theo thời gian.

**Một số thuật toán phổ biến:**

Hồi quy tuyến tính là một kỹ thuật phân tích dữ liệu dự đoán giá trị của dữ liệu không xác định bằng cách sử dụng một giá trị dữ liệu liên quan và đã biết khác. Nó mô hình toán học biến không xác định hoặc phụ thuộc và biến đã biết hoặc độc lập như một phương trình tuyến tính. Ví dụ, giả sử rằng bạn có dữ liệu về chi phí và thu nhập của bạn trong năm ngoái. Kỹ thuật hồi quy tuyến tính phân tích dữ liệu này và xác định rằng chi phí của bạn là một nửa thu nhập của bạn. Sau đó, họ tính toán một chi phí trong tương lai không rõ bằng cách giảm một nửa thu nhập được biết đến trong tương lai.

**Một số ví dụ về bài toán hồi quy:**

* Dự đoán giá của một căn nhà rộng x m2 , có y phòng ngủ và cách trung tâm thành phố z km.
* Dự đoán nhiệt độ trung bình ngày tiếp theo dựa trên dữ liệu lịch sử thời tiết như nhiệt độ, độ ẩm, tốc độ gió, và lượng mưa.
* Dự đoán doanh thu bán hàng của một cửa hàng hoặc sản phẩm dựa trên các yếu tố như chi phí quảng cáo, mùa vụ, khuyến mãi, và xu hướng tiêu dùng.
* Dự đoán điểm số của học sinh trong kỳ thi dựa trên các yếu tố như giờ học, thói quen học tập, điểm số các bài kiểm tra trước, và sự tham gia các hoạt động ngoại khóa.
* Dự đoán tuổi thọ hoặc thời gian sử dụng của một sản phẩm dựa trên các thông số kỹ thuật, điều kiện sử dụng, và dữ liệu lịch sử bảo trì.
* Dự đoán số lượng khách hàng ghé thăm một cửa hàng hoặc sử dụng một dịch vụ trong tương lai, dựa trên dữ liệu lịch sử, xu hướng mùa vụ, và các chiến dịch tiếp thị.

### 1.1.1.Support Vector Machine(SVM)

Support Vector Machine (SVM) là một thuật toán học máy có giám sát được sử dụng cho cả nhiệm vụ phân loại và hồi quy . Mặc dù có thể áp dụng cho các bài toán hồi quy, nhưng SVM phù hợp nhất cho các nhiệm vụ phân loại . Mục tiêu chính của thuật toán SVM là xác định siêu phẳng tối ưu trong không gian N chiều có thể phân tách hiệu quả các điểm dữ liệu thành các lớp khác nhau trong không gian đặc điểm. Thuật toán đảm bảo rằng biên độ giữa các điểm gần nhất của các lớp khác nhau, được gọi là vectơ hỗ trợ , được tối đa hóa.

SVM đặc biệt hiệu quả vì chúng tập trung vào việc tìm siêu phẳng phân cách tối đa giữa các lớp khác nhau trong đối tượng mục tiêu, khiến chúng mạnh mẽ cho cả phân loại nhị phân và đa lớp . Trong phần phác thảo này, chúng ta sẽ khám phá thuật toán Support Vector Machine (SVM), các ứng dụng của nó và cách nó xử lý hiệu quả cả phân loại tuyến tính và phi tuyến tính, cũng như các tác vụ hồi quy và phát hiện ngoại lệ. SVM hoạt động bằng cách: Tìm kiếm một siêu phẳng phân chia dữ liệu thành hai phần khác nhau dựa trên nhãn lớp.

Kích thước của siêu phẳng phụ thuộc vào số lượng các đặc điểm. Ví dụ, nếu có hai đặc điểm đầu vào, siêu phẳng chỉ là một đường thẳng, và nếu có ba đặc điểm đầu vào, siêu phẳng trở thành mặt phẳng 2 chiều. Khi số lượng các đặc điểm tăng lên vượt quá ba, độ phức tạp của việc trực quan hóa siêu phẳng cũng tăng lên.

Đảm bảo khoảng cách giữa siêu phẳng và các điểm dữ liệu gần nhất (gọi là "support vectors") là lớn nhất có thể, giúp mô hình có tính tổng quát cao.

Xét hai biến độc lập x1 và x2 và một biến phụ thuộc được biểu diễn bằng vòng tròn màu xanh hoặc vòng tròn màu đỏ.

* Trong trường hợp này, siêu phẳng là một đường thẳng vì chúng ta đang làm việc với hai đặc điểm (x1 và x2).
* Có nhiều đường thẳng (hoặc siêu phẳng ) có thể phân tách các điểm dữ liệu.
* Thách thức là xác định siêu phẳng tốt nhất có thể tối đa hóa khoảng cách giữa các vòng tròn màu đỏ và màu xanh.



Hình 1.1: Điểm dữ liệu phân tách tuyến tính

Từ hình trên, rất rõ ràng là có nhiều đường (siêu phẳng của chúng ta ở đây là một đường vì chúng ta chỉ xem xét hai đặc điểm đầu vào x1 , x2) phân tách các điểm dữ liệu của chúng ta hoặc thực hiện phân loại giữa các vòng tròn màu đỏ và màu xanh lam.

Một lựa chọn hợp lý cho siêu phẳng tốt nhất trong Support Vector Machine (SVM) là lựa chọn tối đa hóa biên độ phân tách giữa hai lớp. Siêu phẳng có biên độ tối đa , còn được gọi là biên độ cứng , được chọn dựa trên việc tối đa hóa khoảng cách giữa siêu phẳng và điểm dữ liệu gần nhất ở mỗi bên.



Hình 1.2: Siêu phẳng tách dữ liệu khỏi hai lớp

Vì vậy, chúng ta chọn siêu phẳng có khoảng cách từ nó đến điểm dữ liệu gần nhất ở mỗi bên là tối đa. Nếu một siêu phẳng (hyperplane) như vậy tồn tại, nó được gọi là siêu phẳng biên độ tối đa (maximum-margin hyperplane)/biên độ cứng (hard margin). Vì vậy, từ hình trên, chúng ta chọn L2.



Hình 1.3: Siêu phẳng cho dữ liệu có giá trị ngoại lệ

Ở đây chúng ta có một quả bóng màu xanh trong ranh giới của quả bóng màu đỏ. Rất đơn giản! Quả bóng màu xanh trong ranh giới của quả bóng màu đỏ là một ngoại lệ của các quả bóng màu xanh. Thuật toán SVM có đặc điểm là bỏ qua ngoại lệ và tìm ra siêu phẳng tốt nhất để tối đa hóa biên độ. SVM mạnh mẽ với các ngoại lệ.



Hình 1.4: Siêu phẳng tối ưu hóa phân loại

Vì vậy, trong loại điểm dữ liệu này, SVM thực hiện tìm biên độ tối đa như đã thực hiện với các tập dữ liệu trước đó cùng với việc thêm một hình phạt mỗi khi một điểm vượt qua biên độ. Vì vậy, các biên độ trong các loại trường hợp này được gọi là biên độ mềm(soft margins). Khi có biên độ mềm đối với tập dữ liệu, SVM cố gắng giảm thiểu (1/margin+∧(∑penalty)). Mất bản lề là một hình phạt thường được sử dụng. Nếu không có vi phạm thì không có mất bản lề. Nếu vi phạm thì mất bản lề tỷ lệ thuận với khoảng cách vi phạm.

Giả sử, dữ liệu của chúng ta được hiển thị trong hình trên. SVM giải quyết vấn đề này bằng cách tạo một biến mới sử dụng kernel . Chúng ta gọi một điểm xi trên đường thẳng và chúng ta tạo một biến mới yi như một hàm của khoảng cách từ gốc o. vì vậy nếu chúng ta vẽ biểu đồ này, chúng ta sẽ có được thứ gì đó như được hiển thị bên dưới.



Hình 1.5: Ánh xạ dữ liệu 1D sang 2D để có thể tách biệt hai lớp

**Các đặc điểm chính của SVM**

**Hyperplane(Siêu phẳng):** Siêu phẳng là ranh giới quyết định được sử dụng để phân tách các điểm dữ liệu của các lớp khác nhau trong không gian đặc điểm. Đối với phân loại tuyến tính , đây là phương trình tuyến tính được biểu diễn dưới dạng wx+b=0.

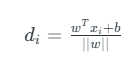
Phương trình cho siêu phẳng tuyến tính có thể được viết như sau:



Hình 1.6: Công thức siêu phẳng

Vectơ W biểu diễn vectơ pháp tuyến của siêu phẳng. tức là hướng vuông góc với siêu phẳng. Tham số b trong phương trình biểu diễn độ lệch hoặc khoảng cách của siêu phẳng từ gốc dọc theo vectơ pháp tuyến w.

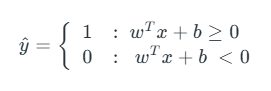
Khoảng cách giữa điểm dữ liệu x\_i và ranh giới quyết định có thể được tính như sau:



Hình 1.7: Công thức ranh giới quyết định

trong đó ||w|| biểu diễn chuẩn Euclid của vectơ trọng số w. Chuẩn Euclid của vectơ pháp tuyến W

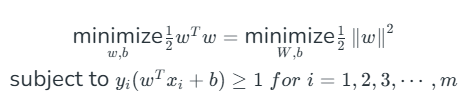
Đối với bộ phân loại SVM tuyến tính:



Hình 1.8: Công thức phân loại tuyến tính

**Optimization(Tối ưu hóa):**

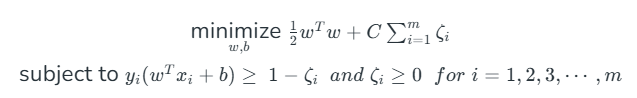
* **For Hard margin linear SVM classifier:**

****

Hình 1.9: Công thức Hard margin

Biến mục tiêu hoặc nhãn cho trường hợp đào tạo thứ i được biểu thị bằng ký hiệu ti trong câu lệnh này. Và ti = -1 cho các trường hợp âm (khi yi = 0) và ti = 1trường hợp dương (khi yi = 1) tương ứng. Bởi vì chúng ta yêu cầu ranh giới quyết định thỏa mãn ràng buộc:   𝑡 𝑖 ( 𝑤 𝑇 𝑥 𝑖 + 𝑏 ) ≥ 1.

* **For Soft margin linear SVM classifier:**



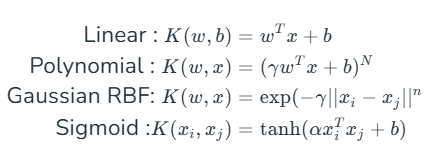
Hình 1.10: Công thức Soft margin

Dựa trên bản chất của ranh giới quyết định, Máy hỗ trợ vectơ (SVM) có thể được chia thành hai phần chính:

* Linear SVM: Linear SVM sử dụng ranh giới quyết định tuyến tính để phân tách các điểm dữ liệu của các lớp khác nhau. Khi dữ liệu có thể được phân tách tuyến tính chính xác, linear SVM rất phù hợp. Điều này có nghĩa là một đường thẳng duy nhất (trong 2D) hoặc một siêu phẳng (trong các chiều cao hơn) có thể chia hoàn toàn các điểm dữ liệu thành các lớp tương ứng của chúng. Một siêu phẳng tối đa hóa biên độ giữa các lớp là ranh giới quyết định.
* SVM phi tuyến tính: SVM phi tuyến tính có thể được sử dụng để phân loại dữ liệu khi không thể tách thành hai lớp theo đường thẳng (trong trường hợp 2D). Bằng cách sử dụng các hàm hạt nhân, SVM phi tuyến tính có thể xử lý dữ liệu có thể tách phi tuyến tính. Dữ liệu đầu vào ban đầu được các hàm hạt nhân này chuyển đổi thành không gian đặc trưng có chiều cao hơn, trong đó các điểm dữ liệu có thể được tách tuyến tính. SVM tuyến tính được sử dụng để định vị ranh giới quyết định phi tuyến tính trong không gian đã sửa đổi này.

**Các hàm kernel phổ biến trong SVM**

Nhân SVM là một hàm lấy không gian đầu vào có chiều thấp và biến đổi nó thành không gian có chiều cao hơn, tức là nó chuyển đổi các vấn đề không thể tách rời thành các vấn đề có thể tách rời. Nó chủ yếu hữu ích trong các vấn đề tách biệt phi tuyến tính. Nói một cách đơn giản, nhân thực hiện một số phép biến đổi dữ liệu cực kỳ phức tạp và sau đó tìm ra quy trình để tách dữ liệu dựa trên các nhãn hoặc đầu ra được xác định.



Hình 1.11: Công thức các loại kernel

## 1.2.Data Mining

Data mining – khai phá dữ liệu là quá trình phân loại, sắp xếp các tập hợp dữ liệu lớn để xác định các mẫu và thiết lập các mối liên hệ nhằm giải quyết các vấn đề nhờ phân tích dữ liệu. Các MCU khai phá dữ liệu cho phép các doanh nghiệp có thể dự đoán được xu hướng tương lai.

Quá trình khai phá dữ liệu là một quá trình phức tạp bao gồm kho dữ liệu chuyên sâu cũng như các công nghệ tính toán. Hơn nữa, Data Mining không chỉ giới hạn trong việc trích xuất dữ liệu mà còn được sử dụng để chuyển đổi, làm sạch, tích hợp dữ liệu và phân tích mẫu.

Tiền xử lý dữ liệu là một bước quan trọng trong Data mining. Giống như việc chúng ta phải sơ chế các nguyên liệu như rửa sạch, gọt vỏ, loại bỏ phần hư hại, thừa,.. trước khi đem chúng vào chế biến. Còn đối với dữ liệu, thông thường ban đầu dữ liệu sẽ ở dạng thô, lộn xộn, chứa dữ liệu sai hoặc không liên quan. Ngoài ra, ngay cả với dữ liệu có liên quan, thông tin đôi khi sẽ bị thiếu. Trong giai đoạn tiền xử lý, bạn xác định các thuộc tính không liên quan của dữ liệu và loại bỏ các thuộc tính đó. Đồng thời, việc xác định các điểm bất thường của tập dữ liệu và gắn cờ chúng là rất cần thiết. Ví dụ, lỗi của con người có thể dẫn đến vô tình hợp nhất hoặc phân tích không chính xác thông tin giữa các cột. Vì vậy dữ liệu phải được kiểm tra để đảm bảo tính toàn vẹn. Cuối cùng, bạn phải phát triển một phương pháp để xử lý dữ liệu bị thiếu và xác định xem dữ liệu bị thiếu ngẫu nhiên hay có hệ thống hay không.

Sau khi đã tiền xử lý dữ liệu, bước tiếp theo là xác định định dạng thích hợp để lưu trữ dữ liệu. Một lưu ý trong Data mining là chúng ta cần giảm số lượng thuộc tính cần thiết xuống mức tối thiểu mà không làm mất đi những thông tin cần thiết để giải thích các hiện tượng. Tức là chúng ta sẽ chuyển đổi dữ liệu thông qua các thuật toán giảm dữ liệu, chẳng hạn như Principal Component Analysis (mọi người có thể research thêm). Ngoài ra, các biến có thể cần phải được chuyển đổi để giúp giải thích hiện tượng đang được nghiên cứu.

Ví dụ, thu nhập của một cá nhân có thể được ghi lại trong bộ dữ liệu dưới dạng nhiều nguồn như:

+ Thu nhập tiền lương.

+ Thu nhập từ tài sản cho thuê.

+ Hỗ trợ các khoản thanh toán từ chính phủ.

+ Thu nhập khác,.... Tổng hợp thu nhập từ tất cả các nguồn này chúng ta sẽ gom thành một thuộc tính với chỉ số đại diện cho tổng thu nhập cá nhân.

Thông thường bạn cần phải chuyển đổi các biến từ loại này sang loại khác. Có thể thận trọng khi chuyển đổi biến liên tục cho thu nhập thành một biến phân loại trong đó mỗi bản ghi trong cơ sở dữ liệu được xác định là cá nhân có thu nhập thấp, trung bình và cao. Điều này có thể giúp nắm bắt các phi tuyến tính trong các hành vi cơ bản.

Dữ liệu đã được chuyển đổi phải được lưu trữ ở định dạng dễ dàng cho việc Data mining. Dữ liệu phải được lưu trữ ở định dạng cung cấp các quyền đọc và ghi ngay lập tức và không hạn chế cho các Data Scientist. Trong quá trình Data mining, các biến mới được tạo ra, được ghi lại vào cơ sở dữ liệu ban đầu, đó là lý do tại sao sơ đồ lưu trữ dữ liệu sẽ tạo điều kiện đọc và ghi hiệu quả vào cơ sở dữ liệu. Nó cũng quan trọng để lưu trữ dữ liệu trên các máy chủ hoặc phương tiện lưu trữ giữ cho dữ liệu an toàn và cũng ngăn chặn thuật toán Data mining không cần thiết tìm khi kiếm các mảnh dữ liệu nằm rải rác trên các máy chủ hoặc phương tiện lưu trữ khác nhau. An toàn và quyền riêng tư, bảo mậtr dữ liệu nên là mối quan tâm hàng đầu khi lưu trữ dữ liệu.

Sau khi dữ liệu được xử lý, chuyển đổi và lưu trữ một cách thích hợp, nó sẽ được mining. Bước này bao gồm các phương pháp phân tích dữ liệu, bao gồm các phương pháp tham số và không tham số, và các thuật toán Machine Learning. Chúng ta nên bắt đầu giai đoạn này bằng việc trực quan hóa dữ liệu, nó giúp chúng ta có góc nhìn đa chiều, xu hướng ấn,... về dữ liệu nhờ vào việc tận dụng khả năng vẽ đồ thị hiện đại của các phần mềm Data mining.

Sau khi kết quả đã được trích xuất từ Data mining, chúng ta cần thực hiện đánh kết quả. Đánh giá có thể bao gồm kiểm tra khả năng dự đoán của các mô hình trên dữ liệu quan sát được để xem các thuật toán đã hiệu quả chưa và hiệu quả như thế nào trong việc tái tạo dữ liệu. Điều này được gọi là "dự báo trong mẫu". Ngoài ra, kết quả được chia sẻ với các bên liên quan (stake holder) để phản hồi, sau đó được kết hợp trong các lần lặp lại sau này của Data mining để tiếp tục cải thiện quy trình. Data mining và đánh giá kết quả trở thành một quá trình lặp đi lặp lại để các Analyst, Data Scientist sử dụng các thuật toán tốt hơn và cải thiện chất lượng kết quả được tạo ra theo phản hồi nhận được từ các bên liên quan.

Các kỹ thuật khai phá dữ liệu phổ biến:

Phân cụm là kỹ thuật chia dữ liệu thành các nhóm (cụm) sao cho các đối tượng trong cùng cụm có tính chất tương đồng hơn so với các cụm khác.

Phân lớp là kỹ thuật dự đoán nhãn (label) của đối tượng dựa trên một tập dữ liệu đã biết trước nhãn, thường áp dụng cho các bài toán có yếu tố dự đoán.

Khai phá mẫu kết hợp là kỹ thuật tìm kiếm mối quan hệ hoặc luật kết hợp giữa các mục trong tập dữ liệu lớn.

## 1.3.Mục tiêu và phạm vi của tiểu luận

Vấn đề Nhận diện chữ số viết tay là một vấn đề phổ biến trong lĩnh vực Nhận diện Ký tự Quang học (OCR), và nó có nhiều ứng dụng thực tiễn, chẳng hạn như số hóa tài liệu, tự động hóa nhập dữ liệu, hoặc xử lý các mẫu viết tay, như hóa đơn hoặc khảo sát. Trong dự án này, chúng tôi sẽ phát triển một hệ thống có thể nhận diện chữ số viết tay từ hình ảnh hoặc chữ số được viết trên màn hình bằng cách sử dụng công nghệ học máy. Hệ thống này có thể được mở rộng để thực hiện một số phép tính cơ bản dựa trên các chữ số đã được nhận diện.

Mục tiêu của dự án này là:

* Xây dựng một hệ thống có thể nhận diện chính xác các chữ số viết tay (0–9) từ hình ảnh đầu vào.
* Chuyển đổi đầu vào viết tay thành định dạng có thể đọc được bởi máy và thực hiện các phép tính cơ bản trên chúng như: cộng, trừ, nhân và chia.
* Thu thập dữ liệu: Trong dự án này, bạn sử dụng tập dữ liệu MNIST, bao gồm 60.000 hình ảnh huấn luyện và 10.000 hình ảnh kiểm tra của các chữ số viết tay. Mỗi hình ảnh có kích thước 28x28 pixel và được gán nhãn với chữ số đúng.
* Tiền xử lý hình ảnh: Chuẩn hóa dữ liệu hình ảnh (ví dụ: chuyển đổi giá trị pixel sang thang từ 0 đến 1), thay đổi kích thước tất cả hình ảnh về cùng một kích thước, và áp dụng giảm nhiễu để cải thiện độ chính xác nhận diện.
* Phát triển mô hình: Chúng tôi sẽ nghiên cứu cách xây dựng và huấn luyện một mô hình sử dụng kỹ thuật tiên tiến phân tách các lớp chữ số khác nhau trong học máy, đó là thuật toán Support Vector Machine (SVM).
* Đánh giá: Kiểm tra hiệu suất của mô hình bằng cách sử dụng tập dữ liệu kiểm tra và đánh giá độ chính xác của nó bằng các chỉ số như độ chính xác, độ tinh khiết, độ hồi tưởng và điểm F1.
* Ứng dụng: Triển khai một giao diện người dùng nơi người dùng có thể tải lên hoặc vẽ chữ số trên màn hình, và hệ thống sẽ trả về chữ số dự đoán dựa trên mô hình đã được huấn luyện. Hệ thống của bạn cũng nên có khả năng thực hiện các phép tính cơ bản.

# CHƯƠNG 2: PHƯƠNG PHÁP VÀ CÔNG CỤ

## 2.1.Phương pháp phổ biến trong Machine Learning

Xử lý trước dữ liệu là bước đầu tiên quan trọng trong việc phân tích dữ liệu. Nó cho phép bạn chuyển đổi dữ liệu thô thành một định dạng dễ hiểu và có thể sử dụng để phân tích. Nó có một quy trình toàn diện để đảm bảo dữ liệu được mồi và sẵn sàng cho các giai đoạn khám phá, mô hình hóa và giải thích tiếp theo.

[Làm sạch dữ liệu](https://www.astera.com/type/blog/data-cleansing/) là một tập hợp con của tiền xử lý, chủ yếu liên quan đến việc xác định và sửa lỗi và sự không nhất quán trong tập dữ liệu. Mặt khác, tiền xử lý dữ liệu là một thuật ngữ bao gồm làm sạch dữ liệu và các quy trình khác như chuẩn hóa, chuyển đổi và trích xuất tính năng, rất cần thiết để chuẩn bị dữ liệu để phân tích.

Trích xuất đặc trưng: Trích xuất tính năng đề cập đến quá trình xác định và chọn các đặc điểm hoặc thuộc tính quan trọng, phù hợp và mang tính thông tin nhất từ một tập dữ liệu nhất định có thể hỗ trợ phân tích dữ liệu hoặc dự đoán dựa trên AI và Machine Learning chính xác và hiệu quả. Về bản chất, mục tiêu của việc trích xuất đặc trưng là chuyển đổi dữ liệu có chiều cao ban đầu thành dạng có chiều thấp hơn, bảo toàn thông tin mong muốn trong khi loại bỏ nhiễu, dư thừa và thông tin không liên quan. Kỹ thuật này cho phép cải thiện hiệu quả tính toán, giảm yêu cầu lưu trữ và có khả năng nâng cao hiệu suất mô hình.

Tầm quan trọng của việc trích xuất tính năng trong bối cảnh AI và Machine Learning chủ yếu bắt nguồn từ cái gọi là lời nguyền về chiều, đề cập đến độ khó ngày càng tăng của việc áp dụng thuật toán học tập và rút ra những hiểu biết có ý nghĩa khi số lượng thứ nguyên (hoặc tính năng) trong tập dữ liệu tăng lên. Bằng cách trích xuất các tính năng quan trọng từ dữ liệu, thuật toán có thể hoạt động hiệu quả hơn trong việc đưa ra dự đoán hoặc hiểu dữ liệu.

Có hai cách tiếp cận chính để trích xuất đặc trưng: phương pháp không giám sát và phương pháp giám sát

Chia dữ liệu thành tập huấn luyện và kiểm tra giúp xác minh khả năng dự đoán của mô hình khi áp dụng lên dữ liệu mới. Các phương pháp phổ biến bao gồm:

Chia ngẫu nhiên: Dữ liệu thường được chia thành tập huấn luyện và kiểm tra theo tỷ lệ như 70/30 hoặc 80/20. Tập huấn luyện dùng để mô hình học, còn tập kiểm tra để đánh giá mô hình. Đối với dữ liệu ít, có thể chia theo tỷ lệ nhỏ hơn để có đủ dữ liệu cho quá trình kiểm tra.

Kỹ thuật K-Fold Cross Validation: Khi dữ liệu ít hoặc cần kiểm tra độ ổn định của mô hình, kỹ thuật này chia dữ liệu thành K phần (folds). Mỗi lần lặp, mô hình sẽ huấn luyện trên K-1 phần và kiểm tra trên phần còn lại, kết quả được tính trung bình qua các lần lặp. Stratified K-Fold là biến thể thường được dùng trong bài toán phân loại để đảm bảo mỗi phần có tỷ lệ các lớp gần như bằng nhau.

Sau khi huấn luyện, đánh giá mô hình là bước quan trọng để kiểm tra độ chính xác và khả năng dự đoán của mô hình trên dữ liệu kiểm tra. Các phương pháp đánh giá bao gồm:

Sử dụng các chỉ số đánh giá: Các chỉ số phổ biến cho bài toán phân loại bao gồm:

* Accuracy (Độ chính xác): Tỷ lệ mẫu dự đoán đúng trên tổng số mẫu.
* Precision (Độ chính xác của dự đoán đúng): Tỷ lệ mẫu đúng trên tổng số mẫu dự đoán là đúng cho một lớp.
* Recall (Độ nhạy): Tỷ lệ dự đoán đúng trên tổng số mẫu thực sự thuộc lớp đó.
* F1-Score: Trung bình điều hòa của Precision và Recall, phù hợp khi cần cân bằng giữa Precision và Recall.

Đối với bài toán hồi quy, các chỉ số phổ biến là:

* MAE (Mean Absolute Error): Trung bình giá trị tuyệt đối của sai số giữa dự đoán và giá trị thực tế.
* MSE (Mean Squared Error): Trung bình bình phương sai số giữa dự đoán và giá trị thực tế.
* RMSE (Root Mean Squared Error): Căn bậc hai của MSE, đánh giá độ lệch dự đoán với giá trị thực.
* Ma trận nhầm lẫn (Confusion Matrix): Đây là công cụ trực quan trong bài toán phân loại để theo dõi số lượng dự đoán đúng/sai ở từng lớp, từ đó dễ dàng tính các chỉ số đánh giá như Precision và Recall.

ROC-AUC (Receiver Operating Characteristic - Area Under Curve): Đối với bài toán phân loại nhị phân, ROC-AUC đánh giá hiệu suất mô hình dựa trên tỷ lệ phát hiện đúng (True Positive Rate) và tỷ lệ báo sai (False Positive Rate) ở các ngưỡng khác nhau. Mô hình có giá trị AUC càng gần 1 thì càng tốt.

## 2.2.Các công cụ hỗ trợ

Python là một ngôn ngữ lập trình được sử dụng rộng rãi trong các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học (ML). Các nhà phát triển sử dụng Python vì nó hiệu quả, dễ học và có thể chạy trên nhiều nền tảng khác nhau. Phần mềm Python được tải xuống miễn phí, tích hợp tốt với tất cả các loại hệ thống và tăng tốc độ phát triển.

Những lợi ích của Python bao gồm:

* Các nhà phát triển có thể dễ dàng đọc và hiểu một chương trình Python vì ngôn ngữ này có cú pháp cơ bản giống tiếng Anh.
* Python giúp cải thiện năng suất làm việc của các nhà phát triển vì so với những ngôn ngữ khác, họ có thể sử dụng ít dòng mã hơn để viết một chương trình Python.
* Python có một thư viện tiêu chuẩn lớn, chứa nhiều dòng mã có thể tái sử dụng cho hầu hết mọi tác vụ. Nhờ đó, các nhà phát triển sẽ không cần phải viết mã từ đầu.
* Các nhà phát triển có thể dễ dàng sử dụng Python với các ngôn ngữ lập trình phổ biến khác như Java, C và C++.
* Cộng đồng Python tích cực hoạt động bao gồm hàng triệu nhà phát triển nhiệt tình hỗ trợ trên toàn thế giới. Nếu gặp phải vấn đề, bạn sẽ có thể nhận được sự hỗ trợ nhanh chóng từ cộng đồng.
* Trên Internet có rất nhiều tài nguyên hữu ích nếu bạn muốn học Python. Ví dụ: bạn có thể dễ dàng tìm thấy video, chỉ dẫn, tài liệu và hướng dẫn dành cho nhà phát triển.
* Python có thể được sử dụng trên nhiều hệ điều hành máy tính khác nhau, chẳng hạn như Windows, macOS, Linux và Unix.

Các thư viện trong Python:

Scikit-Learn: là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering và dimensionality reduction.

Thư viện được cấp phép bản quyền chuẩn FreeBSD và chạy được trên nhiều nền tảng Linux. Scikit-learn được sử dụng như một tài liệu để học tập.

Để cài đặt scikit-learn trước tiên phải cài thư viện SciPy (Scientific Python). Những thành phần gồm:

* [Numpy](https://codelearn.io/sharing/tim-hieu-thu-vien-numpy-trong-python" \t "https://codelearn.io/sharing/_blank): Gói thư viện xử lý dãy số và ma trận nhiều chiều
* SciPy: Gói các hàm tính toán logic khoa học
* [Matplotlib](https://codelearn.io/sharing/ve-bieu-do-voi-thu-vien-matplotlib-p1" \t "https://codelearn.io/sharing/_blank): Biểu diễn dữ liệu dưới dạng đồ thị 2 chiều, 3 chiều
* IPython: Notebook dùng để tương tác trực quan với Python
* SymPy: Gói thư viện các kí tự toán học
* [Pandas](https://codelearn.io/sharing/xu-ly-du-lieu-voi-pandas-trong-python" \t "https://codelearn.io/sharing/_blank): Xử lý, phân tích dữ liệu dưới dạng bảng.

Với sự bùng nổ của lĩnh vực [Trí Tuệ Nhân Tạo – A.I.](https://topdev.vn/it-jobs/ai-kt1555) trong thập kỷ vừa qua, machine learning và deep learning rõ ràng cũng phát triển theo cùng. Và ở thời điểm hiện tại, [TensorFlow](https://www.tensorflow.org/) chính là thư viện mã nguồn mở cho machine learning nổi tiếng nhất thế giới, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn nhiều.

Các hàm được dựng sẵn trong thư viện cho từng bài toán cho phép TensorFlow xây dựng được nhiều neural network. Nó còn cho phép bạn tính toán song song trên nhiều máy tính khác nhau, thậm chí trên nhiều CPU, GPU trong cùng 1 máy hay tạo ra các dataflow graph – đồ thị luồng dữ liệu để dựng nên các model. Nếu bạn muốn chọn con đường sự nghiệp trong lĩnh vực A.I. này, nắm rõ những điều cơ bản của TensorFlow thực sự rất quan trọng.

Được viết bằng C++ và thao tác interface bằng Python nên phần performance của TensorFlow cực kỳ tốt. Đối tượng sử dụng nó cũng đa dạng không kém: từ các nhà nghiên cứu, nhà khoa học dữ liệu và dĩ nhiên không thể thiếu các lập trình viên.

Keras là gì hay Keras python là gì là câu hỏi được nhiều người quan tâm. Keras là một thư viện mã nguồn mở được sử dụng rộng rãi trong lĩnh vực deep learning (học sâu) và mạng nơ-ron. Nó được thiết kế để giúp các nhà phát triển xây dựng và thử nghiệm các mô hình học sâu một cách dễ dàng và linh hoạt.

Mục tiêu chính của Keras là cung cấp một API đơn giản, trực quan và dễ sử dụng, giúp người dùng tập trung vào việc xây dựng mô hình mà không cần lo lắng về chi tiết kỹ thuật phức tạp.

Một trong những điểm mạnh của Keras là khả năng tích hợp với các framework học sâu khác như TensorFlow và Theano, cho phép người dùng tận dụng sức mạnh của cả hai thế giới. Đây chính là điều giúp Keras trở thành một lựa chọn hàng đầu cho các dự án học sâu từ phân loại ảnh, dự đoán chuỗi thời gian, đến các ứng dụng trong lĩnh vực ngôn ngữ tự nhiên và nhiều lĩnh vực khác.

Weka (Môi trường phân tích kiến thức Waikato) là một phần mềm khai thác dữ liệu nguồn mở phổ biến được phát triển tại Đại học Waikato trong New Zealand. Nó được viết bằng Java và cung cấp một tập hợp các thuật toán học máy cho các tác vụ khai thác dữ liệu, bao gồm phân loại, hồi quy, phân cụm, khai thác quy tắc liên kết và lựa chọn tính năng.

Weka có giao diện người dùng đồ họa (GUI) giúp người dùng mới và người dùng có kinh nghiệm dễ sử dụng. Nó cũng hỗ trợ kịch bản và tích hợp với các ngôn ngữ lập trình khác, chẳng hạn như Python và R, thông qua nó Java API.

Một trong những thế mạnh của Weka là tính linh hoạt và khả năng mở rộng của nó. Nó cho phép người dùng thử nghiệm các kỹ thuật khai thác dữ liệu khác nhau một cách dễ dàng và xây dựng các thuật toán và mô hình tùy chỉnh. Nó cũng hỗ trợ nhập và xuất các định dạng dữ liệu khác nhau, bao gồm CSV, ARFF, và Excel.

Weka đã được sử dụng rộng rãi trong các môi trường học thuật và công nghiệp cho các ứng dụng và nghiên cứu khai thác dữ liệu. Nó có một cộng đồng người dùng lớn và vô số tài nguyên trực tuyến, bao gồm hướng dẫn, tài liệu và diễn đàn.

# CHƯƠNG 3: THỰC HIỆN ĐỀ TÀI

## 3.1.Giới thiệu đề tài

Nhận dạng chữ số viết tay là một vấn đề phổ biến trong lĩnh vực Nhận dạng ký tự quang học (OCR) và nó có nhiều ứng dụng thực tế, chẳng hạn như số hóa tài liệu, tự động nhập dữ liệu hoặc xử lý các biểu mẫu viết tay, chẳng hạn như hóa đơn hoặc khảo sát. Trong dự án này, chúng tôi sẽ phát triển một hệ thống có thể nhận dạng chữ số viết tay từ hình ảnh hoặc chữ số được viết trên màn hình bằng cách sử dụng công nghệ máy học. Hệ thống này có thể được mở rộng để thực hiện một số phép tính cơ bản dựa trên các chữ số được nhận dạng. Mục tiêu của dự án này là:

Xây dựng hệ thống có thể nhận dạng chính xác các chữ số viết tay (0–9) từ hình ảnh đầu vào.

Chuyển đổi dữ liệu đầu vào viết tay thành định dạng máy có thể đọc được và thực hiện các phép tính cơ bản trên đó như: cộng, trừ, nhân và chia.

**Một số nhiệm vụ chính:**

**Thu thập dữ liệu:** Trong dự án này, sử dụng bộ dữ liệu MNIST, bao gồm 60.000 hình ảnh đào tạo và 10.000 hình ảnh thử nghiệm có chữ số viết tay. Mỗi hình ảnh có kích thước 28x28 pixel và được gắn nhãn bằng chữ số chính xác.

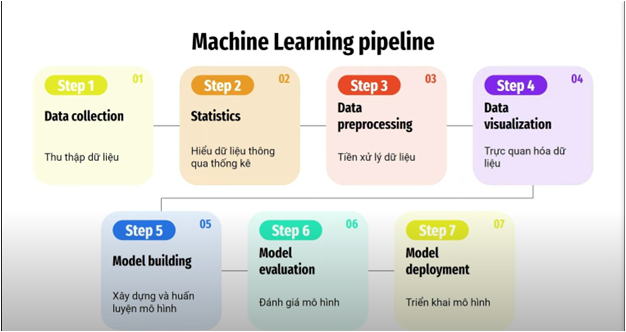
**Xử lý sơ bộ hình ảnh:** Chuẩn hóa dữ liệu hình ảnh (ví dụ: chuyển đổi giá trị pixel thành tỷ lệ từ 0 đến 1), thay đổi kích thước tất cả hình ảnh theo cùng kích thước và áp dụng giảm nhiễu để cải thiện độ chính xác của nhận dạng.

**Phát triển mô hình:** Chúng ta sẽ nghiên cứu cách xây dựng và huấn luyện mô hình bằng cách sử dụng một kỹ thuật nâng cao giúp phân tách các lớp chữ số khác nhau trong học máy, đó là thuật toán Máy vectơ hỗ trợ (SVM).

**Đánh giá:** Kiểm tra hiệu suất của mô hình bằng cách sử dụng tập dữ liệu thử nghiệm và đánh giá độ chính xác của nó bằng các số liệu như độ chính xác, độ chính xác, khả năng thu hồi và điểm F1.

**Ứng dụng:** Triển khai giao diện người dùng nơi người dùng có thể tải lên hoặc vẽ các chữ số trên màn hình và hệ thống sẽ trả về chữ số được dự đoán dựa trên mô hình được đào tạo. Hệ thống của bạn cũng có thể hoạt động với các phép tính cơ bản.

## 3.2.Thực hiện đề tài



Hình 3.1 : Machine learning pipeline

### 3.2.1.Import thư viện

|  |
| --- |
| 1. # Processing image 2. import numpy as np 3. import pandas as pd 4. # Data Visualization 5. import seaborn as sns 6. import matplotlib.pyplot as plt 7. # Data Processsing 8. from sklearn.decomposition import PCA 9. from sklearn.preprocessing import MinMaxScaler 10. from imblearn.over\_sampling import SMOTE 11. from sklearn.model\_selection import train\_test\_split |

Numpy và pandas là các thư viện cơ bản dùng để thao tác với dữ liệu. numpy cung cấp các công cụ để làm việc với mảng (arrays), trong khi pandas hỗ trợ làm việc với dữ liệu dạng bảng (DataFrame), dễ dàng cho việc thao tác và xử lý dữ liệu.

### 3.2.2.Load dữ liệu

|  |
| --- |
| 1. # Dataset 2. from tensorflow.keras.datasets import mnist 3. (X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data() 4. X\_train.shape, y\_train.shape, X\_test.shape, y\_test.shape |

from tensorflow.keras.datasets import mnist: Nhập dữ liệu MNIST từ Keras.

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data(): Tải dữ liệu và chia thành tập huấn luyện và tập kiểm tra:

* X\_train và y\_train: Ảnh huấn luyện và nhãn của chúng.
* X\_test và y\_test: Ảnh kiểm tra và nhãn của chúng.

X\_train.shape, y\_train.shape, X\_test.shape, y\_test.shape: Hiển thị kích thước của từng tập dữ liệu. Kích thước dự kiến:

* X\_train.shape: (60000, 28, 28) – 60,000 ảnh huấn luyện, mỗi ảnh có kích thước 28x28 pixel.
* y\_train.shape: (60000,) – 60,000 nhãn cho các ảnh huấn luyện.
* X\_test.shape: (10000, 28, 28) – 10,000 ảnh kiểm tra.
* y\_test.shape: (10000,) – 10,000 nhãn cho các ảnh kiểm tra.

### 3.2.3.Tiền xử lý dữ liệu

|  |
| --- |
| 1. # Image Preprocessing: Normalize the data (scale to 0-1 range) 2. X\_train = X\_train.astype('float32') / 255 3. X\_test = X\_test.astype('float32') / 255 |

Chuyển đổi kiểu dữ liệu của ảnh từ uint8 sang float32:

* X\_train.astype('float32'): Chuyển đổi X\_train từ kiểu dữ liệu nguyên không dấu uint8 (với giá trị pixel từ 0 đến 255) sang float32.
* Tương tự, X\_test.astype('float32'): Chuyển đổi X\_test sang kiểu float32.
* Việc chuyển đổi sang float32 giúp tăng độ chính xác trong quá trình tính toán, nhất là khi xử lý bằng các mô hình học máy hoặc học sâu.

Chuẩn hóa dữ liệu về phạm vi [0, 1]:

* X\_train.astype('float32') / 255 và X\_test.astype('float32') / 255: Sau khi chuyển đổi, giá trị của mỗi pixel được chia cho 255 để chuyển đổi các giá trị từ phạm vi [0, 255] về [0, 1].
* Chuẩn hóa dữ liệu về phạm vi [0, 1] giúp mô hình học máy hoặc học sâu hội tụ nhanh hơn và có độ ổn định cao hơn, do các giá trị không quá lớn.

|  |
| --- |
| 1. # Reshape 28x28 images to 1D vector of 784 2. X\_train = X\_train.reshape(-1, 784) 3. X\_test = X\_test.reshape(-1, 784) 4. X\_train.shape, X\_test.shape |

Thực hiện việc chuyển đổi kích thước của ảnh từ dạng ma trận 28×28 thành một vector 1 chiều gồm 784 phần tử (bởi vì 28×28=784). Việc chuyển đổi này giúp các mô hình học máy xử lý dữ liệu dễ dàng hơn khi đầu vào là một vector 1D thay vì

ma trận 2D.

|  |
| --- |
| 1. # Chage array NumPy to DataFrame 2. df\_X\_train = pd.DataFrame(X\_train.reshape(-1, 28 \* 28)) 3. df\_y\_train = pd.DataFrame(y\_train) 4. df\_X\_test = pd.DataFrame(X\_test.reshape(-1, 28 \* 28)) 5. df\_y\_test = pd.DataFrame(y\_test) 6. # Print DataFrame 7. df\_X\_train.head() 8. df\_y\_train.head() 9. # Rename column to label 10. df\_y\_train = df\_y\_train.rename(columns={0: 'label'}) 11. df\_digit = pd.concat([df\_X\_train, df\_y\_train], axis=1) 12. df\_digit.head() |

Chuyển đổi dữ liệu ảnh và nhãn từ NumPy sang pandas DataFrame, thực hiện một số thao tác tiền xử lý (như đổi tên cột), và kết hợp dữ liệu ảnh và nhãn để thuận tiện cho việc phân tích và huấn luyện mô hình.

|  |
| --- |
| 1. # Đổi tên các cột 2. new\_column\_names = ['pixel\_' + str(i) for i in range(784)] + ['label'] 3. df\_digit.columns = new\_column\_names 4. df\_symboils = pd.read\_csv('/content/drive/MyDrive/symbols\_dataset\_new.csv') 5. df\_symboils.columns = new\_column\_names 6. # Kết hợp hai bảng DataFrame train và df\_symboils theo chiều dọc (axis=0) 7. df\_concat = pd.concat([df\_digit, df\_symboils]) 8. # Đặt lại chỉ mục của DataFrame kết hợp 9. df\_concat = df\_concat.reset\_index(drop=True) 10. # In ra DataFrame kết hợp 11. df\_concat.tail() |

Tạo tên cột mới cho dữ liệu ảnh và nhãn, áp dụng chúng cho df\_digit và df\_symboils, sau đó hiển thị 5 hàng đầu của df\_symboils.

|  |
| --- |
| 1. # Giả sử df\_concat là DataFrame của bạn 2. label\_mapping = { 3. '+': 10, 4. '-': 11, 5. '\*': 12, 6. '/': 13 7. } 8. # Thay thế nhãn trong cột 'nhãn' bằng cách sử dụng ánh xạ 9. df\_concat['label'] = df\_concat['label'].replace(label\_mapping) |

Thay thế các ký hiệu trong cột 'label' của df\_concat bằng mã số tương ứng theo label\_mapping.

|  |
| --- |
| 1. # Tập dữ liệu Imblearn 2. smote = SMOTE(random\_state = 42) 3. X\_resampled, y\_resampled = smote.fit\_resample(X, y) 4. print(y.value\_counts()) 5. print(y\_resampled.value\_counts()) |

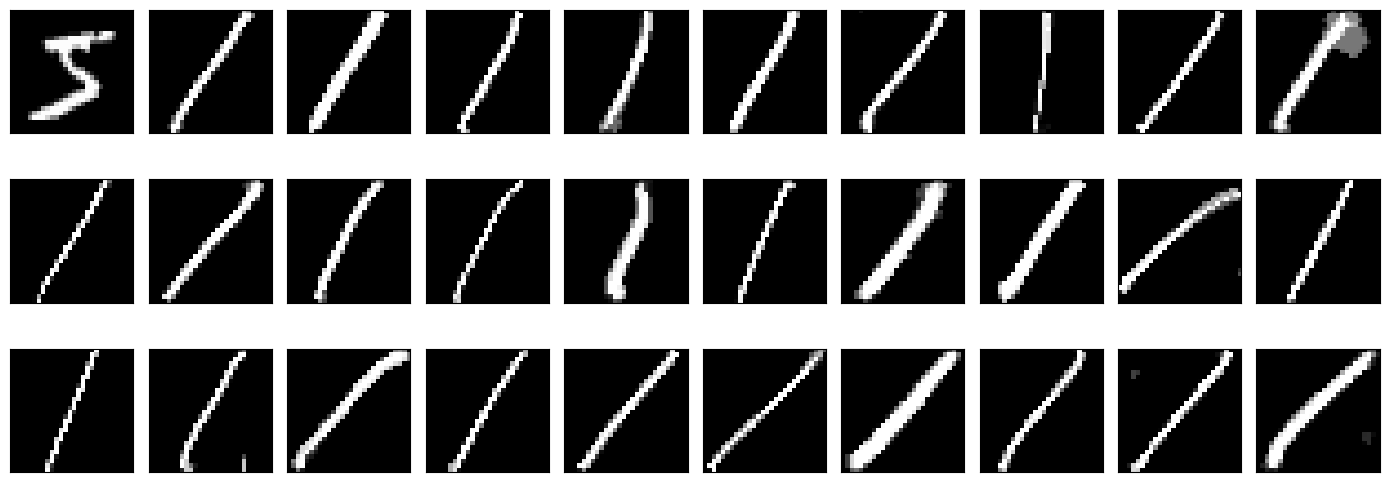
Sử dụng SMOTE để cân bằng lại dữ liệu bằng cách tạo mẫu mới cho lớp thiểu số trong y, sau đó in ra số lượng mẫu trước và sau khi cân bằng.

### 3.2.4.Trực quan hóa dữ liệu

|  |
| --- |
| 1. # Vẽ chữ số hình ảnh 2. plt.figure(figsize=(14, 12)) 3. for digit\_num in range(0, 30): 4. plt.subplot(7, 10, digit\_num + 1) 5. grid\_data = X\_resampled.iloc[-digit\_num].values.reshape(28, 28) 6. plt.imshow(grid\_data, interpolation='none', cmap= 'gray') 7. plt.xticks([]) 8. plt.yticks([]) 9. plt.tight\_layout() 10. plt.show() |

Hiển thị 30 hình ảnh chữ số từ tập X\_resampled dưới dạng lưới 7x10 với kích thước 28x28 pixel.

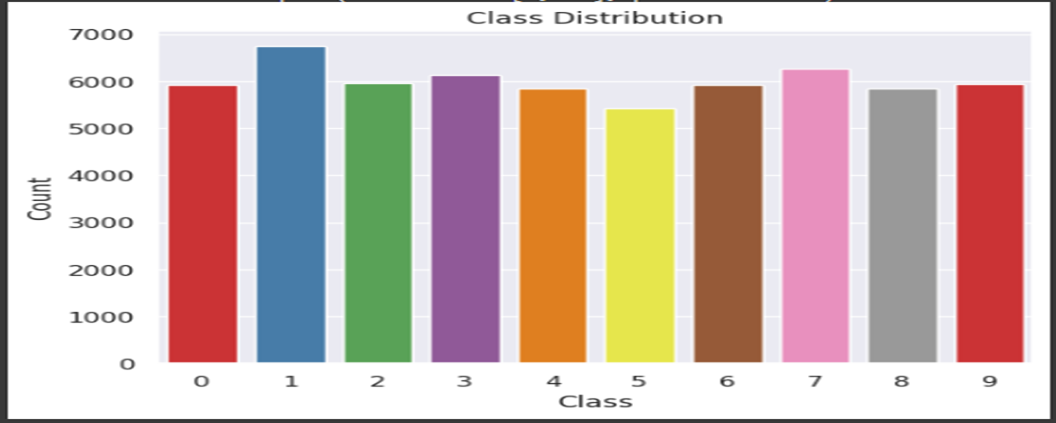
Output:



Hình 3.2: Hình ảnh các label

|  |
| --- |
| 1. # Khám phá sự phân bổ lớp học (gần như phân bổ đều) 2. sns.set(style="darkgrid") 3. # Sử dụng cú pháp đúng cho countplot 4. counts = sns.countplot(x= y\_resampled, palette='Set1') 5. # Thêm nhãn và tiêu đề 6. plt.xlabel('Class') 7. plt.ylabel('Count') 8. plt.title('Class Distribution') 9. plt.show() |

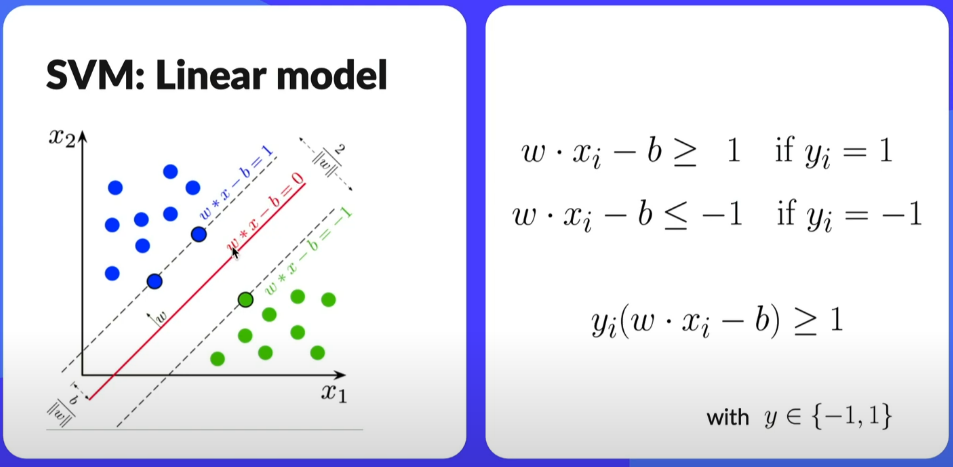
Vẽ biểu đồ phân bố lớp học của y\_resampled bằng countplot, thêm nhãn trục và tiêu đề, rồi hiển thị biểu đồ.



Hình 3.3: Số lượng các label

### 3.2.5.Xây dựng mô hình SVM

#### 3.2.5.1.Hyperlane



Hình 3.4: Công thức Hyperlane

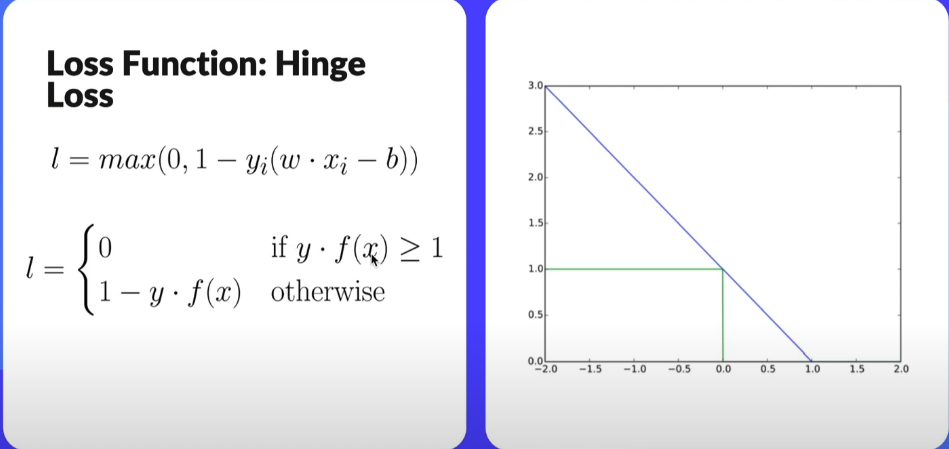
**Siêu phẳng quyết định (Decision Boundary)**:

* Siêu phẳng này là w⋅x − b = 0, được thể hiện bằng đường màu đỏ trên đồ thị. Đây là đường phân chia hai lớp dữ liệu sao cho khoảng cách giữa các điểm gần nhất thuộc mỗi lớp với đường này là lớn nhất.

**Biên (Margins)**:

* Hai đường đứt nét (đen) ở mỗi phía của siêu phẳng, là w⋅x – b = ± 1.
* Khoảng cách giữa hai đường biên này là **khoảng cách lớn nhất** mà mô hình SVM cố gắng tối đa hóa. Khoảng cách này cũng được gọi là "margin".

#### 3.2.5.2.Loss Function



Hình 3.5: Công thức Loss Function

Trong SVM (Support Vector Machine), hàm loss chủ yếu được sử dụng để đo lường mức độ mà mô hình phân loại chính xác các điểm dữ liệu, đặc biệt là các điểm gần biên quyết định (decision boundary). Loss function trong SVM thường có hai loại chính: Hinge Loss.

Hàm loss giúp mô hình SVM tối ưu hóa siêu phẳng để phân biệt dữ liệu thành các lớp một cách tốt nhất, bằng cách tối thiểu hóa loss tổng thể cho tất cả các điểm dữ liệu trong tập huấn luyện.

#### 3.2.5.3.Regularization

Trong SVM (Support Vector Machine), Regularization được sử dụng để điều chỉnh mô hình và tránh hiện tượng overfitting bằng cách cân bằng giữa độ phức tạp của mô hình và độ chính xác của phân loại trên tập huấn luyện.

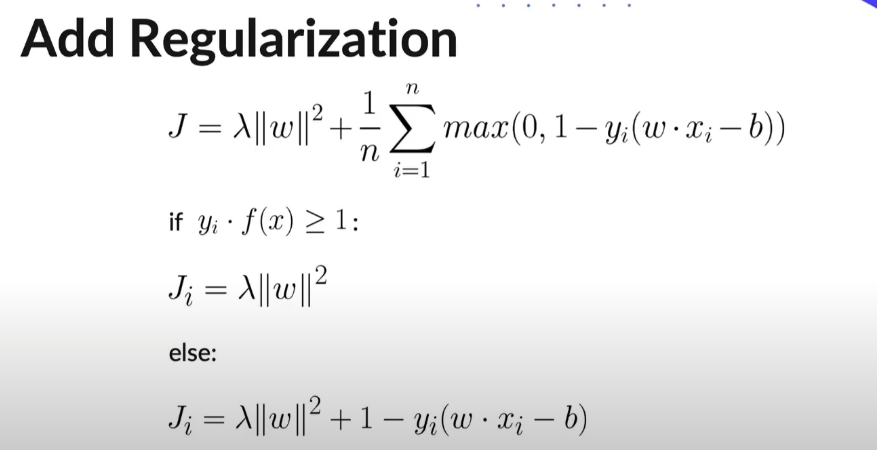
Regularization trong SVM được thực hiện thông qua một tham số **C** trong hàm mục tiêu, tham số này kiểm soát mức độ mà mô hình chịu phạt khi xảy ra lỗi trong phân loại dữ liệu. Dưới đây là các công thức và khái niệm liên quan đến Regularization trong SVM.

**Công thức hàm mục tiêu với Regularization**

Hàm mục tiêu trong SVM thường có hai phần chính:

* Phần thứ nhất là norm của vector trọng số w, nhằm tối thiểu hóa độ phức tạp của siêu phẳng (hyperplane).
* Phần thứ hai là tổng của các giá trị Hinge Loss trên các điểm dữ liệu bị phân loại sai hoặc gần biên.

Công thức hàm mục tiêu tổng quát cho SVM với Regularization là:



Hình 3.6: Công thức Regularization

Trong đó:

* C: là tham số điều chỉnh mức độ ưu tiên giữa Regularization và việc giảm lỗi phân loại.

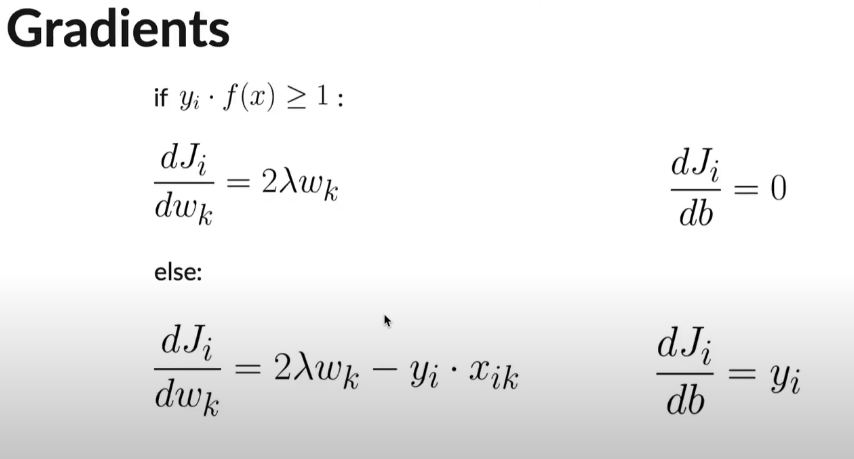
**Tham số Regularization C**

Tham số CCC quyết định mức độ nghiêm khắc mà mô hình xử lý các điểm dữ liệu bị phân loại sai:

* **Cao (C lớn)**: Mô hình sẽ cố gắng phân loại đúng các điểm dữ liệu trong tập huấn luyện và ít chấp nhận lỗi, điều này có thể dẫn đến **overfitting**.
* **Thấp (C nhỏ)**: Mô hình sẽ cho phép nhiều điểm dữ liệu bị phân loại sai hơn, ưu tiên việc tìm một siêu phẳng đơn giản, điều này có thể dẫn đến **underfitting**.

Tóm lại, Regularization trong SVM là phương pháp cân bằng giữa độ chính xác của phân loại và độ phức tạp của siêu phẳng thông qua việc tối ưu hóa một hàm mục tiêu có chứa tham số C. Việc chọn đúng giá trị của C rất quan trọng để đảm bảo mô hình có hiệu năng tốt trên dữ liệu mới, thay vì chỉ khớp tốt trên dữ liệu huấn luyện.

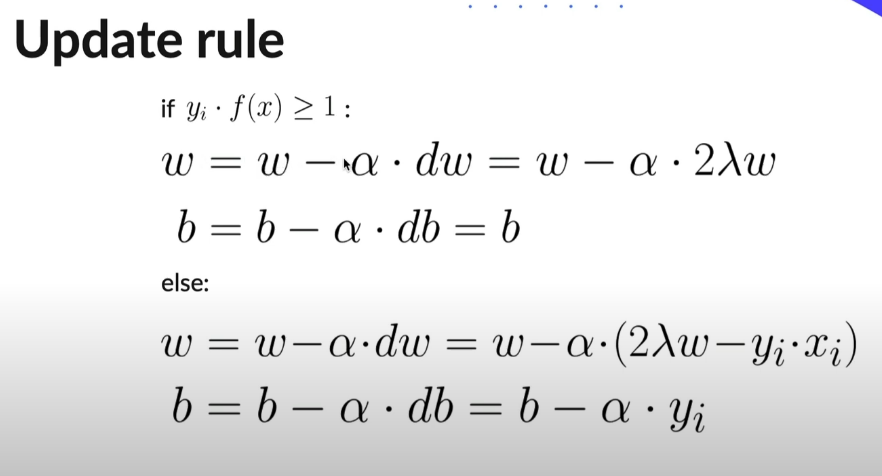
#### 3.2.5.4.Gradient decent



Hình 3.7: Công thức Gradient Decent

Gradient Descent là một phương pháp tối ưu hóa được sử dụng để tìm giá trị tối ưu của các tham số trong các mô hình học máy, bao gồm cả SVM (Support Vector Machine). Đối với SVM, Gradient Descent được sử dụng để tối thiểu hóa hàm mục tiêu bao gồm thành phần Hinge Loss và Regularization, giúp mô hình tìm ra siêu phẳng (hyperplane) tối ưu cho việc phân loại dữ liệu. Dưới đây là công thức và khái niệm khi áp dụng Gradient Descent trong SVM.

Với các Gradient vừa tìm được, ta cập nhật các tham số w và b trong mỗi vòng lặp của Gradient Descent theo công thức:



Hình 3.8: Update Gradient Descent

Trong đó:

* **η** là learning rate, xác định bước nhảy của mỗi lần cập nhật.

Gradient Descent giúp mô hình cập nhật dần các tham số w và b để giảm thiểu hàm mục tiêu, tối ưu hóa siêu phẳng phân loại dữ liệu tốt nhất.

Phương pháp này rất phù hợp với SVM khi áp dụng với các tập dữ liệu lớn, do có thể sử dụng Stochastic Gradient Descent (SGD), chỉ tính gradient cho một phần nhỏ dữ liệu trong mỗi lần cập nhật, giúp tối ưu hóa nhanh hơn và giảm bớt yêu cầu về bộ nhớ.

|  |
| --- |
| 1. class SVM: 2. def \_\_init\_\_(self, learning\_rate, no\_of\_iterations, lambda\_parameter): 3. ''' 4. learning\_rate: Tốc độ học 5. no\_of\_iterations: Số vòng lặp 6. lambda\_parameter: Tham số chính quy 7. model: Chứa các tham số weghts và bias 8. classes: Chứa các nhãn 9. binary\_y: Chứa các nhãn nhị phân (1, -1) 10. w: Chứa các weghts 11. b: Chứa các bias 12. ''' 13. self.learning\_rate = learning\_rate 14. self.no\_of\_iterations = no\_of\_iterations 15. self.lambda\_parameter = lambda\_parameter 16. self.models = [] 17. # Huấn luyện mô hình SVM đa lớp dựa trên nhãn trong y 18. def fit(self, X, y): 19. # Xác định các nhãn duy nhất từ y\_train 20. self.classes = np.unique(y) 21. self.models = [] 22. # Huấn luyện một SVM nhị phân cho mỗi lớp (one-vs-all) 23. for class\_label in self.classes: 24. # Tạo nhãn nhị phân cho one-vs-all: 1 cho nhãn hiện tại, -1 cho các nhãn khác 25. binary\_y = np.where(y == class\_label, 1, -1) 26. # Khởi weights và bias 27. w = np.zeros(X.shape[1]) 28. b = 0 29. # Huấn luyện mô hình SVM nhị phân cho lớp hiện tại 30. for \_ in range(self.no\_of\_iterations): 31. for index, x\_i in enumerate(X): 32. condition = binary\_y[index] \* (np.dot(x\_i, w) - b) >= 1 33. if condition:   dw = 2 \* self.lambda\_parameter \* w  db = 0   1. else:   dw = 2 \* self.lambda\_parameter \* w - x\_i \* binary\_y[index]  db = binary\_y[index]   1. # Cập nhật weights và bias 2. w -= self.learning\_rate \* dw 3. b -= self.learning\_rate \* db 4. # Lưu trọng số và bias cho mô hình của lớp hiện tại 5. self.models.append((w, b)) 6. # Dự đoán nhãn cho mỗi mẫu đầu vào 7. def predict(self, X): 8. scores = np.zeros((X.shape[0], len(self.classes))) 9. # Tính điểm số dự đoán cho mỗi lớp 10. for i, (w, b) in enumerate(self.models): 11. scores[:, i] = np.dot(X, w) - b 12. # Trả về nhãn có điểm số cao nhất 13. return self.classes[np.argmax(scores, axis=1)] |

### 3.2.6.Dự đoán

|  |
| --- |
| 1. # Khởi tạo và huấn luyện mô hình SVC 2. ''' 3. Các giá trị được điểm score cao nhất 4. learning\_rate(Tốc độ học): 0.001 5. no\_of\_iterations(số vòng lặp): 100 6. lambda\_parameter(tham số chính quy): 0.0001 7. ''' 8. model\_svm=SVM(learning\_rate=0.001,no\_of\_iterations=100, lambda\_parameter=0.0001) 9. # Phân loại 10 lớp (các chữ số 0-9) 10. model\_svm.fit(X\_train\_array, y\_train\_array) 11. # Dự đoán nhãn cho dữ liệu mới 12. y\_pred = model\_svm.predict(X\_test\_array) 13. print(y\_pred[:5])  # Trả về nhãn dự đoán cho mỗi mẫu |

Khởi tạo và huấn luyện mô hình SVM với tham số tối ưu, phân loại 10 lớp chữ số, dự đoán nhãn cho dữ liệu kiểm tra, và in 5 nhãn đầu tiên dự đoán được.

### 3.2.7.Đánh giá mô hình

|  |
| --- |
| 1. from sklearn.metrics import accuracy\_score 2. # Kiểm tra điểm score của y\_test(nhãn test) và y\_pred(nhãn dự đoán) 3. test\_accuracy = accuracy\_score(y\_test\_array, y\_pred) 4. print(f"Test Accuracy: {test\_accuracy\*100:.3f}%") |

Đoạn mã này tính và hiển thị độ chính xác của mô hình trên tập kiểm tra (y\_test\_array so với y\_pred) dưới dạng phần trăm.

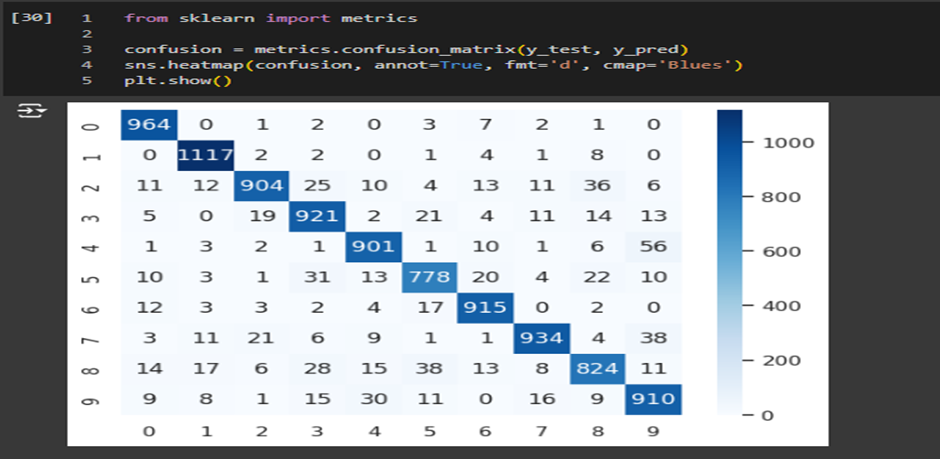
Output:



|  |
| --- |
| 1. from sklearn import metrics 2. confusion = metrics.confusion\_matrix(y\_test\_array, y\_pred) 3. sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues') 4. plt.show() |

Đoạn mã này tạo và hiển thị biểu đồ nhiệt của ma trận nhầm lẫn (confusion matrix) giữa y\_test\_array và y\_pred để trực quan hóa hiệu suất phân loại của mô hình.

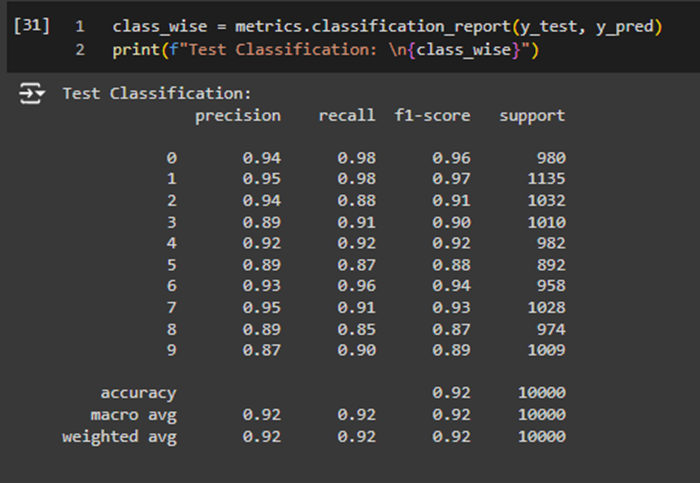
Output:



Hình 3.9: Đánh giá mô hình bằng heatmap

|  |
| --- |
| 1. class\_wise = metrics.classification\_report(y\_test\_array, y\_pred) 2. print(f"Test Classification: \n{class\_wise}") |

Đoạn mã này tính toán và in ra báo cáo phân loại (classification report) cho mô hình, bao gồm các chỉ số như precision, recall, f1-score và hỗ trợ cho từng lớp trong tập kiểm tra (y\_test\_array) và nhãn dự đoán (y\_pred).

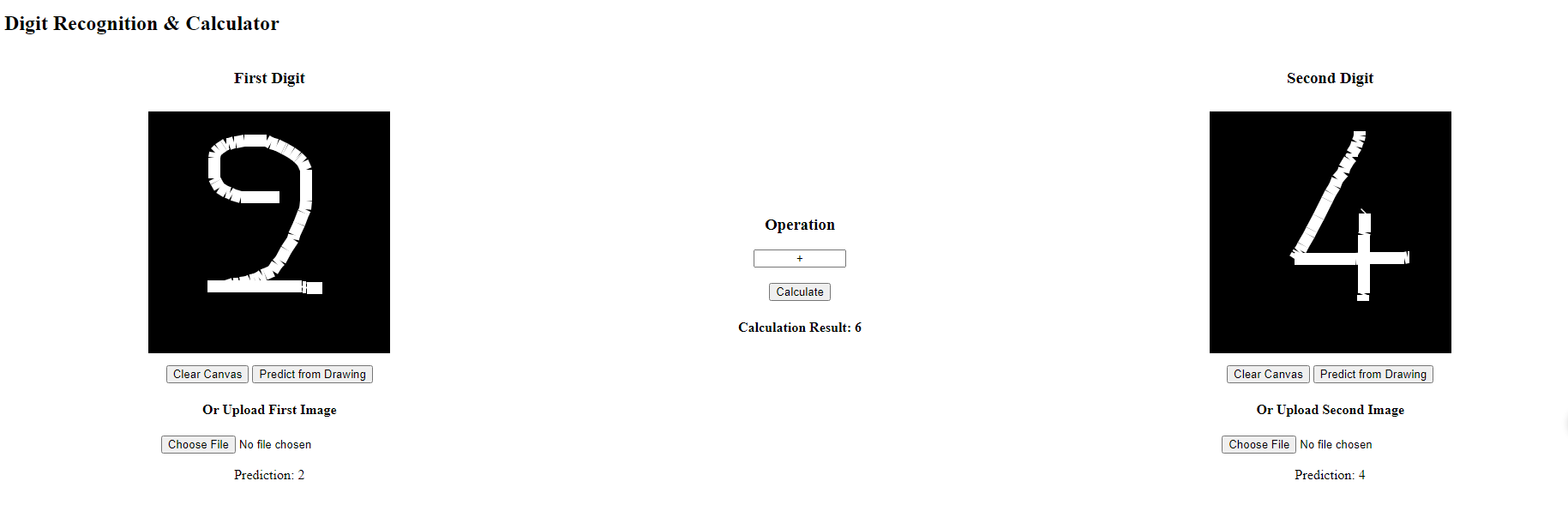


Hình 3.10: Đánh giá mô hình bằng f1-score

### 3.2.8.Triển khai

Để triển khai ứng dụng như yêu cầu của bạn, có thể sử dụng Flask làm framework cho back-end và HTML/JavaScript cho front-end. Mô hình SVM đã được huấn luyện sẽ được sử dụng để dự đoán các chữ số mà người dùng vẽ hoặc tải lên, và ứng dụng cũng có thể hỗ trợ các phép toán cơ bản.

Ứng dụng này sử dụng Flask để triển khai back-end và JavaScript/HTML cho front-end, cho phép người dùng vẽ hoặc tải lên chữ số và nhận dự đoán từ mô hình SVM. Hệ thống cũng có thể thực hiện các phép toán cơ bản nếu được mở rộng thêm tính năng xử lý phép toán.



Hình 3.11: Giao diện người dùng

# CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 4.1.Kết luận

Dự án xây dựng hệ thống nhận dạng chữ số viết tay sử dụng bộ dữ liệu MNIST đã thành công trong việc phát triển một mô hình học máy có khả năng nhận diện chính xác các chữ số từ 0 đến 9. Quá trình phát triển hệ thống đã bao gồm các bước quan trọng như thu thập dữ liệu, tiền xử lý hình ảnh, xây dựng và huấn luyện mô hình học máy, và đánh giá hiệu suất của mô hình thông qua các chỉ số như độ chính xác, độ phủ, độ nhạy và điểm F1. Mô hình SVM đã được chứng minh là hiệu quả trong việc phân loại các chữ số viết tay, mang lại kết quả nhận dạng chính xác cho hầu hết các mẫu thử nghiệm.

Bên cạnh đó, hệ thống cũng đã triển khai giao diện người dùng (UI) cho phép người dùng tải lên hoặc vẽ các chữ số trực tiếp trên màn hình. Hệ thống không chỉ có khả năng nhận dạng chữ số mà còn hỗ trợ thực hiện các phép toán cơ bản như cộng, trừ, nhân và chia, qua đó mở rộng ứng dụng của hệ thống vào các lĩnh vực thực tế như xử lý dữ liệu từ các mẫu giấy tờ viết tay, hóa đơn hoặc phiếu khảo sát.

## 4.2.Hướng phát triển

**Cải thiện độ chính xác của mô hình:** Mặc dù mô hình đã cho kết quả tốt, nhưng có thể cải thiện độ chính xác hơn nữa bằng cách áp dụng các kỹ thuật học sâu (Deep Learning) như Mạng Nơ-ron Tích chập (CNN), vốn thường cho kết quả tốt hơn trong các bài toán nhận diện hình ảnh. Việc sử dụng các phương pháp học sâu sẽ giúp hệ thống nhận diện chữ số viết tay phức tạp hơn với độ chính xác cao.

**Mở rộng ứng dụng:** Ngoài việc nhận dạng các chữ số, hệ thống có thể được mở rộng để nhận diện các ký tự chữ cái, phục vụ cho các ứng dụng trong lĩnh vực OCR (Nhận diện ký tự quang học). Điều này sẽ giúp hệ thống có thể được ứng dụng rộng rãi hơn, chẳng hạn như trong nhận dạng chữ viết tay của người dùng hoặc trong các công việc tự động hóa nhập liệu.

**Xử lý dữ liệu không chuẩn:** Một hướng phát triển quan trọng là mở rộng khả năng xử lý các hình ảnh viết tay không chuẩn, như chữ viết lệch, méo mó hoặc chữ viết có yếu tố nhiễu lớn. Cải thiện khả năng nhận diện cho những trường hợp này sẽ làm cho hệ thống trở nên mạnh mẽ hơn khi đối mặt với các tình huống thực tế.

**Cải thiện giao diện người dùng:** Giao diện người dùng có thể được phát triển thêm với các tính năng như nhận diện chữ số trong ảnh tải lên từ camera, hỗ trợ người dùng trong việc vẽ hoặc nhập chữ số một cách dễ dàng hơn. Ngoài ra, các phép toán phức tạp hơn như căn bậc hai, lũy thừa cũng có thể được tích hợp vào hệ thống.

**Triển khai hệ thống thực tế:** Đưa hệ thống vào sử dụng trong các ứng dụng thực tế như thanh toán điện tử, nhận dạng số trong phiếu khảo sát, hay các công việc nhập liệu tự động. Việc tích hợp hệ thống vào các nền tảng web hoặc di động sẽ làm cho hệ thống trở nên linh hoạt và tiện ích hơn đối với người dùng.

# TÀI LIỆU THAM KHẢO

[1] Zhou, Z. H. (2021). *Machine learning*. Springer nature.

[2] Alpaydin, E. (2021). *Machine learning*. MIT press.

[3] Mitchell, T. M., & Mitchell, T. M. (1997). *Machine learning* (Vol. 1, No. 9). New York: McGraw-hill.

[4] Mammone, A., Turchi, M., & Cristianini, N. (2009). Support vector machines. Wiley Interdisciplinary Reviews: Computational Statistics, 1(3), 283-289.

[5] Noble, W. S. (2006). What is a support vector machine?. Nature biotechnology, 24(12), 1565-1567.

[6] Cohen, G., Afshar, S., Tapson, J., & Van Schaik, A. (2017, May). EMNIST: Extending MNIST to handwritten letters. In 2017 international joint conference on neural networks (IJCNN) (pp. 2921-2926). IEEE.

[7] Baldominos, A., Saez, Y., & Isasi, P. (2019). A survey of handwritten character recognition with mnist and emnist. Applied Sciences, 9(15), 3169.